Web Intelligence 14 (2016) 119–138
DOI 10.3233/WEB-160335
IOS Press

Accurate and efficient query clustering via top ranked search results

Yuan Hong [a,*], Jaideep Vaidya [b], Haibing Lu [c] and Wen Ming Liu [d]
[a] *Department of Information Technology Management, University at Albany, SUNY, USA*
E-mail: hong@albany.edu
[b] *Department of Management Science and Information Systems, Rutgers University, USA*
E-mail: jsvaidya@business.rutgers.edu
[c] *Department of Operations and Management Information Systems, Santa Clara University, USA*
E-mail: hlu@scu.edu
[d] *Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada*
E-mail: l_wenmin@ciise.concordia.ca

AUTHOR COPY

Abstract. To make the search engine more user-friendly, commercial search engines commonly develop applications to provide suggestion or recommendation for every posed query. Clustering semantically similar queries acts as an essential prerequisite to function well in those applications. However, clustering queries effectively is quite challenging, since they are usually short, incomplete and ambiguous. Existing prevalent clustering methods, such as K-Means or DBSCAN cannot guarantee good performance in such a highly dimensional environment. Through analyzing users' click-through query logs, hierarchical agglomerative clustering gives good results but is computationally quite expensive.

This paper identifies a novel feature for clustering search queries based on a key insight – queries' top ranked search results can themselves be used to quantify query similarity. After investigating such feature, we propose a new similarity metric for comparing those diverse queries. This facilitates us to develop two very efficient and accurate algorithms integrated in query clustering. We conduct comprehensive experiments to compare the accuracy of our approach against the known baselines along two dimensions: 1) quantifying the cohesion/separation of clustered queries, and 2) justifying the results by real-world Internet users. The experimental results demonstrate that our two algorithms and the similarity metric can generate more accurate results within a significantly shorter time.

Keywords: Web search, query log, clustering, top-k search results, clustering validation

1. Introduction

As of today, the indexed web contains at least 30 billion pages [1]. In fact, the overall web may consist of over 1 trillion unique URLs, more and more of which is being indexed by search engines every day. Out of this morass of data, users typically search for the relevant information that they want through posing search queries to search engines. The problem that the search engines face is that the queries are very diverse and often quite vague and/or ambiguous in terms of user requirements. Many different queries may refer to a single concept, while a single query may correspond to many concepts. To organize and bring some order to this massive unstructured dataset, search engines cluster these queries to group similar items together. To increase usability, most commercial search engines, such as Google, Yahoo!, and Bing also augment their search facility through additional services such as query recommendation or query suggestion by guessing the users' search intents. For example, query recommendation tries to recommend similar search queries and results based on an executed query [20]; query suggestion tries to provide some candidate completed queries for the user before he/she keys the entire keyword(s) into the search engine [12]. These services make it more convenient for users to issue queries and obtain accurate results from the search engine, and

[*] Corresponding author. E-mail: hong@albany.edu.

thus are quite valuable. From the search engine perspective, effective clustering of search queries is a necessary pre-requisite for these services to function well. Due to all of these reasons, clustering of search engine queries has attracted significant attention in recent years. However, existing prevalent clustering methods, such as K-Means/Medoids or DBSCAN cannot assure a good performance in such highly-dimensional and diverse environment – with constraints on specifying parameters. Hierarchical agglomerative clustering [10] gives good results but is computationally quite expensive – with difficulty on scaling to large datasets.

There are several challenges raised due to the unique nature of the environment. The principal issue is to determine how to measure similarity between queries. To enable more precise information retrieval, a representative and accurate descriptor is indispensable for computing the similarity between queries. The concept of query similarity was originally used in information retrieval studies [19]. Each query is represented by a vector of content-based keywords, and then every pair of queries can be compared using some similarity metrics. However, the problem with using this in the query log environment is that users' search interests are not always the same even if the issued queries contain the same keywords. A typical example is that, the keyword "Apple" may represent either fruit or the "Apple Inc.". Hence, the use of content-based keywords descriptor is rather limited for this purpose. Subsequently, to measure the similarity between two queries, their vectors of the associated clicked URLs in a click-through bipartite graph [10,12,45] are adopted. Take "Apple" and "Apple and Banana" as an example, web users normally click some URLs after submitting them to the search engine respectively. Then, the similarity can be computed by the binary [10] or count [12] vectors of their clicked URLs.

Nevertheless, no matter how large the query log dataset may be, the complete search intent of some relevant queries cannot always be effectively represented by their click-through information. For instance, if there is no common clicked URL for two queries (e.g., "Honda accord Toyota camry", "Civic vs. Corolla") in a given query log (this happens in the AOL real dataset), "Honda accord Toyota camry" and "Civic vs. Corolla" will never be clustered together, though they are semantically similar to each other – also note that content-based keywords descriptor and similarity measure do not work either in this case. In other words, if using query logs, we can only cluster the queries in the click-through bipartite graph, whereas many of their semantically similar queries without any clicked URL in the query log are not available for analysis and applications. Another reason that causes inaccuracy is that the query log data comprise users' click-through information in a specific period, while the search interests might even change over time. Even if we can partition the query logs in terms of time intervals, the transition of the varying search interests is not very clear, and the returned information with respect to the submitted query might be related to a totally different meaning in the past. For example, "Sandy" may refer to different meanings before and after the hurricane happened in late 2012. If we aggregate them together for clustering analysis, the result might be inaccurate.

AUTHOR COPY

Therefore, describing queries only by content-based keywords or purely through click-through data is not always sufficiently accurate for search engine query clustering. As observed from the query logs, each clicked URL is logged with a rank number when the user clicks it. The top ranked search results (shown in the 3rd column of Table 1) represent the top relevant URLs for any query at the time it was posed. They perfectly incorporate both content-based and click-through information by the robust search engine [3,4,32,33,37]. Also, since almost all the queries' top ranked search results can be generated by the search engine at any time, the shortcomings of using query logs for clustering can be appropriately resolved ("Honda accord Toyota camry" and "Civic vs. Corolla" can be recommended for each other; "Sandy" means either a human name or the hurricane at a specific time). Along this line of research, in this paper, we identify a novel feature for clustering search engine queries, that is – using the top ranked search results. Furthermore, while using this new feature for clustering, we define a new similarity metric and develop two early termination strategies for the clustering algorithm, which eliminates some unnecessary processes in similarity computation. Then, the main contributions of this paper are summarized as below:

1. To the best of our knowledge, we take the first step to do query clustering analysis on the top ranked search results, which is based on a key insight – *search engine results can themselves be used to quantify query similarity*.
2. A new similarity metric is given to measure the similarity between every pair of queries based on their top ranked search results, in which every URL and its rank jointly decide the similarity.

Table 1
Query Logs vs. Top Ranked Search Results

Search Query	Query Logs (*Clicked URL and Count*)	Top 5 Search Results (*Rank* and *URL*)
"Honda accord Toyota camry"	www.honda.com (3 times) www.autotrader.com (2 times)	1 www.autotrader.com 2 www.thecarconnection.com 3 www.youtube.com 4 www.autoguide.com 5 automobiles.honda.com
"Civic vs. Corolla"	www.kbb.com (2 times)	1 www.autotrader.com 2 www.kbb.com 3 www.autobytel.com 4 www.autoguide.com 5 www.thecarconnection.com
...	...	...

3. Observing the characteristics of the proposed similarity metric, we develop two early termination strategies that can significantly improve the efficiency of query clustering.
4. The experimental results show that our approach outperforms the state-of-the-art algorithms such as hierarchical agglomerative clustering and K-Means/Medoids clustering in terms of accuracy, efficiency and scalability.

This paper is significantly extended from our preliminary work [28]. Specifically, we proposed a new algorithm besides the algorithm in [28], which also outperforms the state-of-the-art approaches in terms of the efficiency and accuracy. Moreover, we conducted more comprehensive groups of experiments in to evaluate the quality of clustering using both internal and external measures [41] which can bring new insight into the accuracy evaluation of query clustering algorithms from a completely different perspective beyond [28]. More experimental results on testing the scalability and the effectiveness of the parameters in our proposed clustering approach are also presented. Also, we have given additional theoretical study in this context, e.g., proving the properties of our proposed similarity metric, as well as more literature review.

The remainder of this paper is organized as follows. We first describe some preliminaries in Section 2, and review the literature in Section 3. Then we present our similarity metric for the top ranked search results in Section 4, and give the efficient algorithms in Section 5. Before going to the experiments, we discuss the evaluation measures for comparing the clustering results in Section 6. Section 7 demonstrates the experimental results. Finally, we conclude this paper and discuss the future work in Section 8.

2. Preliminaries

Studies on clustering queries using click-through data [10,12,46] generally extract users' queries and clicked URLs from the query logs during a specified time period. Beeferman and Berger [10] proposed an agglomerative clustering approach using the Jaccard similarity metric: for two queries, q_x and q_y, we have

$$Sim(q_x, q_y) = \frac{|\mathcal{N}(x) \cap \mathcal{N}(y)|}{|\mathcal{N}(x) \cup \mathcal{N}(y)|} \tag{1}$$

where $|\mathcal{N}(x) \cap \mathcal{N}(y)|$ denotes the number of common unique URLs that q_x and q_y share while $|\mathcal{N}(x) \cup \mathcal{N}(y)|$ represents the total number of these two queries' unique URLs in the click-through bipartite graph. The main idea of this algorithm is to hierarchically agglomerate clusters according to the similarity between queries. Leung et al. [34] pointed out two limitations of Beeferman et al.'s agglomerative clustering: 1) its low recall rate due to small number of common clicks on the documents; and 2) the susceptibility of the approach to noise (i.e. "noise URLs" cannot be identified since all the clicked links are treated in the same way, and the counts of the clicks are not considered in the clustering process). Furthermore, the method also has a high computational cost.

Similarity between queries can also be measured by other metrics with click-through data such as the Euclidean distance or Cosine similarity. For example, the Euclidean distance of two clicked URLs vectors[12] can be obtained by normalizing q_x and q_y's clicked URLs vectors v_x, v_y into v'_x and v'_y (where the normalized count of two queries' ith distinct URL are $v'_x(i) = \frac{v_x(i)}{\sqrt{\sum_i v_x^2(i)}}$ and $v'_y(i) = \frac{v_y(i)}{\sqrt{\sum_i v_y^2(i)}}$ respectively)

AUTHOR COPY

and computing the distance between two queries by

$$\|q_x - q_y\| = \sqrt{\sum_i \left\| v'_x(i) - v'_y(i) \right\|^2} \tag{2}$$

Similarly, the Cosine similarity of two clicked URLs vectors, v_x and v_y (the number of clicks for each URL), can be computed as

$$Sim(q_x, q_y) = \frac{\sum_{i=1}^{k} v_x(i) \times v_y(i)}{\sqrt{\sum_{i=1}^{m} v_x^2(i)} \times \sqrt{\sum_{i=1}^{n} v_y^2(i)}} \tag{3}$$

where q_x, q_y have m, n distinct URLs respectively and k common URLs.

Once a similarity or distance metric is defined, it can be utilized by any clustering algorithm, such as hierarchical agglomerative clustering [10], K-Means/ Medoids [6], and DBSCAN [44], etc.

3. Related work

Search engine queries are a rich source of information and many applications have been developed to extract similar search information for queries, etc. However, these enhanced query services require effective query clustering. Specifically, Baeza-Yates et al. [6] presented a query recommendation method that is based on clustering groups of semantically similar queries. Cao et al. [12] proposed an approach to context-aware query suggestion by mining click-through and session data. Baeza-Yates et al. also presented a framework [7] for clustering search engine queries that aims at developing relevant applications based on similar queries (e.g., Answer Ranking, Query Recommendation).

Query clustering has its roots in information retrieval research [19]. Each query is represented by a vector of keywords while measuring similarity by the Jaccard coefficient of common keywords. Since most of the keywords are ambiguous, analyzing the content of query keywords or phrases by traditional information retrieval techniques has many limitations. Following that, click-through query logs have been mined to yield similar queries [17,20,39,46]. Beeferman and Berger [10] first introduced the agglomerative clustering method to discover similar queries using query logs but with limitations (noise and small number of common clicks). The query clustering approach adopted in [6] uses a K-Means clustering approach. K-Means algorithm cannot adapt well in query clustering case due to the difficulty on specifying k. Wen et al. [44,45] analyzed both query contents and click-through bipartite graph and applied DBSCAN [21], a density-based method to group similar queries. Fonseca et al. [23] propose to discover related search engine queries by association rules. They consider the query log as a set of transactions while a submitted query is regarded as a single transaction. As addressed in [8], two problems in this method arise: it is difficult to determine query sessions and interesting related queries submitted by different users cannot be discovered. [8] presents methods on extracting semantic relations from query logs and represents queries in a vector space based on a graph derived from click-through bipartite graph. Dupret et al. [20] extract relations among queries that are defined using the rank of clicked URLs. Their goal is discovering relations by click-through data, which is different from our purpose.

More specifically, Leung et al. [34] proposed a two-phase personalized agglomerative clustering algorithm for generating personalized query clusters. In the query clustering stage, they extended agglomerative clustering to a concept-based algorithm. Similar to the approach proposed by Beeferman et al., this method cannot scale up to large datasets whereas our partial transition similarity based algorithms significantly reduce the computational cost. The similarity of queries in [34] is based on concepts rather than clicked URLs because the chance for different queries leading to common clicked URLs is low if using clicked data. Our solution is also not based on click-through data, we select top ranked search results to resolve this problem. Similar to agglomerative query clustering, DBSCAN algorithm adopted in [44,45] requires high computational cost. Meanwhile, Wen et al. linearly combine measures on content-based similarity and cross-references based similarity but it's difficult to set parameters for linear combination of two similarity metrics. However, our ranked search results data (enforced by [3,4,32,33,37]) naturally consider both factors if the top ranked search results are provided by a robust search engine. Moreover, Cao et al. [12] use normalized Euclidean distance to measure similarity between queries in click-through data. We have compared the performance of our transition based algorithms and the normalized Euclidean distance based query clustering algorithms in our experiments. He et al. [26] utilizes the term proximity evidence to improve the probabilistic information retrieval models. Another category of query anal-

AUTHOR COPY

ysis methods map queries into knowledge bases such as linked open data [5,43] and estimate the similarities of queries and/or keywords based on such mappings.

Furthermore, Fu et al. [24] presented a hybrid method to cluster queries by utilizing both the query terms and the returned URLs to the queries. The query similarity in [24] also considers contents and URLs. The second portion of the similarity is computed based on the overlapped URLs of the queries. Bordogna [11] proposed operators to combine clustered search results based on the common URLs. Chuang et al. [15] proposed a hierarchical query clustering approach by organizing the query terms into hierarchical structure and construct query taxonomies. This approach purely considers the terms in snippets rather than an overall similarity/distance measurement for human search intents.

Kendall's tau [22] and some relevant measures [47] can be effective in measuring the accuracy of clustered queries. Since our most significant contribution is observing the fact that search results can be used to perform clustering of query keywords, they might be used instead of our proposed similarity metric as well (though they are not very efficient). Also, since search queries clustering is a well known hard problem, to the best of our knowledge, very few algorithms have been raised for improving the efficiency of clustering queries. This paper is a complement to the literature from this perspective.

Finally, silhouette coefficient is employed as one of our internal measures to measure the cohesion and separation of query clustering results in our preliminary work [28]. Greater Silhouette coefficient indicates large inter-cluster distances and small intra-cluster distances. Davies-Bouldin index [18,38] seeks the same objective on clustering validation. The distinction between silhouette coefficient is that – minimized index generates the best clustering results. Besides clustering, some other techniques can be also used for query suggestion and recommendation (which are relevant to our work), such as random walks [16] and some work summarized in [40]. Our proposed approach clusters search queries using the web search's feature as the ranked URLs of the web pages, and some work on web site/page feature selection [14,42] are also relevant to our study.

4. Similarity metric for top ranked search results

In this section, we introduce our novel similarity metric based on the queries' search results lists.

4.1. Search results list

Recall that Table 1 (2nd column) shows the click-through information of two search queries. A more detailed description of the motivating example is given as follows. The table shows all of the associated URLs for two similar queries – "Honda accord Toyota camry" and "Civic vs. Corolla". Looking at this, it is clear that they cannot be grouped together due to the scarcity of click-through information among the query logs. For most prevalent similarity metrics such as Jaccard Similarity and Euclidean Distance, based on the common clicked URLs, the similarity between them is "0". Thus, if building the query recommendation with this query log, such two queries cannot be recommended for each other.

Instead, we utilize a list of search results for each submitted query at a specific time to measure the query similarity. As also shown in Table 1 (3rd column), each of the two queries has a list of search results. The similarity between them would not be "0" anymore, and thus those two semantically similar queries can be grouped together using the feature of top-k search results. Following this, we define any query q's search results as a list of URLs $\mathcal{L}_q$. We only consider q's "top-k" search results (viz. top-k list) rather than its entire URL list since the lower ranked URLs are not that relevant to q. Hence, we have:

Definition 1 (Top-k List $\mathcal{L}_q(k)$)**.** For any query q with a given list of search results $\mathcal{L}_q$, we define q's top-k search results as $\mathcal{L}_q(k) = \{d_i, \forall i \in [1, k]\}$ where d_i represents q's ith URL in the list.

Note that the URLs on the top-k list have been ranked by the search engine according to the relevance of the URL to the possible search intention of such query. The first few URLs are generally much more relevant to the search intent. Therefore, we can assign a weight $\omega(i)$ to denote the relevance of the ith URL in the list to the exact search intent of that query. Hence, all relevance weights form a "Weight Sequence" $\Omega = \{\omega(i), \forall i \in [1, k]\}$. For all queries, the relevance weight $\omega(i) \in \Omega$ decreases as the rank i increases. This can be observed from any robust search engine and users' behavior study: the search engine ranking algorithms sort the URLs according to the relevance beforehand; if a URL ranks higher, it attracts significantly more clicks (as seen from the query log [9,25]). So we can hypothesize that $\omega(i)$ decreases sharply as the rank i increases. Then, every query's top-k URLs list $\mathcal{L}_q(k)$ and the assigned weight se-

AUTHOR COPY

quence of the relevance Ω can be used to measure the similarity among them.

4.2. Transition similarity with top-k lists

The key procedure for clustering queries is measuring the similarity between all queries. Since $\mathcal{L}_q(k)$ is the top ranked URLs, we could use the *Kendall's tau* [22] as the underlying distance metric. However, Kendall's tau is not that effective if the top-k lists have little overlap, which is often the case for search engine queries' URL lists. Additionally, scalability is an essential concern for the clustering problem of search engine queries, and Kendall's tau is not quite as efficient to compute when large number of top-k lists' comparison is required. To improve on this, we develop a new metric that measures the similarity between any pair of queries' top-k lists. Section 5 shows how to improve the efficiency of the query clustering with this similarity metric.

With the top-k lists, the similarity between two different queries q_x and q_y can be measured through their two ranked URL sets $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$. First, we identify the common URLs in $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$. We denote the number of common URLs as m and the common URLs as $d_1, \ldots, d_m$ where $m \in [0, k]$. However, simply comparing the two lists through their common URLs such as by using the Jaccard coefficient [10], or Euclidean distance [12] is not sufficiently accurate, for the reason – the common URLs determine the similarity between two queries in terms of not only how many common URLs, but also their ranks on two lists. For instance, "Apple" and "Apples" have many common URLs in their search results. If we do not consider the influence of the URL ranks, they might be computed as very similar. Indeed, they refer to two different search intents because users might not use "Apples" to search the information related to "Apple Inc." (the rank of the common URLs varies on these two lists). We then define:

Definition 2 (Rank Transition Value)**.** Given queries q_x and q_y with their top-k lists $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$, the common URL d_i is ranked $r_x(i)$ and $r_y(i)$ in $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$ respectively, then d_i's rank transition value is defined as $\Delta_{d_i}(q_x, q_y) = |r_x(i) - r_y(i)|$ where $r_x(i), r_y(i) \in [1, k]$.

Recall that a common URL in $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$ would have different relevance weight if they are ranked distinctly. Hence, our similarity measure should consider two factors: the relevance weight of the URL on two lists and the rank transition there. We thus define the "Transition Similarity" between two queries q_x and q_y as follows:

$$Sim(q_x, q_y) = \sum_{i=1}^{m} \frac{(\omega[r_x(i)] + \omega[r_y(i)])/2}{\Delta_{d_i}(q_x, q_y) + 1} \qquad (4)$$

Here, m is the number of common URLs on $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$, $r_x(i)$ and $r_y(i)$ represent the rank of common URL d_i on $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$ respectively, and $\Delta_{d_i}(q_x, q_y)$ denotes the rank transition value for d_i. For every common URL d_i, we average its relevance weight on two lists. Since $\Delta_{d_i}(q_x, q_y)$ may be valued at 0, and has an anti-monotonic relationship to the similarity, we let $\Delta_{d_i}(q_x, q_y) + 1$ be the denominator.

Assuming that q_x = "Honda accord Toyota camry", q_y = "Civic vs. Corolla" in Table 1 and $\omega(i) = \frac{1}{2^i}$, the rank of three common URLs varies as $1 \to 1$ (no change), $2 \to 5$ and $4 \to 4$ (no change). Thus, $Sim(q_x, q_y) = 0.5 + \frac{(0.25+0.03125)/2}{3+1} + 0.0625 = 0.703$.

4.3. Properties of transition similarity

Proposition 1. *For any two queries q_x and q_y, we have $0 \leqslant Sim(q_x, q_y) \leqslant \sum_{i=1}^{k} \omega(i)$.*

Proof. Two special cases reflect the maximum and minimum value of transition similarity:

- If $\mathcal{L}_{q_x}(k) = \mathcal{L}_{q_y}(k)$, then $m = k$ and $\forall d_i$, $\Delta_{d_i}(q_x, q_y) = |r_x(i) - r_y(i)| = 0$. Thus, we have $\forall d_i$, $\frac{1}{\Delta_{d_i}(q_x,q_y)+1} = 1$ and $\max\{Sim(q_x, q_y)\} = \sum_{i=1}^{k} \omega(i)$.
- If $\mathcal{L}_{q_x}(k) \cap \mathcal{L}_{q_y}(k) = \emptyset$ (no common URL), $Sim(q_x, q_y)$ can be simply obtained as 0.

Furthermore, if $\sum_{i=1}^{k} \omega(i) = 1$, for instance, $\lim_{k\to\infty} \sum_{i=1}^{k} \omega(i) = 1$ when $\omega(i) = \frac{1}{2^i}$, transition similarity is naturally normalized. □

As shown above, the weight sequence $\omega(i) = \frac{1}{2^i}$ can approximately normalize the transition similarity if k is large. In general, if $\sum_{i=1}^{k} \omega(i) \neq 1$, we can normalize the transition similarity by normalizing Ω as $\omega'(i) = \frac{\omega(i)}{\sum_{i=1}^{k} \omega(i)}$ where $\sum_{i=1}^{k} \omega(i)$ is the maximum value derived above. Hence, given two top-k lists ($\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$) with the relevance weight sequence Ω, we have the normalized transition similarity as below:

$$Sim'(q_x, q_y) = \frac{1}{\sum_{i=1}^{k} \omega(i)} \sum_{i=1}^{m} \frac{(\omega[r_x(i)] + \omega[r_y(i)])/2}{\Delta_{d_i}(q_x, q_y) + 1} \qquad (5)$$

AUTHOR COPY

It can be proven that the distance derived from transition similarity is a metric (detailed proof is given in Appendix A). The transition similarity referred in the following sections has been normalized. For simplifying the notation, we let $Sim(q_x, q_y)$ denote the normalized one.

5. Efficient clustering algorithms

Since computing the transition similarity is an essential step for clustering, we first discuss how to efficiently do so, and then present our approaches for clustering queries with it.

5.1. Partial transition similarity

Some query clustering algorithms (e.g., agglomerative clustering) are implemented based on a given similarity or distance threshold where the distances (or similarities) between every pair of queries in the same cluster cannot exceed (or be less than) the threshold. Thus, computing the similarity/distance for every pair of queries is a preliminary step in such query clustering algorithm.

Nevertheless in some cases, with a given similarity threshold δ, if the query similarity comparison can sufficiently return the result that whether the similarity is over or under δ, the similarity computation can be terminated immediately. Particularly in clustering queries with top-k lists, the top ranked URLs in the lists dominate the similarity comparison and thus possibly lead to the early termination in query comparison. We aim at improving the efficiency of clustering queries by doing so, and term this as "Partial Transition Similarity".

More specifically, since $\omega(i)$ decreases as i increases, the higher ranked URLs essentially contribute a greater portion to the overall similarity. Making use of this, we can develop early termination strategies that only derive the "binary" outcome whether the transition similarity is $\geqslant\delta$ or $<\delta$ instead of computing the exact similarity value. Intuitively, we can terminate the similarity computation if the top ranked URLs have already made the accumulated transition similarity $a \geqslant \delta$. On the other hand, if the overall transition similarity $Sim(q_x, q_y)$ cannot achieve δ even though all the remaining URLs in $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$ are identical (this case gets the maximum overall similarity for all the URLs, but it is still less than δ), we can also terminate the similarity computation process. Hence, we can derive *Partial Transition Similarity* based query clustering algorithms with the minimum similarity threshold δ for each cluster. Before that, we first define two concepts – "Similarity Threshold URL" and "Dissimilarity Threshold URL".

Definition 3 (SIMILARITY THRESHOLD URL $\overrightarrow{d}$)**.** Given a weight sequence Ω and a minimum similarity threshold δ, the nth URL ($n \leqslant k$) of q_x (or q_y) is called Similarity Threshold URL $\overrightarrow{d}$ for $Sim(q_x, q_y)$, if it is the first URL at which the accumulated partial transition similarity of q_x and q_y satisfies $a \geqslant \delta$.

Definition 4 (DISSIMILARITY THRESHOLD URL $\overleftarrow{d}$)**.** Given a weight sequence Ω and a minimum similarity threshold δ, the nth URL ($n \leqslant k$) of q_x (or q_y) is called Dissimilarity Threshold URL $\overleftarrow{d}$ for $Sim(q_x, q_y)$ if it is the first URL at which the accumulated partial transition similarity of q_x and q_y satisfies $a < \delta - max$ where max represents the maximum partial transition similarity obtained from all the remaining URLs in two lists.

Clearly, for any pair of queries q_x and q_y, either the "Similarity Threshold URL" $\overrightarrow{d}$ or "Dissimilarity Threshold URL" $\overleftarrow{d}$ (whichever is detected earlier on two top-k lists) is sufficient to conclude that q_x and q_y should be clustered together or not. Since the density of the similarity matrix for web query clustering is typically very low (less than 2% [41]), the similarity between most queries is "0". Frequently, we can discover $\overleftarrow{d}$ or $\overrightarrow{d}$ at the early stage of similarity computation, and go to the next pair of queries. Thus, the computational cost can be significantly reduced in either case.

We propose two different ways to efficiently find $\overleftarrow{d}$ or $\overrightarrow{d}$. They differ primarily in terms of the how the URLs on the top-k lists are traversed, whether "traverse all the URLs on one top-k list (sequence) first" or "traverse two URLs with the same rank (level) first". Note that we can indeed regard all URLs on a top-k list as a URL *sequence* and the URLs on two top-k lists with the same rank as a *level*. Therefore, we denote these two algorithms as "Sequence-first Traversal" and "Level-first Traversal" respectively, and now present them below.

5.2. Sequence-first traversal algorithm

We first introduce the sequence-first traversal for finding $\overleftarrow{d}$ and $\overrightarrow{d}$ between two queries' top-k lists. With sequence-first traversal, it is immaterial which of the two URL sequences is traversed. As long as all URLs from the first to the last in either sequence are

AUTHOR COPY

traversed, the overall transition similarity computed is the same since the transition similarity shares of all *common* URLs in both sequences have been accumulated. Since the accumulated transition similarity a can be simply calculated by summing over the transition similarity share of each URL on one top-k list, the "Similarity Threshold URLs" $\overrightarrow{d}$ can be easily discovered by terminating the process as soon as $a \geqslant \delta$. However, the steps for discovering $\overleftarrow{d}$ is more complicated: we must somehow compute the maximum partial transition similarity (denoted as *max*) among the remaining URLs in the traversal route, and use it to check the satisfaction condition. Since finding *max* is an optimization problem (Bipartite Matching [36]), the sequence-first traversal is not that efficient if we try to solve the problem at each URL visit. Instead, we can find a slack upper bound – max' for the partial similarity of all the remaining untraversed URLs which is no less than the exact maximum untraversed partial similarity *max* (Lemma 1 shows how to derive this). As a result, the early termination can be efficiently implemented with max' and Lemma 1 (Proven in Appendix B).

Lemma 1. *At q_x's nth URL (sequence-first traversal), the partial transition similarity for untraversed URLs is bounded by value* $\sum_{i=n+1}^{k}\max\{\omega(i), \frac{\omega(i)+\omega(r_y)}{2(|i-r_y|+1)}\}$ *where r_y is the current highest rank of all untraversed URLs on $\mathcal{L}_{q_y}(k)$.*

Since most queries have a very small number of common URLs on their top-k lists, most ranks on q_y's top-k list haven't been occupied while we are traversing q_x's sequence. Therefore, the ranks $\omega(r_y)$ and $\omega(i)$ are generally available. Meanwhile, since $\omega(i)$ drops down sharply, the comparison should be terminated in a very early stage. Then, Algorithm 1 presents the approach to discover dissimilarity threshold URL and similarity threshold URL by sequence-first traversal.

For example in Fig. 1, we traverse URLs in q_x's sequence. Implementing Lemma 1 to compute the slack upper bound max', we can obtain $max' = 0.72, 0.47, 0.34, 0.19, \ldots$ (as shown in Fig. 1, a is accumulated by adding the transition similarity share of the current URL, and max' decreases). Even though $0.72, 0.47, \ldots$ cannot be achieved by *max*, through relaxing the *max* to them, Algorithm 1 can greatly reduce the complexity of clustering and acquire the same result as completely computing the similarity. Finally, the 4th URL in $\mathcal{L}_{q_x}(k)$ can be identified as the dissimilarity threshold URL $\overleftarrow{d}$. Conversely, if we let $\delta = 0.15$ in the example, the similarity threshold URL

Algorithm 1 Sequence-first Traversal

```
Input: two queries $q_x$ and $q_y$, $\delta$ and $\Omega$
Output: $Sim(q_x, q_y) \geqslant \delta$ at $\overrightarrow{d}$ or $Sim(q_x, q_y) < \delta$ at $\overleftarrow{d}$
1:  $\{d_i$ is the $i$th URL of $q_x$ where $i \in [1, k]\}$
2:  while $d_i$ is neither $\overrightarrow{d}$ nor $\overleftarrow{d}$ do
3:      if $d_i$ is the $j$th URL of $q_y$ where $j \in [1, k]$ then
4:          $Sim(q_x, q_y) \leftarrow Sim(q_x, q_y) + \frac{\omega(i)+\omega(j)}{2(|i-j|+1)}$;
5:      end if
6:      $max' \leftarrow \sum_{s=i+1}^{k} \max\{\omega(s), \frac{\omega(s)+\omega(r_y)}{2(|s-r_y|+1)}\}$ and $r_y$
        is the highest available rank in $\mathcal{L}_{q_y}(k)$;
7:      $i++$;
8:      $d_i$ is $\overrightarrow{d}$ if $Sim(q_x, q_y) \geqslant \delta$, and is $\overleftarrow{d}$ if $max' +$
        $Sim(q_x, q_y) < \delta$;
9:  end while
10: return $Sim(q_x, q_y) \geqslant \delta$ or $Sim(q_x, q_y) < \delta$.
```

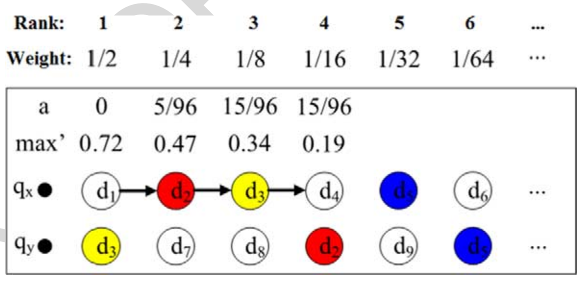


Fig. 1. Sequence-First Traversal ($\delta = 0.35$).

$\overrightarrow{d}$ is detected at the 3rd URL and the algorithm terminates.

Note that while traversing URLs in two queries' top-k URL sequences, the algorithm can choose either q_x or q_y's sequence. In some special cases, e.g., the common URLs d_1, d_2 and d_3 are ranked 1st, 2nd, 3rd in q_x's sequence and 1st, 2nd, 100th in q_y's sequence. At this time, traversing URLs in q_x's sequence might terminate earlier than traversing URLs in q_y's sequence. Indeed, the runtime difference between these two traversals can be not very significant for two reasons. First, the rank difference of d_3 in two sequence is 97, which makes the accumulated similarity of d_3 very close to 0. Then, traversing d_3 in q_x's sequence cannot increase the similarity significantly, and traversal has to continue (like traversing URLs in q_y's sequence). Second, another criterion can terminate the traversal – the sum of max' and accumulated similarity cannot reach the similarity threshold δ in any case. In the aforementioned special case, the traversal is very likely to be terminated at the top few levels while traversing

AUTHOR COPY

URLs in either sequence by meeting one of the following two conditions: (1) d_1 and d_2 make the accumulated similarity greater than δ, or (2) max' decreases rapidly and thus the sum of max' and accumulated similarity cannot reach δ. In summary, Algorithm 1 can traverse either query's URL sequence, though the runtime might be slightly different.

5.3. *Level-first traversal algorithm*

Besides traversing URLs along one query's top-k URLs list, we can also traverse URLs level by level (*the URLs in the same level of two top-k lists hold the same rank*) for discovering similarity threshold URLs and dissimilarity threshold URLs. For sequence-first traversal method, we consider all possibilities in the other query's top-k list and choose the greater one from $\omega(i)$ and $\frac{\omega(i)+\omega(r_y)}{2(|i-r_y|+1)}$ as the individual upper bound of transition similarity with respect to every remaining URL. However, if we check the two URLs in the same level, max can only be reached when the two URLs in all the lower levels are identical. More specifically, if some of the lower level URLs have been chosen to match URLs in higher levels, max cannot be reached for all URLs in lower levels. At this time, we still assume that all URLs in lower levels can be identical on both URLs and ranks such that a slack upper bound of max is also employed (Lemma 2 is also proven in Appendix C).

Lemma 2. *At the nth level (level-first traversal), the partial transition similarity for untraversed URLs is bounded by* $\sum_{i=n+1}^{k}\omega(i)$.

We thus derive the level-first traversal algorithm (Algorithm 2) to find the "Similarity Threshold URLs" or "Dissimilarity Threshold URLs" by traversing URLs along two queries' top-k lists level by level. As shown in Fig. 2, we use the same example to describe this method. a is accumulated by adding the transition similarity share of the URLs at the current level, and the decreasing slack upper bound of max can be computed with Lemma 2. Therefore, we can easily identify a URL (either from $\mathcal{L}_{q_x}(k)$ or $\mathcal{L}_{q_y}(k)$) in the 4th level as the "Dissimilarity Threshold URL" $\overleftarrow{d}$. Conversely, if we let $\delta = 0.15$ in the example, it turns out to discover "Similarity Threshold URL" $\overrightarrow{d}$ by terminating at the 2nd level.

5.4. *Applying early termination to query clustering*

While clustering queries using their top-k lists, our contributions primarily lie in two folds.

Algorithm 2 Level-first Traversal

```
Input: two queries $q_x$ and $q_y$, $\delta$ and $\Omega$
Output: $Sim(q_x, q_y) \geqslant \delta$ at $\overrightarrow{d}$ or $Sim(q_x, q_y) < \delta$ at $\overleftarrow{d}$
 1: {$d_i$ (or $d_i'$) is the $i$th URL of $q_x$(or $q_y$) where $i \in [1, k]$}
 2: while $d_i$ (or $d_i'$) is neither $\overrightarrow{d}$ nor $\overleftarrow{d}$ do
 3:   if $d_i$ is the $j$th URL of $q_y$ where $j \in [1, k]$ then
 4:     $Sim(q_x, q_y) \leftarrow Sim(q_x, q_y) + \frac{\omega(i)+\omega(j)}{2(|i-j|+1)}$;
 5:   end if
 6:   if $d_i'$ is the $j'th$ URL of $q_x$ where $j' \in [1, k]$ then
 7:     $Sim(q_x, q_y) \leftarrow Sim(q_x, q_y) + \frac{\omega(i)+\omega(j')}{2(|i-j'|+1)}$;
 8:   end if
 9:   $max \leftarrow \sum_{s=i+1}^{k}\omega(s)$; $i++$;
10:   $d_i$ (or $d_i'$) is $\overrightarrow{d}$ if $Sim(q_x, q_y) \geqslant \delta$, and is $\overleftarrow{d}$ if $max + Sim(q_x, q_y) < \delta$;
11: end while
12: return $Sim(q_x, q_y) \geqslant \delta$ or $Sim(q_x, q_y) < \delta$.
```

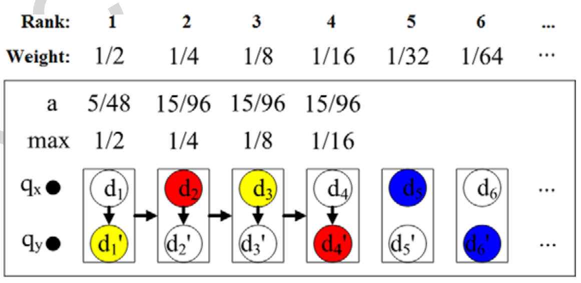


Fig. 2. Level-first traversal ($\delta = 0.25$).

First, we can easily apply sequence-first or level-first traversal method to the "Hierarchical Agglomerative Clustering" algorithm with a specified similarity threshold [10]. We replace the process of query similarity computation in the hierarchical agglomerative clustering by Algorithm 1 or 2. Then, the new algorithms become more efficient as demonstrated in Section 7.

Second, if we formulate a similarity matrix by completely computing the transition similarity for every pair of queries, as a similarity/distance metric, our transition similarity can be equally well adapted in most of the traditional query clustering methods (e.g., hierarchical agglomerative clustering, K-Means/Medoids clustering). To validate this, we also compare the performance of our transition similarity metric with the existing similarity/distance metrics such as Kendall's tau [22], Jaccard similarity, Cosine

AUTHOR COPY

similarity and Euclidean distance under the same experimental setup in Section 7.

Indeed, our early termination (sequence-first or level-first traversal) strategies can be also applied to other similarity metrics (besides the transition similarity) in hierarchical agglomerative clustering based on either top-k search results or query logs. By traversing the URLs in two top-k lists, partial Jaccard similarity, partial Cosine similarity, or partial Euclidean distance can be derived and compared with the threshold to terminate the similarity/distance computation earlier than the prior work.

6. Query clustering evaluation

To validate that search engine query clustering with top ranked search results outperforms that of click-through query logs, we need effective cluster evaluation measures to compare the quality of the clustering results. In this section, we define some measures to evaluate the clustering quality along two dimensions: 1) the internal measure – considering query clustering as a generic clustering problem, evaluate the cohesion and separation of the clustered queries using silhouette coefficient [41], and 2) external measures – since clustered queries represent users' search intents when they use search engine, we also define precision, recall and F-measure for evaluating the accuracy of query clustering based on human search behavior.

6.1. Internal measure: Cohesion and separation of clusters

Silhouette coefficient is a composite measure that judges the clustering quality by the *cohesion and separation* of the clustered objects. In query clustering, we regard every distinct query as a point in the highly-dimensional space and measure the distances between points in the same cluster and the distances between clusters. Referring the definition in [28,41], we can let Dis_1 be the average distance between queries to their centroid query in corresponding clusters, and let Dis_2 be the minimum distance between every pair of centroid queries (the query holding the greatest global similarity to all other queries in the same cluster is chosen as the centroid query). We thus have:

$$\widehat{SC} = \frac{Dis_2 - Dis_1}{\max\{Dis_1, Dis_2\}} \tag{6}$$

where $\widehat{SC} \in [-1, 1]$ and higher value means better quality of clustering. Since $\widehat{SC}$ is derived based on distances measured among the clustered objects, it can be calculated based on any underlying distance metric such as Euclidean distance, the distance metric derived from the Jaccard similarity, Cosine similarity or Transition similarity where distance $||q_x - q_y|| = 1 - Sim(q_x, q_y)$.

6.2. External measure: Accuracy of query clustering

Besides the internal measure (e.g., $\widehat{SC}$), we also evaluate the accuracy of query clustering using some external measures, including Precision, Recall and F-Measure based on the behavioral study on Internet users' search intents. Ideally, the external measures can be applied after capturing the benchmark query clusters for a given set of queries, where queries in the benchmark query clusters are the most similar. Then, with the benchmark query clusters, the scores of Precision, Recall and F-measure (external measures) can be calculated for all the query clustering results derived from different features and/or similarity metrics with the same input – a set of queries.

Therefore, we conducted a similar evaluation as Wen et al. [45] – randomly selecting a subset of query clusters from the query clustering result and manually checking whether the queries in the same cluster are indeed similar or not. More details of our evaluation methodology are given as below:

1. In our external measure evaluations, we compare the clustering accuracy of our proposed approach (*Transition similarity and feature top-k search results*) with other four previously studied similarity metrics (Kendall's tau, Jaccard similarity, Cosine similarity and Euclidean distance) and the corresponding features.
   For each of the five metrics, we select each of two different features (top-k search results and click-through query logs) to derive a query clustering result (viz. grouped queries) with the same agglomerative clustering algorithm [10], input set of queries and parameters. Then, we apply the external measures to compare the accuracy of the above 10 different query clustering results (denoted by $CR_1, \ldots, CR_{10}$), so as to justify the effectiveness of Transition similarity (applied to search queries' top-k search results).
2. We recruited 20 undergraduate students on campus to determine whether two queries are seman-

AUTHOR COPY

tically similar or dissimilar, and whether some query clusters in different clustering results refer to the same search intent or not. Considering the university students as generic Internet users, we invite students from a wide variety of majors to participate in this study (e.g., information technology, nursing, supply chain management, finance, and accounting). As sophomores, juniors or seniors in college, they are supposed to possess the basic computer & Internet skills and have a general understanding of linguistics.

3. In the empirical study, we randomly select 100 query clusters from the first query clustering result (w.l.o.g., derived by the Transition similarity & top-k search results). Then, we generate the benchmark query clusters for these 100 query clusters from 10 different clustering results $CR_1, \ldots, CR_{10}$.
   More specifically, for each query cluster in CR_1 (e.g., "yahoo map, google map"), each of 20 students independently identifies another 9 similar query clusters from the remaining 9 clustering results (e.g., "yahoo map, maps.com, online map", "yahoo map, ask.com"). Notice that the most similar query cluster (to "yahoo map, google map") in each of the 9 other clustering results is decided by all 20 students together: for instance, every student selects the most similar query cluster (to "yahoo map, google map") from the 2nd clustering results CR_2, and the majority of the 20 selected query clusters is "yahoo map, maps.com, online map"; 20 students do this again in the $CR_3, \ldots, CR_{10}$ to generate 10 different query clusters (close to "yahoo map, maps.com, online map") respectively.
   We consider the union of 10 different groups of queries for each of the 100 query clusters as the benchmark query cluster (e.g., "yahoo map, google map, maps.com, online map, ask.com, ..."). Then, the new 100 query clusters (benchmark query clusters) are slightly expanded compared to the original 100 query clusters randomly selected from the first query clustering result, since each of them is a union of 10 similar query clusters. Also, the students manually remove outliers (e.g., "ask.com") in the above benchmark query clusters. Again, all the outliers are decided by all the 20 students – any query is considered as an outlier if at least half of the students (10 students) think so, then it will be removed in the benchmark query clusters.
4. For the corresponding 100 query clusters in each of the 10 clustering results $CR_1, \ldots, CR_{10}$, we then compute the scores of Precision, Recall, and F-Measure using the benchmark query clusters. More specifically, we have:
   (a) Precision: for every clustering result, the ratio of the number of similar queries out of the total number of queries in all 100 clusters, where similar queries can be defined in the benchmark query clusters.
   (b) Recall: for every clustering result, the ratio of the number of similar queries out of the total number of all similar queries in the benchmark query clusters.
   (c) F-Measure: a combination of precision and recall ($2 \cdot \frac{precision \cdot recall}{precision + recall}$).

7. Experimental results

7.1. Experimental setup

Datasets. In our experiments, we make use of search queries' two features (top-k search results and click-through query logs) for search engine query cluster analysis. Since we should cluster the same set of queries for comparing the clustering results of top-k search results versus click-through query logs, we randomly pick 50,000 distinct queries $\mathcal{Q}$ from the AOL query log [9,25] and collect the top-10 search results for $\mathcal{Q}$ from "University Research Program for Google Search" [2]. We send the set of queries to Google and the ranked search results are returned in XML format. Clearly, all of the search engines can internally query themselves to get the results.[1] The characteristics of the datasets are shown in Table 2.

[1]Note that the AOL query log data was collected in 2006. Unfortunately in our experiments, it is extremely challenging to retrieve a big set of queries' full lists of top-k search results in 2006. Instead we collected a set of queries' top-k search results from another source when we design the experiments, where the queries in two collections of data are identical. Since the top-k search results collected within two different time frames might be different, possible bias may occur in the experimental evaluation. To justify that the bias could be insignificant, we further extracted a small set ($\sim$1000) of queries' top-10 search results from the AOL data (collecting such data requires that all the top-k URLs of the queries should be clicked by AOL users in 2006, and thus constrains the data size), and compared the top-k URLs of those queries (retrieved from the AOL dataset) with our experimental data. Each query's two top ranked search results are quite similar. Due to the small size of such data, we did not conduct cluster analysis on it.

AUTHOR COPY

Table 2
Characteristics of The datasets

	AOL	AOL ($\mathcal{Q}$)	Top-10 ($\mathcal{Q}$)
# queries (with clicks)	1,864,860	50,000	50,000
# dist. queries (with clicks)	583,084	50,000	50,000
# dist. query and URL pairs	1,190,491	107,761	500,000
# dist. URLs	373,837	56,904	215,083

We fix the query set by picking the same subsets of queries from $\mathcal{Q}$ for testing all the metrics (e.g., 10000 distinct queries), and cluster these queries using their click-through data or top-k search results. In addition, we select a separate set of queries from $\mathcal{Q}$ to tune the parameters of our algorithms in the experiments, where the sets of queries as well as their features used for testing and tuning are disjointed sets.

Parameters. We let $\omega(i) = \frac{1}{2^i}$ (the weight sequence $\omega(i)$ is also tuned by comparing the clustering performance of using other common sequences e.g., $\omega(i) = \frac{1}{i}$ and $\omega(i) = \frac{1}{3^i}$) and normalize the transition similarity for Ω where i represents the numeric item rank number of each URL. The similarity/distance value of all the metrics are normalized into [0, 1]. Furthermore, we select k from the range [1, 10] for top-k search results in additional parameter tuning using a separate dataset (Section 7.5). In other tests, all the results are based on top-5 search results. We test the clustering results by varying the similarity threshold δ in $\{0.1, 0.2, 0.3, \ldots, 0.7\}$ for all metrics.

Experimental Platform. All the experiments are performed on a Dell machine with Intel Core i7-3770 3.4 GHz CPU and 8 GB RAM running Windows 7 Professional.

AUTHOR COPY

7.2. *Internal measure evaluation: Cohesion and separation*

On one hand, since any pair of queries' top-k search results are two ranked sets of URLs, we can apply not only our transition similarity or Kendall's tau, but also some traditional metrics (e.g., Cosine similarity, Jaccard similarity, and Euclidean distance) to cluster search queries using their top-k search results. On the other hand, each query's clicked URLs in the click-through query log data can be ranked in terms of their clicked counts, then we can extract a list of top-k URLs for each query from the query logs. Thus, not only the traditional metrics (e.g., Cosine similarity, Jaccard similarity, and Euclidean distance) but also the transition similarity and Kendall's tau can be applied to cluster search queries using their query logs. In summary, all five different metrics can be applied to two different features for query clustering analysis.

Hence, we compare the clustering results generated by 5 similarity/distance metrics and search queries' 2 features – top-k search results and query logs, such as Transition similarity & top-k search results, Kendall's tau & top-k search results, and Cosine similarity & query logs. To simplify notation, we denote the metrics as "TS", "KT", "Cos", "JS" and "ED", and utilize "S" and "C" represent top-k search results and click-through query logs respectively. Then, 10 different clustering results can be generated: "TS-S", "KT-S", "Cos-S", "JS-S", "ED-S", "TS-C", "KT-C", "Cos-C", "JS-C", and "ED-C".[2]

One point is worth noting that, since we have to specify a minimum similarity threshold δ for the agglomerative clustering algorithm based on *different similarity/distance metrics*, the clustering results are not directly comparable even if they use an identical δ value. This is because, TS-S/C, KT-S/C, Cos-S/C, JS-S/C and ED-S/C, are defined in different ways so that the same δ value may stand for different levels of minimum similarity threshold. For example, 0.1 is computed with 10 different formulas even though all of them are normalized to [0, 1]. To justify this, we plot a simple illustrative example's normalized similarity scores, measured by different similarity/distance metrics in Fig. 3. In this illustrative example, we have (1) each of two queries has 100 URLs; the number of common URLs varies from 0 to 100, (2) we assume that the common URLs' positions on two lists are identical, otherwise we cannot compare the Transition similarity with other metrics such as JS, Cos and ED, (3) under the above assumption, TS has two cases: the common URLs are ranked from top to bottom (denoted as TS-TopDown) or from bottom to top while increasing the number of common URLs from 0 to 100 (denoted as TS-BottomUp), and (4) we cannot simulate the similarity scores of KT since rank order changes are not applicable to JS, Cos and ED. In Fig. 3, we can find out that the same similarity score refers to completely different number of common URLs in two lists if measuring the similarity by different metrics.

[2]Note that we employ the hierarchical agglomerative clustering algorithm [10] to test the metrics and features for query clustering, and set $\delta = \{0.1, 0.2, 0.3, \ldots, 0.7\}$.

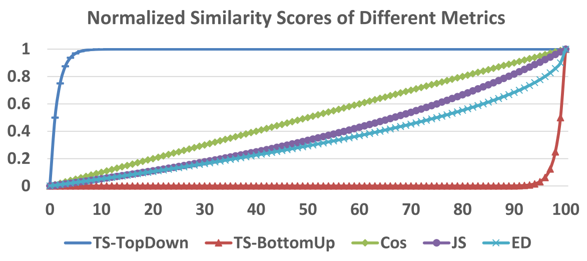


Fig. 3. Normalized Similarity Scores of Different Metrics.

Hence, for each of the 10 clustering results, we should capture the distribution of silhouette coefficient $\widehat{SC}$ with a varying $\delta = \{0.1, 0.2, 0.3, \ldots, 0.7\}$ and then 10 $\widehat{SC}$ curves become comparable since all the curves converge to 1.

Recall that the silhouette coefficient $\widehat{SC}$ is derived based on "distances" between queries. To make the $\widehat{SC}$ of the clustering results comparable, we have to compute $\widehat{SC}$ using the same distance/similarity metric for all the clustering results. In other words, when 10 combinations of metrics and features are used for generating 10 different clustering results, a metric and a feature should be chosen for the evaluation (computing $\widehat{SC}$). At this time, we again have 10 different combinations of the metrics and features used for the evaluation (measuring different versions of $\widehat{SC}$) respectively. In each subfigure of Figs 4 and 5, a distance metric and a feature are chosen to compute $\widehat{SC}$ in the evaluation, and then we can obtain 10 clustering results' $\widehat{SC}$ curves based on $\delta = \{0.1, 0.2, 0.3, \ldots, 0.7\}$.

Essentially the higher $\widehat{SC}$ curves are, the better the queries are separated (with large inter-cluster distances and smaller internal distances in the same cluster). Thus, we summarize the comparison of the results in Figs 4 and 5 in Table 3.[3]

Thus, we can draw the following conclusions for the comparison of all the query clustering results (by different metrics and/or features) evaluated by the internal measure cohesion and separation:

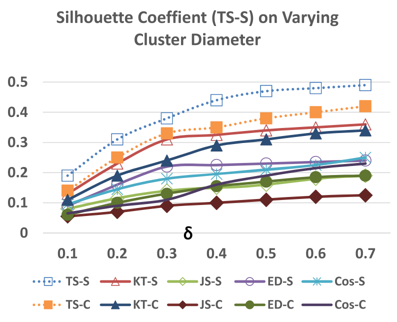
Table 3
Comparison of Different Metrics and Features for Query Clustering (10 Different $\widehat{SC}$ in the Evaluations)

$\widehat{SC}$ (10 Evaluation Metrics/Features)	Best	Second Best
TS-S based $\widehat{SC}$	TS-S	TS-C
KT-S based $\widehat{SC}$	TS-S, KT-S KT-C^3	–
JS-S based $\widehat{SC}$	JS-S	JS-C
ED-S based $\widehat{SC}$	ED-S	ED-C
Cos-S based $\widehat{SC}$	Cos-S	TS-S, Cos-C^3
TS-C based $\widehat{SC}$	TS-C	TS-S
KT-C based $\widehat{SC}$	KT-C	TS-C, KT-S^3
JS-C based $\widehat{SC}$	JS-C	KT-S
ED-C based $\widehat{SC}$	ED-C	TS-S, ED-S TS-C, KT-C^3
Cos-C based $\widehat{SC}$	Cos-C	TS-S, TS-C^3

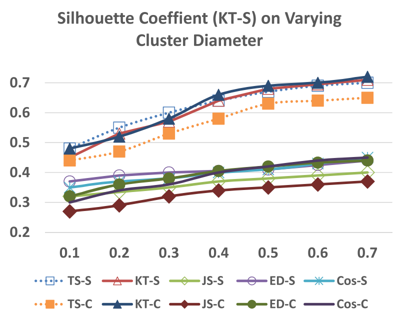
1. In all 10 cases ($\widehat{SC}$ values are computed by 10 different combinations of metrics and features), the same metric/feature selected both for clustering and evaluation produces the best result, as shown in 10 subfigures in Figs 4 and 5. This fact is true, because the clustering algorithm indeed tries to optimize the separation and cohesion based on the selected metric and feature, and the generated result should outperform other results if they are evaluated with such metric and feature.
2. Apart from the best result in each of the 10 cases, the second best results of the 10 cases are obtained as: TS-S 5 times, TS-C 4 times, KT-S 2 times, KT-C 2 times, Cos-C 1 time, JS-C 1 time, ED-C 1 time, ED-S 1 time (notice that in the second row we move TS-S, KT-C to the second best category, though they perform very close to the best result KT-S). Hence, our Transition similarity can produce good clustering results in most of the cases (using either top-k search results or click-through query logs).
3. All the $\widehat{SC}$ values are obtained by Eq. (6), though the distances in the formula are derived by 10 different combinations of the metrics and features in the evaluations. All the $\widehat{SC}$ values (regardless of the evaluation metrics/features) fall into the range $[-1, 1]$. However, the $\widehat{SC}$ values of the same clustering result obtained for the different evaluation metrics/features are incomparable: e.g., the $\widehat{SC}$ values of TS-S in 10 different subfigures are incomparable. This is because the

[3]In Table 3, two $\widehat{SC}$ curves in Figs 4 and 5 are ranked at the equivalent place if and only if the following two criteria are met: (1) they are intersected (if one curve outperforms the other curve for all $\delta = \{0.1, 0.2, 0.3, \ldots, 0.7\}$, then the higher $\widehat{SC}$ curve is ranked higher no matter how close the curves are). (2) for all $\delta = \{0.1, 0.2, 0.3, \ldots, 0.7\}$, the ratio of the maximum difference between the $\widehat{SC}$ values on two curves to the $\widehat{SC}$ values is less than 10%. Although the $\widehat{SC}$ values computed for two different clustering results (by different metrics and/or features, but the same δ) are incomparable, these two criteria can explicitly measure the closeness of two $\widehat{SC}$ curves in Figs 4 and 5.

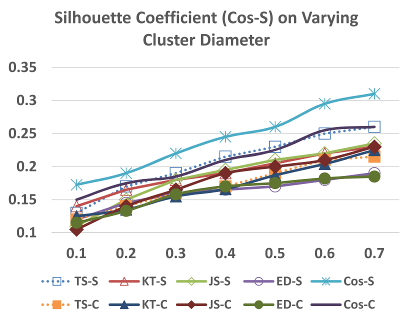
AUTHOR COPY



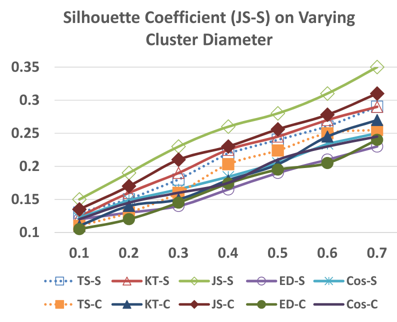
(a) TS-S based $\widehat{SC}$



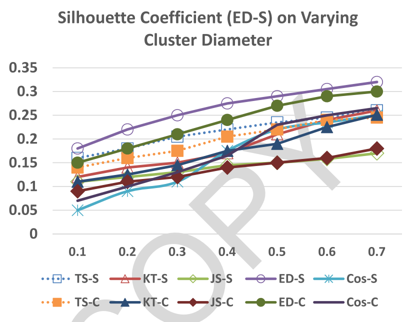
(b) KT-S based $\widehat{SC}$



(c) Cos-S based $\widehat{SC}$

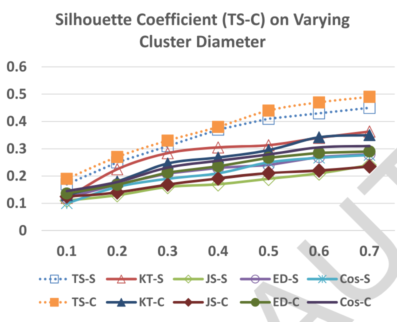


(d) JS-S based $\widehat{SC}$

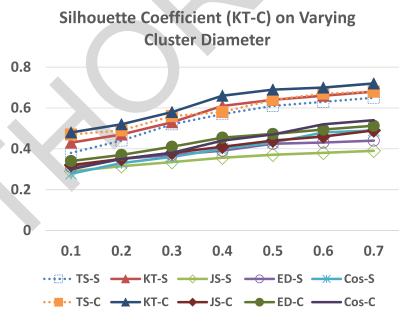


(e) ED-S based $\widehat{SC}$

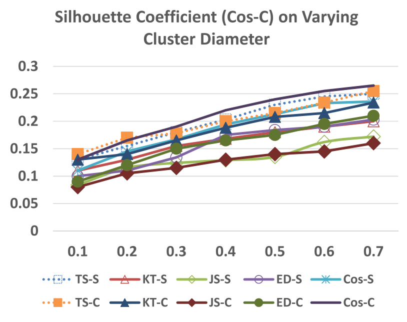
Fig. 4. Silhouette Coefficient $\widehat{SC}$ for Clustering Results of TS-S/C, KT-S/C, Cos-S/C, JS-S/C and ED-S/C – note that distances are measured by the *feature top-k results* and all metrics respectively; # distinct queries = 10,000 out of $\mathcal{Q}$ (50,000).



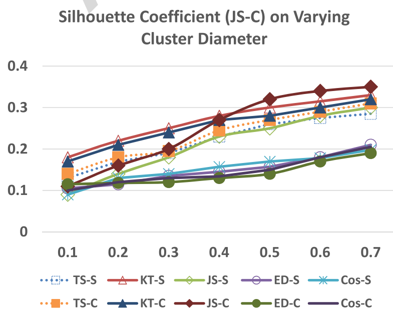
(a) TS-C based $\widehat{SC}$



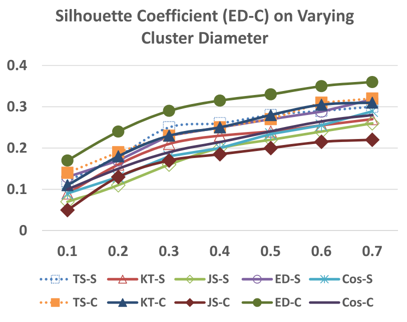
(b) KT-C based $\widehat{SC}$



(c) Cos-C based $\widehat{SC}$



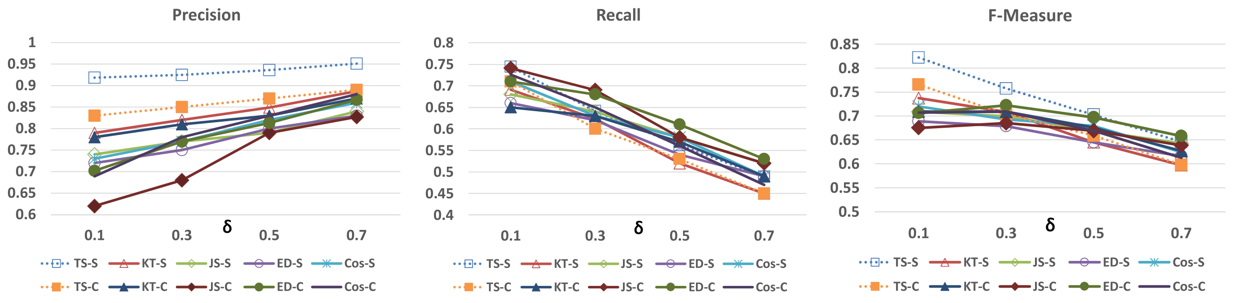
(d) JS-C based $\widehat{SC}$



(e) ED-C based $\widehat{SC}$

Fig. 5. Silhouette Coefficient $\widehat{SC}$ for Clustering Results of TS-S/C, KT-S/C, Cos-S/C, JS-S/C and ED-S/C – note that distances are measured by the *feature query logs* and all metrics respectively; # distinct queries = 10,000 out of $\mathcal{Q}$ (50,000).

AUTHOR COPY



(a) Precision (b) Recall (c) F-Measure

Fig. 6. Query Clustering Accuracy (External Measures).

$\widehat{SC}$ values are obtained using different formulas defined by different evaluation metrics.

4. In Figs 4 and 5 (10 subfigures), the average ranking of TS-S is the highest among all 10 $\widehat{SC}$ curves: 1st, 1st^3, 2nd^3, 4th, 3rd, 2nd, 4th, 2nd^3, ∼5th, 2nd^3. Based on the internal measure evaluation, Transition similarity and queries' top-k search results can generate the best clustering result for the hierarchical agglomerative clustering algorithm [10].

7.3. *External measure evaluation: Accuracy*

Recall that we have described our external measures and the validation setup in Section 6.2. For every similarity threshold $\delta \in \{0.1, 0.2, 0.3, \ldots, 0.7\}$, we calculate the Precision, Recall and F-measure for 10 clustering results (TS-S, KT-S, JS-S, ED-S, Cos-S, TS-C, KT-C, JS-C, ED-C, Cos-C) derived from two features respectively. Figure 6 presents the average Precision, Recall and F-measure for all the results with a similarity threshold δ varying from 0.1 to 0.7.

First, we examine the Precision scores of 10 clustering results. The clustering using top-k search results (especially the Transition similarity) gives more accurate semantically similar queries, since the average Precision is clearly greater than other clustering results. Since the Precision scores of the clustering results of KT-S is also better than that of TS-C, KT-C, JS-C, Cos-C and ED-C, we can conclude that clustering queries using top-k search results is more accurate than using query logs.

Second, as illustrated in Section 6.2, applying the Recall measure to compare the query clustering results is not always appropriate since the complete "benchmark clustering results" cannot be directly obtained. We assume that the similar queries from the union of all the query clustering results as the complete "benchmark clustering results", and plot the Recall scores in Fig. 6(b) for the same clustering results as Precision. We observe that the Recall scores of TS-S is slightly worse than JS-C and ED-C, better than KT-S and almost identical to the remaining results. Since the difference between them is minor, the performance is still tolerable.

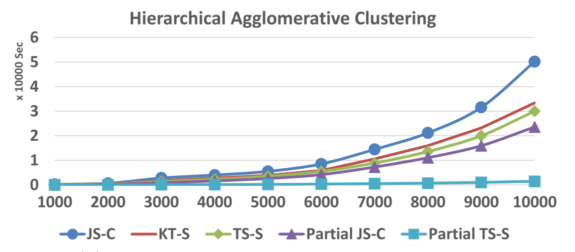
Finally, by combining the Precision and Recall together, we can observe that TS-S reaches the highest F-measure scores in almost all the cases. Thus, top-k search results and transition similarity can be utilized to effectively produce accurate clustering results in terms of the external measures (Precision, Recall and F-measure) as well.

7.4. *Scalability*

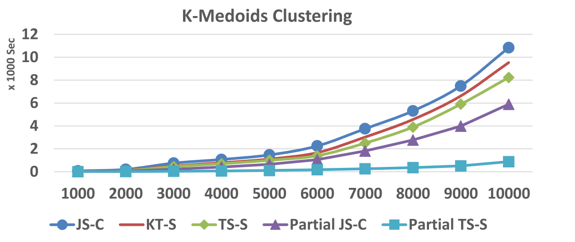
Since many unique URLs link to arbitrarily input queries and every unique URL represents one dimension in the computation, the *"curse of dimensionality"* problem [41] has attracted increasing attention in query clustering. Compared to the existing work, our partial TS based algorithms can significantly reduce the computational cost. We now present the experimental results on the efficiency and scalability.

Our sequence-first and level-first traversal algorithms are mainly based on the early termination strategy in similarity computation by discovering "Similarity Threshold URL" and "Dissimilarity Threshold URL". We applied a pair of partial TS based traversal approaches to the hierarchical agglomerative clustering, and name the new algorithms as "Partial TS (Sequence)" and "Partial TS (Level)" respectively. The computational costs of the "Partial TS (Sequence)" and "Partial TS (Level)" are very close in our tests ($k = 5$), thus we merge them and mark the results as "Partial TS-S" in Figs 7(a) and 7(b). Also, note that TS-S and the "Partial TS-S" (both Sequence-first and Level-first traversal) produce the same clustering results. In other words, two early termination strategies only improves

AUTHOR COPY

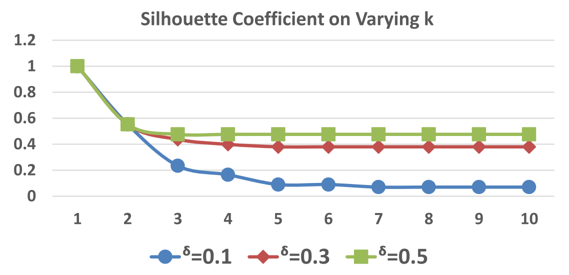


(a) Hierarchical Agglomerative Clustering

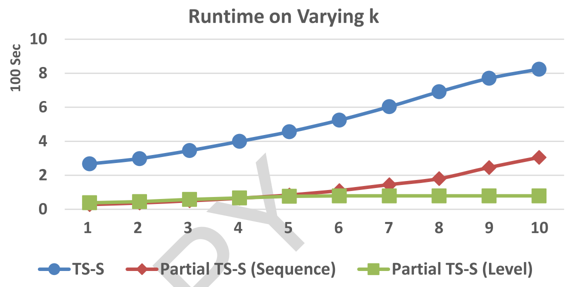


(b) K-Medoids Clustering

Fig. 7. Scalability on Varying Size of Datasets.



(a) $\widehat{SC}$ of TS-S or Partial TS-S



(b) Runtime of TS-S and Partial TS-S

Fig. 8. Tuning Parameter k (Top-k Search Results).

the efficiency and scalability, but do not change the clustering results.

Figure 7 shows the scalability of the above two algorithms – runtime on varying size of datasets, compared to the original hierarchical agglomerative clustering (Fig. 7(a)) and K-Medoids (Fig. 7(b)). Note that in K-Medoids clustering, K is selected as the average number of clusters derived from JS-C, ED-C, Cos-C, TS-S and KT-S under the same input setup. Since the required runtime of JS-C/S, Cos-C/S and ED-C/S are quite similar, we only plot JS-C as a representative in both Figs 7(a) and 7(b). Besides the early termination strategies applied to TS-S, we also tested the runtime of applying early termination strategies to JS-C. As shown in Figs 7(a) and 7(b), although the "Partial JS-C" outperforms the original JS-C per the computational cost, the "Partial TS-S" is more efficient and scalable than the "Partial JS-C".

In summary, clustering large number of search engine queries by either the original hierarchical agglomerative clustering or K-Medoids clustering algorithms is not the best choice. Instead, the "Partial TS-S (Sequence)" and "Partial TS-S (Level)" exhibit better efficiency and scalability than any other algorithms – Figs 7(a) and 7(b) show that a very slow linear runtime increase occurs as the dataset size increases for both of them.

7.5. *Parameter tuning*

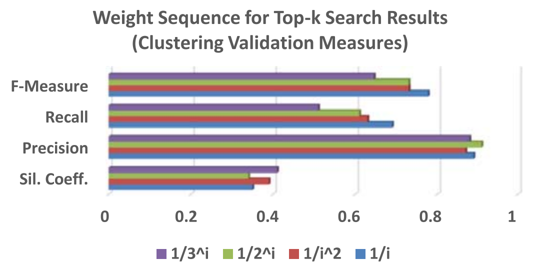
To cluster queries using our two partial TS based algorithms, we first specify an appropriate k for top-k URLs sequence $\mathcal{L}(k)$, a similarity threshold δ and a weight sequence for ranked URLs. Essentially, it is important to discuss how to select parameters with respect to the robustness of novel approaches. We now illustrate additional parameter tuning with regard to these parameters: k value of the top-k search results, similarity threshold δ and estimated weight sequence Ω for ranked URLs. Note that the set of queries and the corresponding feature (top-k search results) for parameter tuning are completely different from the testing data. Both of these two datasets are randomly selected from the 50,000 distinct queries, however the two datasets do not include any common query.

7.5.1. *k value and δ*

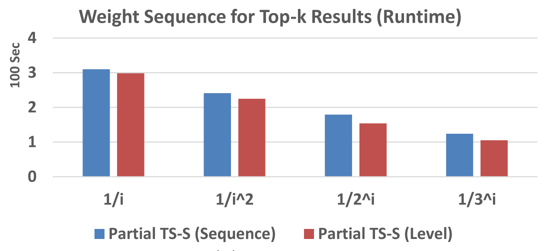
Fixing the estimated weight sequence $\Omega = \{\forall i \in [1,k], \omega_i = \frac{1}{2^i}\}$, we plot the $\widehat{SC}$ values on varying $k = \{1, 2, \ldots, 10\}$ and $\delta = \{0.1, 0.3, 0.5\}$ in Fig. 8(a). Given any normalized similarity threshold δ, $\widehat{SC}$ gets smaller when k increases, but quickly converges to a constant. For different similarity thresholds, $\widehat{SC}$ converges to different constants. Thus, to pursue more precise clusters, we can either set a small k or higher threshold δ.

As shown in Fig. 8(b), sequence-first traversal requires less runtime than level-first traversal if we choose a small k for top-k search results. If k is large, the runtime of level-first traversal converges to a constant but the runtime of sequence-first traversal shows a small increasing trend.

AUTHOR COPY



(a) Clustering Validation



(b) Runtime

Fig. 9. Tuning Different Weight Sequence.

7.5.2. *Normalized weight sequence*

The last question is how to determine weight sequence Ω. As we have discussed, $\omega(i)$ should decrease as i increases. For any arbitrary decreasing sequence, we can implement our partial TS based algorithms. Fixing $\delta = 0.3, k = 5$ (an average setup for parameters δ and k), we examined $\widehat{SC}$, Precision, Recall, F-measure, and the runtime of four weight sequences $\omega_i = \{\frac{1}{i}, \frac{1}{i^2}, \frac{1}{2^i}, \frac{1}{3^i}\}$ (as described in Section 4.3, we normalize every sequence before running this group of experiments). Note that the Precision, Recall, and F-measure are derived by the methods in Section 6.2.

The common advantage of these weight sequences is that the clustered queries can be extremely accurate (with high Precision as shown in Fig. 9(a)). For sharply decreasing weight sequences (i.e. $\omega_i = \frac{1}{3^i}$), the Recall, F-measure and runtime become lower (See Figs 9(a) and 9(b)). Based on these, we can select an appropriate weight sequence to meet different requirements.

8. Conclusion and future work

Commercial search engines collect Internet users' search queries from their aggregated query logs everyday. In order to make the search engine more convenient, they develop some applications such as providing accurate suggestion or recommendation for every posed query. Clustering search engine queries is one of the most important tasks built in those applications. Thus, search engines are always trying to generate more precise and multi-functional clusters of similar queries. However, most prevalent algorithms suffer from certain limitations on clustering this highly dimensional and diverse set of queries.

In this paper, we presented a novel feature for clustering queries – top ranked search results. It incorporates both content and click-through information: while ranking the search results by any robust search engine (e.g., Google, Yahoo! or Bing), the top ranked search results of every query would be quite relevant to the users' search intent. Then, we took advantage of this relevance to quantify the query similarity with a new similarity/distance metric, which facilitated us to develop more efficient algorithms for clustering queries.

To validate our approach, we conducted extensive experimental evaluation along several dimensions – quality of the clustering, computational scalability and parameter tuning. The results of quality of clustering were derived by both internal measures (i.e. cohesion and separation of the query clusters) and external measures (i.e. Precision, Recall and F-measure). We compared the clustering results of our method with several other state-of-the-art metrics and features. The experimental results demonstrated: 1) top-k lists produce query clusters with better cohesion and separation than the click-through query logs, 2) top-k lists generate more accurate query clusters than click-through query logs, 3) two early termination strategies significantly reduces the clustering runtime and scales well to large datasets, and 4) our approach is robust in terms of parameter tuning.

There are several directions for extending the work in the future. For example, if the condition permits, we can evaluate our approach with real-world search engine applications of query suggestion and recommendation. Also, we can look at other utilities of the top ranked search results, such as improving query expansion [31], web personalized application [35], query spelling correction [13] and/or query sanitization with privacy concern [27,29,30]. Finally, besides the search engine queries, we can investigate some other clustering problems in which the similarity between objects can be measured with given ranked lists, e.g., the online items sold at Amazon. We intend to explore all of these in the future.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant No. CNS-0746943 and

AUTHOR COPY

a Faculty Research Grant from University at Albany, SUNY. We also would like to thank the anonymous reviewers for their constructive comments and suggestions.

Appendix A. Metric proof for transition similarity

Proof. Since $Sim(q_x, q_y) = \sum_{i=1}^{m} \frac{[\omega(r_x(i))+\omega(r_y(i))]/2}{\Delta_{d_i}(q_x,q_y)+1}$ (assuming that $\omega(i)$ is normalized), first, we have $D(q_x, q_y) = 1 - \sum_{i=1}^{m} \frac{[\omega(r_x(i))+\omega(r_y(i))]/2}{\Delta_{d_i}(q_x,q_y)+1} \geqslant 0$. Second, if $D(q_x, q_y) = 0$, we have $\forall i$, $\Delta_{d_i}(q_x, q_y) = 0$. Hence, $\mathcal{L}_{q_x}(k) = \mathcal{L}_{q_y}(k)$ and vice versa. Third, since d_i is the common URL of q_x and q_y and we use $\omega(r_x(i)) + \omega(r_y(i))$ and $\Delta_{d_i}(q_x, q_y)$ for calculating $D(q_x, q_y)$, it is symmetric.

We now prove the triangle inequality. For q_x, q_y and q_z, we should prove $D(q_x, q_y) + D(q_y, q_z) \geqslant D(q_x, q_z) \Leftrightarrow Sim(q_x, q_y) + Sim(q_y, q_z) \leqslant 1 + Sim(q_x, q_z)$. Assuming that $S(xy - z)$ denotes the set of URLs in the lists of q_x and q_y but not q_z. Similarly, we define $S(yz - x)$, $S(xz - y)$ and $S(xyz)$. Since the partial similarities for $S(xyz)$ are different when we are comparing different pairs of queries, we use $Sim(S(xyz))_{xy}$ to represent the partial similarity between lists of q_x and q_y, so on and so forth. Hence, we have to prove $Sim(S(xy-z)) + Sim(S(yz-x)) + Sim(S(xyz))_{xy} + Sim(S(xyz))_{yz} \leqslant 1 + Sim(S(xz - y)) + Sim(S(xyz))_{xz}$. Suppose that $Sim(S(xy - z)) + Sim(S(yz-x)) + Sim(S(xyz))_{xy} + Sim(S(xyz))_{yz} > 1 + Sim(S(xz-y)) + Sim(S(xyz))_{xz}$. For $Sim(S(xy - z)) + Sim(S(yz - x)) \leqslant 1$, then $Sim(S(xyz))_{xy} + Sim(S(xyz))_{yz}$ must be greater than $Sim(S(xz-y)) + Sim(S(xyz))_{xz}$. Because $Sim(S(xyz))_{xy} + Sim(S(xyz))_{yz}$ is incomparable to $Sim(S(xyz))_{xz}$ (if a common URL of those three queries locates at high levels of q_x and q_z, but lower levels of q_y, we have $Sim(S(xyz))_{xy} + Sim(S(xyz))_{yz} < Sim(S(xyz))_{xz}$), which is a contradiction. Hence, $Sim(q_x, q_y) + Sim(q_y, q_z) \leqslant 1 + Sim(q_x, q_z)$ and $D(q_x, q_y) + D(q_y, q_z) \geqslant D(q_x, q_z)$.

Hence, the distance based on our transition similarity is a metric. □

Appendix B. Proof of Lemma 1

Proof. Let the ith URL on $\mathcal{L}_{q_x}(k)$ be d_i and $Sim(q_x, q_y)_i$ represent the partial transition similarity from d_i where $i \in [n + 1, k]$. We seek *max*, that is the sum of $\forall d_i, Sim(q_x, q_y)_i$, while the ranks of d_i can be any untraversed rank on $\mathcal{L}_{q_y}(k)$. We thus have $max = \max\{\sum_{i=n+1}^{k}\{Sim(q_x, q_y)_i\}\}$. Hence, the partial transition similarity derived from q_x and q_y's untraversed URLs is bounded by the objective value of this bipartite matching problem which is known to be $\mathcal{O}(V^2 \log(V) + VE)$ where the bipartite graph is represented by $G = \{V = (X, Y), E\}$ [36]. In our case, the number of vertices in $\mathcal{L}_{q_x}(k)$ is $k - n$ whereas the number of vertices in $\mathcal{L}_{q_y}(k)$ can be any number in $[k - n, k]$.

However, we can improve on the efficiency by computing a slack upper bound max' instead of the exact value *max*. Let r_y be the current highest rank of all untraversed URLs in $\mathcal{L}_{q_y}(k)$. Since $Sim(q_x, q_y)_i = \frac{\omega(i)+\omega(j)}{2(|i-j|+1)}$ where d_i is ranked at the jth rank of $\mathcal{L}_{q_y}(k)$, we have the local maximum partial transition similarity for d_i if $j = r_y$ or $j = i$ (according to Discussion 1). That is $\max\{\omega(i), \frac{\omega(i)+\omega(r_y)}{2(|i-r_y|+1)}\}$ where $r_y \leqslant i$ because at most $i - 1$ ranks in $\mathcal{L}_{q_y}(k)$ can be traversed prior to u_i. Therefore, $\forall i \in [n + 1, k]$, we have $Sim(q_x, q_y)_i \leqslant \max\{\omega(i), \frac{\omega(i)+\omega(r_y)}{2(|i-r_y|+1)}\}$ and $max' = \sum_{i=n+1}^{k} \max\{\omega(i), \frac{\omega(i)+\omega(r_y)}{2(|i-r_y|+1)}\}$. Since max' still decreases rapidly in the traversal due to the decreasing weight and a small number of common URLs, we can adopt $\sum_{i=n+1}^{k} \max\{\omega(i), \frac{\omega(i)+\omega(r_y)}{2(|i-r_y|+1)}\}$ to approximately setup upper bound for untraversed URLs with a time complexity $O(n)$. □

Discussion 1 (Similarity on Rank Difference). For a common URL d_i with a fixed position $r_x(i)$ on q_x's top-k list, we have $Sim(q_x, q_y)_i = \frac{\omega(r_x(i))+\omega(r_y(i))}{2(\Delta_{d_i}(q_x,q_y)+1)} = \frac{\omega(r_x(i))+\omega(r_y(i))}{2(|r_x(i)-r_y(i)|+1)}$. Without loss of generality, we let $r_x(i) \geqslant r_y(i)$, $s = r_x(i) - r_y(i)$ and $\omega(r_y(i)) = (1 + g(s))\omega(r_x(i))$ where $g(s)$ is a function with the variable $s \in [0, r_x(i) - 1]$. Hence, $Sim(q_x, q_y)_i = \omega(r_x(i)) \cdot \frac{1+g(s)}{2(s+1)}$. The derivative $\frac{d[\omega(r_x(i)) \cdot \frac{1+g(s)}{2(s+1)}]}{d_s} = \omega(r_x(i)) \cdot \frac{\frac{d}{d_s} g(s) \cdot 2(s+1) - 2g(s)}{4(s+1)^2}$.

If $\omega(i) = \frac{1}{2^i}$, we have $g(s) = 2^s$ and the derivative is $\omega(r_x(i)) \cdot \frac{\ln 2 \cdot 2^s(s+1) - 2^s}{2(s+1)^2}$. For $s \geqslant 1$, it is greater than 0, so $Sim(q_x, q_y)_i$ is monotonic on s and $Sim(q_x, q_y)_i$ is maximized as $s = r_x(i) - 1$. There exists $s \in (0, 1)$ such that the derivative is less than 0 while we have $Sim(q_x, q_y)_i = \omega(i)$ for $s = 0$. Hence, $\max\{Sim(q_x, q_y)_i\} = \max\{\omega(i), \frac{\omega(i)+\omega(r_y)}{2(i-r_y+1)}\}$ where $r_y = r_x(i) - \max\{d\}$.

For other weight sequences, we can also prove $\max\{Sim(q_x, q_y)_i\} = \max\{\omega(i), \frac{\omega(i)+\omega(r_y)}{2(i-r_y+1)}\}$ or

AUTHOR COPY

find alternative slack upper bound in similar discussion.

Appendix C. Proof of Lemma 2

Proof. To prove that the partial similarity is bounded, we first consider that all the URLs in lower levels where $i > n$ haven't been matched with URLs in higher levels $j \leqslant n$. Since all the higher level URLs in both sequences have been traversed by prior level-first route, the lower-level URL (d_i at the ith level of $\mathcal{L}_{q_x}(k)$) cannot be matched to higher-level URL (at the jth level of $\mathcal{L}_{q_y}(k)$). Similar to Lemma 1, we are seeking maximum weighted matching. Once all lower-level URLs of one query can only find matching URLs from positions lower than n in the other sequence, the match should be one-to-one and the number of vertices is no greater than $k - n$ in each side. The **maximum** partial transition similarity is $\sum_{i=n+1}^{k} \omega(i)$ when all the lower level URLs (greater than i where $i \in [n+1, k]$) have a one-to-one match in the same levels. (We can easily prove it by contradiction proof: suppose that there exist two pairs of URLs match at different levels (lower than i) satisfying $\sum_{i=n+1}^{k} Sim(q_x, q_y)_i > \sum_{i=n+1}^{k} \omega(i)$. Hence, there exists a pair of equal transition value $\Delta > 0$, thus we have $\frac{2(\omega(i)+\omega(j))}{2(\Delta+1)} > \omega(i)+\omega(j)$ which means $\Delta < 0$. This is a contradiction).

Now we consider that some URLs (ranked lower than n) have been traversed through the match of URLs in the higher levels of $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$. Then, the number of traversable URLs in lower levels of $\mathcal{L}_{q_x}(k)$ and $\mathcal{L}_{q_y}(k)$ is less than the **maximum** case above. Hence, the partial similarity is bounded by $\sum_{i=n+1}^{k} \omega(i)$. □

References

[1] http://www.worldwidewebsize.com/.

[2] http://research.google.com/university/search/.

[3] E. Agichtein, E. Brill and S.T. Dumais, Improving web search ranking by incorporating user behavior information, in: *Proc. of 29th Annual SIGIR Conference*, 2006, pp. 19–26.

[4] E. Agichtein, E. Brill, S.T. Dumais and R. Ragno, Learning user interaction models for predicting web search result preferences, in: *Proc. of 29th Annual SIGIR Conference*, 2006, pp. 3–10.

[5] I. Augenstein, A.L. Gentile, B. Norton, Z. Zhang and F. Ciravegna, Mapping keywords to linked data resources for automatic query expansion, in: *Proc. of the Second International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data*, 2013, pp. 9–20.

[6] R. Baeza-Yates, C. Hurtado and M. Mendoza, Query recommendation using query logs in search engines, in: *Proc. of 9th International Conference on Extending Database Technology Workshops*, 2004, pp. 588–596.

[7] R.A. Baeza-Yates, C.A. Hurtado and M. Mendoza, Improving search engines by query clustering, *Journal of the Association for Information Science and Technology* **58**(12) (2007), 1793–1804.

[8] R.A. Baeza-Yates and A. Tiberi, Extracting semantic relations from query logs, in: *Proc. of 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 76–85.

[9] M. Barbaro and T. Zeller Jr., A face is exposed for aol searcher no. 4417749, (New York Times), Augest 9, 2006.

[10] D. Beeferman and A.L. Berger, Agglomerative clustering of a search engine query log, in: *Proc. of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 407–416.

[11] G. Bordogna, A. Campi, G. Psaila and S. Ronchi, A language for manipulating clustered web documents results, in: *Proc. of ACM 17th Conference on Information and Knowledge Management*, 2008, pp. 23–32.

[12] 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: *Proc. of 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2008, pp. 875–883.

[13] Q. Chen, M. Li and M. Zhou, Improving query spelling correction using web search results, in: *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, ACM, 2007, pp. 181–189.

[14] W. Cheung and Y. Sun, Identifying a hierarchy of bipartite subgraphs for web site abstraction, *Web Intelligence and Agent Systems* **5**(3) (2007), 343–355.

[15] S.-L. Chuang and L.-F. Chien, Towards automatic generation of query taxonomy: A hierarchical query clustering approach, in: *Proc. of the 2002 IEEE International Conference on Data Mining*, 2002, pp. 75–82.

[16] N. Craswell and M. Szummer, Random walks on the click graph, in: *Proc. of 30th Annual SIGIR Conference*, ACM, 2007, pp. 239–246.

[17] H. Cui, J.-R. Wen, J.-Y. Nie and W.-Y. Ma, Query expansion by mining user logs, *IEEE Transactions on Knowledge and Data Engineering* **15**(4) (2003), 829–839.

[18] D.L. Davies and D.W. Bouldin, A cluster separation measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-1** (Nov. 1977), 224–227.

[19] T.E. Doszkocs and B.A. Rapp, Searching MEDLINE in English: A prototype user inter-face with natural language query, ranked output, and relevance feedback, in: *Proc. of the ASIS Annual Meeting*, 1979, pp. 131–139.

[20] G. Dupret and M. Mendoza, Automatic query recommendation using click-through data, in: *Proc. of IFIP 19th Professional Practice in Artificial Intelligence*, 2006, pp. 303–312.

[21] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: *Proc. of 2th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

AUTHOR COPY

[22] R. Fagin, R. Kumar and D. Sivakumar, Comparing top k lists, *SIAM Journal on Discrete Mathematics* **17**(1) (2003), 134–160.

[23] B.M. Fonseca, P.B. Golgher, E.S. de Moura and N. Ziviani, Using association rules to discover search engines related queries, in: *Proc. of the First Latin American Web Congress*, 2003, pp. 66–71.

[24] L. Fu, D.H.-L. Goh, S.S.-B. Foo and J.-C. Na, Collaborative querying through a hybrid query clustering approach, in: *Proc. of 6th International Conference on Asian Digital Libraries*, 2003, pp. 111–122.

[25] K. Hafner, Researchers yearn to use aol logs, but they hesitate, (New York Times), Augest 23, 2006.

[26] B. He, J. Huang and X. Zhou, Modeling term proximity for probabilistic information retrieval models, *Information Sciences* **181**(14) (2011), 3017–3031.

[27] Y. Hong, X. He, J. Vaidya, N.R. Adam and V. Atluri, Effective anonymization of query logs, in: *Proc. of ACM 18th Conference on Information and Knowledge Management*, 2009, pp. 1465–1468.

[28] Y. Hong, J. Vaidya and H. Lu, Search engine query clustering using top-k search results, in: *Proc. of the 10th IEEE/WIC/ACM International Conference on Web Intelligence*, 2011, pp. 112–119.

[29] Y. Hong, J. Vaidya, H. Lu, P. Karras and S. Goel, Collaborative search log sanitization: Toward differential privacy and boosted utility, *IEEE Transactions on Dependable and Secure Computing* **12**(5) (2015), 504–518.

[30] Y. Hong, J. Vaidya, H. Lu and M. Wu, Differentially private search log sanitization with optimal output utility, in: *Proc. of 15th International Conference on Extending Database Technology*, 2012, pp. 50–61.

[31] J. Huang, J. Miao and B. He, High performance query expansion using adaptive co-training, *Information Processing & Management* **49**(2) (2013), 441–453.

[32] U. Irmak, V. von Brzeski and R. Kraft, Contextual ranking of keywords using click data, in: *Proc. of the 25th International Conference on Data Engineering*, 2009, pp. 457–468.

[33] T. Joachims, Optimizing search engines using clickthrough data, in: *Proc. of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2002, pp. 133–142.

[34] K.W.-T. Leung, W. Ng and D.L. Lee, Personalized concept-based clustering of search engine queries, *IEEE Transactions on Knowledge and Data Engineering* **20**(11) (2008), 1505–1518.

[35] B. Mobasher, Data mining for web personalization, in: *The Adaptive Web*, 2007, pp. 90–135.

[36] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[37] F. Radlinski and T. Joachims, Query chains: Learning to rank from implicit feedback, in: *Proc. of 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005, pp. 239–248.

[38] S. Saitta, B. Raphael and I.F.C. Smith, A bounded index for cluster validity, in: *Proc. of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2007, pp. 174–187.

[39] X. Shi and C.C. Yang, Mining related queries from search engine query logs, in: *Proc. of the 15th International World Wide Web Conference*, 2006, pp. 943–944.

[40] F. Silvestri, Mining query logs: Turning search usage data into knowledge, *Foundations and Trends in Information Retrieval* (2010).

[41] P. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[42] T. Wakaki, H. Itakura, M. Tamura, H. Motoda and T. Washio, A study on rough set-aided feature selection for automatic web-page classification, *Web Intelligence and Agent Systems* **4**(4) (2006), 431–441.

[43] S. Walter, C. Unger, P. Cimiano and D. Bar, Evaluation of a layered approach to question answering over linked data, in: *Proc. of the 11th International Conference on the Semantic Web – Volume Part II*, 2012, pp. 362–374.

[44] J.-R. Wen, J.-Y. Nie and H.-J. Zhang, Clustering user queries of a search engine, in: *Proc. of the 10th International World Wide Web Conference*, 2001, pp. 162–168.

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

[46] J. Yi and F. Maghoul, Query clustering using click-through graph, in: *Proc. of the 18th International World Wide Web Conference*, 2009, pp. 1055–1056.

[47] E. Yilmaz, J.A. Aslam and S. Robertson, A new rank correlation coefficient for information retrieval, in: *Proc. of 31th Annual SIGIR Conference*, 2008, pp. 587–594.

AUTHOR COPY