Interactive Toolbox for Spatial-Textual Preference Queries

Florian Wenzel, Dominik Köppl, Werner Kießling

Department of Computer Science, University of Augsburg
D-86135 Augsburg, Germany
`{wenzel,koeppl,kiessling}@informatik.uni-augsburg.de`

Abstract. Spatial-textual data is ubiquitous on websites such as OpenStreetMap or Wikipedia and is published progressively by Open Government Data Initiatives. Those data sources provide the base for novel mashup applications for Location-Based Services. We present a mobile prototype for the San Francisco area based on *Preference SQL* that allows an intuitive expression of spatial keyword queries. Search can further be extended towards temporal, categorical, and numerical attributes. Our demo provides a fully flexible toolbox for generating extended spatial-textual queries in non-metric spaces which are evaluated using a *Best-Matches-Only* query model. Spatial relevance is defined by an asymmetric routing distance supporting complex geometries. Textual relevance is determined using the Apache Lucene library.

1 Introduction

Websites such as Tripadvisor or Wikipedia provide spatial-textual data in the form of geo-tagged content. Additionally, Open Government Data Initiatives lead to the publication of large amounts of public spatial data by cities such as San Francisco or Berlin. All these data sources are a base for up-and-coming mobile mashup applications that combine base information to provide novel Location-Based Services (LBS). These integrated applications require a dynamic combination of different data sources in form of temporary relations. Additionally, numerical, categorical, and temporal attributes are of importance. Spatial Keyword (SK) Queries have to be extended towards this end to combine all these attributes in a semantically intuitive fashion in order to provide high quality personalized search results. These preconditions render the use of current index-based approaches inapplicable. Furthermore, the heterogeneity of spatial and textual data among data sources demands a flexible query and data model. The representation of spatial resources as a mix of points and complex geometries asks for a flexible spatial relevance definition. As asymmetric distances occur through one-way streets and terrain topology, distance axioms have to be relaxed towards non-metric spaces to allow queries based on net distances. Semi-structured and unstructured text pose additional demands on textual relevance determination. The Preference SQL System [2] provides a rich toolbox for developers that meets these new challenges. Base preference constructors can be combined in a flexible fashion via complex preference constructors to form personalized Preference SQL queries. Spatial relevance is evaluated using an asymmetric routing distance. Textual relevance is determined using the Apache Lucene library. We illustrate how developers can employ Preference SQL for innovative LBS by presenting a mobile LBS application for the San Francisco area. This research prototype illustrates how base preferences can be defined and combined without knowledge of any query syntax based on three provided use cases. Movie scenes, restaurants, or landmarks can be retrieved according to their spatial, textual, temporal, numerical, and categorical attributes. Furthermore, the individual importance of base preferences can be modified to form complex Preference SQL queries. To the best of our knowledge, there is no other framework to treat this kind of extended SK queries on non-metric spaces.

2 Preference SQL Overview

The presented demo application relies on the Preference SQL system as underlying representation of preferences as strict partial orders [1]. Preference SQL enhances the SQL standard by a `PREFERRING` clause that specifies preferences by means of preference constructors given in [2]. These preferences are evaluated as ***soft constraints*** on base of the results generated by SQL hard constraints. The syntax allows for further post-filtering.

Preference evaluation follows a ***Best-Matches-Only query model***, that defines for a preference $P = (A, <_P)$ on a relation $R = (A_1, \cdots, A_n) \supseteq A$ the preference selection operator by

$$\sigma[P]\,(R) := \{t \in R |\ \nexists t' \in R : t.A <_P t'.A\}\,.$$

The Preference SQL system is implemented as Java-middleware on top of conventional database systems such as Oracle or Postgres and provides a JDBC driver for seamless integration into applications. The system implements a parser, heuristic and cost based optimizer, and efficient evaluation algorithms such as BNL, BNL++, LESS, SFS, and Hexagon [2].

2.1 Constructor-Based Approach

The system follows a constructor based approach by dividing preferences into base preferences operating on attributes, and complex preferences that combine multiple preferences. For an overview of the corresponding query syntax we refer to [2]. Base preference constructors displayed in a 'is-a'-hierarchical view in Figure 1 provide a flexible toolbox for the expression of base preferences on spatial, textual, temporal, numerical, and categorical domains. All displayed base preferences are sub-constructors of the $\texttt{SCORE}_d$ preference, which generates a preference order using a scoring function $f(x)$ as follows:

$$x <_P y \text{ iff } f_d(x) > f_d(y) \text{ with } f_d(x) = \begin{cases} \left\lceil \frac{f(x)}{d} \right\rceil & \text{if } d > 0 \\ f(x) & \text{else} \end{cases}$$

The optional d-parameter can be used to form equivalence classes of equally important $f(x)$ values. Numerical preferences define $f(x)$ based on deviation from a desired input value. In the case of `BETWEEN`, numerical values within a user-defined interval are preferred. Categorical preferences such as `LAYERED` define $f(x)$ based on sets of preferred domain values.

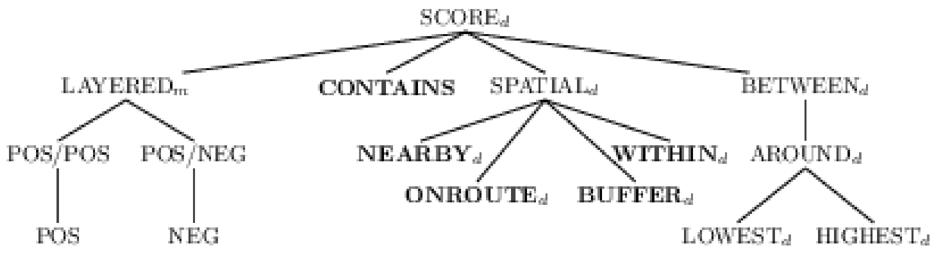


Fig. 1: Taxonomy of base preference constructors

Complex preference constructors provide the ability to either combine multiple preferences or to fine-tune the behavior of a preference. The dual operator δ is such a complex preference. It reverses a preference order, i.e. $a <_{P\delta} b :\Leftrightarrow b <_P a$. Equal importance is achieved by applying a Pareto constructor that performs Skyline evaluation. The Prioritization constructor determines the first mentioned preference as more important. Only in cases of equality or indifference considering the first preference, the second preference is evaluated. Ranked importance can be implied by using the `RANK`$_F$ constructor which performs ranking according to a ranking function F.

2.2 Toolbox of Spatial-Textual Preference Constructors

The presented system provides full flexibility for developers to generate intuitive Preference SQL queries. Within this toolbox, spatial and textual constructors are key components and are thus highlighted consecutively. In contrast to query engines using an exact query model, spatial-textual preference queries are evaluated as soft constraints following the Best-Matches-Only query model.

▷ **Spatial Preferences:** Given a query geometry g_q preferred by the user, spatial preferences determine those data geometries d_i of a database relation that are best matches according to the spatial relevance defined by the preference constructor. The query model uses Keyhole Markup Language (KML) to define g_q. The data model supports geometry types of underlying PostGIS or Oracle Spatial database extensions. As shown in Figure 1, `NEARBY, WITHIN, ONROUTE` and `BUFFER` can be used to express spatial preferences which define relevance based on distance. `WITHIN` describes a preference on an attribute $A \ni d_i$ favoring geometric objects that are within or close to a region g_q. A geometric object d_i is better than d_j if $dist(g_q, d_i) < dist(g_q, d_j)$. Applicable distances are described below. In the given query model, relevance can be expressed by using a point, a region, or a line as query geometries g_q. Hence, the preferences `NEARBY` and `ONROUTE` follow the concept of `WITHIN`, but differ in the fact that `NEARBY` accepts a point and `ONROUTE` a linestring instead of a region for g_q. A more comprehensive intention can be expressed with the `BUFFER` constructor which also accepts a region and treats geometries closer to g_q as more favorable, but geometries within g_q are considered as least favorable.

As sub-constructors of `SCORE`, the preference order is induced by a scoring function $f_{dist}(x) := dist(g_q, x)$. *dist* can be substituted with `ST_MaxDistance`, `ST_Distance` and `net_dist`. `ST_Distance` calculates the minimal distance between two geometries, whereas `ST_MaxDistance` computes the maximal distance:

$$\texttt{ST\_Distance}(A, B) := \min_{a \in A, b \in B} \|a - b\|_2$$

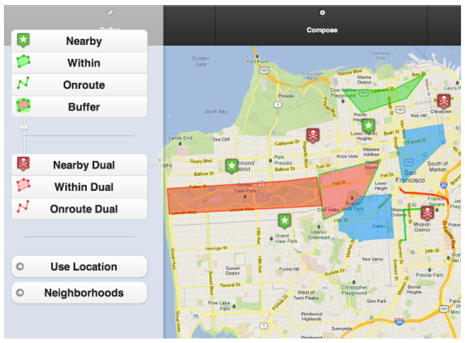
$$\texttt{ST\_MaxDistance}(A, B) := \max_{a \in A, b \in B} \|a - b\|_2$$

Here, $\|\cdot\|_2$ is the classic Euclidean norm and A, B are geometric objects regarded as a set of points. Both functions correspond to the SQL/MM standard and are executed by the database system. Alternatively, `net_dist` is a distance proprietary to Preference SQL as it employs a distance calculated using PgRouting, based on a road network from OpenStreetMap. In contrast to Euclidean distance, the routing result is inherently asymmetric. In urban environments, one-way streets lead to asymmetric results. Using costs such as duration, asymmetry gets even more apparent in mountainous areas where terrain topology becomes of importance. Preference SQL further accounts for transport modality by letting users define routing to be performed for cars or pedestrians.

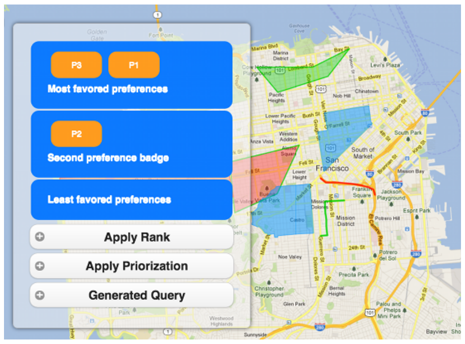
▷ **Textual Preferences:** The `CONTAINS` constructor is provided for textual domains. Apache Lucene is used for evaluation by using Lucene's Score as scoring function, consequently the query model provides full Lucene functionality with respect to search keywords, including wildcards and fuzzy search. These scores can be customized by using implemented similarity distances like tf–idf or by defining new ones. Considering the data model, language stemmer and tokenizer for semi-structured data like Wikipedia entries can be specified. This functionality provides a powerful means to state preferences on unstructured or semi-structured text presented on websites in the form of reviews or descriptions. Furthermore, text-search functionality can be combined with any other kind of base preference constructors with the help of complex preference constructors.

3 Showcase Application

We present a dynamic HTML5 application based on the jQuery Mobile framework which communicates with Preference SQL via JDBC. Based on use cases in the San Francisco area, we demonstrate how Preference SQL can be used by developers to provide users with an intuitive preference based LBS. Spatial preferences can be defined by drawing query geometries on a map. Additional icons allow to express those preferences with the dual operator applied. In a drop-down list, preferred city districts can be selected which are also visualized on a map. The top of the screen defines the three consecutively described operations.



(a) Spatial Preference Selection



(b) Complex Preference Composition

▷ **Define:** A pop-up lets users select one of the following use cases: (U1) combines movie locations listed by the SF Data Project [1] with movie data from the IMDB database [2]. (U2) combines restaurant inspection results from the SF Data Project with reviews from Tripadvisor [3]. (U3) joins geometry data and tags of Points and Regions of Interest (POI/ROI) from OpenStreetMap [4] with geo-tagged content from Wikipedia [5]. For each use case, a form pop-up provides input for individual base preferences on attributes of the joined data sources. A text-search functionality further allows input of complex search terms. Aliases are assigned to each base preference which are displayed in an overview overlay.
▷ **Compose:** A pop-up allows users to compose complex preferences. Initially, three levels are displayed in which base preferences can be dragged from the overview. Levels are arranged in order of importance, with the most important level at the top of the list. As soon as a preference is placed in the lowest level, an additional level is created underneath. Levels are combined using the Prioritization constructor. Preferences within a level are interpreted as equally important by the use of the Pareto constructor. Ranking and further Prioritization can be applied to each level. The generated Preference SQL query is displayed instantly.
▷ **Search:** After clicking the search button, the generated query is evaluated by the Preference SQL system and best-matching results are shown.

Bibliography

[1] Kießling, W.: Foundations of Preferences in Database Systems. In: Proceedings of 28th Int. VLDB conference. pp. 311–322. Morgan Kaufmann (2002)
[2] Kießling, W., Endres, M., Wenzel, F.: The Preference SQL System - An Overview. IEEE Data Engineering Bulletin 34(2), 11–18 (2011)

[1] http://www.datasf.org
[2] http://www.imdb.com/interfaces
[3] http://www.tripadvisor.com
[4] http://www.openstreetmap.org
[5] http://www.wikipedia.org