On Query Result Diversification

Marcos R. Vieira[1], Humberto L. Razente[2], Maria C. N. Barioni[2],
Marios Hadjieleftheriou[3], Divesh Srivastava[3], Caetano Traina Jr.[4], Vassilis J. Tsotras[1]

[1]*University of California, Riverside, CA, USA*
{mvieira,tsotras}@cs.ucr.edu

[2]*Federal University of ABC, Santo André, SP, Brazil*
{humberto.razente,camila.barioni}@ufabc.edu.br

[3]*AT&T Labs – Research, Florham Park, NJ, USA*
{marioh,divesh}@research.att.com

[4]*University of São Paulo, São Carlos, SP, Brazil*
caetano@icmc.usp.br

***Abstract*— In this paper we describe a general framework for evaluation and optimization of methods for diversifying query results. In these methods, an initial ranking candidate set produced by a query is used to construct a result set, where elements are ranked with respect to *relevance* and *diversity* features, i.e., the retrieved elements should be as relevant as possible to the query, and, at the same time, the result set should be as diverse as possible. While addressing relevance is relatively simple and has been heavily studied, diversity is a harder problem to solve. One major contribution of this paper is that, using the above framework, we adapt, implement and evaluate several existing methods for diversifying query results. We also propose two new approaches, namely the Greedy with Marginal Contribution (GMC) and the Greedy Randomized with Neighborhood Expansion (GNE) methods. Another major contribution of this paper is that we present the first thorough experimental evaluation of the various diversification techniques implemented in a common framework. We examine the methods' performance with respect to precision, running time and quality of the result. Our experimental results show that while the proposed methods have higher running times, they achieve precision very close to the optimal, while also providing the best result quality. While GMC is deterministic, the randomized approach (GNE) can achieve better result quality if the user is willing to tradeoff running time.**

I. Introduction

Many database and information retrieval applications have recently started to incorporate capabilities to rank elements with respect to *relevance* and *diversity* features, i.e., the retrieved elements should be as relevant as possible to the query, and, at the same time, the result set should be as diverse as possible. Examples of such applications range from exploratory and ambiguous keywords searches (e.g., jaguar, java, windows, eclipse) [1], [2], [3], diversification of structured databases [4], [5] and user personalized results [6], to topic summarization [7], [8], [9], or even to exclude near-duplicate results from multiple resources [10]. While addressing relevance is comparatively straightforward, and has been heavily studied in both database and information retrieval areas, diversity is a more difficult problem to solve [11], [12].

Typically, in all of the above applications, the final result set is computed in two phases. First, a ranking *candidate set* S, with elements that are relevant to the user's query, is retrieved. Then, in the second phase, a *result set* R, $R \subseteq S$, is computed containing very *relevant* elements to the query and, at the same time, as *diverse* as possible to other ones in the result set R. Since these two components, relevance and diversity, compete with each other, algorithms for query result diversification attempt to find a *tradeoff* between the *relevance* and *diversity* components. Thus, the query result diversification problem can be modeled as a bi-criteria optimization problem. One advantage of using the *tradeoff* parameter to tune the importance between *relevance* and *diversity* is that the user can give more preference on one of these two components. For instance, if a candidate set has a large amount of near-duplicate elements, then a user can increase the *tradeoff* value, as necessary, in order to have more diverse elements in the result set.

Several techniques have been introduced for diversifying query results, with the majority of them exploring a greedy solution that builds the result set in an incremental way [9], [13], [14], [15], [16], [17]. These techniques typically choose the first element to be added to the result set based only on relevance; further elements are added based on an element's relevance and diversity against the current result set. The basic assumption of these techniques is that the result set does not change with the size k of the result set, i.e., $R \subset R'$, $|R| = k$ and $|R'| = k + 1$, which is not always true. In some of the above techniques, there is the additional problem that they are not able to handle different values of *tradeoff* between relevance and diversity; and for the few ones that do support it, e.g., [9], [2], they do not perform well, as we show in our experiments.

In this paper we describe a framework based on a bi-criteria optimization objective function, similar to [2], to compare methods of diversifying query results. Gollapudi and Sharma [2] proposed a set of natural axioms that a diversification system is expected to satisfy and show that no diversification objective can satisfy all the axioms simultaneously. Since there is no single objective function that is suitable for every application domain, the authors describe three diversification objective functions: *max-sum diversification*, *max-min diversification* and *mono-objective*. In our work we particularly focus on the *max-sum diversification* since it can be reduced to dif-

ferent versions of the well-studied facility dispersion problem [18], [19], for which efficient approximation algorithms exist (e.g., [20], [21], [22], [23]). Nevertheless, our work can be easily extended to other functions, e.g. max-min, max-avg.

Since the framework and the methods described in this paper rely only on the relevance and diversity values, our work can be employed in *any* domain where relevance and diversity functions can be defined. Moreover, the evaluation of the different methods described here is done using only the values computed by the optimization objective function, and not using any external information (e.g., subtopic coverage).

In addition, we also describe two new methods that incrementally construct the result set using a scoring function. This function ranks candidate elements using not only relevance and diversity to the existing result set, but also accounts for diversity against the remaining candidates. We present a thorough experimental evaluation of several of the diversification techniques described in this paper. Our experimental results show that the two proposed methods achieve precision very close to the optimal, while also providing the best result quality measured using the optimization objective function.

In this paper we make the following contributions:

1) we present a framework for evaluation and optimization of methods for diversifying query results, where a user can adjust the *tradeoff* between relevance and diversity;
2) we adapt and evaluate several existing methods using the above framework;
3) we propose a new function to compute the contribution of each candidate element to the result set. This function not only considers the relevance value between the candidate element and the query, but also how diverse it is to other elements in the candidate and result sets;
4) we propose two new methods for diversifying query results using the above contribution function;
5) we perform a thorough experimental evaluation of the various existing diversification techniques.

The rest of the paper is organized as follows: in Section II we review the related work; in Section III we define the query result diversification problem; in Sections IV and V we describe, respectively, previously known and two new methods using the proposed framework; the experimental evaluation is shown in Section VI; and Section VII concludes the paper.

II. Related Work

We first present related work on query result diversification and then on the *Max-Sum Dispersion Problem*, which relates to our GNE method.

A. Query Result Diversification

Result diversification has recently been examined and applied to several different domains [24]. In [25], clustering techniques are employed in order to generate diverse results, while in [26], learning algorithms based on users' clicking behavior are used to *re-rank* elements in the answer set. Techniques to extract compact information from retrieved documents in order to test element coverage in the answer set with respect to the query (and at the same time, avoid excessive redundancy among elements in the result) are explored in [7], [8]. Similar techniques are employed for structured query results in [4], [5]. Topic diversification methods based on personalized lists in recommendation systems are proposed in [27]. In [6], [28], [29] techniques to generate related queries to the user's query are employed to yield a more diverse result set for documents. A risk minimization framework where users can define a loss function over a result set that leads to a diverse result is proposed in [30].

Greedy algorithms for explanation-based diversification for recommended items are proposed in [14]. Unlike ours, this approach considers the relevance and diversity of explanation items as two separate components. Therefore, to optimize both relevant and diverse explanations, the proposed algorithms rely on parameter values that typically are not easy to tune.

Agrawal et al. [11] describe a greedy algorithm which relies on an objective function, computed based on a probabilistic model, that admits a sub-modularity structure and a taxonomy to compute diverse results. Relevance of documents to queries is performed using standard ranking techniques while diversity is performed through categorization according to the taxonomy. The aim is to find a set of documents that "covers" several taxonomies of the query, that is, the result set is diverse for the defined query using the taxonomy. [17] presents two greedy algorithms to compute a result set; here the elements in the result set are diverse regarding their frequency in the data collection. However, the proposed algorithms work only on structural data with explicitly defined relationships. In [1], a Bayesian network model is used as a blind negative relevance feedback to *re-rank* documents, assuming that in each position in the rank order all previous documents are irrelevant to the query. An affinity graph is used in [31] to penalize redundancy by lowering an item in the rank order if already seen elements appeared in the rank order. In [32], absorbing Markov chain random walks is used to *re-rank* elements. In this work, an item that has already been ranked becomes an absorbing state, dragging down the importance of similar unranked states.

A main difference between all these works and the ones described in this paper is that we do not rely on external information, like taxonomy, known structure of the data, click-through rates, query logs, etc., to generate diverse results. Since workloads and queries are rarely known in advance, the external information discussed above is typically expensive to compute and provides suboptimal results. Instead, the methods presented here rely only on computed values using similarity (relevance) and diversity functions (e.g., tf/idf cosine similarity, Euclidean distance) in the data domain.

The Maximal Marginal Relevance (MMR) [9] is one of the earliest proposals to use a function that *re-ranks* elements in the answer set in order to generate relevant elements to the query, and, at the same time, diverse to the previously selected elements in the result. Variations of MMR were also proposed for different domains [13], [33], [34]. At each iteration, the MMR function returns the element with the highest value with respect to a *tradeoff* between relevance and diversity

to the current result set (only). Our proposed methods also use a function that computes a *tradeoff* between relevance and diversity. However, GMC and GNE use a function that computes the maximum contribution that an element can provide to the **final result**. Moreover, MMR always picks as first, the element that is most relevant to the query. This can highly affect the other elements to be picked in the result. The MMR is one of several methods that we discuss and compare against in our framework.

[16] proposed a greedy strategy to compute the answer set based on a *Boolean* function: if an element has diversity value more than a predefined threshold to all elements in the current result set, then the element is included in the result set. Therefore, the diversity function is used only to filter out answer set elements. The aim is to maximize the relevance while the result set's diversity remains above a certain threshold. This is different from our methods which maximize the objective function (i.e., relevance and diversity). Furthermore, the threshold parameter has a great influence on the performance of the algorithm, and can easily affect the quality of the results, as the experimental comparison also verifies.

B. Max-Sum Dispersion Problem

A general framework for result diversification appears in [2] with eight axioms that a diversification system is expected to satisfy. The problem formulation that most relates to our paper is the max-sum diversification. [2] also shows how to reduce max-sum diversification to a max-sum dispersion problem (also known as the p-dispersion in the Operations Research literature [18], [19]). Given that the objective function is a metric, it then uses an approximation algorithm [20] to solve the problem.

Instead of an approximation approach, our second proposed method (GNE) uses a randomized algorithm to solve the max-sum diversification. In particular, we use an adaptation of a meta-heuristic called Greedy Randomized Adaptive Search Procedure (GRASP) proposed in [22], [23]. To the best of our knowledge, our proposal is the first randomized approach applied to the result diversification problem. As it will be shown in the experimental section, the approximation algorithm in [2] is very expensive to compute and provides results with limited precision when compared to the optimal solution.

GRASP is a greedy randomized method that employs a multistart procedure for finding approximate solutions to combinatorial optimization problems. It is composed of two phases: *construction* and *local search*. In the *construction* phase a solution is iteratively constructed through controlled randomization. Then, in the *local search* phase, a local search is applied to the initial solution in order to find a locally optimal solution. The best overall solution is kept as the result. These two phases are repeated until a stopping criterion is reached. This same procedure is performed in the GNE method, but with different strategies in the *construction* and *local search* phases, as explained in Section V.

III. Preliminaries

The *Result Diversification Problem* can be stated as a *tradeoff* between finding relevant (similar[1]) elements to the query, and diverse elements in the result set. Let $S = \{s_1, ..., s_n\}$ be a set of n elements, q a query element and k, $k \leq n$, an integer. Let also the relevance (similarity) of each element $s_i \in S$ be specified by the function $\delta_{sim}(q, s_i)$, $\delta_{sim} : q \times S \rightarrow \mathbb{R}^+$, where δ_{sim} is a monotonically increasing function (i.e., a higher value implies that the element s_i is more relevant to query q), and the diversity between two elements $s_i, s_j \in S$ be specified by the function $\delta_{div}(s_i, s_j) : S \times S \rightarrow \mathbb{R}^+$. Informally, the problem of result diversification can be described as follows: given a set S and a query q, we want to find $R \subseteq S$ of size $|R| = k$ where each element in R is relevant to q with respect to δ_{sim} and, at the same time, diverse among other elements in R with respect to δ_{div}.

In our model, we represent elements in S using the vector space model. For example, for δ_{sim} and δ_{div} functions the weighted similarity function tf/idf cosine similarity or the Euclidean distance can be employed to measure the similarity or diversity, respectively.

The search for R can be modeled as an optimization problem where there is a *tradeoff* between finding relevant elements to q and finding a diverse result set R. This optimization problem can be divided into two components, one related to similarity, as defined below, and one related to diversity.

Definition 1: the **k-similar set** R contains k elements of S that:

$$R = \underset{S' \subseteq S, k=|S'|}{\operatorname{argmax}} \; sim(q, S')$$

$$\text{where } sim(q, S') = \sum_{i=1}^{k} \delta_{sim}(q, s_i), s_i \in S'$$

in other words, Definition 1 finds a subset $R \subseteq S$ of size k with the largest sum of similarity distances among all sets of size k in S. Intuitively, $sim(q, S')$ measures the amount of "attractive forces" between q and k elements in S'. Basically, any algorithm that can rank elements in S with respect to δ_{sim}, and then extract the top-k elements in the ranked list can evaluate the *k-similar set* problem.

The diversity component is defined as follows:

Definition 2: the **k-diverse set** R contains k elements of S that:

$$R = \underset{S' \subseteq S, k=|S'|}{\operatorname{argmax}} \; div(S')$$

$$\text{where } div(S') = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \delta_{div}(s_i, s_j), s_i, s_j \in S'$$

hence, the problem in the above definition selects a subset $R \subseteq S$ of size k that maximizes the sum of inter-element distances amongst elements of R chosen from S. Intuitively, $div(S')$ measures the amount of "repulsive forces" among k elements in S'. The above definition is equivalent to the *Max-Sum Dispersion Problem* [21] (i.e., maximize the sum of distances between pairs of facilities) encountered in operations

[1]These two terms are used interchangeably throughout the text.

research literature. For the special case where the distance function δ_{div} is metric, efficient approximate algorithms exist to compute the **k-diverse set** [20].

In both Definitions 1 and 2, the objective function is defined as a *max-sum problem*, but other measures could be used as well, e.g., max-min, max-avg, min-max. The proper measure is very much a matter of the application in hand. Here we develop algorithms specifically for the *max-sum problem* since it seems it has been the most widely accepted among previous approaches (e.g., [2], [9], [13], [14], [16], [33]).

Next, we give the formal definition of our problem: computing a set $R \subseteq S$ of size k with a *tradeoff* between finding k elements that are similar to the query q, defined by Definition 1, and finding k elements that are diverse to each other, defined by Definition 2. Formally, the problem is defined as follows:

Definition 3: given a *tradeoff* λ, $0 \leq \lambda \leq 1$, between similarity and diversity, the **k-similar diversification set** R contains k elements in S that:

$$R = \underset{S' \subseteq S, k=|S'|}{\operatorname{argmax}} \mathcal{F}(q, S')$$

where $\mathcal{F}(q, S') = (k-1)(1-\lambda) \cdot sim(q, S') + 2\lambda \cdot div(S')$

The weight variable λ represents the *tradeoff* between finding similar and diverse elements. Intuitively, $\mathcal{F}(q, S')$ measures the amount of "attractive forces", between q and k elements in S', and "repulsive forces", among elements in S'. Using the same analogy, the above definition selects the "most stable system" R with k elements of S. Note that, since both components sim and div have different number of elements, k and $\frac{k(k-1)}{2}$, respectively, in the above definition the two components are scaled up. The variable λ is a *tradeoff* specified by the user, and it gives the flexibility to return more relevant, or diverse, results for a given query q.

In Definition 3 there are two special cases for the values of λ. The *k-similar diversification set* problem is reduced to *k-similar set* when $\lambda = 0$, and the result set R depends only on the query q. The second case is when $\lambda = 1$, and the *k-similar diversification set* problem is reduced to finding the *k-diverse set*. In this case, the query q does not play any role in the result set R. Note that our formula is slightly different than the *Max-Sum Diversification Problem* of [2] since it allows us to access both extremes in the search space (diversity only, when $\lambda = 1$, or relevance only, when $\lambda = 0$).

Only when $\lambda = 0$ the result R is straightforward to compute. For any other case, $\lambda > 0$, the associated decision problem is NP-hard, and it can be easily seen by a reduction from the maximum clique problem known to be NP-complete [35]. A brute force algorithm to evaluate the *k-similar diversification set* when $\lambda > 0$ is presented in Algorithm 1. This algorithm tests for every possible subset $R \subseteq S$ of size k to find the highest $\mathcal{F}$ value. Algorithm 1 takes time $O(|S|^k k^2)$, where there are $O(|S|^k)$ possible results S of size k to check, each of which has $O(k^2)$ distances that need to be computed.

Algorithm 1 *k-similar diversification query*

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$, with the **highest** possible $\mathcal{F}$ value
1: let R be an arbitrary set of size k
2: **for** each set $R' \subseteq S$ of k size **do**
3: **if** $\mathcal{F}(q, R') > \mathcal{F}(q, R)$ **then**
4: $R \leftarrow R'$

IV. Known Methods

In this section we summarize all known methods for diversifying query results in the literature, for which no extra information other than the relevance and diversity values are used. All methods are casted in terms of the framework presented in the previous section.

A. Swap Method

Perhaps the simplest method to construct the result set R is the *Swap* method [14] (Algorithm 2), which is composed of two phases. In the first phase, an initial result R is constructed using the top-k relevant elements in S. In the second phase, each remaining element in S, ordered by decreasing δ_{sim} values, is tested to replace an element from the current solution R. If there exists such operation that improves $\mathcal{F}$, then the replace operation that improves $\mathcal{F}$ the most is applied permanently in R. This process continues until every element in the candidate set S is tested.

The $\mathcal{F}$ value of the final result R found by the *Swap* method is not guaranteed to be optimal, since elements in the candidate set S are analyzed with respect to their δ_{sim} order. That is, this method does not consider the order of δ_{div} values in S, which can result in solutions that do not maximize $\mathcal{F}$.

Algorithm 2 *Swap*

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$
1: $R \leftarrow \emptyset$
2: **while** $|R| < k$ **do**
3: $s_s \leftarrow \operatorname{argmax}_{s_i \in S}(\delta_{sim}(q, s_i))$
4: $S \leftarrow S \setminus s_s$
5: $R \leftarrow R \cup s_s$
6: **while** $|S| > 0$ **do**
7: $s_s \leftarrow \operatorname{argmax}_{s_i \in S}(\delta_{sim}(q, s_i))$
8: $S \leftarrow S \setminus s_s$
9: $R' \leftarrow R$
10: **for** each $s_j \in R$ **do**
11: **if** $\mathcal{F}(q, \{R \setminus s_j\} \cup s_s) > \mathcal{F}(q, R')$ **then**
12: $R' \leftarrow \{R \setminus s_j\} \cup s_s$
13: **if** $\mathcal{F}(q, R') > \mathcal{F}(q, R)$ **then**
14: $R \leftarrow R'$

B. BSwap Method

The *BSwap* method [14] (Algorithm 3) uses the same basic idea of the *Swap* method for exchanging elements between the candidate set S and the current result R. However, instead of improving $\mathcal{F}$, *BSwap* checks for an improvement in the diversity value div of the current result R, without a sudden drop in δ_{sim} between the exchanged elements.

In each iteration of the *BSwap*, the element in S with the highest δ_{sim} value is exchanged to the one in R which contributes the least to δ_{div}. If this operation improves div but without dropping δ_{sim} by a predefined threshold θ, then the result set R is updated (the two elements are exchanged). This process continues until every element in the candidate set S is tested, or the highest δ_{sim} value in S is below the threshold

θ. This later condition is enforced to avoid a sudden drop in δ_{sim} in R. In summary, this method exchanges elements to increase diversity in detriment of a drop of relevance.

While very simple, a drawback of *BSwap* is the use of θ. Setting this threshold with an appropriate value is a difficult task, and can vary for different datasets and/or queries. If not set appropriately, the *BSwap* may find results which have less than k elements (if θ is set to a very small value), or with a very low quality in terms of $\mathcal{F}$ (if θ is set to a very large value). As shown in the experimental section, this method also suffers in terms of precision of the result set.

Algorithm 3 *BSwap Algorithm*

Input: candidate set S, result set size k, and distance threshold θ
Output: result set $R \subseteq S$, $|R| = k$

$R \leftarrow \emptyset$
while $|R| < k$ **do**
  $s_s \leftarrow \text{argmax}_{s_i \in S}(\delta_{sim}(q, s_i))$
  $S \leftarrow S \setminus s_s$
  $R \leftarrow R \cup s_s$
$s_d \leftarrow \text{argmin}_{s_i \in R}(div(R \setminus s_i))$
$s_s \leftarrow \text{argmax}_{s_i \in S}(\delta_{sim}(q, s_i))$
$S \leftarrow S \setminus s_s$
while $\delta_{sim}(q, s_d) - \delta_{sim}(q, s_s) \leq \theta$ **and** $|S| > 0$ **do**
  if $div(\{R \setminus s_d\} \cup s_s) > div(R)$ **then**
    $R \leftarrow \{R \setminus s_d\} \cup s_s$
    $s_d \leftarrow \text{argmin}_{s_i \in R}(div(R \setminus s_i))$
  $s_s \leftarrow \text{argmax}_{s_i \in S}(\delta_{sim}(q, s_i))$
  $S \leftarrow S \setminus s_s$

C. Maximal Marginal Relevance

The Maximal Marginal Relevance (MMR) [9] (Algorithm 4), iteratively constructs the result set R by selecting a new element in S that maximizes the following function:

$$mmr(s_i) = (1 - \lambda)\delta_{sim}(s_i, q) + \frac{\lambda}{|R|} \sum_{s_j \in R} \delta_{div}(s_i, s_j) \quad (1)$$

The MMR method has two important properties that highly influence the chosen elements in the result set R. First, since R is empty in the initial iteration, the element with the highest δ_{sim} in S is always included in R, regardless of the λ value. Second, since the result is incrementally constructed by inserting a new element to previous results, the first chosen element has a large influence in the quality of the final result set R. Clearly, if the first element is not chosen properly, then the final result set may have low quality in terms of $\mathcal{F}$. We show in the experimental section that the quality of the results for the MMR method decreases very fast when increasing the λ parameter.

D. Motley

Similarly to MMR, the *Motley* approach [16] (Algorithm 5)[2] also iteratively constructs the result set R. It considers the elements in S, ranked by their δ_{sim} values, until R contains k elements. In the first iteration, the element with the highest δ_{sim} value is inserted in R. Then, an element from the candidate set S is chosen to be included in R if it has δ_{div} value greater than a predefined threshold θ' to all the already inserted elements in R. If such condition is not satisfied, then the element is discarded from S and the next top ranked element in S, with respect to δ_{sim}, is evaluated. This process repeats until the result set R has k elements or there are no more elements in S to be analyzed.

[2]This is a simplified version of the Buffered Greedy Approach [16]. In our preliminary experiments, the results of both approaches were very similar, except that the one described here is several orders of magnitude faster than the Buffered Greedy.

Algorithm 4 MMR

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$

$R \leftarrow \emptyset$
$s_s \leftarrow \text{argmax}_{s_i \in S}(mmr(s_i))$
$S \leftarrow S \setminus s_s$
$R \leftarrow s_s$
while $|R| < k$ **do**
  $s_s \leftarrow \text{argmax}_{s_i \in S}(mmr(s_i))$
  $S \leftarrow S \setminus s_s$
  $R \leftarrow R \cup s_s$

Like the MMR method, the *Motley* approach also is affected by the initial element choice. Similarly to the *BSwap*, *Motley* uses a threshold parameter to check the amount of diversity in the result set.

Algorithm 5 *Motley*

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$

$s_s \leftarrow \text{argmax}_{s_i \in S}(\delta_{sim}(q, s_i))$
$S \leftarrow S \setminus s_s$
$R \leftarrow s_s$
while $|R| < k$ **do**
  $s_s \leftarrow \text{argmax}_{s_i \in S}(\delta_{sim}(q, s_i))$
  $S \leftarrow S \setminus s_s$
  if $\delta_{div}(s_r, s_s) \geq \theta', \forall s_r \in R$ **then**
    $R \leftarrow R \cup s_s$

E. Max-Sum Dispersion

The Max-Sum Dispersion (MSD) method [2] (Algorithm 6) is based on the 2-approximation algorithm proposed in [20] for the *Max-Sum Dispersion Problem*. The MSD method employs a greedy approach to incrementally construct the result set R by selecting a pair of elements that are relevant to the query and diverse to each other. More formally, in each iteration of the MSD method, the pair of elements $s_i, s_j \in S$ that maximize the following function is chosen to be part of the result set R:

$$msd(s_i, s_j) = (1 - \lambda)(\delta_{sim}(q, s_i) + \delta_{sim}(q, s_j)) + 2\lambda\delta_{div}(s_i, s_j) \quad (2)$$

The above description applies when the value for k is even, since a pair of elements is selected in each iteration of the MSD method. When k is odd, in the final phase of the MSD method an arbitrary element in S is chosen to be inserted in the result set R.

F. Clustering-Based Method

The Clustering-Based Method (CLT) is based on clustering [25] (Algorithm 7). In its first phase the k-medoid algorithm

Algorithm 6 MSD

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$
1: $R \leftarrow \emptyset$
2: **while** $|R| < \lfloor k/2 \rfloor$ **do**
3: $\{s_i, s_j\} \leftarrow \text{argmax}_{s_i, s_j \in S}(msd(s_i, s_j))$
4: $R \leftarrow R \cup \{s_i, s_j\}$
5: $S \leftarrow S \setminus \{s_i, s_j\}$
6: **if** k is odd **then**
7: choose an arbitrary object $s_i \in S$
8: $R \leftarrow R \cup s_i$

with δ_{div} as distance function is employed in S to generate k clusters $C = \{c_1, c_2, ..., c_k\}$. Then, one element from each C cluster is selected to be part of the result set R. Several strategies on selecting an element from each cluster can be employed. For example, one can choose the element with the highest δ_{sim} value, or the medoid element of each cluster C.

Since it is not possible to incorporate the *tradeoff* λ in the clustering step of the CLT method, the result set R only depends on the strategy employed for selecting an element in each cluster. Thus, for low values of λ, the result set R will be of better quality if the element with the highest δ_{sim} is selected. For higher values of λ, selecting the medoid element of each cluster is more suitable since the medoids typically have high δ_{div} values to each other. In our experiments we only consider the medoid-based strategy since it typically gives more diverse results.

Algorithm 7 CLT

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$
1: $R \leftarrow \emptyset$
2: let $C = \{c_1, c_2, ..., c_k\}$ be the result of the k-medoid algorithm, using δ_{div}
3: **for** each cluster $c_i \in C$ **do**
4: let s_i be a selected element in c_i
5: $R \leftarrow R \cup s_i$

V. Proposed Methods

We proceed with the presentation of two new approaches for diversifying query results, the *Greedy Marginal Contribution* (GMC) and the *Greedy Randomized with Neighborhood Expansion* (GNE) methods. In both methods we employ the same function to rank elements regarding their marginal contribution to the solution. An important difference between the two methods is that, in the GMC, the element with the highest partial contribution is always chosen to be part of the solution, while in GNE, an element is randomly chosen from among the top ranked ones to be included in the solution.

A. GMC Method

The GMC method (Algorithm 8) incrementally builds the result R by selecting the element with the highest maximum marginal contribution (mmc) to the solution constructed so far. In each iteration, the GMC method ranks the elements in the candidate set S using the following function:

$$mmc(s_i) = (1-\lambda)\delta_{sim}(s_i, q) + \frac{\lambda}{k-1} \sum_{s_j \in R_{p-1}} \delta_{div}(s_i, s_j) + \frac{\lambda}{k-1} \sum_{\substack{l=1 \\ s_j \in S - s_i}}^{l \leq k-p} \delta^l_{div}(s_i, s_j) \quad (3)$$

where R_{p-1} is the partial result of size $p-1$, $1 \leq p \leq k$, and $\delta^l_{div}(s_i, s_j)$ gives the l^{th} largest δ_{div} value in $\{\delta^l_{div}(s_i, s_j) : s_j \in S - R_{p-1} - s_i\}$. Using the above function we can compute the maximum contribution of s_i to $\mathcal{F}$ using the elements already inserted in the partial result R_{p-1} (first two components) and the remaining $k-p$ elements in S that could be inserted in R (third component). For the third component, the value is upper bounded using the highest $k-p$ values for δ_{div} which is formed using the remaining elements in S that can be inserted in R. Thus, we employ mmc to rank every element in S using the constructed result R_{p-1} and the remaining elements to be included in the final result R.

Algorithm 8 GMC

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$
1: $R_0 \leftarrow \emptyset$
2: **for** $p \leftarrow 1$ to $p = k$ **do**
3: $s_i \leftarrow \text{argmax}_{s_i \in S}(mmc(s_i))$
4: $R_p \leftarrow R_{p-1} \cup s_i$
5: $S \leftarrow S \setminus s_i$
6: $R \leftarrow R_p$

The intuition behind mmc is the following: When the current result set R_0 is empty, then elements in S are ranked based on their δ_{sim} and their relationships with other elements in S, defined by δ^l_{div}. In this way, mmc gives higher preference to elements that are very relevant to q, and, at the same time, are very diverse to other elements in the candidate set S, regardless of R_0; Whenever elements are inserted in R_p, then mmc also considers the diversity w.r.t. elements in R_{p-1}.

In the first step of the method, since the result set R_0 is empty, the sum of δ_{div} in mmc is zero. Thus, only the δ_{sim} and δ^l_{div} components are considered. In the following iterations mmc considers the elements in R_{p-1}, and the value for the δ^l_{div} component is updated. As more elements are inserted in the result set, the second and third components in the mmc function increase and decrease, respectively. In the last iteration ($p = k$), the third component δ^l_{div} is null since there is only one element to be inserted in R.

Compared to MMR (Section IV-C), the GMC method uses a different function to rank the elements in the candidate set S. Since R is empty in the first step of the MMR, the diversity component does not play any role in selecting the first element in R. In GMC, the selection of the first result element is based on the maximum marginal contribution that it can make to $\mathcal{F}$. Note that mmc combines the relevance of a new element, its diversity to the already inserted elements in the result, and its maximum contribution to $\mathcal{F}$ considering the remaining elements in S.

The δ^l_{div} values for each element $s_i \in S$ are computed only in the first iteration of the GMC method. For each entry $s_i \in S$, a set of pairs $< s_j, \delta_{div}(s_i, s_j) >$ is kept, where s_j is the element that belongs to one of the δ^l_{div}, and its corresponding value $\delta_{div}(s_i, s_j)$. This set starts with $k-1$ entries and it decreases by 1 in each iteration. The set of pairs for each element in S is updated in each iteration of the method. Two cases can occur. The first is when there is an element $s_i \in S$ that has the pair $< s'_i, \delta_{div}(s_i, s'_i) >$ in its set of pairs, where element s'_i is the element previously inserted in the result set R_p. Then the pair $< s'_i, \delta_{div}(s_i, s'_i) >$ is removed from the set of pairs of s_i. The second case occurs when there is an element $s_i \in S$ that does not have the pair $< s'_i, \delta_{div}(s_i, s'_i) >$ in its set of pairs. Now the pair with the lowest $\delta_{div}(s_i, s_j)$ is dropped from the set.

B. GNE Method

Our second proposed method uses the GRASP (Greedy Randomized Adaptive Search Procedure) [22] technique for diversifying query results. To the best of our knowledge, this is the first randomized solution proposed for the diversification problem. Different from the GMC that always selects the top element in the rank, in each iteration of the GNE algorithm a random element, among the top ranked ones, is chosen. Algorithm 9 illustrates the general GRASP algorithm with i_{max} iterations. The two phases of GNE are described below.

Algorithm 9 *GRASP*

Input: candidate set S and result set size k
Output: result set $R \subseteq S$, $|R| = k$
1: $R \leftarrow \emptyset$
2: **for** $i \leftarrow 0$ to $i < i_{max}$, $i \leftarrow i+1$ **do**
3: $R' \leftarrow$ GNE-Construction()
4: $R' \leftarrow$ GNE-LocalSearch(R')
5: **if** $\mathcal{F}(q, R') > \mathcal{F}(q, R)$ **then**
6: $R \leftarrow R'$

1) GNE Construction Phase: At each iteration the choice of the next element to be added in R is determined by a greedy randomized ranking function, which ranks the elements in S according to mmc (Equation 3). Only the elements with the highest individual contributions, defined by mmc, are considered to be stored in the *Restricted Candidate List* (RCL). Therefore, in each iteration all elements in S are re-ordered based on mmc, and only the top elements are stored in the RCL. This iterative process continues until a solution with k elements is constructed.

Algorithm 10 *GNE-Construction*

Output: a candidate solution $R \subseteq S$, $|R| = k$
1: $R_0 \leftarrow \emptyset$
2: **for** $p \leftarrow 1$ to $p = k$ **do**
3: $s_{max} \leftarrow argmax_{s_i \in S}(mmc(s_i))$
4: $s_{min} \leftarrow argmin_{s_i \in S}(mmc(s_i))$
5: $RCL \leftarrow \{s_i \in S | mmc(s_i) \geq s_{max} - \alpha(s_{max} - s_{min})\}$
6: $s_i \leftarrow random(RCL)$
7: $R_p \leftarrow R_{p-1} \cup s_i$
8: $S \leftarrow S \setminus s_i$
9: $R \leftarrow R_p$

Then a random element in the RCL is chosen to be part of the solution (which may not be the element with the highest contribution in the RCL). Employing these two approaches allows for different solutions to be obtained at each GRASP iteration.

The selection of the elements with the highest contributions in the RCL can be performed based on a fixed-number of candidates or on a range of values. The intuition behind limiting the size of RCL is that not every element in S contributes equally to the result set. Therefore, the method performs randomized search over the elements in the RCL that can contribute more to the solution. In our experiments we use the first approach to restrict the number of candidates in the RCL, as illustrated in Algorithm 10.

Parameter α controls how greedy and random is the *Construction* phase. For $\alpha = 0$, the *Construction* corresponds to a greedy construction procedure, and works like the GMC method. For $\alpha = 1$ the method produces a random construction. Since both GNE and GMC methods use greedy strategies to measure the individual contribution of an element to the result set, performing random selections during the construction phase for the GNE method gives the contribution of trying different elements in each multistart construction.

2) GNE Local Search Phase: Starting with the solution provided by the construction phase, the local search progressively improves it by applying a series of local modifications in the neighborhood of the current solution. Repeated applications of the construction phase yield diverse starting solutions for the local search.

Algorithm 11 *GNE-LocalSearch*

Input: candidate solution R of size k
Output: a candidate solution $R \subseteq S$, $|R| = k$
1: $s_{max} \leftarrow argmax_{s_i \in S}(mmc(s_i))$
2: $s_{min} \leftarrow argmin_{s_i \in S}(mmc(s_i))$
3: $RCL \leftarrow \{s_i \in S | mmc(s_i) \geq s_{max} - \alpha(s_{max} - s_{min})\}$
4: **for** each $s_i \in R$ **do**
5: $R' \leftarrow R$
6: **for** each $s_j \in R, s_j \neq s_i$ **do**
7: **for** $l \leftarrow 1$ to $l = k-1$ **do**
8: $s^l_i \leftarrow \{s'_i \in S | \delta^l_{div}(s_i, s'_i)\}$
9: **if** $s^l_i \notin R$ **then**
10: $R'' \leftarrow R' - s_j + s^l_i$
11: **if** $\mathcal{F}(q, R'') > \mathcal{F}(q, R')$ **then**
12: $R' \leftarrow R''$
13: **if** $\mathcal{F}(q, R') > \mathcal{F}(q, R)$ **then**
14: $R \leftarrow R'$

The proposed *local search* (Algorithm 11) performs swaps between elements in the result set R and the most diverse elements to a reference element in R. If this procedure improves the current solution, then a locally optimal solution is found. This is accomplished by a *Neighborhood Expansion* that explores the most diverse elements of each entry in the result set. This expansion is performed as follows: for each element in the result set, the $k-1$ most diverse elements are calculated; then for each of the remaining elements in the result set, a swap is performed with every element among the $k-1$ most diverse elements. If in any of these swaps a local solution is found, then the local solution is made the best optimal solution.

C. Complexity Analysis

The time complexity of GMC is $O(kn^2)$, since we have to compute the mmc for each element in S, which is $O(kn)$. GNE has two parts: Algorithm 10 which has time complexity $O(kn^2)$, and Algorithm 11 which is $O(k^3)$ (δ^l_{div} is computed only once). Nevertheless, the GNE method runs i_{max} times, which makes it slower than GMC in practice.

VI. Experimental Evaluation

We proceed with an evaluation of all methods described in this paper. The method abbreviation, name and strategy employed to construct the result set R appear in Table I. As a baseline for our comparisons, we also included the results for a random method, called *Rand*, which simply chooses, among 1,000 random runs, the result set R with the highest $\mathcal{F}$.

TABLE I
DESCRIPTION OF THE METHODS EVALUATED.

abbrv.	method name	construction of R
Swap	*Swap*	exchanging
BSwap	*BSwap*	exchanging
MMR	*Maximal Marginal Relevance*	incremental
Motley	*Motley*	incremental
MSD	*Max-Sum Dispersion*	incremental
CLT	*Clustering*	exchanging
GMC	*Greedy Marginal Contribution*	incremental
GNE	*GRASP with Neighbor Expansion*	meta-heuristic
Rand	*Random*	random

A. Setup

Five datasets from several different domains were used to evaluate each method with respect to running time, precision and $\mathcal{F}$ of the results. The datasets used in our experiments are: *faces*, *nasa*, *colors*, *docs* and *dblp*. *faces* consists of 11,900 feature vectors extracted from human face images with the Eigenfaces method[3]. *nasa*[4] and *colors*[5] datasets contain, respectively, 40,150 and 68,040 feature vectors from collection of images. Dataset *docs* has 25,960 news articles obtained from the TREC-3 collection[6], while *dblp* is a set of 2,991,212 publication entries extracted from DBLP[7]. In order to generate duplicate entries in a controlled way, and thus test the diversity in the result while increasing the λ parameter, each entry in the *dblp* dataset contains the author's name and the publication title. A DBLP publication with multiple authors thus have multiple entries in *dblp*, one entry per author, sharing the same publication title. The detailed statistics of all datasets are summarized in Table II.

In the experiments we employed the same distance function for both δ_{sim} and δ_{div}. The Manhattan distance is employed for *faces*, the Euclidean distance for *nasa* and *colors*, and the $cosine$ similarity distance for both *docs* and *dblp*. For the *faces*, *nasa*, *colors* datasets δ_{sim} was computed using part (one quarter) of the feature vector, while for δ_{div} the whole feature vector was used. For instance, for *colors* the first 8 attributes are employed to compute the δ_{sim}, and all the 32 attributes for the δ_{div}, as shown in Table II. For the *docs* dataset 10 random terms were used for δ_{sim} while the whole document was utilized for δ_{div}. The same process was employed in *dblp*, but using a few terms extracted from the publication titles.

[3]http://www.informedia.cs.cmu.edu
[4]http://www.dimacs.rutgers.edu/Challenges/Sixth/software.html
[5]http://kdd.ics.uci.edu
[6]http://trec.nist.gov
[7]http://www.informatik.uni-trier.de/∼ley/db

TABLE II
DATASETS STATISTICS.

Dataset	# of elements	Max. # of attributes	Avg. # of attributes	δ_{sim}	δ_{div}
faces	11,900	16	16	$L_1(1..4)$	$1 - L_1(1..16)$
nasa	40,150	20	20	$L_2(1..5)$	$1 - L_2(1..20)$
colors	68,040	32	32	$L_2(1..8)$	$1 - L_2(1..32)$
docs	25,276	15,116	363	$cosine(terms)$	$1 - cosine(docs)$
dblp	2,991,212	68	7	$cosine(terms)$	$1 - cosine(title)$

To generate a query set, we randomly chose 100 queries from each dataset. The extra parameters θ and θ' for *BSwap* and *Motley*, respectively, were set to 0.1, which achieved, on average, the best performance in the experiments. For the GNE we set $i_{max} = 10$ and $\alpha = 0.01$. Clearly, GNE returns better results when increasing the values for those two parameters, but at a cost of higher running time. For example, when increasing the α parameter, the GNE method slowly converges to good solutions since there are many more elements in the candidate set S to be considered. From our preliminary experiments the above values for the GNE parameters provided a good tradeoff between *quality* and running time.

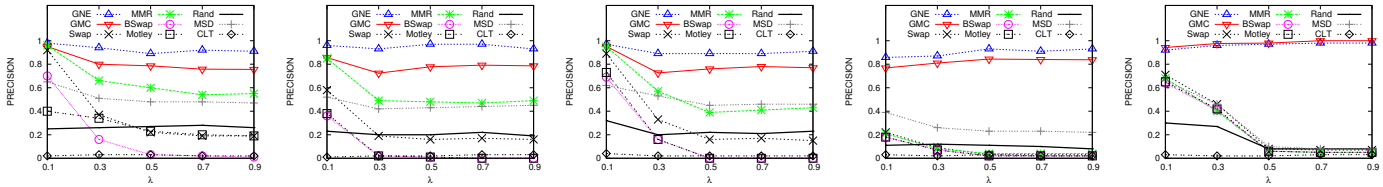
Table III summarizes the testing parameters and their corresponding ranges, in which the default value is marked in bold font. To obtain the candidate set S we employed an *off-the-shelf* retrieval system. That is, for a particular query sample, S contains the top-n elements in the dataset using the ranking function δ_{sim}. Generally speaking, each group of experimental studies is partitioned into two parts: qualitative analysis in terms of $\mathcal{F}$ and precision of each method with respect to the *optimal result set*; and scalability analysis of $\mathcal{F}$ and running time when increasing the query parameters.

TABLE III
PARAMETERS TESTED IN THE EXPERIMENTS.

Parameter	Range (**default**)
tradeoff λ values	0.1, 0.3, 0.5, **0.7**, 0.9
candidate set size $n = \|S\|$	200, **1,000**, 2,000, 3,000, 4,000
result set size $k = \|R\|$	**5**, 15, 25, 35
# of sample queries	**100**

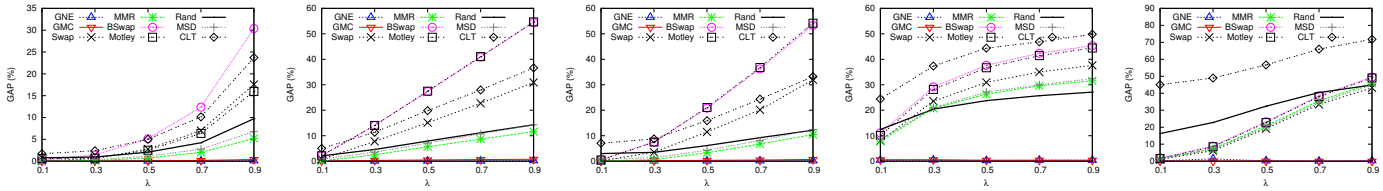
B. Qualitative Analysis

To measure a method's precision we need to compare it against the optimal result set (the one that maximizes $\mathcal{F}$). However, due to the problem complexity, we can easily compute the optimal result set for a limited range of the query parameters. Using the brute force algorithm (Algorithm 1), we computed an optimal result set for each dataset with parameters $S = 200$, $k = 5$ and λ varying from 0.1 to 0.9. The precision is measured by counting the number of common elements in the result set from Algorithm 1 and the result set



(a) *faces* (b) *nasa* (c) *colors* (c) *docs* (d) *dblp*

Fig. 1. Avg. precision vs. *tradeoff* λ values.



(a) *faces* (b) *nasa* (c) *colors* (c) *docs* (d) *dblp*

Fig. 2. Avg. gap vs. *tradeoff* λ values.

of each method using the same query parameters; clearly, the higher the precision, the better the method performs.

Figure 1 shows the average precision of each method for different values of λ (as λ increases, preference to diversity increases). Note that, the precision of *all* methods, except GNE and GMC, typically decreases when increasing λ. Interestingly, four of the previous approaches (*Swap*, *BSwap*, *Motley* and CLT) performed worse than the *Rand* method when λ is greater than 0.5. That is, if diversity is more prominent than relevance, these methods are easily outperformed.

GNE and GMC outperformed all the methods and showed almost constant precision regarding λ. Only in the case where $\lambda = 0.1$ (i.e., when relevance is much more important than diversity), the MMR method have precision similar to GNE/GMC for three datasets (*faces*, *nasa* and *colors*). In this particular case, the optimal result set has many, or all, elements as the top-k result, and since MMR scans S ordered by δ_{sim}, the precision of MMR is high for result sets with very low diversification. Previous methods are influenced by the first most relevant element in their result set, which affects the result precision when diversity has a higher weight than relevance. In contrast, the variation of λ does not affect the precision of GNE and GMC. The average precision for GNE and GMC was at least 75%.

The precision of GNE is better than GMC in all datasets tested, except for the *dblp* dataset. This is because GNE selects elements from RCL using a randomized approach. That is, GNE may not always select the element with the highest contribution to the result set, like GMC. Furthermore, the GNE also uses the local search phase that, given an initial solution, aims at finding better solutions.

We also compared the $\mathcal{F}$ values of each method against the $\mathcal{F}$ value of the *optimal result set*. This experiment is intended to measure how close the $\mathcal{F}$ for a result set is to the maximal $\mathcal{F}$ value. Figure 2 depicts the "gap" in $\mathcal{F}$, i.e., the difference between the $\mathcal{F}$ of a particular method and the optimal $\mathcal{F}$, divided by the optimal $\mathcal{F}$. The *gap* for GNE and GMC is close to zero for every dataset and values of λ, which indicates that these two methods constructed results sets with maximal $\mathcal{F}$ values, while it increases with λ for the other methods. Again *BSwap*, *Swap*, *Motley* and CLT are typically outperformed by the *Rand* method for high values of λ.

The effectiveness of GMC and GNE is also depicted in Table IV which illustrates the result sets for four example queries, using the *dblp* dataset ($k = 5$ and $S = 200$) with $\lambda = 0$ (*no diversification*), $\lambda = 0.3$ (*moderate diversification*) and $\lambda = 0.7$ (*high diversification*). Only GNE is shown since the results for GMC are similar. When $\lambda = 0$ the result set contains the top-5 elements of S ranked with the δ_{sim} scoring function. Since the *dblp* dataset contains several duplicate entries, the result sets with *no diversification* contain several duplicate elements (we also show the author names for each entry as an *id* for that entry). As λ increases, less duplicates are found in the result set, and the elements in the result set "cover" many more subjects (defined by terms in the publication name). Observe that the result set with *high diversification* contains elements that all have the query terms, as well as other terms indicating that the publication is related to different subjects (i.e., hardware accelerator, operations research, privacy in spatiotemporal) among the other publications in the result set.

C. Scalability

We next examine the scalability of the methods with respect to $\mathcal{F}$ values and running time while varying the query parameters λ (from 0.1 to 0.9), k (from 5 to 35) and S (from 200 to 4,000). Due to lack of space, in the rest of the paper we omit the results for the *faces* and *nasa* datasets since they are similar to *colors*.

Figure 3 shows $\mathcal{F}$, with $k = 5$ and $S = 1,000$, when increasing λ. Regardless of the value of λ, the $\mathcal{F}$ values for GNE and GMC are higher among all other methods for any dataset. In the *colors* dataset, $\mathcal{F}$ decreases when increasing λ because the δ_{div} values among elements in S are not as

TABLE IV

RESULT SETS FOR *dblp* DATASET: TOP-5 RESULTS ($\lambda = 0$), AND TWO DIVERSIFIED RESULTS ($\lambda = 0.3$ AND 0.7) PRODUCED BY GNE.

	top-k: no diversification ($\lambda = 0$)	GNE: moderate diversification ($\lambda = 0.3$)	GNE: high diversification ($\lambda = 0.7$)
	q_1 = **database systems**		
1:	Y.Theodoridis: trajectory **database systems**	C.Date: an introduction to **database systems**	R.Deshpande: an electronic **database** delivery **system**
2:	I.Ntoutsi: trajectory **database systems**	J.Rothnie: distributed **database systems**	T.Xu: pbmice: an integrated **database system** of piggyBac ...
3:	N.Pelekis: trajectory **database systems**	N.Pelekis: trajectory **database systems**	C.Matheus: **systems** for knowledge discovery in **databases**
4:	E.Frentzos: trajectory **database systems**	M.Wu: the dbo **database system**	D.Dimitroff: an intelligent image **database system**
5:	A.Elmagarmid: videotext **database systems**	J.Gray: parallel **database systems** 101	F.Ozcan: metu interoperable **database system**
	q_2 = **nearest neighbor**		
1:	H.Alt: the **nearest neighbor**	H.Alt: the **nearest neighbor**	W.Tiller: algorithm for finding all k **nearest neighbors**
2:	G.Yuval: finding **nearest neighbors**	G.Yuval: finding **nearest neighbors**	L.Wu: stepwise **nearest neighbor** discriminant analysis
3:	F.Yao: on **nearest-neighbor** graphs	C.Domeniconi: **nearest neighbor** ensemble	B.Pittel: the random bipartite **nearest neighbor** graphs
4:	M.Paterson: on **nearest-neighbor** graphs	P.Grother: fast implm. of **nearest neighbor** classifiers	R.Meiche: a hardware accelerator for k-th **nearest neighbor** ...
5:	D.Eppstein: on **nearest-neighbor** graphs	M.Paterson: on **nearest-neighbor** graphs	X.Yu: the research on an adaptive k-**nearest neighbors** classifier
	q_3 = **data mining**		
1:	D.Olson: **data mining**	K.Rihaczek: **data mining**	S.Olafsson: operations research and **data mining**
2:	J.Kunz: **data mining**	J.Kunz: **data mining**	S.Kaya: privacy in spatiotemporal **data mining**
3:	K.Rihaczek: **data mining**	D.Olson: **data mining**	M.Steinbach: top 10 algorithms in **data mining**
4:	S.Morishita: **data mining** tools for the ...	B.Toursel: distributed **data mining**	P.Perner: **data mining** concepts and techniques
5:	Y.Ohya: **data mining** tools for the ...	R.Agrawal: whither **data mining**?	C.G-Carrier: personalizing e-commerce with **data mining**
	q_4 = **medical image**		
1:	O.Pianykh: **medical image** enhancement	P.Wang: **medical image** processing	M.Moshfeghi: elastic matching of multimodality **medical images**
2:	J.Tyler: **medical image** enhancement	R.Deklerck: segmentation of **medical images**	S.Zagal: segmentation of **medical images** by region growing
3:	M.Trifas: **medical image** enhancement	C.Meinel: compression of **medical images**	C-C.Chang: detection and restoration of a tampered **medical image**
4:	C.Meinel: compression of **medical images**	J.Tyler: **medical image** enhancement	C.Faloutsos: similarity searching in **medical image** databases
5:	S.Hludov: compression of **medical images**	C-H.Lin: quality of compressed **medical images**	P.Radeva: discriminant snakes for 3d reconstruction in **medical images**

high as their δ_{sim} values. The difference between $\mathcal{F}$ values for GNE and GMC and the other methods are greater as the value of λ increases. This is due to the fact that in all previous methods the diversification of the result set is performed giving preference on the order that the elements are stored in S, which are ordered by δ_{sim}. In other words, all previous approaches do not diversify result sets for arbitrary values of λ, as opposed to GNE and GMC.

Figure 4 shows $\mathcal{F}$ when increasing k, with $\lambda = 0.7$ and $S = 1,000$. The $\mathcal{F}$ values decrease when increasing k because the δ_{div} values among elements in R tend to be smaller for the same candidate set S. In addition, more elements are considered in the computation of $\mathcal{F}$, which in turn decreases $\mathcal{F}$. Regardless of the value of k, the $\mathcal{F}$ values for GNE and GMC are again higher among all other methods and datasets.

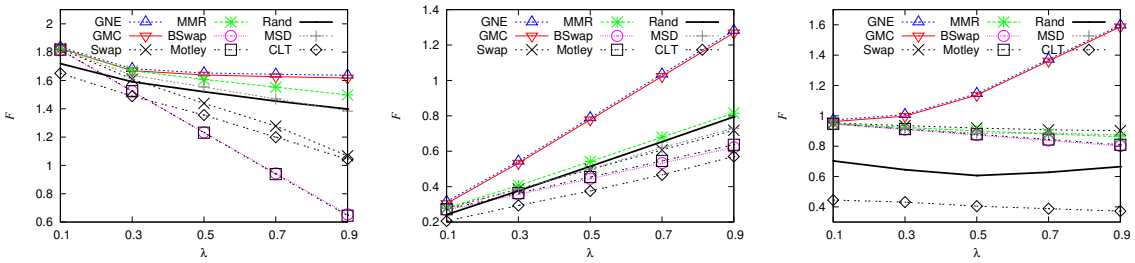
Figure 5 depicts the experimental results when increasing S, with $k = 5$ and $\lambda = 0.7$. For the *dblp* dataset the values of $\mathcal{F}$ are constant for all methods, except *Rand* and CLT. This is because, in the *dblp* dataset there are many more elements that do not have a high value of δ_{sim} and are included in the candidate set S. For this dataset, a good tradeoff between relevance and diversity is found for R of size $k = 5$ with a small size of S (e.g., $S = 200$). For the other two datasets, *colors* and *docs*, a larger size of S is necessary to find a good result set R, in terms of $\mathcal{F}$, with the required tradeoff ($\lambda = 0.7$) between relevance and diversity. Nevertheless, the value of $\mathcal{F}$ in both datasets increases slowly.

Interestingly, the value of $\mathcal{F}$ in the CLT and *Rand* methods, for the *docs* and *dblp* datasets, tends to decrease as the size of S increases. This is because in both methods a fixed number of runs (i.e., 1,000) is performed in order to select the best solution, but as the number of candidates increases more runs are needed for these two methods to converge to good solutions. As a result, they do not scale well for large sizes of S. The other methods do not exhibit this behavior since elements in S are analyzed only once, and then ranked by a scoring function.

Overall, the highest $\mathcal{F}$ values were achieved from the GNE and GMC methods, for every dataset or query parameters. GNE provides slightly better $\mathcal{F}$ values (up to 2% higher than the GMC). Since the GNE approach randomly selects an element in the RCL to be included in the result set R, its performance depends not only on the size of the RCL (defined by α), but also on the size of S. This is because i_{max} is affected by α and the distribution of contribution of each element in S (measured using mmc). Therefore, increasing S for the same α also increases the size of RCL, which enables the GNE method *slowly* to converge to good result sets. One possible approach to cope with this, without interfering with GNE's running time, is to limit the size of RCL and/or increase the value for the i_{max} parameter.

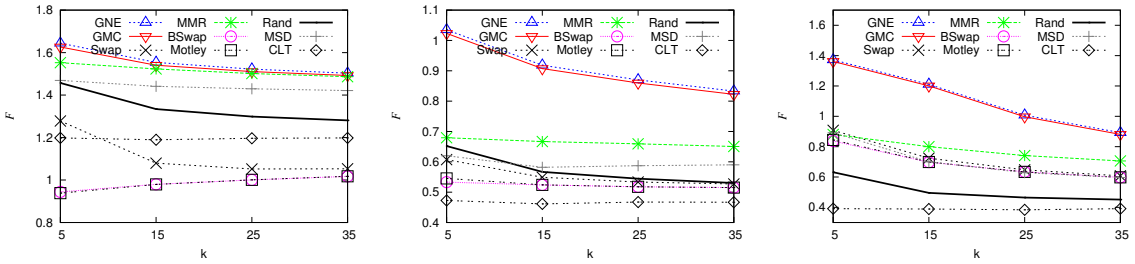
Finally, Figure 6 depicts the running time (in *ms*) for the *dblp* dataset when varying λ ($k = 5$ and $S = 1,000$), k ($\lambda = 0.7$ and $S = 1,000$) and S ($k = 5$ and $\lambda = 0.7$). The same behavior was observed for the running time of the other four datasets and is thus omitted. As expected, the running times are not affected when increasing λ (Figure 6(a)). This parameter does not interfere on how a method operates, but only dictates the selection of elements in the result set. The GNE method has the highest running time among all methods, since it executes a few runs over the candidate set S and has an expensive local search phase in terms of the number of swaps performed.

In Figure 6(b), running times increase proportionally to the size of k, since more iterations are performed over S to build R. The *BSwap* and *Motley* exhibit the slowest increase. The



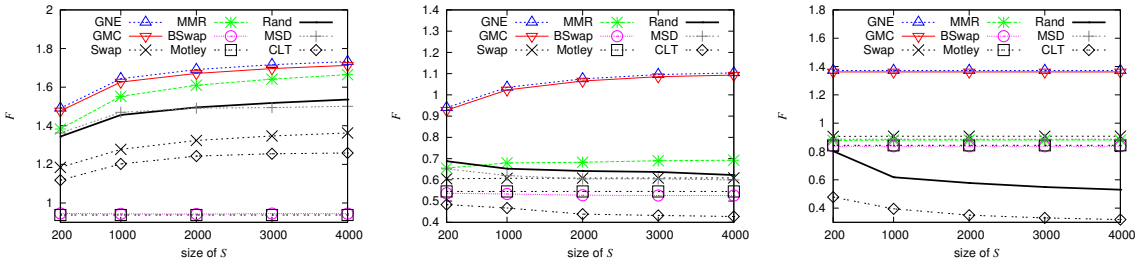
(a) *colors* (b) *docs* (c) *dblp*

Fig. 3. Avg. $\mathcal{F}$ value vs. *tradeoff* λ values.



(a) *colors* (b) *docs* (c) *dblp*

Fig. 4. Avg. $\mathcal{F}$ value vs. result set k size.



(a) *colors* (b) *docs* (c) *dblp*

Fig. 5. Avg. $\mathcal{F}$ value vs. candidate set S size.

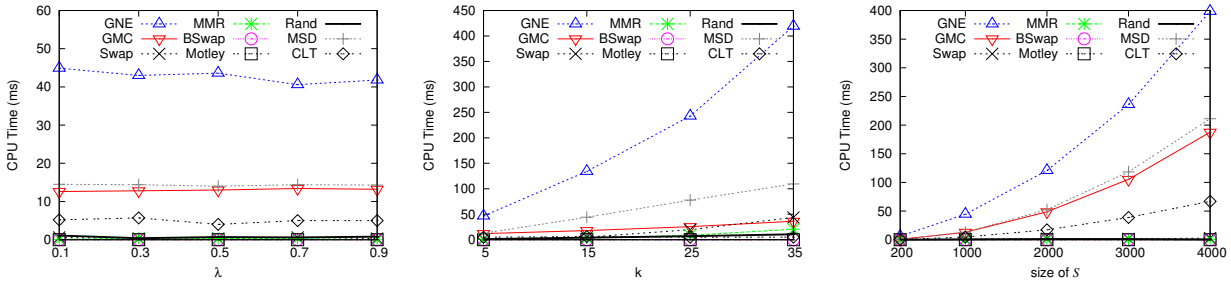
running times for GNE and GMC increase for large values of k since both compute, for each element in S, the $k-1$ elements in S that have the highest δ_{div} values. Nevertheless, GMC is competitive for larger values of k to other methods, having a similar behavior to *Swap*. As for GNE, the larger increase is because the number of runs and exchanges in the local search phase are both proportional to k.

In Figure 6(c), the running time of all methods, except for *BSwap* and *Motley*, increases with S, since every element in S has to be checked. Since *BSwap* and *Motley* have early stopping conditions, defined by the threshold conditions θ and θ', respectively, their performance is not directly influenced by the size of S, but by their threshold values. Again, GNE had the highest running times since it performs several runs, which depend on k and S. GMC and MSD have again similar behavior (increasing with S).

D. Discussion

The *BSwap*, *Swap*, *Motley* and CLT methods were outperformed by the *Rand* approach in terms of $\mathcal{F}$ values in almost all experiments, which indicates that a simple random selection returns better results than these four methods. As expected, the CLT is, among all methods, the one that achieved the lowest $\mathcal{F}$ values for the majority of the experiments. This last result corroborates our initial claim that clustering techniques do not apply well for query result diversification. Besides having an extra parameter that is not straightforward to tune, both *BSwap* and *Motley* provided the lowest precision and $\mathcal{F}$ values in their results. Among the previous methods tested (i.e., excluding GNE and GMC), MMR was typically better in most experiments.

The GNE and GMC methods consistently achieved better results in terms of precision, *gap* and $\mathcal{F}$ values for every dataset and query parameter tested. Among them, GNE generally provided the best results, since it performs several runs in a controlled way. However, this affects its running time; GNE was the slowest method among all competitors. GMC achieved running time which was similar to MSD and CLT. Overall, GMC provided high quality results with acceptable running times. If the user is willing to further improve quality at the expense of higher running time, then GNE should be preferred over GMC.



(a) *dblp*: λ (b) *dblp*: k (c) *dblp*: S

Fig. 6. Avg. running time vs. λ, k and S for the *dblp* dataset.

VII. Conclusion

In this paper we describe a simple yet powerful framework for evaluation and optimization of methods for diversifying query results. One advantage of this framework is that users can set the *tradeoff* between finding the most relevant elements to the query and finding diverse elements in the result set. Based on this framework, we then describe several known methods, as well as two new methods, for diversifying query results. The two new proposed methods, named GMC and GNE, construct the result set in an incremental way using a function that computes the contribution of each candidate element considering its relevance to the query and its diverse value to the elements in the current result set and to other elements in the candidate set. In GMC, the element with the highest contribution value is always included in the result set, while in GNE, a randomized procedure is employed to choose, among the top highest elements, the one to be included in the result set. We show a thorough experimental evaluation of the various diversification methods using real datasets. GMC and GNE methods outperformed all known methods regarding the optimization metric of our framework for any *tradeoff* values between relevance and diversity in the result set. As a next step in this research, we are planning to perform experiments using the TREC ClueWeb09 dataset[8].

Acknowledgements: This research was partially supported by NSF IIS grants 0705916, 0803410 and 0910859. Vieira's work was funded by a CAPES/Fulbright Ph.D fellowship. Razente's work was funded by a Fapesp Ph.D. scholarship 06/00336-5.

References

[1] H. Chen and D. Karger, "Less is more: probabilistic models for retrieving fewer relevant documents," in *Proc. ACM SIGIR*, 2006.
[2] S. Gollapudi and A. Sharma, "An axiomatic approach for result diversification," in *Proc. WWW*, 2009.
[3] D. Rafiei, K. Bharat, and A. Shukla, "Diversifying web search results," in *Proc. WWW*, 2010.
[4] Z. Liu, P. Sun, and Y. Chen, "Structured search result differentiation," *PVLDB*, vol. 2, no. 1, 2009.
[5] E. Demidova et al., "DivQ: Diversification for keyword search over structured databases," in *Proc. ACM SIGIR*, 2010.
[6] F. Radlinski and S. Dumais, "Improving personalized web search using result diversification," in *Proc. ACM SIGIR*, 2006.
[7] C. Clarke et al., "Novelty and diversity in information retrieval evaluation," in *Proc. ACM SIGIR*, 2008.
[8] K. Liu, E. Terzi, and T. Grandison, "Highlighting diverse concepts in documents," in *Proc. SDM*, 2009.
[9] J. Carbonell and J. Goldstein, "The use of MMR, diversity-based reranking for reordering documents and producing summaries," in *Proc. ACM SIGIR*, 1998.
[10] E. Ioannou et al., "Efficient semantic-aware detection of near duplicate resources," in *Proc. ESWC*, 2010.
[11] R. Agrawal et al., "Diversifying search results," in *Proc. ACM WSDM*, 2009.
[12] B. Carterette, "An analysis of NP-completeness in novelty and diversity ranking," in *Proc. ICTIR*, 2009.
[13] M. Coyle and B. Smyth, "On the importance of being diverse: Analysing similarity and diversity in web search," in *Proc. IIP*, 2004.
[14] C. Yu, L. Lakshmanan, and S. Amer-Yahia, "It takes variety to make a world: diversification in recommender systems," in *EDBT*, 2009.
[15] B. Smyth and P. McClave, "Similarity vs. diversity," in *Proc. ICCBR*, 2001.
[16] A. Jain, P. Sarda, and J. Haritsa, "Providing diversity in k-nearest neighbor query results," in *Proc. PAKDD*, 2004.
[17] E. Vee et al., "Efficient computation of diverse query results," in *Proc. IEEE ICDE*, 2008.
[18] O. Prokopyev, N. Kong, and D. Martinez-Torres, "The equitable dispersion problem," *European J. of Operational Research*, 2009.
[19] C.-C. Kuo, F. Glover, and K. Dhir, "Analyzing and modeling the maximum diversity problem by zero-one programming," *Dec. Sci.*, 2007.
[20] R. Hassin, S. Rubinstein, and A. Tamir, "Approximation algorithms for maximum dispersion," *Oper. Res. Lett.*, vol. 21, no. 3, 1997.
[21] M. J. Kuby, "Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems," *Mathematical and Computer Modelling*, vol. 10, no. 10, 1988.
[22] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *J. of Global Optimization*, vol. 6, 1995.
[23] G. Silva et al., "New heuristics for the maximum diversity problem," *J. of Heuristics*, vol. 13, no. 4, 2007.
[24] M. Hadjieleftheriou and V. J. Tsotras, Eds., *Special Issue on Result Diversity*, vol. 32, no. 4. IEEE Data Eng. Bull., 2009.
[25] R. Leuken et al., "Visual diversification of image search results," in *Proc. WWW*, 2009.
[26] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proc. ICML*, 2008.
[27] C.-N. Ziegler et al., "Improving recommendation lists through topic diversification," in *Proc. WWW*, 2005.
[28] R. Santos et al., "Explicit search result diversification through sub-queries," in *Proc. ECIR*, 2010.
[29] R. Santos, C. Macdonald, and I. Ounis, "Exploiting query reformulations for web search result diversification," in *Proc. WWW*, 2010.
[30] C. Zhai, W. Cohen, and J. Lafferty, "Beyond independent relevance: methods and evaluation metrics for subtopic retrieval," in *Proc. ACM SIGIR*, 2003.
[31] B. Zhang et al., "Improving web search results using affinity graph," in *Proc. ACM SIGIR*, 2005.
[32] X. Zhu et al., "Improving diversity in ranking using absorbing random walks," in *Proc. NAACL-HLT*, 2007.
[33] O. Chapelle et al., "Expected reciprocal rank for graded relevance," in *Proc. CIKM*, 2009.
[34] M. Drosou and E. Pitoura, "Diversity over continuous data," *IEEE Data Eng. Bull.*, vol. 32, no. 4, 2009.
[35] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[8] http://boston.lti.cs.cmu.edu/Data/clueweb09