OLAP-based Query Recommendation*

Carlos Garcia-Alvarado
University of Houston
Dept. of Computer Science
Houston, TX, USA

Zhibo Chen
University of Houston
Dept. of Computer Science
Houston, TX, USA

Carlos Ordonez
University of Houston
Dept. of Computer Science
Houston, TX, USA

ABSTRACT

Query recommendation is an invaluable tool for enabling users to speed up their searches. In this paper, we present algorithms for generating query suggestions, assuming no previous knowledge of the collection. We developed an online OLAP algorithm to generate query suggestions for the users based on the frequency of the keywords in the selected documents and the correlation between the keywords in the collection. In addition, performance and scalability experiments of these algorithms are presented as proof of their feasibility. We also present sampling as an additional approach for improving performance by using approximate results. We show valid recommendations as a result of combinations generated using the correlations between the keywords. The online OLAP algorithm is also compared with the well-known Apriori algorithm and found to be faster only when simple computations were performed in smaller collections with a few keywords. On the other hand, OLAP showed a more stable behavior between collections, and allows us to have more complex policies during the aggregation and term combinations. Additionally, sampling showed improvement in the time without a significant change on the suggested queries, and proved to be an accurate alternative with a few small samples.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Data Warehouse and Repository*

General Terms

Algorithms, Experimentation

Keywords

OLAP, IR, Query Processing

1. INTRODUCTION

There have been numerous approaches for improving the user experience while browsing digital collections [9]. ,Query recommendation has been used extensively to provide clever hints or suggestions for concepts related to a given query [4]. Query recommendations are usually based on the analysis of "known queries" stored in query logs. Nevertheless, query suggestion without any prior information or "unbiased query suggestion" requires the analysis of vast amounts of data, usually performed in a precomputed way (e.g. clustering). As a result, an online analysis of the selected documents may look like an unfeasible option. The base assumption is that users will often start their searches with naïve queries, which are somehow related to the target query (e.g. "Chicago Bulls" can lead to "Michael Jordan"), and the target query is reached when the user begins to understand the collection. Given this, the resulting documents can be analyzed to return a set of queries that will allow the users to pick any of the suggestions to refine their searches.

In order to provide such recommendation features, we exploit the knowledge discovery properties of On-Line Analytical Processing (OLAP)[7]. OLAP can be used in query recommendation to build data cubes containing the frequency of candidate sets of keywords to be suggested to the user. A similarly efficient approach is conducted by using the well-known Apriori algorithm, widely used in association rules. Although both methods are quite efficient in the computation of aggregations, performing such a task in large collections may be time prohibitive, especially if an immediate result is needed. Therefore, we decided to use sampling as an alternative for managing large collections in a "vanilla" universe. In addition, the validity of our recommendations is improved by the computation of the correlation of the terms in the collection and analysis of the frequency of occurrence of the recommended terms in the subset of documents.

This paper is divided as follows: In Section 2, we introduce the definitions. Section 3 presents algorithms used for obtaining valid query recommendations. Section 5 discusses previous work in this area. Finally, Section 6 contains our final remarks and future work.

2. DEFINITIONS

Let us focus on defining the notation. Let C be a collection, or corpus, of n documents $\{d_1, d_2, \ldots, d_n\}$. Each document d_i or query q is composed of a set of terms t_{ij}. In addition, let x_i be a frequency vector of all the different keywords in C for each document d_i. The collection is stored in an inverted list format within the table vtf that contains the document id, term and position in the i^{th} document. A summarization table tf is computed from vtf as table with the document id i, the term t, and the term frequency f stored. In addition, let $\hat{d}$ be a subset of documents, $\widehat{tf}$ a

*This research work was partially supported by NSF grants CCF 0937562 and IIS 0914861.

http://doi.acm.org/10.1145/1871437.1871619

subset of terms from tf, and $\widehat{vtf}$ a subset of vtf.

The backbone data structure for the OLAP data cubes is the dimensional lattice, which has a size of $2^{\hat{t}-1}$, where $\hat{t}$ is the number of terms. One level (or depth in the lattice) of the cube is denoted by $\binom{|\hat{t}|}{level}$, such that $level \leq |\hat{t}|$.

3. QUERY RECOMMENDATION

Our query recommendation analysis process is the result of the aggregation of the frequencies of a set of keywords (that have a positive correlation) appearing in a subset of documents. In practice, it is not feasible to obtain the combinations of all the keywords in a set of documents. There are often many keywords have a tractable problem. Thus, in order to speed up the execution of the process, we select the $\hat{d}$ most important documents of the collection based on an initial user query. Unfortunately, even with an abridged set of documents, the number of keywords in such documents can still be considered too overwhelming for efficient correlation analysis to be conducted. As a result, we also obtain the top-k keywords from these $\hat{d}$ documents and apply an importance metric. With these objectives, we adapted two different approaches: using the Apriori algorithm and using an online OLAP algorithm. Since the Apriori algorithm is well known, we will only cover how we altered the OLAP cube algorithm to efficiently compute recommendations.

3.1 Correlation and Metrics

The correlation will represent how related the frequency of one keyword is with another in terms of the whole collection. We adapted an efficient one-pass method to compute the correlation values of unique terms in the collection [6]. In addition to the previous definitions, let L be an additional set containing the total sum of all the different terms in the collection ($L = \sum_{i=1}^{n} x_i$). Moreover, let Q be a lower triangular matrix containing the squared measures between the terms ($Q = \sum_{i=1}^{n} x_i x_i^T$). In the last step, the correlation of a pair of terms a and b is obtained by applying $\rho_{ab} = \frac{nQ_{ab} - L_a L_b}{\sqrt{nQ_{aa} - L_a^2}\sqrt{nQ_{bb} - L_b^2}}$.

Once the combinations have been created and verified for having valid correlations, a final analysis of the importance of the concept can be performed by summarizing the occurrences of such combinations of terms in the subset of ranked documents. We propose three different techniques for ranking these recommendations: a single match method, a distance method, and a correlation method.

The **single match** technique is built on performing a single scan of the data counting all the occurrences based on the appearance of the set of terms in a document. Therefore, the occurrence of a combination will be represented by the **MIN** frequency of any of the terms in the combinations within a particular document. A user-defined threshold is used to filter the final results. The **distance match** method, also called proximity match, takes into account the position of the terms in the document and counts the number of times a set of selected terms in the combination appear within a certain user-defined distance, δ. As a result, highly-ranked recommendations will imply that the concept holds a stronger meaning due to the proximity of the terms. The **correlation method** uses the correlation table to make recommendations. In this approach, the validity of a set of keywords depends on its correlation with the initial query. We obtain the correlation between each keyword of the user query with each keyword of the candidate set. From these, an average correlation is then obtained to determine if the keyword set should be recorded. The difference between these three methods is in how the initial query is handled. For the frequency and distance methods, the initial query is used to filter the document collection into a more manageable size. While the same approach is used by the correlation method, it also takes into account the correlation between the initial user query and the keyword set candidates.

```
Input: level , t̂
1 : Init data structure S
2 : Init List_tf
3 : foreach<i,t̂,v> r in t̂f sorted by i
4 :     if i changed
5 :         List_tf ← {r.t, r.v}
6 :     else
7 :         C ← GetNextCombo(List_tf, level)
8 :         while isCorrelated (C)
9 :             if checkTolerance (v, tol)
10:                 S_C++
11:             else
12:                 S_C ← 1
13:                 C ← GetNextCombo(List_tf, level)
14: return S
```

Figure 1: One-level online OLAP Cube algorithm.

3.2 OLAP Cube

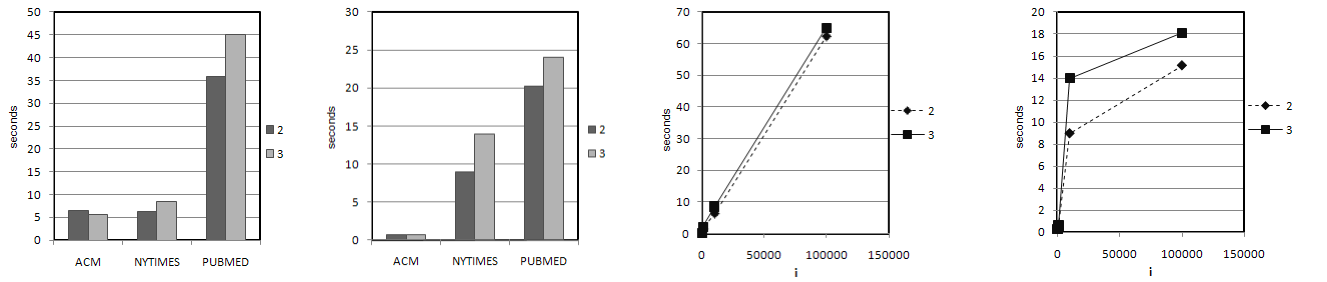
OLAP techniques can be used to efficiently obtain query recommendations. Figure 1 shows a general online algorithm for the computation of aggregations on keywords in the vertical format. In this algorithm, the structure S is a data structure that is used to store the combinations of keywords along with their frequency. The main input table for this algorithm is the keyword table, which always contains document id and keyword columns. The third column, v, changes depending on the final filtering method.

The validity of a set of keywords is determined by the filtering method: single match, distance, or correlation. For the single match method, v is the frequency of the keyword. We select the minimum frequency of the keywords to represent the set. The distance method bases the validity of a set of keywords on their positioning. In this case, v represents the position of the keyword in the document. The correlation method bases the validity on the correlation between the initial query and the sets of keywords. The average correlation of the initial query and the candidate set is obtained by retrieving them from the correlation tables. If multiple levels of the lattice are desired, the online OLAP algorithm can take a bottom-up or top-down approach.

While our algorithm may seem similar to the Apriori algorithm, there are two major differences. First, our algorithm uses main memory to compute all possible combinations using only one pass while the Apriori algorithm requires one pass for each group of itemsets. Second, our one-pass algorithm does not prune the results while the Apriori algorithm is able to prune between different iterations.

3.3 Sampling

To further improve the efficiency, we decided to explore the effect of sampling on both performance and accuracy.



a) Cube: i=10K,topk=5. b) Apriori: i=10K,topk=5. c) Cube: top-5 terms (NYT). d) Apriori: top-5 terms (NYT).

Figure 2: Full document scan performance for level 2 and 3.

The first step is to obtain a smaller list of documents, $\hat{d}$, based on $\phi(q, tf)$, where ϕ is the ranking function based on the query and the original set of documents. From this set, we draw τ samples, each having a $\hat{d} = \{d_1, d_2, \ldots, d_m\}$ with replacement from the result of $\phi(q, tf)$. The probability of any document in $\hat{d}$ of being selected in the τ samples is $P(1-(1-1/m)^{\tau})$. Each of the $\hat{d}$ samples of documents is then used as a source for obtaining its top-k keywords $\widehat{t}$ and computing the techniques. Finally, the scores for each combination are obtained as the average of the frequencies. Notice that the set of keywords $\widehat{t}$ is different between samples. We calculate accuracy as the percentage of total combinations of the top k keywords that are found through the course of the iterations.

4. EXPERIMENTS

The experiments were performed on an Intel Xeon E3110 server with 4 GB of RAM. The algorithms were implemented in SQL and UDFs. Four different collections were used during the experiments (see Table 1). The first collection is a subset of the ACM Digital Library. The NYTIMES and PUBMED were obtained from the UCI Repository. The Texas Water Well Data Set of Texas (TWWD) was used only for analyzing the quality of the results.

The quality of the results was validated in TWWD with ten initial keyword queries (air, america, dissolved, information, quality, residential, texas, treatment, waste, and water). With these queries, we recorded the query recommendations using the frequency, the distance, and the correlation. Table 2 shows the query recommendation returned by the initial query for "information". Of these three approaches, the correlation method definitely returned the most accurate results. In this case, we consider a recommendation accurate if a majority of the returned queries contains data similar to what a user might search for. The recommendations from the correlation method are almost always close to what the user initially queried while those from the other two methods are often more general. For the single and distance methods, the quality of their recommendations heavily depends on the initial query. We found an average of 25% overlap between these two recommendations. The remainder of the queries are very similar to one another, with only subtle differences.

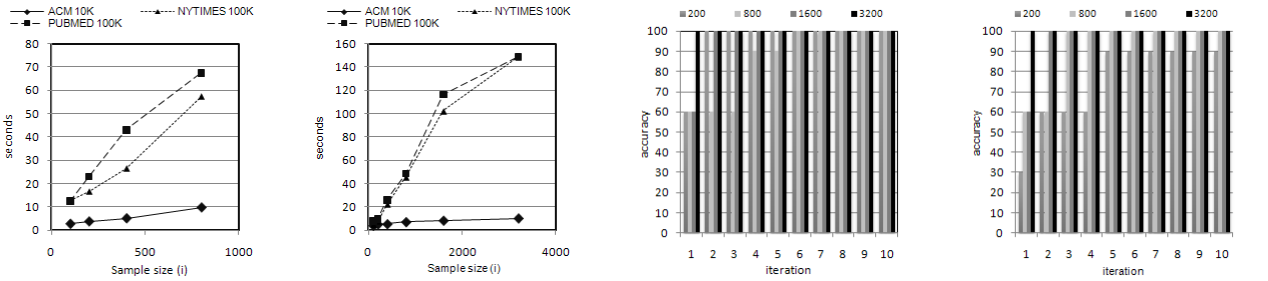
The set of plots in Figure 2 present the performance results of the Apriori and OLAP cube algorithms. Note that the execution time increases with the size of the collection because of the time required to perform the extraction of the $\hat{tf}$ subset. However, because the extraction of this subset is obtained differently in both algorithms, the performance change may also be different. Interestingly, such a difference between the two algorithms provides us with some intriguing trends. For example, the OLAP cube algorithm performs faster on collections in which the average number of keywords per document is large. In such situations, the Apriori algorithm is heavily penalized because of the extra matches that the algorithm must perform. Hence, we believe that OLAP proves to be a more stable option due to its ability to be fairly independent of the size of the collection. Having said this, the Apriori algorithm is still quite fast, especially when $\hat{tf}$ can be extracted quickly and managed in main memory. However, the OLAP approach is favored whenever the computations are complex.

Table 1: Collections.

Collection	C	Total t	Avg. t per d	$\lvert tf \rvert$
ACM	7675	23404	57	0.5M
NYTIMES	299752	92829	226	68M
PUBMED	8200000	134317	58	479M
TWWD	1002	129470	5926	6M

Figures 2c and 2d were focused on the execution time with respect to an increasing the number of documents. The results show that the online OLAP algorithm appears to have a linear reaction. For the Apriori algorithm, when the number documents to be analyzed is small, the performance trend is similar to n^2. However, as the number of documents analyzed increases, the algorithm becomes much more stable, and is significantly faster than the cube approach. For the online OLAP algorithm, we fully expected the performance to be linear (depends on the number of documents).

The execution times for 10K documents become unmanageable. Hence, we analyzed the effect of sampling on both overall performance and safety. Figure 3a and Figure 3b present the performance results of executing $\tau = 10$ iterations. Each iteration would only analyze a small portion of the overall original data set. From these experiments, the Apriori algorithm is slightly faster on data sets with a small average number of keywords per document. In addition, the OLAP Cube algorithm also appears to be more stable than the other options. However, the question arises regarding the validity of using sampling as an approximate solution. Figures 3c and 3d show results on the number of iterations required to arrive to the final solution. Surprisingly, only



a) Cube: Performance. b) Apriori: Performance. c) Cube: Accuracy (ACM). d) Apriori: Accuracy (ACM).

Figure 3: Sampling performance and accuracy with replacement ($\tau = 10$).

Table 2: Recommendations for query="Information" (ranked by frequency and correlation).

	Single Match		Distance Match		Correlation	
rank	combination	cnt	combination	cnt	combination	corr
1	water, system	368	quality, water	246	information, tceq	0.885
2	tceq, water	368	reduction, requirements	211	information, required	0.848
3	monitoring, water	328	water, surface	170	annual, information	0.848

a few iterations are needed (normally less than 5) to reach high accuracy. Hence, bootstrapping proves to be a viable alternative for reducing the times.

5. RELATED WORK

Recently, a term "popularity" approach is explored in [4] in which each term is compared to the query terms. Similarly, the authors present the idea of naïve suggestions, but they do not explore how to efficiently compute such aggregations. Early straightforward attempts on query logs for unbiased query recommendations where presented in [1, 3]. But the authors assume previous knowledge by using these logs. OLAP has been explored less than association rules for query recommendation. However, a couple of applications have implemented OLAP to explore document collections [8, 2]. In [8] a contextualized warehouse (XML data warehouse) of the documents was built, and then queried by the usage of OLAP on relevant measures assigned to the documents. However, they focus on the metadata of the documents. In addition, OLAP sampling was explored initially in [5] as a technique for speeding up OLAP aggregations with approximate results. However, this analysis is not presented in the context of collections of keywords.

6. CONCLUSIONS

We showed that OLAP can greatly enhance query recommendations. Through this paper, we have shown how the analysis of the intrinsic knowledge of keywords in a collection of documents can be efficiently performed. Our experimental results showed that while the Apriori algorithm is slightly faster than our online OLAP algorithm, the latter appears to be more stable in its times. Furthermore, we showed that even with a few iterations, sampling was able to obtain a fairly high accuracies. As a result, the experimental results proved that obtaining a few samples with replacement of large collection can achieve a high accuracy. We also created three main approaches to determining the validity of a set of keywords: single, distance, and correlation. We found that the single match method is the fastest while the correlation method produces better quality results.

Future work includes comparing our system to other query recommender systems and trying different policies for finding the occurrences of the set of keywords. Similarly, query suggestion enhanced with the injection of previous knowledge should be explored (e.g. ontologies). Finally, we want to study the effect of different sampling techniques.

7. REFERENCES

[1] B.M. Fonseca, P.B. Golgher, E.S. De Moura, and N. Ziviani. Using association rules to discover search engines related queries. In *LA-WEB*, pages 66–71, 2003.

[2] C. Garcia-Alvarado, Z. Chen, and C. Ordonez. OLAP with UDFs in digital libraries. In *CIKM*, pages 2073–2074, 2009.

[3] A. Giacometti, P. Marcel, E. Negre, and A. Soulet. Query recommendations for OLAP discovery driven analysis. In *ACM DOLAP*, pages 81–88, 2009.

[4] G. Koutrika, Z.M. Zadeh, and H. Garcia-Molina. Data clouds: summarizing keyword search results over structured data. In *EDBT*, pages 391–402, 2009.

[5] X. Li, J. Han, Z. Yin, J.G. Lee, and Y. Sun. Sampling cube: A framework for statistical OLAP over sampling data. In *Proc. of ACM SIGMOD*, pages 779–790, 2008.

[6] C. Ordonez. Statistical Model Computation with UDFs. *IEEE TKDE*, 22(12):1752–1765,2010.

[7] C. Ordonez and Z. Chen. Evaluating statistical tests on OLAP cubes to compare degree of disease. *IEEE TITB*, 13(5):756–765, 2009.

[8] J.M. Perez, R. Berlanga, M.J. Aramburu, and T.B.Pedersen. R-cubes: OLAP cubes contextualized with documents. In *ICDE*, pages 1477–1478, 2007.

[9] R. R. Kraft and J. Zien. Mining anchor text for query refinement. In *Proc. of WWW*, pages 666–674, 2004.