ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Search-Personalizer: Adaptive dynamic document clustering based search results personalization

T.Youva vyshnavi*
M.Tech Department of CSE,
ATRI, Parvathapur, Uppal,
Hyderabad, India.
youvavyshnavi@gmail.com

T.Nagalakshmi
Sr,Asst.prof, Department of CSE,
ATRI, Parvathapur, Uppal,
Hyderabad, India.
nlakshmi.t@gmail.com

D.Sujatha
Assoc.Prof and HOD Department of CSE
ATRI, Parvathapur, Uppal,
Hyderabad, India.
sujatha.dandu@gmail.com

Abstract: Current search engines present the user a ranked list given the submitted user query. Top ranked search results generally cover few aspects .In many cases the users are interested in the main themes of search results besides the ranked list in order to have a global view. This is often achieved through clustering approaches. Personalized search studies ranking or re-ranking them based on implicit feedback and it also infer user information need based on user search engine interaction and re-rank the search results. Same as this, clustering's 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 and the results should change dynamically to reflect the personalized ranking of search results. Traditional static clustering algorithms based on document similarity cannot achieve this. This paper deals how to incrementally cluster the search results and dynamically update the cluster representation based on user's implicit feedback.

Keywords: search, personalization, clustering, hierarchical.

I. INTRODUCTION

Search engines rank most relevant search results at top. Most of them achieved this by some well-known algorithms like Page Rank [5] and HITS [4]. In many occasions users want to have a global picture of the themes about search results such as clustered result presentation. These 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 re-ranks the general ranked search results based on the created user model, which reflects user short-term or long-term interest. In the personalization framework, new challenges for clustering search results emerge since search results is interactively re-ranked based on user's feedback. So efficient clustering algorithmic desired to perform online clustering of search results. In addition, clustering results should dynamically update to reflect the re-ranking based on user's implicit feedback. Traditional clustering algorithms based on document similarity (for example, cosine similarity) are 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 the Search-personalizer [6]. Just like query expansion and dynamic result re-ranking of search result based on user implicit feedback [3], the cluster presentations also personalized, i.e. dynamically change based on user implicit feedback such as submitted query and click through during user interaction with the search-personalizer. 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 re-ranked. We will reorganize the cluster presentation after we get the user implicit feedback. SEARCH-PERSONALIZER framework is introduced in Section II 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.

An incremental clustering algorithm to incrementally update the cluster structure was developed.

Design and implementation of Fisheye View algorithm to dynamically change the cluster structure based on user's implicit feedback was developed.

The rest of the paper comprises of. Section II introduces the Search-personalizer, which deals with the dynamic clustering algorithm. Similarity measure, the clustering algorithm and clustering presentation are detailed in Section III. Sections IV and V describe the design and implementation of incremental clustering (Inc-DC) algorithm and dynamical clustering (Fisheye View) algorithm. Related work is discussed in Section VI and conclusion in Section VII.

II. PROPOSED FRAMEWORK

This framework provides a search results personalization option at the client side, which can be referred as search-personalizer. It provides the basic functionality that many search engine toolbars such as Google toolbar provide. After the user composes query, the search-personalizer communicates with the search engine and retrieves the search result web page so that the he does not need to visit the search engine web sited search. Besides the basic functionality, Search-personalizer does the query expansion and result re-ranking to improve the retrieval quality.

For the query expansion, when the user submits a query, say, "java", the search-personalizer 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 search-personalizer 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 re-ranking, 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 re-ranked according to the title and summary of previously clicked web pages. 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. Originally relevant web pages ranked lower by the search engine will be pushed up. The user's previous queries and previous click through provide the implicit feedback [3] to the Search-personalizer.

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. A lot of computation such as profile construction and result re-ranking is involved so that this method cannot be integrated with the search-personalizer to provide online adaptive web search. In Search-personalizer, the query expansion and result re-ranking are done in a very efficient way. The user does not feel apparently longer response delay compared with the general search engine toolbars; he has the freedom to control whether the query expansion and result re-ranking are executed implicitly or not. Capture and model user interactions are done in through this people. This thesis focuses on the click through the user makes. Each click through, the original clustering result presentation will be dynamically changed.

III. CLUSTERING ALGORITHM

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

A. *Clustering Algorithm:*

a. *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's-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.

It takes $O(k(n ¡ k)2)$ for each iteration for algorithm referred in figure 1, 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.

b. *Hierarchical Clustering:*

We also tried hierarchical clustering algorithm. It merges two most similar clusters at each level until there are only clusters left or other user specified criterion satisfies. Algorithm 2 outlines the document clustering process using a hierarchical clustering algorithm.

The initial similarity computation takes $O(n^2)$ time. We can maintain a sorted similarity list for each document in O(nlogn). For n documents, the total cost is $O(n^2 \log n)$ .Whenever two clusters are 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 the Search-personalizer. We did a lot of testing. Their effectiveness is both relatively good, in the sense that they can really cluster some similar search results. The number of clusters is hard to decide beforehand. On the efficiency side, we find that K-Medoid clustering is not efficient at all. It is the bottleneck of Search-personalizer computation. For example, it generally takes 3 seconds to cluster the top 50 search results while all other computation of search-personalizer 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.

ALGORITHM 1: DOCUMENT CLUSTERING ALGORITHM

```
Input : Select set of documents to be clustered as input dataset $\{D_i\}$ and
        number of representatives as K
Output: A set of documents $D_{c1}, D_{c2}, ....D_{ck}$
Select cluster centroids, which are randomly picked documents, total number of centroids are K
 foreach docuement D do
 find similariy with all cluster centroids
add document to the cluster c, which is emergedas nearer in similarity check
verify and adjust the centroids of documents
End of foreach
End of the function
```

ALGORITHM II: HIERARCHICAL DOCUMENT CLUSTERING ALGORITHM

```
Input : $D_j$ is a set of documents
Number of representatives K
Output :Termfrequency Vectors $V_{tf}, V_{tfk}$
initialize $k = n$ clusters such that each cluster contains atleat one document
Compute pairwise similarity among clusters
$sim_{ij} = similarity(D_i, D_j)$
until $k <= K$ do
select $sim_{st}$; here $s, t = \arg\max_{ij} sim_{ij}$;
merge clusters $c_s$ and $c_t$ to a new cluster c;
$TF_c = TF_{c_s} + TF_{c_t}$;
continue the similarity check between c and other clusters
$k = k - 1$;
end of until
end of function
```

B. 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.

$$sim(D_i, D_j) = \frac{\sum_{p=1}^{N} w_{ip.w_{jp}}}{\sqrt{\sum_{p=1}^{N} w_{ip}^2 . \sum_{p=1}^{N} w_{jp}^2}} \quad (1)$$

Where $D_i = (w_i^1;;;; w_i^N)$ is the term frequency vector representation of document *Di* and *N* is the size of vocabulary.

C. 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 representative example of a cluster.

However, the user maybe still cannot 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.

IV. INCREMENTAL CLUSTERING

In the Search-personalizer, while the user navigates, the search-personalizer 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 (re-clustering) using all search results downloaded. However, this solution is not efficient. In this project, we proposed an incremental algorithm called incremental document clustering algorithm also referred as inc-DC 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 Inc-DC algorithm against the re-clustering algorithm in terms of time and cluster quality. The cluster quality is defined as 12.

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

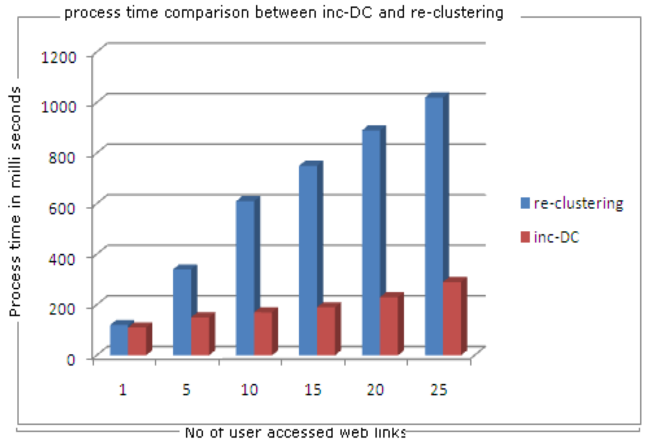

Figure 1. Cluster Time of Inc-DC and Re clustering

V. ADAPTIVE CLUSTER MAINTENANCE

Most prevailing clustering algorithms of search results are "static", which means that the cluster structure and presentation are stable. But in interactive data reclamation such as web search, the search system cannot visibly infer the user information need at first. However, through the user search engine interaction, the user will provide more and more hints about information needed. Moreover, user information need sometimes is drifting during the information pursuing process, which is especially true when the user himself is not sure of what he is probing for. Thus both the search results (re-ranking) and cluster results need to be personalized.

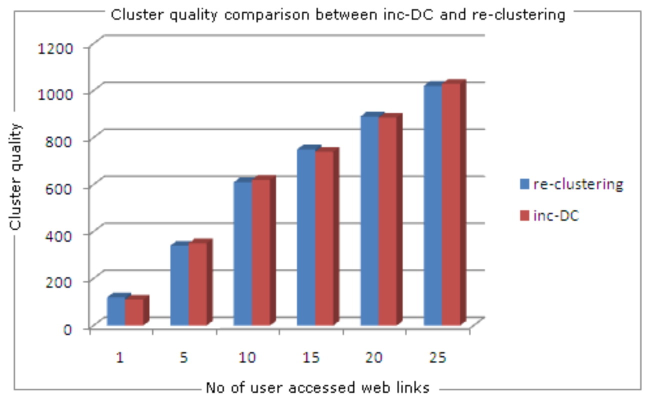

Figure 2. Cluster Quality of Inc-DC and Re clustering

We design and implement the Fish Eye View algorithm to do the dynamic clustering. Fisheye View is a technique recurrently used in information visualization [1], which integrates details in the neighborhood smoothly in wider framework. When we apply the Fisheye View in the dynamical clustering, we show details about the results which are similar with the most recently user click through while we select and show other similar clusters as well, even though they do not contain the search results 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 clicks "Back" (maybe then "Next" button), an incremental clustering to include more recently downloaded search results is done.

Step 3. Pick the cluster which encloses 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 added clusters according to the similarity between the cluster and the click through. We show them in context.

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

We implemented the Fisheye View algorithm into the Search-personalizer. The best evaluation strategy is to do some analysis to measure whether the Fisheye View algorithm can really help the user understand the search results.

Many existing works verified that, in many cases, Fisheye View helps the user view and understand the information [1].

VI. RELATED WORK

There are studies of clustering web search results ([9] and [10]). [10] Models the clustering problems as salient phrase ranking problem. This work takes a supervised learning approach by building a regression model based on human labeled training documents. Given a query and ranked data as search results, this method excerpts and ranks salient phrase as candidate cluster name. The data collected 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. ^{2} We emphasize 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 does not study the efficiency issue. We try to find a very fast clustering algorithm which is also effective. ² Search result presentation will be personalized. For example, when we use clustering strategy, clusters will dynamically change during the user interaction with the search search-personalizer.

VII. 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 Inc-DC algorithm to do incremental clustering and Fisheye 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 interesting works to explore. First, we will explore other dynamically clustering algorithms. Currently, we just use the most recent click through as the clue to update the clusters. Actually there is more information on the search system we can make use of. We will also reconnoiter other incremental clustering algorithms. At present, the cluster structure of Inc-DC algorithm will not change and only contents of single cluster are updated. We will study how to change the cluster structure when we do incremental clustering.

VIII. REFERENCES

[1]. G. W. Furnas. Generalized fisheye views. In *Proceedings of CHI*,, Volume 17 Issue 4, April 1986, New York, NY, USA ©1986 ,pp. 16-23.

[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, Volume 37 Issue 2, 2003, New York, NY, USA, pp. 18 - 28

[4]. J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, Volume 46 Issue 5, Sept. 1999, pp. 604 - 632.

[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, New York, NY, USA ©2004. pp. 675 - 684

[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*, vol.31, 1999, pp. 11-16.

[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*, New York, NY, USA ©2004, pp. 210-217.

Short Bio Data for the Author’s

T.Youva Vyshnavi completed B.Tech degree in cse from Aizza College of eng & Tech, mancherial, India, Affiliated to JNTU in 2009. Her date of birth is July 9th.She is pursuing M.Tech with S.E as specialization at Aurora’s Technological and research institute, Hyderabad, India Affiliated to JNTU. Her expertise includes Knowledge Mining &Data Eng.

T.Nagalakshmi completed M.Tech degree from Aurora’s Technological and research institute, Hyderabad, India Affiliated to JNTU born on 25th August 1984. She is having 5 years of teaching experience. Presently she is working as Asst.Professor in the Department of CSE at Aurora’s Technological and Research Institute, Hyderabad.