Context-Aware Top-k Processing using Views

Silviu Maniu, Bogdan Cautis

▶ **To cite this version:**

Silviu Maniu, Bogdan Cautis. Context-Aware Top-k Processing using Views. ACM Conference on Information And Knowledge Management (CIKM), Oct 2013, San Francisco, United States. hal-00927307

HAL Id: hal-00927307
https://inria.hal.science/hal-00927307

Submitted on 13 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Context-Aware Top-k Processing using Views *

Silviu Maniu
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong
smaniu@cs.hku.hk

Bogdan Cautis
Université Paris-Sud &
INRIA Saclay
91405 Orsay Cedex, France
bogdan.cautis@u-psud.fr

ABSTRACT

Search applications where queries are dependent on their context are becoming increasingly relevant in today's online applications. For example, the context may be the location of the user in *location-aware search* or the social network of the query initiator in *social-aware search*. Processing such queries efficiently is inherently difficult, and requires techniques that go beyond the existing, context-agnostic ones. A promising direction for efficient, online answering – especially in the case of top-k queries – is to materialize and exploit previous query results (views).

We consider context-aware query optimization based on views, focusing on two important sub-problems. First, handling the possible differences in context between the various views and an input query leads to view results having uncertain scores, i.e., score ranges valid for the new context. As a consequence, current top-k algorithms are no longer directly applicable and need to be adapted to handle such uncertainty in object scores. Second, adapted view selection techniques are needed, which can leverage both the descriptions of queries and statistics over their results. We present algorithms that address these two problems, and illustrate their practical use in two important application scenarios: location-aware search and social-aware search. We validate our approaches via extensive experiments, using both synthetic and real-world datasets.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process*

Keywords

top-k processing; context-aware applications; social search; spatial search

1. INTRODUCTION

Retrieving the k best data objects for a given query, under a certain score model, is one of the most common problems in databases and on Web. In many applications, and in particular in current Web search engines, tens of thousands of queries per second need to be answered over massive amounts of data. Significant research effort has been put into addressing the performance of top-k processing, towards optimal algorithms – such as TA (Threshold Algorithm) and NRA (No Random Access algorithm) [7, 10] – or highly-efficient data structures [21] (e.g., inverted lists). In recent research, the use of *pre-computed results* (also called *views*) was identified as a promising avenue for improving efficiency [11, 6].

At the same time, with the advent of location-aware devices, geo-tagging, or online social applications, as a way to improve the result quality and the user experience, new kinds of top-k search applications are emerging, which can be simply described as *context-aware*. The context of a query may represent the *geo-location* where the query was issued or the *social identity* – within a social network – of the user who issued it. Generally, it could represent certain score parameters that can be defined or personalized at query time. For example, a query for *nice vegetarian restaurants* should not give the same results if issued in *Miami* or in *Boston*, as it should not give the same results if issued within a social community of culinary reviewers or within a student community.

Unsurprisingly, taking into account the query context in top-k processing is a new source of complexity, and most of the common approaches employed in context-agnostic scenarios must be revisited [5, 16]. Now, query processing usually entails an exploration of a "neighborhood" space for the closest or most relevant objects, which is often interleaved with some of the classic, context independent top-k processing steps, such as scans over inverted lists.

Consequently, materializing and exploiting in searches the results of previous queries can be even more important for efficient, online processing of queries with context. But, in this direction, a broader view-based answering problem than in the context-agnostic setting needs to be addressed, in which the cached results are modeled as unranked lists of objects having only *uncertain scores* or *score ranges*. The rationale is that, even when the cached results in views do have exact scores with respect to one context, we should expect these to evolve into score ranges if a *context transposition* is necessary. For example, answers to the previous query, for a *Boston* context, may be useful – but only to a certain extent – when the same query is issued in a nearby *Cambridge* context, as one has to adapt the scores of restaurants from the former perspective to the latter one; this, inherently, introduces uncertainty.

The potential impact of view-based algorithms coping with such uncertainty is highly relevant but not limited to context-aware settings. Even when queries are not parameterized by contexts, some of the most efficient algorithms (NRA or TA$_Z$ [7]) support early-termination, outputting unranked results with only score ranges.

*Work done while the authors were affiliated with Institut Mines-Télécom - Télécom ParisTech.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505759.

Motivation 1: Location-aware search. Let us consider the spatial-search scenario in Figure 1 (left), in which we have objects at various locations in an euclidian space (objects $o1, \ldots, o5$ in the figure, as gray dots). Each object (e.g., a Web document) is characterized by a bag of attributes. For instance, $o5$ has attributes $t1$ and $t2$, both with a single occurrence.

Now, users located at various points ask for top-k objects w.r.t. a set of attributes. In response, objects are ranked by a combination between the distance of the object w.r.t. the seeker's location and the object's content. While the details of the spatial ranking model will be clarified in Sec. 7, let us assume here that the location relevance of an object contributes 30% to the object's score. The remaining 70% is the weight of its textual score (e.g., tf-idf).

Consider a new query Q in the system, asking for the top-2 items for attributes $\{t1, t2\}$ at the point marked by a white dot in the figure. Intuitively, spatial search algorithms [5], by using indices such as the R-tree [8], would proceed by incrementally increasing the search distance until enough objects are found. However, an alternative execution plan may be possible, if we assume access to cached results of previous queries (initiated at the black dots).

For example, let us assume that $v1$ gives the top-3 documents for $\{t1, t2\}$, as the ranked list $\{o5 = 1.062, o4 = 1.059, o2 = 1\}$. Also, sharing the same location, we have $v2$ and $v3$. The former gives the top-4 for $\{t1\}$ as $\{o2 = 0.946, o3 = 0.575, o5 = 0.500, o4 = 0.262\}$. The latter gives the top-4 for $\{t2\}$ as $\{o4 = 0.962, o5 = 0.500, o1 = 0.437, o2 = 0.246\}$.

Since $v1$, $v2$ and $v3$ are closer to Q than any of the objects, it would be tempting to use their lists of pre-computed results, instead of looking for the actual objects.

In particular, one may resort to using only $v1$'s results, as it is closest to Q both spatially and textually. For that, we need first to perform a change of context, to account for the fact that objects that were close to $v1$ may be even closer to Q, as they may be farther. This will introduce uncertainty in the scores of $v1$'s result: knowing that the normalized distance between Q and $v1$ is 0.175, for Q's perspective, $v1$'s list should now have objects with *score intervals*, as follows: $\{o5 \in [1.062 - 0.3 \times 0.35, 1.062 + 0.3 \times 0.35], o4 \in [1.059 - 0.3 \times 0.35, 1.059 + 0.3 \times 0.35], o2 \in [1 - 0.3 \times 0.35, 1 + 0.3 \times 0.35]\}$ = $\{o5 \in [0.957, 1.167], o4 \in [0.954, 1.164], o2 \in [0.895, 1.105]\}$. ($0.35$ is obtained as $0.175 + 0.175$, since Q has two attributes (terms); 0.3 represents the 30% score weight.)

We can see that $v1$'s result is not sufficient to answer Q with certainty, since any object among the three candidates may be in the top-2. Yet further refinements are enabled by $v2$ and $v3$, albeit more distant, if we corroborate their results with the ones of $v1$. Knowing that $v2$ and $v3$ are at a normalized distance of 0.25 w.r.t. Q, the transposed scores would be, for $v2$: $\{o2 \in [0.871, 1], o3 \in [0.5, 0.65], o5 \in [0.475, 0.525], o4 \in [0.187, 0.337]\}$, and for $v3$: $\{o4 \in [0.887, 1.037], o5 \in [0.475, 0.525],\ o1 \in [0.362, 0.512],\ o2 \in [0.171, 0.321]\}$. Now, we can conclude, for example, that $o4$ is for sure in Q's top-2 result. This is due to the fact that $o4$ cannot be overpassed by $o5$: on one hand, an upper-bound on $o5$'s score can be inferred as $\min(0.525 + 0.525, 1.167) = 1.050$; on the other hand, a lower bound on $o4$ score can be inferred as $\max(0.187 + 0.887, 0.954) = 1.074$.

Motivation 2: Social-aware search. As a second motivating example, we consider the setting of collaborative tagging applications (as Flickr or Del.icio.us). In these applications, users tag (or bookmark) objects from a common pool of objects (e.g., Web sites). Users form a social network, in which relationships are weighted (e.g., a similarity or proximity value). Such a setting is illustrated in Figure 1, right. For example, user $u1$ has tagged object $o1$ with t_1

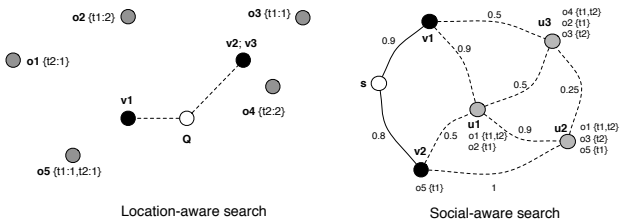

Figure 1: Context-aware search scenarios.

and t_2, and $o2$ with t_1; he is 0.9-close (or similar) to u_2, and he is 0.5 close to v_2. User v_2 tagged object $o5$ with t_1.

We use the social ranking model introduced by Amer-Yahia et al. [1] and extended by Schenkel et al. [16]. Intuitively, with this model, an object's score (e.g., $o1$'s) for a given tag (e.g., t_1) is proportional to the sum of the proximities of the taggers (i.e., u_1 and u_2) w.r.t. the seeker. An object's score for a set of tags is then computed by aggregating the per-tag scores, e.g., by summation.

Let us now assume that the top-2 items for $\{t1, t2\}$ are requested by user s (the seeker). As in location-aware search, early termination algorithms [16, 13, 14] for social search would incrementally explore the most promising users (and their objects) until the top-k is found. This may lead to the visit of a non-negligible fraction of the network. For our query, an exploration of the network would need to go as far as $u2$ to establish a top-2 as $\{o1, o5\}$.

Yet an alternative, more efficient processing approach may rely on pre-computed results. Let us assume that users $v1$, $v2$ have such data: $v2$ has a top-3 for $\{t1\}$, as $\{o5 = 2, o1 = 1.9, o2 = 1.35\}$, and for $\{t2\}$, as $\{o1 = 1.9, o3 = 1.45, o4 = 0.45\}$. $v1$ has a top-4 for $\{t1, t2\}$ as $\{o1 = 3.42, o5 = 1.53, o2 = 1.4, o3 = 1.31\}$.

Knowing that the distance between s and $v1$ is 0.9, in similar way to the spatial-aware scenario, a transposition of context from $v1$ to s (the formal ranking model will be described in Section 7) leads to the following result for $\{t1, t2\}$: $\{o1 \in [3.07, 3.8], o5 \in [1.37, 1.7], o2 \in [1.26, 1.55], o3 \in [1.17, 1.45]\}$. Similarly, knowing the distance between $v2$ and s is 0.8, context transposition leads to the following result for $\{t1\}$: $\{o5 \in [1.6, 2.5], o1 \in [1.52, 2.37], o2 \in [1.08, 1.68]\}$, and the following result for $\{t2\}$: $\{o1 \in [1.52, 2.37], o3 \in [1.16, 1.81], o4 \in [0.36, 0.56]\}$.

As in the previous scenario, by using only the closest view (from $v1$) we cannot obtain the top-2 required by s, because: (i) either of $o2$, $o3$ and $o5$ might enter the top-2 alongside $o1$, and (ii) the objects which are not included in the view might have a score as high as 1.45, meaning they might also be in the top-2. Yet, using also the information in the views of user $v2$ enables us to establish the exact top-2 as $\{o1 \in [3.07, 3.8], o5 \in [1.6, 1.7]\}$, after visiting just two neighbors of s, $v1$ and $v2$.

Our contributions. We formalize and study in this paper the problem of context-aware top-k processing based on possibly uncertain precomputed results, in the form of *views* over the data.

Most informative answer. Answering such top-k queries using *only* the information in views, inevitably, requires an adaptation to the fact that the views may now offer objects having only uncertain scores. So there may be view instances from which an exact top-k cannot be extracted with full confidence. When this is the case, we aim to give a *most informative answer*, in terms of (i) objects G that are guaranteed to be in the top-k result, and (ii) objects P that may appear in the top-k result.

TA adaptation. We formalize this query semantics and describe an adaptation of TA, denoted SR-TA. It handles precomputed lists with *score ranges*, and is *sound* and *complete*, i.e., outputting the

(G, P)-answer. Intuitively, SR-TA implements the illustrated corroboration principle, based on a *linear programming* formulation.

View selection. As in many applications the set of views may be large, we also consider optimizations for SR-TA, based on *selecting some (few) most promising views*. Obviously, with fewer views, the most informative answer (G, P) may no longer be reached; we are in general presented a trade-off between the number of selected views – determining the cost of the top-k algorithm SR-TA – and the result's "quality" (a distance with respect to the most informative answer given by all the views).

Applications – context transposition. Importantly, our algorithms provide a *one-size-fits-all solution* for many applications that are context-dependent, and we show how they can be directly applied in our two motivating application scenarios for context-aware search, social-aware search and location-aware search.

Extensive experiments on synthetic and real-world datasets show the potential of our techniques – enabling high-precision retrieval and important running-time savings. More generally, they illustrate the potential of top-k query optimization based on cached results in a wide range of applications.

Outline. We discuss the main related work in Section 2. We formalize the context-aware search setting and problems in Section 3. We give the SR-TA adaptation of TA, in Section 4. Our optimization approach by view selection is formulated in Section 5. We study the formal properties of SR-TA, and we give our experimental evaluation in Section 8.

2. RELATED WORK

Main research landscape. In addition to the classic TA/NRA [7] algorithms, other techniques for top-k answering using views have been proposed in recent literature: the LPTA algorithm [6] and generalizations of NRA and TA [11], both applicable in settings where aggregation functions are linear combinations of the per-attribute scores. Regarding view selection, the work closest in spirit to ours is [6], whose focus is on finding the optimal top-k execution based on a selection of precomputed views, with all the per-attribute lists being assumed to be part of the view space. Their approach simulates the run of a threshold algorithm over histograms of views. The setting of [6] is fundamentally different from ours. First, any viable selection of views must output the exact, ranked top-k result, which represents a strong limitation for practical purposes. Hence, while their focus is on optimizing the top-k computation, we deal with a different perspective over the view selection problem, towards minimizing the uncertainty of the result.

[17] computes top-k answers on uncertain data – ranked object lists with score ranges and probability-density functions – through a probabilistic ranking model based on partial orders; they do not deal with aggregation of uncertain scores over multiple dimensions.

Other related work. The common data structure for top-k processing is the inverted index file (see [21]'s survey on indexing for top-k processing), over which a key challenge is to optimize response time [19, 20]. Regarding algorithms, among the most cited and used are the early-termination threshold algorithms TA and NRA of [7], which are instance optimal. Many other top-k aggregation algorithms were proposed in recent literature, and we refer the interested reader to the survey [10] and the references therein. The use of precomputed results, either as previous answers to queries [6, 9] or as cached intersection lists [11], has been identified as an important direction for efficiency. A linear programming formulation over score information is first introduced in [6] and extended in [11].

In [17], the authors study top-k processing when only score ranges are known, instead of exact ones, define a probabilistic ranking model based on partial orders and introduce several semantics for ranking queries, but do not deal with aggregation of uncertain scores over multiple dimensions. Another general formulation of ranking in probabilistic databases is presented in [12]. In the area of location-aware retrieval, Cong et al. [5] introduce the concept of LkT queries, for which they include in the ranking model both the distance of a document's location w.r.t. the query point and the textual features of the document. They propose the IR-tree index, consisting of an R-tree [8] in which each node has an inverted list of relevant documents. Other models for top-k location-aware keyword querying have been proposed, for selecting either groups of objects that collectively satisfy a query [3], or the k-best objects scored by the features in their neighborhood [15], or the top-k objects in a given query rectangle [4]. Various approaches for combining textual inverted lists and spatial indexes for keyword retrieval were studied in [4]. In the area of social search, for which bookmarking applications are a popular abstraction, top-k processing using the social network as an integral part of the ranking model has been considered in recent research. [1] is the first to consider this problem, yet under significant restrictions, taking into account only a subset of users and their documents in answers. The CONTEXTMERGE algorithm [16] is the first to address the social-aware search without imposing limitations on the exploration space, and they use the ranking model that we adopted in this paper.

3. FORMAL SETTING AND PROBLEMS

Context-aware score model. We assume a finite collection of objects $\mathcal{O}$ and a countable collection of attributes $\mathcal{T}$. Under a given *context parameter* $\mathcal{C}$ – an abstract notion whose instantiation depends on the application – objects o are associated to certain attributes t, by an object-attribute score function $sc(o, t \mid \mathcal{C})$.

Under a context $\mathcal{C}$, a query Q consists of a set $\{t_1, \ldots, t_n\}$ of attributes; its answer is given by objects $o \in \mathcal{O}$ having the highest scores $sc(o, Q \mid \mathcal{C})$, computed via a monotone aggregation function h (e.g., `sum`, `max`, `avg`) over the object-attribute scores:

$$sc(o, Q \mid \mathcal{C}) = h(sc(o, t_1 \mid \mathcal{C}), \ldots, sc(o, t_n \mid \mathcal{C})).$$

We can formalize the top-k retrieval problem as follows:

PROBLEM 1. *Given a query $Q = \{t_1, \ldots, t_n\} \subset \mathcal{T}$, a context $\mathcal{C}$, an integer k, and a score model specification (sc, h), retrieve the k objects $o \in \mathcal{O}$ having the highest scores $sc(o, Q \mid \mathcal{C})$.*

In certain applications, the context may always be empty or may simply be ignored in the sc scores, and, when necessary, we indicate this in our notation by the '$\perp$' context. We use $sc(o, Q)$ as short notation for $sc(o, Q \mid \perp)$.

We revisit in this paper the class of early termination top-k algorithms known as *threshold algorithms*. These algorithms, applicable in a context-agnostic setting, find the top-k objects for an input query Q by scanning sequentially (for each attribute) and in parallel (for the attribute set of Q), relevant per-attribute lists that are ordered descending by sc values – with inverted lists being a notable example – denoted in the following $L(t)$, as the list for attribute t. During a run, they maintain a set D of already encountered candidate objects o, bookkeeping for each candidate (i) an upper-bound on $sc(o, Q)$ – the best possible score that may still be obtained for o – denoted hereafter $bsc(o, Q)$, and (ii) a lower-bound on $sc(o, Q)$ – the worst possible score – denoted hereafter $wsc(o, Q)$. The objects are ordered in D by their worst scores. hereafter $ws(o, Q)$.

At each iteration, or at certain intervals, threshold algorithms may refine these bounds and compare the worst score of the kth object in D, $wsc(D[k], Q)$, with the best possible score of either (i) objects o in D outside the top k, $bsc(o, Q)$, or (ii) not yet encountered objects, denoted $bsc(*, Q)$. When both these best scores are not greater

Table 1: Example of transposed location-aware views $\mathcal{V}$.

$v1(\{t1,t2\},\perp)$			$v2(\{t1\},\perp)$			$v3(\{t2\},\perp)$		
o	**wsc**	**bsc**	**o**	**wsc**	**bsc**	**o**	**wsc**	**bsc**
o5	0.957	1.167	o2	0.871	1.000	o4	0.887	1.037
o4	0.954	1.164	o3	0.500	0.650	o5	0.475	0.525
o2	0.895	1.105	o5	0.475	0.525	o1	0.362	0.512
			o4	0.187	0.337	o2	0.171	0.321
$*$		1.105	$*$		0.337	$*$		0.321

than the worst score of $D[k]$, the run can terminate, outputting $D[1], \ldots, D[k]$ as the final top-k.

Views and precomputed results. We extend the classic top-k retrieval setting of TA/NRA by assuming access to precomputed query results, called in the following *views*. Each view V is assumed to have two components: (i) a *definition*, $def(V)$, which is a pair query-context $def(V) = (Q^V, \mathcal{C}^V)$ and (ii) a set $ans(V)$ of triples (o_i, wsc_i, bsc_i), representing the *answer* to query Q^V under context $\mathcal{C}^V$. Each such triple says that object o_i has a score $sc(o_i, Q^V \mid \mathcal{C}^V)$ within the range $[wsc_i, bsc_i]$.

Since we are dealing with cached query results, all objects not appearing in $ans(V)$ – represented explicitly in $ans(V)$, to simplify presentation, by one final *wildcard* $*$ *object* – have with respect to query Q^V and context $\mathcal{C}^V$ a worst score of $wsc_* = 0$ and a best possible score of either $bsc_* = \min\{bsc_i \mid (o_i, wsc_i, bsc_i) \in ans(V)\}$, if V's result is complete, in the sense that enough objects had a non-zero score w.r.t. Q^V, or otherwise 0.

Context transposition. Intuitively, when a view V and the to-be-answered query Q do not have the same context, a transposition of the exact scores or score ranges in $ans(V)$ is needed, to obtain valid ranges for $sc(o_i, Q^V|\mathcal{C})$ from those for $sc(o_i, Q^V|\mathcal{C}^V)$. In particular, in the case of spatial or social search, this transformation will inevitably yield a coarser score range. We detail the specific transposition operation for these application scenarios in Sec. 7.

Exploiting views. Given an input query Q and a context $\mathcal{C}$, from a set of views $\mathcal{V}$ sharing the same context – as in $def(V) = (\ldots, \mathcal{C})$ – a first opportunity that is raised by the ability to cache results is to compute for objects $o \in \mathcal{O}$ tighter lower and upper bounds over $sc(o, Q \mid \mathcal{C}))$. This may be useful in threshold algorithms, as a way to refine score ranges. We formalize this task next.

PROBLEM 2. *Given a query $Q = \{t_1, \ldots, t_n\} \subset \mathcal{T}$, a context $\mathcal{C}$, an integer k, a score model specification (sc, h) and a set of views $\mathcal{V}$ sharing the same context with Q, given an object $o \in \mathcal{O}$, compute the tightest lower and upper bounds on $sc(o, Q \mid \mathcal{C})$ from the information in $\mathcal{V}$.*

In this paper, consistent with the most common ranking models and related work on view-based top-k answering [6, 11], we assume that the aggregation h is summation. Consequently, Problem 2 can be modeled by the following linear program (LP), whose variables are given in bold, knowing that $\mathbf{sc}(\mathbf{o}, \mathbf{t_l} \mid \mathcal{C}) \geq 0, \forall t_l \in \mathcal{T}$:

$$\min \sum_{t_i \in Q} \mathbf{sc}(\mathbf{o}, \mathbf{t_i} \mid \mathcal{C}) \quad (3.1) \qquad \max \sum_{t_i \in Q} \mathbf{sc}(\mathbf{o}, \mathbf{t_i} \mid \mathcal{C}) \quad (3.2)$$

$$wsc \leq \sum_{t_j \in Q^V} \mathbf{sc}(\mathbf{o}, \mathbf{t_j} \mid \mathcal{C}) \leq bsc, \ \forall V \in \mathcal{V} \ s.t. \ (o, wsc, bsc) \in ans(V)$$

EXAMPLE 1. *Let us consider the views in Table 1, for the location-aware search scenario discussed previously, after context transposition. We have access to the transposed results of the three views, defined by the sets of attributes $\{t1\}$, $\{t2\}$, and $\{t1, t2\}$. The context is now the same for all views and the query $Q = \{t1, t2\}$. Considering o4, for example, we know that:*

$$\begin{aligned} 0.954 \leq & sc(o4, \{t1\}) & + sc(o4, \{t2\}) & \leq 1.164 \ (by \ v1) \\ 0.187 \leq & sc(o4, \{t1\}) & & \leq 0.337 \ (by \ v2) \\ 0.887 \leq & & sc(o4, \{t2\}) & \leq 1.037 \ (by \ v3) \end{aligned}$$

Then, the lower bound on $sc(o4, Q)$ is obtained as

$$wsc(o4, Q) = \max(sc(o4, \{t1\}) + sc(o4, \{t2\})) = 1.074$$

by combining the worst scores in $v2$ and $v3$. Similarly, combining the best scores of V_2, V_3, $sc(o4, Q)$'s upper-bound is

$$bsc(o4, Q) = \min(sc(o4, \{t1\}) + sc(o4, \{t2\})) = 1.164.$$

We now formulate the problem of answering input top-k queries Q using *only* the information in views, whose semantics needs to be adapted to the fact that views may offer only a partial image of the data. When an exact top-k cannot be extracted with full confidence, a *most informative result* would consist of two disjunctive, possibly-empty sets of objects from those appearing in $\mathcal{V}$'s answer: (i) a set of all the objects guaranteed to be in Q's top-k, and (ii) a set of all objects that may also be in Q's top-k.

Problem 2 gives a way to properly define and identify objects of the former kind – the *guaranteed ones* – as the objects o_x for which

$$\min \sum_{t_i \in Q} sc(o_x, t_i \mid \mathcal{C}) \geq \max \sum_{t_i \in Q} sc(*, t_i \mid \mathcal{C}) \quad (3.3)$$

and at most $k-1$ objects o_y can be found such that

$$\min \sum_{t_i \in Q} sc(o_x, t_i \mid \mathcal{C}) < \max \sum_{t_i \in Q} sc(o_y, t_i \mid \mathcal{C}). \quad (3.4)$$

Similarly, we can identify objects of the latter kind – the *possible ones* – as the objects o_x that are not guaranteed and for which at most $k-1$ objects o_y can be found s.t.

$$\min \sum_{t_i \in Q} sc(o_y, t_i \mid \mathcal{C}) > \max \sum_{t_i \in Q} sc(o_x, t_i \mid \mathcal{C}). \quad (3.5)$$

We now formalize the *top-k retrieval problem using views*.

PROBLEM 3. *Given a query Q=$\{t_1, \ldots, t_n\} \subset \mathcal{T}$, a context $\mathcal{C}$, an integer k, and a score model specification (sc, h), given a set of views $\mathcal{V}$ having the same context as Q, retrieve from $\mathcal{V}$ a* most informative answer *of the form (G, P), with*

- *$G \subset \mathcal{O}$ consisting of all* guaranteed objects *(as in Eq. (3.3) and (3.4), when h is summation); they must be among those with the k highest scores for Q and $\mathcal{C}$.*
- *and $P \subset \mathcal{O}$ consisting of all* possible objects *outside G (as in Eq. (3.5), when h is summation); they may be among those with the k highest scores for Q and $\mathcal{C}$, i.e., there exist data instances where these appear in the top-k.*

In order to solve Problem 3, a naïve computation of upper and lower bounds for all objects o appearing in the views would suffice, but would undoubtedly be too costly in practice. Instead, we show in Section 4 how we can solve Problem 3 in the style of threshold algorithms, by extending TA.

Over any data instance, the exact top-k can be seen as the set G plus the top-k' items from P, for $k' = k - |G|$. To give a *most likely result*, in a probabilistic sense, based on the object sets G and P, we discuss in Sec. 4 possible approaches for estimating the probability of possible top-k' sets from P.

Going further, even when the most promising candidate objects are considered first in SR-TA, their corresponding instances of the linear programs in Eq. (3.1) and Eq. (3.2) may still be too expensive to compute in practice (even when we are dealing with LPs, as in Example 1): the set of views may be too large – of the order $2^{|\mathcal{T}|}$ – and each view contributes one constraint in the program. In our

best-effort approach, which would first select some *most promising views* $\tilde{\mathcal{V}} \subset \mathcal{V}$ for the input query (Section 5), we face a trade-off between the size of the subset $\tilde{\mathcal{V}}$ – which determines the cost of SR-TA – and the "quality" of the result, namely its distance with respect to the most informative answer given by all the views. We quantify the distance between the most informative result by $\tilde{\mathcal{V}}$, denoted $(\tilde{G}, \tilde{P})$, and the most informative answer (G, P) by $\mathcal{V}$ as the difference in the number of possible top-k combinations:

$$\Delta = \binom{|\tilde{P}|}{k - |\tilde{G}|} - \binom{|P|}{k - |G|}. \quad (3.6)$$

We also show in Section 5 how a final refinement step over $(\tilde{G}, \tilde{P})$, based on random accesses in the entire $\mathcal{V}$ set, allows us to reach $\Delta = 0$, i.e., the most informative result by $\mathcal{V}$.

4. THRESHOLD ALGORITHM

We present in this section an adaptation of TA, called SR-TA. It can be applied when the input lists consist of objects with score ranges; SR-TA allows us to solve Problem 3.

Each of the input lists are assumed to be available in two copies, ordered descending by (i) the score lower-bound, and (ii) the score upper-bound. SR-TA will read sequentially in round-robin manner from the former group of lists and, similar to TA, maintains a candidate set D of the objects encountered during the run. At each moment, the read heads of the latter group of lists must give objects that are not yet in D (unseen objects), and sequential accesses are done in SR-TA whenever necessary to maintain this configuration.

D is also ordered descending by score lower-bounds. SR-TA stops when the score of any unseen object – threshold τ – cannot be greater than the one of the kth object in D.

In our setting, τ is the solution to the mathematical program (MP) below, under $\mathbf{sc}(\mathbf{o}, \mathbf{t_l} \mid \mathcal{C}) \geq 0, \forall t_l \in \mathcal{T}$, taking into account from each view V the score upper-bound of objects in $ans(V) - D$:

$$\tau = \max \sum_{t_i \in Q} \mathbf{sc}(\mathbf{o}, \mathbf{t_i} \mid \mathcal{C}) \quad (4.1)$$

$$\sum_{t_j \in Q^V} \mathbf{sc}(\mathbf{o}, \mathbf{t_j} | \mathcal{C}) \leq max(bsc_i), \forall V \in \mathcal{V}, o_i \notin D$$

One can note that when (i) we have only views that give answers to singleton queries, and (ii) $wsc_i = bsc_i$ for each object o_i (i.e., the lists contain exact scores), we are in the setting of the TA family of algorithms over inverted list inputs. Relaxing condition (i), we have the setting of top-k answering using views investigated in [11, 6]. Both these settings and their corresponding algorithms can guarantee that, at termination, the exact top-k is returned. Our more general setting, however, cannot provide such guarantees, as witnessed by the following example.

EXAMPLE 2. *Let us revisit Example 1, for the top-2 query $Q = \{t1, t2\}$. We will not detail the complete run of the algorithm on this example, instead showing what happens at termination. The algorithm stops at the 5th iteration, at depth 2 in the input lists. The threshold value $\tau = 0.849$ is obtained by combining the best score in $v3$ of the not-yet-candidate (not in D) item $o1$ with the score of the wildcard object $*$ in $v2$. The worst score of the 2nd item, $o4$, is 1.042, hence larger than the threshold, enabling termination. This ensures that all the possible candidates for top-k are already present in the list D, as shown below:*

obj	o4	o2	o5	o3	τ
wsc	1.074	1.042	0.957	0.500	
bsc	1.164	1.105	1.050	0.871	0.849

Yet, within this candidate list, there is no combination of 2 objects that represents the top-k. Instead, we can only divide D into 3 sets:

1. *a set $G = \{o4\}$ of guaranteed result objects,*
2. *a set $P = \{o2, o5\}$ of possible result objects,*
3. *a set of the remaining object: $\{o3\}$.*

Algorithm 1 details SR-TA. Its general flow is similar to the one of TA, with the notable addition of the generalized computation of bounds and of the threshold value.

Algorithm 1: SR-TA$(Q, k, \mathcal{V})$

```
Require: query Q, size k, views V
 1: D = ∅
 2: loop
 3:    for each view V ∈ V in turn do
 4:       (o_i, wsc_i, bsc_i) ← next tuple by sequential access in V
 5:       read by random-accesses all other lists V' ∈ V for tuples
          (o_j, wsc_j, bsc_j) s.t. o_i = o_j
 6:       wsc ← solution to the MP in Eq. (3.1) for o_i
 7:       bsc ← solution to the MP in Eq. (3.2) for o_i
 8:       add the tuple (o_i, wsc, bsc) to D
 9:    end for
10:    τ ← the solution to the MP in Eq. (4.1)
11:    wsc_t ← lower-bound score of kth candidate in D
12:    if τ ≤ wsc_t then break
13: end loop
14: {G, P}=PARTITION(D, k)
15: return G, P
```

Partition for most informative result. Once SR-TA's main loop terminates, candidates D are passed as input to a sub-routine whose role is to partition it into sets G and P (line 14 in SR-TA). Algorithm 2 details this step: for each object o in D we test the conditions of Eq. (3.3), (3.4), (3.5).

Algorithm 2: PARTITION(D, k)

```
Require: candidate list D, parameter k
 1: G ← ∅ the objects guaranteed to be in the top-k
 2: P ← ∅ the objects that might enter the top-k
 3: for each tuple (o, bsc, wsc) ∈ D, o ≠ * do
 4:    x ← |{(o', bsc', wsc') ∈ D | o' ≠ o, bsc' > wsc}|
 5:    wsc_t ← lower-bound score of kth candidate in D
 6:    if x ≤ k and for (*, wsc_*, bsc_*) ∈ D, bsc_* ≤ wsc then
 7:       add o to G
 8:    else if bsc > wsc_t then add o to P
 9:    end if
10: end for
11: return G, P
```

At the termination of SR-TA, we are guaranteed that G and P are *sound and complete*, in the following sense:

PROPERTY 1. *An object o is in the output set G of* PARTITION(D, k) *iff, in all possible data instances, o is the top-k for Q and $\mathcal{C}$.*

An object o is in the output set P of PARTITION(D, k) *iff, in at least one possible data instance, o is in the top-k for Q and $\mathcal{C}$.*

Note G's size is at most k, while P's size is at most $|\mathcal{O}|$, hence the need for completeness (maximal G, minimal P).

Extracting a probable top-k' in P. As discussed previously, the actual (inaccessible) top-k answer for the input query can also be seen as being composed of two parts: the guaranteed objects G plus a top-k' over P, for $k' = k - |G|$. By definition, G and P represent the most informative certain result obtainable from the views: there can be no deterministic way to compute a certain top-k' over the P objects, nor a way to further prune the search space towards a more refined P set.

Therefore, we can only hope to improve the quality of the result by a more detailed *probabilistic* description of it, in which a most

likely top-k could be identified from G and P. Since for each object in P we have a lower and upper bound on its exact score, let us assume a known probability-density function (e.g, uniform one) for scores within the known bounds. Based on this, we can reason about the likelihood of a top-k' selection over P.

A naïve way to obtain the most likely top-k' would be the following: enumerate all subsets of P of size k', and compute for each the probability of being the top-k'. Each of these $\binom{|P|}{k'}$ probability values can be easily obtained once we have for each pair of objects $o_1, o_2 \in P$ the probability $Pr(o_1 > o_2)$.

Much more efficient than the naïve enumeration is to adapt to our setting the sampling-based approach of [17] (for top-k answers on object lists with score ranges having probability-density functions), and we will use this approach in our implementation of the sampling algorithms.

5. VIEW SELECTION

We consider now the view selection problem, which may improve the performance of our threshold algorithm SR-TA, possibly at the risk of yielding results that are less accurate. To address this issue, we discuss at the end of this section how results obtained through view selection can be refined to the most informative one. *Throughout this section, we remain in the setting where the query and views are assumed to have the same context.*

We argue first that view selection comes as a natural perspective in the computation of score bounds. Recall that, for a given object $o \in O$, Problem 2 can be modeled by the linear programs (3.1) and (3.2). Put otherwise, we have as dual of the minimization problem (3.1) the following packing LP:

$$\max \sum_{\substack{i=1,\\ (o,wsc,..)\in ans(V_i)}}^{|\mathcal{V}|} wsc \times l_i,$$

$$\sum_{t\in Q^{V_j}} l_j \leq 1, \forall t \in Q, \sum_{t\in Q^{V_j}} l_j = 0, \forall t \notin Q \qquad (5.1)$$

and we have as the dual the maximization problem (3.2) the following covering LP:

$$\min \sum_{\substack{i=1,\\ (o,..,bsc)\in ans(V_i)}}^{|\mathcal{V}|} bsc \times u_i,$$

$$\sum_{t\in Q^{V_j}} u_j \geq 1, \forall t \in Q, \sum_{t\in Q^{V_j}} u_j = 0, \forall t \notin Q \qquad (5.2)$$

Based on the LPs (5.1), (5.2), for each object o, to obtain its most refined bounds, we would need to first *fractionally* select views from $\mathcal{V}$ – as opposed to *integral* selection – s.t. the linear combinations of o's scores with the coefficients u_i and l_i are optimal. In other words, for computing the worst or best score of each object, it would suffice to select and take into account only the views $V_i \in \mathcal{V}$ such that (i) $l_i \neq 0$, for worst scores, or (ii) $u_i \neq 0$, for best scores.

EXAMPLE 3. *Let us consider the views in Table 1, using the LPs (5.1) and (5.2) to illustrate view selection for object o2. For the worst score, we need to optimize*

$\max(0.895l_1 + 0.871l_2 + 0.171l_3),\ s.t. l_1 + l_2 \leq 1,\ l_1 + l_3 \leq 1$

The optimal value is reached when $l1 = 0$ and $l_2 = l_3 = 1$, i.e., relying on the worst scores of o2 from views $v2$ and $v3$.

For the best score, we need to optimize

$\min(1.164l_1 + 0.337l_2 + 1.037l_3)\ s.t. l_1 + l_2 \geq 1,\ l_1 + l_3 \geq 1.$

The optimal value is reached when $l_1 = 1$ and $l_2 = l_3 = 0$, i.e., relying on the best score of o2 from view $v1$.

Solving the LPs (5.1) and (5.2) for each object, as a means to select only the useful views, would obviously be as expensive as solving directly the LPs (3.1) and (3.2). Instead, it would be preferable to solve these LPs and select some most relevant views *independently of any object*, i.e., only once, before the run of the threshold algorithm. Instead of per-object wsc and bsc values, in an approximate version of the two LPs, each view V_i could be represented by two unique values, $wsc(V)$ and $bsc(V)$. Our optimization problems would then simplify as follows:

$$\max \sum_{i=1}^{|\mathcal{V}|} wsc(V_i) \times l_i,\ \sum_{t\in Q^{V_j}} l_j \leq 1, \forall t \in Q,\ \sum_{t\in Q^{V_j}} l_j = 0, \forall t \notin Q$$

$$\min \sum_{i=1}^{|\mathcal{V}|} bsc(V_i) \times u_i,\ \sum_{t\in Q^{V_j}} u_j \geq 1, \forall t \in Q,\ \sum_{t\in Q^{V_j}} u_j = 0, \forall t \notin Q$$

and this would enable us to select the "good" views in the initialization step of the top-k algorithm, those participating to the computation of the optimal, i.e., views having *non-zero* u and l coefficients.

Moreover, for each object o encountered in SR-TA's run, we can now replace Eq. (3.1) and (3.2) (lines 5-6 in SR-TA) by the following estimates using only the selected views $\widetilde{\mathcal{V}}$:

$$\widetilde{wsc} = \sum_{\substack{i=1\\ (o,wsc,..)\in ans(V_i)}}^{|\widetilde{\mathcal{V}}|} wsc \times l_i\ ;\ \widetilde{bsc} = \sum_{\substack{i=1\\ (o,..,bsc)\in ans(V_i)}}^{|\widetilde{\mathcal{V}}'|} bsc \times u_i$$

This is possible since, by the duality property, we are guaranteed that the feasible solutions represent safe bounds for o's scores, i.e., $\widetilde{wsc} \leq wsc$ and $\widetilde{bsc} \geq bsc$. We can similarly simplify Eq. (4.1), for the threshold value (for line 8 in SR-TA).

Candidates for $wsc(V)$ and $bsc(V)$. We follow the described approach – approximating view selection – in two distinct ways. First, per-view score bounds $wsc(V)$ and $bsc(V)$ could be based solely on the view's definition Q^V, and we experimented in this paper with bounds that are defined as $wsc(V) = bsc(V) = |Q^V|$. for each $V \in \mathcal{V}$. The intuition for this choice is that object scores in a view V are proportional to the number of attributes in Q^V. Second, we consider (and experiment with in Section 8) two natural per-view measures that are based on the views' answers: (i) the average score value, and (ii) the maximum score value.

Retrieving (G, P) after view selection. We now discuss how the most informative result (G, P) – obtainable from the complete set of views $\mathcal{V}$ – can still be retrieved by refining a result $(\tilde{G}, \tilde{P})$ obtained on a selection of views $\tilde{\mathcal{V}}$. For that, we need to adopt the following modifications in instances of SR-TA running over a selection of views: when the main loop terminates, compute the optimal bounds for all objects in $\tilde{P}$ by random-accessing their scores in all the views in $\mathcal{V}$, then run for a second time the partition subroutine.

It can be easily shown that, in this way, we obtain the most informative result, i.e., we reach $\Delta = 0$. Therefore, the "bulk" of the work could be done only on a selection of views and its result, potentially few candidate objects, could just be refined at the end using the complete $\mathcal{V}$. We describe in Section 8 the impact of this optimization on the running time of SR-TA.

In summary, we described two sound and complete variants of SR-TA: without view selection (SR-TA^{nosel}), and with view selection (SR-TA^{sel}). For view selection variants, our notation convention will be to replace the sel superscript by a def, max or avg one, depending on the selection method being used.

6. FORMAL RESULTS

Let $\boldsymbol{A}$ be the class of algorithms, including SR-TA, that deterministically output the exact sets P and G, without making "wild guesses".[1] For a given set of views $\mathcal{V}$, we denote by $D(\mathcal{V})$ the class of all instances of answers in those views, i.e., $ans(V), V \in \mathcal{V}$.

Given two algorithms $A_1 \in \boldsymbol{A}$ and $A_2 \in \boldsymbol{A}$, we write $A_1 \preceq A_2$ iff, for all sets of views $\mathcal{V}$, A_2 is guaranteed to cost at least as much as A_1 – in terms of I/O accesses (sequential, random or a linear combination of the two) – over all instances in $D(\mathcal{V})$. Conversely, we write $A_1 \not\preceq A_2$ iff there exists at least one view set $\mathcal{V}$ and an instance in $D(\mathcal{V})$ over which A_2 costs less than A_1. We say that an algorithm $A \in \boldsymbol{A}$ is *instance optimal over* $\boldsymbol{A}$ iff $A \preceq B, \forall B \in \boldsymbol{A}$.

We first consider the question whether one of the two variants of SR-TA is guaranteed to perform better that the other, for all views and answers. The answer to this question is far from obvious: on one hand, SR-TA^{sel} should use fewer views to compute the P and G sets, but it might either go too deep in the selected views or might need additional accesses in other views (see Section 5); on the other hand, SR-TA^{nosel} may go through views that are useless for deriving optimal bounds.

LEMMA 1. SR-TA$^{sel} \not\preceq$ SR-TA$^{nosel} \not\preceq$ SR-TA^{sel}.

Lemma 1 tells us that neither of the two variants of SR-TA can be instance optimal for all possible sets $\mathcal{V}$. However, we describe next a restricted class of views for which: (i) no refinement step is necessary after selecting a subset of the views, and (ii) SR-TA^{sel} becomes instance optimal. Let $\mathbf{V}$ be the class of *sets $\mathcal{V}$ of pairwise disjoint views*, i.e., s.t. $Q^{V_i} \cap Q^{V_j} = \emptyset, \forall V_i, V_j \in \mathcal{V}, V_i \neq V_j$. We say an algorithm $A \in \boldsymbol{A}$ is *instance optimal over* $\boldsymbol{A}$ *and* $\boldsymbol{V}$ if $A \preceq B, \forall B \in \boldsymbol{A}$ and $\forall \mathcal{V} \in \boldsymbol{V}$.

THEOREM 1. SR-TA^{sel} *is instance optimal for* $\boldsymbol{A}$ *and* $\boldsymbol{V}$.

Intuitively, for this class of views, the only way to obtain bounds for a query Q is the following: (i) for lower-bounds, only the views V that have $Q^V \subseteq Q$ are taken into account, while (ii) for upper-bounds all views V that verify $Q^V \cap Q \neq \emptyset$ are used. Note that this method is in effect the view selection algorithm for the class of pairwise disjoint views. Moreover, for this class of views, the $nosel$ variant will use the same set of views as the sel variants. Hence the final refinement step is no longer needed, as there are no other relevant views which can refine the result (P, G). Note also that the classic setting of [7], i.e. per-attribute lists of exact scores, is strictly subsumed by $\mathbf{V}$.

7. CONTEXT TRANSPOSITION

We discussed so far how queries can be answered using pre-computed views, with the important assumption that these share the same context with the input query. We remove now this restriction, considering also views that may have been computed in a different context. We show how we can still answer queries by the techniques discussed so far, by pre-processing views in order to place them in the context of the input query, by what we call *context transposition*.

We give in this section the details on context transpositions for our two motivating application scenarios: location-aware search and social-aware search. In both applications, one view V's context $\mathcal{C}^V$ can be seen as consisting of

1. a *location* (or *start point*) $\mathcal{C}^V.l$, e.g., geo-coordinates in a multidimensional space for location-aware search, or the social identity of a seeker in social-aware search,

[1] These algorithms do not include in their working buffers (e.g., candidate buffer D) items that were not yet encountered in the input lists (they cannot guess that an item might be encountered later).

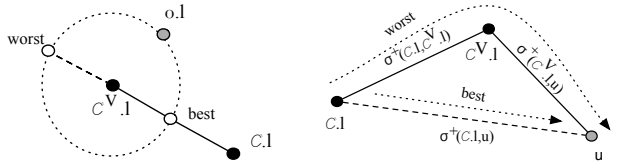

Figure 2: Intuition for start point transpositions.

2. a *contextual parameter* $\mathcal{C}^V.\alpha$, which parameterizes the influence of the spatial or social aspect in scores.

Given an input query Q, a context $\mathcal{C}$ – with $\mathcal{C}.l$ and $\mathcal{C}.\alpha$ – and a view V with a different context (either the location or α may differ, or both), in order to be able to use pre-computed results from V, we need to derive from the existing $ans(V)$ tuples new score bounds: for each $(o, wsc, bsc) \in ans(V)$ we want to obtain a new tuple $(o, f_w(wsc), f_b(bsc))$. The functions f_w and f_b represent the core of the context transposition, their role being to map the worst scores and best scores of objects from $ans(V)$ to new guaranteed bounds for context $\mathcal{C}$. We detail them next for our application scenarios.

7.1 Location-aware search

In location-aware or spatial top-k querying [5], a user having a certain location is interested in the top-k objects that are relevant textually and close spatially.

We revisit here one of the most common ranking models [5, 2], in which the per-attribute score objects are a linear combination of spatial relevance and textual relevance. Each object o consists of a bag of attributes $o.A$ and a location $o.l$. Given an input query Q, with context $\mathcal{C}$ having location and $\mathcal{C}.l$ and parameter $\mathcal{C}.\alpha$, the per-attribute score is obtained as follows:

$$sc(o, t \mid \mathcal{C}) = (1 - \mathcal{C}.\alpha)(1 - \frac{D(\mathcal{C}.l, o.l)}{maxDist}) + \mathcal{C}.\alpha \frac{TF(t, o.A)}{maxTF(t)} \quad (7.1)$$

where D gives the euclidean distance between Q's location (start point) and o's location, $maxDist$ is the maximal distance, $TF(t, o.A)$ is the term frequency of t in $o.A$, and $maxTF(t)$ is a maximal term frequency of t over all objects.

Transposing the location. Given a query Q of context $\mathcal{C}$, with location $\mathcal{C}.l$, given of view V of context $\mathcal{C}^V$, with location $\mathcal{C}^V.l$, for any object $o \in ans(V)$, there are two extreme locations at which o can be situated, relative to $\mathcal{C}.l$ (see Figure 2, left), as follows:

1. on the line connecting $\mathcal{C}.l$ and $\mathcal{C}^V.l$, between them or beyond $\mathcal{C}.l$, resulting in $D(\mathcal{C}.l, o.l) = |D(\mathcal{C}^V.l, \mathcal{C}.l) - D(\mathcal{C}^V.l, o.l)|$.
2. on a line connecting $\mathcal{C}.l$ and $\mathcal{C}^V.l$, beyond $\mathcal{C}^V.l$, giving $D(\mathcal{C}.l, o.l) = D(\mathcal{C}^V.l, \mathcal{C}.l) + D(\mathcal{C}^V.l, o.l)$.

We can now derive the following new bounds for each object o from a tuple $(o, wsc, bsc) \in ans(V)$, which would be valid in a context $\mathcal{C}'$ defined by the query's location $\mathcal{C}.l$ and the view's $\mathcal{C}^V.\alpha$:

$$sc(o, Q \mid \mathcal{C}') \geq wsc - \mathcal{C}^V.\alpha \times |Q^V| \times \frac{D(\mathcal{C}^V.l, \mathcal{C}.l)}{maxDist} = wsc'$$
$$sc(o, Q \mid \mathcal{C}') \leq bsc + \mathcal{C}^V.\alpha \times |Q^V| \times \frac{D(\mathcal{C}^V.l, \mathcal{C}.l)}{maxDist} = bsc' \quad (7.2)$$

Transposing the parameter α. We consider now the transposition for the α component, from $\mathcal{C}^V.\alpha$ to $\mathcal{C}.\alpha$, by which are obtained valid bounds for the input query context $\mathcal{C}$. For space reasons, the detailed steps of this computation are omitted and we give here directly the transposition formulas.

$$sc(o, Q \mid \mathcal{C}) \geq wsc' - |Q^V| \times \mid \mathcal{C}.\alpha - \mathcal{C}^V.\alpha \mid = \mathbf{f_w(wsc)}$$
$$sc(o, Q \mid \mathcal{C}) \leq bsc' + |Q^V| \times \mid \mathcal{C}.\alpha - \mathcal{C}^V.\alpha \mid = \mathbf{f_b(bsc)} \quad (7.3)$$

EXAMPLE 4. *Returning to the example in Motivation 1, we detail how the bounds are computed. Recall* $\mathcal{C} = (l = Q, \alpha = 0.3)$. *From* $v3$*, with context* $\mathcal{C}^{v3} = (l = v3, \alpha = 0.3)$*, knowing that* $\frac{D(\mathcal{C}.l, \mathcal{C}^{v3}.l)}{maxDist} = 0.25$*, we obtain the following bounds for the object* $o4$*, which had an initial score of* 0.962*:*

$$sc(o4, \{t2\} \mid \mathcal{C}) \geq \quad 0.962 - |0.3 - 0.3| - 0.3 \times 0.25 = 0.887,$$
$$sc(o4, \{t2\} \mid \mathcal{C}) \leq \quad 0.962 + |0.3 - 0.3| + 0.3 \times 0.25 = 1.037.$$

7.2 Social-aware search

We consider now the social-aware setting, and we revisit the ranking model developed in [16, 1]. Besides objects and attributes, we have a set of users $\mathcal{U} = \{u_1, \ldots, u_n\}$ who can bookmark (or tag) objects with attributes (tags). Also, users form a social network, seen as an undirected weighted graph: a link between two users u_1, u_2 has a weight, $\sigma(u_1, u_2) \in [0, 1]$, which could stand for proximity, similarity, etc. For pairs of users for which an explicit edge (and proximity) is not given, an extended proximity $\sigma^+(u_1, u_2) \in [0, 1]$ can be computed in the graph by aggregating (e.g., by multiplication) the weights over each path connecting u_1 and u_2, and taking the maximal aggregated score over all paths (this is reminiscent of how trust or similarity propagate, if interpreted as transitive measures):

$$\sigma^+(u_1, u_2) = \max_{p=(u_1,\ldots,u_2)} \prod_{i=0}^{k-1} \sigma(u_i, u_{i+1}).$$

A query context $\mathcal{C}$ consists now of a *seeker* $\mathcal{C}.l$ (the issuer of the query) and the parameter $\mathcal{C}.\alpha$. In manner similar to location-aware search, the per-attribute score is a linear combination between the "social location" of the seeker with respect to the taggers of an object and the classic textual score (e.g., tf/idf or BM25).

The social component of the score is computed as the sum of the proximity values of taggers of o with respect to the seeker, while the textual one is the number of taggers who tagged o with t:

$$sc(o, t \mid \mathcal{C}) = (1 - \mathcal{C}.\alpha) \times \sum_{\substack{u \ tagged \ o \\ with \ t}} \sigma^+(\mathcal{C}.l, u) + \mathcal{C}.\alpha \times TF(o, t) \quad (7.4)$$

Transposing the location. For a query Q and seekers $\mathcal{C}^V.l$ and $\mathcal{C}.l$, let u be a tagger for whom we need to use $\sigma^+(\mathcal{C}.l, u)$ in score bounds. As shown in Figure 2 right, the path with the highest score connecting $\mathcal{C}.l$ to u may either

1. go through $\mathcal{C}^V.l$, and in that case we have:
$$\sigma^+(\mathcal{C}.l, u) = \sigma^+(\mathcal{C}.l, \mathcal{C}^V.l) \times \sigma^+(\mathcal{C}^V.l, u),$$
2. not go through $\mathcal{C}^V.l$, and in that case we have:
$$\sigma^+(\mathcal{C}.l, u) \leq \sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)^{-1} \times \sigma^+(\mathcal{C}^V.l, u).$$

Now, the influence of the social component in the score of o in $ans(V)$ varies inversely with $\mathcal{C}^V.\alpha$. Therefore, the transposition that accounts for the location change should be weighted by its importance in the $sc(o, t \mid \mathcal{C}^V)$ formula, determined by $\mathcal{C}^V.\alpha$ as follows: when $\mathcal{C}^V.\alpha = 1$, the lower and upper bounds should not be affected by the location change, while when $\mathcal{C}^V.\alpha = 0$, they should be affected with weight $\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)$ and $\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)^{-1}$ respectively. We can model this by a coefficient function $c(w, \alpha)$, which applies to a weight w and value α: $c(w, \alpha) = \alpha(1 - w) + w$.

For each object o of tuples $(o, wsc, bsc) \in ans(V)$, we can now derive the following valid bounds for a context $\mathcal{C}'$ defined by the query's seeker $\mathcal{C}.l$ and the view's parameter $\mathcal{C}^V.\alpha$.

$$\begin{aligned} sc(o, t \mid \mathcal{C}') &\geq c(\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l), \mathcal{C}^V.\alpha) \times wsc &= wsc' \\ sc(o, t \mid \mathcal{C}') &\leq c(\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)^{-1}, \mathcal{C}^V.\alpha) \times bsc &= bsc' \end{aligned} \quad (7.5)$$

Transposing the parameter α. We consider now the transposition for the α component, from $\mathcal{C}^V.\alpha$ to $\mathcal{C}.\alpha$, yielding valid bounds for the new context, $\mathcal{C}$. Here, the transposition depends on the relationship between $\mathcal{C}.\alpha$ and $\mathcal{C}^V.\alpha$, and we obtain the following f_w and f_b (for space reasons the detailed description is omitted):

1. $if\ \mathcal{C}.\alpha < \mathcal{C}^V.\alpha, \mathbf{f_w}(\mathbf{wsc}) = \frac{\mathcal{C}.\alpha}{\mathcal{C}^V.\alpha} \times \mathbf{wsc'}, \mathbf{f_b}(\mathbf{bsc}) = \mathbf{bsc'}$
2. $if\ \mathcal{C}.\alpha > \mathcal{C}^V.\alpha, \mathbf{f_w}(\mathbf{wsc}) = \mathbf{wsc'}, \mathbf{f_b}(\mathbf{bsc}) = \frac{\mathcal{C}.\alpha}{\mathcal{C}^V.\alpha} \times \mathbf{bsc'}$

EXAMPLE 5. *Revisiting Motivation 2, recall* $\mathcal{C} = (l = s, \alpha = 0)$*. From* $v1$*, of context* $\mathcal{C}^{v1} = (l = v1, \alpha = 0)$*, having* $\sigma^+(s, v1) = 0.9$*, we know that* $c(\sigma^+(s, v1), 0) = \sigma^+(s, v1) = 0.9$*. So no transposition of* α *is needed, as* $\mathcal{C}.\alpha = \mathcal{C}^{v1}.\alpha = 0$*. We obtain the following bounds for* $o2$*, which had an initial score of* 1.35*:*

$$sc(o2, \{t1, t2\} \mid \mathcal{C}) \geq 0.9 \times 1.4 = 1.26,$$
$$sc(o2, \{t1, t2\} \mid \mathcal{C}) \leq \frac{1}{0.9} \times 1.4 = 1.55.$$

8. EXPERIMENTS

We performed our experiments on a single core of a i7-860 2.8GHz machine equipped with 8GB of RAM. We implemented our algorithms in Java, and we used this implementation for our tests on synthetic data and social data. We also implemented them in C++, for a more reliable comparison with IR-TREE, for spatial data.

Context-agnostic setting with complete views. Our first series of tests, over synthetic data, concerns a setting in which the input queries and the views share the same context (i.e., context plays no role and is ignored in the computation). We generated exact scores in the range $[0, 100]$ for 100,000 objects and 10 attributes, with exponential or uniform distributions. Then, we generated all possible combinations of 2 or 3 attributes, each representing one view. For each of the views, we computed the exact (aggregated) scores over *all* objects; the views are *complete* in that sense. We then made these lists uncertain by replacing each exact value by a score range, using a gaussian distribution with mean equal to the exact value and standard deviation (std, in short) equal to either 5, 10 or 20. the sets of views we obtained, we used 100 randomly-generated input queries consisting of 5 distinct attributes.

We compare in Fig. 3 the SR-TA variants over the two data distributions, for the std values 5 and 10 (to avoid clutter, plots for std 20 are not given). included. We have recorded (i) the relative running-time of the algorithms that use view selection w.r.t. the algorithm using all the views – 3 selection criteria per 2 std values, for 6 plot lines, (ii) the number of sequential accesses by all 4 variants – with the 2 std values, for 8 plot lines, and (iii) the number of random accesses by all 4 variants – with the 2 std values, for 8 plot lines.

One can note that the algorithms with view selection achieve significant savings in terms of both running-time and I/O accesses. The algorithm based on max-statistics, SR-TA^{max}, achieves better performance than the one based on view definitions, SR-TA^{def}, which in turn does better than the one based on average-statistics, SR-TA^{avg}. definitions of the views. Moreover, note that the relative running-time of these algorithms does not depend on the value of k, and the influence of the interval coarseness (by standard deviation) is more important in the exponential distribution. One can also note a "clustering" effect, by standard deviation, in the case of sequential-access measures; this is likely due to the fact that top-k processing on noisier data needs to go deeper in the views to reach termination.

We also compared the performance of SR-TA^{sel} variants, over score ranges with low noise (std of 5), with the one of Fagin's TA over the exact per-attribute inverted lists. We trace two measures: the relative running-time and the minimum precision. (We do not evaluate using NDCG-like measures because the top-k sets we are returning are unranked, for both the *sel* and *nosel* variants. Our goal is to return as many of the G objects as possible, and hence precision is the best measure for our purposes.) The latter is computed as $|G|/k$, i.e., the ratio between the size of the guaranteed set and the required k. The results are presented in Table 2. One can note that

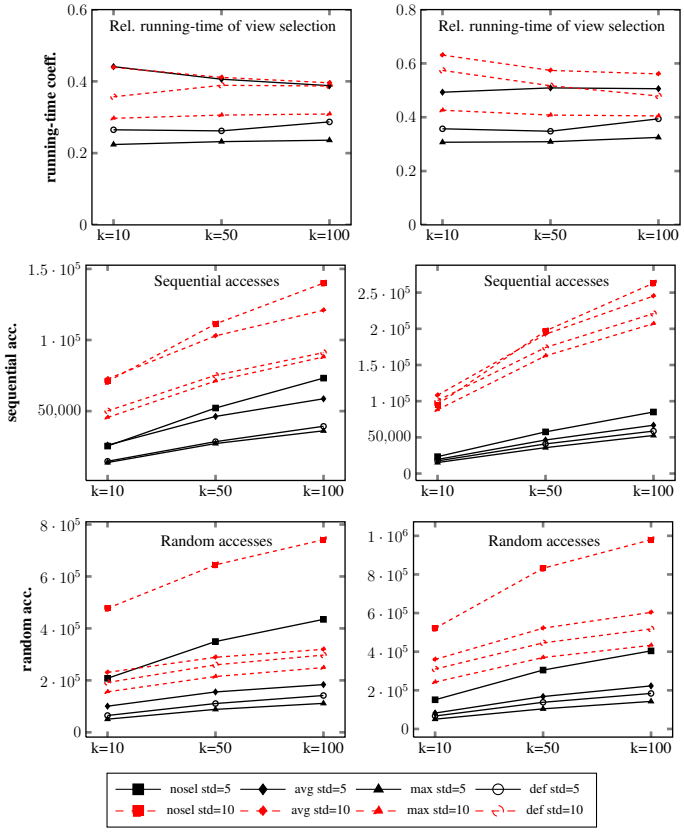

Figure 3: Performance comparison for SR-TA variants over synthetic data with uniform (left column) or exponential distribution (right).

SR-TA^{sel} can have a running-time that is a low fraction of the one of TA (as low as 0.296, with a precision@10 of 0.577). This is mainly due to the fact that, although inexact, we have aggregated scores pertaining to 2 or 3 query terms, while the noise levels are rather low. While using exact lists of aggregated data for top-k processing would certainly improve efficiency, as shown in [11], our experiments show that even relatively noisy aggregated data can lead to improvements, with reasonable precision.

We give in Table 3 the overhead of the refinement step discussed in Section 5, which uses random-accessing to refine a result ($\tilde{G}$, $\tilde{P}$) to the most informative one, (G,P). This overhead is measured as the ratio between the running-time of the base algorithm and the one of the refined algorithm. We also report on the Δ measure. Note that, while the number of possible combinations that are "avoided" increases exponentially with the standard deviation, the overhead of additional I/O accesses is small (range 3%-13%).

Table 2: Comparison between the SR-TA^{sel} variants and TA (exact scores), for uniform and exponential distributions, for std 5.

Type	Rel. running-time			Min. precision			$\|P\|$		
	10	50	100	10	50	100	10	50	100
avg/uni	0.576	0.676	0.712	0.57	0.69	0.72	10	36	64
def/uni	0.350	0.446	0.544	0.57	0.69	0.72	10	36	64
max/uni	0.296	0.395	0.446	0.57	0.69	0.72	10	36	64
avg/exp	0.732	1.128	1.287	0.60	0.63	0.64	10	46	86
def/exp	0.531	0.771	1.003	0.60	0.63	0.64	10	46	86
max/exp	0.456	0.684	0.827	0.60	0.63	0.64	10	46	86

Table 3: Running-time overhead and Δ, for SR-TA^{sel} with or without the final refinement, for k=100, exponential distribution.

Sel.	Std	Overhead	$\|G\| - \|\tilde{G}\|$	$\|P\| - \|\tilde{P}\|$	Δ
avg	5	0.031	38	-208	1.96×10^{70}
avg	10	0.033	35	-734	4.14×10^{129}
avg	20	0.119	15	-4828	2.65×10^{212}
def	5	0.040	37	-206	2.76×10^{69}
def	10	0.038	34	-727	1.96×10^{129}
def	20	0.138	15	-4749	5.93×10^{211}
max	5	0.041	35	-179	5.54×10^{64}
max	10	0.041	33	-575	6.38×10^{119}
max	20	0.117	15	-3592	7.96×10^{200}

Location-aware search. We used in this setting the same PolyBot dataset as before, taking into account for each object its 2D coordinates. We generated 20 views defined by 2-term queries at 5 locations, varying the size of their ans lists (500, 1000 and 2000 entries). We used 10 to-be-answered queries at 5 locations (different to the ones of views) and we varied $k \in \{10, 20\}$ and $\alpha \in \{0.7, 0.8, 0.9\}$ (values which are close to those employed in [5]).

The algorithm we use as the baseline in our evaluation is our implementation of the IR-TREE of [5]. It is based on R-tree indices [8], whose nodes are enriched with inverted lists consisting of the documents located inside the rectangle defined by the node.

We present in Figure 4 the results for relative running-time and precision. The relative running-time is computed as the ratio between the running-time of SR-TA and the one of IR-TREE. Precision is computed as the percentage of top-k items returned by SR-TA that also appear in the output of IR-TREE. Here, we used the sampling method from Section 4 to obtain the most likely top-k from the (G,P) answer, through 1,000 rounds of uniform sampling.

One can note that, for high values of α and low values of k, the response time of SR-TA is significantly lower than that of the IR-TREE (in practice, of the order of milliseconds), with reasonably high precision levels (between 0.86 and 0.92). This is because the top-k answer is based on a large set G of guaranteed objects, which reduces the overhead of the sampling procedure. When the uncertainty introduced by coarser score ranges in views leads to larger sets P instead, the sampling procedure is more costly, but overall the running-time remains a small fraction of the one of the IR-TREE, with a precision around 0.8.

Social-aware search. For this application, we used the existing Delicious tagging data of [18]. We selected a random subset, containing 80,000 users, their tagging on 595,811 objects (items) with 198,080 attributes (tags). For assigning weights to links between users, we generated three similarity networks, by computing the Dice coefficients of either (i) common tags in a tag similarity network, (ii) common items in an item similarity network, or (iii) common item-tag pairs in an item-tag similarity network.

For each of the three similarity networks, we randomly chose 5 seekers for our tests. Then, a number of 10 users were randomly chosen, among those having a link with weight of at most 0.66 to any of the 5 seekers (to ensure that no view is too "useful", having too strong an influence on the running-time and precision). For each of these users and for $\alpha \in \{0.0, 0.1, 0.2, 0.3\}$, we generated 40 views of 1 and 2-tag queries, each containing 500 entries.

We tested on a set of ten 3-tag queries for each of the 5 seekers, varying $\alpha \in \{0.0, 0.1, 0.2, 0.3\}$ and $k \in \{10, 20\}$.

The baseline algorithm we used for the performance comparison is a direct adaptation of the CONTEXTMERGE algorithm of [16]. Depending on the value of α, CONTEXTMERGE alternates between

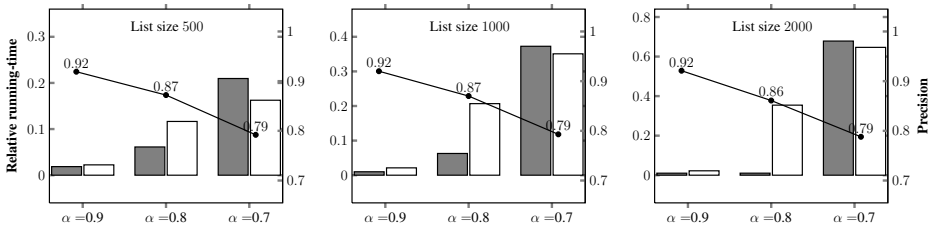

Figure 4: Location-aware search: performance & precision of SR-TA^{sel} vs. exact early-termination algorithm (IR-TREE [5]), for various α values and list sizes (grey=top-10, white=top-20).

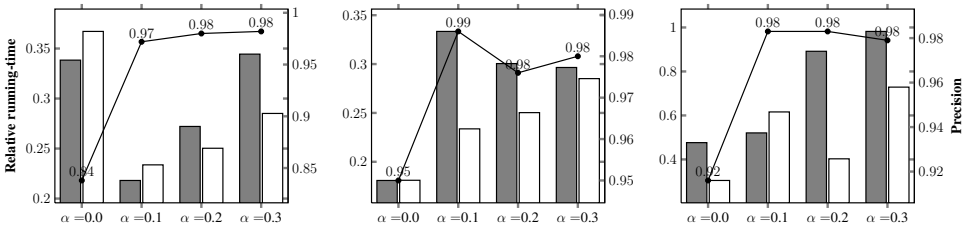

Figure 5: Social-aware search: performance & precision of SR-TA^{sel} vs. exact early-termination algorithm (CONTEXTMERGE [16]), in 3 similarity networks, left-to-right: per tag, per item, per item-tag.

per-attribute inverted lists of objects and an inverted list containing users ordered descending by their proximity relative to the seeker.

Similar to the location-aware search, we present in Figure 5 the results in terms of relative running-time and precision. One can note that the running-time is still a low fraction of the one of the exact algorithm, while the precision levels are considerably higher than in the case of location-aware search. As expected, the lowest precision levels are obtained when the search relies exclusively on the social component of the score. This is due to the fact that the bounds computed by Eq. (7.5) yield coarser score ranges when $\alpha = 0$, which are source of more uncertainty in the scores and the top-k result. Moreover, due to the skew in proximity values in the network, even when α has low non-zero values, the textual component has a strong influence in scores, and thus leads to significant improvements in the top-k estimates (the most likely result).

9. CONCLUSIONS

We formalize and study in this paper the problem of context-aware top-k processing based on uncertain precomputed results, in the form of *views* over the data. This problem is motivated by search applications in which query results depend on a context, and any result caching or pre-computation mechanism needs to perform certain transformations – what we call a *context transposition* – in order to answer new queries, which may pertain to new contexts.

We introduce the query semantics needed for dealing with objects of uncertain scores and describe an algorithm, SR-TA that outputs what we call *the most informative result*. We also consider optimizations based on selecting some (few) most promising views, instead of using the entire set of views.

Acknowledgments. This work was partially supported by the EU project ARCOMEM FP7-ICT-270239.

10. REFERENCES

[1] S. Amer-Yahia, M. Benedikt, L. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. In *VLDB*, 2008.

[2] X. Cao, G. Cong, and C. Jensen. Retrieving top-k prestige-based relevant spatial web objects. In *PVLDB*, 2010.

[3] X. Cao, G. Cong, C. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, 2011.

[4] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. Text vs. space: Efficient geo-search query processing. In *CIKM*, 2011.

[5] G. Cong, C. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In *VLDB*, 2009.

[6] G. Das, D. Gunopulos, N. Koudas, and D. Tsirogiannis. Answering top-k queries using views. In *VLDB*, 2006.

[7] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *JCSS*, 2003.

[8] A. Guttman. R-tree: a dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[9] V. Hristidis and Y. Papakonstantinou. Algorithms and applications for answering ranked queries using ranked views. In *VLDBJ*, 2004.

[10] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. In *ACM Comp. Surv.*, 2008.

[11] R. Kumar, K. Punera, T. Suel, and V. Sergei. Top-k aggregation using intersections of ranked inputs. In *WSDM*, 2009.

[12] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. In *VLDB*, 2009.

[13] S. Maniu and B. Cautis. Taagle: Efficient, personalized search in collaborative tagging networks. In *SIGMOD*, 2012.

[14] S. Maniu and B. Cautis. Network-aware search in social tagging applications: Instance optimality versus efficiency. In *CIKM*, 2013.

[15] J. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Norvag. Efficient processing of top-k spatial preference queries. In *VLDB*, 2010.

[16] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. Parreira, and G. Weikum. Efficient top-k querying over social-tagging networks. In *SIGIR*, 2008.

[17] M. Soliman, I. Ilyas, and S. B.-David. Supporting ranking queries on uncertain and incomplete data. In *VLDBJ*, 2010.

[18] R. Wetzker, C. Zimmermann, and C. Bauckhage. Analyzing social bookmarking items: A del.icio.us cookbook. In *ECAI Mining Social Data*, 2008.

[19] H. Yan, S. Ding, and T. Suel. Inverted index compression and query processing with optimized document ordering. In *WWW*, 2009.

[20] J. Zhang, X. Long, and T. Suel. Performance of compressed inverted list caching in search engines. In *WWW*, 2008.

[21] J. Zobel and A. Moffat. Inverted files for text search engines. In *ACM Comp. Surv.*, 2006.