CS229 Project Report: Improving Search Engine For A Digital Library

Farnaz Ronaghi Khameneh
farnaaz@stanford.edu

***Abstract*— This project introduces a novel approach for using click through data to discover clusters of similar queries and similar URLs. One can simply observe that users reformulate their queries to find a desirable result. We define this sequence of queries as a query chain. Our data set consists of records containing user id, query term, query date and time, clicked item URL and clicked item rank if there exists one. Viewing this data set as a bipartite graph we perform a graph-based agglomerative clustering to find similar queries and similar URLs. We intend to use human intelligence in expressing his information need as a prior knowledge in agglomerative clustering which will result in an earlier convergence and a noticeable improvement in whole system performance. We also describe how to validate this model and how to use query-URL clusters in a real-world search engines.**

***Index Terms*— Machine Learning, Information Retrieval**

I. Introduction

Textual information in our digital library, Library of Congress, forms a massive unstructured data set. With the increasing size of data available in such library, there is an acute need for automatic methods to organize data. Most text clustering algorithms represent document as vectors in a high dimensional space and they put them together in a cluster according to a distance measure [6]. The algorithm that we use for query-URL clustering is basically content ignorant. Similarity measures based on Co-occurrence information from query logs lead the clustering technique [1].

Intuition behind using agglomerative clustering is based on two observations. First, users express their information need with different terms but they may select the same URL as the most relevant result. This shows that these two different queries are expressing the same information need. Second, users may search for same query but click on URLs which seem to be totally unrelated using measures like cosine distance; using latent information in query logs will assist in finding this set of related URLs.

As users search, it is well documented that they try to reformulate their queries instead of switching to a new search engine [2]. We call Sequence of query reformulations, a query chain. To the best of my knowledge researches have considered queries independently for ranking judgments. [7] uses query chains to rerank search engine results by extracting implicit preference judgment from clicked items in a query chain, they assume to have the top ten results returned by search engine which is not available in our data set.

The key contribution of this work is recognizing that we can successfully use evidence of query chains that is present in search engine log files to enhance graph based agglomerative clustering, and improving it with a more reasonable similarity measure. In section II we will propose a method to extract query chains automatically from a search engine log file. Section III will talk about agglomerative clustering algorithm. Section V and IV will give experimental results and discuss various uses of this technique.

II. Extracting Query Chains

In order to make use of human intelligence to express his information need, we should be able to detect query chains. In this section we propose an algorithm for automatically extracting query chains and judge its effectiveness.

[7] uses the simple heuristic that any two queries from the same user which follow each other in less than 30 minutes are in a query chain. Another approach for detecting query chains is training an SVM which needs labeled data. In [7] researchers have manually detected query chains in a log file containing 1285 queries. They have used these chains as a basis for comparing different evaluation

methods. They have trained an SVM using feature vector parameters as described in table I.

TABLE I
PARAMETERS TO TRAIN AN SVM FOR DETECTING QUERY CHAINS.

CosineDistance(q1, q2)
CosineDistance(doc ids of r1, doc ids of r2)
CosineDistance(abstracts of r1, abstracts of r2)
TrigramMatch(q1,q2)
ShareOneWord(q1, q2)
ShareTwoWords(q1, q2)
SharePhraseOfTwoWords(q1, q2)
NumberOfDifferentWords(q1, q2)
$t2 - t1 \leq \{5, 10, 30, 100\}$
seconds $t2 - t1 > 100$ seconds
NormalizedNumberOfClicks(r1)

Using five-fold cross validation for Comparing their 30-minute heuristic and trained SVM with respect to query chains extracted manually, SVM training out performs the heuristic algorithm only for 4 percent. Computing this feature vector for every pair of queries is so expensive, so using the simple heuristic will be more reasonable.

Using 30-minute heuristic will result in grouping lots of unrelated queries in the same query chain. Users may search for another topic without any noticeable time gap between the two. To avoid this problem we add a new simple heuristic. We find the common longest subsequence (LCS) of the two query terms, if the result was larger than a preset threshold we will add them to the same query chain. To extract query chains we make a graph of query terms. There will be a link between two terms if they satisfy 30-minute heuristic and LCS conditions. Query chains are the connected components of this graph.

III. AGGLOMERATIVE GRAPH-BASED CLUSTERING

Clustering queries submitted to a search engine appears to be a very explored task. Query clustering has been used in personalizing search, query suggestion modules and spell correction tasks. Researches on user behavior show that most users try to reformulate their queries in a search engine instead of switching to another search engine. Moreover study of search engine usage patterns shows that have begun to rely more on one-word queries and expect search engines help them in finding their actual information need.

Clustering URLs becomes useful in personalizing search engines. Researches have been investigating the more general problem of document clustering. In this group of algorithms documents are represented as vectors in a high dimensional space.Hierarchical and flat clustering algorithms place documents in the same cluster based on a distance measure in this space. In this project we are going to use a graph-based hierarchical clustering proposed in [1] with a more reasonable similarity measure. According to [1] content-aware algorithms may fail in clustering:

- *Text-free pages*: distance functions based on the vector representation of documents fail on documents which only contain images
- *Pages with restricted access*: Pages may be password protected and their content unavailable to a clustering algorithm.
- *Pages with dynamic content*: A clustering algorithm based on content is more susceptible to placing these URLs in the same cluster

The most important advantage of agglomerative clustering proposed in [1] is that it is content ignorant, URLs are placed in the same cluster based on co-occurance measures. It is not vulnerable to the problems given above.

Algorithm proposed in [1] starts by making a bipartite graph of queries as white nodes and URLs as black node. There will be a link between a query and a URL if at least one user has clicked on that URL as the relevant result for the corresponding query. Intuitively if we define $N(x)$ neighborhood of x and $N(y)$ neighborhood of y, similarity of two nodes x and y should be proportional to $N(x)$ and $N(y)$ overlap. The basic similarity measure proposed in [reference] is as follows:

$$\sigma(x, y) = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|} \tag{1}$$

To use this graph to discover separate clusters of queries that express the same information need or URL clusters that contain URLs related to similar information needs, [1] propose the following algorithm:

Agglomerative Iterative Clustering

1) *Score all pairs of white vertices according to equation 1*

2) *Merge the two vertices with the largest similarity measure*
3) *Score all pairs of black vertices according to equation 1*
4) *Merge two black vertices with the largest similarity measure*
5) *exit if convergence criterion holds else go to step 1*

At first glance it might not be clear that why an iterative approach is necessary. In figure 1 after merging vertices 1 and 2, vertices A and C will suddenly be similar refer to 2. This iterative approach discovers hidden information faster.

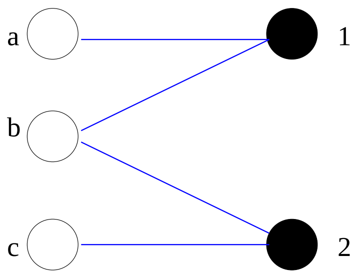

Fig. 1. Necessity of Iterative Clustering

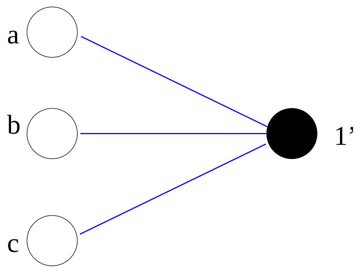

Fig. 2. Necessity of Iterative Clustering

Similarity between vertices will lie between zero and one. One of the major problems with this similarity measure is that it does not distinguish between two vertices having exactly the same neighborhood and vertices having exactly the same two neighborhood. the second problem is that merging two clusters will result in deleting repeated vertices which destroys valuable information about vertex worth in clusters. We can give weight to vertices in a cluster according to the number of times they appear in that cluster. Vertex repetition in clusters shows closer relation cluster and vertex. URL's many repetitions in a cluster representing a special information need, discloses a close relationship between the URL and regarding information need. We use this information to make equation 1 as the sum of intersection members' weights over the sum of union members' weights. This information can further be used for reranking search engine results.

Exploring search engine log files, we found a large number of queries with no selected item URL. These queries happen so often, they are either a miss spelling or user's inability to express his information need. These will result in having isolated vertices in our bipartite graph. Agglomerative clustering algorithm will be unable to place them in any of the resulting clusters as long as there is no co-occurance information about them. Here is the time where query chains show their ability in using human intelligence. Using our prior knowledge on user query reformulations, we will merge all query vertices in the same query chain. This will eliminate isolated vertices. Further more it will result in earlier convergence of agglomerative clustering algorithm.

IV. Usage

Clusters resulting from agglomerative clustering algorithm either represent queries expressing the same information need or URLs which are related to the same information need. Noticing to the fact that query reformulations can result from misspellings. Query clusters can be used for spelling correction purposes.

On the other hand queries in the same cluster represent the same information need. We can use this fact in making query reformulation suggestion systems. [1] used clusters for this purpose and compared their result with existing suggestion system by using user implicit feedback.

We use the resulting query and URL clusters for reranking search engine results. The scenario is as follows:

- User searches for a new term.
- Assign user's search item to one of existing query clusters.
- Retrieve basic search engine results.
- Delete all URLs in the associated URL cluster from basic results.
- Rank URLs according their weight in associated URL cluster.
- Add ranked URLs on top of basic search engine results.

Assigning user's search term to one of existing clusters has a great impact on final ranking. First we compute LCS for query term and all terms in all clusters. We choose the term having maximum LCS as the cluster representative. We select the cluster for which LCS of its representative and query term is maximum. Intuitively ranking URLs in associated cluster according to their weight and adding them to the top of search engine basic results will end in a better ranking. As long as we know queries in the associated cluster are the best matches with user information need. Further more their weight is a good clue of how close they are to user's information need. We expect this system result in more user satisfaction.

V. Experimental Results

This report was supposed to introduce an algorithm to improve a digital library search engine. The library under study is Library Of Congress, a governmental library. There exist specific law for disclosure of its' query log information. We have used AOL search engine published data. It is very huge data set consisting of ten files. We use one file for training and one file for testing, statistics on training data can be found in table II. These query logs are unfiltered, we have filtered them to eliminate objectionable data.

TABLE II
Statistics on training data

Number of queries	3616245
Number of query chains	620155
Number of terms	1229209
Number of URLs	386295

Standard ways of testing user satisfaction from ranking functions always involve running a public search engine for some months. Ask users for explicit judgements or extract judgments implicitly from deployed search engine log files[5], [3]. We didn't have time to do so. We have a novel approach for comparing our model to the basic agglomerative clustering algorithm.

We use one of search engine log files as a virtual online search engine for testing proposed ranking approach. We have the following information about each query in the log file. If user has clicked on any item, we know that item and its rank. If user has reformulated his query we have his query chain so we know which URLs he has clicked on. According to the fact that reformulating queries most of the time is a means for expressing unsatisfied information needs, it is reasonable to give more score to URLs clicked on at the end of a query chain[4]. So our testing approach will be as follows:

- Retrieve a new query from log file.
- Find its chain.
- Assign it to one of query clusters according to proposed mechanism.
- Rank associated URL cluster members according to proposed approach.
- Give one score if clicked items have better rank in proposed approach along with paying attention to the rules given above

This method for testing can only determine number of times that one algorithm outperforms the basic search engine's approach based on given assumptions on use behavior. Without having an online search engine and logging user's implicit feedback by methods proposed in [5], we can not tell anything about actual user satisfaction in each of two approaches. Although our method can compare user satisfaction in different methods. Using this approach our results are given in table III. Applying our method of testing basic agglomerative clustering improves user satisfaction 40 percent, adding query chain information will improve our result to 48 percent and enhancing similarity measure for clustering algorithm will improve result to 52 percent.

TABLE III
experimental results.

clustering proposed in [ref]	65%
add query chain information	73%
improve similarity measure	76%

According to our unusual method of evaluation, in some cases we can not say that we definitely have improved search engine ranking. Noticing that 80 percent of users only view the first two result pages, we can claim that as a whole we have improved user satisfaction on these two pages. We view results that are clicked URLs mostly having a rank between one to ten in basic search engine ranking. On the other hand these results have been proved to be related to user's information need. As results show it is reasonable to claim we are improving user satisfaction in terms of ranking results. We believe testing this approach with standard measures prove it to be affective and will give more meaningful results on its performance.

VI. Conclusion

In this project we used two heuristics to find sequences of user query reformulations named as query chains from search engine log files. We made a bipartite graph from query terms on one side and URLs on the other side. We ran a graph-based agglomerative clustering along with a more reasonable similarity measure to find clusters of URLs and clusters of query terms. Clusters of query terms depict various formulations of an information need. On the other hand, clusters of URLs consist of URLs related to the same information need. Using these facts along with historical studies of user search behavior, we showed a way to improve user satisfaction mostly in terms of ranking.There are lots of queries for which our basic search engine did not return any satisfactory result. Whereas by using this method we are returning are URLs which are expected to be the most related to user information need and in some sense we can say we are improving precision.

The key contribution of this project was understanding the fact that our prior knowledge on query chains can make a major improvement in agglomerative clustering algorithm. This can be used in many application for improving search engine results.

VII. Acknowledgements

Author would like to thank Amin Saberi for his guidance throughout this project, and Nathan Sakunkoo for his suggestions on data set.

References

[1] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416. ACM Press, 2000.

[2] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[3] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666, New York, NY, USA, 2008. ACM.

[4] Laura A. Granka, Thorsten Joachims, and Geri Gay. Eye-tracking analysis of user behavior in www search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 478–479, New York, NY, USA, 2004. ACM.

[5] Thorsten Joachims. Evaluating retrieval performance using clickthrough data. In *In Proceedings of the SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, pages 79–96, 2002.

[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, July 2008.

[7] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback, 2005.