IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

 Volume 10, Iss 2 , 2021

Customized Search Results using web snippets

Dr. N Murali Krishna
Professor & HOD
Department of Computer Science and Engineering
AVN Institute of Engineering and Technology, Hyderabad, India
JNTU (H) India E-mail: muralinamana@gmail.com

Abstract

The exponential growth of information on the Web has introduced new challenges for building effective search engines. A major problem of web search is that search queries are usually short and ambiguous, and thus are insufficient for specifying the precise user needs In this paper, we introduce an effective approach that captures the user's conceptual preferences in order to provide personalized query suggestions. We achieve this goal with two new strategies. First, we develop online techniques that extract concepts from the web-snippets of the search result returned from a query and use the concepts to identify related queries for that query. Second, we propose a new two-phase personalized agglomerative clustering algorithm that is able to generate personalized query clusters.

1. INTRODUCTION

The amount of information available on the web is growing rapidly. Google [4] reported that its index size was over 8 billion pages in 2004, and it was esti-mated that it had 20 billion pages in 2005. As the web keeps expanding, the number of pages indexed in a search engine increases correspondingly. With such a large volume of data, finding relevant information satis-fying user needs based on simple search queries becomes an increasingly difficult task.

Queries submitted by search engine users tend to be short and ambiguous. A study by M. Jansen [20] found that the average query length on a popular search engine was only 2.35 terms. These short queries are not likely to be able to precisely express what the user really needs. As a result, lots of pages retrieved may be irrelevant to the user needs because of the ambiguous queries.

In this paper, we propose a method that provides per-sonalized query suggestions based on a personalized con-cept-based clustering technique. In contrast to existing methods which provide the same suggestions to all users, our approach uses click through data to estimate user‘s conceptual preferences and then provides personalized query suggestions for each individual user according to his/her conceptual needs. The motivation of our research is that queries submitted to a search engine may have multiple meanings. For example, depending on the user, the query "apple" may refer to a fruit, the company Ap-ple Computer or the name of a person, etc. Thus, provid-ing personalized query suggestion (e.g. users interested in "apple" as a fruit get suggestions about fruit, while users interested in "apple" as a company get suggestions about the company's products) certainly helps users to formu-late more effective queries according to their needs.

The underlying idea of our proposed technique is based on concepts and their relations extracted from the submitted user queries, the web-snippets1 and the click-through data. Clickthrough data was exploited in the per-sonalized clustering process to identify user preferences: a user clicks on a search result mainly because the web-snippet contains a relevant topic which

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 2 , 2021

the user is inter-ested in. Moreover, clickthrough data can be collected easily without imposing extra burden on users, and thus providing a low-cost means to capture user's interest.

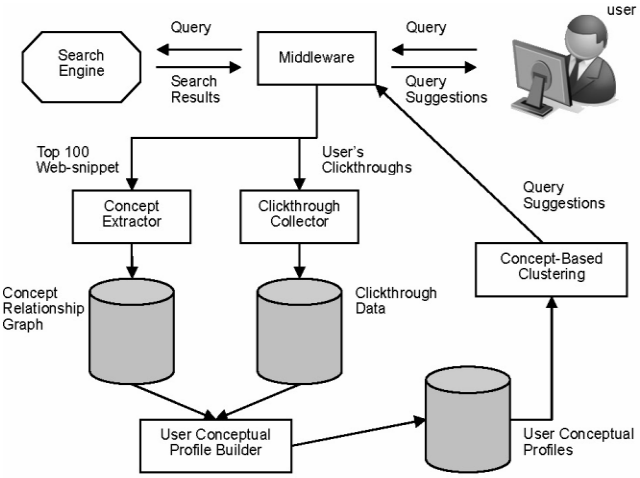

Fig. 1. The general process of concept-based clustering

Our approach consists of the following four major steps. First, when a user submits a query, concepts (i.e. important terms or phrases in web-snippets) and their relations are mined online from web-snippets to build a concept relationship graph. Second, clickthroughs are collected to predict user's conceptual preferences. Third, the concept relationship graph together with the user's conceptual preferences is used as input to a concept-based clustering algorithm that finds conceptually close queries. Finally, the most similar queries are suggested to the user for search refinement. Fig. 1 shows the general process of our approach.

2.RELATED WORK

Query clustering techniques have been developed in di-versified ways. The very first query clustering technique comes from information retrieval studies . Similarity between queries was measured based on overlapping keywords or phrases in the queries. Each query is repre-sented as a keyword vector.

Chuang and Chien proposed to cluster and organ-ize users' queries into a hierarchical structure of topic classes. A Hierarchical Agglomerative Clustering (HAC) algorithm is first employed to construct a binary-tree cluster hierarchy. The binary-tree hierarchy is then parti-tioned in order to create sub-hierarchies forming a multi-way-tree cluster hierarchy like the hierarchical organiza-tion of Yahoo and DMOZ .

Later on, Smyth et al. suggested that user search behaviour is repetitive and regular. They proposed to rerank search results such that the results which have been previously selected for a given query are promoted ahead of other search results.

More recently, Deng et al. proposed an algorithm which combines a spying technique together with a novel voting procedure to de-termine user preferences from the clickthrough data. Dou et al. also performed a large scale evaluation on dif-ferent personalized search strategies, including click-through-based and profile-based personalization. They suggested that click-based personalization strategies per-form consistently and considerably well when compared to profile-based methods.

3.CONCEPT EXTRACTION

Before explaining our concept-based clustering method, we first describe our concept extraction method, which is composed of the following three basic steps: 1) extracting concepts using the web-snippets returned from the search engine, 2) mining concept relations, and 3) creating a

ISSN PRINT 2319 1775 Online 2320 7876
 Volume 10, Iss 2 , 2021

user concept preference profile using the extracted concepts, concept relations and user ‘s clickthroughs

3.1 Concept Extraction Using Web-Snippets

Our concept extraction method is inspired by the well-known problem of finding frequent item sets in data min-ing . When a user submits a query to the search engine, a set of web-snippets are returned to the user for

EXTRACTED CONCEPTS FOR THE QUERY “APPLE”

Concept t_i	*support*(t_i)	Concept t_i	*support*(t_i)
mac	0.1	macintosh	0.05
ipod	0.1	tour	0.05
iphone	0.1	slashdot apple	0.04
hardware	0.09	picture	0.04
software	0.09	apple ii	0.04
big apple	0.08	apple variety	0.04
apple store	0.06	music	0.04
mac os	0.06	farm market	0.04
apple orchard	0.06	apple grower	0.04
apple valley	0.06	gift shop	0.04
apple and macintosh	0.06	apple farm	0.04
apple blossom			

identifying the relevant items. We assume that if a key-word or a phrase appears frequently in the web-snippets of a particular query, it represents an important concept related to the query because it co-exists in close proximity with the query in the top documents.

4 Mining Concept Relations

To find relations between concepts, we apply a well-known signal-to-noise ratio formula from data mining [16] to establish similarity between terms *t1* and *t2*. The similarity value of Church and Hanks' formula always lies between [0,1], and thus can be used directly in Step

Given a query q, and a URL u, let Pop (q, u) be the popularity of u (fraction of clicks) in the answers of q. Let T f(t, u) be the number of occurrences of term t in URL u. We define a vector representation for q, q, where q[i] is the i-th component of the vector associated to the i-th term of the vocabulary (all different words), as follows:

$$q[i] = \sum_{URL\,u} \frac{\mathrm{Pop}(q,u) \times \mathtt{Tf}(t_i,u)}{\max_t \mathtt{Tf}(t,u)}$$

where the sum ranges over all clicked URLs. Note that our representation changes the inverse document frequency by click popularity in the classical tf-idf weighting scheme. Different notions of vector similarity (e.g., cosine function or Pearson correlation) can be applied over the proposed vectorial representation of queries. Here we use the cosine function, which considers two documents similar if they have similar proportions of occurrences of words (but could have different length or word occurrence ordering).

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 2 , 2021

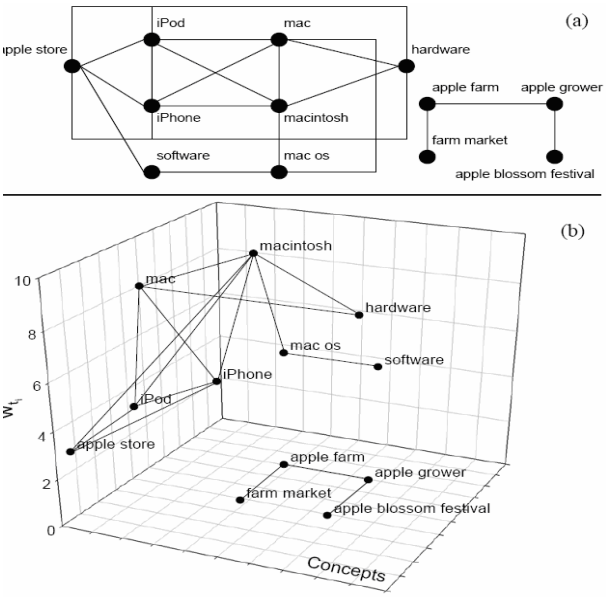

Fig. 5. (a) A concept relationship graph for the query "apple" derived without incorporating user clickthroughs. (b) A concept preference profile constructed using the user clickthroughs and the concept relationship graph in (a). *wti* is the interestingness of the concept *ti*to the user. More
clicks on a concept gradually increase the interestingness *wti* of the concept.

5 Creating User Concept Preference Profile

The concept relationship graph is firstly derived without taking user clickthroughs into account. Intuitively, the graph shows the possible concept space arising from user's queries. The concept space, in general, covers more than what the user actually wants. For example, when the user searches for the query "apple", the concept space derived from the web-snippets contains concepts such as "ipod", "iphone" and "recipe".

If the user is indeed inter-ested in the concept "recipe" and clicks on pages contain-ing the concept "recipe", the clickthroughs should gradu-ally favor the concept "recipe" and its neighborhood (by assigning higher weights to the nodes), but the weights of the unrelated concepts such as "iphone", "ipod" and their neighborhood should remain zero. Therefore, we propose the following formulas to capture user's interestingness *wti* on the extracted concepts *ti* when a clicked web-snippet *sj*, denoted by click(*sj*), is found:

5. Discovering Related Queries

Our algorithm considers only queries that appear in the query-log. A single query (list of terms) may be submitted to the search engine several times, and each submission of the query induces a different query session. Here we use a simple notion of query session similar to the notion introduced by Wen et al which consists of a query, along with the URLs clicked in its answer.

QuerySession: = (query, (clickedURL)_)

A more detailed notion of query session may consider the rank of each clicked URL and the answer page in which the URL appears, among other data that can be considered for improved versions of the algorithm. The query recommender algorithm operates in the following steps:

1.Queries along with the text of their clicked URL's extracted from the Web log are clustered. This is a preprocessing phase of the algorithm that can be conducted at periodical and regular intervals.

2.Given an input query (i.e., a query submitted to the search engine) we first find the cluster to which the input query belongs. Then we compute a rank score for each query in the cluster.

3.Finally, the related queries are returned ordered according to their rank score.

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 2 , 2021

The rank score of a related query measures its interest and is obtained by combining the following notions:

1.Similarity of the query. The similarity of the query to the input query. It is measured using the notion of similarity.

2.Support of the query. This is a measure of how relevant is the query in the cluster.

We measure the support of the query as the fraction of the documents returned by the query that captured the attention of users (clicked documents). It is estimated from the query log as well.

One may consider the number of times the query has been submitted as the support of a query. However, by analyzing the logs in our experiments we found popular queries whose answers are of little interest to users. In order to avoid this problem we define the support of a query as the fraction of clicks in answers of the query.

As an example, the query rental offices has a low popularity (2.52%) in its cluster, but users in the cluster found this query very effective, as its support in Figure 3.1 shows. The similarity and support of a query can be normalized, and then linearly combined, yielding the rank score of the query. Another approach may consider to output a list of suggestions showing the two measures to users, and to let them tune the weight of each measure for the final rank.

5.2 Query clustering

5.2.1 Query Similarity

In order to compute the similarity of two queries, we first build a term-weight vector for each query. Our vocabulary is the set of all different words in the clicked URLs. Stopwords (frequent words) are eliminated from the vocabulary considered. Each term is weighted according to the number of occurrences and the number of clicks of the documents in which the term appears.

Given a query q, and a URL u, let Pop (q, u) be the popularity of u (fraction of clicks) in the answers of q. Let T f(t, u) be the number of occurrences of term t in URL u. We define a vector representation for q, q, where q[i] is the i-th component of the vector associated to the i-th term of the vocabulary (all different words), as follows:

$$q[i] = \sum_{URL\, u} \frac{\mathrm{Pop}(q,u) \times \mathrm{Tf}(t_i,u)}{\max_t \mathrm{Tf}(t,u)}$$

where the sum ranges over all clicked URLs. Note that our representation changes the inverse document frequency by click popularity in the classical tf-idf weighting scheme. Different notions of vector similarity (e.g., cosine function or Pearson correlation) can be applied over the proposed vectorial representation of queries. Here we use the cosine function, which considers two documents similar if they have similar proportions of occurrences of words (but could have different length or word occurrence ordering).

5.2.2 Computing the Clusters

We considered queries extracted from a 15-day query-log of the Todocl search engine. The log contains 6042 queries having clicks in their answers. There are 22190 clicks registered in the log, and these clicks are over 18527 different URL's. Thus in average users clicked 3.67 URL's per query. We compute the clusters by successive calls to a k-means algorithm, using the CLUTO software package4. We

ISSN PRINT 2319 1775 Online 2320 7876
Research paper **Volume 10, Iss 2 , 2021**

chose an implementation of a k-means algorithm for the simplicity and low computational cost of this approach, compared with other clustering algorithms. In addition, the k-means implementation chosen has shown good quality performance for document clustering. We refer the reader to for details.

The quality of the resulting clusters is measured using a criterion function, adopted by common implementations of a k-means algorithm. The function measures the total sum of the similarities between the vectors and the centroids of the cluster that are assigned to. Since in a single run of a k-means algorithm the number of clusters k is fixed, we determine the final number of clusters by performing successive runs of the algorithm. Figure 1 shows the quality of the clusters found for different values of k (function criterion FC). The curve below (DIFF) shows the incremental gain of the overall quality of the clusters. We selected k = 600, for which we obtain a 0.6 average distance of each point to its cluster centroid. We ran the clustering algorithm on a Pentium IV computer, with CPU clock rate of 2.4 GHz, 512MB RAM, and running Windows XP. The algorithm took 64 minutes to compute the 600 clusters.

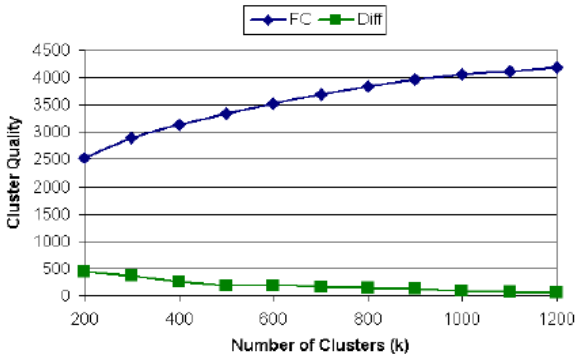

Fig. 3.1 Cluster quality vs. number of clusters.

Conclusion:

As search queries are ambiguous ,in this paper, we propose a new personalized con-cept-based clustering technique which is able to obtain personalized query suggestions for individual users based on their conceptual profiles. The technique makes use of clickthough data and the concept relationship graph mined from web-snippets, both of which can be captured at the backend and as such do not add extra burden to users. An adapted agglomerative clustering algorithm is employed for finding queries which are con-ceptually close to one another. Our experimental results confirm that our approach can successfully generate per-sonalized query suggestions according to individual user conceptual needs. There are several directions for extending the work in the future. First, instead of considering only query-concept pairs in the clickthrough data, we can consider the relationships between users, queries and concepts to obtain more personalized and accurate query suggestions.

REFEERENCES:

[1] http://www.cse.ust.hk/~dlee/tkde08/query.html.

[2] http://www.dmoz.org/.

[3] http://www.sigkdd.org/kdd2005/kddcup.html.

[4] Z. Zhang and O. Nasraoui, “Mining Search Engine Query Logs for Query Recommendation,” *Proc. of WWW Conference*, 2006.

[5] M. Jansen, A. Spink, J. Bateman, and T. Saracevic, “Real Life Informa-tion Retrieval: A Study of User Queries on the Web,” *ACM SIGIR Fo-rum*, vol. 32, pp. 5-17, 1998.

[6] T. Joachims, “Optimizing search engines using clickthrough data,” *Proc. of ACM SIGKDD Conference*, 2002.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **Volume 10, Iss 2 , 2021**

[7] Chen, G. and Choi, B. (2008**)** 'Web page genre classification', *Proceedings of the ACM symposium*
[8] *on Applied computing*, pp. 2353-2357.

[9] Ricardo, B., Carlos, H., and Marcelo, M. (2004) 'Query recommendation using query logs in search
[10] engines', *EDBT Workshops*, pp. 588-596.

[11] Roelleke, T., and Wang, J. (2008) 'TF-IDF uncovered: a study of theories and probabilities.

[12] Tan, P., Steinbach, M., and Kumar, V. (2006) *Introduction to Data Mining: Concepts and*
[13] *Techniques*, Addison