DoS: An Efficient Scheme for the Diversification of Multiple Search Results

Hina A. Khan
School of ITEE
The University of Queensland
Queensland, Australia
h.khan3@uq.edu.au

Marina Drosou *
Computer Science Dept.
University of Ioannina
Ioannina, Greece
mdrosou@cs.uoi.gr

Mohamed A. Sharaf
School of ITEE
The University of Queensland
Queensland, Australia
m.sharaf@uq.edu.au

ABSTRACT

Data diversification provides users with a concise and meaningful view of the results returned by search queries. In addition to taming the information overload, data diversification also provides the benefits of reducing data communication costs as well as enabling data exploration. The explosion of big data emphasizes the need for data diversification in modern data management platforms, especially for applications based on web, scientific, and business databases. Achieving effective diversification, however, is rather a challenging task due to the inherent high processing costs of current data diversification techniques. This challenge is further accentuated in a multi-user environment, in which multiple search queries are to be executed and diversified concurrently. In this paper, we propose the DoS scheme, which addresses the problem of scalable diversification of multiple search results. Our experimental evaluation shows the scalability exhibited by DoS under various workload settings, and the significant benefits it provides compared to sequential methods.

1. INTRODUCTION

Users interact with a number of applications by submitting queries to satisfy their information needs. Today, the explosion of the available data in many web, scientific and business domains dictates the need for the development of effective methods to assist users in quickly locating results of high interest. Towards this, *search result diversification* is one such method that has been widely employed to assist users in that direction [3]. In particular, search result diversification is to select an interesting representative subset of the available results.

Realizing an effective *diversification system*, however, is a rather challenging task. This is primarily due to the inherent high processing costs of current data diversification algorithms. This challenge is further complicated in a multi-user environment, in which multiple search queries are to be executed and diversified concurrently. For instance, in the Sloan Digital Sky Survey (SDSS) science database[1], the SDSS Catalog Archive Server (CAS) employs a multi-server, multi-queue batch job submission and tracking system [7]. From a system perspective, this leads to the simultaneous execution of a large number of queries that are essentially of exploratory nature [2]. Such an environment highlights the need for a scalable diversification system that is able to effectively facilitate data exploration tasks.

In this paper, we propose the *DoS* (Diversification of Multiple Search Results) scheme that addresses the problem of efficiently diversifying the results of multiple queries. Towards this goal, DoS leverages the natural overlap in search results in conjunction with the concurrent diversification of those overlapping results. This enables DoS to provide the same quality of diversification as that of the sequential methods, while significantly reducing the processing costs.

Our experimental evaluation on both real and synthetic data sets shows the scalability exhibited by DoS under various workload settings, and the significant benefits it provides compared to sequential methods.

Roadmap: We define search result diversification in Section 2. We introduce multiple search result diversification and our DoS scheme in section 3. The evaluation results are reported in Section 4, and we conclude in Section 5.

2. BACKGROUND AND RELATED WORK

In this work, we consider queries that retrieve a number of results, or items, from the database. Database results can be generally viewed as a set of attribute values represented as data points in a multi-dimensional data space. In such systems, a query is first processed against the stored database, and the generated results are then further processed in-memory through a diversification system.

Let $X = \{x_1, \ldots, x_m\}$ be a set of results for some user query. In general, the goal of result diversification is to select a subset S^* of X with $|S^*| = k$, $k \leq m$, such that the diversity of the results in S^* is maximized.

In this work, we primarily focus on the widely used content-based definition of diversity. Content diversity is an instance

*Work done while author was visiting the School of ITEE at the University of Queensland.

[1]http://www.sdss.org

SSDBM '13, July 29 - 31 2013, Baltimore, MD, USA

Algorithm 1 Greedy.

Input: A set of results X, an integer k.
Output: A set S with the k most diverse results in X.

1: $x_i \leftarrow$ random result $\in X$
2: $S \leftarrow \{x_i\}$
3: **while** $|S| < k$ **do**
4: $\quad x_i \leftarrow \text{argmax}_{x_i \in X} \frac{1}{|S|} \sum_{x_j \in S} d(x_i, x_j))$
5: $\quad S \leftarrow S \cup \{x_i\}$
6: **end while**
7: **return** S

of the *p-dispersion* problem [4], whose objective is to maximize the overall dissimilarity within a set of selected objects. In particular, given a metric d that measures the distance between two results, e.g., the Euclidean distance among two data points, the diversity of a set S is measured by a *diversity function* $f(S, d)$ that captures the dissimilarity between the results in S. To that end, a number of different diversity functions have been employed in the literature, among which previous research has mostly focused on measuring diversity based on either the *average* or the *minimum* of the pairwise distances between results [3, 9]. We focus on the first of those variants (i.e., average), as it adopts a more balanced view that considers all the results in S, defined as:

$$f(S, d) = \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j>i}^{k} d(x_i, x_j)$$

DEFINITION 1. *Let X be the set of results that satisfy a user query and k be a positive integer such that $k \leq |X|$. Let also d be a distance metric and f a diversity function. Then, Search Result Diversification is defined as selecting a subset S^* of X, such that:*

$$S^* = \underset{\substack{S \subseteq X \\ |S|=k}}{\text{argmax}} f(S, d)$$

In general, search result diversification has been shown to be NP-hard (e.g., [4]), which essentially renders the location of an optimally diverse set S^* impractical, especially for online search engines. Meanwhile, the need for low-cost (i.e., efficient) and high-accuracy (i.e., effective) solutions to the diversification problem has motivated a large body of research and led to the development of different practical heuristics (e.g., [9, 1]) for locating a near-optimal solution S to approximate S^*. Among those heuristics, a commonly used simple *Greedy* approach has been proven to provide a $1/2$-approximation of the optimal solution ([8]).

The Greedy heuristic initializes the diversified set S by first selecting a random result in X. Then, it proceeds with a number of iterations, until k results have been selected. At each iteration, the result with the maximum sum of pairwise distance from the already selected results is added to the diverse set S. Clearly, the complexity of this Greedy heuristic in terms of computed pairwise distances is $O(k^2|X|)$, which makes it a very reasonable choice in interactive online search environments. (see Algorithm 1.)

3. DIVERSIFICATION OF MULTIPLE SEARCH RESULTS (DoS)

In environments where multiple queries are submitted by a number of different users, we define the diversification of multiple search results as follows:

DEFINITION 2. *Let $\mathbb{Q} = \{Q_1, Q_2, \ldots, Q_n\}$ be a set of n user queries. Let X_i be the set of results that satisfy Q_i and k_i be a positive integer with $k_i \leq |X_i|$, $1 \leq i \leq n$. Let also d be a distance metric and f a diversity function. Then,* Multiple Search Result Diversification *is defined as selecting a set $\mathbb{S}^*$ of n subsets $\{S_1^*, S_2^*, \ldots, S_n^*\}$, such that:*

$$S_i^* = \underset{\substack{S_i \subseteq X_i \\ |S_i|=k_i}}{\text{argmax}} f(S_i, d)$$

According to Definition 2, a diversification system should ideally locate an optimal set of diverse subsets $\mathbb{S}^*$ for the input queries. However, due to the inherit NP-hardness of the diversification problem, this is not feasible. The success of any diversification system can be easily measured in terms of two parameters: total value of diversity across the set of diversified subsets and total data processing cost incurred to create the set of diversified subsets. In particular, our goal in this work is to develop diversification scheme that is able to diversify results efficiently with the quality of diversification as good as that provided by greedy heuristic. Next, we present two methods for the diversification of multiple search results namely, DoS-Naïve (Section 3.1), and *DoS-Overlap* (Section 3.2).

3.1 DoS-Naïve

A naïve solution to simultaneously diversifying multiple search results would be to retrieve the search results X_i for each query $Q_i \in \mathbb{Q}$ and then apply the Greedy heuristic (i.e., Algorithm 1) on each X_i separately to detect its respective diverse subset S_i. We call this method "DoS-Naïve".

Under this baseline approach, each search result is diversified independently and, thus, the complexity of DoS-Naïve is simply computed as: $O(k^2|X_1|) + \ldots + O(k^2|X_n|)$. Hence, the total processing cost of DoS-Naïve is:

$$C_{\text{DoS-Naïve}}(\mathbb{S}) = O(nk^2 \max_i |X_i|)$$

DoS-Naïve treats each user query independently, hence its complexity essentially increases linearly with the increase in the number of result sets to be diversified (i.e., n). In many real-life applications, however, it is often the case for many queries to have overlapping result sets.

EXAMPLE 1. *Consider two queries submitted simultaneously to the SDSS database. The first query, Q_1, retrieves all galaxies that are brighter than magnitude* 22*, given that local extinction is larger than* 0.75*. The second query, Q_2, retrieves all galaxies that are brighter than magnitude* 18*, given that local extinction is larger than* 0.85*. Both queries will retrieve all available results concerning galaxies brighter than magnitude* 22 *where the local extinction is greater than* 0.85*. Depending on the available data, there may be significant data overlap between the two result sets X_1 and X_2. This data overlap might potentially translate into further overlap between the two diverse subsets S_1, S_2.*

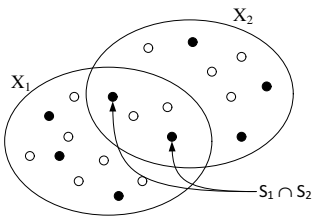


Figure 1: Result and diverse sets for two queries. Diverse items are shown in bold.

Our DoS scheme, described next, attempts to leverage this overlap among the result sets of the various queries for the efficient evaluation of their respective diverse subsets.

3.2 DoS-Overlap

In comparison to the Naïve approach, instead of processing the common (or shared) portions of the results multiple times, DoS-Overlap processes those portions only once leading to an overall amortized processing cost. In principle, DoS-Overlap can be perceived as an instance of the partial aggregation technique typically used in multiple query optimization (e.g., [6, 5]). Result diversification, however, introduces further complexities due to the dependency between each result set (i.e., X_i) and its respective diverse set (i.e., S_i) during the computation of distance functions. In particular, result diversification is an iterative process, in which a raw result set X_i is continuously processed in conjunction with its respective partial diverse set S_i until the final diversity goal is achieved. Handling such a dependency, while at the same time exploiting the opportunities of shared data processing, is the principle feature underlying DoS-Overlap.

DoS-Overlap (as shown in Algorithm 2) is based on the DoS-Naïve algorithm presented in Section 3.1. In DoS-Overlap, however, all the search results are processed concurrently and in each iteration, one item of X_i is selected for inclusion in S_i, $1 \leq i \leq n$. Towards leveraging the overlap in search results, we make the following two observations:

1. the distance function d between any item $x_j \in X_i$ and the items in S_i is computed independently of any other items in X_i, and

2. the distance function d between any item $x_j \in X_i$ and the items in S_i can be evaluated in parts.

To illustrate the second observation above, consider the following example:

EXAMPLE 2. *Assume two queries Q_1 and Q_2 and their respective result sets X_1 and X_2. Further, assume that in any arbitrary iteration of DoS-Overlap, S_1 and S_2 are the current diverse sets of X_1 and X_2, respectively. In such iteration, S_1 and S_2 might have shared results (as shown in Figure 1). In that case, S_1 can be clearly expressed as the difference between S_1 and S_2 union their intersection. That is, $S_1 = ((S_1 \backslash S_2) \cup (S_1 \cap S_2))$. S_2 can be expressed similarly.*

In the example above, for a single item $x_j \in X_1$, the distance function $d(x_j, S_1)$ can be computed as:

$$d(x_j, S_1) = d(x_j, S_1 \backslash S_2) + d(x_j, S_1 \cap S_2)$$

Algorithm 2 DoS-Overlap for N Queries.

Input: Result sets (X_1, $X_2 ... X_n$), an integer k.
Output: S_1, $S_2 ... S_n$ with the k most diverse items of (X_1, $X_2 ... X_n$) respectively.

```
1: X ← X1 ∪ X2 ... ∪ Xn
2: for all Queries do
3:    qi.S ← random x where x ∈ Xi
4: end for
5: while |q.S| < k do
6:    for all xi ∈ X do
7:       Q ← all queries sharing xi
8:       S' ← q1.S ∩ q2.S ... ∩ q|Q|.S
9:       dx ← d(xi,S')
10:      for all qi ∈ Q do
11:         ds ← dx + d(xi, q.S\S')
12:         if (ds > d(q.candidate, q.S) then
13:            q.candidate ← xi
14:         end if
15:      end for
16:   end for
17:   for all Queries do
18:      qi.S ← qi.S ∪ {qi.candidate}
19:   end for
20: end while
21: return q1.S, q2.S, ..., qn.S
```

Similarly, for a single item $x_j \in X_2$, the distance function $d(x_j, S_2)$ can be computed as:

$$d(x_j, S_2) = d(x_j, S_2 \backslash S_1) + d(x_j, S_1 \cap S_2)$$

Hence we can compute $d(x_j, S_1 \cap S_2)$ only once for every item $x_j \in X_1 \cap X_2$ when computing $d(x_j, S_1)$ and $d(x_j, S_2)$.

Clearly, the calculation of $d(x_j, S_1)$ outlined above is an example of applying partial aggregation, in which the final value of the distance is easily assembled from its partial values. This is applicable over all distributive and algebraic distance functions that are typically used in measuring (dis)similarity, such as all variants of L_p norm including the Euclidean distance. The combination of the first and second observation listed above allows DoS-Overlap to exploit the data overlap exhibited by the queries in Example 2 along two orthogonal dimensions:

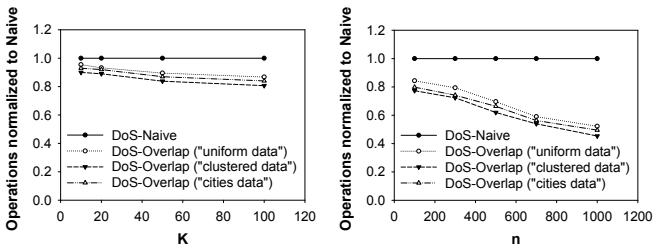
1. **Overlap in Result Sets:** Process the sets $X_1 \backslash X_2$, $X_2 \backslash X_1$ and $X_1 \cap X_2$ separately at each iteration of the algorithm, and

2. **Overlap in Diverse Sets:** Process the set $S_1 \cap S_2$ only once at each iteration of the algorithm.

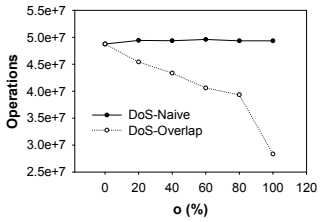
Hence, the processing cost $C(S_1, S_2)$ incurred by DoS-Overlap in processing the two result sets X_1 and X_2 in Example 2 can be expressed as:

$$C(S_1, S_2) = C_{\text{DoS-Naïve}}(S_1, S_2) - O(k|S_1 \cap S_2||X_1 \cap X_2|)$$

Compared to DoS-Naïve, DoS-Overlap produces exactly the same set of diverse sets ($\mathbb{S}$). During each iteration, however, DoS-Overlap clearly requires less operations for the computation and comparison of distance functions. A fundamental component in the design of DoS-Overlap is the efficient detection of overlapping results, i.e., $|X_1 \cap X_2|$. To achieve this



(a) Impact of varying k. (b) Impact of varying number of queries.



(c) Impact of varying overlap.

Figure 2: DoS-Overlap vs. DoS-Naïve

goal, we leverage hashing schemes. In particular, all results are hashed into their corresponding buckets using a multi-dimensional hash function. Hence, instead of comparing all pairs of results, only the results within the same bucket are compared to find duplicates. Clearly the number of comparisons required depends on many factors including hash function and bucket size. However, the reductions provided by DoS-Overlap during locating the diverse sets clearly outweighs that overhead as shown in the next section.

4. EXPERIMENTAL EVALUATION

We perform a number of experiments to evaluate the efficiency of our DoS scheme. We compared the performance of our DoS-Overlap algorithm, presented in Section 3.2, against the baseline approach of using the DoS-Naïve algorithm for multiple search result diversification. The performance of both algorithms is measured based on cost $C(\mathbb{S})$, that is measured as the sum of operations performed to locate set $\mathbb{S}$ of n subsets $\{S_1, S_2, \ldots, S_n\}$, where each operation represents one distance computation and data comparison task.

We use normalized synthetic and real data sets. Our synthetic data sets consist of points in the 2-dimensional Euclidean space. Points are either uniformly distributed ("Uniform") or form clusters around a random number of points ("Clustered"). Our real data set "Cities" contains points representing the locations of 5922 towns in a 2-dimensional space (previously used in [3]). We generate a random set of range queries. For each experiment, the number of queries n is in the range [20–1000]. Each query is associated with the size of diverse set k, which takes values in the range [10–100]. Further, we introduce a simulation parameter o to tune the amount of overlap between search results of any two range queries, where o varies between 0% to 100%.

Impact of Diverse Set Size: Next, we report on the impact of the required number of diverse results k. Figure 2(a) shows the average number of operations performed by DoS-Naïve and DoS-Overlap algorithm for different values of k, over different data sets, where n is equal to 20. The y-axis is normalized to the corresponding values of DoS-Naïve. We see that, DoS-Overlap is performing up to 14% less operations when compared to DoS-Naïve. The cost savings are larger for larger values of k. The reason for this is that, for larger values of k, even a small percentage of overlap between result sets will provide more shared diverse results.

Impact of Number of Queries: Figure 2(b) compares the performance of DoS-Naïve and DoS-Overlap when varying the number of concurrent queries n, over different data sets. Here value of k is 100. The y-axis is again normalized to the corresponding values of DoS-Naïve. Figure 2(b) shows that for all the data sets, cost savings of DoS-Overlap increase with n, since, in the presence of more queries, more overlapping regions can be exploited by our DoS-Overlap. Number of operations performed by DoS-Overlap are almost 50% less as n approaches 1000.

Impact of Query Overlap: To study the impact of the overlap between queries, we first generate one query Q_1 for our "Uniform" dataset. Then, we generate an identical second query Q_2 and we "slide" Q_2 over Q_1 to control the overlapping area between the result sets of the queries. As we see in Figure 2(c), the number of operations performed by DoS-Naïve remain almost constant, however, the cost of DoS-Overlap decreases as the amount of overlap increases.

5. CONCLUSIONS

In this paper, we focused on the NP-hard problem of diversifying the results of multiple queries and proposed DoS, an efficient scheme for the diversification of multiple search results. DoS exploits the natural overlap among the result sets of the various queries in conjunction with the concurrent diversification of those overlapping results. DoS provides solutions of quality equal to that of sequential methods, while significantly reducing processing costs.

Acknowledgment. We would like to thank the anonymous reviewers as well as Abdullah Albarrak for their valuable comments and suggestions. This work is partially supported by Australian Research Council grant DP110102777.

6. REFERENCES

[1] A. Angel and N. Koudas. Efficient diversity-aware search. In *SIGMOD Conference*, 2011.
[2] U. Çetintemel et al. Query steering for interactive data exploration. In *CIDR*, 2013.
[3] M. Drosou and E. Pitoura. Search result diversification. *SIGMOD Record*, 39(1):41–47, 2010.
[4] E. Erkut et al. A comparison of p-dispersion heuristics. *Computers & OR*, 21(10):1103–1113, 1994.
[5] S. Guirguis et al. Optimized processing of multiple aggregate continuous queries. In *CIKM*, 2011.
[6] A. Gupta et al. Aggregate-query processing in data warehousing environments. In *VLDB*, 1995.
[7] W. O'Mullane et al. Batch is back: Casjobs, serving multi-tb data on the web. In *ICWS*, 2005.
[8] S. S. Ravi et al. Facility dispersion problems: Heuristics and special cases. In *WADS*, 1991.
[9] M. R. Vieira et al. On query result diversification. In *ICDE*, 2011.