

Dynamical Clustering of Personalized Web Search Results

Xuehua Shen
CS Dept, UIUC
xshen@cs.uiuc.edu

Hong Cheng
CS Dept, UIUC
hcheng3@uiuc.edu

Abstract

Most current search engines present the user a ranked list given the submitted user query. Top ranked search results generally cover few aspects of all search results. However, in many cases, the users are interested in the main themes of search results besides the ranked list so that the user will have a global view of search results. This goal is often achieved through clustering approaches. Personalized search studies ranking or reranking the search results based on implicit feedback. The personalized search system will infer user information need based on user search engine interaction and rerank the search results. Same as the ranked search result lists, clusterings of search results intuitively should also be dynamically tuned according to user search system interaction. Thus it brings interesting clustering challenges in the personalized search framework. Clustering results should change dynamically to reflect the personalized ranking of search results. However, traditional static clustering algorithms based on document similarity cannot achieve this goal. In this paper, We study how to incrementally cluster the search results and dynamically update the cluster representation based on user's implicit feedback.

1. Introduction

One important task of a search engine is to rank most relevant search results at top. Most search engines achieved this goal by some well-known algorithms like PageRank [5] and HITS [4]. However, in many occasions, users want to have a global picture of the different themes about search results such as clustered result presentation. These presentations will be beneficial to the user search experience. Such functionality is often achieved through clustering algorithm or supervised learning such as regression [10] etc.

Personalization of the search results is considered as one of major approaches to improve web search. Personalized search ranks the search results or reranks the general ranked search results based on the created user model, which re-

flects user short-term or long-term interest. In the personalization framework, new challenges for clustering search results emerge since search results is interactively reranked based on user's feedback. So efficient clustering algorithm is desired to perform online clustering of search results. In addition, clustering results should dynamically update to reflect the reranking based on user's implicit feedback. Traditional clustering algorithms based on document similarity (for example, cosine similarity) is regarded as "static" since the clustering result cannot reflect the changes in the ranking of documents and user interaction [8]. The third key issue is the result presentation strategy. Organizing the clustering results in a clear and effective way is very important to enhance user's search experience.

In this paper, we study the clustering and result presentation strategy in the personalization framework. Our goal is to design an effective and efficient dynamically updating clustering algorithm. We base our work on the assumption that clustered results can augment top ranked result representation, which has been proved by a lot previous work such as [9]

Our work is based on theUCAIR toolbar [6]. Just like query expansion and dynamic result reranking of search result based on user implicit feedback [3], the cluster presentation is also personalized, i.e. dynamically change based on user implicit feedback such as submitted query and click-through during user interaction with the toolbar. For example, when the user clicks a document, we can know more about the user's information need. We can change the cluster structure and presentation based on this implicit feedback. One simple strategy is that if we do hierarchical clustering, we can show next-level clusters of the cluster to which the document clicked belongs and push down other top level clusters. Just like when the user clicks one result and then clicks *Back* button, the top ranked result will be reranked. We will reorganize the cluster presentation after we get the user implicit feedback.UCAIR framework is introduced in Section 2 in detail.

The major contribution of this project is as follows.

- We implemented efficient clustering algorithms and used two different result presentation strategies at the

client side.

- We developed an incremental clustering algorithm to incrementally update the cluster structure.
- We design and implement Fisheye View algorithm to dynamically change the cluster structure based on user’s implicit feedback.

The rest of the paper is organized as follows. Section 2 introduces the UCAIR toolbar, under which we study the dynamic clustering algorithm. The details of the similarity measure, the clustering algorithm and clustering presentation are described in Section 3. In Section 4 and 5, we describe the design and implementation of incremental clustering (FixInc) algorithm and dynamical clustering (Fisheye View) algorithm respectively. Related work is discussed in Section 6 and conclude our study in Section 7.

2. UCAIR Framework

Our study is based on UCAIR [6] toolbar. UCAIR toolbar is a personalized search toolbar at the client side. It provides the basic functionality that many search engine toolbars such as Google toolbar provide. After the user composes a query, the toolbar communicates with the search engine and retrieves the search result web page for the user so that the user does not need to visit the search engine web site to search. Besides the basic functionality, UCAIR toolbar does the query expansion and result reranking to improve the retrieval quality.

For the query expansion, when the user submits a query, say, “java”, the toolbar will look through the user’s previous queries and decide whether the previous queries are correlated with the current query or not. If so, the toolbar will select meaningful query terms from previous queries, e.g., “programming” for the query expansion and submit the expanded query to the search engine. Thus the representation of the user information need is augmented and the possible search term ambiguity is reduced.

For the result reranking, if a user views the search result web page and clicks one search result, when he clicks *Back* button or *Next* button, the search result will be automatically reranked according to the title and summary of previously clicked web pages. We believe that the user clicks one search result because the appealing summary of the search result, instead of the whole content of clicked web page [7], reflects the user information need. Some originally relevant web pages ranked lower by the search engine will be pushed up. Here, the user’s previous queries and previous click-through provide the implicit feedback [3] to the UCAIR toolbar. Through the implicit feedback from the user, the web search results are personalized and the user search experience is improved.

In [7], the user profile is constructed to do adaptive web search. However, a lot of computation such as profile construction and result reranking is involved so that this method can not be integrated with the search toolbar to provide online adaptive web search. In UCAIR toolbar, the query expansion and result reranking are done in a very efficient way. The user does not feel apparently longer response delay compared with the general search engine toolbars. The user also has the freedom to control whether the query expansion and result reranking are executed implicitly or not.

In our study of this project, we also capture and model user interactions. We focus on the clickthrough the user makes. Given each clickthrough, the original clustering result presentation will be dynamically changed.

3. Clustering Algorithm

In this section, we discuss the clustering algorithms, the document similarity measure and the cluster result representation.

3.1 Clustering Algorithm

3.1.1 K-Medoids

K-Medoids (or PAM) [2] is a partition based clustering algorithm. “Medoid” is the most centrally located object in a cluster. K-Medoids starts with an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it reduces the total distance of the resulting cluster. K-Medoids is more robust than K-Means in presence of noise and outliers because a medoid is less influenced by outliers than a mean.

Algorithm 1 outlines the document clustering process using a K-Medoids approach.

Algorithm 1 Document Clustering: K-Medoids

Input: A collection of document $\{D_i\}$,

Number of representatives K ,

Output: A set of medoid documents D_{C_1}, \dots, D_{C_k} .

- 1: randomly select K documents as the initial cluster centers;
 - 2: **for** each document D_i **do**
 assign its membership to the cluster C_j that has the
 largest similarity $\text{sim}(\vec{D}_i, \vec{D}_{C_j})$;
 - 3: find the most centrally located document in each cluster;
 - 4: repeat Lines 2-3 til small change in total sum of similarity;
 - 5: **return**;
-

It takes $O(k(n - k)^2)$ for each iteration in K-Medoids, where k is the number of clusters and n is the number of

documents. Multiple iterations are needed before convergence. The output is k documents as representatives of each cluster center.

3.1.2 Hierarchical Clustering

We also tried hierarchical clustering algorithm. It merges two most similar clusters at each level until there are only k clusters left or other user specified criterion satisfies.

Algorithm 2 outlines the document clustering process using a hierarchical clustering algorithm.

Algorithm 2 Document Clustering: Hierarchical Clustering

Input: A collection of document $\{D_i\}$,

Number of representatives K ,

Output: A set of term frequency vectors $TF_{C_1}, \dots, TF_{C_k}$.

- 1: initialize $k = n$ clusters, each of which has one document;
 - 2: compute the pairwise similarity among C_1, \dots, C_k ,
 $sim_{ij} = sim(\vec{D}_i, \vec{D}_j)$;
 - 3: **while** ($k > K$)
 - 4: select sim_{st} such that $s, t = \operatorname{argmax}_{i,j} sim_{ij}$;
 - 5: merge clusters C_s and C_t to a new cluster C ;
 - 6: $TF_C = TF_{C_s} + TF_{C_t}$;
 - 7: calculate the similarity between C and the remaining clusters;
 - 8: $k = k - 1$;
 - 9: **return**;
-

The initial similarity computation takes $O(n^2)$ time. We can maintain a sorted similarity list for each document in $O(n \log n)$. For n documents, the total cost is $O(n^2 \log n)$. Whenever two clusters are merged into a new cluster C , a new similarity list is created and sorted in time $O(n \log n)$. So the total time complexity is $O(n^2 + n^2 \log n)$.

The output of Algorithm 2 is k term frequency vectors representing the cluster centers.

We implemented both K-Medoid and Hierarchical clustering algorithm into theUCAIR toolbar. We did a lot of testing. Their effectiveness are both relatively good, in the sense that they can really cluster some similar search results. The number of clusters is hard to decided beforehand. On the efficiency side, we find that K-Medoid clustering is not efficient at all. It is the bottleneck ofUCAIR toolbar computation. For example, it generally takes 3 seconds to cluster the top 50 search results while all other computation ofUCAIR generally takes less than 1 second. On the other hand, Hierarchical clustering is very efficient and the computation time of hierarchical clustering can be neglected comparing with other computation such as that of indexing the search results. So when we do the study of

incremental clustering and dynamical clustering update, we all use hierarchical clustering algorithm.

3.2 Document Similarity Measure

We use the term frequency vector to represent a document. Using this representation, we can measure the similarity between two documents based on the similarity of their term vectors. Several measures are available to fulfill this requirement. Cosine similarity is a widely used one. It measures the normalized similarity between the term vectors of two documents.

$$sim(\vec{D}_i, \vec{D}_j) = \frac{\sum_{p=1}^N w_{ip} \cdot w_{jp}}{\sqrt{\sum_{p=1}^N w_{ip}^2 \cdot \sum_{p=1}^N w_{jp}^2}} \quad (1)$$

where $\vec{D}_i = (w_{i1}, \dots, w_{iN})$ is the term frequency vector representation of document D_i and N is the size of vocabulary.

3.3 Clustering Result Presentation

Another important task in this paper is to study different clustering result presentation strategies. Clustering process organizes the whole collection of documents into different groups. Clear and effective presentation format provides users with good interpretability and understanding.

In our project, we present the ranked list of documents based on personalization. In addition, we replace ‘‘Google Sponsored Link’’ with the clustering results. We study the different strategies to present clustering results.

For K-Medoids approach, we present the central document in each cluster. In this case, the user will clearly see an representative example of a cluster. However, the user maybe still can not see the main theme of this cluster.

For hierarchical clustering approach, we present the central term frequency vector in each cluster. A term vector usually has a very high dimension. For effectiveness, we only present the top- K frequent (or distinctive) terms. In this case, the user will not see a specific search result.

4 Incremental Clustering

In theUCAIR toolbar, while the user navigates, the toolbar will continue downloading search results. When we get some results and partition them into clusters while new search results are downloaded at the same time, how should we update the cluster? One solution is to do the clustering from scratch (reclustering) using all search results downloaded. However, this solution is not efficient. In this project, we proposed an algorithm called FixInc to do incremental clustering. It works as follows.

Step 1. Partition the initial set of search results into clusters;

Step 2. As new results come, assign each of them to a cluster where the similarity of the new document and the cluster center is the highest.

Our assumption is, if the initial set of search results (usually are top-ranked pages by Google) can relatively represent the cluster partition well, the following results can be assigned into the clusters with a greedy assignment.

We evaluate the FixInc algorithm against the reclustering algorithm in terms of clustering time and cluster quality. The cluster quality is defined as

$$Q = \sum_{i=0}^k \sum_j sim(\vec{D}_{ij}, \vec{C}_i) \quad (2)$$

The higher Q is, the better the clustering quality. Figures 1 and 2 show the clustering time and cluster quality of FixInc and reclustering. It is shown that FixInc is much more efficient than reclustering from scratch but the quality is very close. We compared the clustering time and cluster quality of several queries, the conclusions are consistent.

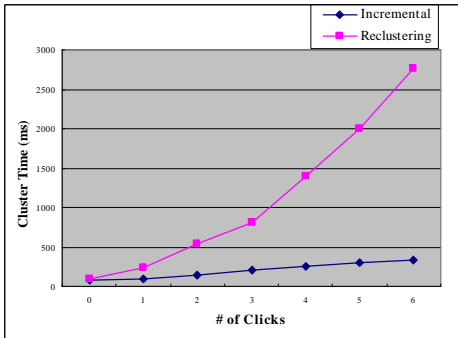


Figure 1. Cluster time of FixInc and Reclustering

5 Dynamic Clustering Maintenance

Most existing clustering algorithms of search results are “static”, in the sense that the cluster structure and presentation are fixed. But in interactive information retrieval such as web search, the search system can not clearly infer the user information need at first. However, through the user search engine interaction, the user will provide more and more hints about information need. Moreover, user information need sometimes is drifting during the information seeking process, which is especially true when the user

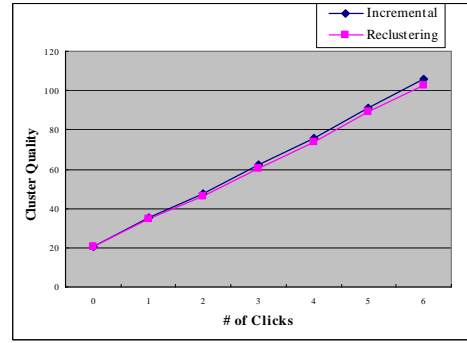


Figure 2. Cluster Quality of FixInc and Reclustering

himself has no clear idea about what he is really searching. Thus not only should the search results be personalized (e.g. through reranking), but also the cluster results should be personalized.

We design and implement the FishEye View algorithm to do the dynamic clustering. Fisheye View is a technique frequently used in information visualization [1], which integrates details in the neighborhood smoothly in wider context. When we applies the Fisheye View into the dynamical clustering, we show more details about the results which are most similar with the most recently user clickthrough while we select and show several other similar clusters, although they do not contain the search result the user just clicked. It works as follows.

Step 1. Partition the initial set of search results into clusters using traditional clustering algorithms such as hierarchical clustering algorithm;

Step 2. As the user clicks a search result and then click “Back” (maybe then “Next” button), do an incremental clustering to include more recently downloaded search results.

Step 3. Pick the cluster which contains the search result the user just clicked. First pick the most similar search results belonging to same cluster as that containing the clicked search result and show them in details. Then, we rank the other clusters according to the similarity between the cluster and the clickthrough. We show them in context.

Here we can show as details the summary and title of each shown search results and show as context top frequent terms.

We implemented the Fisheye View algorithm into the UCAIR toolbar. The best evaluation strategy is to do some user study to measure whether the Fisheye View algorithm can really help the user understand the search results. But

many existing work have verified that, in many cases, Fish-eye View help the user view and understand the information [1].

6. Related Work

There are some study of clustering web search results ([9] and [10]). [10] models the clustering problem as a salient phrase ranking problem. This work takes a supervised learning approach by building a regression model based on human labelled training data. Given a query and ranked documents as search results, this method extracts and ranks salient phrases as candidate cluster name. The collection of documents are assigned to relevant salient phrases to form candidate clusters. The final clusters are generated by merging those candidate clusters.

However, our work has the following unique characteristics.

- We focus the study on the efficiency of different web search result presentation as well as the effectiveness. Most, if not all, previous work on web result clustering do not study the efficiency issue. We try to find a very fast clustering algorithm which is also effective.
- Search result presentation will be personalized. For example, when we use clustering strategy, clusters will dynamically change during the user interaction with the search toolbar.

7. Conclusion and Future Work

In this project, we study how to provide the effective and efficient clustering search results to the user at the client sides. Moreover, we design and implement FixInc algorithm to do incremental clustering and Fish-eye View algorithm to do dynamical clustering. We have done some quantitative analysis about efficiency of clustering algorithm and incremental clustering and quality of incremental clustering. We will do a user study to evaluate the effectiveness of clustering algorithm and dynamically update of clusters.

There are a lot of interesting work to explore. First, we will explore other dynamically clustering algorithms. Currently, we just use the most recent clickthrough as the clue to update the clusters. Actually there are more information the search system can make use of. We will also explore other incremental clustering algorithms. Currently, the cluster structure of FixInc algorithm will not change and only contents of each cluster are updated. We will study how to change the cluster structure when we do incremental clustering.

References

- [1] G. W. Furnas. Generalized fisheye views. In *Proceedings of CHI*, 1986.
- [2] L. Kaufman and P. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Son, 1990.
- [3] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 2003.
- [4] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1998.
- [5] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*, 1998.
- [6] X. Shen, B. Tan, and C. Zhai. Intelligent search using implicit user model. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005.
- [7] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of WWW 2004*, 2004.
- [8] Vivisimo. <http://vivisimo.com>.
- [9] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. In *Proceeding of WWW 1999*, 1999.
- [10] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *Proceeding of SIGIR 2004*, 2004.