Top-K Search Query Grouping using SOM Clustering for Search Engine

Sami Uddin
Computer Science and Engineering
VNS Group of Institute
Bhopal, India

Amit Kumar Nandanwar
Computer Science and Engineering
VNS Group of Institute
Bhopal, India

ABSTRACT
Clustering is important task for any recommendation system. Clustering method suggested by many researchers for search engine optimization. Search engine help user for better searching by user's query recommendation. Clustering is helpful for finding actual relation between different queries which are not same as they seems. But do clustering of user query is also a difficult task because of user enters lots of type and varying queries. Many time these queries may very short to get their real meaning and also can generate different meanings. Any single query may have various meaning on other hand many different query words may have common meaning for searching contents. Lots of clustering methods are given in last decades for search engine optimization but these methods unable to proper utilization various information hidden in user query log. This paper gives a novel clustering approach based on to identify query similarity and apply SOM clustering for effective clustering results. We propose a novel similarity matrix for user queries by uses of URL clicked by user trough searching results. Text similarity and time similarity are also measure for calculating similarity between two queries. This method shows good results within clustering performance to compare with other existing methods.

1. INTRODUCTION
Clustering of program queries has attracted vital attention in recent years. Several program applications like query recommendation need query agglomeration as a pre-requisite to operate properly. Indeed, agglomeration is critical to unlock verity price of query logs. However, agglomeration search queries effectively are kind of difficult, owing to the high diversity and capricious input by users. Search queries are sometimes short and ambiguous in terms of user necessities. Many alternative queries might confer with one construct, whereas one query might cowl several ideas. Existing current agglomeration ways, like K-Means or DBSCAN cannot assure sensible leads to such a various atmosphere. Clustered agglomeration offers sensible results however is computationally quite pricey. This paper presents a unique agglomeration approach supported a key insight – program results may themselves be wont to determine query similarity. This work proposes query matter similarity and time thresholds for a lot of strong approach that leverages search query logs. This can be wont to develop a awfully economical and correct formula for agglomeration queries. This system can useful for varied search engines and search applications like query suggestions, result ranking, query alterations, sessionization, and cooperative search.

With the event in data technology, the net [1] has clad to be a huge data repository covering virtually each space, within which an individual's user may be concerned. In spite of recent advances in net program technologies, there are still several things within which user is bestowed with unwanted and non- relevant pages within the high most results of the hierarchal list. Program usually has difficulties in forming a pithy and precise illustration of the response pages appreciates a user query. Providing a group of websites supported user query words isn't an enormous drawback in program. The problem arises at the user finish as he must sift through the long result list, to search out his desired content. This drawback is remarked as data Overkill drawback [2].

The design [3] of the program is shown in Figure1.

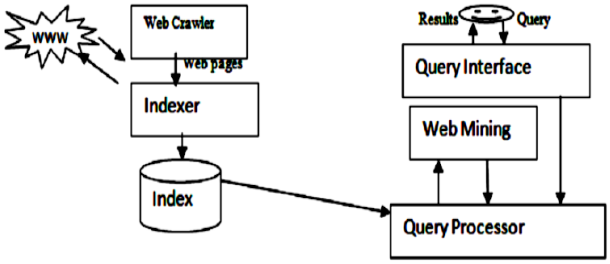

Fig. 1. Architecture of Search Engine

There are three parts in programmed referred to as Crawler, skilled worker and Ranking mechanism. The crawler is additionally known as mechanism that navigates the online and downloads the online pages. The downloaded pages area unit transferred to associate classification module and erect the index supported the keywords in individual pages. once a query is being floated by a user, it means that the query transferred in terms of keywords on the interface of a quest engine, the query mainframe section examine the query keywords with the index and precedes the URL's of the pages to the shopper. However gifting the pages to the shopper a ranking mechanism is completed by the search engines to present the foremost relevant pages at the highest and fewer vital pages at all-time low.

As of nowadays, the indexed net contains a minimum of thirty billion pages [3]. In fact, the general net could comprise over one trillion distinctive URLs, a lot of and a lot of that is being indexed by search engines each day. Out of this quagmire of knowledge, users usually rummage around for the relevant info that they require by move search queries to go looking engines. The matter that the search engines face is that the queries area unit terribly various and sometimes quite imprecise and/or ambiguous in terms of user needs. Many various queries could ask one thought, whereas one query could correspond to several ideas. To arrange and convey some order to the current huge unstructured dataset, search engines cluster these queries to cluster similar things

along. To extend usability, most business search engines, like Google, Yahoo!, Bing, and raise additionally augment their search facility through extra services like query recommendation or query suggestion. These services create it a lot of convenient for users to issue queries and acquire correct results from the programmed, and so area unit quite valuable. From the programmed perspective, effective clump of search queries could be a necessary pre-requisite for these services to perform well. As a result of all of those reasons, clump of program queries has attracted vital attention in recent years. However, existing current clump strategies, like K-Means or DBSCAN cannot assure smart ends up in such a various atmosphere. There area unit many challenges expose by the distinctive nature of the atmosphere. The first issue is to work out the way to live similarity between queries. To change a lot of precise info retrieval, a representative and correct descriptor is indispensable for computing the similarity between queries.

The thought of query similarity was originally utilized in info retrieval studies [4]: measurement the similarity between the content-based keywords of 2 queries. However, the matter with victimization this within the query log atmosphere is that users' search interests aren't continuously identical although the issued queries contain identical keywords. for example, the keyword "Apple" could represent a well-liked quite fruit whereas it's additionally the keyword of a well-liked company "Apple Iraqi National Congress.". Hence, the utilization of content-based keywords descriptor is very restricted for this purpose.

1.1 Query Logs

The log keeps user's queries and their clicks further as their browsing activities. The standard logs [5] of program embody the subsequent entries:

1) User IDs
2) Query Q issued by the user
3) Address u clicked by the user
4) Rank r of the address u clicked for the query Q
5) Time t at that the query has been submitted

The information contained in query logs may be utilized in many ways [6, 7], example to produce context throughout search, to classify queries. Query log is shown in Table one.

Table 1: Query Logs

User Id	Query Clicked	URL	r	Time
Admin	Data Mining	www.dming.com	6	12:10
Admin	Data ware housing	www.dming.com	5	8:30
Admin	Data Mining	www.google.com	5	11:10

In this paper, we tend to survey the prevailing strategies for computing query recommendations. We tend to prohibit the scope of this survey to strategies that, given a user's query, use it or rework it into another query, with a supposed intercalary price for the user's exploration. We tend to propose a proper definition of this downside, specifically to visualize the advice of queries for exploration functions as a recommending perform A survey of query recommendation techniques for exploration taking as input: The query log, a specific query session known as the present session, a user profile, a instance, associated an expectation perform. Given these parameters this recommending perform outputs a group of suggested queries, every with a given rating indicating the interest of the query for the present session.

Subsequently, to live similarity between 2 queries, the query illustration of a vector of URLs in a very clickthrough bipartite graph [8][9] has been adopted. even so, despite however giant the query log information set is, it's doable that the entire search intent of some queries might not be adequately portrayed by the obtainable click-through info. for example, in a very specific large-scale query log, there is also no clicked address for the query "Honda vs Toyota". Therefore, although it's clearly relevant to the query "Honda", on the idea of this click-through information, there's no similarity. Therefore, existing query log information isn't correct enough for analyzing users' search intent, particularly for those queries with none clicked address. One more reason that causes quality is that the query log information comprises users' click-through info in a very specific amount, whereas search interests may even modification over time. If we tend to utilize associate aggregative query logs collected in a very long amount to check and cluster queries, the accuracy is also wedged.

2. RELATED WORK

Therefore, describing queries solely by content-based keywords or strictly through click-through knowledge isn't forever correct for program query clump. During this paper, our main contribution is to propose a completely unique query descriptor for scrutiny queries. this is often supported a key insight – program results may themselves be wont to establish query similarity; and so incorporate each content and click on through info [11][12][13][14]. Supported this, we tend to additionally outline a replacement similarity metric which will be utilized in any distance primarily based clump algorithmic rule. owing to the variety of queries and therefore the curse of spatiality, current clump algorithms have high machine price. we tend to additionally propose AN economical clump algorithmic rule to scale back machine price. we tend to compare the query clump results of our approach with many existing state of the art strategies and show that our algorithmic rule provides smart cohesion, separation on clustered queries and considerably reduced runtime.

Query clump has its roots in keywords-based info retrieval analysis [4]. Since most of the keywords area unit ambiguous, analyzing the content of query keywords or phrases by ancient info retrieval techniques has several limitations. Following that, click-through query logs are mined to yield similar queries [20]. Beeferman and Berger [8] 1st introduced the agglomerate clump methodology to get similar queries mistreatment query logs however with limitations (noise and little variety of common clicks). The query clump approach adopted in

[21] uses a K-Means clump approach. K-Means algorithmic rule cannot adapt well in query clump case owing to the issue on specifying k. Wen et al. [22] analyzed each query contents and clickthrough bipartite graph and applied a density-based algorithmic rule DBSCAN [23] to cluster similar queries. The same as agglomerate query clump, DBSCAN algorithmic rule adopted in [22] needs high computation price. Meanwhile, Wen et al. linearly mix measures on content-based similarity and cross-references primarily based similarity however it's tough to line parameters for linear combination of 2 similarity metrics. However, our hierarchical search results knowledge (enforced by [11][12][13][14]) naturally contemplate each factors. Moreover, we've got compared the accuracy of our approach with the geometer distance primarily based query log clump [10]. Moreover, Kendall's letter [16] and a few relevant measures [24] are often effective in mensuration the accuracy of clustered queries. Since our most vital contribution is in observant the very fact that search results are often wont to perform query clump, they could be effective metrics additionally. However, since search queries clump may be a acknowledge exhausting downside, there's no economical algorithmic rule that has been projected for clump queries with reference to existing top-k lists scrutiny measures.

Finally, tend to use profile constant to live the cohesion and separation of query clump results. Larger profile constant indicates giant inter-cluster distances and little intra-cluster distances. Davies-Bouldin index [25][26] seeks identical objective on clump validation. the excellence between silhouette constant is that – decreased index generates the simplest clump results.

3. INFORMATION USED FOR QUERY PROCESS

Although search and browse log knowledge offer nice opportunities for enhancing internet search, there are many challenges before such knowledge are often utilized in varied applications. First, the scale of log knowledge is sometimes terribly giant. In observe the scale of search and browse log knowledge at an exploration engine is usually at the magnitude of tens of terabytes day after day. Second, log knowledge area unit quite shouting. For instance, queries could also be issued by machines for experiments; user input in computer address boxes could also be redirected to look engines by internet browsers; and clicks on search result pages could also be arbitrarily created by users.

To overcome noise and volume, one will combination raw log knowledge in preprocessing. By summarizing common patterns in data, the scale of information is often greatly reduced. Moreover, when aggregation, we tend to could prune patterns with low frequencies to scale back noise.

One query is a way to summarize raw log knowledge for varied log mining tasks. In fact, search and browse log knowledge have terribly complicated knowledge structures with varied sorts of knowledge objects and relationships. the info objects could embody users, sessions, queries, search result pages, clicks on search results, and follow-up clicks. These differing types of information objects kind a hierarchy. At the highest level, every user incorporates a series of sessions, wherever every session contains a sequence of queries. In a query, a user could open many WebPages. Finally, a user could additional follow the hyperlinks within the WebPages of search results and browse additional WebPages. Additionally to the hierarchal relationship between differing types of information objects, the info objects at identical level typically kind a successive relationship. Here, we tend to introduce four sorts of knowledge account that area unit wide utilized in log mining, namely, query histograms, click-through bipartite, click patterns, and session patterns. Among the literature reviewed during this survey, ninetieth of the papers on log mining utilized a minimum of one in all the four sorts of knowledge account.

3.1 Query Bar Graph

A query bar graph represents the quantity of times every query is submitted to an exploration engine. As shown in Figure three, query bar graph contains query strings and their frequencies. As an easy statistics, query bar graph are often utilized in a good form of applications, like query motor vehicle completion and query suggestion.

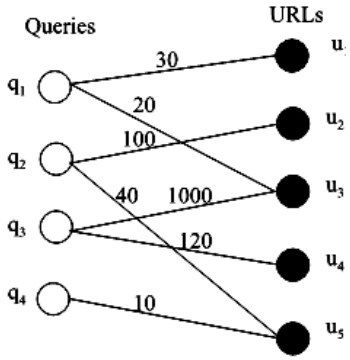

Fig 2. An example of click-through bipartite graph. In a click-through bipartite graph, nodes represent queries and URLs, and edges represent click relations between queries and URLs

3.2 Click-through Bipartite

A click-through bipartite graph, like Figure a pair of, summarizes click relations between queries and URLs in searches. The bipartite graph consists of a collection of query nodes and a collection of computer address nodes. A query and a computer address area unit connected by a position if the computer address is clicked by a user once it's came back as a solution to the query. A weight cij could also be related to AN edge eij, indicating the full variety of times URL uj is clicked with reference to query qi. Click-through bipartite is maybe the foremost wide used organization in log mining. As we are going to see within the following sections, it are often used for query transformation, query classification, document annotation, and plenty of alternative tasks.

3.3 Click Patterns

Click patterns summarize positions of clicked URLs in search results of queries. To be specific, every search result (also called search impression) Iq with respect to query q are often portrayed by Iq=(q;L), wherever L may be a list of triples (u,p,c), wherever u is that the universal resource locator of a page, p is that the position of the page, and c indicates whether or not the page is clicked. The identical search results are more collective to 1 click pattern Pq =(q;L;cc), wherever cc is that the variety of search results, samples of click patterns. In follow, an

inventory L solely includes the highest N URLs. Compared with a click-through bipartite, click patterns contain richer info. A click-through bipartite solely represents collective clicks of URLs, whereas click patterns more represent the positions of the clicked URLs furthermore as unclicked URLs. As are going to be seen within the later sections, click patterns will facilitate several tasks in search, like classifying steering and informational queries, learning pairwise document preference, building ordered click models, and predicting user satisfaction.

3.4 Session Patterns

Session patterns summarize transitions among queries, clicks, and browses at intervals search sessions. In fact, session patterns are often outlined in several ways in which looking on specific applications. As an example, sequences of queries as sessions and extract frequent query sequences as session patterns. In different cases, session patterns might involve not solely queries however additionally clicked URLs. as an example, outlined session patterns supported sequences of queries and their clicked URLs. Since session patterns represent users' search behaviors in an exceedingly additional precise means, it's been used extensively. As are going to be seen later, session patterns are wide utilized in tasks like query transformation, document ranking, and user satisfaction prediction.

4. QUERY SIMILARITIES FOR SEARCH LOGS

We currently develop the machinery to outline the query connection supported internet search logs. Our live of connection is geared toward capturing two necessary properties of relevant queries, namely:

1) Queries that regularly seem along as reformulations and

2) Queries that have evoked the users to click on similar sets of pages.

4.1 Search Behaviour

We derive three kinds of graphs from the search logs of an advert computer programmed. The query reformulation graph, QRG, represents the connection between a combine of queries that area unit doubtless reformulations of every different. The query click graph, QCG, represents the connection between 2 queries that regularly cause clicks on similar URLs. The query fusion graph, QFG, merges the data within the previous 2 graphs. All 3 graphs area unit outlined over a similar set of vertices VQ, consisting of queries that seem in a minimum of one among the graphs, however their edges area unit outlined otherwise.

4.1.1 Query Reformulation Similarity

One way to spot relevant queries is to think about query reformulations that area unit usually found inside the query logs of an exploration engine. If two queries that area unit issued consecutively by several users occur oftentimes enough, they're doubtless to be reformulations of every different. to live the connection between two queries issued by a user, the time-based metric, simtime, makes use of the interval between the timestamps of the queries inside the user's search history. In distinction, our approach is outlined by the applied math frequency with that two queries seem next to every different within the entire query log, over all of the users of the system.

To this end, based on the query logs, we construct the query reformulation Similarity, QRS=(VQ,EQR), whose set of edges, EQR, are constructed as follows: for each query pair (qi,qj), where qi is issued before qj within a user's day of activity, we count the number of such occurrences across all users' daily activities in the query logs, denoted countr(qi,qj). Assuming infrequent query pairs are not good reformulations of each other, we filter out infrequent pairs and include only the query pairs whose counts exceed a threshold value,r. For each (qi, qj) with countr(qi, qj)>=Tr, we add a directed edge from qi to qj to EQR. The edge weight, wr(qi,qj), is defined as the normalized count of the query transitions

$$w_r(q_i, q_j) = \frac{\text{count}_r(q_i, q_j)}{\sum_{(q_i,q_k)\epsilon\varepsilon_{QR}} \text{count}(q_i q_k)}$$

4.1.2 Query Top-K search Similarity

The key procedure for clustering queries is measuring the similarity or distance between all queries. Since the top-klist along with the weight sequence Ωgives a topk ranked list, we could use theKendall's tau [38] as the underlying distance metric. However, Kendall's tau is not that effective if the top-klists have little overlap, which is quite often for search engine query URL lists. Additionally, scalability is an essential concern for search engine query clustering. Computing Kendall's tau is not very efficient in large number of top-k lists' comparisons, which is necessary in clustering queries. To improve this, we propose a new similarity measure for comparing top-k lists. With top-k lists, the similarity between two different queries qx and qy can be measured through their two sorted URL sets Lqx (k) and Lqy (k). First, we identify the common URLs in Lqx (k) and Lqy (k). Since the maximum number of common URLs is limited tok, we denote these common URLs as{∀di ∈Lqx (k)∩Lqy (k) wherei ∈[1,m] and m≤k}. However, simply comparing the two lists through their common URLs such as by using the Jaccard coefficient [38], or Euclidean distance is not sufficiently accurate. The similarity between two queries is determined not only in terms of the number of common URLs, but also according to the ranks of each common URL in two top-k lists. For instance, "Apple" and "Apples" have many common URLs in the search results. If we do not consider the influence resulting from the positions of each URL, the similarity between these two queries might be very large. However, the first few URLs of "Apple" are related to "Apple Inc." and fruit related URLs are ranked not so high, whereas the search results of "Apples" have an inverse order of these two meanings. Actually, they are two totally different searches. Hence, the ranks of the same URL in both queries' topk lists impact the precision of the similarity as well. As illustrated in Section II-A, a common URL in Lqx(k) and Lqy(k)should have different relevance weight according to their ranks. Hence, our similarity measure should consider two factors: the relevance weight in two top-k lists and the "rank transition" (the rank difference in two queries' top k lists). We thus define the Transition Similarity between two queries qx and qy as follows:

$$Sim(q_x, q_y) = \frac{1}{2} \sum_{i=1}^{m} \frac{\omega[r_x(i)] + \omega[r_y(i)]}{|r_x(i) - r_y(i)| + 1} \quad (1)$$

Here, m is the number of common URLs in Lqx (k)and Lqy (k), rx (i) and ry (i) represent the rank of common URL di in Lqx(k) and Lqy (k), respectively, and |rx(i)−ry(i)| denotes the rank difference of di. Since|rx(i)−ry(i)| might be0, and has an anti-monotonic relationship with the similarity, we use the inverse value of|rx(i)−ry(i)|+1to measure the similarity w.r.t. rank difference.

4.1.3 Text Similarity

On a different note, we may assume that two query groups are similar if their queries are textually similar. Textual similarity between two sets of words can be measured by metrics such as the fraction of overlapping words or characters. We can thus define the following two text based relevance metrics that can be used in place of sim.

Simtext(sc,si) is defined as the fraction of common words between qc and qi as follows:

$$sim(s_c, s_i) = \frac{|words(q_c) \cap words(q_i)|}{|words(q_c) \cup words(q_i)|}$$

4.2 Final Query Similarity

The Final Query Similarity, the query click graph calculated by reformation similarity, top-k search similarity and text similarity. As follows:

$$w_f(q_i, q_j) = a \times w_r(q_i, q_j) + b \times w_c(q_i, q_j) + c \times w_{txt}(q_i, q_j)$$

The relative contribution of the two weights is controlled by a, and we denote a query final similarity constructed with a particular value of a, b and c such as a+b+c =1 for FQS.

4.3 Proposed Algorithm

Fig 3 explains proposed work flowchart and overall proposed algorithm in sequential manner may be given as follow.

Algorithm for Search Engine Query Clustering Using SOM

Input:

The queries Dataset containing the current query id, time, rank and click urls.

A set of existing query groups S={s1,…., sm}

A similarity threshold th, 0<th<1

Output: The query group s that best matches Sc, or a new one if necessary

Step 1: Set the initial parameters

Let the users, S={s1, s2, s3, ….., sn}

Current query and clicks, {qc,clkc}.

Weight of query Time Similarity =wr

Weight of Query Top-K=wc

Weight of Text similarity graph=wtxt

a =0.33, b =0.33, c= 0.33

Step 2: load Query log data set.

Step 3: Feature Extraction of time , query click , URLs etc .

Simtime(sc,si)=1/time (qc)-time(qi)

Step 4: Creation of Query Reformulation similarity.

Weight is calculated as

{

$$w_r(q_i, q_j) = \frac{count_r(q_i, q_j)}{\sum_{(q_i,q_k)\epsilon\varepsilon_{QR}} count(q_i q_k)}$$

Step 5: Creation of Query Top-K by click url.

Weight is calculated as

$$w_c(q_i, q_j) = \frac{\sum_{u_k} \min(count_c(q_i, u_k), count_c(q_j, u_k))}{\sum_{u_k} count_c(q_j, u_k)}$$

Step 6: Creation of Text similarity

Weight is calculated as

$$w_{txt}(q_i, q_j) = \frac{words(q_i) \cap words(q_j)}{words(q_i) \cup words(q_j)}$$

Step 7: Calculate the weights for final similarity

$$w_f(q_i, q_j) = a \times w_r(q_i, q_j) + b \times w_c(q_i, q_j) + c \times w_{txt}(q_i, q_j)$$

}

Step 8: this similarity matrix passed to SOM tool for k-means clustering, clustering method results output similar query groups as one cluster.

Step 9: Get the final groups accuracy and comparison.

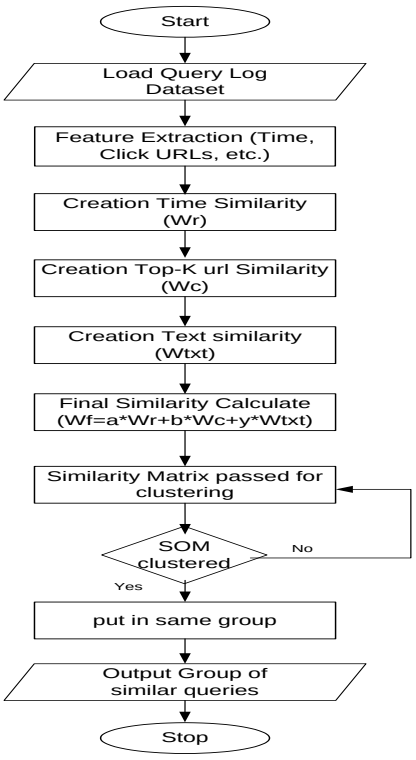

Fig 3. Proposed work flow diagram

5. EXPERIMENT RESULTS

In this section, we study the behavior and performance of our algorithms on partitioning a user's query history into one or more groups of related queries. For example, for the sequence of queries "caribbean cruise"; "bank of america"; "expedia"; "financial statement", we would expect two output partitions: first, {"caribbean cruise," "expedia"} pertaining to travel-related queries, and, second, {"bank of america," "financial statement"} pertaining to money-related queries.

5.1 Dataset

To this finish, we tend to obtained the query reformulation and query click graphs by merging variety of monthly search logs from a billboard computer program. every monthly photo of the query log adds around twenty four % new nodes and edges within the graph compared to the specifically preceding monthly photo, whereas around ninety two % of the mass of the graph is obtained by merging 9 monthly snapshots. to cut back the result of noise and outliers, we tend to cropped the query reformulation graph by keeping solely query pairs that appeared a minimum of double (Tq ¼2), and therefore the query click graph by keeping solely query-click edges that had a minimum of ten clicks (Tc ¼10).

In order to make check cases for our algorithms, we tend to used the search activity (comprising a minimum of 2 queries) of a collection of two hundred users (henceforth known as theRand200data set) from our search log. to get this set, users were picked every which way from our logs, associated 2 human labelers examined their queries and appointed them to either an existing group or a brand new group if the labelers deemed that no connected group was gift. A user's queries were enclosed in theRand200 knowledge set if each labelers were in agreement so as to cut back bias and subjectiveness whereas grouping. The labelers were allowed access to the online so as to see if 2 ostensibly distant queries were truly connected (e.g., "dainik bhaskar " and "star news"). the common variety of groups within the knowledge set was three.84 with thirty % of the users having queries classified in additional than three groups.

5.2 Performance Metric

To measure the standard of the output groupings, for every user, we tend to begin by computing query pairs within the labeled and output groupings. Two queries kind a try if they belong to an equivalent cluster, with lone queries pairing with a special "null" question. To judge the performance of our algorithms against the groupings made by the labelers, we'll use the Rand Index [35][39] metric, that could be a ordinarily used live of similarity between 2 partitions. The Rand Index similarity between 2 partitions X,Y of n components every is outlined as RandIndex(X,Y)=(a+b)/n, wherever a is that the range of pairs that square measure within the same set in X and also the same set in Y, and b is that the range of pairs that square measure in several sets in x and in several sets in Y. Higher RandIndex values indicate higher ability of grouping connected queries along for a given formula. Our formula to realize the most effective performance onRand200 supported the RandIndex metric. We tend to follow an equivalent approach for the baselines that we tend to enforced also. We'll additionally appraise the approaches on further take a look at sets (Lo100, Me100, and Hi100). to achieve a live of usage data for a given user, we glance at the common outdegree of the user's queries (average outdegree), also because the average counts among the outgoing links (average weight) within the query reformulation graph. so as to check the results of usage data on the performance of our algorithms, we tend to created 3 further take a look at sets of a hundred users every. The sets we tend tore additionally manually labeled as we delineated. The primary set, Lo100contains the search activity of a hundred users, with average out-degree < 5. Similarly, Me100 contains user activity for users having out-degree >5 but <10 end Hi100 having out-degree >10.

Based on these information sets, we tend to judge once more the performance of our algorithms and that we show the ends up in rock bottom 3 lines of Table one. As we will see from the table, for QFG, subsets with higher usage info conjointly tend to own higher RandIndex values [39][40][41]. Hi100 (RandIndex=0.89) performs higher than Me100(RandIndex=0.947), that successively outperforms Lo100(RandIndex=1.0). ATSP shows the same trend (higher usage shows higher performance) and it outperforms QFG at the Lo100data set. CoR's performance is additional or less similar for the various information sets that is anticipated because it doesn't use the graphs directly. For Jaccard, it's best once the property round the queries among a user's session is comparatively low. We tend to don't observe any important distinction within the performance of the opposite baselines (Time and Levenshte in) in these new information sets.

TABLE 2. Comparative Performance of Our Methods

	Levenshtein	Jaccard	CoR	ATSP	QFG	Proposed
Rand200	0.721	0.750	0.807	0.831	0.860	0.867
Lo100	0.732	0.762	0.794	0.832	0.821	1.000
Me100	0.712	0.748	0.802	0.857	0.868	0.947
Hi100	0.729	0.742	0.809	0.871	0.882	0.890

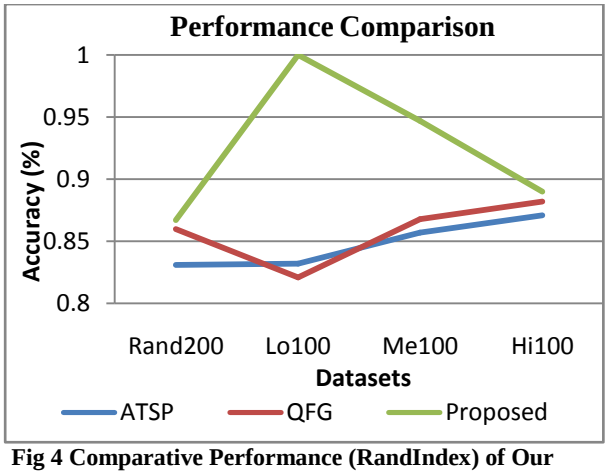

Fig 4 Comparative Performance (RandIndex) of Our Methods

Figure 4 and table 2 show comparative analyses between proposed work and various methods. Figures 4 shows result comparison of performance metric for Rand2oo dataset with other existing algorithms and its verifies that

proposed method outperformed better than others. Proposed method implemented reformation graph, click graph as well as association query information, hence Gives better results. The proposed method also compare with other dataset low100, mid100 and hi100 dataset as explain above for all dataset proposed method gives high performance index for grouping queries.

6. CONCLUSION

The final similarity between any two queries is calculated by taking advantages of text similarity, time similarity and clicked urls similarity. Hence lots of hidden relation between two queries can be determining by proposed method which is very help to clustering queries efficiently. This paper show how this hidden information may use for clustering users' search log effectively and used for their recommendation system. This method may be useful for search engine optimization and may be applying for web based clustering approaches. This work may extend in future research such as document clustering, image clustering, and web page recommendation. This work is only limited to query recommendation which may be extends till document and other file type recommendation by combining other similarity methods like cosine similarity or any other supervised learning.

7. REFERENCES

[1] Dr. G. K. Gupta, "Introduction to Data Mining with Case Studies", PHI Publication, 2005.

[2] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, Pang-Ning Tan, "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data", SIGKDD Explorations, Vol. 1, No. 2, 2000, Page 12-23.

[3] Adel T. Rahmani and B. Hoda Helmi, "EIN-WUM an AIS-based Algorithm for Web Usage Mining", Proceedings of GECCO'08, Atlanta, Georgia, USA, ACM978-1-60558-130-9/08/07, 2008, Pp. 291-292.

[4] Shailey Minocha, Nicola Millard, Lisa Dawson, "Integrating Customer Relationship Management Strategies in (B2C) E-Commerce Environments", IFIP Conference on Human-Computer Interaction-INTERACT, 2003.

[5] C. Ramya, G. Kavitha, K. S. Shreedhara, "Preprocessing: A Prerequisite for Discovering Patterns in Web Usage Mining Process", Computing Research Repository - CORR, vol. abs/1105.0, 2011.

[6] V. Chitraa, Antony Selvdoss Davamani, "A Survey on Preprocessing Methods for Web Usage Data", Computing Research Repository-CORR, Vol. abs/1004.1, 2010. Nizar R. Mabroukeh, Christie I. Ezeife, "A taxonomy of sequential pattern mining algorithms", ACM Computing Surveys - CSUR, Vol. 43, No. 1, 2010, Pp. 1-41.

[7] Francesco Moscato, Nicola Mazzocca, Valeria Vittorini, Giusy Di Lorenzo, Paola Mosca, Massimo Magaldi, "Workflow Pattern Analysis in Web Services", High Performance Computing and Communications - HPCC, 2005, Pp. 395-400.

[8] Heasoo Hwang, Hady W. Lauw, Lise Getoor, and Alexandros Ntoulas, "Organizing User Search Histories", IEEE Transactions On Knowledge And Data Engineering, Vol. 24, NO. 5, IEEE, 2012, Page 912-925.

[9] R. Jones and K.L. Klinkner, "Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs," Proc. 17th ACM Conf. Information and Knowledge Management (CIKM), 2008.

[10] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The Query-Flow Graph: Model and Applications," Proc. 17th ACM Conf. Information and Knowledge Management (CIKM), 2008.

[11] P. Anick, "Using Terminological Feedback for Web Search Refinement: A Log-Based Study," Proc. 26th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, 2003.

[12] B.J. Jansen, A. Spink, C. Blakely, and S. Koshman, "Defining a Session on Web Search Engines: Research Articles," J. the Am. Soc. for Information Science and Technology, vol. 58, no. 6, pp. 862-871, 2007.

[13] L.D. Catledge and J.E. Pitkow, "Characterizing Browsing Strategies in the World-Wide Web," Computer Networks and ISDN Systems, vol. 27, no. 6, 1995, pp. 1065-1073.

[14] D. He, A. Goker, and D.J. Harper, "Combining Evidence for Automatic Web Session Identification," Information Processing and Management, vol. 38, no. 5, 2002, pp. 727-742.

[15] R. Jones and F. Diaz, "Temporal Profiles of Queries," ACM Trans. Information Systems, vol. 25, no. 3, 2007, p. 14.

[16] A.L. Montgomery and C. Faloutsos, "Identifying Web Browsing Trends and Patterns," Computer, vol. 34, no. 7, July 2001, pp. 94-95.

[17] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz, "Analysis of a Very Large Web Search Engine Query Log," SIGIR Forum, vol. 33, no. 1, 1999, pp. 6-12.

[18] H.C. Ozmutlu and F. C¸ avdur, "Application of Automatic Topic Identification on Excite Web Search Engine Data Logs," Information Processing and Management, vol. 41, no. 5, 2005, pp. 1243-1262.

[19] T. Lau and E. Horvitz, "Patterns of Search: Analyzing and Modeling Web Query Refinement," Proc. Seventh Int'l Conf. User Modeling (UM), 1999.

[20] F. Radlinski and T. Joachims, "Query Chains: Learning to Rank from Implicit Feedback," Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD), 2005.

[21] J. Yi and F. Maghoul, "Query Clustering Using Click-through Graph," Proc. the 18th Int'l Conf. World Wide Web (WWW '09), 2009.

[22] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy, "Clustering Query Refinements by User Intent,"

Proc. the 19th Int'l Conf. World Wide Web (WWW '10), 2010.

[23] T. Radecki, "Output Ranking Methodology for Document- Clustering-Based Boolean Retrieval Systems," Proc. Eighth Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, 1985, pp. 70-76.

[24] V.R. Lesser, "A Modified Two-Level Search Algorithm Using Request Clustering," Report No. ISR-11 to the Nat'l Science Foundation, Section 7, Dept. of Computer Science, Cornell Univ., 1966.

[25] R. Baeza-Yates, "Graphs from Search Engine Queries," Proc. 33rd Conf. Current Trends in Theory and Practice of Computer Science (SOFSEM), vol. 4362, pp. 1-8, 2007.

[26] K. Collins-Thompson and J. Callan, "Query Expansion Using Random Walk Models," Proc. 14th ACM Int'l Conf. Information and Knowledge Management (CIKM), 2005.

[27] N. Craswell and M. Szummer, "Random Walks on the Click Graph," Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07), 2007.

[28] Spink, M. Park, B.J. Jansen, and J. Pedersen, "Multitasking during Web Search sessions," Information Processing and Management, vol. 42, no. 1, pp. 264-275, 2006

[29] D. Beeferman and A. Berger, "Agglomerative Clustering of a Search Engine Query Log," Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2000.

[30] R. Baeza-Yates and A. Tiberi, "Extracting Semantic Relations from Query Logs," Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2007.

[31] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The Query-Flow Graph: Model and Applications," Proc. 17th ACM Conf. Information and Knowledge Management (CIKM),2008

[32] Lecture Notes in Data Mining,M. Berry, and M. Browne, eds. World Scientific Publishing Company, 2006.

[33] V.I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals,"Soviet Physics Doklady,vol. 10, pp. 707-710, 1966

[34] Fuxman, P. Tsaparas, K. Achan, and R. Agrawal, "Using the Wisdom of the Crowds for Keyword Generation" Proc. the 17th Int'l Conf. World Wide Web (WWW '08),2008.

[35] W.M. Rand, "Objective Criteria for the Evaluation of Clustering Methods" J. the Am. Statistical Assoc.,vol. 66, no. 336, pp. 846-850, 1971.

[36] Spink, M. Park, B.J. Jansen, and J. Pedersen, "Multitasking during Web Search Sessions,"Information Processing and Management,vol. 42, no. 1, pp. 264-275, 2006.

[37] R. Baeza-Yates and A. Tiberi, "Extracting Semantic Relations from Query Logs,"Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD),2007.

[38] Yuan Hong, Jaideep Vaidya and Haibing Lu, "Search Engine Query Clustering using Top-k Search Results", IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, IEEE, 2011.

[39] Tahira Tabassum, Amit Dubey, "User Search Query Grouping using Association Fusion Graph", International Journal of Advanced Research in Computer Science and Software Engineering, Page 259-267, Volume 4, Issue 4, April 2014.

[40] Heasoo Hwang, Hady W. Lauw, Lise Getoor, Alexandros Ntoulas, "Organizing User Search Histories", IEEE Transactions on Knowledge & Data Engineering, vol.24, no. 5, pp. 912-925, May 2012.

[41] J. Re.ddy Susmitha & K. Srinivasa Rao, "Systematize Online Query Search With Application Interface", IJAEA, Vol-3 Issue-1, PP 13-17, 2010.