Preprint of paper to be presented at
16th International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE), October 2017, Rhodes, Greece

# PFSgeo: Preference-enriched Faceted Search for Geographical Data

Panagiotis Lionakis* and Yannis Tzitzikas

Institute of Computer Science, FORTH-ICS, GREECE
Computer Science Department, University of Crete, GREECE
{lionakis,tzitzik}@ics.forth.gr

**Abstract.** In this paper we show how an *exploratory search process*, specifically the *Preference-enriched Faceted Search* (PFS) process, can be enriched for exploring datasets that also contain *geographic* information. In the introduced extension, that we call *PFSgeo*, the objects can have geographical coordinates, the interaction model is extended, and the web-interface is enriched with a map which the user can use for inspecting and restricting his focus, as well as for expressing *preferences*. Preference inheritance is supported as well as an automatic scope-based resolution of conflicts. We detail the implementation of the interaction model, elaborate on performance and report the positive results of a task-based evaluation with users. The value of *PFSgeo* is that it provides a generic and interactive method for aiding users to select the desired option(s) among a set of options that are described by several attributes including geographical ones, and it is the first model that supports map-based preferences.

**Keywords:** RDF and geospatial data; faceted search; exploratory search; preferences; map visualization;

## 1 Introduction

A plethora of datasets contain geographic information and there are several approaches that combine Linked Data[1] and spatial data, create virtual geospatial RDF graphs on top of geographical databases [4] or use well-established existing controlled vocabularies (thesauri) and ontologies to enhance metadata documents with synonyms and translated terms, as well as location names for an improved discovery and linked-data eligibility using bounding box or text-based search [21]. For example, LinkedGeoData[2] is an effort to add a spatial dimension to the Web of Data / Semantic Web. LinkedGeoData uses the information collected by the OpenStreetMap[3] project and makes it available as an RDF dataset according to the Linked Data principles. Similar to this, GeoLinkedData [9] is an

* Corresponding author. Email : lionakis@ics.forth.gr
[1] http://lod-cloud.net/
[2] http://linkedgeodata.org/
[3] http://www.openstreetmap.org/

open initiative of the Ontology Engineering Group (OEG) whose aim is to enrich the Web of Data with Spanish geospatial data. Moreover, there are works such as GeoNames that provides a geographical database available and accessible under a Creative Commons Attribution 3.0 License, containing over 10 million geographical names corresponding to over 9 million unique features [19]. Great Britain's national mapping agency, Ordnance Survey[4], has been the first national mapping agency committed to make publicly available various kinds of geospatial data from Great Britain as open Linked Data. Similar to this, data.geohive.ie[5] aims to provide an authoritative platform for serving Ireland's national geospatial data including Linked Data. Consequently there is a need for generic, flexible and interactive methods that allow users to explore such datasets and get an overview of the results on the map which is the most prominent technique in various domain-specific commercial platforms (e.g. Booking.com, Airbnb etc.).

*Faceted search* [12] is the more prominent technique in e-commerce and tourism services, and has been generalized also for RDF datasets (see [17] for a survey). The enrichment of faceted search with *preferences*, hereafter *Preference-enriched Faceted Search* [18], for short PFS, has been proven useful for recall-oriented information needs, because such needs involve decision making that can benefit from the gradual interaction and expression of preferences. PFS supports user clicks that correspond to either *hard* or *soft* constraints. The first kind corresponds to the actions of the classical faceted search where the user can *restrict* (i.e. filter) his focus gradually while getting an overview of the information space (just like in Booking.com). The second kind of actions corresponds to actions that specify *preferences* that *rank* accordingly the information space, e.g. a user can say that 2-star is the most preferred category and this statement will *rank first* the 2-star hotels but will not vanish the rest. The distinctive features of PFS is that it allows expressing preferences over attributes whose values can be hierarchically organized (and/or multi-valued), it supports preference inheritance, and it offers scope-based rules for resolving automatically the conflicts that may arise. As a result the user is able to restrict his current focus through the faceted interaction scheme (hard constraints) that lead to non-empty results, and rank the objects of his focus according to preference (soft constraints). Recently, PFS has been used in various domains, e.g. for offering a flexible process for the identification of fish species [16], as well as a Voting Advice Application [15].
However the plain PFS does not exploit the geographical aspect(s) of the data. To tackle this requirement, in this paper we introduce `PFSgeo`, an extension of PFS appropriate for exploring datasets with objects that have *spatial coordinates*. Consider a small information base comprising information about hotel rooms where each room is described by various attributes (e.g. location, stars, room type, price, accessories, address, etc) including two spatial attributes, i.e. Longitude and Latitude. The Longitude-Latitude pair defines a point on the map. We adopt the coordinate system that is used by Google Maps (i.e. the WGS84 standard). For example, the way these two attributes are represented,

[4] http://data.ordnancesurvey.co.uk/
[5] http://data.geohive.ie/

as well as the coordinates of one hotel, are shown in the corresponding RDF in the TURTLE serialization format:

```
@prefix hippalus: <http://ics.forth.gr/isl/hippalus/#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@@prefix owl: <http://www.w3.org/2002/07/owl#> .
@@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
...
hippalus:Longitude a rdf:Property ;
rdfs:domain hippalus:Hippalus_Id ;
rdfs:range xsd:float .

hippalus:Latitude a rdf:Property ;
rdfs:domain hippalus:Hippalus_Id ;
rdfs:range xsd:float .
...
hippalus:Lato_Boutique_Hotel
a hippalus:hippalusID ;
hippalus:Latitude ''35.341751'' ^^xsd:float ;
hippalus:Longitude ''25.136610'' ^^xsd:float .
```

The extension of PFS for geographical data raised several issues and challenges including (a) how to come up with an intuitive user interface that can be used easily by casual users, (b) how to avoid cluttering the map with too many objects, and how to make evident the rank of these objects on the map, and (c) how to distribute the required computational tasks to the server-side and external sources (map APIs) for achieving good performance.

We introduce `PFSgeo` an extension of PFS that offers the ability (a) to show the corresponding objects on a map, and (b) to use the map for restricting the information space and/or expressing preferences (by selecting an area of the map and then issuing a statement). To grasp the idea, consider the simple scenario shown in Fig. 1 where the user has zoomed over the area of interest and has already expressed (through the facet exploration panel in the left) as most preferred the 2-star hotels amongst other actions. The updated hotels are shown not only as a list of textual entries in the upper half of the central part of the screen (where we can see three buckets of hotels based on the actions of this scenario), but also as markers on the map. The labels on the markers indicate the preference order, e.g. the label "Rank1" on a marker indicates the existence of a hotel in the first bucket, while the number on a cluster of markers (markerclusterer) indicates how many markers this cluster contains. PFS supports *preference inheritance in facets with hierarchically organized values and a scope-based method for* automatically resolving conflicts. The motivation is that it is very common (and practical) also in natural language to state something general and then to provide the exceptions. Indeed, this makes sense also in the geographic domain as it can offer flexibility which is evident also from the following example: Consider a user that prefers hotels in the broader area of Heraklion city but not in the area of the city center because it might be noisy there. To this end he could issue the following actions b1, b2:

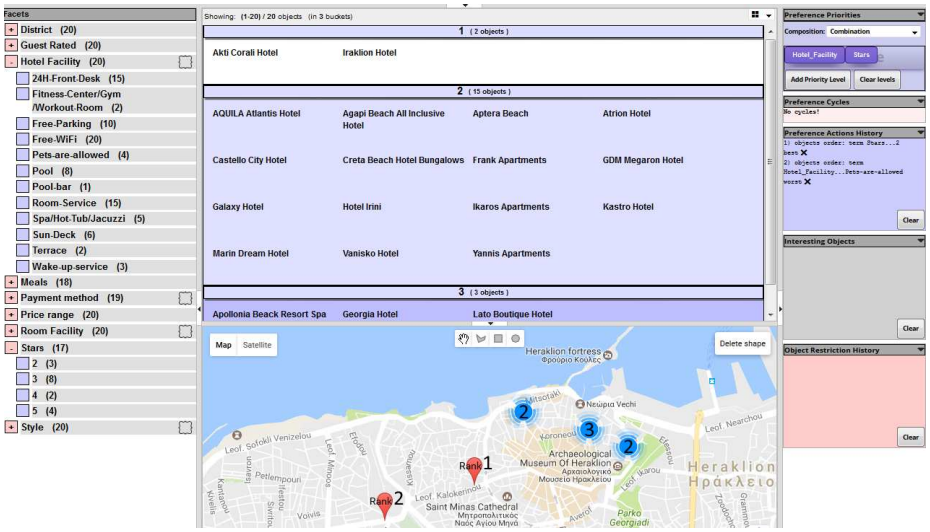



Fig. 1. A simple case scenario of exploration using both the facet exploration panel in the left and shapes over the map.

```
b1: BEST  x1,y1,x2,y2     // broader area of Heraklion city
b2: WORST x1',y1',x2',y2' // Heraklion city center
```

assuming that the first four coordinates capture the entire city of Heraklion, while those of $b2$ capture only a subarea (city center). Notice that the area of $b2$ is included in the area of $b1$, as shown in Fig. 2, therefore the produced ranking should have first all hotels of Heraklion except those in the subarea (rank 1), followed by those not in Heraklion (rank 2), and finally those in the city centre (rank 3). In general the user through left and right clicks can gradually formulate complex restrictions and rankings based on statements expressed over the facet values and the map (in any order). For example, the user can gradually (and without having to type anything) can express information needs like: *"hotels in Heraklion with price less than 90 Euros per night and free wifi while preferring (a) 2-star hotels, (b) the areas that I have marked on the map, (c) tennis than basketball facilities".* In a nutshell the key contributions of this paper are: (a) it is the first work about exploratory search with preferences over geographical data that supports preference inheritance and automatic resolution of preference conflicts, (b) it details an implementation of the model in the context of a publicly accessible prototype that supports PFS and uses Google Maps, (c) it reports experimental results regarding performance and scalability, and (d) it reports the results of a task-based evaluation with users.

The introduced approach can be applied over RDF data, over the results of SPARQL queries, or over flat-structured files after a transformation to RDF. The proposed interaction can be very useful in e-commerce applications and geo-related services (e.g. in tourism). The rest of the paper is organized as follows. At first we describe background and related work, then we describe the approach


Fig. 2. Auto conflict resolution of preferences over intersected areas.

and its implementation and finally we report experimental results about the performance and a task-based evaluation with users. Finally, we conclude and identify issues that are worth further research. A video demonstrating the PFS interaction model as implemented by the web system Hippalus (without maps) is available at `https://www.youtube.com/watch?v=Cah-z7KmlXc` while a video of PFSgeo (demonstrating PFS with maps) is available at `https://youtu.be/h7Ov5QXajYM`.

## 2 Related Work and Novelty

There are several works in the database world that provide a preference based ranking of spatial data, e.g. [22, 11, 1] study a number of algorithms and indexes for top-$k$ spatial preference queries that rank spatial objects based on qualities of features in their spatial neighbourhood. Other works like GEORank [3] provide a location-aware and temporal ranking of news according to user's current or a fixed location set in the user profile. In GeoRank, this location is static, hence they do not provide exploration services which is the focus of our work. Efforts such as 3cixty [14], deal with the process of building comprehensive knowledge bases that contain spatial data about events and activities of cities, collected from numerous local and global data providers.

As regards systems that support exploratory search, the system Facete [13] provides faceted search over data that contain geographic information. Mappify [8] is a web application to create map views displaying concept based points of interest. Providing faceted exploration capabilities, Mappify allows one to define custom concepts on a SPARQL endpoint. In comparison to Facete or Mappify, we do not use maps just for displaying objects; the user can use the map also for restricting the objects (by selecting the desired area) and also for defining his/her preferences (by selecting preferred or non preferred areas). Furthermore at any point the user is free to issue an action through the map or through the non geographical facets.

In brief, and to the best of our knowledge, `PFSgeo` is the first work about exploratory search with preferences and geographic information. It uses geographical maps not only for displaying the focus objects during the interaction (as in

previous works and systems), but also as an input method. Moreover it supports preference inheritance in the hierarchies and a scope-based method for automatically resolving conflicts in the geographic domain based on inference. Table 1, categorizes the aforementioned frameworks and commercial booking/real estate platforms (Booking.com, Airbnb, rightmove.co.uk) and PFSgeo, according to various aspects like Faceted restriction, Preferences, display of focus on the map, shape-based (map) restriction and preferences.

Table 1. Related approaches and comparison with `PFSgeo`.

| | Research Prototypes | | Commercial platforms | | | PFSgeo |
|---|---|---|---|---|---|---|
| **Feature** | *Facete* | *GEORank* | *Booking.com* | *Airbnb* | *Rightmove* | *PFSgeo* |
| **Faceted restrictions** | ✓ | - | ✓ | ✓ | ✓ | ✓ |
| **Preferences** | - | - | - | - | - | ✓ |
| **Display of focus on map** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Shape-based (map) restrictions** | - | - | - | - | ✓ | ✓ |
| **Shape-based (map) preferences** | - | - | - | - | - | ✓ |

# 3 The PFSgeo Approach

## 3.1 Extensions of the Language of PFS

Note that PFS offers actions that allow the user to order facets, values, and objects using *best, worst, preferTo* actions (i.e. relative preferences), *aroundTo* actions (over a specific value), or actions that order them lexicographically, or based on their values or count values. Furthermore, the user is able to *compose* object related preference actions, using *Priority, Pareto, Pareto Optimal* (i.e. skyline) and other. To realize `PFSgeo` we extended the language described in [18] with actions that have *spatial scope*. Based on the current implementation, the user's actions when interacting with the map is BEST and WORST, in contrast to the typical exploration paradigm of PFS that also supports other actions such as $preferTo$. The latter kind of actions would be difficult to be expressed over the map. We extended the PFS grammar with geo-anchored actions and we adapted accordingly the algorithms so that geo-anchored actions can be combined with actions over the rest facets. In the syntax "geo" denotes something with a spatial representation, e.g. rectangle, polygon, circle or point (our prototype focuses on rectangles for the time being). For example the extension of the *syntax* for expressing an action "BEST lat1, lon1, lat2, lon2" for a rectangle specified by the coordinates of its bottom-left and upper-right corners, expressed according to WGS84[6] standard, has the following form:

```
object order: geo rectangle [(lat1,lon1),(lat2,lon2)] best
```

## 3.2 Extensions of the System Hippalus

Then we extended Hippalus[7] which is a publicly accessible web system that implements the PFS interaction model. The information base that feeds Hippalus

[6] https://en.wikipedia.org/wiki/World_Geodetic_System
[7] `http://www.ics.forth.gr/isl/Hippalus/`

is represented in RDFS[8] (using a schema adequate for representing objects described according to dimensions with hierarchically organized values). We have implemented a tool called HDT (Hippalus Data Transformer) that transforms CSV files as well as the results of SELECT SPARQL results to an RDF file that is directly loadable by Hippalus. The values of an attribute can be hierarchically organized, and set-valued attributes are supported. The entire process is sketched in Fig. 3. For loading and querying such files, Hippalus uses Jena[9], a

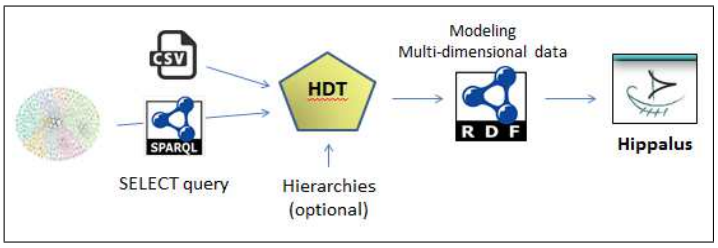



Fig. 3. The process for selecting and transforming data for being loadable by Hippalus.

framework for building Semantic Web applications. Hippalus offers a web interface for Faceted Search enriched with preference actions offered through HTML 5 context menus[10]. The performed actions are internally translated to statements of the preference language described in [18], and are then sent to the server through HTTP requests. The server analyzes them, using the language's parser, and checks their validity. If valid, they are passed to the appropriate preference algorithm. Finally, the respective preference bucket order is computed and the ranked list of objects according to preference, is sent to the user's browser.

For supporting `PFSgeo`, apart from extending the language, we extended Hippalus with Google Maps. We used libraries of Google Maps JavaScript API, specifically the *drawingManager Library*[11] which provides a graphical interface for drawing shapes such as rectangles, and the *Geometry Library*[11] that offers utility functions for computations involving polygons and polylines e.g. for getting the markers within a shape. Fig. 4 shows the new architecture that shows how Google Maps are involved as well as the extensions that were made at the server-side. For tackling cases where there are too many objects which are very close, the system implements the *MarkerClusterer v3*[12], which allows the display of multiple GEO locations as a labeled cluster. This cluster is basically a united marker indicating the number of the containing single markers within this area. The combined use of Google Map and MarkerClusterer allows the system to update the displayed results both on demand or on the fly. Both *restrictions*

[8] http://www.w3.org/TR/rdf-schema/
[9] http://jena.apache.org/
[10] Available only to firefox 8 and up.
[11] https://developers.google.com/maps/documentation/javascript/reference
[12] https://developers.google.com/maps/articles/toomanymarkers

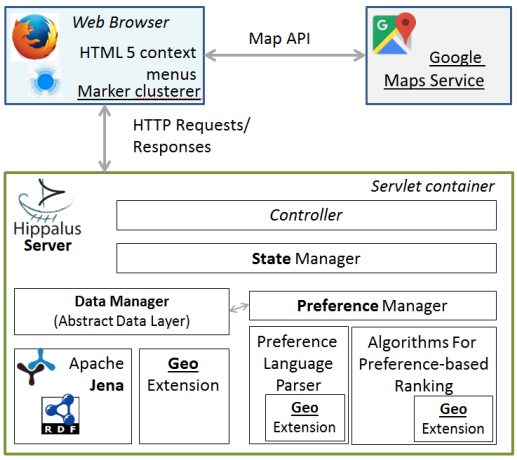



Fig. 4. Client - Server architecture of Hippalus with geo-extension.

and *preferences* over geographical data are possible through a pop-up menu, using the *right click* after a *shape* selection. As regards preferences, the user can issue either $BEST$ or $WORST$ preference actions. Each action is translated into a HTTP POST request method to Hippalus server using the JSON data-interchange format. This object contains the long-lat pair of the markers within this area, given by Google Maps JavaScript API V3.

### 3.3 Algorithmic Perspective

The *algorithmic perspective* concerns how restriction and preference actions are managed efficiently in case these actions have been issued through the map. We have identified two methods that we shall describe through an example. Without loss of generality, let's assume that the user selects an area in the map through a rectangular shape, defined by its coordinates as described previously and through right click he selects one option, say $BEST$.

**Method 1.** In this method at first (and at client-side, i.e. browser-side) we find the hotel markers that fall in selected area of the map (by issuing a call to Google Maps). Suppose that these markers correspond to a set of $k$ hotels denoted by $Hshape = \{h_1, \ldots, h_k\}$. Then the browser sends to the (Hippalus) server a set of preference actions, specifically $|Hshape|$ in number $BEST$ actions, i.e. the following set of actions: $\{\ BEST\ h \mid h \in Hshape\}$. According to Method 1 Longitude and Latitude are treated as ordinary facets from the server-side. As we shall see in the section with the experiments, Method 1 proved inefficient for preference actions issued through the map. The first reason is that a lot of data are sent to the server. The second reason is that the complexity of the algorithm for preference-based ranking (as proposed in [18]) that runs on the server side of Hippalus, has time complexity $O(|A||B|^2)$ where $A$ is the set of objects in the focus, and $B$ is the number of preference actions. The algorithm is also given below in Alg. 1, adapted to our context. It follows that by increasing the number of

actions (e.g. in our example although the user issued 1 $BEST$ action, the server received $|Hshape|$ in number $BEST$ actions) we actually affect negatively the performance of the algorithm. This is the motivation for Method 2.

**Method 2.** This method follows a different approach that is better aligned with the PFS and its algorithms. The key point is that instead of sending to the server the set of actions $\{\ BEST\ h \mid h \in Hshape\}$ we sent just one action: $BEST\ x1, y1, x2, y2$ and this is the motivation for extending the language of [18] with actions that have geographical anchors (as mentioned in Subsection 3.1). We can already see the benefits: less data have to be transferred and smaller $|B|$, thus faster production of the ranked list. It is also worth noting that the extension of the language with such actions does not require changing the core algorithms of [18] since that framework already captures actions with *scope* (for supporting preference inheritance in facets with hierarchically organized values). In our case the scope is spatial. Moreover, the scope-based method for automatically resolving conflicts makes sense also in the geographic domain as explained in the introductory section. Note that in the geographical context an action $b$ is narrower than a $b'$, denoted by $b \sqsubseteq b'$, if and only if the area of $b$ defined by $(x1, y1, x2, y2)$ is included in the area of $b'$ defined by $(x1', y1', x2', y2')$. Since we use the rectangle shape as a proof of concept, it is not hard to see that this holds if $(x1 \geq x1') \wedge (y1 \geq y1') \wedge (x2 \leq x2') \wedge (y2 \leq y2')$ and this is actually how $CheckSubScopeOf$ (of Alg. 1) has to be implemented for working over preference actions anchored to geographical areas. It follows that the geographical areas do not add any cost in the computation of $(B, \sqsubseteq)$.

The second part of the algorithm computes the *active scope* of each $b \in B$. In our case the active scope of an action $b$, is defined by excluding from the area of $b$ all areas that are narrower than $b$. To implement this part of the algorithm we need to adapt $IsInScope(e, b)$ for our case. This function should return True if $e$ belongs to the area of $b$. Obviously this can be decided very fast, since a point $(x, y)$ falls in the area defined by $(x1, y1, x2, y2)$ iff $(x1 \leq x \leq x2) \wedge (y1 \leq y \leq y2)$. Subsequently we use the active scopes, that we have just computed, for extending $B$ to a set $B'$, specifically for an action $b = BEST\ (x1, y1, x2, y2) \in B$, we add to $B'$ the following set of actions: $\{\ BEST\ h \mid h \in ActiveScope[b]\}$

The third part the algorithm just parses the set $B'$ in order to get the sets $Be$, $Wo$, i.e. collecting those hotels with $BEST$ and those with $WORST$ and then produces the ranked hotels by topological sorting over the graph that contains the following preference relationships $\{e \succeq e' \mid e \in Be, e' \in Wo\}$ This is all that is required for producing the preference-based ranking of hotels.

As noted, conflict resolution based on scope is supported. However, it is possible to have two actions, e.g. one BEST and one WORST, over two *overlapping* areas. In that case none of these actions is narrower than another. If there is not any object (hotel in our case) that falls in the intersection of these areas, then this conflict does not have any impact on ranking of objects. If however, one or more objects belong to the intersection of the overlapping areas, then we have a non-resolvable conflict. In that case, the algorithm prevents cyclic preferences and produces the bucket order up to the objects that have the conflict.

Another important (for scalability) characteristic of the algorithm is that it never computes the entire scope of any action, i.e. its scope in the entire information base. Instead, it checks whether the elements of the current focus $E$ belong to the scopes of issued actions. This means that its computational complexity does not depend on $|Obj|$, but on $E$ (where $E \subseteq Obj$) which we can assume that is not big because the user applies preferences after he has focused on a smaller set of objects. The same is true for restrictions performed through the map. Furthermore, it is not useful (for the user) to show on the map more than a few hundreds objects (since the markers would clutter the space). For this reason, only if the focus is less than a configurable threshold (say 100) it is useful to show the objects on the map and rank them according to the issued preference actions. The algorithms can handle this load since for producing the preference-based ranking for foci of even 1,000 objects takes less than 3 seconds.

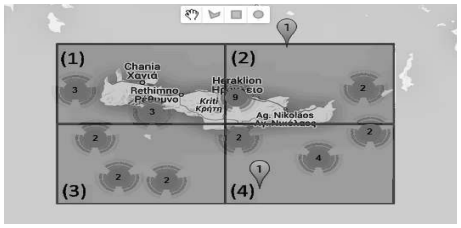
**Algorithm 1** AlgClearOpt($E$, $B$)
**Input:** the set of elements $E$, the set of actions $B$
**Output:** a bucket order over $E$

1: */** Part (1): Computation of $(B, \sqsubseteq)$ where $\sqsubseteq$ orders actions according to their scope*/*
2: $Visited \leftarrow \emptyset$
3: $R_{\sqsubseteq} \leftarrow \emptyset$ ▷ $R_{\sqsubseteq}$ corresponds to $\sqsubseteq$
4: **for** each $b \in B$ **do**
5: **for** each $b' \in B \setminus Visited$ **do**
6: **if** $CheckSubScopeOf(b, b')$ **then** ▷ if the scope of $b$ is included in the scope of $b'$
7: $R_{\sqsubseteq} \leftarrow R_{\sqsubseteq} \cup \{(b \sqsubseteq b')\}$
8: **else if** $CheckSubScopeOf(b', b)$ **then**
9: $R_{\sqsubseteq} \leftarrow R_{\sqsubseteq} \cup \{(b' \sqsubseteq b)\}$
10: **end if**
11: $Visited \leftarrow Visited \cup \{b\}$
12: **end for**
13: **end for**
14: */** Part (2): Efficient Computation of Active Scopes */*
15: **for** each $b \in B$ **do**
16: $C(b) \leftarrow$ direct children of $b$ wrt $R_{\sqsubseteq}$
17: ActiveScope[$b$] $\leftarrow \{e \in E \mid IsInScope(e, b) \wedge$
18: $(\forall c \in C(b)$ it holds $IsInScope(e, c) =$ False$)\}$
19: **end for**
20: */** Part (3): Derivation of the final bucket order */*
21: Use the active scopes to expand the set of actions $B$ to a set of actions $B'$
22: $(Be, Wo) \leftarrow$ Parse($B'$) ▷ $Be$ the best, $Wo$ the worst
23: Sort the elements of $E$ by applying topological sorting over the $Be$-$Wo$ graph.
24: **return** the produced bucket order of $E$

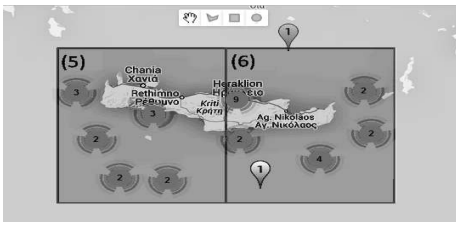
# 4 Performance Evaluation

## 4.1 Testing Scalability

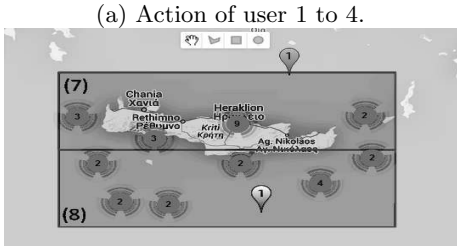
**Information Base.** We used a synthetically produced dataset with information about hotels in the region of the Crete island where each hotel is represented by various attributes (as described in the introductory section). Two main scenarios were designed each having the objective to measure the performance of the system over geo data. Each scenario is essentially a sequence of requests that simulates a user that interacts with the system using the map. To conduct the experiments, we used a laptop PC (dual-core 1.8 GHz CPU, 6 GB of main memory) that deployed PFSgeo locally. The **First Scenario** concerns *restrictions* using the map, i.e. it simulates a user that issues multi restriction actions using a shape. The **Second Scenario** concerns *preferences* using the map, i.e. it simulates a user that issues preference actions using a shape. We used the following datasets: a) Dataset 1 containing 1,000 hotels, b) Dataset 2 containing 5,000 hotels and c) Dataset 3 containing 20,000 hotels.



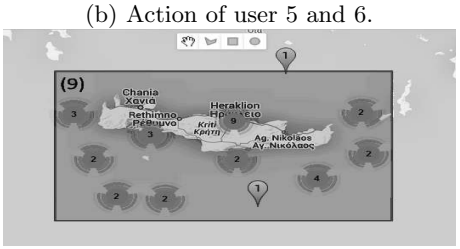

(a) Action of user 1 to 4.




(b) Action of user 5 and 6.




(c) Action of user 7 and 8.




(d) Action of user 9.

Fig. 5. The simulated different user's actions for preferences and restrictions.

**Simulation.** The simulation platform considers the area and the information base described previously. In order to evaluate the system's performance we apply the scenarios on predefined shapes on the map that include a set of markers. The number of markers differs depending on the size of the dataset. In each of these two scenarios we issue 10 requests from 9 different (simulated) users. For this reason we have created 9 different non-intersecting shapes within the boundaries of the simulation area, as shown in Fig. 5. The division of the area ensures that any possible subarea will contain markers, while the sequence simulates a user that issues actions with gradually bigger shapes, meaning that the

corresponding load becomes heavier since more instances have to be passed as parameters. Note that in shape 9 the user issues an action over all instances of the dataset. Each shape is represented by the action of the respective user accordingly. We should mention that a restriction action based on intersecting shapes, say a shape $A$ and a shape $B$, eventually has the same outcome as the restriction action using as shape the intersection $A \cap B$.

### 4.2 Analysis of the Performance

**Standalone Experiment Results.** The analysis aims to evaluate the performance by measuring the delay that a user experiences (note that the delay while exploring a geo dataset should not overcome the user's tolerance [7]). To this end, we introduce the following metrics:

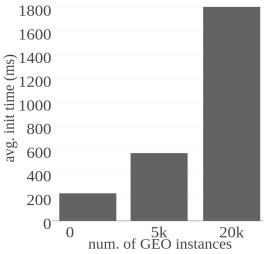
- **initTime:** this is the time the user experiences for loading the instances on startup and for displaying them on the map.
- **restriction time-*1st scenario:*** this is the total response delay for restricting the focus using a multi *restriction query*.
- **preference time-*2nd scenario:*** this is the total response delay when the user issues a *preference query* using a shape.

Based on the current implementation these metrics do not concern only the management of the geographic information, in the sense that during the exploration of a spatial dataset, *initTime*, *restriction time* and *preference time* are affected not only by the geo aspect but also by the various delays of Hippalus. Specifically, both **restriction time** and **preference time** include the time required for *"updateHierarchy"* and *"updateHistory"* which is the total time required to update the facet-values and history displays, as well as, *"getGeoLocations"* and *"updateInstances"* which is the total required time to get the locations from Hippalus server and the time to create-update the markers on the map and update the instances in buckets. The reported times include the time required for rendering the map using third party APIs calls, i.e. Google Maps.

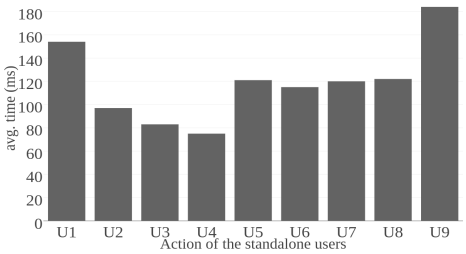
As regards the results, as we can see in Fig. 6a the average **init time** for datasets with 20,000 instances is 1.8 seconds. Figures 6b, 6c and 6d show the results of the **restrictions** for Dataset 1, Dataset 2 and Dataset 3.

### 4.3 Comparative Experimental Results

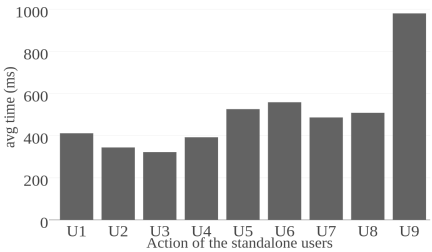
Here we report experimental results for comparing Method 1 with Method 2. Notice that for more than 1,000 instances the system with Method 2 has an average preference time of 2.8 sec in contrast to the 98.4 sec of the system with Method 1, i.e. this method is one order of magnitude faster than the previous one (specifically 35 times faster). To this respect, Fig. 7a presents a comparison of those 2 methods regarding the preference action of the users, for the Dataset 1. Finally, Fig. 7b shows the average query time for 10 requests that get served by the system for each of the three metrics for all datasets, where Table 2 shows the results of the preference action of Method 2 compared to Method 1. Regarding the Method 1 note that for datasets 2 and 3 the system could not respond and terminated with a timeout error.



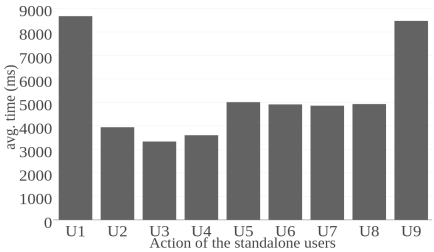

(a) Average **init time** over the *3 datasets*.




(b) Average **restriction times** of *Dataset 1* for each standalone user.

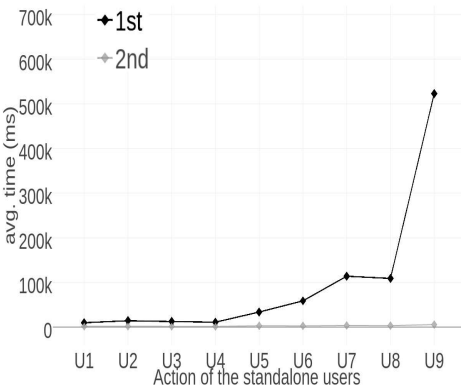



(c) Average **restriction times** of *Dataset 2* for each standalone user.

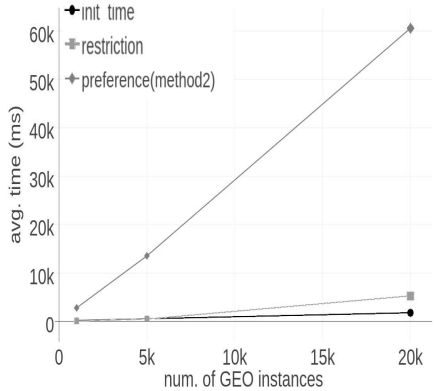



(d) Average **restriction times** of *Dataset 3* for each standalone user.

Fig. 6. **Init times** and **restriction times using Method 1**.




(a) Average **preference** times of *Dataset 1* for each standalone user*(Method 1 compared to Method 2)*.




(b) Average times for the *3 metrics* of *3 datasets* for all users.

Fig. 7

Table 2. Average **preference** times of the simulated users over the *3 datasets* (*Method 1* comparing to *Method 2*).

| | | | **Method 1** | **Method 2** | | |
|---|---|---|---|---|---|---|
| *Actions* | *Dataset 1* | *Dataset 2* | *Dataset 3* | *Dataset 1* | *Dataset 2* | *Dataset 3* |
| User1 | 9,794 ms | (timeout) | (timeout) | 1,880 ms | 7,868 ms | 14,792 ms |
| User2 | 14,265 ms | (timeout) | (timeout) | 1,925 ms | 8,343 ms | 51,608 ms |
| User3 | 12,469 ms | (timeout) | (timeout) | 1,637 ms | 8,932 ms | 51,403 ms |
| User4 | 11,018 ms | (timeout) | (timeout) | 1,484 ms | 8,569 ms | 53,232 ms |
| User5 | 33,871 ms | (timeout) | (timeout) | 3,007 ms | 14,424 ms | 2,396 ms |
| User6 | 58,785 ms | (timeout) | (timeout) | 2,774 ms | 15,211 ms | 82,511 ms |
| User7 | 114,007 ms | (timeout) | (timeout) | 3,709 ms | 15,484 ms | 47,586 ms |
| User8 | 108,983 ms | (timeout) | (timeout) | 3,240 ms | 15,684 ms | 103,110 ms |
| User9 | 522,978 ms | (timeout) | (timeout) | 5,682 ms | 27,535 ms | 137,783 ms |

## 5 Evaluation with Users

Although Hippalus has been evaluated positively by users in various contexts (the interested reader can refer to [16, 15]), since the extension with maps is novel, we conducted a comparative task-based evaluation in which users would have to explore and express their preferences and/or restrictions using both UIs (with and without maps). We prepared some tasks and asked the participants to carry them out. The tasks are based on the following general scenario: *You are planning to stay in a hotel either for vacations or for a business trip in a specific area. You want to discover hotels in that area, however you do not have any a-priori knowledge about the location names in that area. You have at your disposal a set of restrictions (e.g. price, stars, facilities) but you cannot apply a restriction based on location name because you do not have such knowledge.*

### 5.1 Preparation

**Purpose**. Our objective was to investigate whether even in a small dataset, with only 20 hotels, the map extension would make search more effective and users more satisfied with the interaction. We distinguish two different UIs:

a Classical model of PFS interaction model UI1
b PFS exploration using the map both as visualization and input for the set of the users actions UI2

The different UIs can be selected by the user using the display options menu as shown in the video tutorial. The default view is "Text and Map" which displays the information space both as textual and map-based representation. The evaluation performed for both UIs in order to extract the users' satisfaction level and exploit any possible problems and difficulties.

**Training.** We created a Google form and we invited all the candidate participants to fill it in. Initially, the users were asked to watch two video tutorials: one demonstrating Hippalus without maps (the video is available at `https://www.youtube.com/watch?v=Cah-z7KmlXc`) and one demonstrating Hippalus with maps (the video is available at `https://youtu.be/h7Ov5QXajYM`).

**Participants.** According to [20, 6, 10] Jakob Nielsen proposed that for a single

usability test, 10 participants are enough for revealing severe usability problems. However since we are not interested only in usability problems we decided to involve more users. In total 20 persons participated, (75% male and 25% female), with various age groups (20% were between the ages of 20 and 25, 65% were between the ages of 25 and 30, 5% were between the ages of 30 and 35, while 10% were 35 and over) and various professions (65% CS-related employees, 20% researchers and the rest 15% were other). The evaluation started on March 10 and ended on April 6 of 2017.

**Tasks.** We asked the users to explore a dataset of Hotels (containing various information as in Booking.com) first without map and then with map, and carry out a sequence of basic tasks, including restrictions and expressions of preferences, as shown in Table 3. Note that we have 4 different tasks for each of the two different UIs *(IDs a,b)* which eventually gives a total number of 8. In each task the user had to provide the name of a single hotel (for tasks 2 and 3), two hotels (for tasks 1 and 4) or "none" (if no answer can be given) as possible answers. For example, for the task *find the hotels that are closer to the sea* it makes sense to assume that this could be a subjective issue since the statement "closer" in terms of distance may differ according to user's perspective. To this respect, we count every hotel name as correct whereas we only deemed the "none" answer as mistaken, since the objective of this evaluation was to evaluate the usability of the system, using the *System Usability Score* ***(SUS)*** [2]. For this reason, we asked the users at the end to rate each system using the following scale ⟨Useless, Not Useful, Neutral, Useful, Very Useful ⟩, and state which of these they prefer.

**Results.** As shown in Table 4, 98.75% of the participants managed to find a hotel using the `PFSgeo`, in contrast to 78.75% of plain PFS. In the overall rating, shown in Fig. 9, we can see that 50% of the participants rated the system Very Useful and 40% Useful, hence in total 90% of the participants were positive. Note that only 15% of the participants rated the PFS Hippalus Very Useful. It is also interesting to note that that 85% of the participants stated that they prefer the `PFSgeo` over the PFS Hippalus (Fig. 9b). In terms of statistical significance, the exact (Clopper-Pearson) 95% Confidence Interval is (62%, 97%) indicating that the PFSgeo is strongly preferred by the users over the Hippalus.

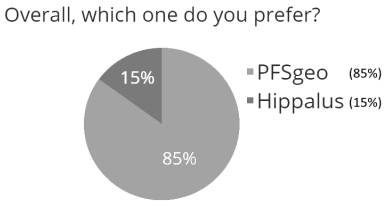


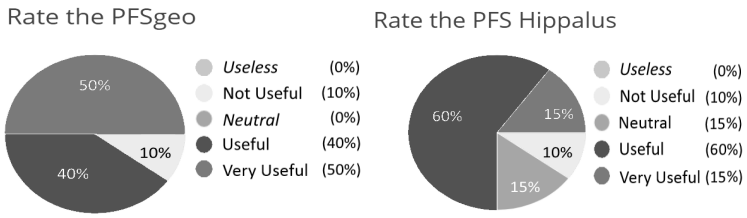

Fig. 8. `PFSgeo` vs Hippalus.

Table 4. Number of wrong answers in the evaluation with users.

| *Task Id* | *PFSgeo* | *PFS* |
|---|---|---|
| 1 | 0/20 | 5/20 |
| 2 | 1/20 | 4/20 |
| 3 | 0/20 | 2/20 |
| 4 | 0/20 | 6/20 |

Table 3. Tasks Description for user evaluation over the two UIs.

| *ID* | *Task Description* |
|---|---|
| a1, b1 | You are interested in "3-star" Hotels in "Heraklion", and you prefer these which are close to the sea. Browse the system and give the 2 hotels that you prefer most. |
| a2, b2 | You would like to visit the "Archaeological Museum of Heraklion" (located in the center of the city). Find the hotel that is closer to this museum and prefer the most economical. |
| a3, b3 | You would like to find a hotel in Heraklion with Price range "49-99" as BEST but you do not prefer those close to the sea. Use the system and provide one such hotel. |
| a4, b4 | You are interested in "3-star" Hotels in Crete but you do not like hotels located in cities (Heraklion) unless they are close to a sightseeing (Archaeological Museum). Use the system and return the two more preferred hotels. |




(a) Overall rating of `PFSgeo`. (b) Overall rating of Hippalus.

Fig. 9. Overall ratings of `PFSgeo` and Hippalus.

## 6 Concluding Remarks

In this paper we showed how a model for exploratory search, the Preference-enriched Faceted Search (PFS), can be enriched for being appropriate for exploring datasets that contain geographical information. The extended model, that we call `PFSgeo`, uses geographical maps not only for displaying the objects of the focus during the interaction (as in previous works and systems), but also as an input means for restricting the focus and for defining preferences. We should also stress that the proposed framework supports preference inheritance in the hierarchies and a scope-based method for automatically resolving conflicts. Subsequently we presented an implementation of the proposed model as an extension of the system Hippalus. Since there is not any system that supports PFS over datasets with geographic information the challenging task is how to support this interaction for big datasets. For foci that contain large amounts of objects (which is expected in the context of exploratory search), we support marker clustering to avoid cluttering the map. Finally we elaborated on performance issues and we provided measurements over synthetically produced datasets about hotels.

Based on this analysis, we identified those tasks that affect the scalability of the approach, and proposed methods that can be used for improving the efficiency of the approach. The user evaluation highlighted that when exploring datasets with geo aspect the functionality of the map is a very promising direction regarding the usability of the system. Overall 90% of the participants rated `PFSgeo` very positive (50% as Very Useful and 40% as Useful) and 98.75% of the total tasks were completed successfully.

Since this is the first work on extending Preference-enriched Faceted Search for geographical data, there are several directions that are worth further research. For instance, the current implementation supports only one geographical facet, however more than one are required for various kinds of objects (e.g. flights have two geographical facets "from" and "to"). Moreover, since a geographical area is continuous it is worth investigating also radius-based preferences, for instance if the user selects an point/area and issues a BEST action, then all objects could be ranked according to their distance to the point/area (e.g. show me all the restaurants within 20 min driving time around me). For achieving performance over very big data sets, one could investigate enriching the server-side with a spatial index (e.g. R-Tree, Quadtree, etc.), or adopting a triplestore that supports GeoSPARQL. Moreover it is worth investigating extensions of Triple Pattern Fragments for the geospatial Linked Data (like [5]). For enhancing applicability it is worth supporting more standards for representing coordinates and geometries. Finally, the interaction model assumes "single entity type objects" (according to the taxonomy in [17]), therefore it could be extended for objects of multiple entity-types. The system Hippalus is publicly accessible through `http://www.ics.forth.gr/isl/Hippalus/` (requires Firefox version 8 or higher).

## References


1. de Almeida, J.P.D., Rocha-Junior, J.B.: Top-k spatial keyword preference query. Journal of Information and Data Management 6(3), 162 (2016)
2. Bangor, A., Kortum, P., Miller, J.: Determining what individual sus scores mean: Adding an adjective rating scale. Journal of usability studies 4(3), 114–123 (2009)
3. Bao, J., Mokbel, M.F.: Georank: An efficient location-aware news feed ranking system. In: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 184–193. ACM (2013)
4. Bereta, K., Koubarakis, M.: Ontop of geospatial databases. In: International Semantic Web Conference. pp. 37–52. Springer (2016)
5. Debruyne, C., Clinton, É., OSullivan, D.: Client-side processing of geosparql functions with triple pattern fragments. In: Proceedings of the Workshop on Linked Data on the Web, LDOW (2017)
6. Faulkner, L.: Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. Behavior Research Methods 35(3), 379–383 (2003)
7. Galletta, D.F., Henry, R., McCoy, S., Polak, P.: Web site delays: How tolerant are users? Journal of the Association for Information Systems 5(1), 1–28 (2004)
8. Lehmann, J., Athanasiou, S., Both, A., Garcia Rojas M., A., Giannopoulos, G., Hladky, D., Le Grange, J.J., Ngonga Ngomo, A.C., Sherif, M., Stadler, C., Wauer,

M., Westphal, P., Zaslawski, V.: Managing geospatial linked data in the geoknow project (01 2015)
9. Lopez-Pellicer, F.J., Silva, M.J., Chaves, M., Zarazaga-Soria, F.J., Muro-Medrano, P.R.: Geo linked data. In: Database and Expert Systems Applications. pp. 495–502. Springer (2010)
10. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 249–256. ACM (1990)
11. Rocha-Junior, J.B., Vlachou, A., Doulkeridis, C., Nørvåg, K.: Efficient processing of top-k spatial preference queries. Proceedings of the VLDB Endowment 4(2), 93–104 (2010)
12. Sacco, G.M., Tzitzikas, Y.: Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience. Springer Publishing Company, Incorporated, 1st edn. (2009)
13. Stadler, C., Martin, M., Auer, S.: Exploring the web of spatial data with facete. In: Proceedings of the companion publication of the 23rd international conference on World wide web companion. pp. 175–178. International World Wide Web Conferences Steering Committee (2014)
14. Troncy, R., Rizzo, G., Jameson, A., Corcho, O., Plu, J., Palumbo, E., Hermida, J.C.B., Spirescu, A., Kuhn, K.D., Barbu, C., Rossi, M., Celino, I., Agarwal, R., Scanu, C., Valla, M., Haaker, T.: 3cixty: Building comprehensive knowledge bases for city exploration. Web Semantics: Science, Services and Agents on the World Wide Web (2017), http://www.sciencedirect.com/science/article/pii/S1570826817300318
15. Tzitzikas, Y., Dimitrakis, E.: Preference-enriched faceted search for voting aid applications. IEEE Transactions on Emerging Topics in Computing PP(99), 1–1 (2016)
16. Tzitzikas, Y., Bailly, N., Papadakos, P., Minadakis, N., Nikitakis, G.: Using preference-enriched faceted search for species identification. International Journal of Metadata, Semantics and Ontologies 11(3), 165–179 (2016)
17. Tzitzikas, Y., Manolis, N., Papadakos, P.: Faceted exploration of RDF/S datasets: a survey. Journal of Intelligent Information Systems (2016)
18. Tzitzikas, Y., Papadakos, P.: Interactive exploration of multidimensional and hierarchical information spaces with real-time preference elicitation. Fundamenta Informaticae 20, 1–42 (2012)
19. Vatant, B., Wick, M.: Geonames ontology (2006). Online at http://www.geonames.org/ontology (2006)
20. Virzi, R.A.: Refining the test phase of usability evaluation: How many subjects is enough? Human Factors: The Journal of the Human Factors and Ergonomics Society 34(4), 457–468 (1992)
21. Vockner, B., Mittlböck, M.: Geo-enrichment and semantic enhancement of metadata sets to augment discovery in geoportals. ISPRS International Journal of Geo-Information 3(1), 345–367 (2014)
22. Yiu, M.L., Lu, H., Mamoulis, N., Vaitis, M.: Ranking spatial data by quality preferences. Knowledge and Data Engineering, IEEE Transactions on 23(3), 433–446 (2011)