Deep Mining of Web Logs for Search-Query Clustering and Disambiguation

ABSTRACT

Query clustering and disambiguation have been extensively explored in the past to capture web-users' information needs and improve the performance of search. However, these techniques are also plaque by sparse and noisy data. In this paper, we propose a novel algorithm to solving this problem, by simultaneously clustering and disambiguating queries into different *concepts* using both Web pages and query logs. To address the sparseness of the log data, we propose an iterative algorithm to alternate between refining Web page clustering and refining queries' feature representation using query logs as bridge. The resultant Web page clusters provide better *concept* candidates for queries to be associated with each other. A by-product of the process is that queries are naturally clustered around similar concepts. Our extensive experiments lend evidence to our solving the data sparseness and achieving a higher accuracy in query association and disambiguation. We achieve a 29% improvement in the F-measure on a semi-synthetic dataset and 11% in search precision on the real data.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Clustering*;

General Terms

Algorithms, Performance

Keywords

Query Clustering, Query Disambiguation, Query Log, Web Page Clustering, Iterative

1. INTRODUCTION

One of the most fundamental and challenging issues for Web search is to identify users' information needs. What makes this issue challenging is that most Web queries are short. Recent analysis of search engine logs revealed that the average length of a query is about 2.3 words long [8][18]. Shorter queries lead to higher word ambiguity where the same query term may represent totally different information needs for different users. Query clustering [4][22][3] and disambiguation [1][11] are two essential approaches to tackling these issues. In particular, query clustering aims to group similar queries together, by covering the common interests of users [22]. Query disambiguation tries to partition queries into different senses or concepts. However, due to the noisy and sparseness nature of the Web, much remains to be desired in successfully solving either issues.

Among the different methods that have been developed, a main focus is placed on query clustering. Due to the short length of query terms, researchers have started to exploit query logs to enhance performance [3][22]. These query logs is a gold-mine of information. For example, Figure 1 shows a portion of a query log data, where the query terms and their intended pages are connected by edges. The number of times a query refers to its intended pages can also be useful in identifying user needs; for example, from the connections shown in Figure 1, we see that q_2 and q_3 are similar Web pages.

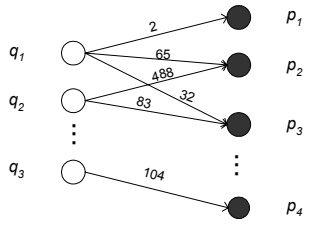


Figure 1. A portion of log data. The number above each directed edge denotes the number of times that q_i refers to p_j in the log. We assume that p_2, p_3, and p_4 are similar Web pages.

While utilizing Web logs promises to provide the much needed information to uncovering similar pages and queries that correspond to the same concepts, our experience in working with very large Web logs have revealed several challenging issues. First, a query only refers to very few Web pages as compared to the whole Web space. If we adopt a vector representation using the log data, the corresponding matrix will be very sparse. Furthermore, due to the diversity and duplicity of Web pages, many similar pages exist in the Web space. Users may visit *similar* Web pages that are not necessarily the identical ones for similar queries; for example we can see this in Figure 1 where q_2 and q_3 are similar even though they do not point to the same page. These problems cause data sparseness, which in turn causes major difficulty in clustering.

Another challenge for query clustering algorithms is caused by the ambiguity of short queries. A well-known example of such a problem is the meaning of "Java" as a query: it could mean coffee, an island in Indonesia or a programming language. Existing clustering algorithms often ignore the ambiguity issue and assign only one cluster for such a query. This solution is clearly very naïve, because it precludes a user from user the same term to mean different things. Query disambiguation, often dealt with by linguistic methods [1], is also a challenging problem. However, in the Web environment, a linguistic is not feasible due to the magnitude of Web space. Furthermore, the linguistic method often fail to reflect the different intentions of users.

In this paper, we propose a novel clustering algorithm for solving the query-data sparseness and the query-term disambiguation problems. To address the sparseness issue, we propose an iterative

SIGIR'04, Month 1–2, 2004, City, State, Country.

procedure in which Web pages are clustered first, and then the initial Web-page clusters are used to compute a feature representation for queries that is less sparse. This result is then used to reinforce Web page clustering. As the procedure goes on, query and Web clusters corresponding to similar user intentions are discovered.

A by product of the clustering procedure is that it solves the query ambiguity issue as well. When clustering terminates, each Web-page cluster expresses a meaningful concept. Such concepts are shared by queries, allowingseemingly different queries to be clustered around similar concepts.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes our iterative procedure to refine the query feature representation and Web page clustering. Section 4 gives our clustering and disambiguation method in the query-feature space. Section 5 presents extensive experiments to evaluate our methods. Finally, Section 6 concludes the paper.

2. RELATED WORK

The characteristics of the short Web query and the noisy search results have led researchers to investigate the query clustering problem [4]. However, using query keywords directly is not reliable due to their short length and word ambiguity. In [4], the top documents retrieved by a search engine for each query are considered as the data source of the query and a hierarchical clustering algorithm is applied. Based on the *click-through data*, in [3], a bipartite graph of queries and documents is constructed and then a graph based agglomerative iterative clustering method is applied to merge vertices of graph continually until a termination condition reaches. Going beyond the clicked URLs in the search engine query and browsing logs, [22] developed one algorithm to use query keywords and the clicked documents to estimate the similarity between queries.

The above methods to utilize query logs are similar to collaborative filtering [14][18][8]. Based on the user-item rating matrix, traditional CF calculates the k-nearest neighbors for the active user and then recommends items that the active user's neighbors like. Therefore, a similarity measure between users should be defined. As for query logs, it is natural to defined a similarity measure from the logs as [22] has done. However, such a method faces data sparseness problem as discussed in Section 1.

Several methods have been proposed to overcome the data sparseness. In [21], a unified framework has been proposed to cluster multi-type interrelated objects (such as Web pages and queries). One type of objects' clustering will be updated by the clustering results of other types of objects and then reinforce others. In our paper, we reinforce the Web page clustering by the refined feature representation of queries instead of query clustering result. Also [21] didn't consider query disambiguation.

Query disambiguation is explored in the Cross-Language Information Retrieval (CLIR) area [2][13] and for queries in mixed languages [7]. In both areas, a major problem is how to translate the query from a *primary language* to a *secondary language*. There may be more than one candidate in the secondary language for a query term. How to choose the correct meaning appropriately is called query disambiguation.

In [1][6], a query is assumed ambiguous based on the underlying document collection. [1] introduced a query disambiguation method by considering part-of-speech patterns of the query's context terms. For each query, [1] analyzed its context in the data collection and classify each occurrence of the query to a question which reflects its context type. Ambiguity is resolved by choosing a question from the generated question list. Such a method suffers from scalability issues and is not applicable to the Web environment. [11] proposed a way to classify user's query to the Open Directory Project (ODP) categories to solve query ambiguity. However, it relies on the user's profile.

3. THE ITERATIVE REFINEMENT CLUSTERING PROCEDURE

A simple assumption to use query logs is that queries are similar if they refer to the *same* Web pages results in data sparseness because similar queries tend to refer to *similar* but not necessary the *same* Web pages. One possible way to overcome the problem of the sparseness of log data is to cluster the similar Web pages together first. If we treat each cluster as one dimension and queries have one nonzero value in a dimension if they refer to the corresponding cluster, the query feature is not as sparse as original. Furthermore, the more accurate the Web page clustering is the better and more meaningful the query feature representation could be. Our motivation is to enhance the Web page clustering result by the refined query feature. To achieve this goal, we project each Web page to the query feature space through the query logs. Thus, besides the content feature, each Web page has an additional feature in the query feature space. By doing so, the refined query features can reinforce the Web page clustering because they refine the Web page features. The enhanced Web page clustering in turn can refine the query features further. Therefore, such an iterative procedure is a mutual reinforcement process.

In this section, we present our iterative procedure in detail. We will later show that this procedure not only refines the feature representation of queries but also yields a more meaningful Web page clustering result which provides good concept candidates for queries to be associated. In the next section, we will describe how to associate the queries to these concept candidates.

3.1 Query and Web Page Vectors

Suppose that query logs refer to m Web pages and n queries in Figure 1. From the logs, we can construct a query by page matrix $X_{n\times m}$ with each entry (i,j) denoting the times that ith query refers to jth Web page. Therefore, each row of X represents a feature vector for the corresponding query q and each column represents a feature vector for the corresponding Web page w:

$$V(q) = [w_1, w_2, \wedge, w_m]^T$$

$$U(w) = [q_1, q_2, \wedge, q_n]^T$$

Because these two feature vectors are constructed from the query logs, we call them *log feature vectors*.

Besides the log feature vector, each Web page has its own content feature. When measuring the similarity between two Web pages, we need to consider both types of information. Two similarities are used for any pair of Web pages: $S_{content}$ is the similarity for content features and S_{logs} for features defined from query logs. Cosine similarity measure can be used to implement each component function. Then, a combined similarity S is induced by a liner combination of the two similarities:

$$S = \alpha \cdot S_{content} + (1-\alpha) \cdot S_{logs} \quad \text{where} \quad 0 \le \alpha \le 1 \qquad (1)$$

A straightforward application of clustering on the query and Web page vectors will result in very poor performance, because in general the vectors are very sparse. Only a small fraction of Web pages are visited by each query and each Web page is only

relevant to a small fraction of queries. Thus, we must find a way to resolve the sparseness of the query logs. We discuss how to do this next.

3.2 Iterative Refinement Clustering (IRC)

We first give some notations used in the following:

- $X_{n\times m}$ is a *query by page* matrix of query logs as defined in Section 3.1.
- $P_{m\times m}$ is the *page to page* similarity matrix.
- $Q_{n\times k}$ is the query feature matrix based on Web page clusters with each row denoting one query.
- $S_{n\times n}$ is the *query to query* similarity matrix.

Q is the matrix that we want to refine. It is equal to refine S because $S = Q \times Q^T$. To see the convergence of the iterative algorithm, we select S to define the termination condition. Therefore, in the following description, we aim to refine S.

A naïve way to obtain this matrix is to simply consider the web log alone. This naïve way is like CF and corresponds to the following computation: $S = X \times X^T$, where X^T is the transpose of X. However, such a way faces the logs sparseness.

By applying an iterative refinement algorithm, we can do much better. This algorithm is based on iteration on two steps. First, we perform a clustering operation on pages based on page similarity. Then, we refine the query feature and compute the query similarity using the newly computed page clustering. Each iteration l gives a new matrix for S and P. The iteration continues until the matrix $S(l)$ converges to a fixed point.

The clustering phase is used as a subroutine in our overall procedure. Thus, we describe it here first.

Before detailing the algorithm, we present our motivation in matrix formulation. The matrix can give us an intuition of the reinforcement process.

For simplicity, we ignore the content feature of pages and queries and assume that all matrices below take only 1 or 0 values. A value 1 means that the two entities are related. 0 means otherwise. We choose k-means clustering for Web pages, so we first present the matrix formulation of it.

Soft k-means algorithm

Assume $A_{m\times w}$ is the document by word matrix, $C(0)_{m\times k}$ is a const matrix with 1 at locations $\{[(m / k) \times j+1], j)$: $j = 1,\ldots,k\}$ and 0 otherwise. $C(0)_{m\times k}$ is used to select the initial centroid vector.

Algorithm: Soft k-means

Step 1. Initialize the centroids by $A^T\times C(0)$. (In fact the jth centroid vector is the $[(m / k) \times j+1]$th document vector.)

Step 2. Iterate Step 3 to 4 t times.

Step 3. For the lth iteration, calculate the similarity between documents and centorids: $C(l) = A\times(A^T\times C(l\text{-}1)) = (A\times A^T)\times C(l\text{-}1)$, each entry (i,j) denote the similarity between the ith document and the jth centroid.

Step 4. Refine the centroids by $A^T\times C(l)$.

Step 5. Return $C(t)$.

From above, the whole function can be set $C(t) = (A\times A^T)^t\times C(0)$. $A\times A^T$ is the similarity matrix between documents and $C(t)$ is the document by cluster (centroid) matrix which can be used to cluster documents.

Notation

- $V_{n\times m}$ is a *query by page* matrix of query logs, with each row equal to $V(q)$ and each column equal to $U(w)$in Section 3.1.
- $P_{m\times m}$ is the *page to page* similarity matrix.
- $Q_{n\times k}$ is the query feature matrix based on Web page clusters with each row denoting one query.
- $S_{n\times n}$ is the *query to query* similarity matrix.

Q is the matrix that we want to refine. It is equal to refine S because $S = Q\times Q^T$. To see the convergence of the iterative algorithm, we select S to define the termination condition.

Iterative Refinement Clustering (IRC)

To give a close form of the iterative procedure, we ignore Web page content in the following description. For simplicity, we assume that all matrices below take only 1 or 0 values. A value 1 means that the two entities are related. 0 means otherwise.

Let $P(l)$ be the page-to-page similarity, where $P(0)=X^T\times X$ because we ignore their contents. Then, for each $P(l)$, the k-means clustering result is $C_l(t) = (P(l))^t\times C(0)$

For convenience, we refine the page similarity by query similarity in the below description. In fact, it is equal to refine page feature by query feature because

$$\begin{aligned} P(l) &= X^T \times S(l-1)\times X \\ &= X^T \times Q(l)\times Q(l)^T \times X \\ &= (X^T \times Q(l))\times (X^T \times Q(l))^T \end{aligned}$$

$X^T\times Q(l)$ is the refined page feature by query feature.

The whole process can be described as:

Iterative Refinement Algorithm IRC:

Step 1. Initialize matrices and parameters:

l=1;

$S(0) = I$, where I is the identity matrix;

Step 2. The page-to-page similarity matrix:

$$P(l) = X^T \times S(l-1)\times X$$

Step 3. Apply the k-means algorithm on $P(l)$ to obtain page-to-cluster matrix:

$$C_l(t) = (P(l))^t \times C(0)$$

Step 4. Obtain refined query feature matrix:

$$Q(l) = X \times C_l(t)$$

Step 5. Obtain a new query-to-query similarity matrix

$$S(l) = Q(l)\times Q(l)^T$$

Step 6. If $S(l) = = S(l\text{-}1)$ then exit. Otherwise, $l=l+1$, normalize the matrix S and P, and go to Step 2.

The closed form of the iteration can then be expressed as (where we omit the normalization function):

$$\begin{aligned} & S(l+1) \\ &= Q(l)\times Q(l)^T \\ &= X\times C_l(t)\times (X\times C_l(t))^T \\ &= X\times (P(l))^t \times C(0)\times ((P(l)^t \times C(0))^T \times X^T \\ &= X\times (X^T \times S(l)\times X)^t \times C(0)\times ((X^T \times S(l)\times X)^t \times C(0))^T \times X^T \end{aligned}$$

We now discuss how to *normalize* the matrix. For each row of the matrix S and P, if an element is greater than zero, then we set it to be one. Otherwise, we set it to be zero.

We now show that this iteration converges to a fixed point for a constant t value and for the case where all elements of matrices are either 1 or zero.

Lemma 1. In each iteration, if an element of the similarity matrix S is greater than zero, it will never become zero for future iterations.

Lemma 2. In each iteration, if an element of the similarity matrix P is greater than zero, it will never become zero for future iterations.

Theorem 1. Iterative Algorithm IRC will converge in a finite number of iterations.

Proof. Since each step of the iteration never removes a value one from the matrices S and P, also since S and P have finite dimensions, in $O(m^2+n^2)$ steps the algorithm must terminate. □

However, in reality the algorithm will converge much faster than the theoretical bound mentioned above. We show this in our experiments below.

In the experiments, we use hard k-means algorithm and also consider Web page content using formula (1). Furthermore, we refine Web pages' features based on refined query feature to avoid the quadratic calculation complexity to calculate the similarity matrix.

4. CLUSTER AND DISAMBIGUATE QUERIES

Assume that we obtained k clusters of the Web pages $\{C_1, C_2, ^, C_k\}$ and there are l_i Web pages $\{P_{i1}, P_{i2}, ^, P_{ili}\}$ in each cluster C_i. Recall that each Web page P_{ij} has a log vector U_{ij}' in the query feature space obtained by $X^T \times Q(l)$. In order to project a Web page cluster to the query feature space to form a concept, we extract the centroid of the cluster. In particular, each Web page cluster corresponds to a vector in the query feature space and we call it *concept vector*:

$$CV(i) = \frac{1}{l_i} \sum_{j=1}^{l_i} U_{ij}'$$

where $CV(i)$ is the concept vector of the cluster C_i.

Here, we implicitly assume that each Web page cluster corresponds to a meaningful concept, a fact that we will verify in the experimental section. In addition, we assume that the concepts are shared by the queries. This is reasonable because when a user issues a query and finally visits a Web page, he/she, in most cases, considers the browsed Web page has the meaning or concept of the query in his/her mind. Ambiguous queries can be used by different users to visit Web pages associated with different concepts. Thus the concepts of all clusters of Web pages are shared among the queries. Therefore, it is natural to project the Web page clusters onto the query feature space to form a concept set.

In the query feature space, we obtain refined representations of the queries and also more meaningful concept vectors from the Web page clustering result. For each query q, we can calculate a similarity between q's refined representation and a concept vector CV using consine similarity measure. Disambiguating a query corresponds to assigning the query to the concept CV if the similarity between q and CV is larger than a pre-defined threshold β. An ambiguous query has high similarities with several different concept vectors and thus can be given different meanings. Apparently, the result can be influenced by the pre-defined threshold β. The lower the threshold is, the coarser each cluster would be.

5. EVALUATION

In this section, we present the performance evaluation of the proposed method to cluster and disambiguate queries. Experimental results conducted on both the semi-synthetic data and the real data are studied. The semi-synthetic data are generated to simulate real search engine logs and the real data are extracted and sampled from MSN search logs. In both experiments, we use the k-means algorithm to cluster Web pages and set k = 150 for the semi-synthetic data and k = 200 for the real data.

5.1 Baseline (CF)

To illustrate the impact of the sparseness of log data, we employ the baseline algorithm which uses a similarity measure similar to CF to find nearest neighbors. CF defines a similarity measure between "users" based on the user-item rating matrix. Recall in Figure 1, we can also define a measure between queries based on their "ratings" of Web pages. Here the "rating" is the times that a query refers to one Web page.

Based on the defined similarity measure, we apply k-means algorithm to cluster queries. Here we modify standard k-means algorithm to fit the disambiguation problem. In the experiment, first, we use standard k-means to find the local optimal cluster models, i.e. cluster centroids. Then we assign each query to a cluster if the similarity between the query vector and the corresponding centorid is larger than a pre-defined threshold γ. Similar to the threshold β, the performance relies on γ. However, for the sparseness problem of baseline algorithm, β is larger than γ to achieve the best result generally.

5.2 Naïve Way

In the experiment, we also present a naïve way to compare with our IRC. The naïve way is to just cluster the Web pages one time based on content, change the query feature based on the clustering result and then cluster and disambiguate the queries like our algorithm. This way is equal to the first iteration of IRC.

5.3 Semi-Synthetic Data

The semi-synthetic data is a simulated search-engine query log consisting of two types of objects, the Web pages and the queries. In the synthetic data, the Web pages are collected from real sources, while the queries and their referred Web pages are automatically generated from a probabilistic model whose parameters are specified in advance. The real categories of the Web pages and queries are hidden from the proposed algorithm and are taken as a ground-truth for the performance evaluation. In the following, after a brief introduction of the data generation process and the performance evaluation metrics, we present our comparative experiments.

5.3.1 Data Generation

The data generation process is composed of 3 steps.

First, all Web pages used in the experiment are collected from the Open Directory Project [15]. All the categories are selected from the second-level categories of the directory. In all, we selected 126 categories and each size is above 100. The number of

associated Web pages is more than 40,000 and the biggest category has about 1,000 Web pages. In the experiments, we only extract the pure text from each Web page.

Second, we generate the queries automatically and the categories of each query are set the same as its preferred Web page categories. The queries are generated in two types: unambiguous queries and ambiguous queries. Each unambiguous query has only one preferred category of Web pages. We generate one set of unambiguous queries for each category of web pages and the size of the query set ranges from 30 to 100, uniformly distributed among the Web page categories. In all, we generate about 8,000 unambiguous queries. To generate ambiguous queries, we first assign to each query a number ranging from 2 to 6 which corresponds to the number of its different concept meanings. Then, for each query, we randomly select 2-6 Web page categories equal to its meaning number as its preferences. In the experiments we generated 10,000 ambiguous queries totally. For each ambiguous query, we assign one importance level for each of its preferences. The importance level ranges from 3 to 6 and is proportional to the frequency that the users associate a meaning (preference) to a query.

Finally, page hits are generated based on existing queries, existing Web pages, and query preferences and their importance levels. A page hit is generated in the following process: (1) select a query randomly, (2) get the preferences of the query, (3) if the query is ambiguous, randomly select a preference according to the probability which is determined by its preference importance levels, (4) randomly select a Web page from the corresponding Web page category according to Web pages' importance values which are equal to their values of PageRanks [16]. In the experiments, we generate the unambiguous and ambiguous page hits separately. The differences between unambiguous and ambiguous queries' hits are: in step (3), there is only one preference for an unambiguous query. The average number of hits an ambiguous query takes is proportional to the number of its preferences. And the hit numbers of unambiguous ones are uniformly distributed. At last, we generate 100,000 unambiguous hits and 300,000 ambiguous hits. Therefore, on average, there are 11 page hits per unambiguous query and 30 page hits per ambiguous query. Because each ambiguous query has 4 preferences on average, the number of page hits per preference is about 7-8.

To simulate the reality, we generated some noise page hits as well. In the noise generation model, we generate totally random page hits which are uniformly distributed on all possible pairs of queries and Web pages. In the experiments, we will study the performance under different ratio of noise page hits.

5.3.2 Evaluation Methods

We adopt entropy to evaluate the Web page clustering result and F measure to evaluate the query clustering and disambiguation result.

5.3.2.1 Entropy

Entropy is defined in information theory [5], which measures the uniformity or purity of a cluster. Specially, given a cluster A and category labels of data objects inside it, the entropy of cluster A is defined:

$$H(A) = -\sum_j p_j \cdot \log_2(p_j)$$

where p_j is the proportion of jth category C_j's data in the cluster A. That is:

$$p_j = \frac{|A \cap C_j|}{|A|}$$

The clustering entropy is the weighted sum of all the clusters. Assume D is the whole data collection, then:

$$H = \sum_{All\ Clusters\ A_k} \frac{|A_k|}{|D|} H(A_k)$$

5.3.2.2 F measure

The precision, recall and F measure are commonly used metrics in information retrieval area to evaluate the retrieval results [20]. To evaluate query clustering and disambiguation result, we use F measure [10]. Assume we know correct categories of the queries. For each cluster, we calculate the precision and recall of it for each given category. The F measure is defined by combining the precision and recall together. More specifically, for cluster j and category i

$$Recall\ (i, j) = \frac{n_{ij}}{n_i}$$

$$Precision\ (i, j) = \frac{n_{ij}}{n_j}$$

$$F\ (i, j) = \frac{2 \times Precision(i, j) \times Recall(i, j)}{Precision(i, j) + Recall(i, j)}$$

where n_{ij} is the number of members of category i in cluster j, n_i is the number of members in category i, n_j is the number of members in cluster j and $F(i, j)$ is the F measure of cluster j and category i.

The whole F measure of the query clustering and disambiguation result is defined as a weighted sum over all categories as follow:

$$F = \sum_i \frac{n_i}{n} \max_j (F(i, j))$$

where the *max* is taken over all clusters and n is the total number of queries.

The query clustering and disambiguation result is sensitive to the similarity threshold β. When β increases, the precision becomes higher and the recall becomes lower. The F measure is influenced too. To give an objective evaluation, we change the similarity threshold β from 0.1 to 0.9 with step 0.1 and select the largest F measure as the evaluation of the clustering and disambiguation result. For the baseline CF, we set $\gamma = 0.05$ because at this value it yields the best result.

5.3.3 Experimental Results

The following experiments illustrate the refinement of our iterative procedure on the semi-synthetic data. To cluster the Web pages, we adopt the similarity defined in formula (1). We rewrite it here:

$$S = \alpha \cdot S_{content} + (1 - \alpha) \cdot S_{logs}, \text{ where } \quad 0 \le \alpha \le 1$$

Here α adjusts the weight between content and log features, which has significant impact on Web page clustering accuracy and also the query clustering and disambiguation accuracy. We will illustrate its impact in the following. Since our proposed algorithm is an iterative process, in our experiment we let the algorithm run

10 iterations, i.e. reinforcing the Web page clustering by the refined query feature representation 10 times. In the following, we use all the generated 8000 unambiguous queries and select the correct number of ambiguous queries according to the ratio.

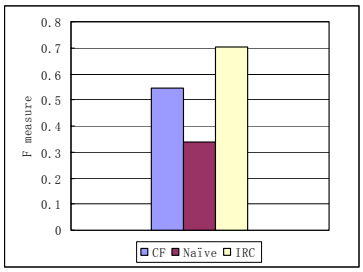
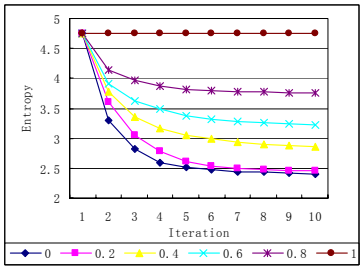


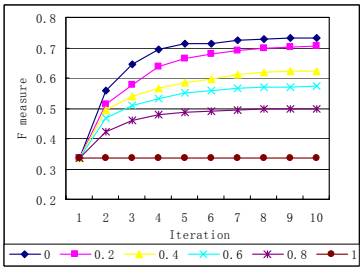
Figure 2. Results with α=0.2, ambiguous ratio=70%, noise ratio=70%

Figure 2 is the query clustering and disambiguation results which are evaluated by F measure. Here we set α=0.2, the ratio of ambiguous to unambiguous queries at 70%, the ratio of the noise logs to the correct ones at 70% to simulate the real search engine logs. In the figure, we give three results: CF, the naïve way, and our algorithm IRC. From the results, IRC is far better than CF (29% relative improvement) and naïve way (108% relative improvement). As we have discussed above, the baseline CF is influenced by the sparse problem of the logs and the naïve way does not perform well because the cluster result of Web pages based on content is highly noisy. IRC combines the content and log feature to yield a better result. This confirms that our algorithm is very effective to reinforce the Web page clustering and refine query feature by overcoming the log data sparseness.

Then we study the impact of α and the convergence property of IRC with the ambiguous ratio and noise ratio set as above.



(a)



(b)

Figure 3. The impact of α and the convergence of IRC

In Figure 3, we show the results against the iteration number. To see the influence of our iterative procedure (IRC), we present the Web page clustering result as entropy values in Figure 3 (a). Lower entropy denotes a result which has a higher accuracy. The best clustering result is achieved when alpha is 0.2 or 0. When alpha is set to 1, which means only the content feature is considered, the clustering results could not benefit from the iteration process. Hence the entropy keeps constantly at a high value. This illustrates that the content feature is highly noisy and problematic. At other levels of α, IRC reduces the entropy significantly. For example, when α = 0.2, we achieve 48% entropy reduction. Furthermore, the results also confirm that the IRC can converge to a fix point after 5-6 iterations. The result in Figure 3 (b) is similar and the best result is attained when α = 0, which means the log feature is more reliable than content feature in this data set.

5.3.4 Parameter Impacts

On the semi-synthetic data, we did extensive experiments to study the performance of our algorithm. In this section, the results on different parameter settings are studied. These parameters include: the sparseness of the logs, the ratio of ambiguous queries, and the ratio of the noise logs.

5.3.4.1 Impact of Sparseness

In this experiment, we also set α = 0.2 and ambiguous queries ratio as 70%. To illustrate the impact of sparseness of the log data, we adjust the degree of density and randomly select 20%(very slightly), 40%(slightly), 60%(moderately), 80%(strongly) and 100%(very strongly) of the correct logs. For each degree, we add noise logs with ratio as 70%. Figure 4 is the result. From this plot, we can see that the sparseness has a huge impact for all the algorithms. With few logs, the effect of our algorithm is much degraded. However, when the log data increase, our algorithm has a huge improvement compared to CF and naïve way. With moderate ratio of logs, our algorithm can perform fairly well. Besides, we also *additionally* generate more logs dense enough (densely). In the figure, when the logs are not sparse, CF performs fairly well and IRC benefits little. This means when the logs are dense enough it is not necessary to refine the query feature as IRC because of the noise introduced by the Web page clustering.

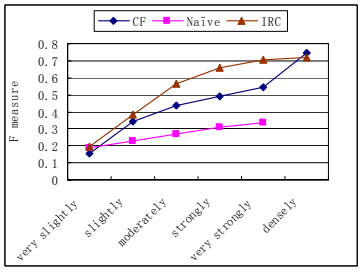


Figure 4. Accuracy when dataset evolves form slightly related to densely related

5.3.4.2 Impact of Ambiguous Queries

Next, we illustrate the impact of number of the ambiguous queries. Here we set α = 0.2 for Web page clustering, noise ratio

Deleted: 4
Deleted: Figure 4
Deleted:
Deleted:
Deleted: 3
Deleted: gure 3
Deleted: Figure 5
Deleted: 5

as 70%, and adjust the ambiguous ratio from 10% to 90% with step 20%.

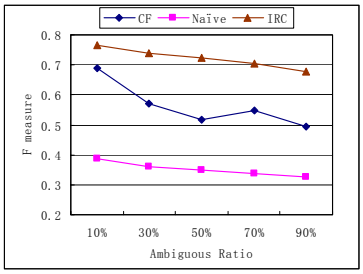


Figure 5. Impact of ambiguous queries where the ratio ranges from 10% to 90% with step 20%

Figure 5 shows that the number of ambiguous queries has no large impact on the results. When the number of ambiguous queries increases, all the performances drop. Compared to CF, IRC is insensitive and always performs better.

5.3.4.3 Impact of Noise Logs

This experiment is to test the robustness of our algorithm. In the above experiments, we set the noise ratio at 70%. Here we set alpha = 0.2 and the ratio of ambiguous queries at 70%, then adjust the ratio of the noise query logs to the correct ones from 10% to 90 with step 20%. Also we run the CF and naive algorithms on such data to compare. Apparently, in Figure 6, more noise will decrease the performance. Compared with CF, at all level of noise logs in the figure, IRC is better. The more noise is, the more relative improvement of IRC is. Therefore, IRC is more robust than CF.

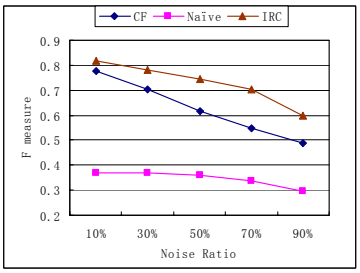


Figure 6. Accuracy along with noise ratio

5.4 Real Data

In this section, we study the performance of our algorithm on real data. The query logs are obtained from the MSN search engine. Our data is sampled from the query logs of one week in August 2003.

To obtain an easily manageable scale, we select the queries used most frequently, the URLs most frequently visited after those queries, and the corresponding logs from the raw data. After preprocessing, we get about 10,000 queries, 100,000 URLs and about 200,000 different logs. For each URL, we downloaded its content from Internet. In the experiment, we ignore the query keywords for convenience. In addition, we set α = 0.2 in this clustering algorithm.

It is difficult to evaluate the performance of our algorithm due to the lack of correct category labels for Web pages and queries. To address this problem, we propose an intuitive method similar to [12], i.e. precision. For convenience, we only evaluate the query cluster results. We ask human users to judge each cluster C_i, which allows us to capture the major concept meaning contained in the cluster. From C_i., we select all the queries expressing the major concept of C_i as the set M_i. Finally we calculate the precision of each cluster by the following formula:

$$P_i = \frac{|M_i|}{|C_i|}$$

where $|C_i|$ is the number of queries in the cluster C_i, and $|M_i|$ is the set of queries contained in the cluster that are relevant to the major concept. To get an overall evaluation of the cluster result, we adopt the Micro- and Macro- metrics commonly used in IR:

$$Micro\_P = \sum_{i=1}^{k} \frac{|C_i|}{N} P_i$$

$$Macro\_P = \frac{1}{k} \sum_{i=1}^{k} P_i$$

where k is the number of clusters to be evaluated and N is the sum of the numbers of queries in those clusters.

Because we evaluate the performance based on human judgments, we wish to avoid being over-burdened by the labeling work. Therefore, we only selected the biggest ten clusters for evaluation in the experiment. Given a query cluster C_i, we ask 5 volunteers to judge the major concept of it and then find the corresponding query set M_i. Each query is judged by the volunteers themselves ~~or by submitting it to Google Directory search box to get its~~ possible concepts or categories.

5.4.1 Results

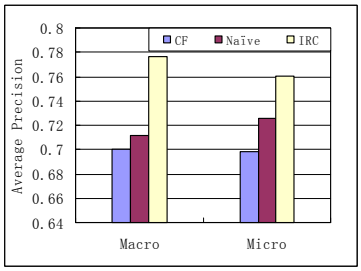


Figure 7. The average precision of the biggest ten query clusters

To get an objective comparison, we set similarity threshold β = 0.5 for naïve way and our IRC. Because the dimension of the query feature for the CF is far lager than the naïve and IRC, we set γ = 0.05 for CF to keep its cluster sizes comparable with the former two. In Figure 7, we present both micro-precision and macro-precision. In both metrics, we achieve significant improvements. For example, when using macro-precision to evaluate, CF is about 70% and IRC is about 78%. We improve the performance by 11% relatively. Compared with naïve method, we also achieve 9% relative improvement.

5.4.2 Case Study

The results on real data show IRC can produce more meaningful clusters. For example, there is a cluster about "university" in the result of IRC but there is not for CF. Another example is the cluster about "cars". This is because such queries always refer to similar Web pages instead of the same ones. For example, two

Deleted: 8

Deleted: 7

Deleted: Figure 8

queries “kia” and “jeep” seldom click the same Web pages. Therefore, due to the sparseness of query logs, CF fails to find such clusters.

Here we give some examples to illustrate the effect of query disambiguation. The query “windows” has two meanings: Microsoft windows operating system or the windows in a house. In the clusters we get, “windows” is assigned to two different clusters. The first cluster is about computer and includes queries such as “Microsoft”, “windows xp”, and “computer software” etc. The second cluster is about house and includes queries such as “doors”, “shower curtains”, and “mini blinds” etc. We also investigated the query logs about “windows” and found that the Web pages it referred to indeed belong to the two categories. Another example is “Webster”, which may mean a kind of dictionary or a university name.

5.5 Discussion

The extensive experiments prove that our algorithm can achieve better clustering and disambiguation results of search queries by reinforcing the Web page clustering using query logs. In addition, our algorithm is robust to the number of ambiguous queries and the ratio of noise. The time complexity of our algorithm is not high because we always get a high improvement after second iteration and the iteration always converges after five to six iterations. Thus, the time complexity is acceptable compared to its significant performance improvement.

In the semi-synthetic data, the log information we generated is more accurate than the content, therefore, the baseline method outperform the naïve way. While in the real data, the content feature seems to be more reliable than the log feature, so the result is different. In both data, our algorithm combines both features and overcomes the sparseness of the log data to achieve the best results.

However, there is a question that how to select the parameters α, β, and the number of Web page clusters. We believe the optimal values depend on the data collections and applications. It is difficult to determine them in advance. But it can be adjusted over time and in light of the system’s use.

6. CONLUSIONS

Query clustering has received more and more attention in recent years, but most existing clustering algorithms seldom consider query disambiguation. In this paper, we proposed a unified framework to simultaneously cluster and disambiguate queries by mining query logs. The proposed iterative procedure not only resolves the data sparseness of query logs, but also gives more meaningful Web page clusters. The concept set which corresponds to the Web page cluster result is used to assign the queries. In this way, ambiguous queries could be assigned to concepts with different meanings. The experimental results proved that our algorithm is very effective to cluster and disambiguate queries. We get 29% and 11% performance improvement on semi-synthetic and real data respectively compared to the baseline. The case study of our algorithm also illustrates our algorithm’s effectiveness to disambiguate queries and find meaningful clusters. In the future, we wish to continue to explore iterative methods to utilize log data more effectively.

7. REFERENCES

[1] Allan J. and Raghavan H. Using part of speech patterns to reduce query ambiguity, In Proceedings of the 25th annual international ACM SIGIR, pages 307--314, 2002.

[2] Ballesteros L. and Croft W. B. Resolving ambiguity for cross-language retrieval, In Proceedings of the 21st annual international ACM SIGIR, pages: 64 - 71, 1998.

[3] Beeferman D. and Berger A. Agglomerative clustering of a search engine query log, In Proceedings of the sixth ACM SIGKDD, pages 407--415, 2000.

[4] Chuang S.-L. and Chien L.-F., Towards automatic generation of query taxonomy: a hierarchical term clustering approach, In Proceedings of 2002 IEEE International Conference on Data Mining, (ICDM), 2002.

[5] Cover T. M. and Thomas J. A. Elements of Information Theory, Wiley, 1991.

[6] Cronen-Townsend S. and Croft W. B. Quantifying query ambiguity. In Proc. of HLT, pages 94--98, 2002.

[7] Fung P., Liu, X., and Cheung, C. S. Mixed-language Query Disambiguation, In Proceedings of ACL ‘99, Maryland, June 1999.

[8] Herlocker J. L., Konstan J. A., Borchers A., and Riedl J.. An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR, pages 230-237, 1999.

[9] Jansen B.J., Spink A., Bateman J. and Saracevic T., Real Life Information Retrieval: A Study of User Queries on the Web, In SIGIR Forum, Vol. 31, pp. 5-17, 1998

[10] Larsen B. and Aone C. Fast and Effective Text Mining Using Linear-time Document Clustering, KDD-99, San Diego, California, 1999.

[11] Liu, F., Yu, C., and Meng, W., Personalized Web search by mapping user queries to categories. In Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM '02). USA, 558--565.

[12] Liu T., Liu S., Chen Z., Ma W.-Y. An Evaluation on Feature Selection for Text Clustering, ICML-2003, Washington DC, 2003.

[13] Maeda A., Sadat F., Yoshikawa M., and Uemura S. Query term disambiguation for Web cross-language information retrieval using a search engine, In Proceedings of the fifth international workshop on Information retrieval with Asian languages, 2000.

[14] McNee, S., Albert, I., Cosley, D., Gopalkrishnan, P., Lam, S.K., Rashid, A.M., Konstan, J.A., and Riedl, J., On the Recommending of Citations for Research Papers. In Proceedings of ACM 2002 Conference on Computer Supported Cooperative Work (CSCW2002), New Orleans, LA, pp. 116-125

[15] Open Directory Project. http://dmoz.org/.

[16] Page L., Brin S., Motwani R., and Winograd T. The PageRank citation ranking: Bringing order to the web, Technical Report, Computer Science Department, Stanford University, 1998.

[17] Sanderson M. Word sense disambiguation and information retrieval, In Proceedings of the 17th annual international ACM SIGIR, pages 142--150, 1994.

[18] Sarwar B.M., Karypis G., Konstan J.A., and Riedl J., Item-based Collaborative Filtering Recommendation

Algorithms. In: Proceedings of the 10 International WWW Conference, Hong Kong (2001)

[19] Silverstein C., Henzinger M., Marais H., and Moricz M., Analysis of a very large altavista query log. Technical Report SRC 1998-014, Digital Systems Research Center, 1998.

[20] Steinbach M., Karypis G., and Kumar V. A comparison of document clustering techniques, In KDD Workshop on Text Mining, 2000.

[21] Wang J., Zeng H.-J., Chen Z., Lu H., Li T., and Ma W.-Y. ReCoM: reinforcement clustering of multi-type interrelated data objects, In Proceedings of the 26th annual international ACM SIGIR, pages: 274 - 281, 2003.

[22] Wen J.-R., Nie J.-Y., and Zhang H.-J. Query Clustering Using User Logs, In ACM TIOS, vol. 20, no. 1, pages 59-81, 2002.

[23] Xu J. and Croft W. B. Improving the effectiveness of information retrieval with local context analysis, In ACM TOIS, vol. 18, no. 1, pages 79--112, January 2000.