Content Free Clustering for Search Engine Query Log

Mehdi Hosseini
Sharif University of Technology
Web Intelligence Research Laboratory
Azadi Avenuse , Tehran
IRAN

Hassan Abolhassani
Sharif University of Technology
Web Intelligence Research Laboratory
Azadi Avenuse , Tehran
IRAN

Mohsn Sayyadi Harikandeh
Sharif University of Technology
Computer Engineering Department
Azadi Avenuse, Tehran
IRAN

Abstract: Web query clustering is widely used by web information systems. In this paper we present a new content free method for web query log clustering. Query clustering has many applications including page ranking in web search, personalizing search result and web query expansion. In our approach, we first construct a bipartite graph for queries and visited URLs of a query log. Most of the clusters of queries are connected together with noisy users selections. So some huge connected components are produced. To eliminate such noisy links all queries and related URLs are projected in reduced dimensional space by applying singular value decomposition. Finally, a clustering algorithm will be applied in each pruned connected component, in new space. The method has been evaluated using a real world data set and by comparing it to existing approaches, the results show promising improvements.

Key–Words: Query Log, Web Query Clustering, Dimension Reduction.

1 Introduction

As web contents grow, the importance of search engines becomes more critical and at the same time users satisfaction decreases. To improve search results, query log analysis is applied to better find out the users' search demands. Each search engine has a repository of query log which can be used to facilitate understanding of users' needs [1]. Our aim is to propose new query clustering method that is appropriate for noisy query log domain. Query log clustering is important for different purposes:

- **Improvement of search results:** when a query is submitted, the results of queries in the same cluster with the entered query that are clicked through can be used to improve search results.
- **Query expansion:** as the researches shows that average query terms are near two [9]. So most of the time, queries are ambiguous. One possible remedy for this problem is to expand a query with new terms. Query clustering helps to find relevant terms for this expansion which can be applied in the two ways:(1) Query expansion by terms of similar queries [3]. (2) Expansion by the terms of selected pages of similar queries [2] .
- **Personalization of search results:** here the goal is to post process the search results and display different sets to different users based on their attributes. By clustering user queries one can find a user's interests and demands. In [4] a profile for each user is created based on the past queries entered by them.
- **Query recommendation:** this method helps users to submit appropriate query to a search engine. With query clustering it is possible to find such appropriate queries. For example, [5] reports a method for this purpose. This strategy is different from expanding queries terms because the expansion method construct artificial query, while the query recommendation gives actual related queries formulated by other users that had the same information in the past.

In this paper a content-free query clustering method is introduced. In it, a bipartite graph for a query log is created where the left nodes are queries, right nodes show the selected URLs, and an undirected link connects a query to the URLs visited for that query. Using a fast algorithm that is described later, connected components of the graph are recognized. Since the formations of such connected components are highly sensitive to the noisy selections of users, Singular Value Decomposition (SVD) [12] is used to reduce noise inside a connected component. Finally to create main clusters a k-mean [13] algorithm is applied for each component. Therefore contribution of this paper is: new method for clustering queries in reduced dimensions space which decreases noise influence. In what follows, we discuss on related works in section 2 and introduce our method in detail in section 3. Evaluation of the method is discussed in

section 4. Finally section 5 presents conclusion.

2 Related Works

Query clustering methods can be divided in two groups: those that make clusters based on contents and those that do the clustering according to the relationships between queries and URLs. However [9] states that there is another work which uses a combination of the above two methods for this task.

2.1 Clustering by content

In these methods similarity between queries are calculated based on terms appearing in them. Such similarity is computed by different methods:

- **Keyword based**: after elimination of stopwords, each query is represented as a vector of terms and a vector similarity measure such as cosin silmilarity measure is applied [7].
- **Natural language processing based**: here NLP techniques are employed is used to find a query similar to the user query. Unlike the keyword based method that eliminates terms like what, where and when, in NLP based methods all of the query terms are used. However, this method is usually applied in Question-Answering systems where a query is in the form of a question sentence [8].
- **String comparisons**: in these methods measures like edit distance [2] is used to find a query near the user query having most similarity with it in terms of number of insertions, deletions and replacement of letters which is needed to convert user query to that. The main usage of such methods is spell correction of user query.

2.2 Clustering by Query-URL

Most search engines record the click through data in their query log. Each item of such a data is of the following form:

$$< \textit{IP Address, Query, URL, Time} > \qquad (1)$$

In this format each record contains following information:

- IP address of user machine
- Query of the user
- Link clicked by user (URL)
- Time of the query

Such click through data can be regarded as Implicit Feedback. When a user selects a limited number of URLs among a large number of results and such a selection is repeated by other users, one can infer that there should be a semantic similarity between the selected links. On the other hand when for some queries similar URLs are selected that means that those queries are similar even if they are not similar in lexical terms. One example is given in [9]. Queries like Atomic, Manhattan Project, Nagasaki and Nuclear Weapon Bomb are not similar in lexical forms but similar links are selected for them. A major advantage of this approach is the ability to cluster pages as well as queries without any need to have costly processing of contents of the pages. It is shown in [1] that the users' feedbacks are generally very useful. Therefore one of the important advantages of this method is discovery of semantic relations that can be used to find synonymy and polysemy of queries. For the first time in [6] a bipartite graph for synchronous clustering of links and queries were used where clustering is done in an iterative approach. In each iteration two queries having maximum similarity are put in a cluster. Then based on the similarity of queries two URLs having maximum similarity are put in a cluster. Such steps are iterated and finally each connected component is recognized as a cluster. The time complexity for each iteration is of $\theta(V^2)$ that V equals to all number of nodes in bipartite graph. The number of iterations has depended to the number of nodes, so total complexity of this algorithm is close to $\theta(V^3)$. The similarity of queries are calculated by equation (2) where $\mathrm{N}(q_k)$ represents the set of URLs selected for query q_k.

$$Sim(q_1,q_2) = \begin{cases} \frac{(N(q_1)\cap N(q_2)}{N(q_1)\cup N(q_2)} & if N(q_1)\cup N(q_2) > 0 \\ 0 & else \end{cases} \qquad (2)$$

In [11] the similarity measure of [6] (After this we refer to [6] as Beeferman)is modified to count the number of times a URL has been selected. Equation (3) shows this similarity computation. In this method $\mathrm{L}(q_1,q_2)$ is the set of links connecting q_1 and q_2 to the same documents, $\mathrm{L}(q_1)$ and $\mathrm{L}(q_2)$ are all the URLs connecting to q_1 and q_2, respectively, and $|L()|$ is the cardinally of $\mathrm{L}()$.

$$Sim(q_1,q_2) = \begin{cases} \frac{|L(q_1,q_2)|}{|L(q_1)\cup L(q_2)|} & if L(q_1)\cup L(q_2) > 0 \\ 0 & else \end{cases} \qquad (3)$$

Considering the fact that selection of links by users is not always deterministic and accurate [1], a connected

component may contain queries that are related to different topics which harm the quality of clustering. FIGURE 1 (a) shows a sample of noise affects on clustering results. Labels on the edges show the number of times a URL is selected for a query. As shown in this figure actually there are two query clusters which are connected together by a noisy selection. Using al-

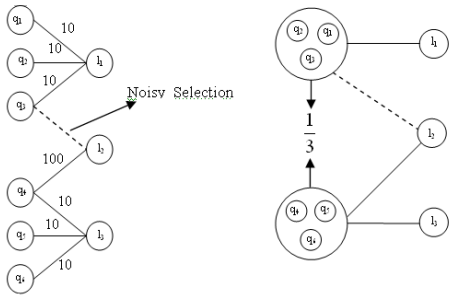

Figure 1: (a) Query-Link bipartite graph with a noisy selection. (b) Query clustering by using algorithm in Beeferman.

gorithm of Beeferman as shown in FIGURE 1 (b), the noisy selection causes two different clusters merge together and using Equation 2, the similarity between them rises to $\frac{1}{3}$. Using a method like Equation 3 the similarity value increases to $\frac{100+1}{100+1+30+30} = \frac{101}{161} \cong 0.63$. In what follows we introduce a new method which used SVD to reduce the effects of noises on clustering. The other advantage of this method is its lower time complexity compared to the related works.

3 SVD based Query Clustering

Our clustering is a content-free method. As explained before we assume that a bipartite graph of queries and URLs are built based on a query log. First we apply a fast algorithm to find the connected components of the graph. Then for each high density connected component (connected component with large number of nodes), we construct a query-URLs relation matrix and apply SVD to find singular vectors corresponding to largest singular values. Finally a K-Mean algorithm is run to find clusters inside each pruned connected component. FIGURE 2 shows the steps of the algorithm.

In the remaining of this section we explain each step in details.

3.1 Bipartite Graph Construction

First from a query log all the records with a same query are combined to a single record. FIGURE 3 depicts such a combined record. Then a bipartite graph

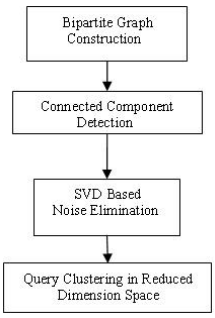

Figure 2: Steps of SVD based Query Clustering

is constructed from those records where right nodes are unique URLs (L) and left nodes are unique queries (Q).

query	links	Number of Selection
q_1	http://.....	1
	http://.....	2
	http://.....	3

Figure 3: click through data record

Each query is represented as a vector where element l_i^k shows the ratio of the number of times that i^{th} URL is selected for k^{th}query to the number of times which has selected for all queries. Also each URL is represented by a vector of queries where element q_k^i shows the ratio of the number of selections of i^{th} URL for k^{th} query to the total number of selection for this query. These definitions are shown in Equation 4 and 5.

$$\overrightarrow{q_k} = (l_1^k, l_2^k, l_3^k, ..., l_m^k); l_i^k = \frac{|l_i^k|}{|l_i|} \quad i = 1, 2, .., m \quad (4)$$

$$\overrightarrow{l_i} = (q_1^i, q_2^i, q_3^i, ..., q_n^i); q_k^i = \frac{|q_k^i|}{|q_k|} \quad k = 1, 2, .., n \quad (5)$$

3.2 Connected Component Detection

To recognize connected components of a bipartite graph we can treat the graph as a forest where a tree represents a connected component. In such trees, nodes in alternate levels belong to queries or URLs. FIGURE 4 shows a sample of constructing such trees for a bipartite graph. To build such tree, we start from a query node and add all its linked nodes which are not visited earlier to the other side of the graph as its children in the next level. This process is repeated until all nodes of a connected component are added to the tree. Then we select another query node which not visited earlier and repeat the above routine to build another

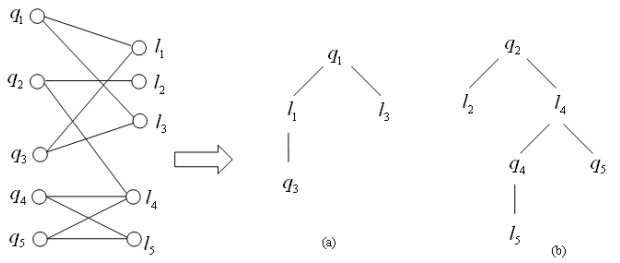

Figure 4: Converting a bipartite graph to a forest

tree. Formally the algorithm for connected component recognition has been shown in FIGURE 5. In

Q is the list of all queries, L the list of all links and C_f are queries and links related to the *f-th* connected component of the bipartite graph.

$Q = \{\overrightarrow{q_1}, \overrightarrow{q_2}, \ldots, \overrightarrow{q_n}\}$

$L = \{\overrightarrow{l_1}, \overrightarrow{l_2}, \ldots, \overrightarrow{l_m}\}$

$C_f = \{Q_f', L_f'\}$, $Q_f' \subseteq Q, L_f' \subseteq L$

Also R_q is an empty set for queries and R_l is an empty set for links.

Steps of the algorithm:

1. r=1
2. Select an unmarked node q_i from Q put it in R_q and do the following steps. If there is not such a q_i stop the algorithm.
3. While R_q is not empty do:
   1. while R_q is not empty do the following steps
      a. Remove the first element of R_q let it be $\overrightarrow{q_{first}}$
      b. If $\overrightarrow{q_{first}}$ is not marked then:
         i. Put all of its elements (links) in R_l.
         ii. Mark $\overrightarrow{q_{first}}$ and put it in C_f
   2. While R_l is not empty do the following steps
      a. Remove the first element from R_l let it be $\overrightarrow{l_{first}}$
      b. If $\overrightarrow{l_{first}}$ is not marked then:
         i. Put all its elements (queries) in R_q.
         ii. Mark $\overrightarrow{l_{first}}$ and put it in C_f.
4. Let r=r+1 and go to step 2.

Figure 5: Connected Component recognition algorithm (Algorithm 1)

each period firstly, a new (not traversed) query node is selected. Then all related URLs and queries are putted in a new connected component where there is a path from the first query to them. Since the algorithm is according to breadth first traversing then it look likes Breadth First Search (BFS) algorithm on a graph, so it can be easily shown that totally, the complexity of this algorithm is of $\theta(E+V)$ [10] where E is the neumber of edges. It is considerable against algorithm has mentioned in Beeferman that in this approach complexity of algorithm is $\theta \approx (V^3)$. Although this algorithm is an agglomerative clustering but finally each connected component is considered as a cluster.

3.3 SVD Based Noise Elimination

In this step using SVD, noisy links inside each recognized connected component from previous step is eliminated. For each C_f as a connected component, a query-URL relation matrix is built. It is like what shown in FIGURE 6. A_{ij} denotes to the value of the element placed in i^{th} row and j^{th} column in matrix and computed by Equation 6. For more information you can see again Equation 4 and 5.

$$A_{ij} = \frac{l_i^j + q_j^i}{2} \tag{6}$$

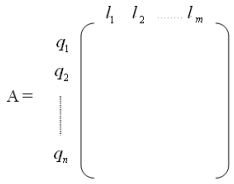

Figure 6: Query-Link Matrix

3.3.1 Singular Value Decomposition (SVD):

SVD is a method for dimensionality reduction. Using it, we obtain singular vectors that have most dependency of data. Such vectors correspond to largest singular values. According to SVD, matrix A can be decomposed as: $A = Q_{n\times r} \times S_{r\times r} \times L^T_{r\times m}$ Where:

- $r \leq \min(n, m)$
- Q is an $n \times r$ matrix where columns are Orthogonal singular vectors of $(AA^T)_{n\times n}$.
- S is a diagonal $r \times r$ matrix with singular values sorted in descending on the main diagonal line of the matrix. Those singular values belong to $(AA^T)_{n\times n}$ and $(A^TA)_{m\times m}$ matrixes.
- L is an $m \times r$ matrix where columns of it are Orthogonal singular vectors of $(A^TA)_{m\times m}$.

3.4 Query Clustering in Reduced Dimension Space

Singular vectors resulted from previous steps construct a new space. We consider all queries vectors in this new space. Eliminating weak singular values and corresponding singular vectors, we obtain a $Q'_{n\times r'}$ low rank of $Q_{n\times r}$ where $r' \leq r$. Since the number of dimensions in new space is r', so each row of $Q'_{n\times r'} \times S_{r'\times r'}$ is a query vector in reduced dimension

space. Using a K-Mean clustering method queries are categorized. We use K-Mean because of its simplicity and its appropriateness for text documents clustering [13].

4 Experimental Results

To evaluate our method we used a real query log of AOL search engine. This collection consists of ∼20M web queries collected from ∼650k users from 01 march to 31 may 2006. We extracted randomly 40,000 queries of this query log for clustering. This collection consists of 22,642 unique queries and 21,486 different selected URLs. The queries were classified into 7 topical categories by a team of approximately twenty human assessors, during a mount. The categories have been labeled as: Computer, Entertainment, Information, Living, Online Community, Shopping and Sport. Afterwards, the connected components of bipartite graph have been detected by result of Algorithm 1(Figure 5). As can be seen in Figure 7 there is only one huge connected component (C_3) which contains 55% of all queries in the collection. Since other connected components had low density, it doesn't seem these components are formed with queries of different categories. Therefore we continued our experiments by C_3. Using Beeferman all queries in C_3 are

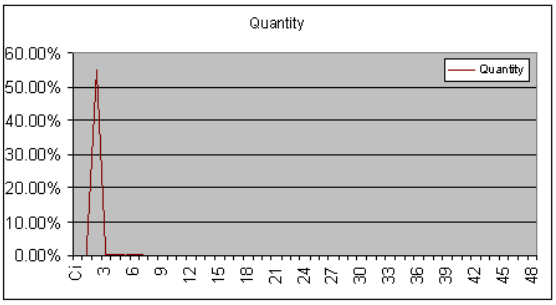

Figure 7: Quantity of connected component and C_3 is hugest connected component.

placed in a same cluster, however as shown in Table 1, this component contains queries of different categories, and if we take it as a cluster, so the correctness of clustering is very low. As mentioned before noisy users' selections influences on quality of clustering and it causes the queries of different context are collected in a same component. For example there are 26% queries related to Entertainment category and also there are 30% queries related to Information category, so context of this cluster is not clear. After constructing queries and URLs vectors of C_3, relation matrix has been formed. Then we applied SVD to project all queries and URLs into reduced dimension space. We used Frobenius norm to estimate the number of dimensions in new space. Using such estimation we obtained 2217 which was suitable value for the number of dimension in our experiment. Also to find the best value for the number of clusters (k), we applied Mean Square Error [13] and reached to value 4. We computed precision measure for Entertainment, Information, Living and Shopping categories in each cluster to verify quality of clusters (these categories have been used because the most of queries belong to them). For computing measure of precision we use Equation 7 that $|Q^{cat_j}_{class_j}|$ is the number of queries related to i^{th} category into j^{th} cluster and $|Q_{class_j}|$ is the total number of queries in j^{th} cluster:

Category Name	Number of Queries in C_3
Computer	0.3%
Entertainment	26%
Information	30%
Living	23%
Online Community	0.2%
Shopping	20%
Sport	0.4%

Table 1: Correspondence of the connected component C_3 to different categories

$$p_{ij} = \frac{|Q^{cat_i}_{class_j}|}{Q_{class_j}} \tag{7}$$

Results of clustering have high quality, if only the precision of one category can be high for each cluster, and also the category with highest precision in a cluster should be different from the categories with highest precision in the other clusters. FIGURE 8 demonstrates quality of clusters which have been extracted from our query clustering on C_3. As can be seen C_3 is divided to 4 clusters. Precision of categories for each cluster are computed then context of each cluster has been labeled by name of category with highest precision value. So Living category has highest precision in Cluster 1, also Entertainment and Information categories have highest precision for Cluster 2 and Cluster 4 respectively. Cluster 3 has low quality because there is no category with considerably high precision, but it can be seen, Shopping category has highest precision compared to other categories which has low precision on other clusters. For more evaluation the result of our clustering, we computed information entropy [14] for each cluster which has been computed by Equation 8. Where $|S|$ is the number of clusters, $|S^i_r|$ is the number of queries of r^{th} cluster labeled by i^{th} category

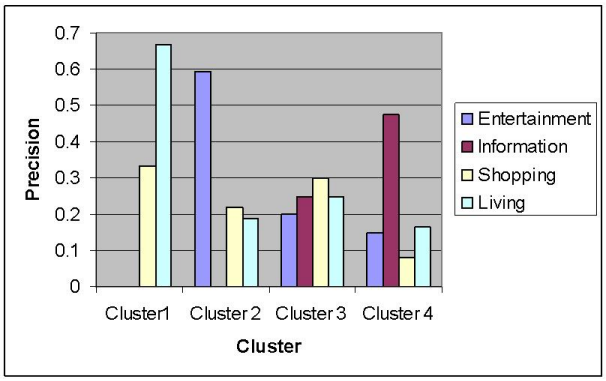

Figure 8: Precision measure for each cluster extracted from C_3

and $|S_r|$ is the total number of queries in r^{th} cluster.

$$E(S_r) = -\frac{1}{\log(|S|)} \sum \frac{|S_r^i|}{|S_r|} \log(\frac{|S_r^i|}{|S_r|}) \qquad (8)$$

$E(S_r)$ is a real value between 0 and 1. If $E(S_r)$ is close to zero , it seems quality of r^{th} cluster is high otherwise the quality is low. Table 2 demonstrates the information entropy of four different extracted clusters from S_1. As can be seen, cluster 1 has lowest entropy and cluster 3 has highest entropy. As mentioned before, the most of main clusters are

Cluster (S_r)	$E(S_r)$
S_1	0.4670
S_2	0.6851
S_3	0.9927
S_4	0.7628

Table 2: Information entropy of four different cluster extracted from C_3

connected together with noisy visited URLs and make a huge connected component. Throughout evaluating our method, we perceived that for each collection of queries and related URLs, there is only one huge connected component which most of the queries were placed to it. To show this observation, we applied Algorithm 1 on several collections with different number of data. It seems, if there is a collection with huge amount of data, so that is enough to apply SVD on the hugest connected component.

5 Conclusion and Future Works

In this article we proposed a new content-free method for clustering queries, and the results of expriments show promising improvement.

References:

[1] T. Joachims, L. Granka, P. Bing, H. Heleneon and G. Geri, Accurately interpreting clickthrough data as implicit feedback, *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*, 2005, pp. 154–161.

[2] H. Cui, J. Wen, J. Nie, and W. Ma, Probabilistic query expansion using query logs, *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, 2002, pp. 325–332.

[3] k. Sparck, Automatic keyword classification for information retrieval, *Butterworths, London, UK*, 1971.

[4] M. Speretta and S. Gauch, Personalized search based on user search histories, *Web Intelligence, Proceedings. The IEEE/WIC/ACM International Conference*, pp. 622–628.

[5] R. Baeza-Yates, C. Hurtado and M. Mendoza, Query Recommendation Using Query Logs in Search Engines, *Current Trends in Database Technology EDBT Workshops*, 2004, pp. 588–596.

[6] D. Beeferman and A. Berger, Agglomerative clustering of a search engine query log, *Knowledge Discovery and Data Mining*, 2000, pp. 407-416.

[7] G. Salton and M. Mcgill, Introduction to Modern Information Retrieval, *McGraw-Hill, Inc*, 1986.

[8] V. Kulyukin, K. Hammond and R. Burke, Answering Questions for an Organization Online, *AAAI/IAAI*, 1998, pp. 532–537.

[9] J. Wen and H. Zhang, Query Clustering in the Web Context, *Clustering and Information Retrieval, Kluwer*, 2003, pp. 195–226.

[10] T. Cormen, C. Leiserson, R. Rivest and C. Stein, Introduction to Algorithms, Second Edition, *The MIT Press*, 2001.

[11] W. Chan, W. Leung and D. Lee, Clustering Search Engine Query Log Containing Noisy Clickthroughs, *SAINT, IEEE Computer Society*, 2004, pp. 305–308.

[12] G. Golub and C. Van Loan, Matrix Computation, *Johns Hopkins, Baltimore, 2nd edition*, 1989.

[13] Y. Zhao and G. Karypis, Evaluation of hierarchical clustering algorithms for document datasets, *CIKM, ACM*, 2002, pp. 515–524.

[14] Quinlan and R.Ross, C4.5: programs for machine learning, *Morgan Kaufmann Publishers Inc*, 1993.