


## Enhancing web search with queries of equivalent intents

Ruihua Song<sup>1</sup>  · Dingquan Wang<sup>2</sup> · Jian-Yun Nie<sup>3</sup> ·  
Ji-Rong Wen<sup>4</sup> · Yong Yu<sup>5</sup>

Received: 6 January 2016 / Accepted: 26 August 2016 / Published online: 2 September 2016  
© Springer Science+Business Media New York 2016

**Abstract** Users often issue all kinds of queries to look for the same target due to the intrinsic ambiguity and flexibility of natural languages. Some previous work clusters queries based on co-clicks; however, the intents of queries in one cluster are not that similar but roughly related. It is desirable to conduct automatic mining of queries with equivalent intents from a large scale search logs. In this paper, we take account of similarities between query strings. There are two issues associated with such similarities: it is too costly to compare any pair of queries in large scale search logs, and two queries with a similar formulation, such as “SVN” (Apache Subversion) and support vector machine (SVM), are not necessarily similar in their intents. To address these issues, we propose using the similarities of query strings above the co-click based clustering results. Our method improves precision over the co-click based clustering method (lifting precision from 0.37 to 0.62), and outperforms a commercial search engine’s query alteration (lifting  $F_1$  measure from 0.42 to 0.56). As an application, we consider web document retrieval. We

---

✉ Ruihua Song  
rsong@microsoft.com

Dingquan Wang  
wdd@cs.jhu.edu

Jian-Yun Nie  
nie@iro.umontreal.ca

Ji-Rong Wen  
jirong.wen@gmail.com

Yong Yu  
yyu@cs.sjtu.edu.cn

<sup>1</sup> Microsoft Research Asia, Beijing 100080, China

<sup>2</sup> John Hopkins University, Baltimore, MD 21218, USA

<sup>3</sup> University of Montreal, Montreal 200240, Canada

<sup>4</sup> Renmin University, Beijing 100872, China

<sup>5</sup> Shanghai Jiao Tong University, Shanghai 200240, China

aggregate similar queries' click-throughs with the query's click-throughs 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.

**Keywords** Mining similar queries · Query intent · Web search

## 1 Introduction

In this paper we address the issue of automatically identifying queries with the same intent from large scale search logs. In traditional IR, a query is often regarded as a unique intent. However, we observe that users may use hundreds of different query strings to search the same targets. For example, we identify 561 unique queries to locate “environmental protection agency” in a set of 3-month Bing search logs and 102 unique queries to search for “pluto”. If each query is considered independent from other queries and used alone to search documents, many relevant documents will be missing. A better solution is to recognize groups of queries with the same search intents and then to aggregate the information of these queries to search documents. Compared to an individual query, such aggregation has an obvious advantage: it gives the system a higher chance to locate relevant documents because it benefits from all other queries with the same intent. However, this advantage can materialize only if the queries are correctly grouped, i.e. they genuinely correspond to the same search intent.

Many studies have investigated the problem of determining similar queries. For example, query alteration (Gao et al. 2010; Dang and Croft 2010) is developed to help users input correct words and search relevant results. The technology is widely applied by search engines online. As results will be shown to users, query alteration optimizes precision rather than recall to avoid incorrect suggestions or irrelevant search results. Clustering queries by co-clicks proves effective in improving context-aware query suggestion (Cao et al. 2008). The queries sharing co-clicks are somehow related but do not necessarily share the same intent. We ask human annotators to label queries with equivalent intents within 100 co-click based clusters. Table 1 shows an example. Although all the queries are in one cluster, the annotators further group them into three equivalent intents. For example, “charlie horse relief” is considered different from “cause charlie horses legs” in search intent. The previous studies on mining query intents (Hu et al. 2012; Sakai et al. 2003) or search result diversification (Agrawal et al. 2009) are also closely related. However, all

**Table 1** An example of manually labeled queries with the same intents

No.	Query (frequency)
1	Charlie horse relief (5), charlie horse remedies (5), how do you treat a charlie horse in your leg (5)
2	Cause charlie horses legs (15), charlie horse cramps (90), charlie horse cramp (5), charlie horse in foot (6), charlie horse in leg (25), charlie horse legs(10), charlie horses in the legs (7), charlie horse in legs (6), charlie horse in leg muscle (7), charlie horse leg cramps (165), leg cramps charlie horse (5), charlie horse muscle (5), charlie horse muscle cramps (12), charlie horse pain (11), charlie horses (17)
3	Severe leg cramps and stiffness (5)

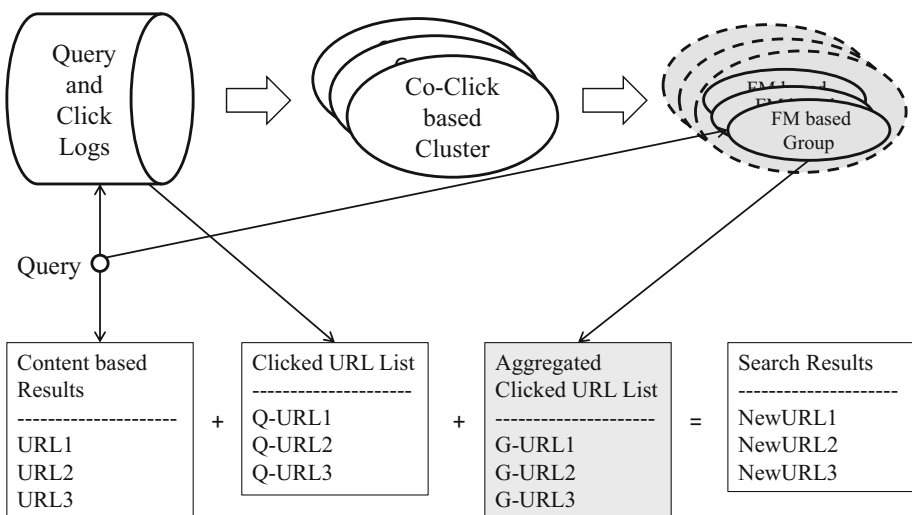
these studies aim to determine a similarity function between queries or a set of possible topics or aspects, which serve to determine a set of diversified results but do not have to correspond strictly to the same intents as identified by human annotators. To the best of our knowledge, no previous work has addressed specifically the problem we mentioned—to group queries with the same intent. This is the problem we address in this paper, which further extends problems investigated in previous studies.

From manually annotated queries, we observe that queries with similar intents often look similar to each other. Some have misspellings; some have abbreviations; some contain a few stop words; some contain the same keywords but in different order, and so on.

A possible naive approach to identify queries with identical intent is to compare a query’s string similarity. It is, however, not as 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 is available, we have no confidence to infer which queries have the same intent. For example, “droken” could be “draken” or “broken”.

In this study, we address this problem by exploiting both different types of information in two steps (See Fig. 1). First, we apply the method proposed in Cao et al. (2008) to cluster related queries based on co-clicks. This method has proven effective for grouping similar Web queries. Then, within each cluster, we propose grouping queries with similar intents based on fuzzy match of query strings. The use of string similarity within each cluster will limit the complexity of the operation.

An important question for this problem is evaluation. While previous studies often perform indirect evaluations (e.g. by evaluating the search results based on the query similarity proposed), we will perform an additional direct evaluation by comparing groups created by our method with the ground truth annotated by human annotators. Our approach achieves 0.56  $F_1$  measure. It outperforms the query clustering results based on co-clicks by 40 % and a commercial search engine’s query alteration by 30 % in terms of  $F_1$  measure.



**Fig. 1** An illustration of our proposed approach

Once clusters of queries are formed, they can be used for web search in different ways. In this paper, we focus on the use of click-through information of queries in the same cluster. We propose two strategies to aggregate the click-throughs of similar queries to a given query, namely, a replacing strategy and a smoothing strategy. The click-through information so generated is combined with the BM25F model (Robertson et al. 2004) used by Bing Search in document ranking (See Fig. 1). 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 less effective because it introduces more noise.

In the rest of the paper, we review related works in Sect. 2. Section 3 describes 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 conclude in Sect. 6.

## 2 Related work

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

Query reformulation/alteration try to change query terms for reducing the mismatch between queries and documents to achieve maximum higher relevance (Dang and Croft 2010). Some studies like (Cucerzan and Brill 2004; Li et al. 2006; Ahmad and Kondrak 2005; Gao et al. 2010; Brill and Moore 2000) focus on spelling correction. For example, Gao et al. (2010) build the n-gram language model in the source channel model (Och 2002). Some other studies extend the correction technique to reformulating a query, in which even correctly spelled words may be reformulated for better relevance in retrieval. Wang and Zhai (2008) look into a query log and mine term level relevance. Guo et al. (2008) incorporate the character, term and query level mismatches into a unified CRF model. Dang and Croft (2010) propose to enhance reformulation relevance by applying anchors. The aim of query reformulation is different from ours. They aim to find one or a few reformulations that can improve the current query, while our goal is to mine as many similar queries as possible that describe the same intents. In other words, they emphasize precision more than recall, while recall is important in our case. Experimental results in Sect. 5.3 will clearly show this difference. In addition, most approaches rely on global optimization to determine the best formulation while our approach also takes full advantage of local information within a co-click based cluster. This provides us richer information with which to judge whether two queries are similar. This aspect will be further discussed in Sect. 5.3.

Some previous studies investigate query clustering in different applications (Beeferman and Berger 2000; Wen et al. 2002). Wen et al. (2002) utilize cross references between queries and click documents to cluster queries, which expands a similarity measure based on query strings. Cao et al. (2008) propose a co-click based clustering algorithm based on Web search logs. Their goal is to solve the sparsity issue of session logs. By mapping queries into different concepts, they successfully suggest queries based on session data. There are also hybrid approaches combining different types of information (Sadikov et al. 2010; Tyler and Teevan 2010). Sadikov et al. (2010) propose a Markov model to combine document and session features in predicting user intents. While previous studies often

perform indirect evaluations, we will perform a direct evaluation by comparing the groups created by our method with the ground truth annotated by human annotators.

A lot of previous studies use click-through data for ranking Web documents. We can divide them into two groups. One group of approaches takes advantage of click-throughs as training data to learn ranking models (Joachims 2002; Chapelle and Zhang 2009; Zhu et al. 2010; Zhang et al. 2011; Guo et al. 2009). For example, Joachims (2002) introduce a novel retrieval model called Rank SVM trained on the relevance pairs of documents according to clicks. Zhang et al. (2011) 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 environment. Other group of approaches integrate click-through information into ranking functions or models. Xue et al. (2004) extract queries for each clicked document in search logs and embed them as metadata of the document. Their experimental results indicate that adding such a metadata significantly improves retrieval performance measured by Precision@20. Agichtein et al. (2006) incorporate user click-through data into implicit user feedback model for ranking and show significant improvement of retrieval than traditional content-based approaches. Our work falls in the second group, but our focus is to enrich click-throughs by mining similar queries, rather than integrate relevance features from click-throughs.

Sparsity of click-throughs is a well-known problem (Granka et al. 2004). As most queries have only a few clicks, the approaches based on click-throughs cannot impact many queries. Some ideas have been proposed to address the sparsity problem. Wu et al. (2011) 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 their baseline method. Song and He (2010) introduce 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. (2009) propose a smoothing approach to expand query-document features by predicting missing values of documents using click streams. In this paper, we also aim to enrich the click-through information of a query. Different from previous studies, we achieve this goal by explicitly identifying a group of queries with similar intents to the query. We believe that such an explicit grouping of queries according to intents provides a better way to leverage the appropriate queries to be used for enrichment. In addition, we also use fuzzy matching as an additional criterion to group queries. This additional means allows us to reduce possible query drifts, which is commonly observed on co-clicks (i.e. queries sharing co-clicks may often represent different search intents). Thus the click-throughs incorporated are expected to be less noisy.

### 3 Mining intents

In this section, we describe our proposed two-step approach to mine queries with the same intents. The first step measures query similarity based on co-clicks. We adopt the implementation of query clustering proposed by previous work (Cao et al. 2008). The second step further considers fuzzy match of query strings in measuring query similarity. We propose applying the second step within clusters that we obtain in the first step. In so doing, we can afford the complexity of the second step and ensure a level of quality at the same time.

### 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 not enough evidence to infer their intents.

Similar to Cao et al. (2008), 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 number of clicks. 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 Cao et al. (2008) 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 tests the diameter of  $C \cup \{q\}$ . If the diameter is not larger than  $D_{max}$ ,  $q$  is assigned to  $C$ , which is then updated. Otherwise, a new cluster is created for  $q$ .

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

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

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

The diameter measure is defined as

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

The diameter is ranged from 0 to  $\sqrt{2}$ . We set  $D_{max}$  as 1 in our experiments as suggested by Cao et al. (2008). More details of the clustering algorithm can be found in Cao et al. (2008).

**Table 2** An example query where the co-click based method performs worse than the fuzzy match method in terms of NDCG@10 when replacing strategy is applied

Query: alabama adventureland (3)			
Co-Clicked + Replacing		Fuzzy Match + Replacing	
Similar Queries: 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), ...		Similar Queries: alabama adventure land (12), alabama adventurland (1)	
No.	URL(Rating)	No.	URL(Rating)
1	alabamaadventure.com(Fair)	1	alabamaadventure.com(Fair)
2	themeparkcity.com(Fair)	2	adventurelandthemepark.com(Perfect)

Tables 2 and 3 show two example co-clicked based clusters on the left side. A cluster is usually related to one concept. For example, the cluster in Table 3 contains various queries on Michael Jackson and how/when/where he died. 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 related but different in topic, 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 it is difficult to apply it to large scale query clustering due to its complexity. We solve the efficiency problem by applying fuzzy match within clusters that we obtain by applying the co-clicks based method.

The use of fuzzy matching is motivated by our observation that misspelling often occurs in real users’ query logs, e.g., “great”, “graet” and “greet”. Previous research reveals that approximately 10–15 % of queries issued to search engines contain misspellings (Cucerzan and Brill 2004). 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., “quicktime” or “quick time”. As queries in a cluster have already had some common clicks, we are more confident that some of them are seeking the same targets if their expressions are similar enough.

Based on our observations, we propose the following procedures in our fuzzy match step.

First, we take five actions to transform an original query *q*. They are spelling correction, stop words removal, replacing abbreviation by full name, stemming and building a term vector (to ignore the order of key terms). Here we only describe how we correct spelling and discover abbreviations as the other three actions are straightforward.

**Table 3** An example query where the co-click based method performs better than the fuzzy match method in terms of NDCG@10 when smoothing strategy is applied

Query: michael jackson is dead (33)			
Co-Click + Smoothing		Fuzzy Match + Smoothing	
Similar Queries: michael jackson dies(100), how michael jackson died (94), when michael jackson died (62), where michael jackson died (12), mickal jackson picture (1), ...		Similar Queries: micheal jacksons dead (3),michael jacksons dead (2), michaels jacksons death (2), miichael jacksons death (2), michaels jackson s death (1), ...	
No.	URL(Rating)	No.	URL(Rating)
6	en.wikipedia...jackson(Excellent)	8	abcnews.go.com...(Good)
7	abcnews.go.com...(Good)	12	en.wikipedia...jackson (Excellent)

- We utilize the classical noise-channel model proposed by Kernighan et al. (1990) as our general spell correction framework. We also incorporate the string edit distance and machine translation model proposed by Gao et al. (2010) as the error model in our approach. Specifically, for each co-click based cluster, we divide its member queries into groups based on the edit distance between queries and regard the query with maximum frequency in each group as the correct spelling. Then we learn local models for the cluster. Lastly, we use the models to correct spellings.
- We detect abbreviation in two steps: (1) Context sensitive synonym detection. This approach is similar to the work proposed by Peng et al. (2007). Each phrase is represented by a context vector derived from the query in which it occurs. The phrase pairs with the same context vector are selected as synonym candidates for the next step. (2) Abbreviation detection by alignment. For each synonym candidate pair, the phrase of short length is considered the abbreviation candidate  $s$  and the longer element is considered the full name. That is, we consider  $l = \langle t_1, t_2, \dots \rangle$ , where  $t$  is the term, as a full name and  $s$  is the abbreviation of  $l$ , if and only if there exists an ordered alignment  $A$  from  $s$  to  $l$ . An abbreviation usually does not include stop words. For example, “National Aeronautics and Space Administration” is abbreviated to “NASA”. Thus we set the stop words removal action before the action of replacing abbreviation by full name.

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

For each node  $n, n' \in C$ , an edge is added 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 (Wagner and Fischer 1974) 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.  $\theta_{hard}$  is called the *hard distance threshold*, which limits the maximum character difference between two queries. This threshold mainly helps for short queries.  $\theta_{soft}$  is called the *soft distance threshold*, which is related to the length of the two queries. This threshold mainly helps for long queries because it can tolerate larger character differences.

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

Tables 2 and 3 show two example groups on the right side. Compared to the co-clicks based results, the similar queries identified by this step more likely stick to a narrow intent. For example, there are only two similar queries to “alabama adventureland”. There is a slight difference in spelling between them.

As building the query graph requires one to one query comparison, the time complexity is  $O(N^2)$ , where  $N$  is the number of queries in cluster  $C$ . As the size of clusters follows roughly Zipf distribution (we will show the distribution in Sect. 5.2.1), most clusters have a small number of queries. Furthermore, the maximum size of the 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 retrieval method

Previous work has demonstrated that click-throughs are effective in improving Web search (Joachims 2002; Xue et al. 2004; Agichtein et al. 2006). Instead of complex models, we apply simple ways to aggregate click-throughs and fuse information with a content-based ranking function. This is sufficient to show the impact of enhancing click-through data by the mined intents, which is our primary goal. More sophisticated models can be developed later to better take advantage of the query groups we create.

### 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 BM25F model (Robertson et al. 2004) as our ranking function based on content of Web documents. Robertson et al. extends the famous BM25 model (Robertson and Walker 1994) to multiple fields for structure documents. In Bing Search, each Web document contains four fields: anchor, title, body and URL. Weights for different fields are trained on large scale datasets. The BM25F model is one of the strongest features of Bing Search ranker. We retrieve the top 20 documents using the BM25F formula 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 Count method (1781) to combine the two lists:

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

where  $r_b$  is the rank of document  $d$  in the list of top 20 documents returned by the BM25F formula, and  $r_c$  is the rank of  $d$  in the list of clicked documents that are also among the former list.  $\lambda$  is a trade-off parameter to balance two lists.

### 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:

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

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* strategy. 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 will test 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.,  $\langle \text{query}, \text{query time}, \text{clicked URL} \rangle$ . Thus, by aggregating, we can obtain the tuple of  $\langle \text{query}, \text{clicked URL}, \text{\#clicks} \rangle$  for each query. We apply the algorithms described in Sect. 3 to these data and mine similar queries. Notice that the scale of logs is very large. No previous study has reported experiments on query logs of comparable size.

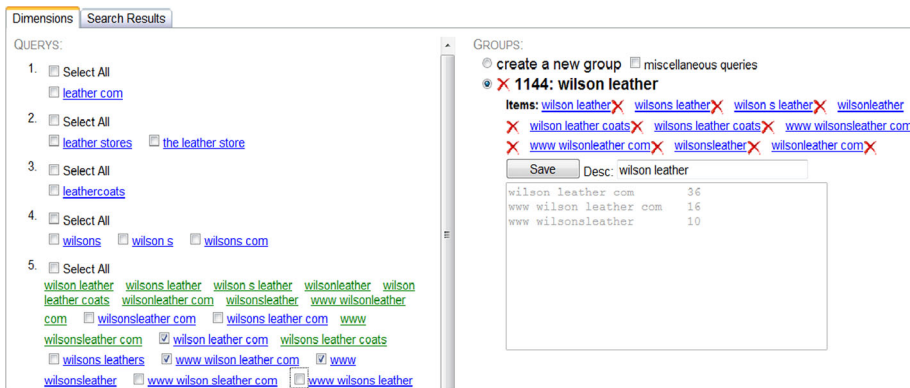
Although the co-click based query clustering algorithm (Cao et al. 2008) is well designed, we still cannot afford clustering all queries in the 3-month logs on one machine due to limitation of memory. We parallelize the process of clustering queries. First, we split the logs by week into 12 subsets. For each week, we build a click-through bipartite and run the algorithm and then cluster the queries in a week into level-1 clusters. Then, we extract one representative query from each level-1 cluster. We merge the click-through bipartite of representative queries from adjacent 2 weeks into level-2 bipartite and cluster the queries again into level-2 clusters. Now we have six independent sets of level-2 clusters. Next, we prune those containing only one query no matter in level-2 cluster or its corresponding level-1 cluster. Most likely they are too rare to have similar intent queries. Based on the pruned level-2 clusters, we extract representative queries again, merge adjacent two sets, and cluster the queries into level-3 clusters. Now we have three independent sets of level-3 clusters. Finally, we merge the representative queries extracted from the three sets and cluster queries into final level-4 clusters. We assign unique global identifiers to level-4 clusters and the pruned level-2 clusters. Any query inherits the same global identifier from its representative queries. Thus, we can get a map from each query to a global cluster identifier. Or by grouping the queries with the same global cluster identifiers, we can know which queries each final cluster contains.

#### 5.1.1 Evaluation of query clustering

To evaluate the proposed approach, we create a dataset of 100 sampled co-click based clusters. In each cluster the queries with the same intent are manually labeled by assessors.

We begin by randomly selecting 100 co-click based clusters from the Bing Search logs of January 2009. We discard the clusters that contain too many (e.g.,  $>120$ ) queries because it is difficult for assessors to handle. The number of queries in the selected clusters ranges from 2 to 120 with a median of 8. The average number of queries in a cluster is about 16.

Then, we design an easy-to-use tool (See Fig. 2) to assist assessors in merging the queries with the same intents. Queries in a cluster are shown in a list on the left side with a checkbox next to each of them. For example, in the figure, “leather com”, “leather stores”, “the leather store”, “leathercoats”, “wilsons” on the left part are queries for grouping. Assessors can refer to search results anytime if they are not sure about the



**Fig. 2** A screenshot of labeling tool of query clustering

meaning of a query. Every query has a link to its search results as shown in Fig. 2. An assessor can easily check the queries with almost the same intents. All checked queries are then shown as a merged group on the right side. The assessor is also asked to give a name that represents the group. For example, a group named “wilson leather” has been created on the right. The tool also provides an easy way to modify existing query groups. For example, most queries of the created group are about the official website for “wilson leather”, except for “wilson leather coats” and “wilsons leather coats”. Thus assessors can delete the two queries by clicking on the removing icon. They can also easily add some other queries, such as “wilson leather com”, “www.wilson leather com”, and “www wilsonleather”, to the wilson leather group by checking the queries on the left.

We hire three native English speakers as our assessors. One of them is a college student and the other two hold bachelor degrees. Their majors are diverse and include Journalism, Medicine and Surgery, and Hospitality Management. Their ages range from 23 to 32. Given a cluster, they are asked to split queries into intent groups. They label the 100 clusters independently.

To measure the inter-assessor agreement, we calculate Fleiss’ Kappa among the assessors. First, we compose pairs of any two queries in each co-click based cluster. There are 63,490 pairs for all 100 clusters. Given a query pair, if an assessor groups the two queries into one intent group, we regard the pair’s label as “yes”; whereas, if an assessor does not group them together, we regard the pair’s label as “no”. Thus we have two label categories as two columns. Next, we count the number of assessors who rate “yes” and the number of assessors who rate “no” as a row for each pair. Finally, we calculate the Fleiss Kappa over the matrix. Our Fleiss’ Kappa result is 0.3133 with a standard error of 0.0023. According to the interpretation in [https://en.wikipedia.org/wiki/fleiss%27\\_kappa](https://en.wikipedia.org/wiki/fleiss%27_kappa), it is a fair agreement. This indicates the divergence of understanding of intents and the difficulty of clustering queries.

We aggregate three assessors’ labels to obtain a ground-truth dataset. If two or more assessors agree on a label category, we assign the label as a ground-truth label for a query pair. Among 63,490 query pairs, 10,328 pairs have “yes” labels, which means they share the same intent.

Given a query set (in our case it is a co-click based query cluster), we denote the set of query pairs with the ground-truth label “yes” as *Y* and the set of query pairs with the ground-truth label “no” as *N*. When an algorithm clusters the queries, any two queries

from the same cluster compose a set of predicted query pairs with the same intent, denoted by  $P$ . Then, precision and recall are defined as:

$$\text{precision} = \frac{|Y \cap P|}{|P|} \quad (2)$$

$$\text{recall} = \frac{|Y \cap P|}{|Y|} \quad (3)$$

$F_1$  measure is the harmonic average of precision and recall. Please refer to <http://nlp.stanford.edu/ir-book/html/htmledition/evaluation-of-clustering-1.html> to learn more about clustering evaluation.

We calculate precision, recall and  $F_1$  for each query set and then average them as the mean precision, recall and  $F_1$ . These are usually called Macro averages. To compare two methods, we can also conduct a significant test of paired  $t$  test over 100 pairs of precision/recall/ $F_1$  values.

### 5.1.2 Evaluation of document retrieval

We use a query set that contains 13,315 queries for tuning parameters and another query set that contains 13,920 queries for evaluating retrieval effectiveness. On average, there are more than 100 documents 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 sets with our processed 12-week query logs, we obtain 2829 queries for training and 7453 queries for testing. We conduct ranking experiments on these queries and their corresponding relevance assessments of documents.

We apply widely used normalized discounted cumulated gain (NDCGs) (Järvelin and Kekäläinen 2002) as evaluation metrics of retrieval. The five relevance ratings have gains of  $2^{\text{rating}} - 1$ . 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 first glance, whereas NDCG@10 measures deeper performance, which corresponds to the first page of search results.

By tuning over the training dataset, we empirically set  $\lambda$  as 0.9 for all methods,  $\beta$  as 0.7 for the method combining smoothed clicks based on co-clicks with BM25F, and  $\beta$  as 0.4 for the method combining smoothed clicks based on fuzzy match.

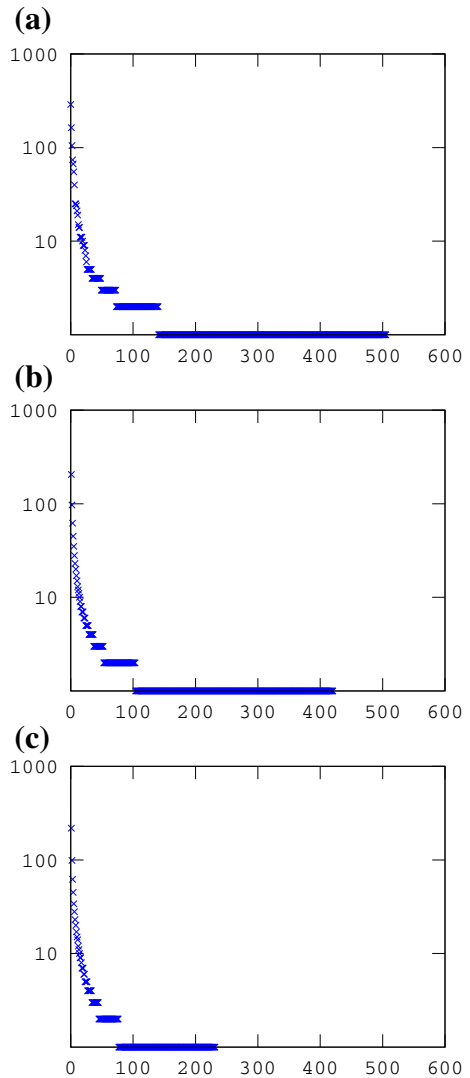
## 5.2 Statistics

### 5.2.1 On query clusters

We plot frequencies in descending order in Fig. 3. The x axis can stand for queries, fuzzy matched based intent groups, or co-click based clusters. Group frequency is defined as the sum of frequencies of all queries belonging to the group. Cluster frequency is defined as the sum of frequencies of all queries belonging to the cluster.

Previous studies have discovered that query frequency follows Zipf distribution (Fagni et al. 2006). As shown in Fig. 3a, a long tail corresponds to the huge amount of queries that have been issued only once during the 12 weeks. When we group queries with similar intents together, the distribution of group frequency also follows Zipf law (See Fig. 3b); however, more queries move close to the y axis with their groups. As a result, the long tail

**Fig. 3** Frequency distribution based on different granularity. **a** Queries, **b** FM based Groups, **c** CC based clusters



is not as long as that of queries. In our experiments, the number of groups is about 83 % of the number of unique queries. This indicates that the intents 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 Fig. 3c. 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

We do statistics of similar queries and click-throughs over the 7543 queries that are used for evaluating web retrieval effectiveness. We average the number of similar queries, the

**Table 4** Comparison of the number 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

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.

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 increases the number of clicked URLs and the number of summed clicks. The CC method increases the number of clicked URLs to 6.6 times and the number of clicks to 4.6 times. As expected, the FM method brings fewer click-throughs than the CC method, but the number of clicked URLs still increases by about two times and the number of clicks increases by more than three times.

### 5.3 Query clustering evaluation experiment

We evaluate the groups that our proposed approach generates (denoted as FM) and compare them with the clusters based on co-clicks (denoted as CC) and the clusters based on Bing's query alterations (denoted as Bing). Bing Search integrates some state-of-the-art query reformulation/alteration models, such as Gao et al. (2010), Guo et al. (2008), and tunes the models on a huge amount of data and resources. We obtain all alterations of queries in the labeled dataset from Bing Search. If two queries or their alterations are the same, we merge them into one cluster. Results are shown in Table 5.

As results show, our proposed FM method improves the CC method by 40 % in terms of  $F_1$  measure. It indicates that our proposed method is more effective in identifying the queries with the same intent. In particular, the FM method improves precision from 0.37 to 0.62 over the CC method. It is not surprising that the recall of FM method drops from 0.65 to 0.62 because we ask annotators to split queries in co-click based clusters into ground-truth intent groups. When compared with Bing's query alteration based clusters, we find that Bing outperforms our proposed groups in precision, i.e., 0.82 versus 0.62; whereas, our

**Table 5** Evaluating different methods over manually labeled data, we conduct a paired  $t$  test with a two tailed distribution to compare the FM method with the other two methods

Metrics	Precision	Recall	$F_1$ measure
CC	0.36	<b>0.65</b>	0.40
FM	0.62	0.62	<b>0.56</b>
Bing	<b>0.82</b>	0.37	0.43

The best result is given in bold

Results indicate that the difference between FM method and other two methods in all metrics is significant with  $p$  value less than 0.01

proposed approach performs better in recall, i.e., 0.62 versus 0.37. Overall, in terms of  $F_1$  measure, our proposed approach significantly outperforms Bing by about 30 %.

Our proposed approach does a better job of identifying the same intents for two main reasons. One is that query alteration is used online and thus it has to be cautious to ensure high precision. Its goal is not to find all possible reformulations of the user’s query, but to suggest a few highly related ones. It is therefore highly precision-oriented. The other is that we fully leverage the co-click based cluster in offline mode while query alteration has no such click information when a query is just submitted online. Given that users click the same URLs for two queries and these queries are expressed by similar strings, we have high confidence to infer that they have actually the same intent. For example, we can group “allmusic” with “all music” but query alteration cannot. Query alteration would more stick to users’ original spelling. Our approach relies also on co-clicks and can thus merge “all media guide” with “all music guide”, which alteration cannot. Indeed, both “music” and “media” are correct words, and query alteration will not reformulate any of them. In addition, our approach can recognize “amg” as an abbreviation of “all music guide” while query alteration cannot.

### 5.4 Web document retrieval experiment

We evaluate the ranking results retrieved using six different methods: the BM25F method (BM25F), the method combing query clicks with BM25F (Q + BM25F), the method combining aggregated clicks based on co-clicks with BM25F (CC + BM25F), the method combining aggregated clicks based on fuzzy match with BM25F (FM + BM25F), the method combining smoothed clicks based on co-clicks with BM25F [(Q,CC) + BM25F], and the method combining smoothed clicks based on fuzzy match [(Q,FM) + BM25F]. Results over the test dataset are shown in Table 6. The baseline method is Q + BM25F. All statistical difference is tested relative to it.

#### 5.4.1 Analysis on replacing strategy

The third and fourth methods replace a query’s clicks by the aggregated clicks mined from the CC method and the FM method. Although the CC method finds more related queries,

**Table 6** Comparing retrieval effectiveness of the methods that are using click-throughs of similar queries based on different mining algorithms

Methods	NDCG@1 (Imp.)	NDCG@3 (Imp.)	NDCG@10 (Imp.)
BM25F	54.16 (−15.86 % <sup>†</sup> )	51.05 (−9.55 % <sup>†</sup> )	51.22 (−5.77 % <sup>†</sup> )
Q + BM25F	64.37 (−)	56.45 (−)	54.35 (−)
CC + BM25F	63.69 (−1.06 % <sup>†</sup> )	56.12 (−0.58 % <sup>†</sup> )	54.30 (−0.09 %)
FM + BM25F	64.49 (0.18 %)	56.53 (0.16 % <sup>*</sup> )	54.43 (0.15 % <sup>†</sup> )
(Q,CC) + BM25F	64.36 (−0.01 %)	56.47 (0.04 %)	<b>54.45 (0.19 %<sup>†</sup>)</b>
(Q,FM) + BM25F	<b>64.57 (0.31 %<sup>†</sup>)</b>	<b>56.54 (0.16 %<sup>†</sup>)</b>	54.43 (0.15 % <sup>†</sup> )

The best result is given in bold

Imp. means percentage improvement. We apply  $t$  test in significant test. The baseline is Q + BM25F

<sup>†</sup> means the difference is statistically significant with  $p$  value  $\leq 0.01$

<sup>\*</sup> means the difference is statistically significant with  $p$  value  $\leq 0.05$

combining corresponding aggregated clicks with BM25F performs worse than the  $Q + \text{BM25F}$  method, our baseline method, in all NDCGs. The difference is significant in  $\text{NDCG}@1$  and  $\text{NDCG}@3$ . This indicates that the CC method may bring some clicks that hurt retrieval performance. The FM method is stricter in grouping similar intents, thus bringing less irrelevant click-throughs. As the table shows, combining the aggregated clicks with BM25F outperforms the baseline  $Q + \text{BM25F}$  method in all NDCGs. The difference is significant in  $\text{NDCG}@3$  and  $\text{NDCG}@10$ .

We conduct a case study of the query shown in Table 2. The number in parentheses behind a query is the number of summed clicks for the query. When looking closely into similar queries, we find that the CC method may cluster related queries, but some of them may have different intents from the given query. For instance, besides similar queries on adventure park in Alabama, the CC method also mines some other parks in Alabama, such as water parks, amusement parks 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](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 boosts a perfect URL <http://adventurelandthepark.com> to No. 2 search result. In such a case,  $\text{NDCG}@3$  increases when the FM method is applied while it decreases when the CC method is applied.

#### 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 BM25F. As Table 6 shows, the two methods perform better than the corresponding methods using only aggregated clicks of similar queries respectively. Compared to the baseline  $Q + \text{BM25F}$  method, the  $(Q, \text{CC}) + \text{BM25F}$  method is significantly better in terms of  $\text{NDCG}@10$ , although there is no significant improvement in terms of  $\text{NDCG}@1$  and  $\text{NDCG}@3$ . It indicates that the CC method may bring less relevant clicked URLs because of not-very-similar queries. Such URLs hurt top retrieval effectiveness but help the retrieval effectiveness of deeper cutoffs, in particular when we use the click-throughs to smooth query clicks.

The  $(Q, \text{CC}) + \text{BM25F}$  method is even better than the  $(Q, \text{FM}) + \text{BM25F}$  method in terms of  $\text{NDCG}@10$ , although the difference is not significant. We show such a case in Table 3. For instance, although the query “Michael Jackson is dead” is only clicked 33 times, similar queries that are mined by the CC method are much more popular and cover related subtopics like when/how/where Michael Jackson died. These related queries do not drift away from the query and contribute clicks to the authoritative URL about the death of Michael Jackson from Wikipedia. As a result, the  $(Q, \text{CC}) + \text{BM25F}$  method performs better than the  $(Q, \text{FM}) + \text{BM25F}$  method in  $\text{NDCG}@10$ .

The evaluation results in Table 6 indicate that the  $(Q, \text{FM}) + \text{BM25F}$  method significantly outperforms the baseline  $Q + \text{BM25F}$  method in all NDCGs. It is also the best among the six methods in terms of  $\text{NDCG}@1$  and  $\text{NDCG}@3$ . This indicates that the queries mined by the FM method are effective in improving retrieval performance. It is also necessary to use the click-throughs of similar queries to smooth query clicks, rather than replacing query clicks. For example, given a query “transformers 2”, the FM method can mine some similar queries like “transformer”. In fact, the movie Transformer 2 is the sequel to Transformer. Thus two homepages from IMDB 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 + BM25F method ranks the good URL at No. 2 while the FM + BM25F 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.

In addition, the small improvement of average measurements is caused by only a small portion of the large query set is influenced; however, it is meaningful for a commercial search engine because many users can still experience the significant difference when they submit queries who are affected by our proposed methods.

We conduct a case study to see why our approach (Q,FM) + BM25F can significantly improve the baseline method. 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 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 “greatest movie quotes”, the fuzzy match method can find more popular queries with similar search intents, such as “great movie quotes”. When leveraging click-throughs from these similar queries, more authoritative URLs pop up at the top. The fuzzy match method is helpful even for the most popular queries, such as “free quicktime download”. One reason may be that spam often follows the most popular queries. Aggregating click-throughs for different similar queries can recover some really wanted URLs.

**Table 7** Cases where the (Q,FM) + BM25F method performs better than the Q + BM25F method

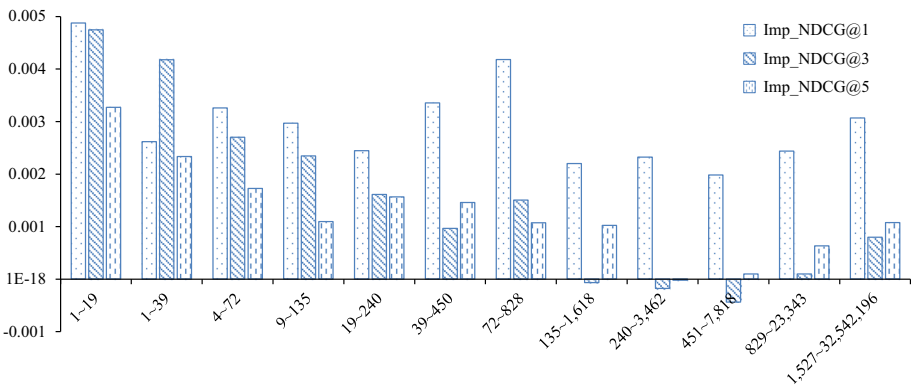
(Q,FM) + BM25F	Q + BM25F
Query: rice gardens (1)	
Similar queries: rice garden (99), rice gardenb (1)	
Top three URLs and ratings	
thericegarden.com(Perfect)	allrecipes.com/.../detail.aspx(Bad)
ricegarden.biz/locations.php(Fair)	gardenrice.com(Bad)
ricegarden.biz/menu.php(Fair)	gardensofricecreek.com(Bad)
Query: greatest movie quotes (302)	
Similar queries: great movie quotes (1530), great moive quotes (3), greates movie quotes (2), greastest movie quotes (2), graet movie quotes (1), gret movie quote (1), grrreat movie quotes (1), grwat movie quotes (1), great movie quoets (1), greata movie quotes (1), grrreat movie quote (1)	
Top three URLs and ratings	
filmsite.org/greatfilmquotes.html(Good)	en.wikipedia.org/...quotes(Fair)
en.wikipedia.org/...quotes(Fair)	filmsite.org/greatfilmquotes.html(Good)
franksreelreviews.com/...quote.htm(Good)	franksreelreviews.com/...quote.htm(Good)
Query: free quicktime download (4122)	
Similar queries: free quick time download (398), free quick time down load (24), freequicktimedownload (16), free quicktime dowload (14), free quicktime download (14), free quicktime downlode (8), fee quicktime download (6), free quictime downloads (6), ...	
Top three URLs and ratings	
apple.com/quicktime/download(Perfect)	quicktime-download.info(Good)
quicktime-download.info(Good)	apple.com/quicktime/download(Perfect)
softwarepatch.com...quicktime.html(Good)	softwarepatch.com...quicktime.html(Good)

### 5.4.3 Analysis over query segments

To understand which queries benefit from click-throughs of similar queries, we further analyze the results of the best method (Q,FM) + BM25F and the baseline method Q + BM25F. First, we sort all queries by the number of summed clicks in 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) + BM25F method over the baseline method on the 2000 queries. We get the first three columns in Fig. 4. The number of clicks for 2000 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 Nos. 501–2500 queries. There are still 2000 queries in the second query segment and there are 1500 queries that overlap 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 2000 queries. Two adjacent segments have 1500 overlapped queries, except for the last two query segments. As we have 7453 queries, the last query segment contains Nos. 5454–7453 queries and the second last segment contains Nos. 5001–7000 queries. The last two segments have 1546 overlapped queries. Finally we get twelve groups of columns as shown in Fig. 4.

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 improve, particularly 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 % of the queries in our dataset). For the queries with more than 135 clicks, our proposed method cannot significantly improve retrieval effectiveness measured on the top three results.

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



**Fig. 4** Analyze improvement of the (Q,FM) + BM25F method over the Q + BM25F method on different query segments that are partitioned based on the total number of clicks for a query. Imp means Improvement

more query segments. They are Nos. 1, 2, 3, 4, 6 and 7 query segments. This indicates that the aggregated click-throughs can benefit more queries, with the number of clicks up to 828 (about 50 % of the queries in our dataset), 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 relies 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. However, such results still indicate that our proposed method is helpful in improving users' satisfaction.

## 6 Conclusion and future work

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

We propose a two-step approach to address the problem. We implement a scalable query clustering method based on co-clicks. Then within each cluster, we group queries by fuzzy match between query strings. Experimental results indicate that our proposed approach improves the precision of co-click based cluster from 0.37 to 0.62. It also outperforms Bing Search's query alteration (lifting from 0.42 to 0.56) in terms of  $F_1$  measure.

For the application of web retrieval, we propose aggregating 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 replacement strategy. When we use 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 best in terms of NDCG@10. Based on the clusters, our proposed method using fuzzy match wins against the baseline method in terms of all three NDCGs, and the differences are statistically significant. Through analysis over different query segments, we find that similar queries' click-throughs benefit queries with fewer clicks more than the queries with rich clicks and show significant improvement over 25 % of the queries in terms of NDCG@3 and over 50 % of the queries in terms of NDCG@10.

For future work, we plan to examine more clustering methods on our manually labeled dataset. There is still room to improve in terms of recall. Also we are going to try more sophisticated models that integrate click information into ranking functions. Based on our analysis on which queries are improve or not improved, another interesting problem is how we could further improve ranking by selective use of click-throughs of similar queries. As a common pre-processing module of log mining, our approach can be used for other applications such as query suggestion. We will test how it work for such cases.

## References

- Agichtein, E., Brill, E., & Dumais, S. (2006). 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* (pp. 19–26). ACM.
- Agrawal, R., Gollapudi, S., Halverson, A., & Ieong, S. (2009). Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining* (pp. 5–14). New York, NY, USA: ACM.

- Ahmad, F., & Kondrak, G. (2005). Learning a spelling error model from search query logs. In *Proceedings of the 5th conference on HLT and EMNLP* (pp. 955–962). ACL.
- Beeferman, D., & Berger, A. (2000). Agglomerative clustering of a search engine query log. In *Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 407–416). ACM.
- Brill, E., & Moore, R. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting of the association for computational linguistics* (pp. 286–293). ACL.
- Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., & Li, H. (2008). Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 875–883). ACM.
- Chapelle, O., & Zhang, Y. (2009). A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on world wide web* (pp. 1–10). ACM.
- Cucerzan, S., & Brill, E. (2004). Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP* (vol. 4, pp. 293–300).
- Dang, V., & Croft, B. (2010). Query reformulation using anchor text. In *Proceedings of the 3rd ACM international conference on web search and data mining* (pp. 41–50). ACM.
- de Borda, J. C. (1781). Mémoire sur les élections au scrutin. *Histoire de l'Académie Royal des* (pp. 657–665). Paris.
- Fagni, T., Perego, R., Silvestri, F., Orlando, S., Ca, U., & Venezia, F. (2006). Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM Transactions on Information Systems*, 24(1), 51–78.
- Gao, J., Li, X., Micol, D., Quirk, C., & Sun, X. (2010). A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 358–366). ACL.
- Gao, J., Yuan, W., Li, X., Deng, K., & Nie, J. (2009). Smoothing clickthrough data for web search ranking. In *Proceedings of the 32nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 355–362). ACM.
- Granka, L., Joachims, T., & Gay, G. (2004). 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* (pp. 478–479). ACM.
- Guo, F., Liu, C., & Wang, Y. (2009). Efficient multiple-click models in web search. In *Proceedings of the second ACM international conference on web search and data mining* (pp. 124–131). ACM.
- Guo, J., Xu, G., Li, H., & Cheng, X. (2008). 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* (pp. 379–386). ACM.
- Hu, Y., Qian, Y., Li, H., Jiang, D., Pei, J., & Zheng, Q. (2012). Mining query subtopics from search log data. In *Proceedings of the 35th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 305–314). ACM.
- Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4), 422–446.
- Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 133–142). ACM.
- Kernighan, M., Church, K., & Gale, W. (1990). A spelling correction program based on a noisy channel model. In *Proceedings of the 13th international conference on computational linguistics* (vol. 2, pp. 205–210). Association for Computational Linguistics.
- Li, M., Zhang, Y., Zhu, M., & Zhou, M. (2006). Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics* (pp. 1025–1032). ACL.
- Och, F. (2002). *Statistical machine translation: from single-word models to alignment templates*. Aachen: RW TH Aachen.
- Peng, F., Ahmed, N., Li, X., & Lu, Y. (2007). Context sensitive stemming for web search. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 639–646). ACM.
- Robertson, S., Zaragoza, H., & Taylor, M. (2004). Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on information and knowledge management* (pp. 42–49). New York, NY, USA: ACM.
- Robertson, S.E. and Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR*

- conference on research and development in information retrieval* (pp. 232–241). New York, NY, USA: Springer.
- Sadikov, E., Madhavan, J., Wang, L., & Halevy, A. (2010). Clustering query refinements by user intent. In *Proceedings of the 19th international conference on world wide web* (pp. 841–850). ACM.
- Sakai, T., Dou, Z., Yamamoto, T., Liu, Y., Zhang, M., & Song, R. (2003). Overview of the ntcir-10 intent-2 task. In *Proceedings of NTCIR-10*.
- Song, Y., & He, L. (2010). Optimal rare query suggestion with implicit user feedback. In *Proceedings of the 19th international conference on world wide web* (pp. 901–910). ACM.
- Tyler, S., & Teevan, J. (2010). Large scale query log analysis of re-finding. In *Proceedings of the 3rd ACM international conference on web search and data mining* (pp. 191–200). ACM.
- Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1), 168–173.
- Wang, X., & Zhai, C. (2008). Mining term association patterns from search logs for effective query reformulation. In *Proceeding of the 17th ACM international conference on information and knowledge management* (pp. 479–488). ACM.
- Wen, J., Nie, J., & Zhang, H. (2002). Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1), 59–81.
- Wu, W., Xu, J., Li, H., & Oyama, S. (2011). Learning a robust relevance model for search using kernel methods. *Journal of Machine Learning Research*, 12, 1429–1458.
- Xue, G., Zeng, H., Chen, Z., Yu, Y., Ma, W., Xi, W., & Fan, W. (2004). Optimizing web search using web click-through data. In *Proceedings of the thirteenth ACM international conference on information and knowledge management* (pp. 118–126). ACM.
- Zhang, Y., Chen, W., Wang, D., & Yang, Q. (2011). User-click modeling for understanding and predicting search-behavior. In *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1388–1396). ACM.
- Zhu, Z., Chen, W., Minka, T., Zhu, C., & Chen, Z. (2010). A novel click model and its applications to online advertising. In *Proceedings of the third ACM international conference on web search and data mining* (pp. 321–330). ACM.