The Tantalizing New Prospect of Index-Based Diversified Retrieval

George Tsatsanifos
supervised by Timos Sellis

gtsat@dblab.ece.ntua.gr

SIGMOD'13 PhD Symposium

New York, NY, USA, June 23, 2013

Outline

Outline

Is this problem really important?

Search Result Diversification addresses a variety of problems

1. counteracts over-specialization
   - when retrieving too homogenuous results
   - users quickly stop as they do not expect to learn more
   - need to ameliorate user satisfaction
   - need to decrease query abandonment
2. reduces the risk that none of the results satisfies a user's intent.
3. facilitates (near) duplicates elimination.

Conventional Search Limitations

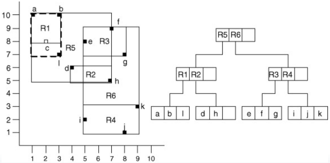
- It is in general insufficient to simply return a set of relevant results, since correlations among them are also important.
- Documents should be selected progressively according to the relevance of the documents that come before it.
- Therefore, there is the need to identify the documents that are relevant and novel.
- Find the relevant documents that are dissimilar to the previously delivered, and thus comprise new information.

Challenges

- Relevance corresponds to documents' similarity to the query.
- Diversity is defined by how much each document of the answer-set differs from the others.
- These two important goals are contradictory to each other,
- But still they must be combined. How exacly...?

Motivation

- We address this problem over multi-dimensional disk-resident data.
- We employ the R-Tree to serve as an indexing infrastructure.



Our goal is to ameliorate performance (IO and execution time) by accessing only disk pages that can actually contribute to the result.

Outline

Preliminaries

Given query object q, a universe of objects U, its subset $S \subset U$,

- let ranking function $f(S|q)$ which quantifies S's diversity properties.
- Also, let object $o \in U \setminus S$ to be added in the result.
- Then, how much is $f(S \cup \{o\}|q) - f(S|q)$?
- How are affected the diversity properties of S by inserting o' instead.
- Is there a way to capture indexed objects' eligibility beforehand?

Definitions

- Let $\phi(o|S,q) = f(S \cup \{o\}|q) - f(S|q)$.
- Henceforth, we assume that well diversified sets achieve **low** values.
- Then, it holds for the best object o' to be added in S that $\phi(o'|S) \leq \phi(o|S),\ \forall o \in U \setminus S$.
- We generalize ϕ for high-dimensional intervals in the form $[\vec{\ell}, \vec{h}]$.
- More formally, $\phi([\vec{\ell}, \vec{h}]|\vec{q}, S) = \min_{\vec{p} \in [\vec{\ell}, \vec{h}]} \phi(\vec{p}|\vec{q}, S)$, and can be approximated by a lower bound depending on f's form.
- Thereby, we are in position of comparing two different branches of the R-tree in terms of how promising they are.

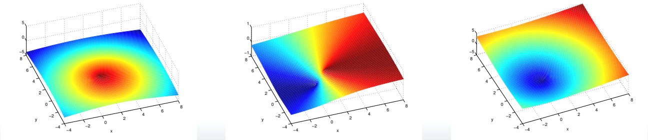
Ranking Function #1: min-sum

For example,

- if $f(S|q) = \lambda \sum_{s \in S} d(s,q) - \frac{1-\lambda}{|S|^2} \sum_{s_1 \in S} \sum_{s_2 \in S} d(s_1, s_2)$,
- then, $\phi(o|S) = \lambda d(o,q) - \frac{1-\lambda}{|S|} \sum_{s \in S} d(o,s)$,
- and $\phi([\vec{\ell}, \vec{h}]|S)$ can be approximated by a lower bound as, $\phi([\vec{\ell}, \vec{h}]|\vec{q}, S) \geq \lambda \min_{\vec{x} \in [\vec{\ell}, \vec{h}]} d(\vec{x}, \vec{q}) - \frac{1-\lambda}{|S|} \sum_{\vec{s} \in S} \max_{\vec{y} \in [\vec{\ell}, \vec{h}]} d(\vec{y}, \vec{s})$.

Search Space Restrictions -i-

- An insight of the score density according to ϕ for 2 dimensions.
- Well diversified objects achieve low values.
- Better ranked objects are located in the blue areas.

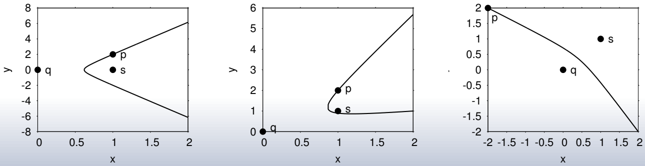


(a) $\lambda = 0.3$ (b) $\lambda = 0.5$ (c) $\lambda = 0.7$

Figure: Score density distribution for $\vec{s} = (1, 0)$.

Search Space Restrictions -ii-

- Ranking function ϕ serves as a discriminant function dividing the key-space into two distinct parts.
- Given an object $\vec{p}$ with its ϕ-value equal to a score threshold
- The possible positions for the objects that achieve the same score as $\vec{p}$ are depicted with a solid line
- All objects at the left of the dividing curve are better ranked than $\vec{p}$.

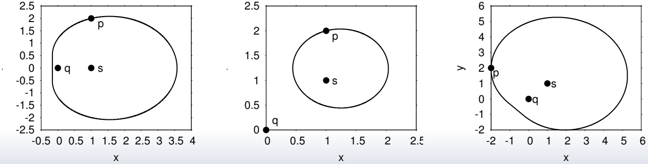


(a) $\vec{s} = (1, 0)$, $\vec{p} = (1, 2)$ (b) $\vec{s} = (1, 1)$, $\vec{p} = (1, 2)$ (c) $\vec{s} = (1, 1)$, $\vec{p} = (-2, 2)$

Figure: Dividing curves for $\lambda = 0.5$.

Search Space Restrictions -iii-

- When $\lambda \neq 0.5$ the balance between relevance and diversity is broken.
- We are interested in the area outside the curves for $\lambda < 0.5$.
- Only nodes who overlap with this area are read when searching for a better ranked object.



(a) $\vec{s} = (1, 0)$, $\vec{p} = (1, 2)$

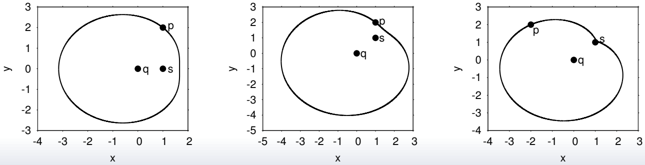
(b) $\vec{s} = (1, 1)$, $\vec{p} = (1, 2)$

(c) $\vec{s} = (1, 1)$, $\vec{p} = (-2, 2)$

Figure: Dividing curves for $\lambda = 0.3$.

Search Space Restrictions -iv-

- We are interested in the enclosed area for $\lambda > 0.5$.
- When more items are comprised in S, we search for better ranked objects in the area which corresponds to the **intersection** of all areas with improved objects for each element $\vec{s_i} \in S$.



(a) $\vec{s} = (1, 0)$, $\vec{p} = (1, 2)$

(b) $\vec{s} = (1, 1)$, $\vec{p} = (1, 2)$

(c) $\vec{s} = (1, 1)$, $\vec{p} = (-2, 2)$

Figure: Dividing curves for $\lambda = 0.7$.

Ranking Function #2: min-max

Now, let $f(S|q) = \lambda \max_{s \in S} d(s, q) - (1 - \lambda) \min_{s_1 \in S, s_2 \in S} d(s_1, s_2)$. Thereby,

- if $d(\vec{p}, \vec{q}) \leq \max_{\vec{x} \in S} d(\vec{x}, \vec{q})$ and $\min_{\vec{x} \in S} d(\vec{p}, \vec{x}) \geq \min_{\vec{y}, \vec{z} \in S} d(\vec{y}, \vec{z})$, then $\phi(\vec{p}|\vec{q}, S) = 0$,
- if $d(\vec{p}, \vec{q}) > \max_{\vec{x} \in S} d(\vec{x}, \vec{q})$ and $\min_{\vec{x} \in S} d(\vec{p}, \vec{x}) \geq \min_{\vec{y}, \vec{z} \in S} d(\vec{y}, \vec{z})$, then $\phi(\vec{p}|\vec{q}, S) = \lambda(d(\vec{p}, \vec{q}) - \max_{\vec{x} \in S} d(\vec{x}, \vec{q}))$,
- if $d(\vec{p}, \vec{q}) \leq \max_{\vec{x} \in S} d(\vec{x}, \vec{q})$ and $\min_{\vec{x} \in S} d(\vec{p}, \vec{x}) < \min_{\vec{y}, \vec{z} \in S} d(\vec{y}, \vec{z})$, then $\phi(\vec{p}|\vec{q}, S) = (1 - \lambda)(\min_{\vec{x}, \vec{y} \in S} d(\vec{x}, \vec{y}) - \min_{\vec{z} \in S} d(\vec{p}, \vec{z}))$,
- otherwise, we take that, $\phi(\vec{p}|\vec{q}, S) = \lambda(d(\vec{p}, \vec{q}) - \max_{\vec{x} \in S} d(\vec{x}, \vec{q})) + (1 - \lambda)(\min_{\vec{y}, \vec{z} \in S} d(\vec{y}, \vec{z}) - \min_{\vec{x} \in S} d(\vec{p}, \vec{x}))$.

Ranking Function #3: max-rank

Also applicable when sum/min combine, or the sign of the function changes, and hence, we are interested in high values instead, as in

$$f(S|q) = (1-\lambda)\min_{s_1,s_2\in S} d(s_1,s_2) - \lambda \sum_{s\in S} d(s,q)$$

Thereby,

- when $\min_{s_1,s_2\in S} d(s_1,s_2) \leq \min_{s\in S} d(o,s)$, we have that $\phi(o|S) = -\lambda d(o,s)$
- otherwise, $\phi(o|S) = -\lambda d(o,s) - (1-\lambda)(\min_{s_1,s_2\in S} d(s_1,s_2) - \min_{s\in S} d(o,s))$

One step at a time

(Sub-)Problem Definition

Given a universe of objects U, its subset $S \subset U$, a query object $\vec{q}$, and a ranking function f, we want to find the object $\vec{p} \in U \setminus S$, which when added to S we obtain the most diversified result, $f(S \cup \{\vec{p}\}|\vec{q}) \leq f(S \cup \{\vec{x}\}|\vec{q}), \forall p, x \in U \setminus S$.

Diversified Retrieval

In essence the best way to access the disk includes,

1. Initializing a heap with the root node.
2. While the heap is not empty
   - pop at each iteration the R-tree node that is ranked highest.
   - if a leaf node is encountered break and return the result.
   - otherwise insert into the heap the children nodes that satisfy a given diversity threshold (which becomes stricter and stricter).

The leaf node encountered first is guaranteed to achieve a better score than any other leaf by construction.

Other Aspects

- What if we just want to improve S instead of augmenting it with the most "diverse" indexed object?
- Should we try to keep the best available subset of S, where $f(S \setminus \{s'\}) \leq f(S \setminus \{s\}), \forall s \in S$?
- No! The best reduced result is not necessarily part of the best answer. What if it can be augmented with worse replacements only?
- Should we replace the "worst" element $s' \in S$, with $\phi(s'|S \setminus \{s'\}) \geq \phi(s|S \setminus \{s\})$?
- No! The least diverse element of the result is not necessarily the best option to remove. A better element from S might exist to be replaced with an item that overall diversifies S a great deal.
- In general, we choose s', o' in such a way that, $f((S \setminus \{s'\}) \cup \{o'\}) \leq f((S \setminus \{s\}) \cup \{o\}),\ \forall s \in S, \forall o \in U \setminus S.$

Outline

In search for a better result

Starting with initial result S, we want to replace s with p, so that

$$\begin{aligned} f(S') &< f(S) \\ f(S \setminus \{\vec{s}\} \cup \{\vec{p}\}) &< f(S) \\ f(S \setminus \{\vec{s}\}) + \phi(\vec{p}|S \setminus \{\vec{s}\}) &< f(S \setminus \{\vec{s}\}) + \phi(\vec{s}|S \setminus \{\vec{s}\}) \\ \phi(\vec{p}|S \setminus \{\vec{s}\}) &< \phi(\vec{s}|S \setminus \{\vec{s}\}) \qquad (1) \end{aligned}$$

Refinement

Then, the next replacement should be better that the previous,

$$
\begin{aligned}
f(S'') &< f(S') \\
f(\underbrace{S \setminus \{\vec{s_j}\}}_{S_j} \cup \{\vec{p_j}\}) &< f(\underbrace{S \setminus \{\vec{s_i}\}}_{S_i} \cup \{\vec{p_i}\}) \\
f(S_j) + \phi(\vec{p_j}|S_j) &< f(S_i) + \phi(\vec{p_i}|S_i) \\
\phi(\vec{p_j}|S_j) &< \phi(\vec{p_i}|S_i) + \underbrace{f(S_i) - f(S_j)}_{\delta}
\end{aligned} \tag{2}
$$

Consequently, the threshold becomes even stricter for $\delta < 0$!

And this is what we will show how to do next.

Optimizations -i-

The turn we examine each element of S is also important.
Assume that we visit s_i before s_j, iff $\phi(s_i) \geq \phi(s_j)$.

$$\begin{aligned} f(\overbrace{S \setminus \{\vec{s_i}\}}^{S_i} \cup \{\vec{s_i}\}) &= f(\overbrace{S \setminus \{\vec{s_j}\}}^{S_j} \cup \{\vec{s_j}\}) \\ f(S_i \cup \{\vec{s_i}\}) &= f(S_j \cup \{\vec{s_j}\}) \\ f(S_i) + \phi(\vec{s_i}|\vec{q}, S_i) &= f(S_j) + \phi(\vec{s_j}|\vec{q}, S_j) \\ f(S_i) &\leq f(S_j) \end{aligned} \tag{3}$$

And thus, result-set S_i constitutes a better answer than S_j.

Optimizations -ii-

So, why don't we try to improve S_i first!

- Then, refining S_j is more focused since we only search for objects that would make it at least as good as $S_i \cup \{p_i\}$.
- The next replacements must be very highly ranked in order to compensate for starting from a worse partial result S_j.
- Only a small part of the key-space corresponds to such quality.
- Non-overlapping R-Tree nodes are never accessed.
- In effect, whole branches of the R-Tree are pruned accordingly.

Putting it all together

1. Starting from an initial result-set S,
2. Sort the elements of S by their ϕ-value descending.
3. For each element s_i in S find its optimal stored replacement.
4. Set the threshold value equal to the score of the candidate result, $S' \leftarrow (S \setminus \{s_i\}) \cup \{p_i\}$.
5. Continue with the next element s_j.
6. Search for s_j's replacement **only** in the part of the keyspace that contains objects that result in an answer-set with a score better than the previous threshold set by s_i's replacement in S'.
7. If there is such a point keep the new result, $S'' \leftarrow (S \setminus \{s_j\}) \cup \{p_j\}$.
8. Repeat until S cannot be improved anymore.

Outline

Setting

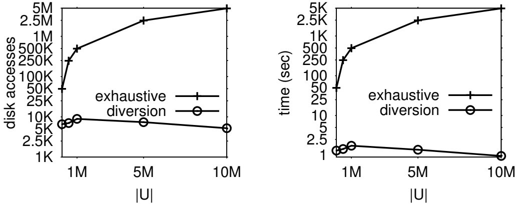
- We use the **MinMax** function because of its properties.
- We want to distribute the representatives evenly, regardless of the densities of the underlying clusters.
- Specifically, **MinSum** returns more objects from a dense cluster
  1. reduces distances of many points to their nearest representatives
  2. outweights the benefit of trying to reduce such distances of points in a faraway sparse cluster

Parameter	Range	Default
dataset size $\lvert U \rvert$	100K, 500K, 1M, 5M, 10M	1M
dimensionality $\lvert D \rvert$	$2, 3, 5, 7, 11, 13$	3
result-size K	10,20,30,40,50,60,70,80,90,100	30

Table: System parameters and configurations.

Results -i-: More Data

- As the data grow larger, so does the R-tree, and therewith disk accesses increase.



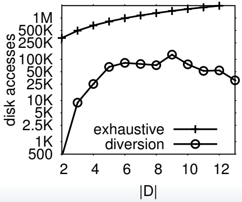
(a) Disk accesses (b) Execution time

Figure: in terms of $|U|$

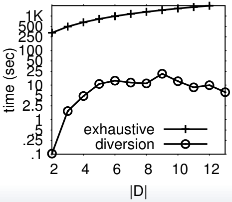
Results -ii-: The curse of dimensionality

Dimensionality causes the R-tree to grow bigger

1. High-dimensional entries require more space
2. Fewer entries can fit into a single disk page



(a) Disk accesses

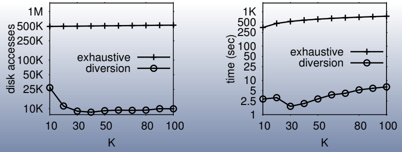


(b) Execution time

Figure: in terms of $|D|$

Results -iii-: The result-size effect

- Increasing the result-size has a bilateral impact on performance.
- More items need to be examined in the result-set whether they should be replaced or not.
- We expected that performace would impair due to the additional operations for computing all possible replacements.
- BUT, when examining one item from S, there are $K-1$ other objects restricting the searched area of the key-space.
- And thus, as K increases, more restrictions are imposed and performance ameliorates (until some K value).



(a) Disk accesses (b) Execution time

Improvements

- Can we do any better?
- Actually, when starting from a greedily constructed initial result-set, about $\frac{1}{4}$ of the IO shown is required overall.
- Including creating the original set!
- The initialization policy of the result-set should be further studied.

Outline

Axiomatic Approach for Diversification i

A 2-Approximation Algorithm for the MinDispersion Problem.

Input : Universe U, k
Output: Set S ($|S| = k$) that maximizes $f(S)$

Initialize the set $S = \emptyset$; Find $(u, v) = \text{argmax}_{x,y \in U}\, d(x, y)$ and set $S = \{u, v\}$; For any $x \in U \setminus S$, define $d(x, S) = \min_{u \in S} d(x, u)$;
while $|S| < k$ **do**
    Find $x \in U \setminus S$ such that $x = \text{argmax}_{x \in U \setminus S}\, d(x, S)$;
    Set $S = S \cup \{x\}$;
end

where $d'(u, v) = \lambda d(u, v) - \frac{1}{2}(d(u, q) + d(v, q))$

- Time consuming process as $O(K^2|U|)$ time is required.
- K iterations where $|U|$ items are compared to $O(K)$ from the result.

Axiomatic Approach for Diversification

A 2-Approximation Algorithm for the SumDispersion Problem.

Input : Universe U, k
Output: Set S ($|S| = k$) that maximizes $f(S)$

Initialize the set $S = \emptyset$
for $i \leftarrow 1$ **to** $\lfloor \frac{k}{2} \rfloor$ **do**
  Find $(u, v) = \operatorname{argmax}_{x,y \in U} d(x, y)$
  Set $S = S \cup \{u, v\}$
  Delete all edges from E that are incident to u or v
end
If k is odd, add an arbitrary document to S

where $d'(u, v) = 2\lambda d(u, v) - d(u, q) - d(v, q)$

Top-k Bounded Diversification

- The result is built incrementally.
- Each time the object that maximizes an objective function is added.
- Areas around specific points are probed by enacting incremental nearest neighbor queries.
- The most promising probing locations are points as far as possible from the elements of S.
- The Voronoi diagram of the points in S is constructed at each iteration.
- Search is focused around the edges of the diagram and especially the points where many edges meet.

K-Nearest Diverse Neighbors -i-

Problem definition

Given a point query q, a desired result cardinality of K, and a *MinDiv* threshold, the goal of the K-Nearest Diverse Neighbor (K-NDN) problem is to find the set of K mutually diverse tuples in the database, whose score is the maximum, after including the nearest tuple to q in the result set.

K-Nearest Diverse Neighbors -ii-

1. Traverse stored objects in a nearest neighbor fashion and prune the ones within *MinDiv* distance from any element of the result.
2. A more sophisticated solution can also be found which relies on the same principle (MOTLEY algorithm - buffered greedy).
3. On the downside, not any ranking function can be supported, just a diversity threshold is satisfied.

Outline

Conclusions

1. Our scheme does not have to produce a larger number of recommendations out of which the final K will be selected.
2. And it neither reorders a larger result of $L >> K$ items to present the first K most "diversified" objects.
3. Other works rely on exhaustive search to produce a diversified result-set, an approach clearly inept for disk-resident data.
4. Our paradigm empowers diversified search with just a single access to the disk pages that may contain objects that can improve the current candidate solution.
5. We also suggest an effective policy for excluding from search a large portion of the key-space that cannot contribute to the solution set.
6. Our paradigm requires reduced resources, and thus, enhances performance and scalability.

Distributed Diversified Search

- We have also applied our scheme to a distributed indexing scheme which resembles a distributed k-d tree to receive similar results.
- A peer preserves a link to a peer on the other side of each of the split-points on its path to the root.
- A query is forwarded to the links that represent promising areas.
- The score of a link is computed over the area defined by the respective split-points.
- Score bounds are not as strict as we would like, as the areas defined by the split-points are larger than the peers' areas of responsibility.

Questions?