An Approach to Web Query Recommendation System

Anshu Kumar, Bhaskar Biswas
Department of Computer Engineering, Indian Institute of Technology (BHU), Varanasi-221005 India
anshu.kumar.cse08@itbhu.ac.in, bhaskar.cse@itbhu.ac.in

Abstract—In this work, the authors address the problem of finding desirable and accurate information on the web by search engines due to two problems of low precision and low recall caused by significant and rapid growth in amount of information and the number of users. Based on the studies performed on query recommendation system using keyword extraction from the search results for a query, an algorithm is proposed to suggest alternate search topics which are related to the input query provided. Based on this cluster labels were identified and once these labels are distinguished, search results are assigned to them using frequency of occurrence of any particular label in the corresponding document.

Keywords—web mining, query recommendation system.

1. Introduction

Contrary to the popular belief, search engine don’t really answer questions. It is just a way to provide fast access to information people have put on web. In-fact popular search engines tend to present search results which match the user’s question and not the answer. The real problem arises when the information need of the topic is very large or too broad or simply ill defined. Then the range of topics covered becomes unimaginably large and very confusing for the user. This problem of finding the relevant information about any input query is often referred as Web Search Problem [1]: “the problem of finding all information on the Web relevant to a given query”. Due to recent explosion of data on internet, query recommendation system has become an essential requirement of any user-oriented search engine. Most of search engine gives user the option to provide the search query as a list of keywords. In [2], from detailed study of log of a popular search engine, Jansen concluded that an average of 2.84 queries are submitted per user per session and most of them are very short, which are more likely to be ambiguous. Users rarely use the advanced features like Boolean operators provided in search engines. [3] shows that less than 5% of users actually use the Boolean operators in their search queries. Therefore, query recommendation is very necessary for users to formulate queries. Users searching for some information may formulate their queries in different ways. In order to formulate effective queries, user may need to be familiar with the terminologies of the knowledge domain of the information. But this is not always the case. Sometimes user has very little knowledge of the information domain – even zero sometimes. For these cases, we can use the queries which were previously used by similar searching who may by refining query through iterations, turned into expert users. Queries submitted by these expert users in their query sessions can be used to help those who are novices in the field.

Clustering of search results also seems much promising approach for making the search results even more understandable to the user. When taking in context of web search results, query clustering aims at rearranging the results into clearly defined thematic groups. In this way, the naïve user who is very inexperienced to formulate an effective query can be presented with many related topics to the field and can be helped to pick a more effective search query. Also advanced users who may be formulating a query to have an insight about the sub topics or related topics about a particular domain will not have to scan hundreds of documents to have a clear picture of the topics. Related sub topics can also be presented to user as cluster labels. While creating clusters of search results, the emphasis should be on the speed with which the clusters are generated. In the past, several clustering engines for web search results have been implemented. In [4], Sadikov et.al proposed an approach by performing multiple random walks on a Markov graph that approximates user search behavior. In [5], Wen et.al describes an attempt to cluster similar queries according to their contents as well as user logs. Carrot2 Clustering Engine [6], which is an open source search engine for clustering of search queries into topics and iBoogie [7] are excellent examples of search engines which divide the search results into well-defined thematic groups. They help user to identify the related topics of a particular query and also provide documents containing information about those topics.
An Approach to Web Query Recommendation System

The data on the web is huge in amount, distributed and heterogeneous, unstructured and dynamic which adds another dimension to the problem. The main goal of this work is to suggest an algorithm for recommending queries to the user and dividing them into clusters to let user understand more about the subtopics along with their associated documents. Formally describing the problem statement: From the user submitted query to a search engine, (which may not be sufficient to extract the most relevant information of user’s interest) suggest some other related sub topics which can help user to understand more about the knowledge domain of the query and may help an inexperienced user to formulate more efficient queries.

One of the components in search engines is search result presentation. Detailed study of web search result was done in [8]. One of the popular methods of presentation is Ranked List presentation which has some disadvantages. Users may have to sift through a long list of documents, some of which are irrelevant. The reason a document was included in the results and the relation of a document to the query is not explicit. No explicit information is provided about the relationships between documents on the list.

Clustering of web search results is an attempt to organize the results into a number of thematic groups in the manner a web directory does it. The main requirements of such clustering algorithms are listed in [9], which are Relevance, Browsable Summaries, Overlapping clusters, Snippet tolerance, Speed, Incremental processing. Carrot2 [6] is a search engine that organizes search results into a hierarchical structure of thematic groups.

2. Algorithm

When designing a web clustering algorithm, special attention must be paid to ensuring that both contents and description (labels) of the resulting groups are meaningful to the users. The majority of currently used text clustering algorithms follows a scheme where cluster content discovery is performed first, and then, based on the content, the labels are determined.

The general idea behind our algorithm is to first find the meaningful clusters using the documents shown in search results of the search engine by making use of abstract concepts discussed in the documents as cluster labels and then associating documents to them. While associating documents to clusters, we should note that a document may contain information relating to more than one topic so, there should not be a constraint that a document may only belong to only one cluster. Specifically, a document should belong to multiple clusters with different weights.

```java
/** Phase 1: Preprocessing */
for each document
{
    do text filtering;
    apply stemming;
    mark stop words;
}
/** Phase 2: Feature extraction */
discover frequent terms and phrases from each document;

/** Phase 3: Cluster label induction */
Count frequency on the phrases over all documents;
Select top results of the most frequent phrases;

/** Phase 3: Cluster content discovery */
for each cluster label{
    add the documents containing the label in decreasing order of frequency;
}
```

Among all term co-occurrences in a collection of documents, frequent phrases seem to be most meaningful to the users, being at the same time relatively inexpensive to find. As observed in they are very often collocations or proper names, which make them both informative and concise. That is why, we have decided to use the most frequent phrases over all document as the cluster labels. To decide the document which should belong to the cluster, we take the cluster label (which is actually an abstract concept that is being talked about) and find its frequency in the respective document. Document having highest frequency comes higher in the cluster.
components and a document that does not contain any occurrence of the concept in it does not belong to the cluster. To become a full-featured clustering algorithm, the process of finding cluster labels and contents must be preceded by some preprocessing of the input collection. This stage should encompass text filtering, document’s language recognition, stemming and stop words identification. It is also recommended that post-processing of the resulting clusters be performed to eliminate groups with identical contents and to merge the overlapping ones. The main phases of our algorithm are:

2.1. **Input to the algorithm**

For any query passed to the algorithm, we pass the query to a search engine using HTTP request. Currently we are passing the query to Google Search API REST Service [10] and it sends back the result in JSON format. One such response for a request of ‘data mining’ is shown in Fig 1. One important issue is the number of results to fetch from search engine. While very less number of documents gives very inaccurate cluster labels, high number of document put a constraint on algorithm runtime and data structure limitations. Very less number of document gives inaccurate result because it may be possible that an abstract concept is not present in much of them and this may be overridden by some other phrase which although is present in more number of documents but is not important. Fetching more number of documents and processing them takes more time and the requirement of it being online and fast put timing restrictions on that. Due to these reasons, we tried different combinations and saw that it is the middle value which given the best result, being around 100. After fetching the JSON response from Google, we send the request for getting the source of all these URLs in different threads.

![Figure 1: JSON response from Google for a search query ‘Data Mining’](image)

2.2. **Text Preprocessing**

In web search result clustering and query recommendation it is the web snippets and document source which act as the input to the algorithm. Therefore the need of proper preprocessing of data becomes very important. And if we are talking about the frequencies of text phrases involved, it becomes further more important removing stopwords and certain phrases and tags from the document source. Having done this poorly, results will contain very poor cluster labels thus making them useless. Thus, the primary aim of the preprocessing phase is to remove all characters and terms from the input documents that can possibly affect the quality of clusters.

In the text filtering step, all terms that are useless or would introduce noise in cluster labels are removed from the input documents. Porter’s algorithm [11] is then used for stemming. After all the words are reduced to their stem forms, stopword removal [12] and phrase removal is done. While getting the HTML source of web pages as input to the algorithm, there are some phrases like “All rights reserved” or “Copyright 2012” etc. which actually has no meaning in context of the document and these should not be ignored while looking for the abstract concepts in the page. For this purpose a list of phrases which should be ignored is created and they are also removed from the source.

2.3. **Cluster Label Generation**

The output of text preprocessing phase is a set of words which contain only important words referring to the context of the document. Bases on this set of keywords, different phrases are identified and they along with their frequency are kept in a sorted hashmap. For e.g. such a hashmap for a document may look like
After maps are generated for all the web documents, the phrases are stored in a separate map. The frequency associated with the phrase is counted one per document it appears in. This is done by iterating over all the maps and checking the entries in the common map. If the entry is not present, it is entered into the common map with a value of 1, else its frequency is incremented by 1. After this is done, in final map we have entries from all the document sources. Entries having more frequency value depicts that the phrase occurred in more number of documents that the entries having less value. Thus entries present at top of the common map may act as cluster labels for generated clusters.

Choosing appropriate number of documents and top results from common map gives sufficiently good phrases to act as cluster labels. These labels are identified and then there weight with the all the documents are compared. Document having more frequency of any cluster label indicates that particular phrase has occurred repetitively in the document and it may have some sufficient information enabling the document to belong to that category. A document may have information relating it to more than one cluster depending on the frequency it has for the cluster label. We then prepare a document – phrase matrix. Rows in this matrix depicts document and column in this matrix depicts a phrase. An entry in the matrix tells us the frequency with which the phrase appearing in column appears in the document appearing in the row. For e.g. choosing top 5 entries from common map will result in a document - phrase matrix shown in Fig 4.

The row vector in this matrix is the membership of a document in the selected cluster labels. For e.g. for document 1 the associated row membership values are \( \{1, 2, 2, 1, 0\} \). We then normalize these values along separate rows, so that each membership value is a value in between \([0, 1]\). More the value more the association of the document to the cluster label.
After this is done, documents are attached to the cluster labels in decreasing order of the frequency of phrase in them. For e.g. phrase 1 occurs in all the documents. Document 3 is most related to phrase 1 and document 1 is least related to it. Phrases 3, 4 and 5 occur in only 1 document each. Document 1 has content related to 4 topics whereas document 3 has specific content related to ‘data mining’ only.

3. Experiment

For evaluating our methods, we used the documents suggested by google search for a query of “data mining”. All the cluster labels identified are presented here in the form of a tag cloud so as to enable user to distinguish between most discussed topics with the least discussed one. For the user query of “data mining”, our algorithm was able to extract labels like data analysis, data warehousing, knowledge discovery, data mining tools etc. Complete result is shown in Fig 6.

![Figure 6 Tag cloud showing suggestion for a query of ‘data mining’](image)

While most of the labels generated represent valid abstract concepts related to the field of search query, a few of them are present as noise. This is due to the following facts:

1. We cannot accurately mine the abstract concepts discussed in the document. Although, the method of frequent phrases gives sufficiently good result, this method is not full proof.

2. Our algorithm mines the frequent phrases all over the documents and then choose cluster labels based on the results. Due to this, even if a phrase is not actually valid but is coming in many of the search results, it is assumed to be a valid topic by the algorithm.

4. Conclusion and Future Works

Being a very young modification to search engines, web search results’ clustering is becoming increasingly promising. On the one hand, extensive research is being conducted on faster and more precise clustering algorithms, and on the other, the scientific advances are followed by practical implementations of the web search clustering idea. We conclude this work by stating that although the algorithm that we have proposed that makes use of phrase frequency to generate the cluster labels, the quality of the cluster labels can be further improved by using the markedness of phrase in place of its frequency. The concept of markedness is introduced in [13]. This approach ensures that the visual information present in the node is taken care of while assigning the weight to the words. The designer of the web site generally put different weights to all the texts according to the context in which they are appearing. For example, section headings are bold and font size differs from rest of the page and styling features may vary. We believe that using this information together with the frequency of occurrence of phrase will assign more weight to probable cluster labels. Thus the quality of cluster labels can be further improved.
An Approach to Web Query Recommendation System

In this paper we addressed an algorithm for clustering of web search results while turns out to be fairly effective but there are still some problems which are left to be managed. Currently we are downloading the page on the system and processing the source code to suggest alternative queries and generate clusters. The effect of this is twofold. Firstly, it will cause additional load on server to download all the pages using different threads and data structures will also be used for this may causing memory mismanagement. Additionally, this will take more time to generate the search results. Non-patient user needs the results to be generated immediately. This can be addressed using only web search results snippets for generation of clusters and not the complete source code.

Currently the cluster labels which are generated are independent of each other. Our algorithm may generate clusters which are too general or too large. We can improve this by using some hierarchical clustering algorithm which will generate the clusters in hierarchical format.

References