Integrating Similarity Retrieval and Skyline Exploration via Relevance Feedback

Yiming Ma, and Sharad Mehrotra *

Dept. of Information and Computer Science, University of California, Irvine
Irvine, CA, USA

Abstract. Similarity retrieval have been widely used in many practical search applications. A similarity query model can be viewed as a logical combination of a set of similarity predicates. A user can initialize a query model, but model parameters or the model itself may be inadequately specified. As a result, a retrieval system cannot guarantee that it has presented all the relevant tuples to the user. In this paper, we propose a framework that integrates the similarity retrieval and skyline exploration. Using the relevance feedback as a way to constrain the search space, our framework can intelligently explore only a necessary portion of data that contains all the relevant tuples. Our framework is also flexible enough to incorporate model refinement techniques to retrieving relevant results as early as possible.

1 Introduction

Similarity retrieval is attractive since it presents results quickly to the user in relevance order and allows the search to stop when enough results are seen (as contrasted to a potentially large collection of results from which a user must choose the relevant ones). A fundamental weakness of the similarity query model is that it requires a user to accurately specify the model parameters which, given the complexity of search spaces, might be a difficult (or impossible) task. If the user does not specify the parameters accurately, the system cannot guarantee to retrieve all the relevant results. For instance, if the user stops the search after retrieving k objects because the latest objects retrieved were irrelevant, there is no guarantee that the unseen objects are also irrelevant. It is possible that the *best* answer resides in the unseen results. One approach that can guarantee the answer set containing best results is by using a skyline [1]. In a skyline setting, the system pessimistically assumes no knowledge of the query model; it knows only the similarity predicates in a user's search. Instead of returning objects based on relevance to the user, a skyline operator returns a set of objects that are not dominated by any other object in at least one similarity dimension (formed by a similarity predicate). This way, the top result is guaranteed to be in the return set (irrespective of the user's similarity query model). While skyline

* This research was sponsored by NSF Award number 0331707 and 0331690 to the RESCUE project

retrieval offers the guarantee to the best results, it suffers from three problems: (1) the size of the return set (the skyline) may be large and the skyline size increases as the dimensionality of the similarity space increases, (2) since the returned objects are no longer based on any ranking criteria (within a skyline), if a user stops the search prior to viewing the entire skyline, the top results may be missed, and (3) it is not an interactive process (i.e., does not consider relevance feedback).

In this paper, we build a search strategy that combines the positive aspects of both similarity retrieval and skyline retrieval into one single technique, so that we can retrieve results in the order of relevance, yet support the notion of completeness. The key aspect of our technique relies on exploiting the relevance feedback gathered from a user. We use relevance feedback in two ways, which also represent the major contributions of this work:

• Initialized by a ranked retrieval, we use irrelevant (negative) feedback to progressively form an interactive skyline (I-Skyline), which dynamically constrains the search space (Section 3).

• Using both relevant (positive) and irrelevant (negative) feedback, we introduce query model refinement techniques to improve the search quality; so that the relevant tuples will be retrieved more effectively from the search space bounded by I-Skyline (Described in the full version [3]).

2 Related Work

To the best of our knowledge, no previous work uses relevance feedback as a bridge to effectively integrate similarity retrieval and skyline retrieval. Our work significantly differs from these retrieval and refinement approaches (e.g., [4, 7]) since we focus on the existing query formulation, and attempts to give a user a sense of the query completion as we dynamically prune the search space based on the user feedback. In addition, our refinement techniques are built on top of the dynamic search space. The techniques are not available in any of the refinement systems since those systems may alter the query formulation (e.g., predicate addition/deletion). During a search session, a user may never know if his initial query has been completed or not. In fact, he may be even confused by the returned tuples since he does not know the exact query formulation used to rank the returned tuples. Our work is also very different from the work on skyline (e.g., [5]), which mainly focus on the efficiency issues. Many of these techniques assumes the ranking function is defined in a feature/data space, and an index structure (e.g., R-tree) is available in the feature space. By making these assumptions, it imposes limitations on the query, such that every attribute can appear at most once in the query, and similarity predicates can only use distance measures in the feature space. Therefore, it does not support general similarity queries. Furthermore, it does not exploit the relevance feedback information to bound the search space or refine the ranking function.

P1: MinPrice (Price, 150)
P2: MinStop(#Stop, 0)
P3: MinFlightTm(FlightTm, 2.5)
P4: ArrTm (ArrTm, 1500)
Query Model: $(P1 \vee (P2 \wedge P3)) \wedge P4$

Fig. 1. Example Similarity Predicates and Query Model

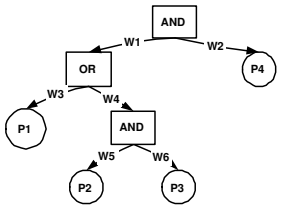

Fig. 2. A Logic Tree $(P1 \vee (P2 \wedge P3)) \wedge P4$

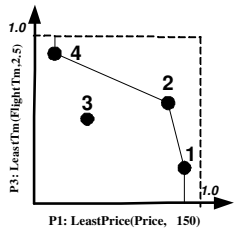

Fig. 3. Skyline on P1, P3

Terminology	Explainations
Similarity Predicate	Similarity based logic predicate.
Query Model (Q)	A logic combination function built on top of the similarity predicates.
Logic Combination Function	Used exchangeably with query model.
Monotone Function	Defined in [2], a query model is also a monotone combination function.
Parameters in Query Model	Weights and p values used in the query model (P-Norm [6]).
Logic Tree (LT)	An operator tree representation of query model.
Full Similarity Space (FS)	A similarity space defined by all the similarity predicates.

Fig. 4. Terminologies

3 I-Skyline Framework

In Figure 4, we summarize the concepts and the terminologies used in this paper. In this paper, we assume that users can specify all the similarity predicates of interest to their information needs. We focus on the similarity query model which, when similarity semantics are involved, can be difficult to specify correctly. For instance, a flight ticketing database has four attributes: price, number of stops, flight time, and time of arrival. A typical query is to find flights that conform to a certain desirable hypothesis expressed as a similarity query. In our example, the search has four similarity predicates: $MinPrice$, $MinStop$, $MinFlightTm$ and $ArrTm$ with obvious semantics. Figure 1 shows an example of similarity predicates and a query model. Given a data tuple, the query model aggregates the predicate level scores to a single relevance score using a set of logical operators (AND, OR). A user invokes this query to find a flight with the cheapest fare or least number of stops with the shortest flight time; the flight should also arrive in Seattle at around 3pm. $(P1 \vee (P2 \wedge P3)) \wedge P4$ nicely captures this search request. In general, a query model using logical operators can be always viewed as an operator tree. Figure 2 shows such an operator tree; an internal node is a logical operator, and a leaf node corresponds to a similarity predicate. Outgoing edges from an internal node connect the components used in a logical operator. In this paper, we use P-Norm [6] to interpret logical operators. Because of the tunable parameters (weights and P values), the P-Norm can be expressive. However, if the initial parameter settings of a query model differ from the ideal ones, the order of the returned tuples can change considerably. Skyline could be utilized to retrieve the best answer. Given any d similarity predicates, if we define a d-dimensional space on these predicates and project data points into the space using their similarity scores, the skyline is guaranteed to consist the

```
Input: Database(D), FullSimSpace(FS), Monotone Comb. Func. (F)
Output: I-Skyline, RelevantSet
1: I-Skyline = ∅, RelevantSet = ∅
2: RL=compute_RankedList(D, FS, F)
3: for each tuple t in RL do
4:   if t is NOT Dominated by any tuple t2 ∈ I-Skyline then
5:     FB = getFeedbackFromUser(t)
6:     if FB == Irrelevant then
7:       I-Skyline.insert(t)          // Grow I-Skyline set.
8:     else
9:       RelevantSet.insert(t)        // Grow Relevant set.
```

Fig. 5. I-Skyline_Base

best point under any monotone query models [1]. Figure 3 shows an example of a 2-dimension space defined on predicates P1 and P3 in Figure 1. A skyline retrieval will return tuples 1,2 and 4.

Instead of retrieving one best record as the skyline retrieval, in this paper, the goal is to retrieving all the relevant tuples. We assume there is an optimal query formulation Q^{opt}. The relevant tuples are a list of top tuples that having similarity scores above a threshold τ. Without knowing the Q^{opt} and τ, the problem is how to retrieve all the relevant tuples with minimum number of irrelevant tuples given an initial query Q.

We now present the framework I-Skyline algorithm called *I-Skyline_Base* in pseudo-code (Figure 5). We first define I-Skyline as a skyline on all irrelevant tuples in a given full similarity space FS (Figure 4). The algorithm *I-Skyline_Base* sits in between a ranked retrieval system (line 2) and a user. It interacts with the user (line 5) and progressively selects tuples that the user needs to see (line 3 to 9). During the process, only two sets – I-Skyline set and Relevant set – are dynamically constructed (line 7 and line 9). It can be formally proved that these two sets contain necessary (optimal) set of tuples that the user needs to interact with if there is no prior knowledge to the query model.

The baseline algorithm can be easily extended and enhanced in various ways such as exploiting the partial knowledge provided to the query structure or aggressively improving the retrieval quality by incorporating various refinement and learning techniques. In the full version of this paper [3], we provide detail discussions and extensive evaluations to these strategies.

References

1. S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*. 2001.
2. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.
3. Y. Ma and S. Mehrotra. I-skyline: A systematic approach in integrating similarity retrieval and skyline exploration via relevance feedback. *UCI Technical Report available at http://www. ics. uci. edu/˜ maym/publications/iskyline. pdf*, 2006.
4. Y. Ma, S. Mehrotra, and Q. Zhong. RAF: An Activation Framework for Refining Similarity Queries Using Learning Techniques. In *DASFAA*, 2006.
5. D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, 2003.
6. G. Salton, E. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 1983.
7. L. Wu, C. Faloutsos, K. Sycara, and T. Payne. FALCON: Feedback adaptive loop for content-based retrieval. In *VLDB*, 2000.