A Preference-enriched Faceted Search for Geographical Data

Panagiotis Lionakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes Campus, GR-70013 Heraklion, Crete, Greece

Thesis Advisor: Associate Prof. *Yannis Tzitzikas*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.
The work has been supported by the Foundation for Research and Technology – Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

A Preference-enriched Faceted Search for Geographical Data

Thesis submitted by
Panagiotis Lionakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author:	Panagiotis Lionakis
Committee approvals:	Yannis Tzitzikas Associate Professor, Department of Computer Science University of Crete, Thesis Supervisor
	Dimitris Plexousakis Professor, Department of Computer Science University of Crete, Committee Member
	Konstantinos Stefanidis Assistant Professor, Faculty of Natural Sciences Tampere University, Committee Member
Departmental approval:	Antonis Argyros Professor, Department of Computer Science University of Crete, Director of Graduate Studies

Heraklion, February 2017

Abstract

A plethora of datasets contain geographic information or can be linked to geographic information. In this thesis, we show how an exploratory search process, specifically the Preference-enriched Faceted Search (PFS) process, can be enriched for being appropriate for exploring datasets that contain GEOgraphic information. In brief, 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 his focus gradually while getting an overview of the information space), while the latter corresponds to actions that specify preferences that rank accordingly the information space. In this thesis we extend PFS for advancing the exploration services for datasets that contain also geographic information. In the extended model, the user can use the map not only for inspecting the current focus, but also for restricting the focus, as well as for expressing preferences. Subsequently we describe an implementation of the approach by extending the system Hippalus and a pilot application of the resulting system. Finally, we apply and evaluate the proposed approach using a synthetic dataset containing information about Hotels in Crete, and we discuss issues related to the scalability of the approach. To the best of your knowledge, this is the first system that supports exploratory search with preferences over datasets that contain also geographic information.

Μια εκτεταμένη με προτιμήσεις Πολυεδρική Αναζήτηση για Γεωγραφικά Δεδομένα

Περίληψη

Υπάρχει σήμερα ένας μεγάλος αριθμός από σύνολα δεδομένων που περιέχουν γεωγραφική πληροφορία ή έχουν τη δυνατότητα να συνδεθούν με γεωγραφική πληροφορία. Σε αυτήν την εργασία, δείχνουμε πως ένα μοντέλο εξερευνητικής αναζήτησης, συγκεκριμένα το Εκτεταμένο με Προτιμήσεις μοντέλο της Πολυεδρικής Αναζήτησης (PFS), μπορεί να εμπλουτιστεί ώστε να προσφέρεται για εξερεύνηση συνόλων δεδομένων που περιέχουν γεωγραφική πληροφορία. Εν συντομία, το Εκτεταμένο με Προτιμήσεις μοντέλο της Πολυεδρικής Αναζήτησης επιτρέπει στο χρήστη να περιορίσει το επίκεντρό του σταδιακά ενώ εποπτεύει τον πληροφοριακό χώρο και συνάμα του προσφέρει ενέργειες που του επιτρέπουν να ορίσει προτιμήσεις οι οποίες ανακατατάσσουν αναλόγως τον πληροφοριακό χώρο. Στην παρούσα εργασία επεκτείνουμε αυτό το μοντέλο ώστε να είναι κατάλληλο για σύνολα δεδομένων που περιέχουν και γεωγραφική πληροφορία. Ο χρήστης μπορεί να χρησιμοποιήσει το χάρτη όχι μόνο για να εποπτεύσει τα δεδομένα αλλά και για να περιορίσει το επίκεντρό του ή για να εκφράσει προτιμήσεις. Στη συνέχεια, περιγράφουμε μια υλοποίηση της προσέγγισής μας επεκτείνοντας το σύστημα Hippalus και μια πιλοτική εφαρμογή. Εν συνεχεία εφαρμόζουμε και αξιολογούμε την προτεινόμενη προσέγγιση σε μια συνθετική βάση δεδομένων με πληροφορίες για τα ξενοδοχεία στην Κρήτη και περιγράφουμε θέματα που αφορούν την κλιμακωσιμότητα του συστήματος. Το παρόν σύστημα είναι το πρώτο που στηρίζει εξερευνητική αναζήτηση με προτιμήσεις επί δεδομένων που περιέχουν και γεωγραφική πληροφορία.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επόπτη καθηγητή μου κ. Γιάννη Τζίτζικα, για τη νουθέτηση και την εμπιστοσύνη που μου έδειξε, όσο και για την άψογη συνεργασία, την ορθή καθοδήγηση και την ουσιαστική συμβολή του στην ολοκλήρωση της παρούσας μεταπτυχιακής εργασίας. Ακόμη θα ήθελα να εκφράσω τις ευχαριστίες μου στον κ. Δημήτρη Πλεξουσάκη και τον κ. Κώστα Στεφανίδη για την προθυμία τους να συμμετάσχουν στην τριμελή επιτροπή.

Ακόμα ευχαριστώ το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υποτροφία που μου προσέφερε, καθώς και για την πολύτιμη υποστήριξη σε υλικοτεχνική υποδομή και τεχνογνωσία ως ένας φορέας που προάγει και προωθεί την επιστήμη τόσο σε θεωρητικό υπόβαθρο όσο και ως ουραγός εξωστρέφειας αποτελώντας ένα τεχνολογικό φάρο διεθνούς επιπέδου.

Στο σημείο αυτό, θα ήθελα να ευχαριστήσω ιδιαιτέρως τους γονείς μου, για τη συμπαράσταση και την υποστήριξη που μου προσέφεραν καθ΄ όλη τη διάρκεια των σπουδών μου. Έπειτα του συνεργάτες από το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας που ήταν πάντα πρόθυμοι να ρίξουν τους προβολείς των γνώσεων τους επάνω μου. Τέλος, τους φίλους μου για την υποστήριξή τους όλα αυτά τα χρόνια.

στους γονείς μου

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

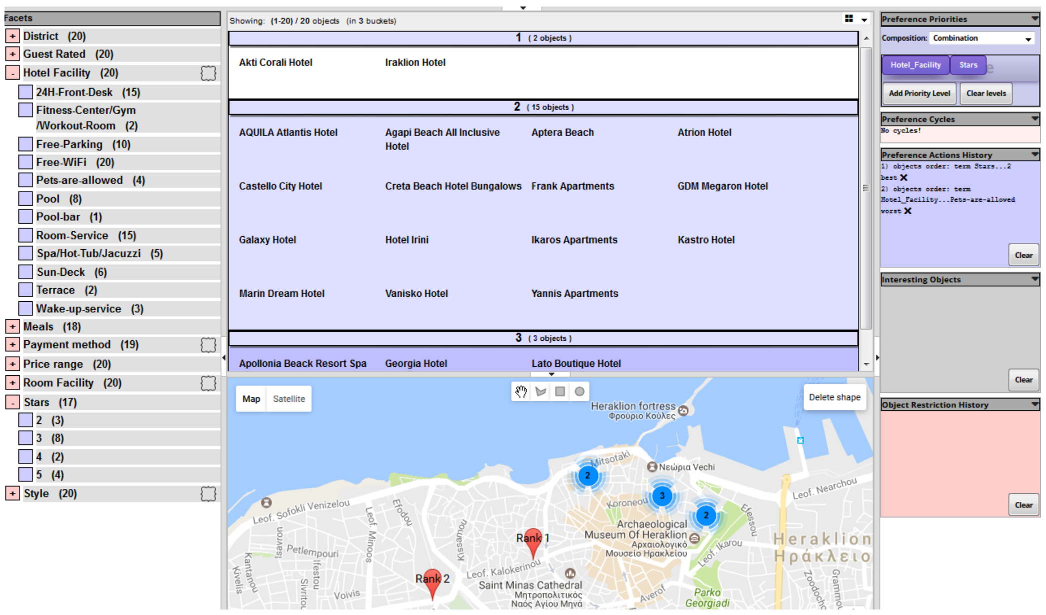
Introduction

A plethora of datasets contain geographic information or can be linked to geographic information. There are several approaches that combine Linked Data[1] and spatial data. For example, LinkedGeoData [1] is an effort to add a spatial dimension to the Web of Data / Semantic Web. LinkedGeoData uses the information collected by the OpenStreetMap project and makes it available as an RDF according to the Linked Data principles [2]. Similar to this, GeoLinkedData [3] is an 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 [4].

In this thesis we propose how an exploratory search process, specifically the *Preference-enriched Faceted Search* (PFS) process, proposed in [5], can be enriched for being appropriate for exploring datasets that contain geographic information. In brief, PFS supports user clicks that correspond to either hard and soft constraints. The first kind corresponds to the actions of the classical faceted search where the user can restrict 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 3-stars is the most preferred category, and this statement will *rank first* the 3-stars hotels but will not vanish the rest). In this thesis we extend PFS for advancing the exploration services that is provided for datasets that also contain geographic information. In brief, this requires the ability to show the corresponding objects on a map and the ability to use the map for restricting the

[1]http://lod-cloud.net/

information space and/or expressing preferences (by selecting an area of the map and then issuing a statement). To this end, we describe the required extensions of the interaction model and we detail the GUI that we have designed. Then we describe an implementation of the approach in the system Hippalus through which we demonstrate the interaction.



Showing: (1-20) / 20 objects (in 3 buckets)

1 (2 objects)
Akti Corali Hotel, Iraklion Hotel

2 (15 objects)
AQUILA Atlantis Hotel, Agapi Beach All Inclusive Hotel, Aptera Beach, Atrion Hotel, Castello City Hotel, Creta Beach Hotel Bungalows, Frank Apartments, GDM Megaron Hotel, Galaxy Hotel, Hotel Irini, Ikaros Apartments, Kastro Hotel, Marin Dream Hotel, Vanisko Hotel, Yannis Apartments

3 (3 objects)
Apollonia Beack Resort Spa, Georgia Hotel, Lato Boutique Hotel

Composition: Combination

Preference Cycles
No cycles!

Preference Actions History
1) objects order: term Stars...2 best ✗
2) objects order: term Hotel_Facility...Pets-are-allowed worst ✗

Figure 1.1: A simple case scenario of exploration using both the facet exploration panel in the left and shapes over the map.

To grasp the idea, consider the simple scenario shown in Figure 1.1 where the user has zoomed over the area of interest and has already expressed (though the facet exploration panel in the left) as most preferred the 3-star hotels. The updated hotels are shown not only the textual list (where we can see three buckets of hotels), but also on the map. The labels on the markers indicate the preference order i.e. the marker "Rank1" indicates the existence of a hotel in the first bucket, i.e. in the bucket of the most preferred hotels (note that in early implementations the label is referred only with a number e.g. "1" instead of "Rank1"). The user is then able to continue and issue actions using the shapes for narrowing the exploration results or expressing new preference actions over the updated results, and so on.

Finally we discuss issues related to the scalability of the approach. The value of the extended PFS model 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.

In a nutshell the key contributions of this thesis are:

- To the best of our knowledge, it is the first work about exploratory search with preferences and geographic information
- We show how the model can be implemented and we detail an implementation of the model by extending a publicly accessible system (Hippalus) and exploiting Google Map.
- We report experimental results regarding performance and then we describe how the model can be implemented for being applicable on big datasets
- Introduce a map based visualized categorization of spatial data for PFS

The extension of PFS for geographical data raised several issues including

a how to come up with an intuitive user interface that can be used easily by casual users,

b how to divide the required computational tasks to the Hippalus server and the map API for achieving good performance,

c how to avoid cluttering the space with too many objects,

Since there is not any system that supports faceted search over datasets with geographic information and uses the map for restrictions and preferences, the challenging task is how to support this interaction for big datasets. We should also stress that the proposed framework supports preference inheritance in the hierarchies and a scope-based method for automatically resolving conflicts in the geographic domain that is supported based on inference.

1.1 Outline of Thesis

The rest of this thesis is organized as follows:

Chapter 2 discusses the context and describes related work and what distinguishes the current one in contrast to the known literature.

Chapter 3 presents the proposed geo extension of PFS, that we call PFS-geo, describes the interaction model through running examples and provides architectural and applicability details.

Chapter 4 evaluates the performance of the system, as regards metrics for different actions using simulated users. Then, it reports and analyzes the experimental results over 3 large scale synthetic datasets

In Chapter 4.2 we propose an alternative method for managing efficiency geographical preferences and present comparative experimental results and implementation details for this stable prototype.

Finally, Chapters 5 and 6 concludes and identifies directions for future research, while Appendix A gives an overview of the RDF/S files for a hotels' dataset and the Syntax of GEO extended Preference Language.

A prototype is already publicly accessible and available online[2].

[2]http://www.ics.forth.gr/isl/Hippalus/

Chapter 2

Context and Related Work

Section 2.1 describes geographic data and linked data. Section 2.2.1 describes approaches for exploring such combinations of data and overviews the placement of the proposed system in contrast to the known literature. Section 2.2.2.1 provides the required background information on Faceted and Dynamic Taxonomies. Section 2.2.2.2 and 2.2.2.3 introduces our approach and describes in brief Preference-enriched Faceted Search and the system Hippalus.

2.1 Background

Geographic Data.

A plethora of datasets contain geographic information or can be linked to GEOgraphic information. In the last few years, there has been significant effort of publishing geospatial data using RDF and Linked Data (LD) principles. To this respect, although various LD management systems have been proposed, only a few of them are able to handle geospatial data [6]. In addition, Great Britain's national mapping agency, Ordnance Survey[1], has been the first national mapping agency committed to making publicly available various kinds of geospatial data from Great Britain as open linked data. Other works like, GeoTriples [7] deals with the transformation of geospatial information into RDF graphs in a semiautomated manner while GeoNames [4] provide a geographical database available and accessible with over 10 million geographical names. Other efforts include various deployments of the map4rdf [8] which allow visualising and interacting with Linked Geospatial Data available in various SPARQL endpoints.

[1]http://data.ordnancesurvey.co.uk/

To our knowledge, spatial data, also known as geospatial data or geographic data, traditionally represents a geo location, as a projection on map. These entities or points, include attributes such as coordinates (i.e. latitude-longitude pairs) and topology, in order to be mapped and further processed, analyzed and visualized through Geographic Information Systems (GIS).

Quite commonly one has to manage Linked Data and geographic data. DBpedia approaches this issues by providing various ontology classes, such as Place, an an ability to represent coordinates. Besides coordinates, DBpedia contains additional geo-coordinates for more than 1 million geographic locations, expressed using the W3C Basic Geo Vocabulary[2]. Similarly, in Europeana Data Model (EDM), spatial information can be represented and visualized by tools like as Europeana 4D.

As regards querying, GEOSPARQL [9] is a standard for querying geospatial linked data for the Semantic Web. It provides a topological ontology in RDFS/OWL that uses GML (Geography Markup Language) and Well Known Text (WKT) literals and supports simple features topological relationship vocabularies and ontologies for qualitative reasoning. In addition it provides a SPARQL querying interface that uses a set of topological SPARQL extensions for quantitative reasoning and a set of Rule Interchange Format (RIF) inference rules for query transformation and interpretation.

2.2 Related Work

2.2.1 Approaches for Exploring Datasets that contain Linked Data and Geographic Data

There are a number of works in the database world that provide a preference based ranking of spatial data. For example works like [10], [11] 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 [12] 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.

As regards systems that support exploratory search, there is the system Facete [13] that provides faceted search over data that contain geographic information. Facete implements a spatial data exploration paradigm based on the following three key components: First, a domain independent faceted

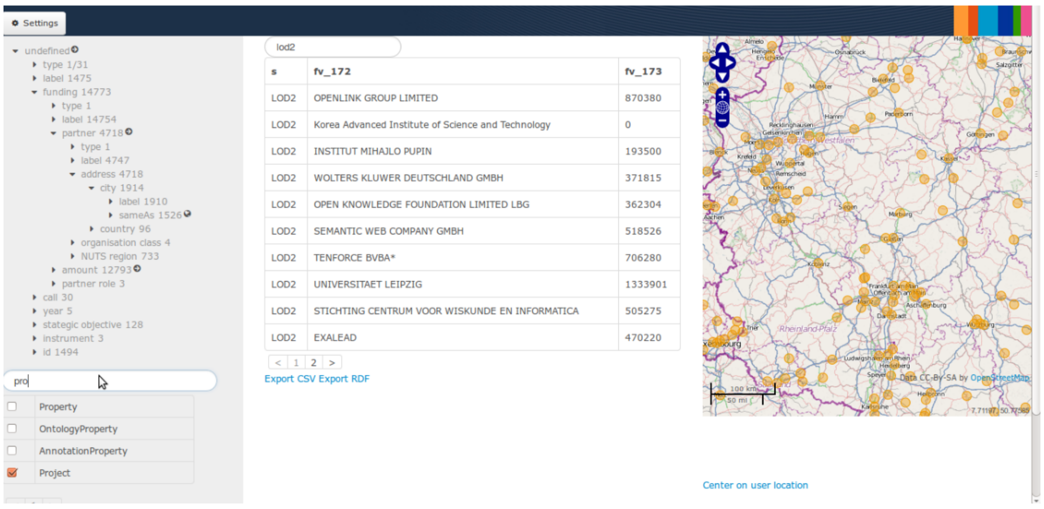
[2]http://wiki.dbpedia.org/services-resources/dbpedia-data-set-2014#4.5

filtering module, which operates directly on SPARQL and supports nested facets. Second, an algorithm that efficiently detects spatial information related to those resources that satisfy the facet selection. The detected relations are used for automatically presenting data on a map. And third, a workflow for making the map display interact with data sources that contain large amounts of geometric information, as shown in Figure 2.1. In comparison to Facete, in our work we focus on the user's interaction regarding the GEO aspect, both as input and output, rather the visualization of the underlying dataset in terms of exploration. Specifically, we use the map for the navigation of the spatial dimension(s) as well as an enriched interactive display. This user friendly GUI enables the user to interact with the system in a two way, enable to explore or navigate and visualize data with spatial dimension(s) simultaneously.

Regarding the placement of our work Table 2.1 categorizes the aforementioned frameworks according to various aspects like Faceted restriction, Preferences, Display of focus on the map, Map-based restriction and preferences, Linked Data and SPARQL Endpoint. Firstly, Facete [13] supports Faceted restrictions over Linked Data using a SPARQL Endpoint and displays the results on a map. Similar to this, both LinkedGeoData [1] and GeoLinkedData [3] are enhanced by the same features as shown in Figure 2.2. On the contrary, GEORank [12] can only produce the output on a map with no other characteristic. Finally, compared to other tools, our proposed system (PFSgeo) can fully support both Faceted restrictions and preferences and displays the output on a map which can be also used as input method for expressing restrictions and preferences *(Map-based restriction, Map-based preferences).*

	System				
Feature	*Facete*	*GEORank*	*LinkedGeoData*	*GeoLinkedData*	***PFSgeo***
Faceted restrictions	√	-	√	√	√
Preferences	-	-	-	-	√
Display of focus on map	√	√	√	√	√
Map-based restrictions	-	-	-	-	√
Map-based preferences	-	-	-	-	√
Linked Data	√	-	√	√	-
SPARQL Endpoint	√	-	√	√	-

Table 2.1: Placement of our Work



- undefined
  - type 1/31
  - label 1475
  - funding 14773
    - type 1
    - label 14754
    - partner 4718
      - type 1
      - label 4747
      - address 4718
        - city 1914
          - label 1910
          - sameAs 1526
        - country 96
      - organisation class 4
      - NUTS region 733
    - amount 12793
    - partner role 3
  - call 30
  - year 5
  - stategic objective 128
  - instrument 3
  - id 1494

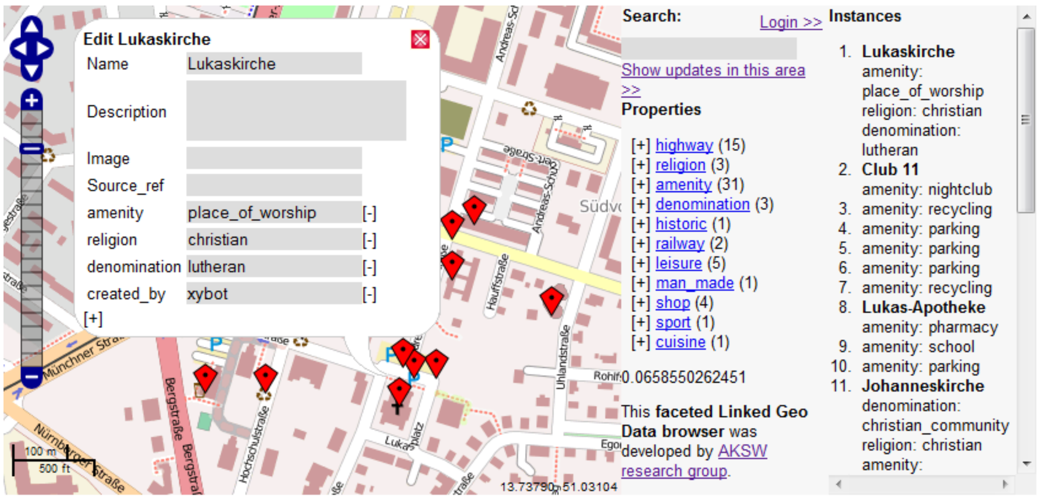
pro

- [ ] Property
- [ ] OntologyProperty
- [ ] AnnotationProperty
- [✓] Project

lod2

s	fv_172	fv_173
LOD2	OPENLINK GROUP LIMITED	870380
LOD2	Korea Advanced Institute of Science and Technology	0
LOD2	INSTITUT MIHAJLO PUPIN	193500
LOD2	WOLTERS KLUWER DEUTSCHLAND GMBH	371815
LOD2	OPEN KNOWLEDGE FOUNDATION LIMITED LBG	362304
LOD2	SEMANTIC WEB COMPANY GMBH	518526
LOD2	TENFORCE BVBA*	706280
LOD2	UNIVERSITAET LEIPZIG	1333901
LOD2	STICHTING CENTRUM VOOR WISKUNDE EN INFORMATICA	505275
LOD2	EXALEAD	470220

Figure 2.1: An indicative screenshot of Facete.


Edit Lukaskirche

Name	Lukaskirche
Description	
Image	
Source_ref	
amenity	place_of_worship [-]
religion	christian [-]
denomination	lutheran [-]
created_by	xybot [-]

[+]

Search: Login >>

Show updates in this area >>

Properties

[+] highway (15)
[+] religion (3)
[+] amenity (31)
[+] denomination (3)
[+] historic (1)
[+] railway (2)
[+] leisure (5)
[+] man_made (1)
[+] shop (4)
[+] sport (1)
[+] cuisine (1)

0.0658550262451

This **faceted Linked Geo Data browser** was developed by AKSW research group.

Instances

1. **Lukaskirche** amenity: place_of_worship religion: christian denomination: lutheran
2. **Club 11** amenity: nightclub
3. amenity: recycling
4. amenity: parking
5. amenity: parking
6. amenity: parking
7. amenity: recycling
8. **Lukas-Apotheke** amenity: pharmacy
9. amenity: school
10. amenity: parking
11. **Johanneskirche** denomination: christian_community religion: christian amenity:

Figure 2.2: LinkedGeoData uses the information collected by the Open-StreetMap project and makes it available as an RDF knowledge base.

2.2.2 Faceted and Dynamic Taxonomies (FDT) and Preference-enriched Faceted Search (PFS)

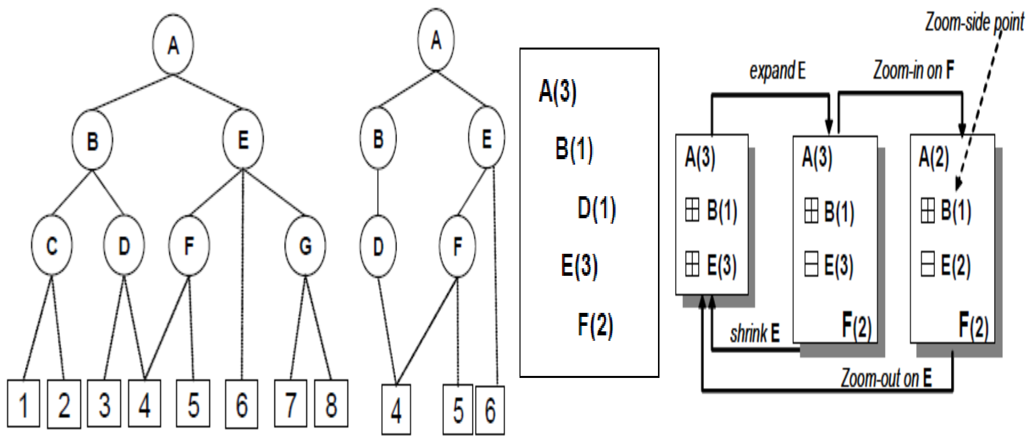
For reasons of self-containedness Section 2.2.2.1 reviews FDT and Section 2.2.2.2 preferences

2.2.2.1 Faceted and Dynamic Taxonomies (FDT)

A highly prevalent model for exploratory search is the interaction of *Faceted and Dynamic Taxonomies* (FDT) [14],[2] usually called *Faceted Search*, which allows the user to get an overview of the information space (e.g. search results) and offers him various groupings of the results (based on their attributes, metadata, or other dynamically mined information).

As stated in [5], modern environments should guide users in exploring the information space and in expressing their information needs in a progressive manner. Systems supporting FDT offer a simple, efficient and effective way for explorative tasks [2]. Dynamic taxonomies (faceted or not) is an interaction framework based on a multidimensional classification of (may heterogeneous) data objects allowing users to browse and explore the information space in a guided, yet unconstrained way through a simple visual interface. Features of this framework include: (a) display of current results in multiple categorization schemes (called facets - or just attributes), (b) display of categories (i.e. attribute values) leading to non-empty results only, (c) display of the count information of the indexed objects of each category (i.e. the number of results the user will get by selecting that category), and (d) the user can refine his focus gradually, i.e. it is a session-based interaction paradigm in contrast to the query-and-response dialog of current Web Search Engines (WSE) which is stateless.

An example of the idea of dynamic taxonomies assuming only one facet, is shown in Figure 2.3. Figure 2.3a shows a taxonomy comprising 7 terms (A-G) and 8 indexed objects (1-8). Figure 2.3b shows the dynamic taxonomy if we restrict our focus to the objects 4,5,6. Notice that it comprises only 5 terms, those that lead to objects in 4,5,6. Figure 2.3c shows the browsing structure that could be provided at the GUI layer (e.g. at the left side bar), and Figure 2.3d sketches user interaction, based on the restriction shown in Figure 2.3b. Notice the count number next to each term. These groupings enable the user to restrict his focus gradually and in a simple way (through clicks, i.e. without having to formulate queries), enabling him to locate resources that would be difficult to locate otherwise (especially the low ranked ones). This model is currently the de facto standard in various domains: e-commerce (e.g. eBay), booking applications (e.g. booking.com), library and bibliographic portals (e.g. ACM Digital Library), museum portals (e.g. Europeana), mobile phone browsers, social networking systems (e.g. LinkedIn), Adaptive Faceted Search on Twitter [3,4] and many others. The are also models that use faceted views over structured and semi structured large scale data for providing interactive browsing over the LOD world with the use of combined full text search, structured querying and result ranking [15]. For a recent survey of methods for applying faceted search over RDF datasets see [16].



(a) (b) (c) (d)

Figure 2.3: Real caption[5]

2.2.2.2 Preference-enriched Faceted Search (PFS)

The enrichment of search mechanisms with *preferences*, hereafter *Preference-enriched Faceted Search* [5], [17], 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. The distinctive features of PFS is the ability to express preferences over attributes whose values can be hierarchically organized, and/or multi-valued, while scope-based rules resolve automatically the conflicts. As a result the user is able to restrict his current focus by using the faceted interaction scheme (hard restrictions) that lead to non-empty results, and rank according

[3]http://ceur-ws.org/Vol-730/paper4.pdf
[4]http://www.wis.ewi.tudelft.nl/research/faceted-search
[5]Source: [5]

to preference the objects of his focus. Recently, PFS has been used also for offering a flexible process for the identification of fish species [18].

2.2.2.3 Hippalus

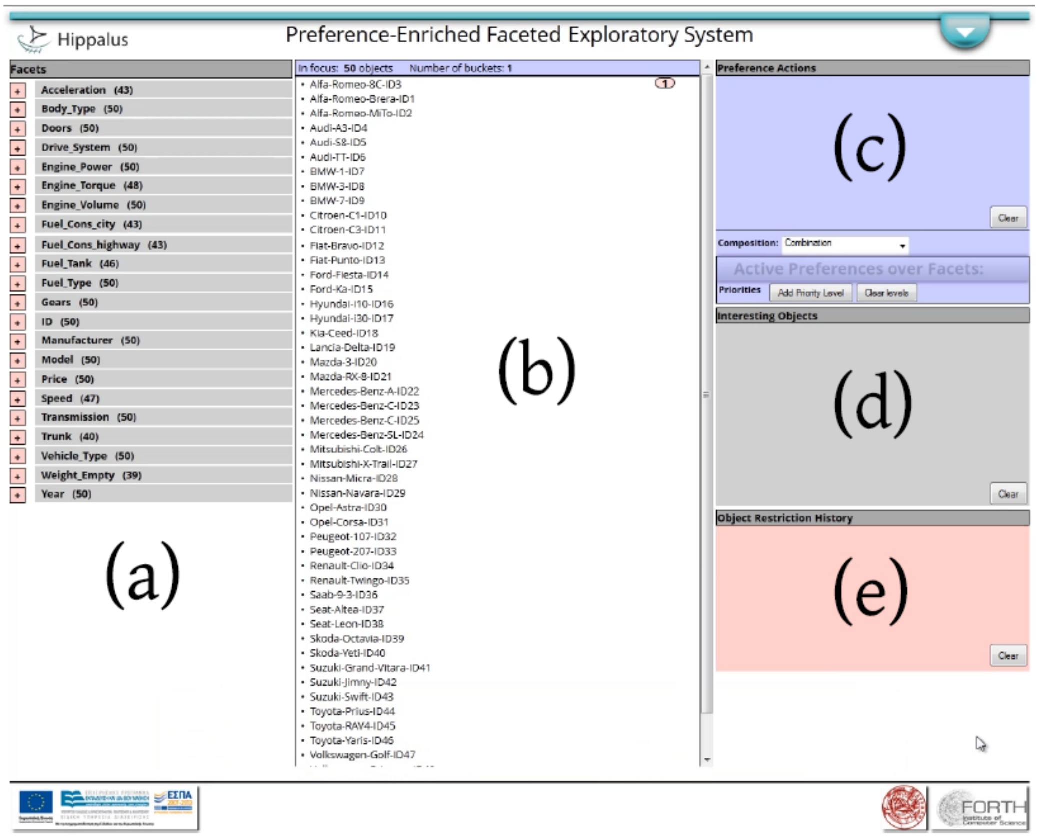
Hippalus [19] is a publicly accessible web system that implements the PFS interaction model [5]. It offers actions that allow the user to order facets, values, and objects using *best, worst, prefer to* actions (i.e. relative preferences), *around to* actions (over a specific value), or actions that 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.

The information base (IB) that feeds Hippalus is represented in RDF/S[6] (using a schema adequate for representing objects described according to dimensions with hierarchically organized values). For loading and querying such information, Hippalus uses Jena[7], a Java framework for building Semantic Web applications. Hippalus offers a web interface for Faceted Search enriched with preference actions. The latter are offered through HTML 5 context menus[8]. The performed actions are internally translated to statements of the preference language described in [5], 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.

Hippalus displays the preference ranked list of objects in the central part of the screen, while the right part is occupied by information that relates to the information thinning (object restrictions), preference actions history and preference composition. The preference related actions are offered through right click activated pop-up menus (through HTML5 context menus) and the interaction is demonstrated in next section. To this respect, consider a simple scenario of an international dealer of used cars and suppose that the available cars are stored in a relational table of the form: *Car(id, manufacturer, category, price, color, power, year, mileage, fuel, location, comment, accessories).* In addition there are four taxonomies that have been designed in order to provide an hierarchical organization for the values of the attributes manufacturer, fuel, location and category. The leaves of these taxonomies

[6]http://www.w3.org/TR/rdf-schema/
[7]http://jena.apache.org/
[8]Available only to firefox 8 and up.

are the domains of the corresponding attributes which are recorded in the tuples of the produced relational table.



Facets

- Acceleration (43)
- Body_Type (50)
- Doors (50)
- Drive_System (50)
- Engine_Power (50)
- Engine_Torque (48)
- Engine_Volume (50)
- Fuel_Cons_city (43)
- Fuel_Cons_highway (43)
- Fuel_Tank (46)
- Fuel_Type (50)
- Gears (50)
- ID (50)
- Manufacturer (50)
- Model (50)
- Price (50)
- Speed (47)
- Transmission (50)
- Trunk (40)
- Vehicle_Type (50)
- Weight_Empty (39)
- Year (50)

In focus: 50 objects Number of buckets: 1

1

- Alfa-Romeo-8C-ID3
- Alfa-Romeo-Brera-ID1
- Alfa-Romeo-MiTo-ID2
- Audi-A3-ID4
- Audi-S8-ID5
- Audi-TT-ID6
- BMW-1-ID7
- BMW-3-ID8
- BMW-7-ID9
- Citroen-C1-ID10
- Citroen-C3-ID11
- Fiat-Bravo-ID12
- Fiat-Punto-ID13
- Ford-Fiesta-ID14
- Ford-Ka-ID15
- Hyundai-I10-ID16
- Hyundai-i30-ID17
- Kia-Ceed-ID18
- Lancia-Delta-ID19
- Mazda-3-ID20
- Mazda-RX-8-ID21
- Mercedes-Benz-A-ID22
- Mercedes-Benz-C-ID23
- Mercedes-Benz-C-ID25
- Mercedes-Benz-SL-ID24
- Mitsubishi-Colt-ID26
- Mitsubishi-X-Trail-ID27
- Nissan-Micra-ID28
- Nissan-Navara-ID29
- Opel-Astra-ID30
- Opel-Corsa-ID31
- Peugeot-107-ID32
- Peugeot-207-ID33
- Renault-Clio-ID34
- Renault-Twingo-ID35
- Saab-9-3-ID36
- Seat-Altea-ID37
- Seat-Leon-ID38
- Skoda-Octavia-ID39
- Skoda-Yeti-ID40
- Suzuki-Grand-Vitara-ID41
- Suzuki-Jimny-ID42
- Suzuki-Swift-ID43
- Toyota-Prius-ID44
- Toyota-RAV4-ID45
- Toyota-Yaris-ID46
- Volkswagen-Golf-ID47

Figure 2.4: The Main Page of Hippalus. (a) Shows the Area where Facets and Terms are Displayed, (b) the Ranked Objects Area, (c) the Preference Actions History and Composition Tool, (d) 'Interesting Objects' Tool (i.e. Like a Shopping Cart) and (e) the Object Restriction History

Figure 2.4 shows the main page of **Hippalus** over the collection of 50 cars. Specifically, part (a) shows the attributes, their values (which can be hierarchically organized), accompanied by the number of their occurrences, where the user can restrict his focus or express preferences anchored to them. Part (b) depicts the objects area, which is ranked according to preference, part (c) shows the preference actions history and composition tool, part (d) displays the 'Interesting Objects' tool (i.e. like a shopping cart) and finally, part (e) the object restriction history. Hippalus has been evaluated very positively by users in various contexts (the interested reader can refer to [19, 20, 21]).

Chapter 3

PFSgeo

We will describe PFSgeo through a running example over a small dataset.

3.1 Dataset of the Running Example

Consider a small information base comprising information about rooms of 8 hotels in Heraklion. Each hotel can be inclusively described by the following attributes: a) a set-valued attribute *Accessories* that describes the accessories offered by a room, b) the *Address* attribute with the address of the hotel/apartment, c) a set-valued attribute *Conditions* with the conditions of booking a room/apartment, d) the *Latitude* attribute describing the latitude of the room/apartment, e) the *Location* attribute, which is an hierarchically organized attribute describing the location of the room (e.g. Center/Heraklion/Crete/Greece), f) the *Longitude* attribute describing the longitude of the room/apartment, g) the *Name* attribute with the name of the hotel/apartments, h) the *Persons* attribute with the number of persons included in the booking price, i) the *Price* attribute describing the price of the room/apartment for 4 nights, j) the *Rating* attribute that describes the rating of the hotel according to other users, k) the *Room* hierarchically organized attribute that describes the type of the room and finally, l) the *Star* attribute describing the number of stars. Notice that only Address, Location, Latitude and Longitude facets describe spatial data. The pair of Latitude and Longitude defines a point on the map.

For example, the way these attributes are represented regarding a hotel room are shown in the corresponding RDF/TTL file:

```
a hippalus:Hippalus_Id , hippalus:Center ;
hippalus:Accessories "Free-WiFi" ^^xsd:string ,
"Air-conditioning" ^^xsd:string , "Flat-screen-TV" ^^xsd:string
;
hippalus:Address "Leof_62_Martiron_1"^^xsd:string ;
hippalus:Conditions "Breakfast-included"^^xsd:string ,
"Special-conditions"^^xsd:string ;
hippalus:Latitude "35.336291"^^xsd:float ;
hippalus:Longitude "25.123639"^^xsd:float ;
hippalus:Name "Castello_City_Hotel"^^xsd:string ;
hippalus:Persons "1"^^xsd:int ;
hippalus:Price_4_nights_in_Euros "240"^^xsd:int ;
hippalus:Rating "9.0"^^xsd:float ;
hippalus:Room "Single"^^xsd:string ;
hippalus:Stars "3"^^xsd:int .
```

In a nutshell, the inherent spatial dimension of the data is a set of triples in RDF/TTL file, describing a point as an abstract representation of the GEO aspect with attributes such as Location, Latitude and Longitude. The purpose of this generic format is to take advantage of the Linked Data with context such as Europeana or DBpedia.

3.2 The Extension of the Interaction Model in Brief

Here we describe the interaction model of PFSgeo as an extension of the PFS. We shall hereafter call a facet geographic if it corresponds to geographic coordinates. In brief the extensions of the interaction model are twofold:

- **Object Display:** When a facet corresponds to geographic information (e.g. a facet location for hotels), the user should be able to see the positions of the focused objects on a map.

- **User Actions:** In Faceted Search the user can select one term by clicking for restricting the focus. In PFS the user can select one term and with right click he can express one preference action. In PSFgeo, if a facet is geographic, then instead of selecting values (for restricting the focus or for expressing a preference), the user can select the desired values through the map (through shapes such as circle, polygon or just zoom in/out and clicking). In various scenarios this can be more intuitive and simple for the user (e.g. if the user knows the desired area he can focus there without having to know the names of the places as he should in FS and PFS). Consequently, selections of that kind usually concern several values.

We should also mention that the proposed system is a state-of-the-art faceted enriched Geo-visualization system in contrast to a Single Page Application (SPA), thus we can fully support Preference-enriched Faceted Search exploration and visualization of different data sets with GEO aspect simultaneously, as shown in Figure 3.1.

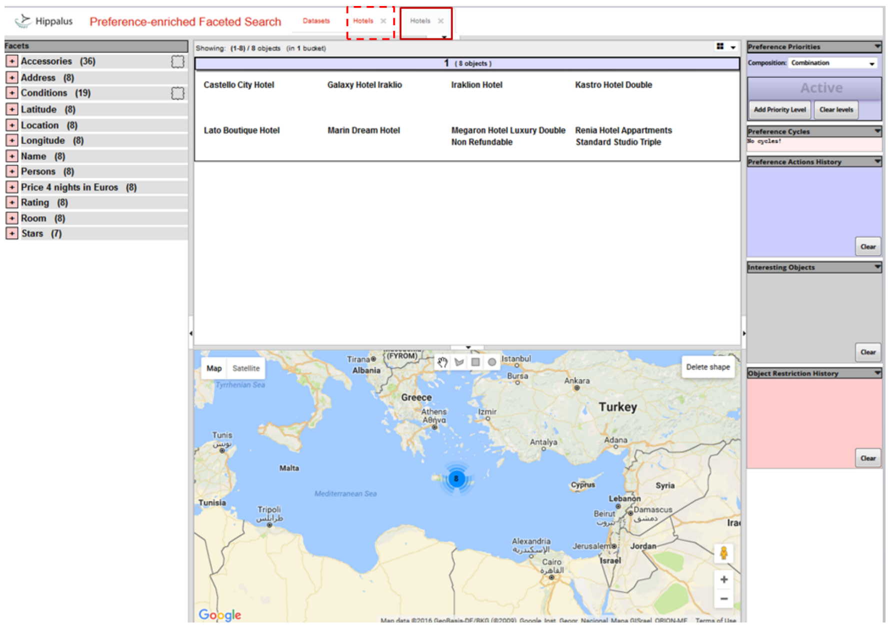


Figure 3.1: PFS geo exploration in different geo datasets(tabs).

The Interaction Model over the Running Example

The focus can be displayed in the classical way in FS and PFS, i.e. as a list of textual elements, but also in the map, or both list and map. Figure 3.2 shows the textual mode. From the top right the user can switch to the map mode as shown in Figure 3.3 (the user can also switch to an image mode if the objects are described also by images, e.g. if our dataset was enriched with hotel photographs). Figure 3.4 shows the screen if the map mode is active. We can see that the objects are displayed on a map. Figure 3.4 shows the screen if the map mode is active. We can see the objects are displayed on the map.

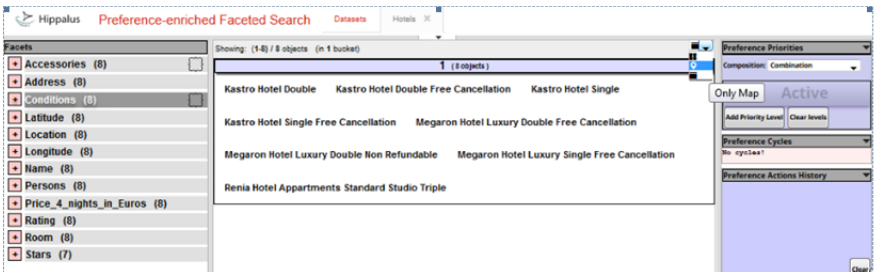

Hippalus
Preference-enriched Faceted Search
Datasets
Hotels
Facets
Accessories (8)
Address (8)
Conditions (8)
Latitude (8)
Location (8)
Longitude (8)
Name (8)
Persons (8)
Price_4_nights_in_Euros (8)
Rating (8)
Room (8)
Stars (7)
Showing: (1-8) / 8 objects (in 1 bucket)
1 (8 objects)
Kastro Hotel Double
Kastro Hotel Double Free Cancellation
Kastro Hotel Single
Kastro Hotel Single Free Cancellation
Megaron Hotel Luxury Double Free Cancellation
Megaron Hotel Luxury Double Non Refundable
Megaron Hotel Luxury Single Free Cancellation
Renia Hotel Appartments Standard Studio Triple
Only Map
Preference Priorities
Composition: Combination
Active
Add Priority Level
Clear levels
Preference Cycles
No cycles!
Preference Actions History
Clear

Figure 3.2: The first screen showing the elements of the focus (in the center) as a list of textual descriptions.

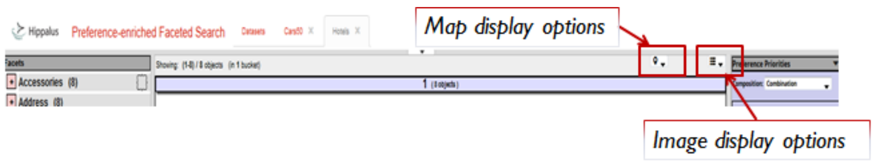


Figure 3.3: Different view options for the Map and Image display

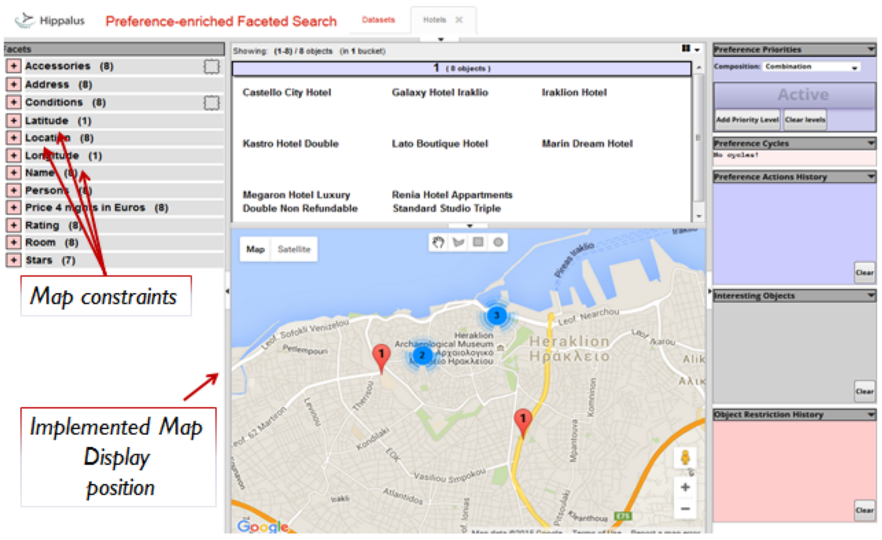
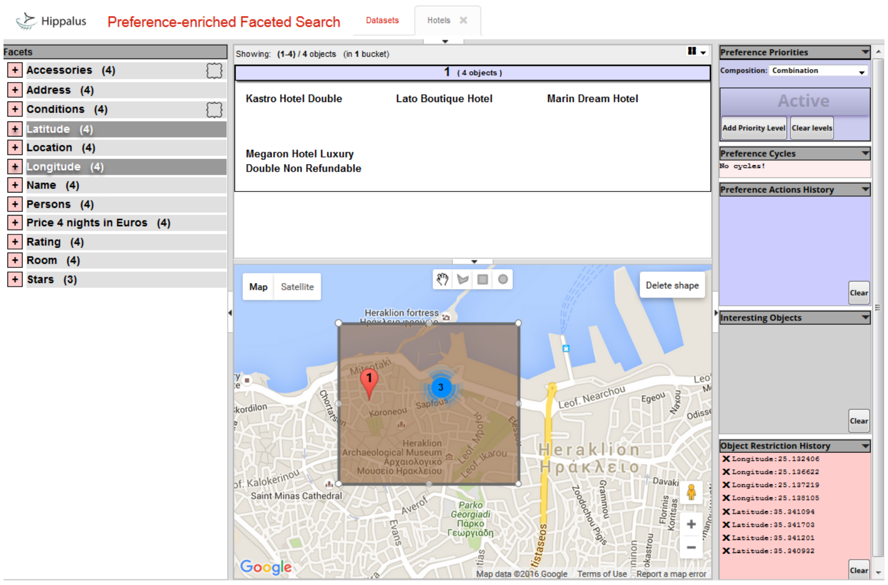


Figure 3.4: An indicative screenshot of Hippalus enhanced with GEO aspect.

Let us now describe the interaction of user with the map. For instance, the user zooms in an area and can draw a rectangle over the markers he or she is interested in. This shape will trigger a request to Google Map that will return the containing markers, that can be processed in Hippalus server according to user's action rules, e.g. as a restriction as shown in the Figure 3.5.



Kastro Hotel Double
Lato Boutique Hotel
Marin Dream Hotel
Megaron Hotel Luxury Double Non Refundable

Longitude:25.132406
Longitude:25.136622
Longitude:25.137219
Longitude:25.138105
Latitude:35.341094
Latitude:35.341703
Latitude:35.341201
Latitude:35.340932

Figure 3.5: Multi restriction on the hotels within a drawable shape-left click.

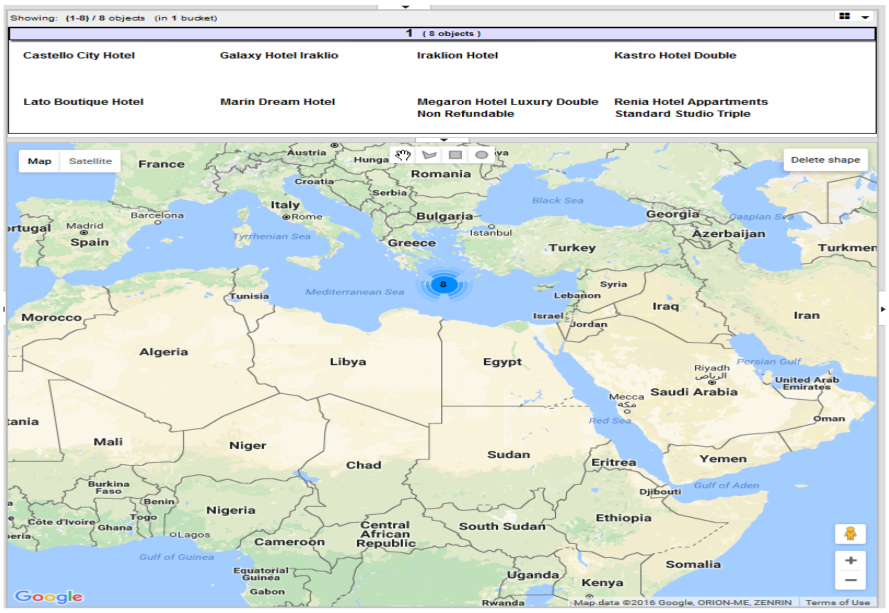
Handling cases with big number of markers

If there are too many objects which are very close, the system implements the MarkerClusterer v3[1], 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. For example, as shown in Figure 3.6 we get only a clustering marker for the greater area of Heraklion, whereas when the zoom level is changed, we can distinguish that we have hotels located in two different major areas, Heraklion and Agia Pelagia.

Making evident the preferences on the map

Another significant characteristic of the map is that it allows the user to view the results according to the expressed preferences. For instance, suppose that the user has restricted the set of hotels to those located in the center of Heraklion (Center/Heraklion/Crete/Greece), either explicitly using the facet terms, either implicitly expressing his actions through the map. Then the user can further restrict his focus or issue a preference action, e.g. he can state that he prefers (over the others) those hotels that fall in a restricted area. This will eventually create two different buckets: the first having the more preferred hotels, and the second having the less preferred ones. Currently, there is full consistency between the text-based and the map-displayed results. Thus, the geo aspect as shown on the map, is directly attached to the instances containing in the buckets. For instance, the user can select a rectangle shape to set a restriction over an area containing four hotels, represented by 4 markers, as shown in Figure 3.5.

[1]https://developers.google.com/maps/articles/toomanymarkers

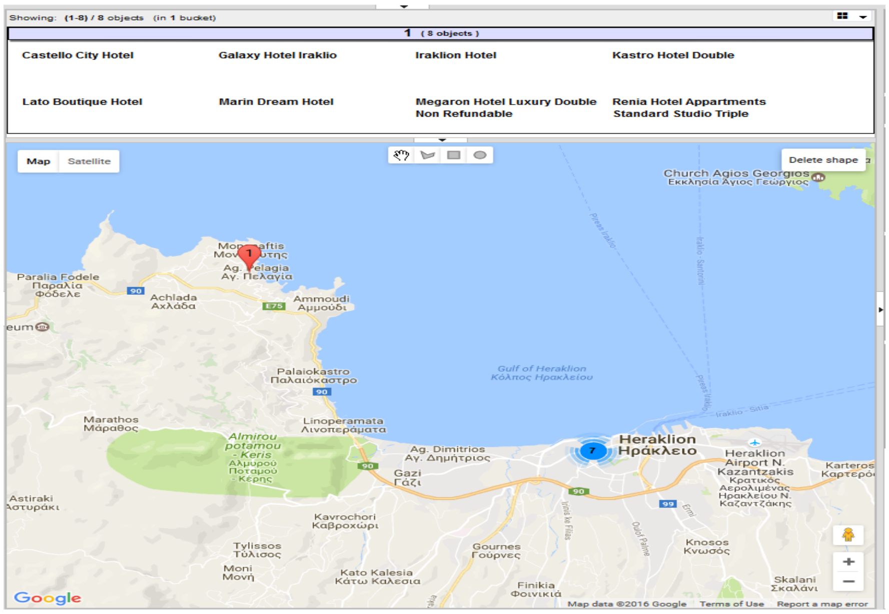


Showing: (1-8) / 8 objects (in 1 bucket)

1 (8 objects)

Castello City Hotel	Galaxy Hotel Iraklio	Iraklion Hotel	Kastro Hotel Double
Lato Boutique Hotel	Marin Dream Hotel	Megaron Hotel Luxury Double Non Refundable	Renia Hotel Appartments Standard Studio Triple

(a)



Showing: (1-8) / 8 objects (in 1 bucket)

1 (8 objects)

Castello City Hotel	Galaxy Hotel Iraklio	Iraklion Hotel	Kastro Hotel Double
Lato Boutique Hotel	Marin Dream Hotel	Megaron Hotel Luxury Double Non Refundable	Renia Hotel Appartments Standard Studio Triple

(b)

Figure 3.6: Marker clustering for display GEO location on Google Map 3.6a and zooming in 3.6b

3.3 The Interaction Model By Examples

To aid understanding the interaction model of the system we highlight the benefits of geo extension on PFS for decision making through the following running scenarios:

Find a Hotel Room in Heraklion

Consider a dataset with hotels and a user with a very complicated information need and preferences, aiming to find the desired hotel very easily. To this respect, you are planning to stay in a Hotel in a specific area. Although you do have a priori preferences e.g. hotels by the sea or near Points of Interests (POIs), you don't know nor the area or any location information that can be applied as soft or hard constraints in the classical faceted search. Note that this does not apply in PFSgeo, since the user is able to locate POIs over the map in order to apply any actions. E.g. *"You are interested in 3-star Hotels in the greater area of Heraklion city and you don't prefer those located in the center, but you prefer those near to the Archaeological Museum".*

IT Departments in Greece

In addition, consider a dataset that contains information about all departments of Computer Science and Engineering in Greece for aiding students to select the department to apply. At first we formulate the description of 2 possible examples and afterwards we describe the details of the task based actions for creating them. A sketch of the examples is followed:

a *You know you don't want to study in islands. You also want to study around the greater area of Athens. Despite that, you don't want to study in the areas (suburbs) of Πειραιάς and in Αιγάλεω-Περιστέρι.*

b *You have an estimation regarding the result ranking of your exams* $\sim 14,000$ *and an a prior preference of the years of studies (you prefer 4 years over 5 years) and you don't prefer **T.E.I**. Although you don't want to study in **islands**, you do like the departments in **Crete** since you live in Crete.*

Table 3.1 and 3.2 displays the description of the tasks for the aforementioned examples.

Table: Task Description of example (a)

Number of Task	*Description*
1st	Set Νησιά in Περιοχή as Worst.
2nd	Use the rectangle shape on the map and set the major area of Athens as Best.
3rd	Use the rectangle shape set the area of Πειραιάς as Worst.
4th	Use the rectangle shape set the area of Αιγάλεω-Περιστέρι as Worst.

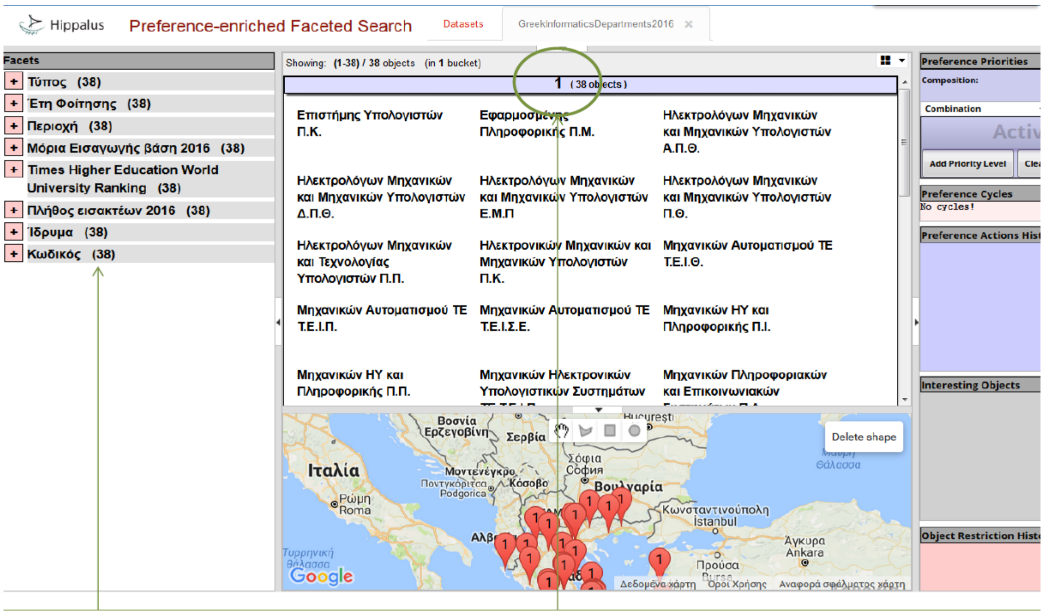
Table 3.1: Task Description over Greek IT Departments (a) example.

Table: Task Description of example (b)

Number of Task	*Description*
1st	Prefer **4** over **5** value in Έτη Φοίτησης.
2nd	Set T.E.I. in Τύπος as Worst.
3rd	Set Νησια in Περιοχή as Worst.
4th	Set Κρήτη in Περιοχή as Best.
5th	Μόρια Εισαγωγής βάση 2016 value around 14.000 (13967).

Table 3.2: Task Description over Greek IT Departments on (b) example.

To get an overview of the interaction Figure 3.7, 3.8 and 3.9 shows the task based sequence regarding the **example (b)**.



Facets
- Τύπος (38)
- Έτη Φοίτησης (38)
- Περιοχή (38)
- Μόρια Εισαγωγής βάση 2016 (38)
- Times Higher Education World University Ranking (38)
- Πλήθος εισακτέων 2016 (38)
- Ίδρυμα (38)
- Κωδικός (38)

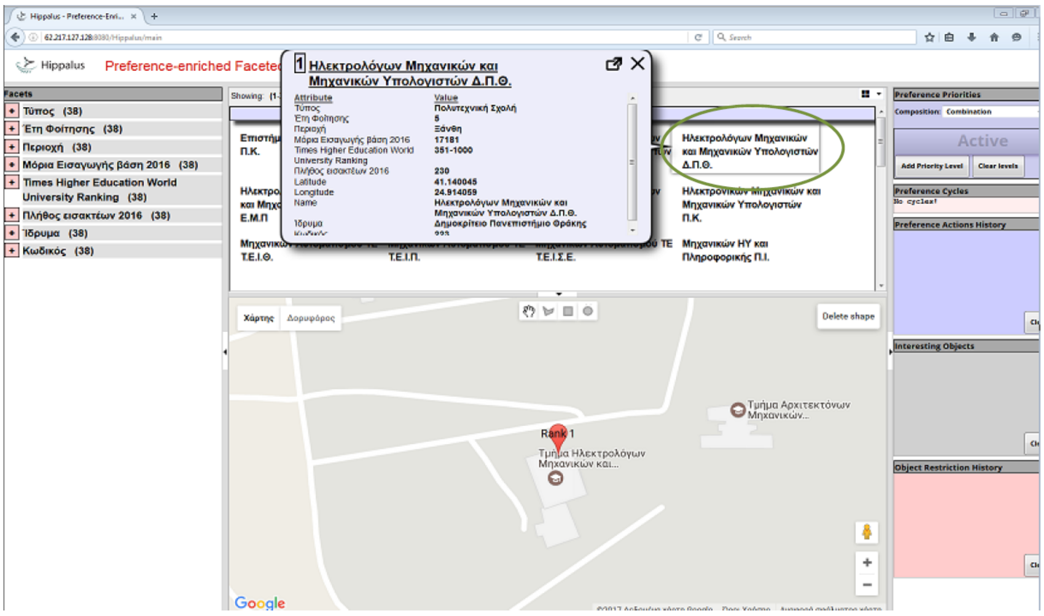
Showing: (1-38) / 38 objects (in 1 bucket)

1 (38 objects)

Επιστήμης Υπολογιστών Π.Κ.	Εφαρμοσμένης Πληροφορικής Π.Μ.	Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Α.Π.Θ.
Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Δ.Π.Θ.	Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Ε.Μ.Π	Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Π.Θ.
Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών Π.Π.	Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών Π.Κ.	Μηχανικών Αυτοματισμού ΤΕ Τ.Ε.Ι.Θ.
Μηχανικών Αυτοματισμού ΤΕ Τ.Ε.Ι.Π.	Μηχανικών Αυτοματισμού ΤΕ Τ.Ε.Ι.Σ.Ε.	Μηχανικών ΗΥ και Πληροφορικής Π.Ι.
Μηχανικών ΗΥ και Πληροφορικής Π.Π.	Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων	Μηχανικών Πληροφοριακών και Επικοινωνιακών

No cycles!

(a)


Hippalus - Preference-Enri...
62.217.127.128:8080/Hippalus/main
Search
Hippalus
Preference-enriched Faceted Search
Preference Priorities
Composition: Combination
Active
Add Priority Level
Clear levels
Preference Cycles
Preference Actions History
Interesting Objects
Object Restriction History
Χάρτης
Δορυφόρος
Delete shape
Rank 1
Τμήμα Ηλεκτρολόγων Μηχανικών και...
Τμήμα Αρχιτεκτόνων Μηχανικών...

1 Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Δ.Π.Θ.

Attribute	Value
Τύπος	Πολυτεχνική Σχολή
Έτη Φοίτησης	5
Περιοχή	Ξάνθη
Μόρια Εισαγωγής βάση 2016	17181
Times Higher Education World University Ranking	351-1000
Πλήθος εισακτέων 2016	230
Latitude	41.140045
Longitude	24.914059
Name	Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Δ.Π.Θ.
Ίδρυμα	Δημοκρίτειο Πανεπιστήμιο Θράκης

Facets
- Τύπος (38)
- Έτη Φοίτησης (38)
- Περιοχή (38)
- Μόρια Εισαγωγής βάση 2016 (38)
- Times Higher Education World University Ranking (38)
- Πλήθος εισακτέων 2016 (38)
- Ίδρυμα (38)
- Κωδικός (38)

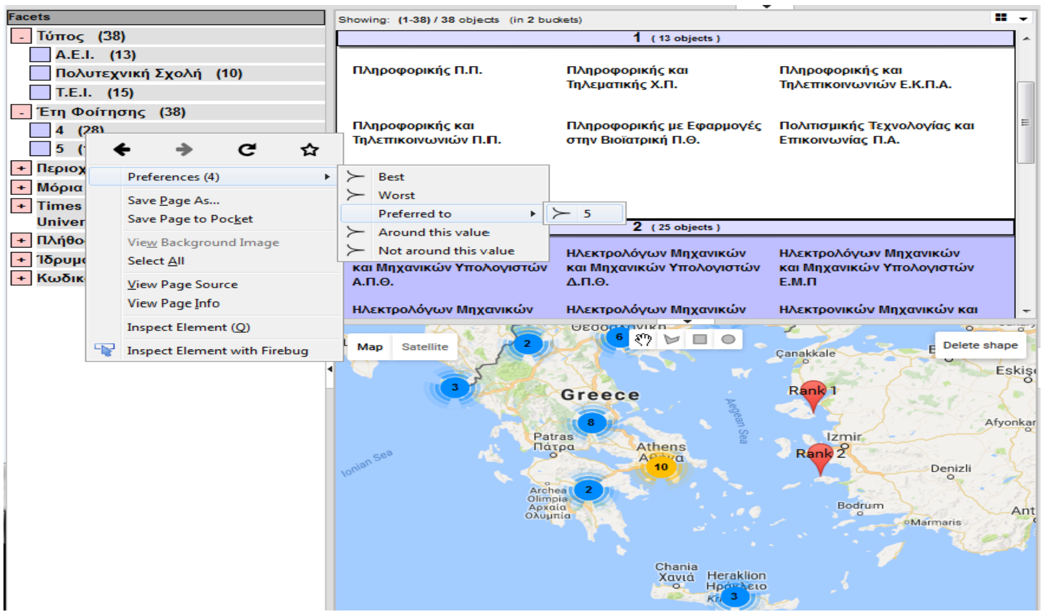
Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Δ.Π.Θ.

Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών Π.Κ.

Μηχανικών ΗΥ και Πληροφορικής Π.Ι.

(b)

Figure 3.7: An overview of the available Departments in Greece 3.7a. Click on a Department to show more information and focus the current view on it's position on the map 3.7b .



Showing: (1-38) / 38 objects (in 2 buckets)

1 (13 objects)

Πληροφορικής Π.Π. | Πληροφορικής και Τηλεματικής Χ.Π. | Πληροφορικής και Τηλεπικοινωνιών Ε.Κ.Π.Α.

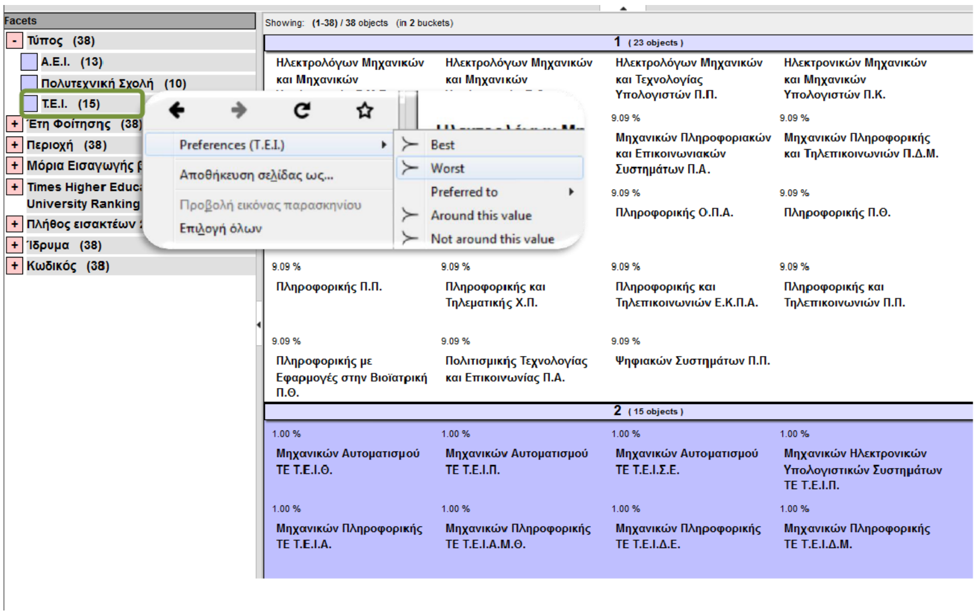
Πληροφορικής και Τηλεπικοινωνιών Π.Π. | Πληροφορικής με Εφαρμογές στην Βιοϊατρική Π.Θ. | Πολιτισμικής Τεχνολογίας και Επικοινωνίας Π.Α.

2 (25 objects)

και Μηχανικών Υπολογιστών Α.Π.Θ. | Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Δ.Π.Θ. | Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Ε.Μ.Π

Ηλεκτρολόγων Μηχανικών | Ηλεκτρολόγων Μηχανικών | Ηλεκτρονικών Μηχανικών και

(a)



Showing: (1-38) / 38 objects (in 2 buckets)

1 (23 objects)

Ηλεκτρολόγων Μηχανικών και Μηχανικών | Ηλεκτρολόγων Μηχανικών και Μηχανικών | Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών Π.Π. | Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών Π.Κ.

9.09 % Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων Π.Α. | 9.09 % Μηχανικών Πληροφορικής και Τηλεπικοινωνιών Π.Δ.Μ.

9.09 % Πληροφορικής Ο.Π.Α. | 9.09 % Πληροφορικής Π.Θ.

9.09 % Πληροφορικής Π.Π. | 9.09 % Πληροφορικής και Τηλεματικής Χ.Π. | 9.09 % Πληροφορικής και Τηλεπικοινωνιών Ε.Κ.Π.Α. | 9.09 % Πληροφορικής και Τηλεπικοινωνιών Π.Π.

9.09 % Πληροφορικής με Εφαρμογές στην Βιοϊατρική Π.Θ. | 9.09 % Πολιτισμικής Τεχνολογίας και Επικοινωνίας Π.Α. | 9.09 % Ψηφιακών Συστημάτων Π.Π.

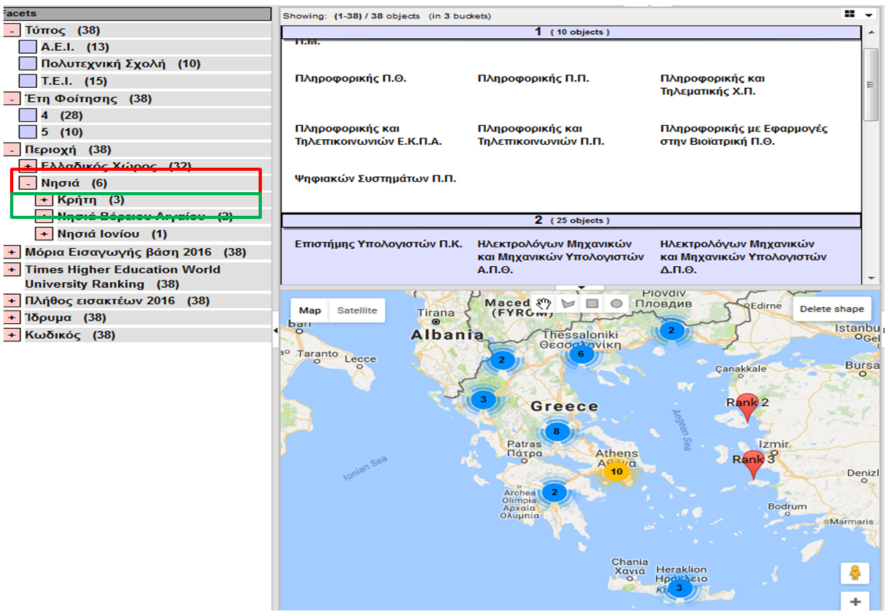
2 (15 objects)

1.00 % Μηχανικών Αυτοματισμού ΤΕ Τ.Ε.Ι.Θ. | 1.00 % Μηχανικών Αυτοματισμού ΤΕ Τ.Ε.Ι.Π. | 1.00 % Μηχανικών Αυτοματισμού ΤΕ Τ.Ε.Ι.Σ.Ε. | 1.00 % Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων ΤΕ Τ.Ε.Ι.Π.

1.00 % Μηχανικών Πληροφορικής ΤΕ Τ.Ε.Ι.Α. | 1.00 % Μηχανικών Πληροφορικής ΤΕ Τ.Ε.Ι.Α.Μ.Θ. | 1.00 % Μηχανικών Πληροφορικής ΤΕ Τ.Ε.Ι.Δ.Ε. | 1.00 % Μηχανικών Πληροφορικής ΤΕ Τ.Ε.Ι.Δ.Μ.

(b)

Figure 3.8: You prefer Departments with 4 years over 5 of studies 3.8a. Express you don't prefer those in T.E.I (but not exclude them) 3.8b .


Facets

- Τύπος (38)
  - Α.Ε.Ι. (13)
  - Πολυτεχνική Σχολή (10)
  - Τ.Ε.Ι. (15)
- Έτη Φοίτησης (38)
  - 4 (28)
  - 5 (10)
- Περιοχή (38)
  - Ελλαδικός Χώρος (32)
  - Νησιά (6)
    - Κρήτη (3)
    - Νησιά Βόρειου Αιγαίου (2)
    - Νησιά Ιονίου (1)
- Μόρια Εισαγωγής βάση 2016 (38)
- Times Higher Education World University Ranking (38)
- Πλήθος εισακτέων 2016 (38)
- Ίδρυμα (38)
- Κωδικός (38)

Showing: (1-38) / 38 objects (in 3 buckets)

1 (10 objects)

Π.Μ.

Πληροφορικής Π.Θ. | Πληροφορικής Π.Π. | Πληροφορικής και Τηλεματικής Χ.Π.

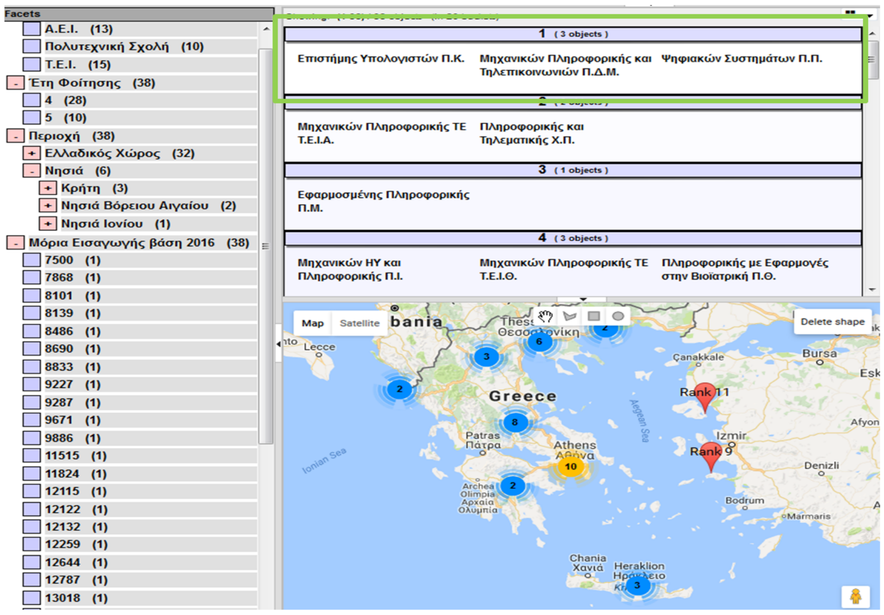
Πληροφορικής και Τηλεπικοινωνιών Ε.Κ.Π.Α. | Πληροφορικής και Τηλεπικοινωνιών Π.Π. | Πληροφορικής με Εφαρμογές στην Βιοϊατρική Π.Θ.

Ψηφιακών Συστημάτων Π.Π.

2 (25 objects)

Επιστήμης Υπολογιστών Π.Κ. | Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Α.Π.Θ. | Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Δ.Π.Θ.

(a)


Facets

- Α.Ε.Ι. (13)
- Πολυτεχνική Σχολή (10)
- Τ.Ε.Ι. (15)
- Έτη Φοίτησης (38)
  - 4 (28)
  - 5 (10)
- Περιοχή (38)
  - Ελλαδικός Χώρος (32)
  - Νησιά (6)
    - Κρήτη (3)
    - Νησιά Βόρειου Αιγαίου (2)
    - Νησιά Ιονίου (1)
- Μόρια Εισαγωγής βάση 2016 (38)
  - 7500 (1)
  - 7868 (1)
  - 8101 (1)
  - 8139 (1)
  - 8486 (1)
  - 8690 (1)
  - 8833 (1)
  - 9227 (1)
  - 9287 (1)
  - 9671 (1)
  - 9886 (1)
  - 11515 (1)
  - 11824 (1)
  - 12115 (1)
  - 12122 (1)
  - 12132 (1)
  - 12259 (1)
  - 12644 (1)
  - 12787 (1)
  - 13018 (1)

1 (3 objects)

Επιστήμης Υπολογιστών Π.Κ. | Μηχανικών Πληροφορικής και Τηλεπικοινωνιών Π.Δ.Μ. | Ψηφιακών Συστημάτων Π.Π.

2

Μηχανικών Πληροφορικής ΤΕ Τ.Ε.Ι.Α. | Πληροφορικής και Τηλεματικής Χ.Π.

3 (1 objects)

Εφαρμοσμένης Πληροφορικής Π.Μ.

4 (3 objects)

Μηχανικών ΗΥ και Πληροφορικής Π.Ι. | Μηχανικών Πληροφορικής ΤΕ Τ.Ε.Ι.Θ. | Πληροφορικής με Εφαρμογές στην Βιοϊατρική Π.Θ.

(b)

Figure 3.9: You prefer those in Crete but not those in islands (conflicting preference) 3.9a. Express you prefer those around 14,000 (13967) to get the final overview 3.9b .

For your consideration you have the ability to use the system, using Firefox version 8 or higher at `http://www.ics.forth.gr/isl/Hippalus/` to apply these scenarios (or a variation) that will be available as *"Explore Hotels in Crete"* in the category "e-Tourism". As regards the Departments of Computer Science in Greece is available as *"Explore Greek IT Departments"* in category "Public Services".

3.4 Architecture

The GEO extension follows the original architecture of Hippalus 2.2.2.3, however several extensions were required for implementing PFSgeo which are described below. To this respect, the architecture of the system and it's components enhanced with the geo aspect is given in Figure 3.10. This servlet based framework is described in Figure 3.11 showing the AJAX sequence diagram.

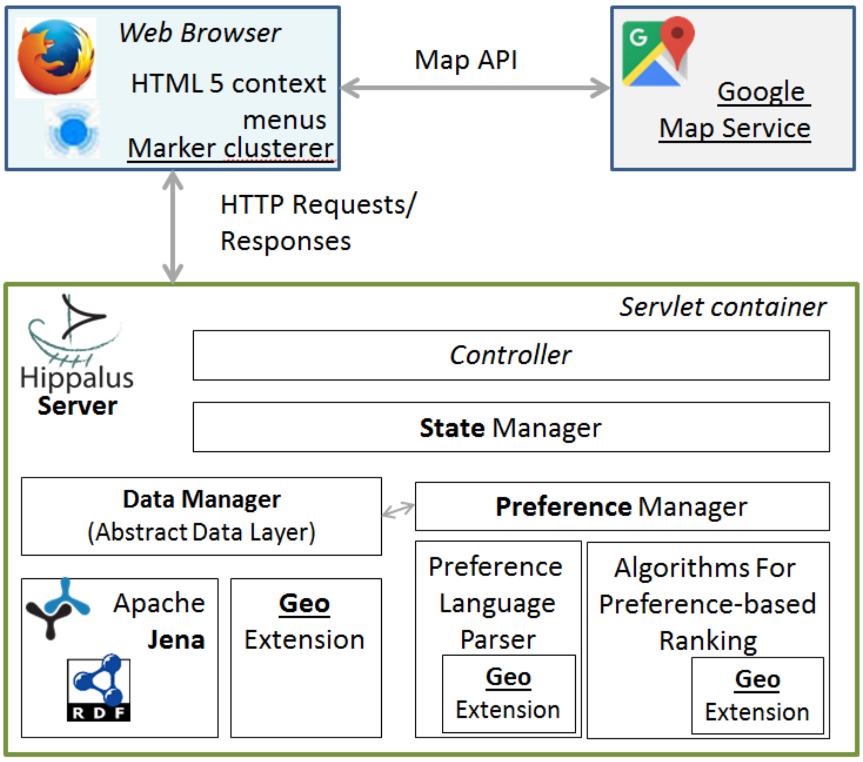


Figure 3.10: Client - Server architecture of Hippalus with GEO extension.

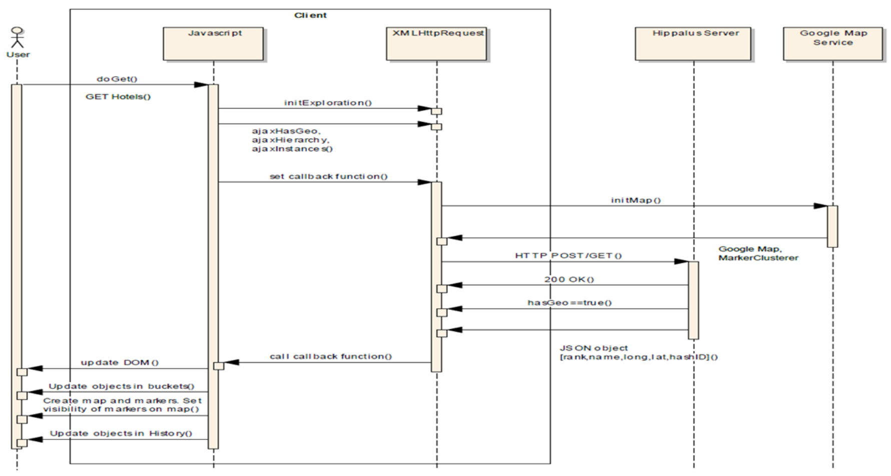

Client
User
Javascript
XMLHttpRequest
Hippalus Server
Google Map Service
doGet()
GET Hotels()
initExploration()
ajaxHasGeo, ajaxHierarchy, ajaxInstances()
set callback function()
initMap()
Google Map, MarkerClusterer
HTTP POST/GET()
200 OK()
hasGeo==true()
JSON object [rank,name,long,lat,hashID]()
call callback function()
update DOM()
Update objects in buckets()
Create map and markers. Set visibility of markers on map()
Update objects in History()

Figure 3.11: AJAX sequence diagram.

3.4.1 The Client

The client uses the *Google Maps v3 API*. Specifically we use *MarkerClusterer v3*, in order to generate the clusters of markers on a map. We also take advantage of the *Drawing Layer library* for Google Maps, that provides a graphical interface that enables the users to draw shapes such as polygons, rectangles and circles that can be further used for restriction, as an alternative input for the facet exploration. The main idea is that the user should be able to interact with the map independently of the faceted exploration system, when he deals with GEO data. The user can choose different shapes and select any of these respectively or delete them and then set and apply new rules. Both restrictions and preferences are possible through left and right click respectively after a shape selection. As regards preferences, in the current prototype the user though a pop-up menu can issue 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.

3.4.2 Server

On server side, when dealing with GEO data we first identify whether the dataset has GEO information. Having this knowledge, Hippalus server creates a JSON object on demand as response to a *getGEOLocations* call of the client. This ensures consistency between the active instances of the buckets and the displayed markers displayed on the map. Specifically, the JSON structured data is basically a GEO representation describing each instance as a point with the following attributes: *Rank, Name*, the *Longitude-Latitude* pair and a hash, as a unique *ID* for each point.

JSON format for each instance =[Rank, Name, Longitude, Latitude, hashID]

Conceptually, this JSON structure is a JSON array of labeled GEO points. The label for each GEO point indicates the bucket number, whereas the GEO point is the representation of the instance using its name and the set of the corresponding coordinates.
Now consider a user that selects to restrict an area using a shape. This will create a JSON array of the coordinates of the containing markers. In contrast to simple restriction, i.e. on a restriction action based on a term of one facet, the restriction here should be performed in a bifold manner, as a pair of longitude and latitude. The final step is performed to the Information Base (IB) in order to find the objects that satisfy this restriction request. Eventually, the performed actions are internally translated to statements of the preference language described in [5], and are then sent to the server through HTTP requests. The formation of the spatial manner query is based upon the same syntax and grammar. 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. It is feasible to assume that the same statements can be applied regarding the restriction, as another form or explicit preference.

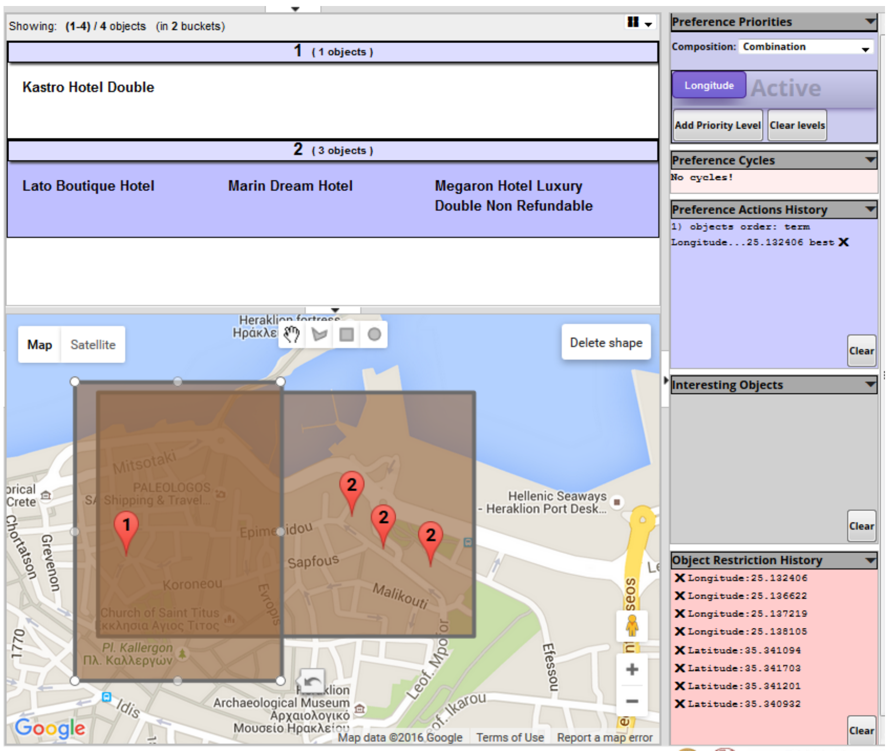
3.4.3 Restrictions (single and multiple)

We currently support two kinds of restrictions using the map feature. In the first, the user restricts the focus using a single marker. As described earlier, each marker is a GEO point which is actually the spatial representation of an object (hotel in our running example). This action, is sent to the server as a HTTP request *ajaxPivot()*, supporting both single or multiple restrictions with parameters a JSON object containing the corresponding latitude and longitude of this marker-instance.
The second kind of restriction is through shapes. When the user draws a shape, e.g. a rectangle using the features of Google Maps, we extract the coordinates of this area and use them again as input in order to query the Information Base (described in section *Preference-enriched Faceted Search (PFS)*) on the server. Based on the provided shapes we can choose the amount of information we want to use in order to set the restriction. For instance, based on the features of the drawing layer library, we can either (a) get the radius and the center of corresponding circle, or (b) get the markers within this area and then pass them on the server as input for the restriction. In order to evaluate the performance of our system when dealing with big amount of data, we investigate both approaches.

In the first prototype, we follow the second approach, i.e. we get the markers within an area as a set of lat and long pairs, thus a significant process is performed on client side which already has the knowledge based on the Google map v3. However, and as we shall see in the next section, this is not a panacea. For the markers within a polygon we apply the Ray casting algorithm implemented in js, to overcome the limitations of the Google map v3 API, in order to get the containing points within a given polygon on the map.

3.4.4 Preferences

We follow the same approach (as in restrictions) when the user expresses preferences. Here, instead of using left click to focus and restrict on the markers within a shape, the user can choose to set the markers as *BEST* using the right click as shown in Figures 3.5 and 3.12. By such an action the user expresses that the containing markers are the best over the others. Internally such an action produces a sequence of (best) preference actions for the latitude longitude pair of each marker.


Showing: (1-4) / 4 objects (in 2 buckets)

1 (1 objects)

Kastro Hotel Double

2 (3 objects)

Lato Boutique Hotel Marin Dream Hotel Megaron Hotel Luxury Double Non Refundable

Preference Priorities

Composition: Combination

Preference Cycles

No cycles!

Preference Actions History

1) objects order: term

Longitude...25.132406 best

Interesting Objects

Object Restriction History

Longitude:25.132406

Longitude:25.136622

Longitude:25.137219

Longitude:25.138105

Latitude:35.341094

Latitude:35.341703

Latitude:35.341201

Latitude:35.340932

Figure 3.12: Set a new preference (BEST) for hotels within a different shape-right click.

Chapter 4

Performance Evaluation and Optimizations

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 the attributes described in a previous 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 regarding the size of the dataset and the request load, that simulates a user that interacts with the system using the map:

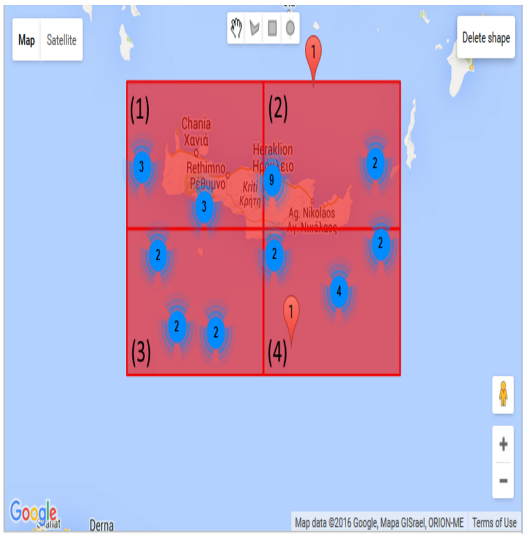
- **First scenario**. This scenario concerns restrictions using the map, i.e. it simulates a user that issues multi restriction actions using a shape.
- **Second scenario**. This scenario concerns preferences using the map, i.e. it simulates a user that issues preference actions using a shape.

We used the following datasets.

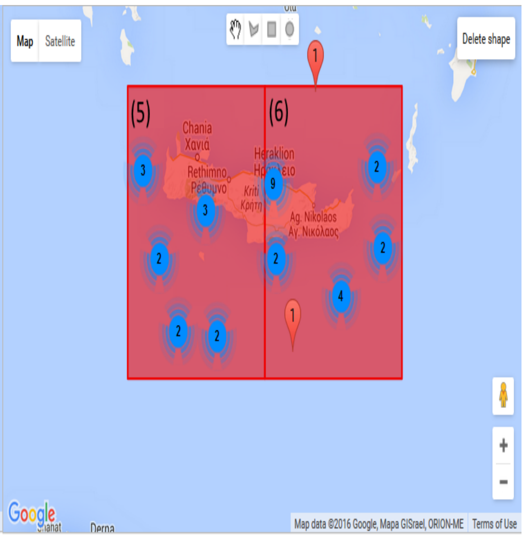
- Dataset 1: 1,000 synthetic hotels
- Dataset 2: 5,000 synthetic hotels
- Dataset 3: 20,000 synthetic hotels

Simulation

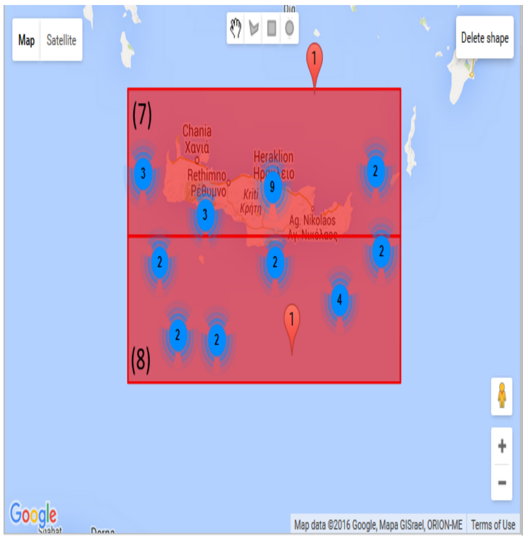
The simulation platform considers the area and the information base described previously. In order to evaluate the system's performance we apply the scenarios on predefined shapes on the map that include a set of markers. The number of markers differs depending on the size of the dataset. In each of 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 Figure 4.1. 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) query based on intersecting shapes, say a shape A and a shape B, eventually has the same outcome as a restriction query using as shape the intersection $A \cap B$.



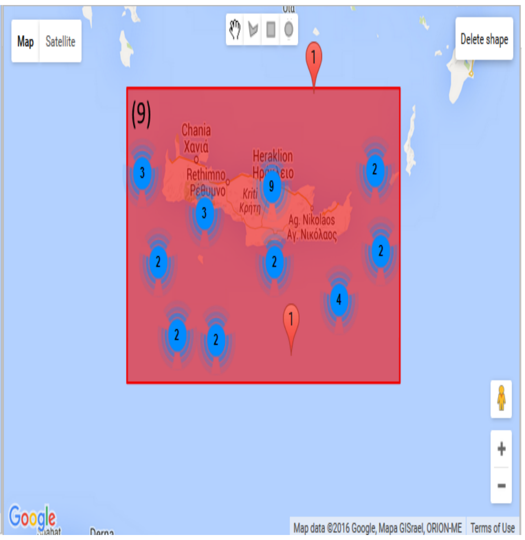
(a) Action of user 1 to 4



(b) Action of user 5 and 6



(c) Action of user 7 and 8



(d) Action of user 9

Figure 4.1: The simulated different user's actions for preferences and restrictions.

4.1.1 Analysis

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 [22]). 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 experiment:*** this is the total response delay for restricting the focus using a multi restriction query.
- **preference time-*2st experiment:*** 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, i.e. when exploring 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 map. Tables 4.1 and 4.2 summarize the total initTime for the three standalone datasets. As we can see in Figure 4.2 the average init time for datasets with 20,000 instances is 1.8 seconds.

Table 4.3 shows the average query time for restrictions, while Figure 4.3, 4.4 and 4.5 show the results of the first experiment for dataset1, dataset2 and dataset3 respectively. Figure 4.6 shows the average query time for 10 requests that get served by the system for each of the three metrics for all datasets.

Table: Init Time for each request

Number of Requests	*Dataset1*	*Dataset2*	*Dataset3*
1st	291 ms	597 ms	2449 ms
2nd	294 ms	521 ms	1932 ms
3rd	197 ms	815 ms	2084 ms
4th	236 ms	532 ms	1743 ms
5th	242 ms	527 ms	1630 ms
6th	238 ms	593 ms	1750 ms
7th	227 ms	562 ms	1716 ms
8th	179 ms	556 ms	1582 ms
9th	226 ms	487 ms	1574 ms
10th	198 ms	542 ms	1634 ms

Table 4.1: Init time of geo instances over the *3 datasets* for each request.

Table: Average Init Time

Number of Dataset	*Average time*
Dataset1	233 ms
Dataset2	573 ms
Dataset3	1810 ms

Table 4.2: Average init time of geo instances over the *3 datasets* for the *10 requests.*

Table: 1st experiment Average Time

Number of Action	*Dataset1*	*Dataset2*	*Dataset3*
User1	154 ms	411 ms	8672 ms
User2	97 ms	344 ms	3944 ms
User3	84 ms	322 ms	3336 ms
User4	76 ms	392 ms	3604 ms
User5	121 ms	526 ms	5011 ms
User6	116 ms	559 ms	4915 ms
User7	120 ms	486 ms	4860 ms
User8	122 ms	508 ms	4930 ms
User9	184 ms	981 ms	8470 ms

Table 4.3: Average query times for restriction of the simulated users over the *3 datasets*.

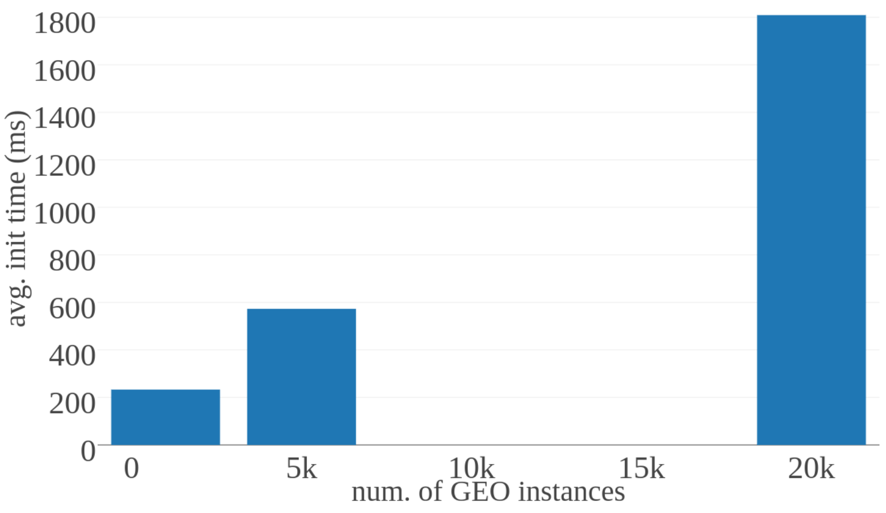


Figure 4.2: Average init time over the *3 datasets*.

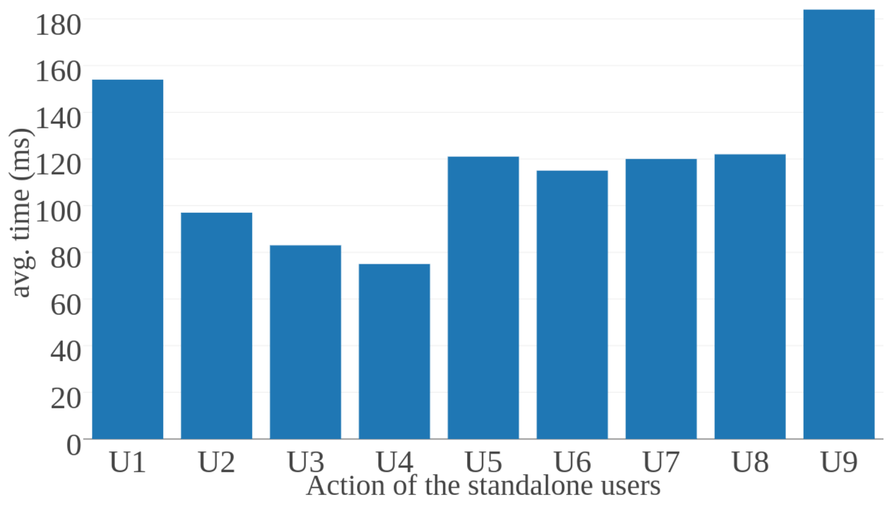


Figure 4.3: Average restriction times of *dataset1* for each standalone user.

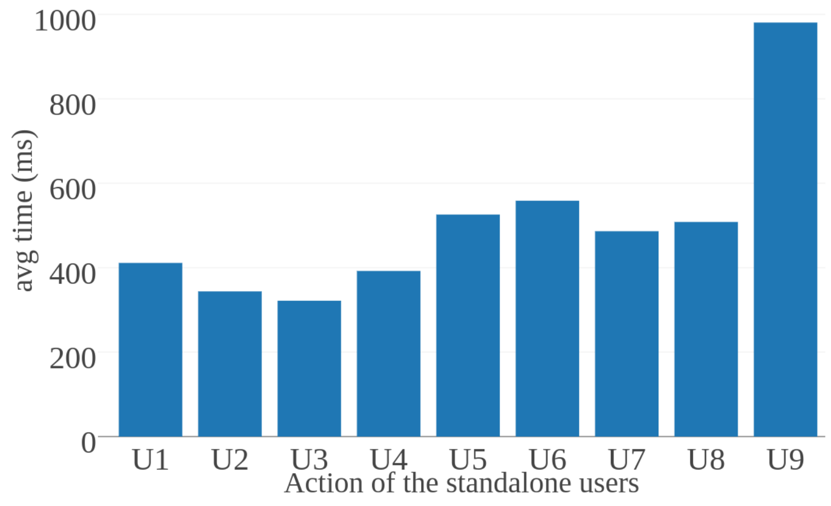


Figure 4.4: Average restriction times of *dataset2* for each standalone user.

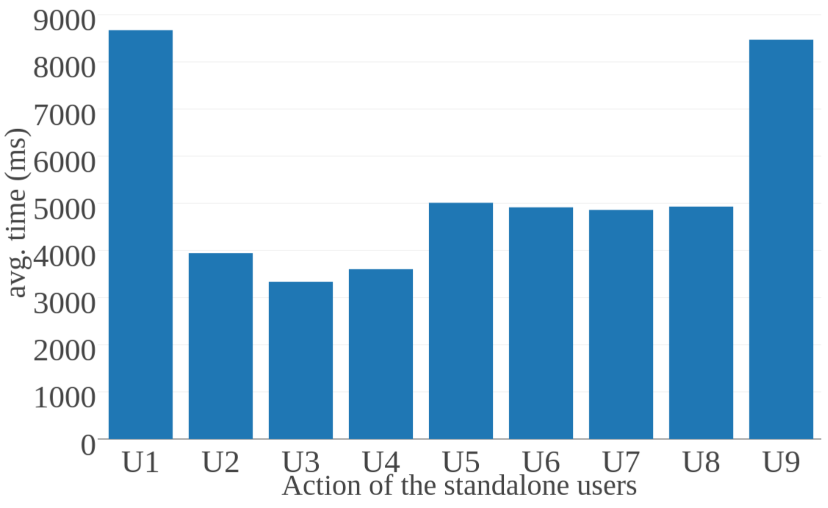


Figure 4.5: Average restriction times of *dataset3* for each standalone user.

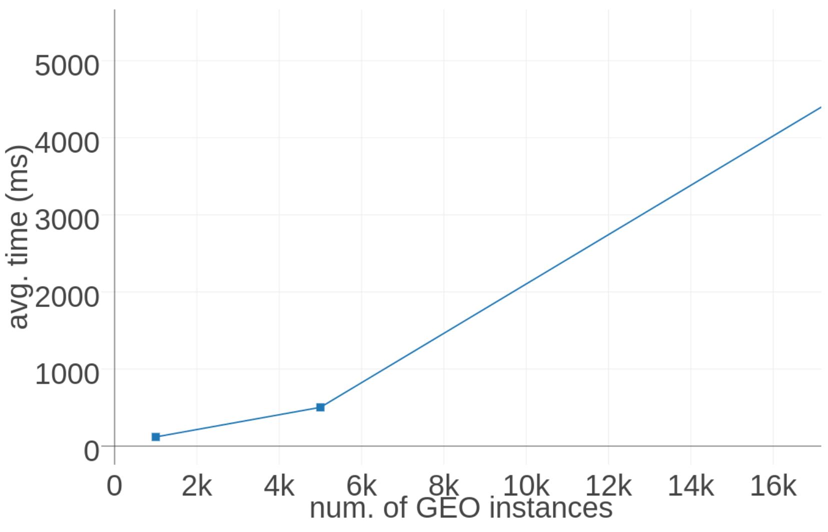


Figure 4.6: Average restriction time for the *all datasets* for all the users.

Table: 2nd experiment Average Time

Number of Action	*Dataset1*
User1	9794 ms
User2	14265 ms
User3	12469 ms
User4	11018 ms
User5	33781 ms
User6	58785 ms
User7	114007 ms
User8	108983 ms
User9	522978 ms

Table 4.4: Average query times for preference of the simulated users over *dataset1*.

To summarize, we can clearly see that although in both scenarios (when interacting over the map) the dataset load is affecting the overall system performance the scale is significant different regarding the action applied. In brief, for restrictions the system can respond in real time (in less than 4 seconds) if the dataset contains more than 5,000 hotels. If the datasets contains more than than 20,000 instances the system can respond in approximately 5 seconds. For preference actions, the system can respond in real time if the dataset contains less than 100 hotels. The reason for this major difference, Table 4.4, is the high computational complexity of the scopes. As such, for producing the ranked list of objects has time complexity $O(|A||\mathcal{B}|^2)$ where A is the set of objects in the focus, and $\mathcal{B}$ is the number of preference actions. This proposed system aims to assistant the users in relation to complex exploratory tasks over large scale geo datasets using visualization tools such as map and shapes, resolving intelligently possible conflicts with the scope-based method thanks to the hierarchically organized values, and the preference inheritance.

4.2 An efficient method for geo preferences

The objective is to speed up the derivation of the ranked list of objects when the user issues preference actions through the map. Without loss of generality, let's assume that the user selects an area in the map through a rectangular shape, defined by the coordinates of the bottom left corner and the upper right corner (say $(x1, y1)$ and $(x2, y2)$ respectively), and through right click he selects one option, say $BEST$. In the method that was described in the previous sections, we first (at client-side) find the hotel markers that fall in selected area. Suppose that these markers correspond to a set of hotels denoted by $Hshape = \{h_1, \ldots, h_k\}$. Then the browser sends to the server a set of preference actions, specifically $|Hshape|$ in number $BEST$ actions, i.e. the following set of actions: $\{\ BEST\ h \mid h \in Hshape\}$. Obviously this approach implies that a lot of data are sent to the server. Moreover the complexity of the algorithm (proposed in [5]) that runs on the server side for producing the ranked list of objects has time complexity O($|A||\mathcal{B}|^2$) where A is the set of objects in the focus, and $\mathcal{B}$ is the number of preference actions. It follows that by increasing the number of actions we actually affect negatively the performance of the algorithm.

Therefore in this section we investigate an alternative approach, an 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: $BEST\ x1, y1, x2, y2$. To support such actions we actually extend the language of [5] with actions that have geographical anchors.

We can already see the benefits: less data have to be transferred and smaller $|\mathcal{B}|$, thus faster production of the ranked list. Now the extension of the language with such actions does not require changing the core algorithms of [5] 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 it will be indicated by the running example that we will use below for detailing the proposed method.

Another important (for scalability) characteristic of the algorithm [5] 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 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.

Running Example. For example consider a user that prefers hotels in Crete but not inside the city of Heraklion. To this end he could issue the following two actions:

```
b1: Best  x1,y1,x2,y2     // island of Crete
b2: Worst x1',y1',x2',y2' // Heraklion city
```

assuming that the first four coordinates capture the entire island of Crete, while those of $b2$ capture only the Cretan city of Heraklion. Note that the area of $b2$ is included in the area of $b1$, therefore the produced ranking should have first all hotels of Crete except those in Heraklion, followed by those hotels in Heraklion.

For producing the ranked list of hotels we can follow the steps of the algorithm proposed in [5], also given below in Alg. 1, adapted to our context. The algorithm takes as input two parameters, an object set E, and a set of actions $\mathcal{B}$ (also a policy for the inactive elements).

The first part of the algorithm orders the actions according to their scope, i.e it defines a partially ordered set $(B, \sqsubseteq)$. In the geographical context an action b is narrower than 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 $x1', y1', x2', y2'$). It is not hard to see that this holds iff $(x1 \geq x1') \wedge (y1 \geq y1') \wedge (x2 \leq x2') \wedge (y2 \leq y2')$ and this is actually how `CheckSubScopeOf` 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 $b \in \mathcal{B}$. 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 $\mathcal{B}$ to a set $\mathcal{B}'$. If an action $b = BEST\ (x1, y1, x2, y2) \in \mathcal{B}$, we add to $\mathcal{B}'$ the following set of actions: $\{\ BEST\ h \mid h \in ActiveScope[b]\}$.

The third part the algorithm just parses the set $\mathcal{B}'$ in order to get the sets B, W, i.e. collecting those hotels with BEST and those with WORST and then it calls the algorithm Apply (that is given and is described below) that produces the ranked hotels by topological sorting. This is all that is required for producing the preference-based ranking of hotels.

Algorithm 1 AlgClearOpt(E, $\mathcal{B}$, $Policy$)
Input: the set of elements E, the set of actions $\mathcal{B}$, and $Policy$ for *inactive* elements
Output: a bucket order over E

```
1: /** Part (1): Computation of $(\mathcal{B}, \sqsubseteq)$ */
2: $Visited \leftarrow \emptyset$
3: $R_{\sqsubseteq} \leftarrow \emptyset$ // $R_{\sqsubseteq}$ corresponds to $\sqsubseteq$
4: for each $b \in \mathcal{B}$ do
5:     for each $b' \in \mathcal{B} \setminus Visited$ do
6:         if CheckSubScopeOf$(b, b')$ then
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:    endfor
14: end for
15: endfor
16:
17: /** Part (2): Efficient Computation of Act. Scopes */
18: for each $b \in \mathcal{B}$ do
19:     $C(b) \leftarrow$ direct children of $b$ wrt $R_{\sqsubseteq}$
20:     ActiveScope$[b] \leftarrow \{e \in E \mid$ IsInScope$(e, b) \wedge$
21: $(\forall c \in C(b)$it holdsIsInScope$(e, c) =$ False$)\}$
22: end for
23: endfor
24:
25: Use the active scopes to expand the set $\mathcal{B}$ to a set $\mathcal{B}'$
26: /** Part (3): Derivation of the final bucket order */
27: $(B, W, R_{\succ}) \leftarrow$ Parse$(\mathcal{B}')$
28: return Apply$(E, B, W, R_{\succ}, Policy)$ // call to Alg. 1
```

Note that the argument $R_{\succ}$ in Apply is empty (i.e. $R_{\succ} = \emptyset$) since we do not support this type of actions because it would be difficult for the users to express them over a map.

The cost of the *first* part of the algorithm is in $\mathcal{O}(|\mathcal{B}|^2)$. Note that as long the user is not submitting a new action, $(\mathcal{B}, \sqsubseteq)$ can be preserved and reused when the user is changing his focus (so $\mathcal{O}(|\mathcal{B}|^2)$ is payed once). The *second* part of the algorithm has $|\mathcal{B}|$ iterations. The cost of each iteration is $|A|(1 + deg)$ where deg is the average number of direct children of an action w.r.t $\sqsubseteq$. It follows that the cost of the second part is $|\mathcal{B}|(|A|(1 + deg)) = |\mathcal{B}||A| + |\mathcal{B}|deg = |A|(|\mathcal{B}| + |\sqsubseteq|)$.

The *last* part of the algorithm is the cost of Alg. AlgApply, which in our context is expressed as $\mathcal{O}(|A|^2)$. However the complexity of this step in normal scenarios is linear.

4.2.1 Details about the algorithm Apply

At first the algorithm Apply constructs a graph by connecting each best to each worst element (so best/worst are interpreted as "each best is preferred to each worst"). Then it adds to the graph the relationships in $R_{\succ}$. We should note here that the parameters B and W actually define a set of relationships (R_{bw} at line 3 of the algorithm), so they could have been expressed directly through the $R_{\succ}$ parameter, however we keep them separate as they constitute an easily enacted (for the user) shorthand. Although a linear or bucket order could be produced by traversing the graph in a breadth first search (BFS) manner (where the first block will contain the more preferred elements, the second the next more preferred, etc), if the transitive reduction is a DAG (Directed Acyclic Graph, i.e. not a tree), then BFS could yield wrong results. Using *topological sorting*[1] instead of BFS, e.g. Alg. AlgTopological as shown above, we can always get a linear order that *respects* R. In particular, Alg. AlgTopological is based on the *source removal algorithm* described in [23], satisfying the condition that all removed maximal nodes are inserted in the same bucket. It begins by finding all the maximal elements of R, moves them into a bucket, and continues with the maximal elements of their children, and so on. In the worst case all elements of E are involved and the most expensive task is that of topological sorting. The topological sorting is in $\mathcal{O}(|E|+|R|)$, thus w.r.t. E we can say that it is in $\mathcal{O}(|E|^2)$. If the actions are object-scoped, i.e. E corresponds to Obj, then the complexity of AlgApply is in $\mathcal{O}(|Obj|^2)$.

[1] Topological sorting yields a linear ordering of the nodes of a DAG such that each node comes before all nodes to which it has outbound edges.

Algorithm 2 Apply(E, $B, W, R_{\succ}, Policy$)
Input: the set of elements E, the set of best elements B, the set of worst elements W, a set of relative relationships $R_{\succ}$, and *Policy* for *inactive* elements
Output: a bucket order over E that respects R

```
1: $R_{bw} \leftarrow \{(b, w) \mid b \in B, w \in W\}$ // each best is preferred than each worst
2: $R \leftarrow R_{bw} \cup R_{\succ}$ //add relative prefs
3: $L \leftarrow AlgTopological(R)$ //produce blocks with boundaries
4: $I \leftarrow E \setminus (B \cup W \cup dom(R_{\succ}))$ // $I$ contains the inactive elements
5: $L' \leftarrow AlgInactive(L, I, Policy)$
6: return $L'$
```

Algorithm 3 AlgTopological(R)
Input: a binary relation R over E
Output: a bucket order over E that respects R

```
1: $L \leftarrow \langle\rangle$
2: repeat
3:     $S \leftarrow maximal_{\succ}(R)$
4:     $R \leftarrow R \setminus \{(x \succ y) \in R \mid x \in S\}$ // Remove maximal
5:     $L \leftarrow L.append(S)$ // Append a bucket to $L$
6: until $S \neq \varnothing$
7: return $L$
```

4.2.2 Syntax and Semantics

Here we introduce an extension of statements regarding the geo aspect in order to support geo preferences based on shapes described in previous algorithms. Each action has a scopeType (either facet, terms order, or object order) determining which kind of elements it affects (facets, terms, or objects). To this respect, each action is "anchored" to one element which can be a facet, term or geo and this allows enacting the preference actions through the GUI straightforwardly. Geo is the spatial representation, which can be rectangle (which used as a proof of concept), polygon, circle or point. Each action is associated to a rank description (rankSpec) which can be lexicographic (for ordering strings), count (for ordering elements based on the number of objects that are classified to them), and value (for ordering numerically-valued facets). The language also defines actions for defining best/worst (i.e. preferred/non-preferred) elements, and relative preferences. Syntactically, preference actions are defined through the following grammar (in BNF):

$$
\begin{array}{lll}
\langle\text{stmt}\rangle & ::= & < scopeType >< spec > \\
\langle\text{scopeType}\rangle & ::= & facets \quad order : | terms \quad order : | object \quad order : \\
\langle\text{spec}\rangle & ::= & < anchor >< rankSpec > \\
\langle\text{anchor}\rangle & ::= & facet < F_i > \\
 & | & term < t_j > \\
 & | & object < o_k > \\
 & | & geo < \text{Area} > \\
 & | & \epsilon \qquad \triangleright \text{ the empty string} \\
\langle\text{Area}\rangle & ::= & rectangle < point >< point > \\
 & | & polygon < point >< point >< point > + \\
 & | & circle < point >< rad > \\
\langle\text{point}\rangle & ::= & float \quad , \quad float \\
\langle\text{rad}\rangle & ::= & int \\
\langle\text{rankSpec}\rangle & ::= & \{lexicographic | count | value\}\{min | max\} \\
 & | & best | worst \\
 & | & usescoreFunction < score() > \{min | max\}
\end{array}
$$

In the above grammar F_i, t_j and o_k denote names that match a facet, a term or object, while score is the name of a real-valued function provided by the user or the application programmer (e.g. around operator for proximity search, which can be the edit distance for categorical or absolute value of distance for numerical values).

The formal description of the *spatial anchors* is listed below:
(a) geo rectangle [($Point_1$),($Point_2$)]
(b) geo polygon [($Point_1$),($Point_2$),($Point_3$),. . . ,($Point_N$)]
(c) geo circle [($Point_1$),radius]
(d) geo point (x_1),(y_1)

Based on the *semantics*, each shape can be defined by it's geo characteristics, as shown in previous section. To this respect, [Appendix A.2] geo rectangle expects 4 coordinates describing the 2 edges, geo polygon expects a list of coordinates defining the path of the polygon and the geo circle expects 2 coordinates defining the point and the radius. Finally, based on the current implementation the user is able to order objects using best and worst (i.e. relative preferences attached to the geo anchors).
Some examples follows:
(a1) object order: geo rectangle [(x1,y1),(x2,y2)] best
(a2) object order: geo rectangle [(x1,y1),(x2,y2)] worst

Action *a1* places all objects classified that fall within the rectangle's borders at the top of the object ordering, whereas the action *a2* places them at the bottom of the object ordering.

4.2.3 Implementation Details

Regarding the implementation of this version some details follow. In this prototype user can issue both ***best*** and ***worst*** preferences actions by using the predefined shapes. Filtering (restriction) and preferences can be selected by left clicking using the corresponding options of the popup menu triggered by the right click as shown in Figure 4.9. For your consideration, based on this version the label of a marker indicates the preference order (i.e. the label "Rank1" on a marker indicates the existence of a hotel in the first bucket, i.e. in the bucket of the most preferred hotels), where the number on the markercluster shows how many markers this cluster contains, as in earlier prototypes. To further ease the inspection of preference inheritance of spatial hierarchy, recall the scenario *Find a Hotel Room in Heraklion* described in 3.3, where although you do not prefer an area (i.e. set the area of Heraklion Center as Worst) you do prefer the hotels close to POIs (i.e. set as Best the hotels close to Archaeological Museum), as shown in Figure 4.8. Note that in this scenario we show only a simple case of the interaction that uses a *best* preference and a single subarea. To the best of your knowledge, more complicated cases including multi subareas , as shown in Figure 4.7, and worst-best combination of preference actions are also implacable but omitted since they are considered trivial. We also evaluate and rely on the extend use of the *rectangle* shape. Although all shapes can be drawn and used as an input, the preference action is applicable only on this shape for simplicity reasons. It is feasible to assume that a scalability regarding other shapes e.g. circle and polygon is possible but out of the scope of this evaluation.

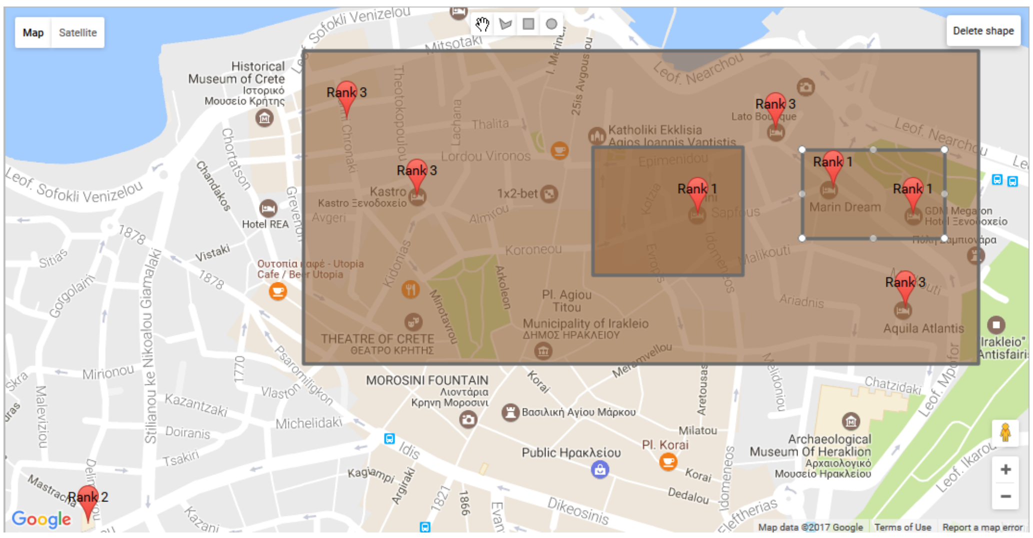


Figure 4.7: Auto conflict resolution over multi subareas within the same area.

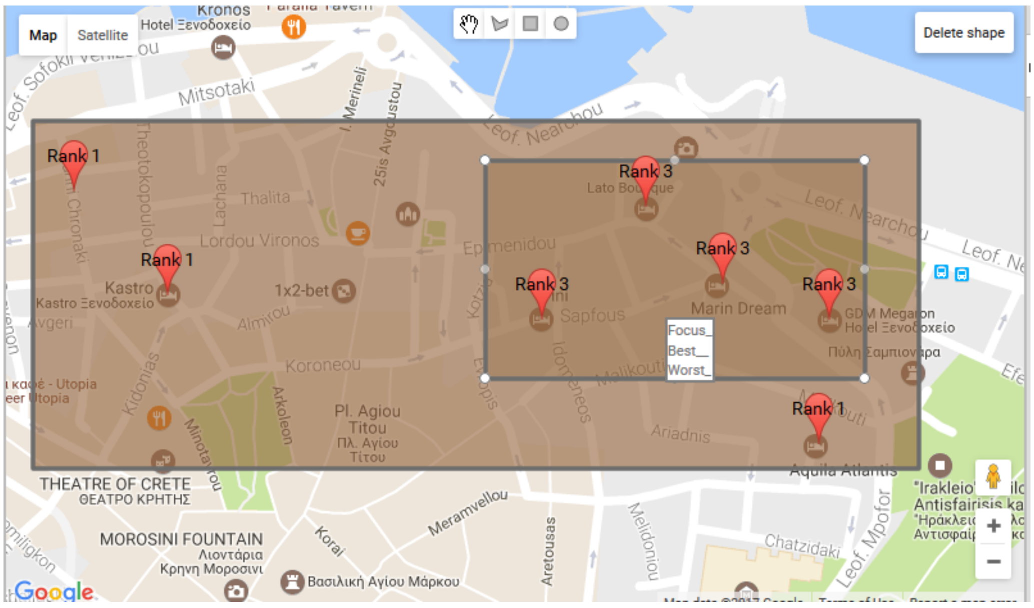


Figure 4.8: Auto conflict resolution of preferences over intersected areas.

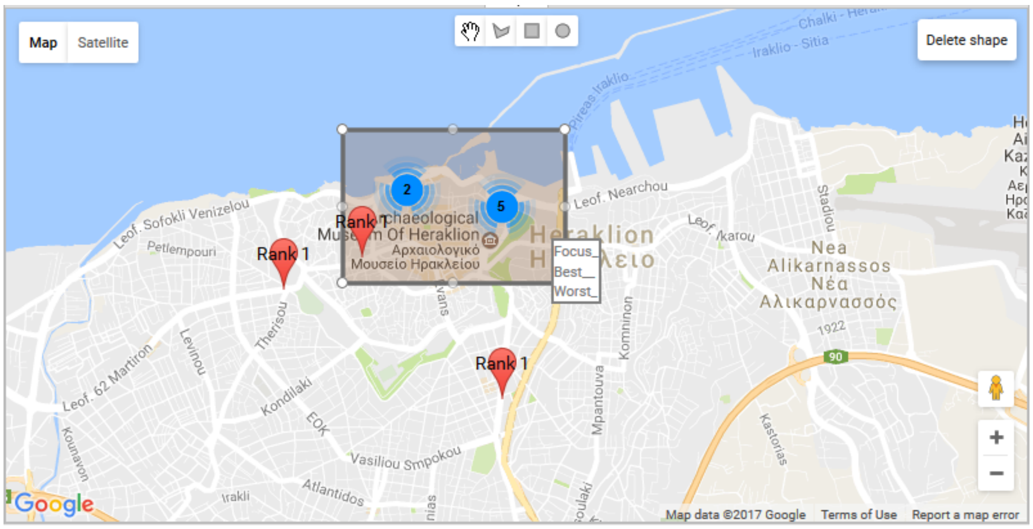
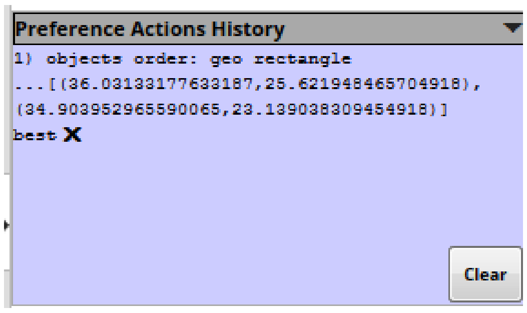


Figure 4.9: The right-click popup menu with the available actions.

4.2.3.1 The Client-Server model

Let as assume user draws a shape e.g. rectangle. Now the user has to pass as arguments the 2 points of this shape instead of the containing markers (eventually 4 edges-since a rectangle can be described using the North East (NE) and South West (SW) point). Based on the current implementation we support actions for the rectangle shape, although the principles are the same. Each shape can be described by it's geographical description and a label which is the type of the shape. An example using the rectangle shape is shown in the Figure 4.10.



```
1) objects order: geo rectangle
...[(36.03133177633187,25.621948465704918),
(34.903952965590065,23.139038309454918)]
best X
```

Figure 4.10: A best preference action based on the alternative method.

To this respect, we support 3 shapes:
(a) rectangle, which can spatial described by two points (i.e. the NE, SW edges).
(b) circle, which can spatial described by one point (i.e. the center) and the radius.
(c) polygon, which can be spatial described by sequence of the points of this polygon.

The grammar and the syntax are implied by the rules of the preference language described in [5]. We extended the preference language in order to fully support the geo aspect for all the predefined shapes. The extension is described in the Appendix A.2. To this respect, let us consider a best preference using the rectangle shape. We use this knowledge and the geo characteristics that describe the shape to send it to the server. Appendix A.3 gives an overview of the extended language.

4.2.4 Comparative Experimental Results

Here we report experimental results for comparing this method with the previous one. To this respect, Table 4.5 shows the average query times for preference regarding the alternative method as described previously. Notice that for more than 1,000 instances the system has an average preference time of 2.8 sec in contrast to the 98.4 sec of the previous method, giving promising results [Figure 4.11], i.e. this method is one order of magnitude faster than the previous one (specifically 35 times faster). To this respect, Figure 4.12 presents a comparison of those 2 methods regarding the preference action of the users, for the dataset 1. Finally, Figure 4.13 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 4.6 shows the results of the alternative method*method2* compared to the previous one (*method1*).

	Table: 2nd experiment Average Time-alterative		
Number of Action	*Dataset1*	*Dataset2*	*Dataset3*
User1	1880 ms	7868 ms	14792 ms
User2	1925 ms	8343 ms	51608 ms
User3	1637 ms	8932 ms	51403 ms
User4	1484 ms	8569 ms	53232 ms
User5	3007 ms	14424 ms	2396 ms
User6	2774 ms	15211 ms	82511 ms
User7	3709 ms	15484 ms	47586 ms
User8	3240 ms	15684 ms	103110 ms
User9	5682 ms	27535 ms	137783 ms

Table 4.5: Average query times for preference (alternative) of the simulated users over the *3 datasets*.

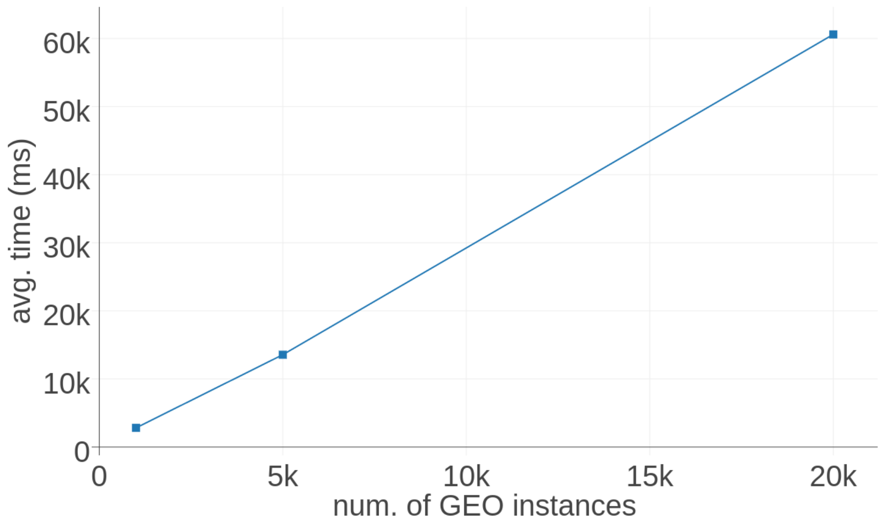


Figure 4.11: Average (alternative) preference time for the *3 datasets* for all the users.

Table: Method1 comparing to method2 (alternative)

	Method1			**Method2**		
Number of Action	*Dataset1*	*Dataset2*	*Dataset3*	*Dataset1*	*Dataset2*	*Dataset3*
User1	9794 ms	-	-	1880 ms	7868 ms	14792 ms
User2	14265 ms	-	-	1925 ms	8343 ms	51608 ms
User3	12469 ms	-	-	1637 ms	8932 ms	51403 ms
User4	11018 ms	-	-	1484 ms	8569 ms	53232 ms
User5	33871 ms	-	-	3007 ms	14424 ms	2396 ms
User6	114007 ms	-	-	2774 ms	15211 ms	82511 ms
User7	108983 ms	-	-	3709 ms	15484 ms	47586 ms
User8	522978 ms	-	-	3240 ms	15684 ms	103110 ms
User9	9794 ms	-	-	5682 ms	27535 ms	137783 ms

Table 4.6: *Method1* comparing to *method2(alternative)* over the *3 datasets* for *all users.*

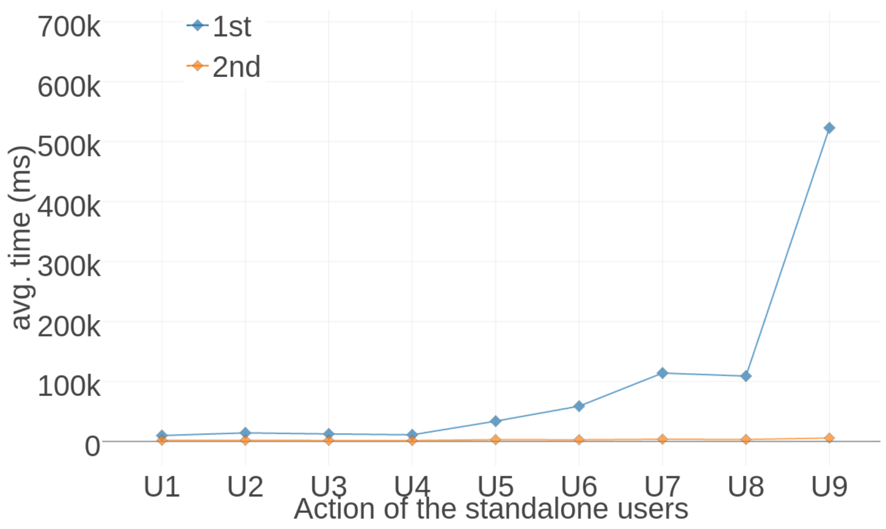


Figure 4.12: Average preference times of *dataset1* for each standalone user*(method1 compared to method2)*.

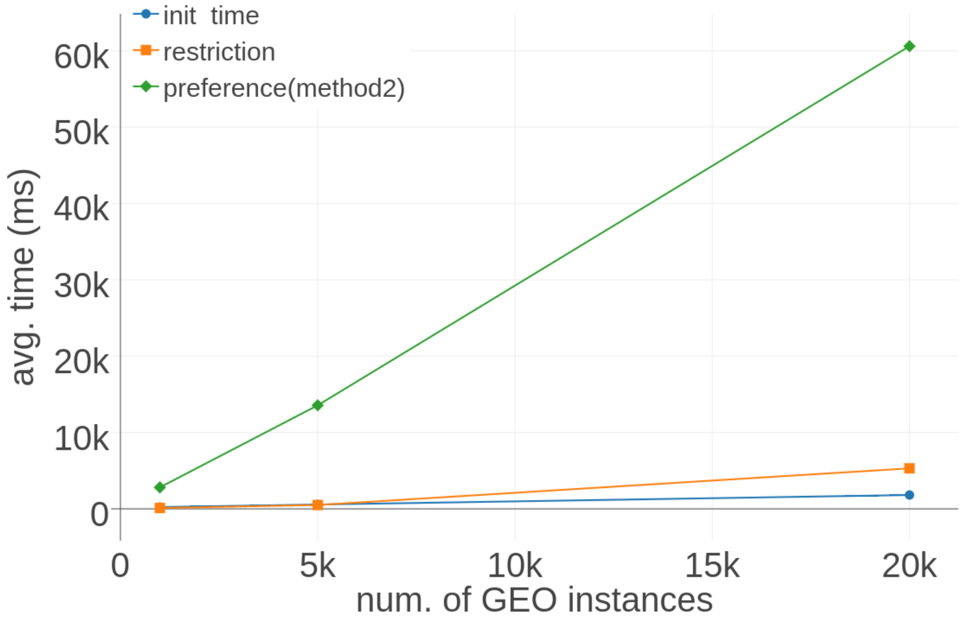


Figure 4.13: Average query times for the *3 metrics* of *3 datasets* for all users.

Chapter 5

Discussion and Open Issues

5.1 Evaluation With Users[1]

In contrast to the traditional preference actions as that described and evaluated in previous works [19], [20], [21] in our work we exploit the map as a new means for expressing these actions. To this respect, it is interesting to perform a comparative task-based evaluation in which users would have to explore and express their preferences and/or restrictions using both UIs. We have prepared a scenario (described below) comprising tasks. We plan to perform a preliminary evaluation with a few experts and then an evaluation with more users.

Purpose

We will conduct a task based evaluation with users similar to the running examples shown in 3.3. The objective is to investigate whether even in a small dataset such an example of 20 hotels, the addition of preferences to FDT in respect to the exploration of the spatial dimension would make users more satisfied with no a priori knowledge of the system, using simple actions with shapes over the map. To this end, we distinguish two different UIs, as shown in Figure 3.3:
a) UI_1 : Classical model of PFS interaction model
b) UI_2: PFS exploration using the map both as visualization and input for the set of the user's actions, as dictated by the proposed system.
The evaluation will be performed over both UIs, in order to extract the users' satisfaction level and exploit any possible problems and difficulties.

[1]This process is ongoing

Participants

According to [24], [25], [26] 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 *(at least 20 participants).*

Preparation and Training

To this respect, we have created a Google form as shown in Figure 5.2. Initially, the users were prompted to view a video tutorial related to each system and expect from the participants to fill the online questionnaire accordingly. Subsequently, they had to carry out the tasks derived from the scenario described.

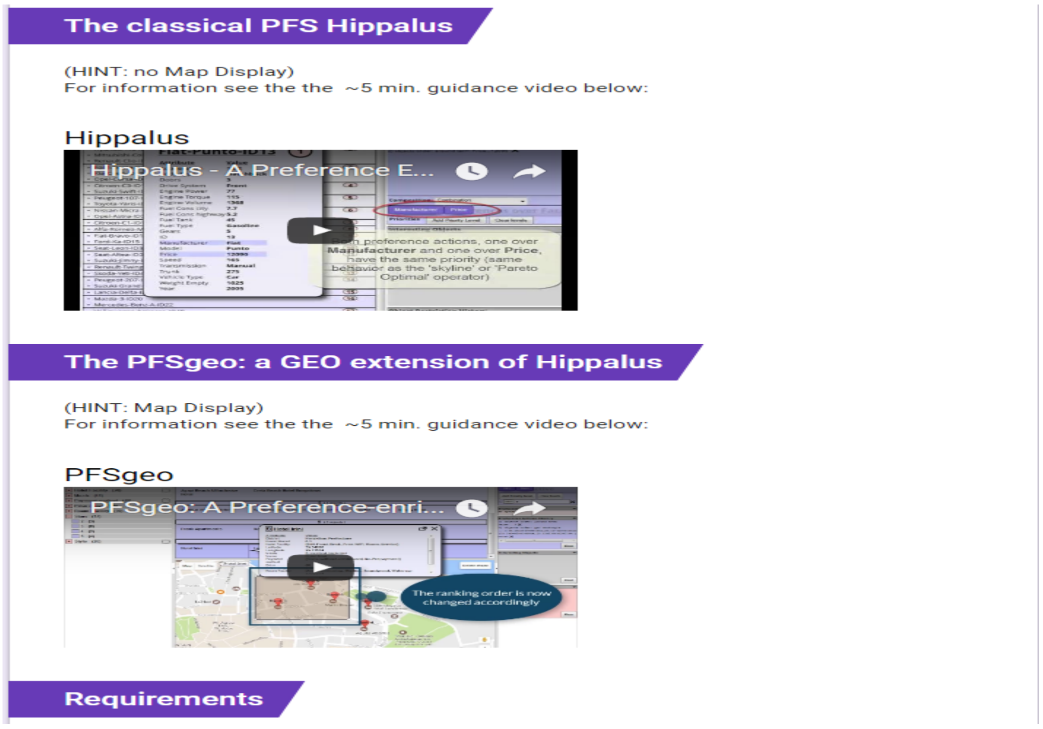
Tasks and Scenario

The tasks are based on the following scenario:
"Consider you are planning to stay in a hotel in the greater area of Heraklion, and you are interested in 3-stars hotels. In general you prefer hotels by the sea or close to Points of Interests (POIs), like museums, but not at the city center (because it can be noisy there)."
Table 5.1 lists the tasks used for this evaluation. In each task the user has to provide an answer corresponds to hotel or a set of hotels. The type of expected answer, is also listed in the same Table. We also ask from the evaluators to rate the overall performance of the proposed system by using the scale *<Useless, Not Useful, Neutral, Useful, Very Useful>* and also give some personal information about their profile and optional comments or suggestions.

Evaluation Tasks

Number of Task	*Description*	*Result*
1st	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.	Set of strings
2nd	You would like to visit the "Archaeological Museum of Heraklion" (located in the center of the city). Find the hotel that is closer to Museum and you prefer the most economical.	String
3rd	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.	String
4th	You are interested in "3-star" Hotels in Crete but you don't 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.	Set of strings

Table 5.1: Task Description of user's evaluation.



The classical PFS Hippalus

(HINT: no Map Display)
For information see the the ~5 min. guidance video below:

Hippalus

Hippalus - A Preference E...
preference actions, one over Manufacturer and one over Price, have the same priority (same behavior as the 'skyline' or 'Pareto Optimal' operator)

The PFSgeo: a GEO extension of Hippalus

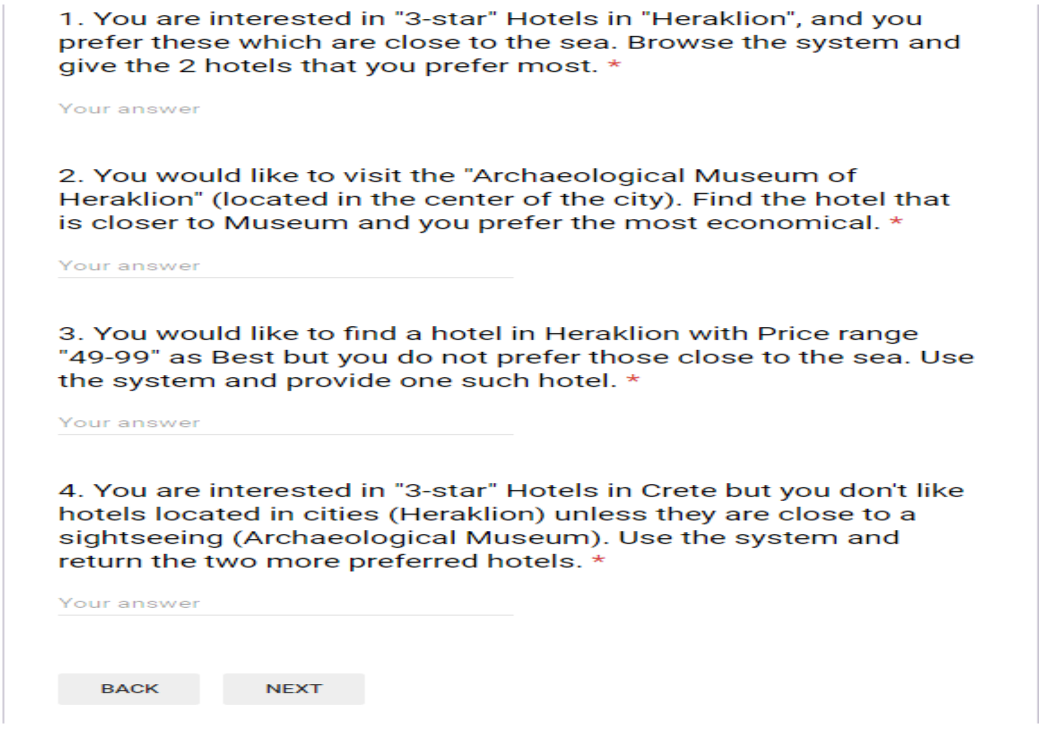
(HINT: Map Display)
For information see the the ~5 min. guidance video below:

PFSgeo

PFSgeo: A Preference-enri...
The ranking order is now changed accordingly

Requirements

(a)



1. 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. *

Your answer

2. You would like to visit the "Archaeological Museum of Heraklion" (located in the center of the city). Find the hotel that is closer to Museum and you prefer the most economical. *

Your answer

3. 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. *

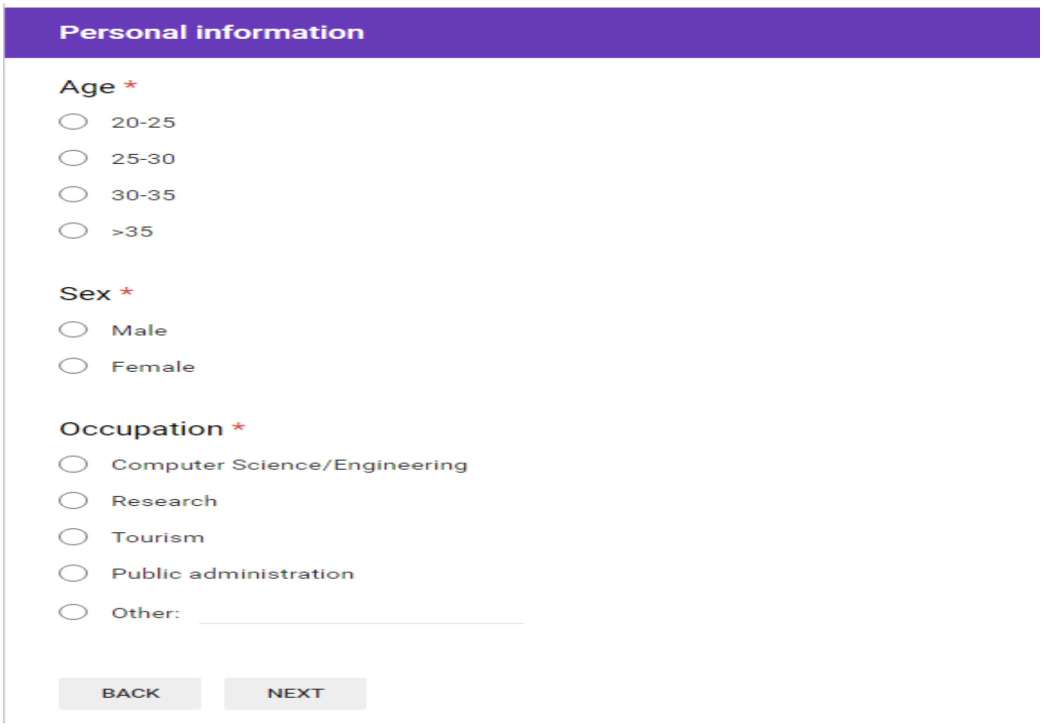
Your answer

4. You are interested in "3-star" Hotels in Crete but you don't 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. *

Your answer

(b)

Figure 5.1: The first section of the Google form 5.1a. The different tasks for each of the UIs 5.1b.



Personal information

Age *

- ◯ 20-25
- ◯ 25-30
- ◯ 30-35
- ◯ >35

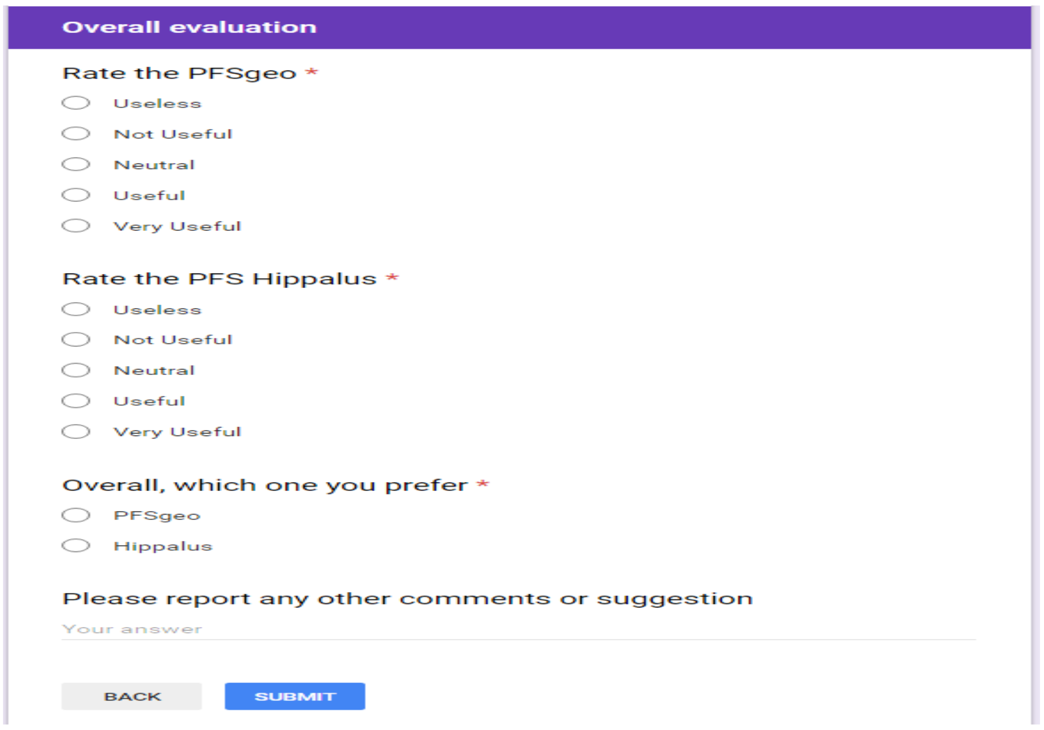
Sex *

- ◯ Male
- ◯ Female

Occupation *

- ◯ Computer Science/Engineering
- ◯ Research
- ◯ Tourism
- ◯ Public administration
- ◯ Other: ____

(a)



Overall evaluation

Rate the PFSgeo *

- ◯ Useless
- ◯ Not Useful
- ◯ Neutral
- ◯ Useful
- ◯ Very Useful

Rate the PFS Hippalus *

- ◯ Useless
- ◯ Not Useful
- ◯ Neutral
- ◯ Useful
- ◯ Very Useful

Overall, which one you prefer *

- ◯ PFSgeo
- ◯ Hippalus

Please report any other comments or suggestion

Your answer

(b)

Figure 5.2: Section with the personal information of the users 5.2a. Section of overall evaluation of the system 5.2b.

5.2 Optimization and Further Work

Here we elaborate more on the scalability of supporting preference-based ranking over large datasets.

As it has been suggested in [5], if the dataset is very big, then a reasonable policy is to compute the preference-based ranking only after the user has restricted the focus to a smaller number of objects (say 1,000 objects). If we follow this policy then no issue of scalability is expected since restrictions are very fast, and preference-based ranking is real time for up to 1,000 objects (this is true for all the 3 metrics). That means whenever the user focuses on a set with less objects than this threshold, the ranking is performed, whenever he zooms out to sets bigger than this threshold the ranking is not computed.

Nevertheless, if one would certainly like to produce a preference-based ranking of a very large collection of objects without first restricting his/her focus, then one could investigate several methods for speeding up Alg. 1 (specifically those parts of the algorithm whose complexity depends on $|A|$, i.e. Part 2 and Part 3). Some ideas follow:

(a) One method is to reduce the cost of Apply (in line 24 of Alg. 1) by returning only the top preferred elements. In this way topological sorting will be called only once. However the response would contain only the most preferred elements (not all the objects of the focus).

(b) Another approach would be to investigate applying a kind of fast clustering before carrying out Part 2 and Part 3 of the algorithm, so that to apply these parts not on A but on the representatives of the clusters of A. One method would be to divide the area using a grid and all objects that fall in the area of one cell of the grid to be represented by one artificial object. The set of artificial objects A' (which will be much smaller than $|A|$) can be ranked using Alg. 1. After that, we could rank only the actual objects that correspond to the area of the most preferred artificial object. In brief, this method requires, producing the set A' of artificial objects (linear time), running Alg. 1 on A', and finally running Alg 1 on the objects of the most preferred cell of the grid.

Based on our schema, we represent the spatial objects as shown in previous chapter. By that, points were the fundamental pillars of the geo aspect as an extension to our system. This assumes that although the points still remain at the bottom of the hierarchy level we can represent spatial objects using areas. An area can be divided in many sub-areas e.g. Center is a subarea of Heraklion, which respectively is a subarea of area of Crete. An area consists of a number of geographic objects (e.g. Hotels instances). For convention every area is called region and subareas as sub regions. We also make clear that given the complexity of the geographic real world entities a region can be further divided in many possible levels regarding the composition hierarchy. The decomposition size of an area depends of the level of resolution. We basically divide the map as a grid plane of fixed size cells. Once more, we use as an example the main land of the Region of Crete island. The representation of the original area is of prime importance and depends on the policies followed. For example, a possible representation could be based on the regional unions of Crete or on the municipalities units. It is obvious that the union of these cells will produce the regional ontology of Crete. Another method could be a grid representation of predefined cells of an area, a technique similarly used in evaluation.

Chapter 6

Concluding Remarks

A plethora of datasets contain GEOgraphic information or can be linked to GEOgraphic information. In this thesis we showed how a recently proposed method for exploratory search, the Preference-enriched Faceted Search (PFS) process, can be enriched for being appropriate for exploring datasets that contain geographical information.

The extended model, that we call PFSgeo, exploits geographical maps not only for displaying the focus objects during the interaction (as in previous works and systems), but also as an input means for restricting the focus and for defining preferences. Subsequently, we presented an implementation of the proposed model as an extension of the system Hippalus and we detailed how we used Google Maps. 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. Overall PFSgeo provides a generic and flexible 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. Finally we elaborated on performance issues and we provided measurements over synthetically produced datasets about hotels. Based on this analysis, we identified those tasks the affect the scalability of PFSgeo, and we have sketched a method that can be used for further improving the scalability of the approach.

Based on this analysis we provided an alternative method that is one order of magnitude faster.

6.1 Directions that are worth further work and research

Since this is the first work on extending Preference-enriched Faceted Search for geographical data, there are several directions that are worth further research including the design and comparative evaluation of methods for further improving scalability [Section 5.2], as well as task-based evaluations with users [Section 5.1] for evaluating and quantifying the benefits of the proposed extension in terms of task completion and user satisfaction in general.

Object Display

The markers of the clusters currently do not provide any information to the user about the rank of the clustered objects. One possible improvement would be to make the label of the cluster more informative, e.g. instead of *"(5)"* the cluster label could be *"(5)Rank2-3"* to indicate that the marker groups 5 objects whose rank fall in the interval *"[2,3]"*.

Ability to support arbitrary shapes

This can be supported easily in the map, however it would also require an extension of the algorithms. For instance the methods *IsInScope* and *CheckSubScopeOf* should be extended for covering any shape (not only rectangles). Various algorithms could be employed for this. In case this extension affects negatively the performance, one approach to tackle this problem would be to enrich the server-side with a spatial index (e.g. R-Tree, Quadtree, etc. [27]) that would index all shapes that have been used by the user and could then be exploited for deciding fast whether a point belongs to an area or whether an area is subarea of another.

Multi Dimensional geographical facets

Another extension is to consider datasets that contain more than one geographical facet, for instance if the objects are flights then each flight is characterized by two locations: the departure airport and the destination airport. The user interface could be extended for supporting more than one map frame, or for supporting methods for indicating on a single map different attributes in a clear manner.

Around preference on geographical domain

Since the geographical area is a continuous function it is worth investigating also radius-based preferences for the case of geographical data. For instance, if the user selects an area and issues a Best action, then all objects could be ranked according to their distance to the area (this is the analog of the AROUND preference for numerical facets, but here an extension of AROUND for the 2D space would be required).

Appendix A

Appendix

A.1 RDF Files

Running Example - Hotels.ttl

```
@prefix hippalus: <http://ics.forth.gr/isl/Hippalus/#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

hippalus:Location a rdfs:Class .

hippalus:Center
        a rdfs:Class ;
        rdfs:subClassOf hippalus:Heraklion .

hippalus:Ag.Pelagia
        a rdfs:Class ;
        rdfs:subClassOf hippalus:Heraklion .

hippalus:Heraklion
        a rdfs:Class ;
        rdfs:subClassOf hippalus:Crete .

hippalus:Crete
        a rdfs:Class ;
        rdfs:subClassOf hippalus:Greece .
```

```
hippalus:Greece
        a rdfs:Class ;
        rdfs:subClassOf hippalus:Location .

hippalus:Kastro_Hotel_Double
        a hippalus:Hippalus_Id , hippalus:Center ;
        hippalus:Accessories "Free−WiFi"^^xsd:string , "Flat−screen−TV"^^xsd:string ,
        "Air−conditioning"^^xsd:string ;
        hippalus:Address "Theotokopoulou_22"^^xsd:string ;
        hippalus:Conditions "Special−conditions"^^xsd:string , "Breakfast−included"^^xsd:string ;
        hippalus:Latitude "35.341094"^^xsd:float ;
        hippalus:Longitude "25.132406"^^xsd:float ;
        hippalus:Name "Kastro_Hotel"^^xsd:string ;
        hippalus:Persons "2"^^xsd:int ;
        hippalus:Price_4_nights_in_Euros
                "340"^^xsd:int ;
        hippalus:Rating "9.0"^^xsd:float ;
        hippalus:Room "Double"^^xsd:string .

hippalus:Name a rdf:Property ;
        rdfs:domain hippalus:Hippalus_Id ;
        rdfs:range xsd:string .

hippalus:Latitude a rdf:Property ;
        rdfs:domain hippalus:Hippalus_Id ;
        rdfs:range xsd:float .

hippalus:Address a rdf:Property ;
        rdfs:domain hippalus:Hippalus_Id ;
        rdfs:range xsd:string .

hippalus:Lato_Boutique_Hotel
        a hippalus:Hippalus_Id , hippalus:Center ;
        hippalus:Accessories "Flat−screen−TV"^^xsd:string , "Air−conditioning"^^xsd:string ,
         "Free−WiFi"^^xsd:string ;
        hippalus:Address "Epimenidou_15, "^^xsd:string ;
        hippalus:Conditions "Breakfast−included"^^xsd:string , "Free−cancellation"^^xsd:string ,
         "Pay−later"^^xsd:string ;
        hippalus:Latitude "35.341703"^^xsd:float ;
        hippalus:Longitude "25.136622"^^xsd:float ;
        hippalus:Name "Lato_Boutique_Hotel"^^xsd:string ;
```

```
          hippalus:Persons "1"^^xsd:int ;
          hippalus:Price_4_nights_in_Euros
                    "240"^^xsd:int ;
          hippalus:Rating "9.0"^^xsd:float ;
          hippalus:Room "Single"^^xsd:string ;
          hippalus:Stars "3"^^xsd:int .

hippalus:Marin_Dream_Hotel
          a hippalus:Hippalus_Id , hippalus:Center ;
          hippalus:Accessories "Free−WiFi"^^xsd:string , "Air−conditioning"^^xsd:string ,
          "Flat−screen−TV"^^xsd:string ;
          hippalus:Address "Epimenidou_46,"^^xsd:string ;
          hippalus:Conditions "Free−cancellation"^^xsd:string ,
            "Breakfast−included"^^xsd:string ,
            "Pay−later"^^xsd:string ;
          hippalus:Latitude "35.341201"^^xsd:float ;
          hippalus:Longitude "25.137219"^^xsd:float ;
          hippalus:Name "Marin_Dream_Hotel"^^xsd:string ;
          hippalus:Persons "2"^^xsd:int ;
          hippalus:Price_4_nights_in_Euros
                    "380"^^xsd:int ;
          hippalus:Rating "9.0"^^xsd:float ;
          hippalus:Room "Double"^^xsd:string ;
          hippalus:Stars "3"^^xsd:int .

hippalus:Longitude a rdf:Property ;
          rdfs:domain hippalus:Hippalus_Id ;
          rdfs:range xsd:float .

hippalus:Room a rdf:Property ;
          rdfs:domain hippalus:Hippalus_Id ;
          rdfs:range xsd:string .

hippalus:Megaron_Hotel_Luxury_Double_Non_Refundable
          a hippalus:Hippalus_Id , hippalus:Center ;
          hippalus:Accessories "Flat−screen−TV"^^xsd:string , "Sea−view"^^xsd:string ,
            "Air−conditioning"^^xsd:string , "Balcony"^^xsd:string ,
            "Free−WiFi"^^xsd:string ,
            "Bathtub"^^xsd:string , "32−m2"^^xsd:string ;
          hippalus:Address "D.Beaufort_9"^^xsd:string ;
          hippalus:Conditions "Non−refundable"^^xsd:string ,
        "Buffet−breakfast−included"^^xsd:string ;
        hippalus:Latitude "35.340932"^^xsd:float ;
        hippalus:Longitude "25.138105"^^xsd:float ;
        hippalus:Name "Megaron_Hotel"^^xsd:string ;
        hippalus:Persons "2"^^xsd:int ;
        hippalus:Price_4_nights_in_Euros
                "828"^^xsd:int ;
        hippalus:Rating "9.5"^^xsd:float ;
        hippalus:Room "Luxury/Double"^^xsd:string ;
        hippalus:Stars "5"^^xsd:int .

hippalus:Stars a rdf:Property ;
        rdfs:domain hippalus:Hippalus_Id ;
        rdfs:range xsd:integer .

hippalus:Hippalus_Id a rdfs:Class .

hippalus:Galaxy_Hotel_Iraklio
        a hippalus:Hippalus_Id , hippalus:Center ;
        hippalus:Accessories "Sea−view"^^xsd:string , "Flat−screen−TV"^^xsd:string ,
        "Bathtub"^^xsd:string , "Balcony"^^xsd:string , "Free−WiFi"^^xsd:string ,
        "Air−conditioning"^^xsd:string , "32−m2"^^xsd:string ;
        hippalus:Address "Dimokratias_75"^^xsd:string ;
        hippalus:Conditions "Pay−later"^^xsd:string , "Free−cancellation"^^xsd:string ;
        hippalus:Latitude "35.330466"^^xsd:float ;
        hippalus:Longitude "25.138276"^^xsd:float ;
        hippalus:Name "Galaxy_Hotel_Iraklio"^^xsd:string ;
        hippalus:Persons "1"^^xsd:int ;
        hippalus:Price_4_nights_in_Euros
                "760"^^xsd:int ;
        hippalus:Rating "9.5"^^xsd:float ;
        hippalus:Room "Luxury/Single"^^xsd:string ;
        hippalus:Stars "5"^^xsd:int .

hippalus:Rating a rdf:Property ;
        rdfs:domain hippalus:Hippalus_Id ;
        rdfs:range xsd:float .

hippalus:Price_4_nights_in_Euros
        a rdf:Property ;
        rdfs:domain hippalus:Hippalus_Id ;
        rdfs:range xsd:integer .

hippalus:Conditions a rdf:Property ;
        rdfs:domain hippalus:Hippalus_Id ;
        rdfs:range xsd:string .

hippalus:Castello_City_Hotel
        a hippalus:Hippalus_Id , hippalus:Center ;
        hippalus:Accessories "Free-WiFi"^^xsd:string , "Air-conditioning"^^xsd:string ,
        "Flat-screen-TV"^^xsd:string ;
        hippalus:Address "Leof_62_Martiron_1"^^xsd:string ;
        hippalus:Conditions "Breakfast-included"^^xsd:string ,
        "Special-conditions"^^xsd:string ;
        hippalus:Latitude "35.336291"^^xsd:float ;
        hippalus:Longitude "25.123639"^^xsd:float ;
        hippalus:Name "Castello_City_Hotel"^^xsd:string ;
        hippalus:Persons "1"^^xsd:int ;
        hippalus:Price_4_nights_in_Euros
                "240"^^xsd:int ;
        hippalus:Rating "9.0"^^xsd:float ;
        hippalus:Room "Single"^^xsd:string ;
        hippalus:Stars "3"^^xsd:int .

hippalus:Renia_Hotel_Appartments_Standard_Studio_Triple
        a hippalus:Hippalus_Id , hippalus:Ag.Pelagia ;
        hippalus:Accessories "30-m2"^^xsd:string , "Balcony"^^xsd:string ,
        "Pool-and-City-view"^^xsd:string ;
        hippalus:Address "Arkadiou"^^xsd:string ;
        hippalus:Conditions "Breakfast-included"^^xsd:string ,
        "Non-refundable"^^xsd:string ;
        hippalus:Latitude "35.406877"^^xsd:float ;
        hippalus:Longitude "25.016027"^^xsd:float ;
        hippalus:Name "Renia_Hotel_Appartments"^^xsd:string ;
        hippalus:Persons "3"^^xsd:int ;
        hippalus:Price_4_nights_in_Euros
                "216"^^xsd:int ;
        hippalus:Rating "7.7"^^xsd:float ;
        hippalus:Room "Standard_Studio/Triple"^^xsd:string ;
        hippalus:Stars "3"^^xsd:int .

hippalus:Iraklion_Hotel
```

```
          a hippalus:Hippalus_Id , hippalus:Center ;
          hippalus:Accessories "Balcony"^^xsd:string , "Free-WiFi"^^xsd:string ,
          "Bathtub"^^xsd:string , "Sea-view"^^xsd:string ,
          "Flat-screen-TV"^^xsd:string ,
          "Air-conditioning"^^xsd:string , "32-m2"^^xsd:string ;
          hippalus:Address "Kalokerinou_126"^^xsd:string ;
          hippalus:Conditions "Buffet-breakfast-included"^^xsd:string ,
          "Free-cancellation"^^xsd:string , "Pay-later"^^xsd:string ;
          hippalus:Latitude "35.338139"^^xsd:float ;
          hippalus:Longitude "25.128923"^^xsd:float ;
          hippalus:Name "Iraklion Hotel"^^xsd:string ;
          hippalus:Persons "2"^^xsd:int ;
          hippalus:Price_4_nights_in_Euros
                    "992"^^xsd:int ;
          hippalus:Rating "9.5"^^xsd:float ;
          hippalus:Room "Luxury/Double"^^xsd:string ;
          hippalus:Stars "5"^^xsd:int .

hippalus:Accessories a rdf:Property ;
          rdfs:domain hippalus:Hippalus_Id ;
          rdfs:range xsd:string .

hippalus:Persons a rdf:Property ;
          rdfs:domain hippalus:Hippalus_Id ;
          rdfs:range xsd:integer .
```

A.2 Complete Syntax of GEO extended Preference Language

preferencesFDT.g4 antlr

```
/*
 * Hippalus - A Preference-Enriched Faceted Exploratory System
 *
 * FOUNDATION OF RESEARCH AND TECHNOLOGY - HELLAS (FORTH-ICS)
 *
 * INFORMATION SYSTEMS LABORATORY (ISL)
 *
 * http://www.ics.forth.gr/isl
 *
 * LICENCE: TO BE ADDED
 *
 * Copyright 2012−2016
 *
 */

grammar preferencesFDT;


//@header {
//package preferencesFDT.parser;
//import java.util.HashMap;
//}

@lexer::header{
    package gr.forth.ics.isl.hippalus.pfs.parser;
}

@parser::header{
    package gr.forth.ics.isl.hippalus.pfs.parser;
    import gr.forth.ics.isl.hippalus.pfs.actions.∗;
    import gr.forth.ics.isl.hippalus.pfs.∗;
    import java.net.URL;
    import java.io.IOException;
    import org.apache.log4j.Logger;
}

@parser::members {
  // Initialize class which holds preference related things
  private PreferencesFDT prefsFDT;
  // Holding what we order
  private Order order;
  // Holding anchor
  private Anchor anchor;
  // Hold the anchor id
  private String anchorID;
  // Holds the current preference action
  private PreferenceAction currentAction;
  // Holds the current preference id
  private int id = 0;
  // Holds if this action is to add or remove
  private boolean remove = false;

  private static final Logger logger = Logger.getLogger("PreferencesParserG");

  // Enum for what we order
  public enum Order {
      FACETS, TERMS, OBJECTS, NOTHING;
  };

  // Enum for where we anchor the action
  public enum Anchor {
      FACET, TERM, OBJECT, NOTHING, GEO;
  };

  // Set what we order
  public void setOrder(Order order) {
          this.order = order;
  }

  // Set where we anchor
  public void setAnchor(Anchor anchor) {
          this.anchor = anchor;
  }

  public void setPreferencesFDT(PreferencesFDT current) {
    prefsFDT = current;
  }

}


// LEXER
// Keywords

  AROUND : 'around';
  BEST : 'best';
  COLON : ':';
  COMMA : ',';
  COUNT : 'count';
  DISTFUNCTION : 'distFunction';
  FACET : 'facet';
  FACETS : 'facets';
  INTERVAL : 'interval';
  LEXICOGRAPHIC : 'lexicographic';
  MIN : 'min';
  MAX : 'max';
  NOT : 'not';
  OBJECT : 'object';
  OBJECTS : 'objects';
  ORDER : 'order';
  PARENTHESES : '()';
  PARETO : 'pareto';
  PREFER : 'prefer';
  PRIORITY : 'priority';
  REMOVE: 'remove';
  SCOREFUNCTION : 'scorefunction';
  SKYLINE : 'skyline';
  TERM : 'term';
  TERMS : 'terms';
  TO : 'to';
  VALUE : 'value';
  WORST : 'worst';
  GEO : 'geo';
  POINT : 'point';
  RECTANGLE : 'rectangle';
  POLYGON : 'polygon';
  CIRCLE : 'circle';

/*----------------------------------------
 * PARSER RULES
 *----------------------------------------*/

// Starting point
pref: (prefline NEWLINE)* {
            // Store Preference Action to prefsFDT since it has been parsed
            currentAction.setAction($prefline.text); // Store action string
            //currentAction.print(); // Print the action currently added
            if(!remove)
                prefsFDT.addPreference(currentAction);
            else
                prefsFDT.removePreference(currentAction, false);
```

```
                              }
| prefline EOF {
               // Store Preference Action to prefsFDT since it has been parsed
               currentAction.setAction($prefline.text); // Store action string
               //currentAction.print(); // Print the action currently added
               if(!remove)
                    prefsFDT.addPreference(currentAction);
               else
                    prefsFDT.removePreference(currentAction, false);
               }
;

prefline: scopeType spec {remove = false;}
          | scopeType composition {remove = false;}
          | remove scopeType spec {}
          | remove scopeType composition {}
;

remove: REMOVE COLON {remove = true;}
;

scopeType: FACETS ORDER COLON {
                                   setOrder(Order.FACETS);
                                 }
               | TERMS ORDER COLON {
                                   setOrder(Order.TERMS);
                                 }
               | OBJECTS ORDER COLON {
                                   setOrder(Order.OBJECTS);
                                 }
;

spec: anchor rankSpec {}
;

specList: spec (COMMA spec)∗ {}
;

anchor: FACET ID {
                 setAnchor(Anchor.FACET);
```

```
                    anchorID = $ID.text;
                }
| TERM ID {
                    setAnchor(Anchor.TERM);
                    anchorID = $ID.text;
               }
| OBJECT ID {
                    setAnchor(Anchor.OBJECT);
                    anchorID = $ID.text;
                }
| GEO RECTANGLE TWO_POINTS_LIST {
                    setAnchor(Anchor.GEO);
                    anchorID = $TWO_POINTS_LIST.text;
                }
| GEO POLYGON N_POINTS_LIST {
                    setAnchor(Anchor.GEO);
                    anchorID = $N_POINTS_LIST.text;
                }
| GEO CIRCLE CIRCLE_POINTS_LIST {
                    setAnchor(Anchor.GEO);
                    anchorID = $CIRCLE_POINTS_LIST.text;
                }
| GEO POINT A_POINT {
                    setAnchor(Anchor.GEO);
                    anchorID = $A_POINT.text;
                }
| TERM INTERVAL ID ID {}
| {}// empty
;

rankSpec: BEST
{
  // ADD THIS TO THE SET OF BEST
  // Check what we order
  if(order == Order.FACETS) {
    // FACET ORDERING
    if(anchor == Anchor.FACET) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.BEST, anchorID);
      currentAction.setId(id);
      currentAction.setActionsOrderFacets();
      currentAction.setAnchorFacet();
    } else if (anchor == Anchor.TERM){
        // DO NOTHING, we got here wrongly
        logger.warn("PARSER ERROR: We do not support best for facet ordering actions
          anchored to terms");
    } else if (anchor == Anchor.OBJECT){
        // DO NOTHING, we got here wrongly
        logger.warn("PARSER ERROR: We do not support best for facet ordering actions
          anchored to objects");
    }
} // TERMS ORDERING
else if(order == Order.TERMS) {
    // FACET ORDERING
    if(anchor == Anchor.FACET) {
        // DO NOTHING, we got here wrongly
        logger.warn("PARSER ERROR: We do not support best for term ordering actions
          anchored to facets");
    } else if (anchor == Anchor.TERM){
        currentAction = new PreferenceAction(PreferenceAction.TYPE.BEST, anchorID.
        split("\\.\\.\\.")[1]);
        currentAction.setId(id);
        currentAction.setActionsOrderTerms(anchorID.split("\\.\\.\\.")[0]);
        currentAction.setAnchorTerm();
    } else if (anchor == Anchor.OBJECT){
        logger.warn("PARSER ERROR: We do not support best for term ordering actions
          anchored to objects");
    }
} // OBJECTS ORDERING
else if(order == Order.OBJECTS) {
    // FACET ORDERING
    if(anchor == Anchor.FACET) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.BEST, anchorID);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects(anchorID);
        currentAction.setAnchorFacet();
    } else if (anchor == Anchor.TERM){
        currentAction = new PreferenceAction(PreferenceAction.TYPE.BEST, anchorID.
        split("\\.\\.\\.")[1]);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects(anchorID.split("\\.\\.\\.")[0]);
        currentAction.setAnchorTerm();
    } else if (anchor == Anchor.OBJECT){
        currentAction = new PreferenceAction(PreferenceAction.TYPE.BEST, anchorID);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects();
        currentAction.setAnchorObject();
      } else if (anchor == Anchor.GEO){
        currentAction = new PreferenceAction(PreferenceAction.TYPE.BEST, anchorID.
        split("\\.\\.\\.")[1]);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects("GEO");
        currentAction.setAnchorGeo();
      }
    }
    id++;
  }
  | COUNT MIN {
    if(order == Order.FACETS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.COUNT_MIN);
        currentAction.setId(id);
        currentAction.setActionsOrderFacets();
        currentAction.setAnchorNone();
    }
    else if(order == Order.TERMS) {
    if(anchor == Anchor.FACET) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.COUNT_MIN,
      anchorID;
      currentAction.setId(id);
      currentAction.setActionsOrderTerms(anchorID);
      currentAction.setAnchorFacet();
    } else {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.COUNT_MIN);
      currentAction.setId(id);
      currentAction.setActionsOrderTerms();
      currentAction.setAnchorNone();
      }
    }
    id++;
  }
  | COUNT MAX {
    if(order == Order.FACETS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.COUNT_MAX);
        currentAction.setId(id);
        currentAction.setActionsOrderFacets();
        currentAction.setAnchorNone();
    }
    else if(order == Order.TERMS) {
      if(anchor == Anchor.FACET) {
          currentAction = new PreferenceAction(PreferenceAction.TYPE.COUNT_MAX,
            anchorID);
          currentAction.setId(id);
          currentAction.setActionsOrderTerms(anchorID);
          currentAction.setAnchorFacet();
        } else {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.COUNT_MAX);
        currentAction.setId(id);
        currentAction.setActionsOrderTerms();
        currentAction.setAnchorNone();
      }
   }
    id++;
}
| DISTFUNCTION ID PARENTHESES MIN {}
| DISTFUNCTION ID PARENTHESES MAX {}
| LEXICOGRAPHIC MIN {
    if(order == Order.FACETS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MIN);
        currentAction.setId(id);
        currentAction.setActionsOrderFacets();
        currentAction.setAnchorNone();
    }
    else if(order == Order.TERMS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MIN);
        currentAction.setId(id);
        currentAction.setActionsOrderTerms();
        currentAction.setAnchorNone();
    }
    else if(order == Order.OBJECTS) {
      if(anchor == Anchor.FACET) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MIN,
          anchorID);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects();
        currentAction.setAnchorFacet();
    } else {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MIN);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects();
      currentAction.setAnchorNone();
    }
  }
  id++;
}
| LEXICOGRAPHIC MAX {
  if(order == Order.FACETS) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MAX);
      currentAction.setId(id);
      currentAction.setActionsOrderFacets();
      currentAction.setAnchorNone();
  }
  else if(order == Order.TERMS) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MAX);
      currentAction.setId(id);
      currentAction.setActionsOrderTerms();
      currentAction.setAnchorNone();
  }
  else if(order == Order.OBJECTS) {
    if(anchor == Anchor.FACET) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MAX,
       anchorID);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects();
      currentAction.setAnchorFacet();
    } else {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.LEX_MAX);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects();
      currentAction.setAnchorNone();
    }
  }
  id++;
}
| PREFER FACET better=ID TO than=ID
{
  anchor = Anchor.FACET;
  anchorID = $better.text;

  // ADD THIS TO THE SET OF RELATIVE
  // Check what we order
  if(order == Order.FACETS) {
    // FACET ORDERING
        currentAction = new PreferenceAction(PreferenceAction.TYPE.RELATIVE,
         $better.text, $than.text);
        currentAction.setId(id);
        currentAction.setActionsOrderFacets();
        currentAction.setAnchorFacet();
  } // TERMS ORDERING
  else if(order == Order.TERMS) {
        logger.warn("PARSER ERROR: We do not support relative preference
           for term ordering actions anchored to facets");
        //prefsFDT.addRelative(anchor, $better.text, $than.text, order);
  } // OBJECTS ORDERING
  else if(order == Order.OBJECTS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.RELATIVE,
         $better.text, $than.text);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects();
        currentAction.setAnchorFacet();
  }
  id++;
}
| PREFER TERM better=ID TO than=ID
{
  anchorID = $better.text;
  anchor = Anchor.TERM;

  // ADD THIS TO THE SET OF RELATIVE
  // Check what we order
  if(order == Order.FACETS) {
        // DO NOTHING, we got here wrongly
        logger.warn("PARSER ERROR: We do not support relative preference
for
         facet ordering actions anchored to terms");
  } // TERMS ORDERING
  else if(order == Order.TERMS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.RELATIVE,
         $better.text.split("\\.\\.\\.")[1], $than.text.split("\\.\\.\\.")[1]);
        currentAction.setId(id);
        currentAction.setActionsOrderTerms($better.text.split("\\.\\.\\.")[0]);
        currentAction.setAnchorTerm();
    } // OBJECTS ORDERING
    else if(order == Order.OBJECTS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.RELATIVE,
         $better.text.split("\\.\\.\\.")[1], $than.text.split("\\.\\.\\.")[1]);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects($better.text.split("\\.\\.\\.")[0]);
        currentAction.setAnchorTerm();
    }
    id++;
}
| PREFER OBJECT better=ID TO than=ID
{
    anchorID = $better.text;
    anchor = Anchor.OBJECT;

    // ADD THIS TO THE SET OF RELATIVE
    // Check what we order
    if(order == Order.FACETS) {
        // DO NOTHING, we got here wrongly
        logger.warn("PARSER ERROR: We do not support relative preference
for
         facet ordering actions anchored to objects");
    } // TERMS ORDERING
    else if(order == Order.TERMS) {
        // DO NOTHING, we got here wrongly
        logger.warn("PARSER ERROR: We do not support relative preference
for
         term ordering actions anchored to objects");
    } // OBJECTS ORDERING
    else if(order == Order.OBJECTS) {
        currentAction = new PreferenceAction(PreferenceAction.TYPE.RELATIVE,
         $better.text, $than.text);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects();
        currentAction.setAnchorObject();
    }
    id++;
```

```
}
| SCOREFUNCTION ID PARENTHESES MIN {}
| SCOREFUNCTION ID PARENTHESES MAX {}
| AROUND TERM termAround=ID
{
  if(order == Order.OBJECTS) {
       currentAction = new PreferenceAction(PreferenceAction.TYPE.AROUND,
        $termAround.text.split("\\.\\.\\.")[0]);
       currentAction.setId(id);
       currentAction.setActionsOrderObjects($termAround.text.split("\\.\\.\\.")[0]);
       currentAction.setAroundTerm($termAround.text.split("\\.\\.\\.")[1]);
       currentAction.setAnchorFacet();
     }

  id++;
}
| NOT AROUND TERM termNotAround=ID
{
  if(order == Order.OBJECTS) {
       currentAction = new PreferenceAction(PreferenceAction.TYPE.NOT_AROUND,
        $termNotAround.text.split("\\.\\.\\.")[0]);
       currentAction.setId(id);
       currentAction.setActionsOrderObjects($termNotAround.text.split("\\.\\.\\.")[0]);
       currentAction.setAroundTerm($termNotAround.text.split("\\.\\.\\.")[1]);
       currentAction.setAnchorFacet();
     }
  id++;
}
| VALUE MIN {
  if(order == Order.FACETS) {
       currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MIN);
       currentAction.setId(id);
       currentAction.setActionsOrderFacets();
       currentAction.setAnchorNone();
  }
  else if(order == Order.TERMS) {
    if(anchor == Anchor.FACET) {
         currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MIN,
          anchorID);
         currentAction.setId(id);
         currentAction.setActionsOrderTerms(anchorID);
```

```
            currentAction.setAnchorFacet();
        } else {
            currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MIN);
            currentAction.setId(id);
            currentAction.setActionsOrderTerms();
            currentAction.setAnchorNone();
        }
    }
    else if(order == Order.OBJECTS) {
        if(anchor == Anchor.FACET) {
            currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MIN,
             anchorID);
            currentAction.setId(id);
            currentAction.setActionsOrderObjects(anchorID);
            currentAction.setAnchorFacet();
        }
        else if(anchor == Anchor.TERM) {
            currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MIN,
             anchorID.split("\\.\\.\\.")[1]);
            currentAction.setId(id);
            currentAction.setActionsOrderObjects(anchorID.split("\\.\\.\\.")[0]);
            currentAction.setAnchorTerm();
        }
    }
    id++;
}
| VALUE MAX {
    if(order == Order.FACETS) {
            currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MAX);
            currentAction.setId(id);
            currentAction.setActionsOrderFacets();
            currentAction.setAnchorNone();
    }
    else if(order == Order.TERMS) {
        if(anchor == Anchor.FACET) {
                currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MAX,
                 anchorID);
                currentAction.setId(id);
                currentAction.setActionsOrderTerms(anchorID);
                currentAction.setAnchorFacet();
            } else {
```

```
        currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MAX);
        currentAction.setId(id);
        currentAction.setActionsOrderTerms();
        currentAction.setAnchorNone();
      }
  }
  else if(order == Order.OBJECTS) {
    if(anchor == Anchor.FACET) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MAX,
        anchorID);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects(anchorID);
      currentAction.setAnchorFacet();
    }
    else if(anchor == Anchor.TERM) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.VALUE_MAX,
      anchorID.split("\\.\\.\\.")[1]);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects(anchorID.split("\\.\\.\\.")[0]);
      currentAction.setAnchorTerm();
    }
  }
  id++;
}
| WORST
{
  // ADD THIS TO THE SET OF WORST
  // Check what we order
  if(order == Order.FACETS) {
    // FACET ORDERING
    if(anchor == Anchor.FACET) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.WORST, anchorID);
      currentAction.setId(id);
      currentAction.setActionsOrderFacets();
      currentAction.setAnchorFacet();
    } else if (anchor == Anchor.TERM){
      // DO NOTHING, we got here wrongly
      logger.warn("PARSER ERROR: We do not support worst for facet ordering
        actions anchored to terms");
    } else if (anchor == Anchor.OBJECT){
      // DO NOTHING, we got here wrongly
      logger.warn("PARSER ERROR: We do not support worst for facet ordering
       actions anchored to objects");
    }
  } // TERMS ORDERING
  else if(order == Order.TERMS) {
    // FACET ORDERING
    if(anchor == Anchor.FACET) {
      // DO NOTHING, we got here wrongly
      logger.warn("PARSER ERROR: We do not support worst for term ordering
       actions anchored to facets");
    } else if (anchor == Anchor.TERM){
      currentAction = new PreferenceAction(PreferenceAction.TYPE.WORST, anchorID.
      split("\\.\\.\\.")[1]);
      currentAction.setId(id);
      currentAction.setActionsOrderTerms(anchorID.split("\\.\\.\\.")[0]);
      currentAction.setAnchorTerm();
    } else if (anchor == Anchor.OBJECT){
      logger.warn("PARSER ERROR: We do not support worst for term ordering actions
       anchored to objects");
    }
  } // OBJECTS ORDERING
  else if(order == Order.OBJECTS) {
    // FACET ORDERING
    if(anchor == Anchor.FACET) {
      currentAction = new PreferenceAction(PreferenceAction.TYPE.WORST, anchorID);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects(anchorID);
      currentAction.setAnchorFacet();
    } else if (anchor == Anchor.TERM){
      currentAction = new PreferenceAction(PreferenceAction.TYPE.WORST, anchorID.
      split("\\.\\.\\.")[1]);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects(anchorID.split("\\.\\.\\.")[0]);
      currentAction.setAnchorTerm();
    } else if (anchor == Anchor.OBJECT){
      currentAction = new PreferenceAction(PreferenceAction.TYPE.WORST, anchorID);
      currentAction.setId(id);
      currentAction.setActionsOrderObjects();
      currentAction.setAnchorObject();
    } else if (anchor == Anchor.GEO){
      currentAction = new PreferenceAction(PreferenceAction.TYPE.WORST, anchorID.
        split("\\.\\.\\.")[1]);
        currentAction.setId(id);
        currentAction.setActionsOrderObjects("GEO");
        currentAction.setAnchorGeo();
      }
    }
    id++;
}

composition: SKYLINE specList {}
        | PARETO specList {}
          // Can be used for anchors facets and object scope!
          // Also for object−scoped actions over terms but it is not so intuitive,
          // only for FDT with multi−indexing regarding terms
            | PRIORITY specList {}
 ;
// Support also greek and coptic '\u0370'..'\u03FF (Maybe in the future we should
 support all languages)
// plus modern greek \u1f00 \u1fff
ID : ('a'..'z'|'A'..'Z'|'\u0370'..'\u03FF'|'\u1F00'..'\u1FFF'|
'0'..'9'|'_'|'.'|'−'|'('|')'|'&'|'%')+ // We use the dot to get terms of specific
 facet, i.e. Location.Cefalonia
;

TWO_POINTS_LIST: FOCUS LBR LPR COORDINATE COMMA COORDINATE
 RPR COMMA LPR COORDINATE
 COMMA COORDINATE RPR RBR;
N_POINTS_LIST: FOCUS LBR LPR COORDINATE COMMA COORDINATE
 RPR COMMA LPR COORDINATE
 COMMA COORDINATE RPR (COMMA LPR COORDINATE COMMA COORDINATE
  RPR)* RBR;
CIRCLE_POINTS_LIST: FOCUS LBR LPR COORDINATE COMMA COORDINATE
 RPR COMMA LPR RAD RPR RBR;
A_POINT: FOCUS LPR COORDINATE COMMA COORDINATE RPR;
FOCUS: '...';
LBR: '[';
RBR: ']';
LPR: '(';
RPR: ')';
COORDINATE : ('0'..'9')*'.'+('0'..'9')*;
RAD : FLOAT ;
```

```
FLOAT : '0'..'9'+'.'+('0'..'9')*;
NEWLINE:'\r'? '\n' ;
WS : (' '|'\t')+ -> channel(HIDDEN); // instead of skip()
```

A.3 Client Side

Request on alternative method

```
function prefAction(epilogiAction){
// first we create the JSON array
// and use the edges of the RECTANGLE as input
// where preferenceAction is in format of our preference language
var action = [];
var NE,SW,NW,SE;
// we get the edges of the rectangle
NE=rectangle.getBounds().getNorthEast();
SW=rectangle.getBounds().getSouthWest();
NW = new google.maps.LatLng(NE.lat(),SW.lng());
SE = new google.maps.LatLng(SW.lat(),NE.lng());
action.push(NE);
action.push(SW);
var actionepilogi='rectangle'+'+'+epilogiAction;
var preferenceAction={};
preferenceAction[actionepilogi]=action;
ajaxAddPreference(JSON.stringify(preferenceAction));
setSelection(rectangle, tabID);
}
```

Bibliography

[1] C. Stadler, J. Lehmann, K. Höffner, and S. Auer, "Linkedgeodata: A core for a web of spatial open data," *Semantic Web*, vol. 3, no. 4, pp. 333–354, 2012.

[2] G. M. Sacco and Y. Tzitzikas, *Dynamic taxonomies and faceted search: theory, practice, and experience.* Springer Science & Business Media, 2009, vol. 25.

[3] F. J. Lopez-Pellicer, M. J. Silva, M. Chaves, F. J. Zarazaga-Soria, and P. R. Muro-Medrano, "Geo linked data," in *Database and Expert Systems Applications.* Springer, 2010, pp. 495–502.

[4] B. Vatant and M. Wick, "Geonames ontology (2006)," *Online at http://www.geonames.org/ontology*, 2006.

[5] Y. Tzitzikas and P. Papadakos, "Interactive exploration of multidimensional and hierarchical information spaces with real-time preference elicitation," *Fundamenta Informaticae*, vol. 20, pp. 1–42, 2012.

[6] K. Kritikos, Y. Rousakis, and D. Kotzinos, "A cloud-based, geospatial linked data management system," in *Transactions on Large-Scale Data- and Knowledge-Centered Systems XX.* Springer, 2015, pp. 59–89.

[7] K. Kyzirakos, I. Vlachopoulos, D. Savva, S. Manegold, and M. Koubarakis, "Geotriples: a tool for publishing geospatial data as rdf graphs using r2rml mappings," in *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272.* CEUR-WS. org, 2014, pp. 393–396.

[8] A. d. Leon, F. Wisniewki, B. Villazón-Terrazas, and O. Corcho, "Map4rdf-faceted browser for geospatial datasets," in *Proceedings of PMOD workshop.* Informatica, 2012.

[9] R. Battle and D. Kolas, “Enabling the geospatial semantic web with parliament and geosparql,” *Semantic Web*, vol. 3, no. 4, pp. 355–370, 2012.

[10] M. L. Yiu, H. Lu, N. Mamoulis, and M. Vaitis, “Ranking spatial data by quality preferences,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 3, pp. 433–446, 2011.

[11] J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Nørvåg, “Efficient processing of top-k spatial preference queries,” *Proceedings of the VLDB Endowment*, vol. 4, no. 2, pp. 93–104, 2010.

[12] J. Bao and M. F. Mokbel, “Georank: An efficient location-aware news feed ranking system,” in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 2013, pp. 184–193.

[13] C. Stadler, M. Martin, and S. Auer, “Exploring the web of spatial data with facete,” in *Proceedings of the companion publication of the 23rd international conference on World wide web companion.* International World Wide Web Conferences Steering Committee, 2014, pp. 175–178.

[14] D. Tunkelang, “Faceted search,” *Synthesis lectures on information concepts, retrieval, and services*, vol. 1, no. 1, pp. 1–80, 2009.

[15] O. Erling and I. Mikhailov, “Faceted views over large-scale linked data,” in *Linked Data on the Web (LDOW)*, 2009.

[16] Y. Tzitzikas, N. Manolis, and P. Papadakos, “Faceted exploration of RDF/S datasets: a survey,” *Journal of Intelligent Information Systems*, 2016.

[17] P. Papadakos and Y. Tzitzikas, “Comparing the effectiveness of intentional preferences versus preferences over specific choices: a user study,” *International Journal of Information and Decision Sciences*, vol. 8, no. 4, pp. 378–403, 2016.

[18] Y. Tzitzikas, N. Bailly, P. Papadakos, N. Minadakis, and G. Nikitakis, *Species Identification Through Preference-Enriched Faceted Search.* Cham: Springer International Publishing, 2015, pp. 77–88. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-24129-6_7

[19] P. Papadakos and Y. Tzitzikas, “Hippalus: Preference-enriched faceted exploration,” in *EDBT/ICDT Workshops*, vol. 172, 2014.

[20] Y. Tzitzikas, N. Baily, P. Papadakos, N. Minadakis, and G. Nikitakis, "Using preference-enriched faceted search for species identification," *International Journal of Metadata, Semantics and Ontologies*, 2016, to appear.

[21] Y. Tzitzikas and E. Dimitrakis, "Preference-enriched faceted search for voting aid applications," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[22] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, "Web site delays: How tolerant are users?" *Journal of the Association for Information Systems*, vol. 5, no. 1, pp. 1–28, 2004. [Online]. Available: http://d-scholarship.pitt.edu/13919/

[23] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM*, vol. 5, no. 11, pp. 558–562, 1962.

[24] R. A. Virzi, "Refining the test phase of usability evaluation: How many subjects is enough?" *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 34, no. 4, pp. 457–468, 1992.

[25] L. Faulkner, "Beyond the five-user assumption: Benefits of increased sample sizes in usability testing," *Behavior Research Methods, Instruments, & Computers*, vol. 35, no. 3, pp. 379–383, 2003.

[26] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, 1990, pp. 249–256.

[27] P. Rigaux, M. Scholl, and A. Voisard, *Spatial databases: with application to GIS.* Morgan Kaufmann, 2001.