Facetize: An Interactive Tool for Cleaning and Transforming Datasets for Facilitating their Exploration

Anna Kokolaki

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 University Campus, Heraklion, GR-70013, 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

Facetize: An Interactive Tool for Cleaning and Transforming Datasets for Facilitating their Exploration

Thesis submitted by
Anna Kokolaki
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: ______________________________
Anna Kokolaki

Committee approvals: ______________________________
Yannis Tzitzikas
Associate Professor, University of Crete
Thesis Supervisor

____
Dimitris Plexousakis
Professor, University of Crete
Committee Member

____
Giorgos Flouris
Researcher, FORTH-ICS
Committee Member

Departmental approval: ______________________________
Antonis Argyros
Professor, University of Crete
Director of Graduate Studies

Heraklion, September 2018

Facetize: An Interactive Tool for Cleaning and Transforming Datasets for Facilitating their Exploration

Abstract

There is a plethora of datasets in various formats which are usually stored in files, hosted in data catalogs or accessed through SPARQL endpoints. In most cases, these datasets cannot be straightforwardly explored by end users. To fill this gap, in this thesis we present the design and implementation of `Facetize` an editor that allows plain users with no particular technical background, to transform datasets, either static (i.e. stored in files), or dynamic (i.e. being the results of SPARQL queries), to datasets that can be directly explored effectively by themselves or other users. The latter is achieved through the familiar interaction paradigm of Faceted Search (and Preference-enriched Faceted Search). We describe the requirements, we introduce the required set of transformations, and then we detail the functionality and the implementation of the editor `Facetize` that realizes these transformations. The supported operations cover a wide range of tasks (selection, visibility, deletions, edits, definition of hierarchies, intervals, derived attributes, and others). In this way `Facetize` enables the user to carry them out in a user-friendly and guided manner, without presupposing any technical background (regarding data representation or query languages). Finally we present the results of task-based evaluation with users which was quite positive.

Facetize: Ένα Διαλογικό Εργαλείο για τον Καθαρισμό και το Μετασχηματισμό Συνόλων Δεδομένων για τη Διευκόλυνση της Εξερεύνησής τους

Περίληψη

Υπάρχει πληθώρα συνόλων δεδομένων σε διάφορες μορφές τα οποία συνήθως αποθηκεύονται σε αρχεία, φιλοξενούνται σε καταλόγους δεδομένων ή προσπελαύνονται μέσω των τελικών σημείων SPARQL . Στις περισσότερες περιπτώσεις, αυτά τα σύνολα δεδομένων δεν μπορούν να εξερευνηθούν απ ευθείας από τους τελικούς χρήστες. Για να συμπληρώσουμε αυτό το κενό, σε αυτή τη διατριβή παρουσιάζουμε το σχεδιασμό και την υλοποίηση του Facetize, ενός διαλογικού εργαλείου επεξεργασίας που επιτρέπει στους απλούς χρήστες, χωρίς τεχνικό υπόβαθρο, να μετασχηματίζουν σύνολα δεδομένων είτε στατικά (δηλ. αποθηκευμένα σε αρχεία) είτε δυναμικά (δηλ. απαντήσεις επερωτήσεων), σε σύνολα δεδομένων που μπορούν να εξερευνηθούν άμεσα από τους ίδιους ή άλλους χρήστες. Το τελευταίο επιτυγχάνεται μέσω του γνωστού παραδείγματος εξερευνητικής αναζήτησης, που ονομάζεται Πολυεδρική Αναζήτηση (και της Πολυεδρικής Αναζήτησης με Προτιμήσεις). Στη διατριβή περιγράφονται οι απαιτήσεις, εισάγεται το απαιτούμενο σύνολο μετασχηματισμών και στη συνέχεια αναλύεται η λειτουργικότητα και η υλοποίηση του εργαλείου Facetize που πραγματώνει αυτούς τους μετασχηματισμούς. Οι υποστηριζόμενες λειτουργίες καλύπτουν ένα ευρύ φάσμα εργασιών (επιλογή, ορατότητα, διαγραφές, τροποποιήσεις, ορισμός ιεραρχιών και διαστημάτων, παράγωγα γνωρίσματα και άλλα) και το Facetize επιτρέπει στο χρήστη να τις εκτελέσει με φιλικό τρόπο, χωρίς να προϋποθέτεται οποιοδήποτε τεχνικό υπόβαθρο (όσον αφορά τις γλώσσες αναπαράστασης δεδομένων ή τις γλώσσες επερωτήσεων). Τέλος παρουσιάζονται τα αποτελέσματα μιας αξιολόγησης με τους χρήστες που ήταν θετική.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επόπτη καθηγητή μου κ. Γιάννη Τζίτζικα για την άψογη συνεργασία και ουσιαστική συμβολή του στην ολοκλήρωση της παρούσας μεταπτυχιακής εργασίας. Ακόμη θέλω να εκφράσω τις ευχαριστίες μου στον κ. Δημήτρη Πλεξουσάκη και στον κ. Γιωργο Φλουρή για την προθυμία τους να συμμετέχουν στην τριμελή επιτροπή.

Ακόμα ευχαριστώ το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την πολύτιμη υποστήριξη σε υλικοτεχνική υποδομή και τεχνογνωσία, καθώς και για την υποτροφία που μου προσέφερε κατα τη διάρκεια της μεταπτυχιακής μου εργασίας.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου για την συμπαράσταση και την υποστήριξη που μου έδωσαν όλα αυτά τα χρόνια. Επειτα θα ήθελα να ευχαριστήσω τον μεταδιδακτορικό ερευνητή Παναγιώτη Παπαδάκο για την ενασχόληση του με κομμάτια την εργασίας αυτής και τον χρόνο που αφιέρωσε, τον Άγγελο Μαυρουλάκη για την εργασία του που παρέλαβα και επέκτεινα, καθώς και τους Μαρκετάκη Γιάννη και Παναγιώτη Λιονάκη για τον χρόνο που επίσης αφιέρωσαν για την εργασία μου. Θα ήθελα επίσης να ευχαριστήσω όλους όσους είχαν την προθυμία να συμμετέχουν στην αξιολόγηση του συστήματος και για τα χρήσιμα σχόλια που έκαναν για επέκταση και βελτίωση του συστήματος στο μέλλον.

...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

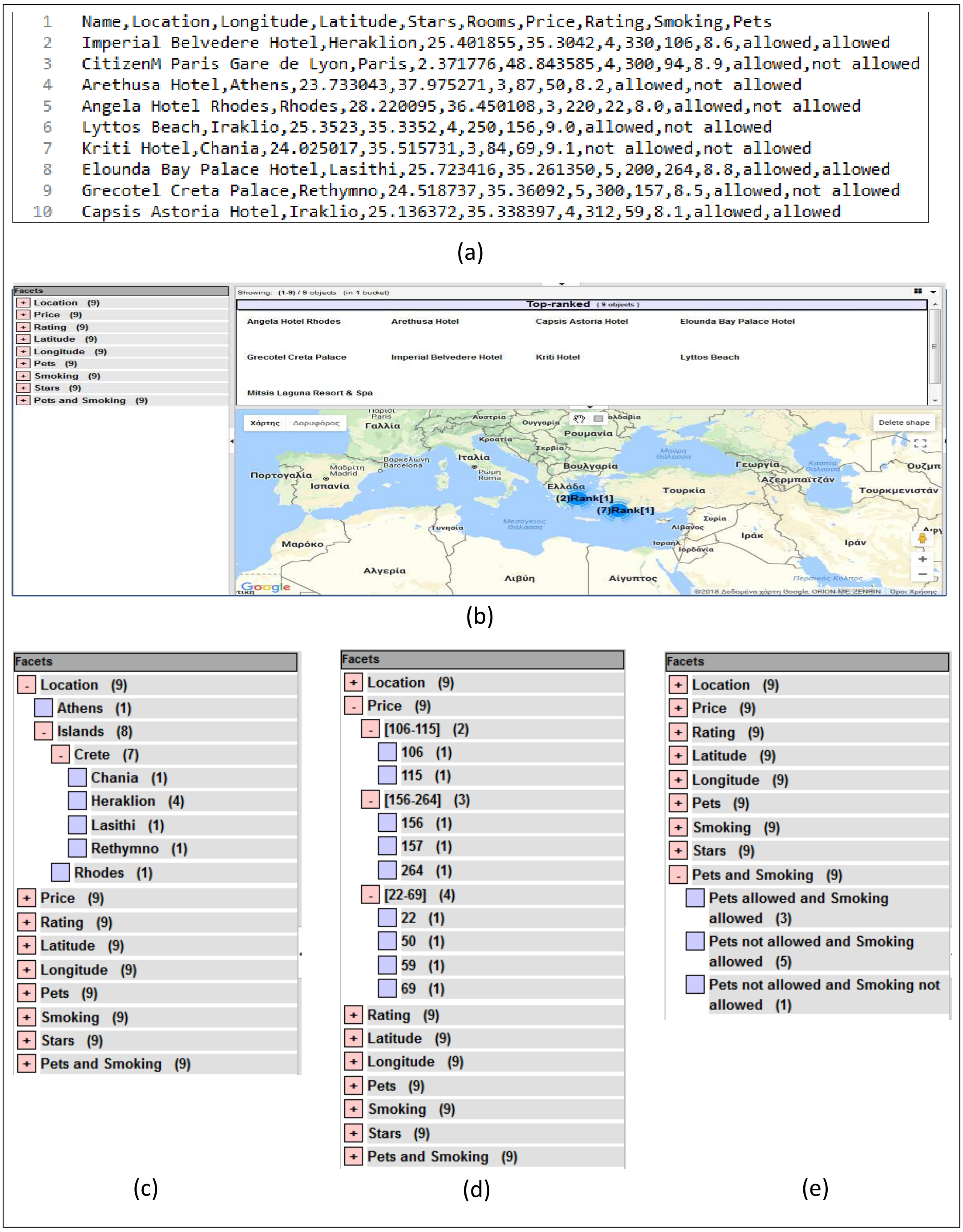
There is a plethora of datasets in various formats which are usually hosted in catalogs. Although there is a tendency to Linked Data, the majority of datasets is still represented in simple data file formats such as CSV and TSV. Such datasets cannot be easily exploited by end users. A user could either have to view directly these CVS files using a text editor, or spreadsheet application, or on the other extreme he should build dedicated applications for offering a more user friendly exploration service.

To tackle this difficulty, our objective is to provide a general purpose solution that enables users to directly explore such datasets and/or to configure the way they are explored. To this end, we rely on a quite familiar access method, specifically on *Faceted Search*, which is the defacto standard in e-commerce and booking applications. However, a straightforward loading of such datasets in a faceted search system will not always result to a satisfying solution, since additional tasks are usually required. This includes deciding (a) the parts of the dataset that should be explorable, (b) the attributes that should visible and their order, (c) the transformations and/or enrichments that should be done, (d) the groupings (hierarchical or not) of the values that should be made, (e) the addition of derived attributes, and others. For this reason in this thesis we present the design and implementation of an editor, called `Facetize` that allows the user to carry out these tasks in a user-friendly and guided manner, without presupposing any technical background (regarding the data representation language and the query languages).

To grasp the idea, Figure 1.1(a) shows a CSV file containing information about 9 hotels each described with 10 properties. Now suppose that one would like from this dataset to produce an explorable dataset, as sketched in Figure 1.1(b), with the following specific requirements:

1^{o} The dataset must contain hotels, only located in Greece (so one row should be deleted).

2^{o} The dataset should be enriched with a new hotel with values: *Mitsis Laguna*



```
Name,Location,Longitude,Latitude,Stars,Rooms,Price,Rating,Smoking,Pets
Imperial Belvedere Hotel,Heraklion,25.401855,35.3042,4,330,106,8.6,allowed,allowed
CitizenM Paris Gare de Lyon,Paris,2.371776,48.843585,4,300,94,8.9,allowed,not allowed
Arethusa Hotel,Athens,23.733043,37.975271,3,87,50,8.2,allowed,not allowed
Angela Hotel Rhodes,Rhodes,28.220095,36.450108,3,220,22,8.0,allowed,not allowed
Lyttos Beach,Iraklio,25.3523,35.3352,4,250,156,9.0,allowed,not allowed
Kriti Hotel,Chania,24.025017,35.515731,3,84,69,9.1,not allowed,not allowed
Elounda Bay Palace Hotel,Lasithi,25.723416,35.261350,5,200,264,8.8,allowed,allowed
Grecotel Creta Palace,Rethymno,24.518737,35.36092,5,300,157,8.5,allowed,not allowed
Capsis Astoria Hotel,Iraklio,25.136372,35.338397,4,312,59,8.1,allowed,allowed
```

(a)

Facets
- + Location (9)
- + Price (9)
- + Rating (9)
- + Latitude (9)
- + Longitude (9)
- + Pets (9)
- + Smoking (9)
- + Stars (9)
- + Pets and Smoking (9)

Showing: (1-9) / 9 objects (in 1 bucket)

Top-ranked (9 objects)

Angela Hotel Rhodes | Arethusa Hotel | Capsis Astoria Hotel | Elounda Bay Palace Hotel
Grecotel Creta Palace | Imperial Belvedere Hotel | Kriti Hotel | Lyttos Beach
Mitsis Laguna Resort & Spa

(b)

Facets
- - Location (9)
  - □ Athens (1)
  - - Islands (8)
    - - Crete (7)
      - □ Chania (1)
      - □ Heraklion (4)
      - □ Lasithi (1)
      - □ Rethymno (1)
    - □ Rhodes (1)
- + Price (9)
- + Rating (9)
- + Latitude (9)
- + Longitude (9)
- + Pets (9)
- + Smoking (9)
- + Stars (9)
- + Pets and Smoking (9)

(c)

Facets
- + Location (9)
- - Price (9)
  - - [106-115] (2)
    - □ 106 (1)
    - □ 115 (1)
  - - [156-264] (3)
    - □ 156 (1)
    - □ 157 (1)
    - □ 264 (1)
  - - [22-69] (4)
    - □ 22 (1)
    - □ 50 (1)
    - □ 59 (1)
    - □ 69 (1)
- + Rating (9)
- + Latitude (9)
- + Longitude (9)
- + Pets (9)
- + Smoking (9)
- + Stars (9)
- + Pets and Smoking (9)

(d)

Facets
- + Location (9)
- + Price (9)
- + Rating (9)
- + Latitude (9)
- + Longitude (9)
- + Pets (9)
- + Smoking (9)
- + Stars (9)
- - Pets and Smoking (9)
  - □ Pets allowed and Smoking allowed (3)
  - □ Pets not allowed and Smoking allowed (5)
  - □ Pets not allowed and Smoking not allowed (1)

(e)

Figure 1.1: Scenario Example

Resort & Spa,Heraklion,
25.371359,35.307237, 5,385,115,8.7,allowed,not allowed

3^{o} The properties with name *Longitude* and *Latitude*, must be marked as properties that contain geographic information for longitude and latitude respectively, for enabling an exploration system (like Hippalus) to show the location of each hotel on a map.

4^{o} The entities should take as names their corresponding values of property *Name*.

5^{o} The value *Iraklio* in property *Location*, must be replaced with the value *Heraklion*, in all entities that contain that value.

6^{o} The property *Rooms* should not appear in the list of facets: it should either be hidden or deleted.

7^{o} A new property with name *Pets and Smoking* should be created, with values as shown in Figure 1.1(e), and each hotel should be associated with the right value.

8^{o} The values in property *Location* should be organized hierarchically as shown in Figure 1.1(c).

9^{o} The values in property *Price*, must be organized in interval hierarchies, as shown in the bottom part of Figure 1.1(d).

10^{o} The order of the facets should be as shown in Figure 1.1.

With the approach that we describe in this thesis, and the tool `Facetize` a naive user can make all these transformation in an easy manner and produce an output file that is directly loadable to Hippalus, which the users can directly explore through a GUI as shown in Figure 1.1(b). The overall process is sketched in Figure 1.2.

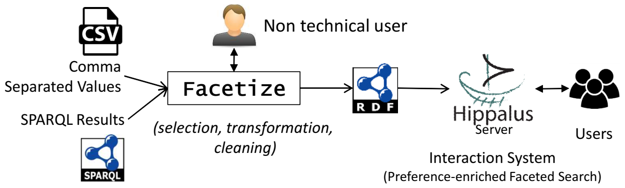


Figure 1.2: The process

In a nutshell, the key contributions of this thesis are: (a) we identify the basic requirements for this kind of tasks, (b) we present a set of operations (for selecting, transforming, cleaning or enriching a dataset), (c) we describe in detail the design and implementation of `Facetize` that supports these operations over static datasets (i.e. stored in files) as well as dynamic (i.e. being the results of SPARQL queries) and (d) we report the results of a task-based evaluation with users.

In comparison to related systems, OpenRefine is probably the more relevant system. One distinctive feature of `Facetize` is that it supports the creation of hierarchies and numeric intervals and these features are crucial for managing the complexity of large datasets, and producing datasets that can be easily explored. Moreover, `Facetize` can fetch data directly from SPARQL endpoints, making it appropriate for dynamic datasets.

1.1 Outline of Thesis

The rest of this thesis is organized as follows:

Chapter 2 describes background information.

Chapter 3 identifies the requirements and

Chapter 4 discusses related work.

Chapter 5 describes the target data structure.

Chapter 6 describes transformations,

Chapter 7 presents the editor and describes the structures of system's projects.

Then Chapter 8 presents the results of the evaluation with users of the scenario discribed in this Chapter,

and finally Chapter 9 concludes the thesis and identifies directions for future research.

Chapter 2

Context and Background

Section 2.1 represents the statistics of representation and categories of datasets in CKAN data hubs and portals, Section 2.2 discusses faceted search, Section 2.3 discusses PFS: Preference-enriched Faceted Search, Section 2.4 discusses the Preference-enriched faceted search system Hippalus.

2.1 Datasets

There has been a lot of activity in open data around the world. CKAN data hubs and portals[1] exist in Austria, Brazil, US, Africa and many other countries. The top-five most used data categories[2] are government and public sector (11.9%), economy and finance (11.6%), regions and cities (10.1%), population and society (9.5%), and environment (8.9%). These five data categories together represent 52% of the total re-use of Open Data. The least used data categories are international issues (3.6%), health (4.2%) and justice, legal system and public safety (4.2%).

`Govdata.de` is one of the most popular open data portals, with coherent and compatible licensing policy and interesting, politically relevant dataset. In `govdata.de` exist almost 26.76% files in CSV format[3].

`Data.gov.uk` is the official open data portal of the UK Government. The majority of files in that catalogue is in HTML format (about 33.85% of existing files), while 18.70% of them are in CSV format.

`Dati.gov.it` open data portal of Italy, accommodates around 29.61% files in CSV format.

In `european data portal.eu` (EDP) exist around 11.68% CSV files. The EDP data category offering most mapped data sets is the justice, legal system and public safety category (27.8%), followed by environment (23.6%), regions and cities (12.0%), science and technology (11.9%) and population and society (5.5%).

[1] https://ckan.org/

[2] https://www.europeandataportal.eu/sites/default/files/re-using_open_data.pdf

[3] From open data formats statistics of the main catalogs (November 2016)

The category government and public sector provides only 3.6% of the total mapped data sets while the economy and finance category only provides 4.4%.

The majority of datasets in `data.gov` open data portal of U.S. Government's is in HTML format (21.44%) and there are 3.84% files in CSV format. Also, there are 61 alive SPARQL endpoints [4] that are cataloged in CKAN.

In average, in all popular open data portals, there are around 18.1% files in CSV format.

Moreover sensors and scientific instruments usually produce data expressed in CSV. Finally, data in CSV are very easy to produce even by plain users using a text editor or a spreadsheet application.

There are also large amounts of data published as Linked Data, e.g. see [29].

2.2 Faceted Search

Faceted Exploration (or *Faceted Search*) is a widely used interaction scheme for Exploratory Search. It is the de facto query paradigm in e-commerce [38, 41]. It has been generalized also for RDF datasets (see [44] for a recent survey). In a short (and rather informal) way we could define it as a *session-based interactive method for query formulation (commonly over a multidimensional information space) through simple clicks that offers an overview of the result set (groups and count information), never leading to empty results sets.*

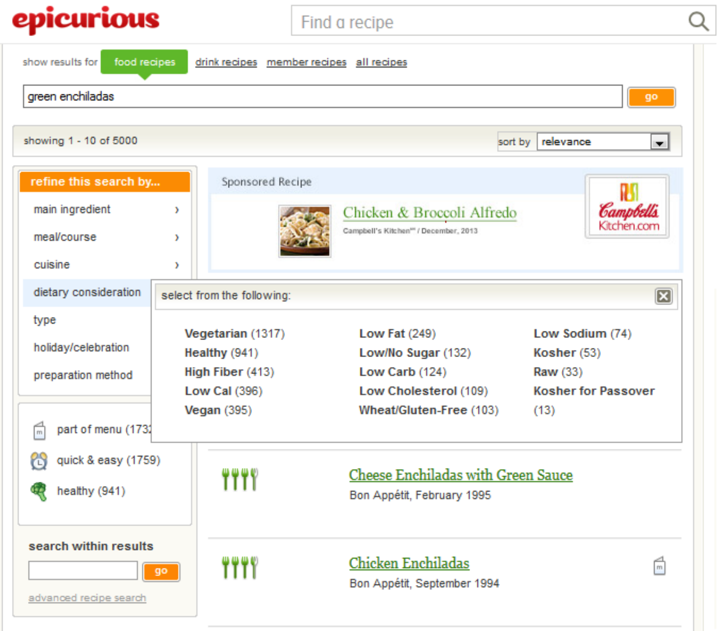
Filters are also tools that help users find information. They analyze a given set of content to exclude items that don't meet certain criteria. One important difference between *Filters* and *Faceted Search* is that *Faceted Search* provides multiple filters, one for each different aspect of the content. *Faceted Exploration* is thus more flexible and more useful than systems which provide only one or two different types of filters, especially for extremely large content sets. Because *Faceted Search* describes many different dimensions of the content, it also provides a structure to help users understand the content space, and give them ideas about what is available and how to search for it.

For example, Figures 2.1 and 2.2 contain snapshots of systems that use *Faceted Search* for exploration. The full-fledged faceted navigation on Epicurious.com (Fig. 2.1) allows users to narrow results by several different dimensions, including Cuisine, Main Ingredient, and Dietary Consideration. In this case it's a simple matter to view only healthy recipes.

2.3 PFS: Preference-enriched Faceted Search

The enrichment of faceted search with *preferences*, hereafter *Preference-enriched Faceted Search* [33, 45], for short PFS, has been proven useful for recall-oriented information needs, because such needs involve decision making that can benefit

[4] https://www.w3.org/wiki/SparqlEndpoints



epicurious

Find a recipe

show results for food recipes drink recipes member recipes all recipes

green enchiladas go

showing 1 - 10 of 5000 sort by relevance

refine this search by...

- main ingredient ›
- meal/course ›
- cuisine ›
- dietary consideration
- type
- holiday/celebration
- preparation method

select from the following:

Vegetarian (1317)
Healthy (941)
High Fiber (413)
Low Cal (396)
Vegan (395)
Low Fat (249)
Low/No Sugar (132)
Low Carb (124)
Low Cholesterol (109)
Wheat/Gluten-Free (103)
Low Sodium (74)
Kosher (53)
Raw (33)
Kosher for Passover (13)

part of menu (173
quick & easy (1759)
healthy (941)

search within results go

advanced recipe search

Sponsored Recipe

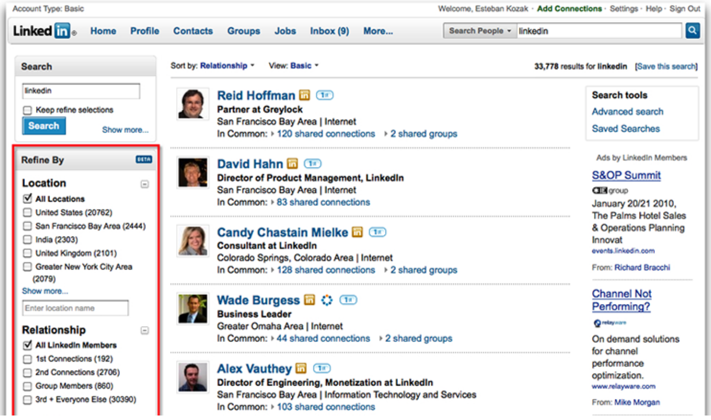
Chicken & Broccoli Alfredo
Campbell's Kitchen℠ / December, 2013

Campbell's Kitchen.com

Cheese Enchiladas with Green Sauce
Bon Appétit, February 1995

Chicken Enchiladas
Bon Appétit, September 1994

Figure 2.1: The faceted navigation on Epicurious.com allows the flexibility to narrow results by many different criteria, including Main Ingredient, Cuisine, and Dietary Consideration.



Account Type: Basic

Welcome, Esteban Kozak · Add Connections · Settings · Help · Sign Out

LinkedIn Home Profile Contacts Groups Jobs Inbox (9) More...

Search People linkedin

Search

linkedin

□ Keep refine selections

Search Show more...

Refine By BETA

Location

☑ All Locations
□ United States (20762)
□ San Francisco Bay Area (2444)
□ India (2303)
□ United Kingdom (2101)
□ Greater New York City Area (2079)

Show more...

Enter location name

Relationship

☑ All LinkedIn Members
□ 1st Connections (192)
□ 2nd Connections (2706)
□ Group Members (860)
□ 3rd + Everyone Else (30390)

Sort by: Relationship View: Basic 33,778 results for linkedin [Save this search]

Reid Hoffman
Partner at Greylock
San Francisco Bay Area | Internet
In Common: 120 shared connections 2 shared groups

David Hahn
Director of Product Management, LinkedIn
San Francisco Bay Area | Internet
In Common: 83 shared connections

Candy Chastain Mielke
Consultant at LinkedIn
Colorado Springs, Colorado Area | Internet
In Common: 128 shared connections 2 shared groups

Wade Burgess
Business Leader
Greater Omaha Area | Internet
In Common: 44 shared connections 2 shared groups

Alex Vauthey
Director of Engineering, Monetization at LinkedIn
San Francisco Bay Area | Information Technology and Services
In Common: 103 shared connections

Search tools
Advanced search
Saved Searches

Ads by LinkedIn Members

S&OP Summit
group
January 20/21 2010, The Palms Hotel Sales & Operations Planning Innovat
events.linkedin.com
From: Richard Bracchi

Channel Not Performing?
relayware
On demand solutions for channel performance optimization.
www.relayware.com
From: Mike Morgan

Figure 2.2: A canonical example of a faceted navigation at LinkedIn.

from the gradual interaction and expression of preferences. PFS 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 distinctive features of PFS is that it allows expressing preferences over attributes whose values can be hierarchically organized (and/or multi-valued), it supports preference inheritance, and it offers scope-based rules for resolving automatically the conflicts that may arise. As a result the user is able to restrict his current focus by using the faceted interaction scheme (hard restrictions) that lead to non-empty results, and rank according to preference the objects of his focus. Recently, PFS has been used in various domains, e.g. for offering a flexible process for the identification of fish species [43], as a Voting Advice Application [42] and it has been expanded with geographic anchors for being appropriate for the exploration of datasets that contain also geographic information [27]. Finally, applications of the model in the context of spoken dialogue systems are also emerging, e.g. see [34].

2.4 Hippalus

Hippalus [32] is a publicly accessible web system that implements the PFS interaction model. The information base that feeds Hippalus is represented in RDF/S[5] (using a schema adequate for representing objects described according to dimensions with hierarchically organized values). For loading and querying such information, Hippalus uses Jena[6], a Java framework for building Semantic Web applications. Hippalus offers a web interface for Faceted Search enriched with preference actions offered through HTML 5 context menus[7]. The performed actions are internally translated to statements of the preference language described in [45], 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.

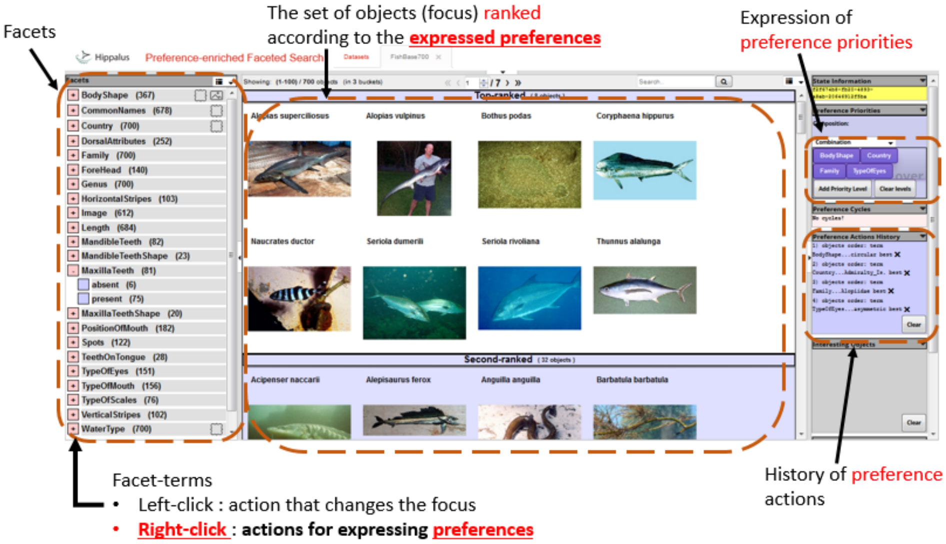
Hippalus displays the preference ranked list of objects in the central part of the screen, while the right part is occupied by information that relates to the information thinning (object restrictions), preference actions history and preference composition. Figure 2.3 shows a screenshot of Hippalus over a data set that contains fish species. The preference-related actions are offered through right-click activated pop-up menus (through HTML5 context menus). Hippalus has been evaluated very positively by users in various contexts (the interested reader can

[5] http://www.w3.org/TR/rdf-schema/

[6] http://jena.apache.org/

[7] Available only to firefox 8 and up.

refer to [32, 42, 43]).


Facets

The set of objects (focus) ranked according to the **expressed preferences**

Expression of preference priorities

Facet-terms

- Left-click : action that changes the focus
- **Right-click : actions for expressing preferences**

History of preference actions

1) objects order: term BodyShape...circular best ✕
2) objects order: term Country...Admiralty_Is. best ✕
3) objects order: term Family...Alopiidae best ✕
4) objects order: term TypeOfEyes...asymmetric best ✕

Figure 2.3: Hippalus preference actions

Chapter 3

Requirements

We dichotomize requirements to transformation requirements (described in §3.1) and to workbench-related requirements (described in §3.2).

3.1 Transformation Requirements

Even from the running example in the introductory chapter it is evident that for preparing a dataset appropriate for exploration [5, 24, 39, 48] one should be able to define the facets that should be visible, and their order. As regards the terms of the facets, it should be possible to define hierarchical groupings of terms for aiding the exploration (and for avoiding cluttering the GUI), as well as intervals to numerical values. Moreover it should be possible to define new facets whose terms are derived by applying functions over the terms of other facets. In addition, the user should be able to specify the type of the terms of a facet, e.g. identifier, integer, float, string, longitude and latitude. By defining the type of a facet as identifier, the entities will take as names their corresponding values in this facet, and all values in this facet should be distinct.

Finally, it should be possible to add individual rows, edit individual rows and delete individual rows. Moreover it should be able to delete all rows with a specific value or a condition. Moreover it should be possible to replace each distinct value in a facet with a new one in each row of the dataset.

3.2 Workbench-related Requirements

The notion of project, should be supported, allowing the user to create, open, edit and save the changes on disc.

The system should be able to keep the history of transformations that have been applied and offer to the user the ability to undo the desired ones (and redo).

Moreover, the user should be able to open a project and change the input dataset, i.e. by giving a more recent version of the dataset, or the file with the hierarchical information. This is very important for datasets that change over time, since we would not like to loose the transformations that have been defined for a past version of the dataset.

Finally it should be possible to export the transformed dataset in RDF according to a schema that is compliant with Hippalus, for enabling the straightforward loading by Hippalus. Table 3.1 shows the data cleansing and transformations requirements.

Table 3.1: Data Cleansing and transformations requirements.

Category	Feature
Import/Export	Import/Export text files, EXCEL Import/Export Data from databases Import/Export RDF files Export R2RML/JSON files Execute SPARQL queries to retreive data from a source
Data Visualization	Ability to view data in plain text as well as in tabular format
Data cleaning: Rows	Deletion of rows Removal of duplicate rows Addition of row Removal of empty rows
Data cleaning: Values	Deletion of records containing missing values Character removal, text replacement, date conversion Value editing Impute missing values Creation of value hierarchies Creation of intervals for arithmetic values Correction of bad values using string similarity metrics Clustering of the values in a property that contain specific characters (e.g. same prefix)
	Delete/move attributes
Data cleaning: Columns	Filtering Renaming of Columns Set column types Creation of new columns using expression Split of columns Deletion of columns Rearrangement of columns Hiding of columns Addition/merging of columns Split records into columns
Editor Facilities	Undo/redo transformations Display of the distinct values of a property and number of their occurrences Ability to reapply same range of transformations
Other Export Functions	Export macro or a C program, or a Perl program Extraction of Entities Export of Statistics, Table, File, Database, Visualize

Chapter 4

Related Work

This chapter is organized as follows: Section 4.1 discusses the process of Data Cleansing, the problems we encounter when we manage large datasets and the solutions that exist. Section 4.2 describes the various data cleansing and transformation tools that exist, and Section 4.3 summarizes the describes and the placement of `Facetize` in the landscape.

4.1 Data Cleansing

Data Cleansing [36] deals with detecting and removing errors, impurities and inconsistences from data collections, such as files or databases, in order to improve the quality and accuracy of data. Data quality problems occurs due to misspelling during data entry, missing information or other invalid data. Organizations around the world generate huge amount of data from their day-to-day activities for their operations. These organizations will not survive if the data they generate remains dirty or erroneous.

Data Warehouses [6, 21] require and provide extensive support for data cleaning. They load and continuously refresh huge amounts of data from a variety of sources so the probability that some of the sources contain ”dirty data” is high. Furthermore, data warehouses are used for decision making, so that the correctness of their data is vital to avoid wrong conclusions. For instance, duplicated or missing information will produce incorrect or misleading statistics (“garbage in, garbage out”). Due to the wide range of possible data inconsistencies and the sheer data volume, data cleaning is considered to be one of the biggest problems in data warehousing.

4.1.1 Types of Data Errors

In large datasets, it is more likely to contain errors. [2] Data error is an atomic value that is different from its given real fact and can be categorized into quantitative or qualitative [30].

In qualitative errors, are included:

1. **Rule violations**, refer to values that violate any kind of integrity constraints, such as not null and uniqueness constraints.

2. **Pattern violations**, refer to values that violate syntactic and semantic constraints, such as misspelling, formatting and semantic data types.

In quantitative errors, are included:

1. **Outliers**, refer to values that deviate from the distribution of values in the column of a table. These data values do not follow the statistical distribution of the bulk of the data.

2. **Duplicates**, refer to different records that refer in the same entity. These can have exactly the same or different values.

4.1.2 State of the Art

Available data cleaning solutions and tools [2], [28] belong to one or more of the following four categories:

- *Rule-based detection algorithms* [1,7,14,17] that can be embedded into frameworks, such as NADEEF [9, 24], where a rule can vary from a simple "not null" constraint to multi-attribute functional dependencies (FDs) to user-defined functions. Using this class of tools, a user can specify a collection of rules that clean data will obey, and the tool will find any violations.

- *Pattern enforcement and transformation tools* such as OpenRefine[1], Data Wrangler [23], Trifacta[2], Katara [8], and DataX-Former [4]. These tools discover patterns in the data, either syntactic (e.g.,OpenRefine and Trifacta) or semantic (e.g., Katara), and use these to detect errors (cells that do not conform with the patterns). They can also be used to change data representation and expose additional patterns.

- *Quantitative error detection algorithms* that expose outliers, and glitches in the data [3, 11, 35, 47].

- *Record linkage and de-duplication algorithms* for detecting duplicate data records, such as the Data Tamer system [40]. These tools perform entity consolidation when multiple records have data for the same entity. Conflicting values for the same attribute can be found, indicating possible errors.

[1] http://openrefine.org/

[2] http://www.trifacta.com

4.2 Tools for Data Cleansing and Transformations

There is a variety of cleansing and transformation tools in the market. In some cases, the data cleaning tools are unable to remove completely all the anomalies and therefore the user involvement in the data cleaning process cannot be overlooked. We must mention, that there is no single dominant tool. Each tool needs to work in collaboration with other tools. In this section, we will describe some of the existing data cleansing and transformation projects.

4.2.1 DataPreparator

DataPreparator[3] is a free software tool designed to assist with common tasks of data preprocessing in data analysis and data mining. As we can see in Figure 4.1, it supports data access from text files, relational databases and Excel workbooks. So, it can handle large volumes of data (since data sets are not stored in the computer memory, with the exception of Excel workbooks and result sets of some databases where database drivers do not support data streaming).

The system provides data cleaning methods, such as character removal, text replacement and date conversion on the data (Fig. 4.2). A user can delete or move the selected attributes, discretize numeric attributes with equal width or equal frequency (Fig. 4.3), scale numeric attributes with options such as Decimal, Linear, Hyperbolic tangent, Soft-max, Z-score (Fig. 4.4) and handle missing values (Fig. 4.5). In case of missing values, user can delete records and remove attributes containing missing values, impute missing values and predict them from models such as dependency tree and Naive Bayes model.

The system creates different forms of outputs and they can be statistics, tables, files, databases and visualizations. Visualization output can be for numeric attributes such as Bar chart, Box plot, Histogram, Lag plot and many others. For the categorical attributes, some visualizations can be with Bar chart, pie chart, Pareto chart and Stacked chart. Also for numeric and nomimal attributes, DataPreparator creates Dependency tree (Fig. 4.6) and Parallel coordinates (Fig. 4.7).

DataPreparator is a stand alone tool, written in Java and requires Java Runtime Environment (JRE) to be installed on the machine (JRE 1.6 or later). It runs on Windows, Mac OS/X, and Linux operating systems.

4.2.2 Potter's Wheel

Potter's Wheel [37] is an interactive data cleansing system that integrates data transformation and error detection using spreadsheet-like Interface. Users can immediately see the effects of the performed operations in tuples that are visible on screen. Error detections are applied automatically in the background.

[3] http://www.datapreparator.com/

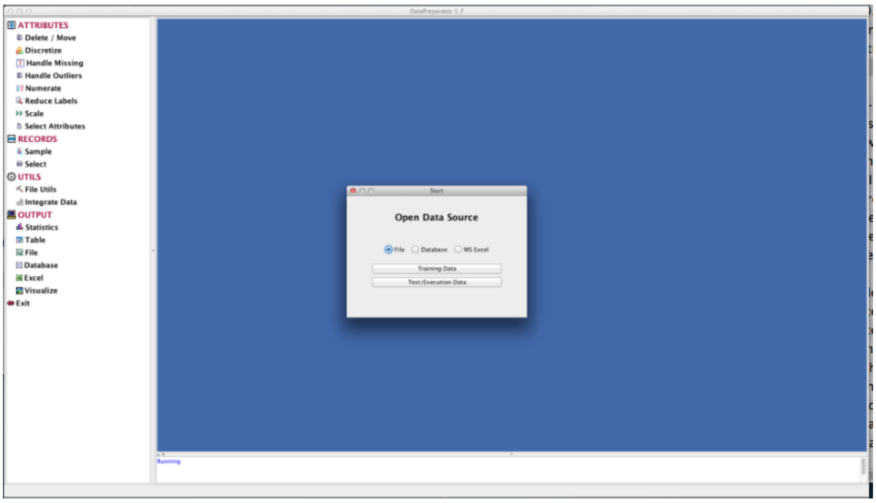
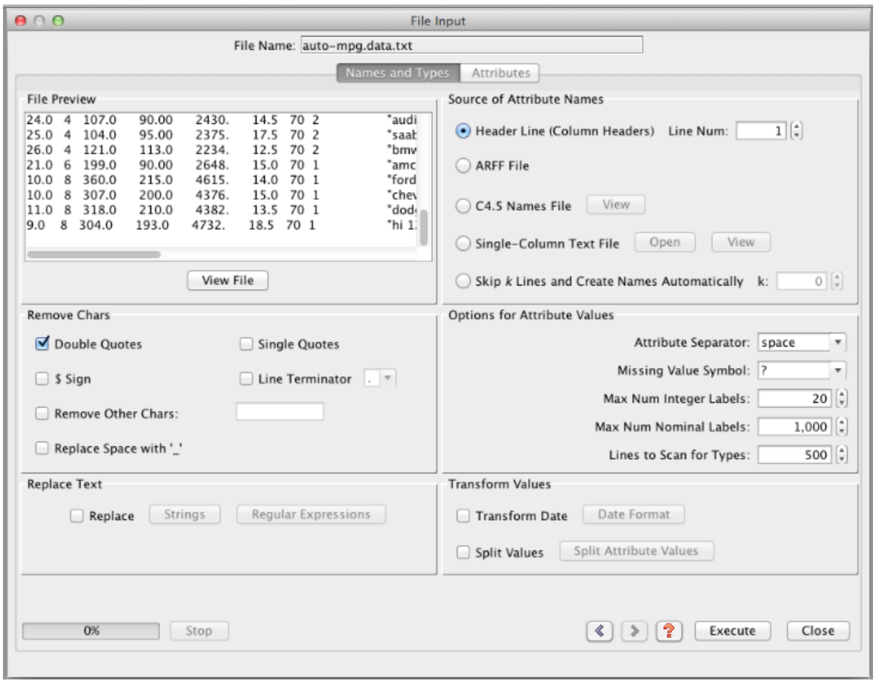


Figure 4.1: The initial window in DataPreparator.



File Name: auto-mpg.data.txt

```
24.0  4  107.0   90.00   2430.   14.5  70  2        "audi
25.0  4  104.0   95.00   2375.   17.5  70  2        "saab
26.0  4  121.0   113.0   2234.   12.5  70  2        "bmw
21.0  6  199.0   90.00   2648.   15.0  70  1        "amc
10.0  8  360.0   215.0   4615.   14.0  70  1        "ford
10.0  8  307.0   200.0   4376.   15.0  70  1        "chev
11.0  8  318.0   210.0   4382.   13.5  70  1        "dodg
9.0   8  304.0   193.0   4732.   18.5  70  1        "hi 1
```

● Header Line (Column Headers) Line Num: 1

○ ARFF File

○ C4.5 Names File

○ Single-Column Text File

○ Skip *k* Lines and Create Names Automatically k: 0

☑ Double Quotes

□ Single Quotes

□ $ Sign

□ Line Terminator .

□ Remove Other Chars:

□ Replace Space with '_'

Attribute Separator: space

Missing Value Symbol: ?

Max Num Integer Labels: 20

Max Num Nominal Labels: 1,000

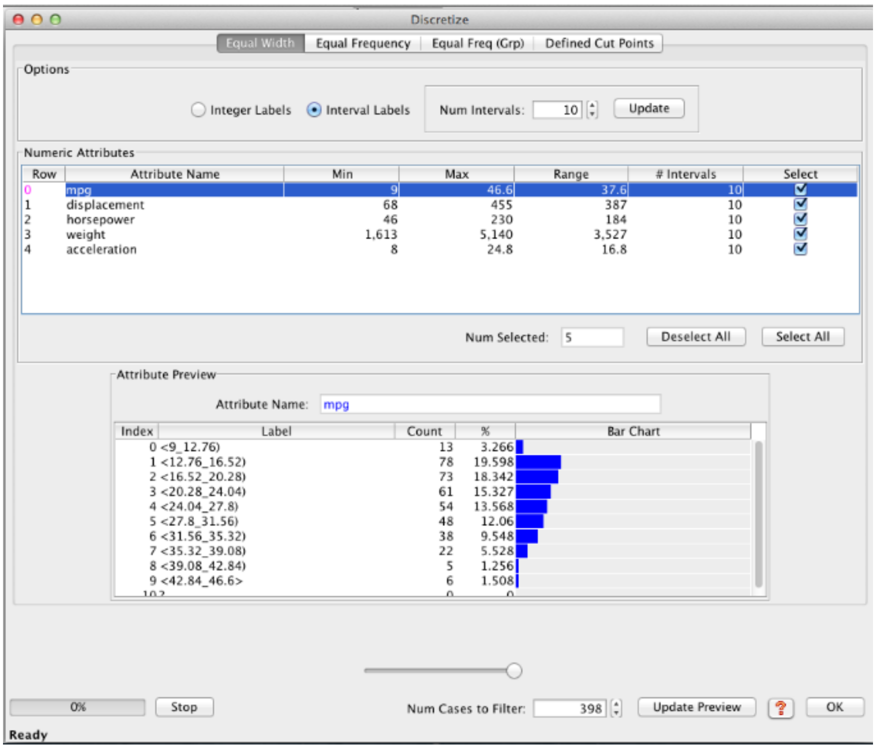
Lines to Scan for Types: 500

□ Replace

□ Transform Date

□ Split Values

Figure 4.2: File Input Options in DataPreparator.



Options

○ Integer Labels ● Interval Labels

Num Intervals: 10

Numeric Attributes

Row	Attribute Name	Min	Max	Range	# Intervals	Select
0	mpg	9	46.6	37.6	10	☑
1	displacement	68	455	387	10	☑
2	horsepower	46	230	184	10	☑
3	weight	1,613	5,140	3,527	10	☑
4	acceleration	8	24.8	16.8	10	☑

Num Selected: 5

Attribute Preview

Attribute Name: mpg

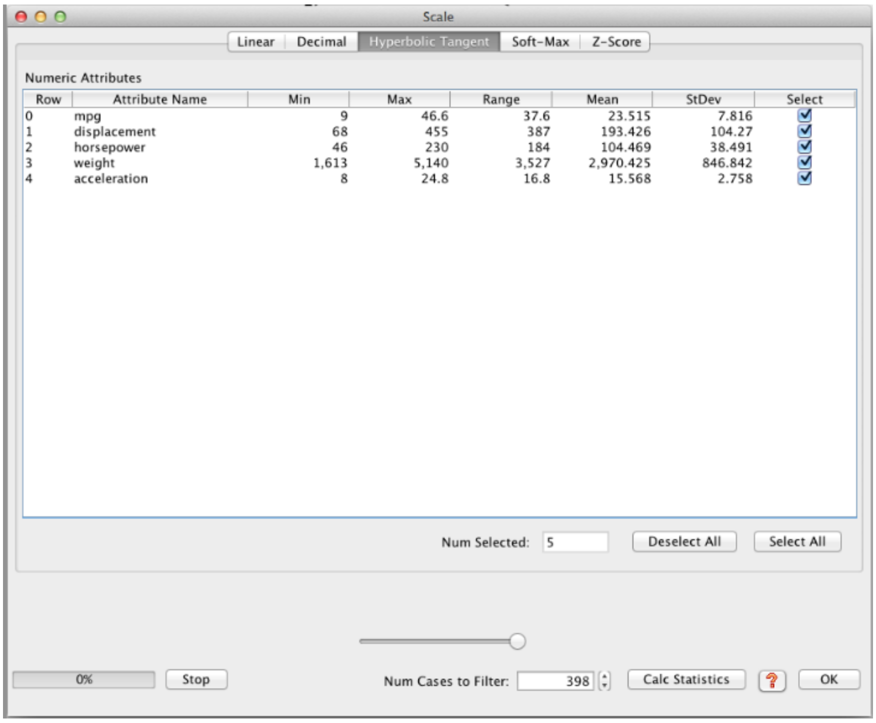
Index	Label	Count	%	Bar Chart
0	<9_12.76)	13	3.266	
1	<12.76_16.52)	78	19.598	
2	<16.52_20.28)	73	18.342	
3	<20.28_24.04)	61	15.327	
4	<24.04_27.8)	54	13.568	
5	<27.8_31.56)	48	12.06	
6	<31.56_35.32)	38	9.548	
7	<35.32_39.08)	22	5.528	
8	<39.08_42.84)	5	1.256	
9	<42.84_46.6>	6	1.508	
10 ?		0	0	

0%

Num Cases to Filter: 398

Ready

Figure 4.3: Discretization dialog contains options for discretizing numeric attributes in DataPreparator.



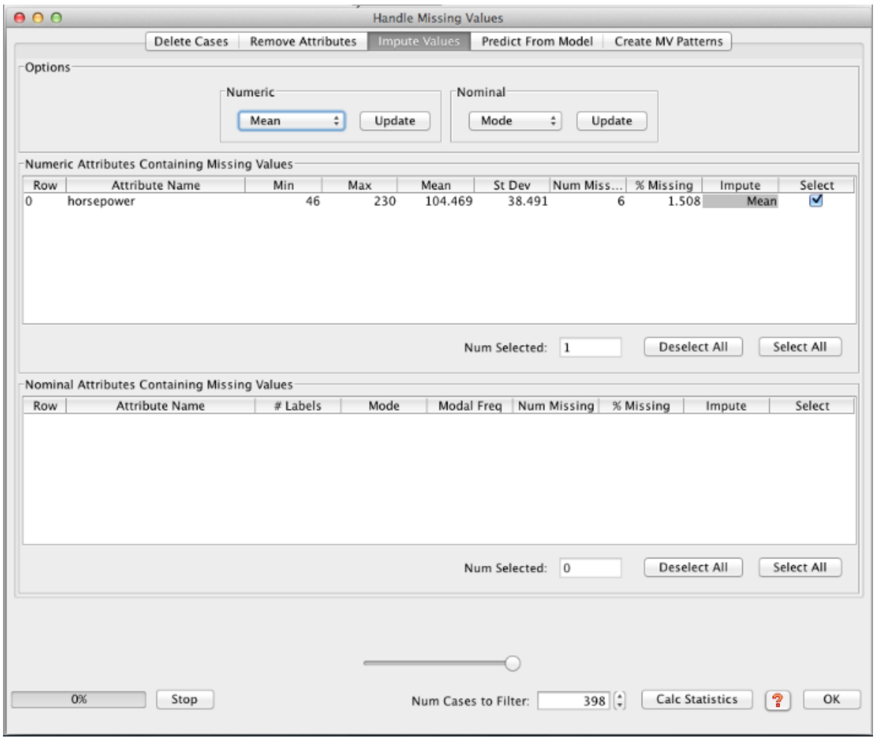
Numeric Attributes

Row	Attribute Name	Min	Max	Range	Mean	StDev	Select
0	mpg	9	46.6	37.6	23.515	7.816	☑
1	displacement	68	455	387	193.426	104.27	☑
2	horsepower	46	230	184	104.469	38.491	☑
3	weight	1,613	5,140	3,527	2,970.425	846.842	☑
4	acceleration	8	24.8	16.8	15.568	2.758	☑

Num Selected: 5

Num Cases to Filter: 398

Figure 4.4: Dialog containing options for scaling of numeric attributes in DataPreparator.



Numeric: Mean

Nominal: Mode

Numeric Attributes Containing Missing Values

Row	Attribute Name	Min	Max	Mean	St Dev	Num Miss...	% Missing	Impute	Select
0	horsepower	46	230	104.469	38.491	6	1.508	Mean	☑

Num Selected: 1

Nominal Attributes Containing Missing Values

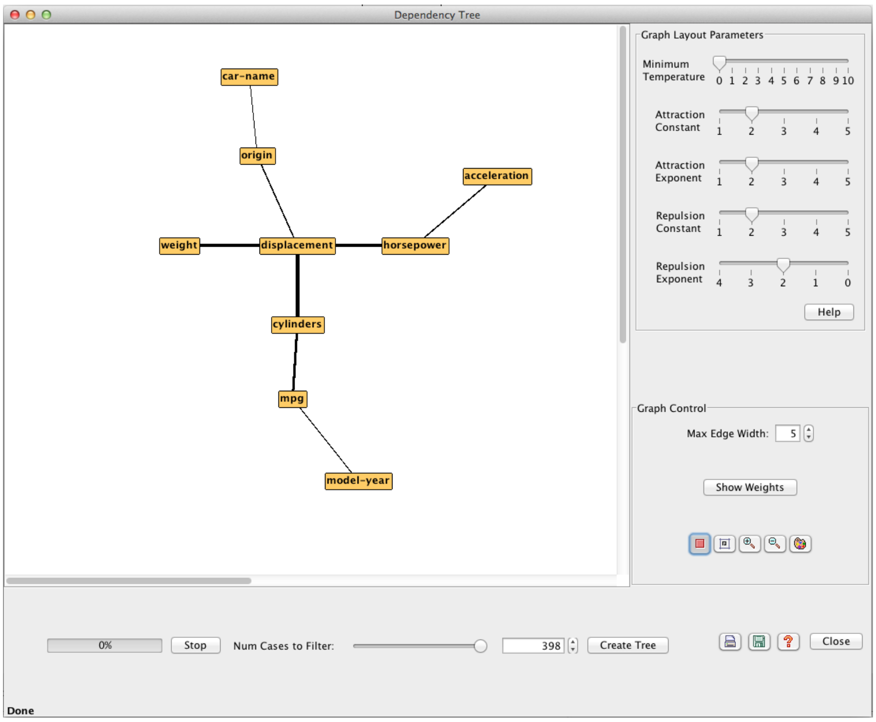
Row	Attribute Name	# Labels	Mode	Modal Freq	Num Missing	% Missing	Impute	Select

Num Selected: 0

0%

Num Cases to Filter: 398

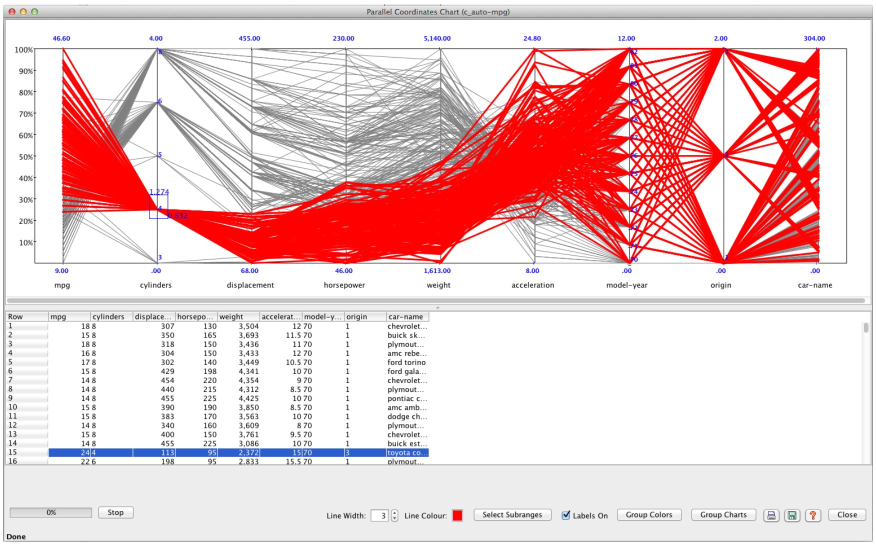
Figure 4.5: The dialog containing options for handling missing values in DataPreparator.



Max Edge Width: 5

Num Cases to Filter: 398

Figure 4.6: Example dependency tree in DataPreparator. The nodes represent variables and the links the relationships between the variables.



Row	mpg	cylinders	displace...	horsepo...	weight	accelerat...	model-y...	origin	car-name
1	18	8	307	130	3,504	12	70	1	chevrolet...
2	15	8	350	165	3,693	11.5	70	1	buick sk...
3	18	8	318	150	3,436	11	70	1	plymout...
4	16	8	304	150	3,433	12	70	1	amc rebe...
5	17	8	302	140	3,449	10.5	70	1	ford torino
6	15	8	429	198	4,341	10	70	1	ford gala...
7	14	8	454	220	4,354	9	70	1	chevrolet...
8	14	8	440	215	4,312	8.5	70	1	plymout...
9	14	8	455	225	4,425	10	70	1	pontiac c...
10	15	8	390	190	3,850	8.5	70	1	amc amb...
11	15	8	383	170	3,563	10	70	1	dodge ch...
12	14	8	340	160	3,609	8	70	1	plymout...
13	15	8	400	150	3,761	9.5	70	1	chevrolet...
14	14	8	455	225	3,086	10	70	1	buick est...
15	24	4	113	95	2,372	15	70	3	toyota co...
16	22	6	198	95	2,833	15.5	70	1	plymout...

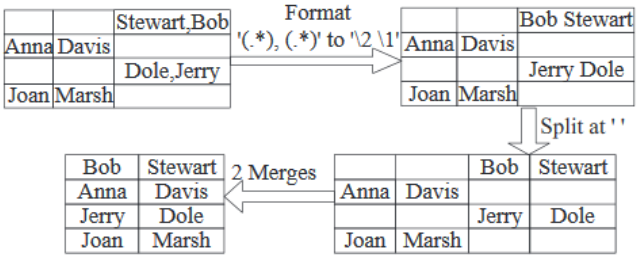
Line Width: 3 Line Colour: 🟥 ☑ Labels On

Figure 4.7: Example Parallel Coordinates in DataPreparator.

It supports a set of operations, called transforms. These are i) Value Translation, ii) One-to-one Mappings of Rows and iii) Many-to-Many Mappings of Rows. Value Translation transforms, apply a function to every value in a column. One-to-one transforms are column operations that transform individual rows. As illustrated in Figures 4.8 and 4.9, they can be used to unify data collected from different sources. The operations that are supported in this case are the $Merge$ transform that concatenates values in two columns, optionally interposing a constant (the delimiter) in the middle, to form a single new column. *Split* that splits a column into two or more parts, by specifying character positions and regular expressions. *Drop*, *Copy*, *Add* transforms allow user to drop, copy or add a column. *Divide* transform conditionally divides a column, sending values into one of two new columns based on a predicate.

Many-to-Many transforms help to tackle higher-order schematic heterogeneities where information is stored partly in data values and partly in the schema, as shown in Figure 4.10. These transforms, include operation *Fold*, that converts one row into multiple rows, folding a set of columns together into one column and replicating the rest. Conversely, $Unfold$ operation takes two columns, collects rows that have the same values for all the other columns, and unfolds the two chosen columns. Values in one column are used as column names to align the values in the other column. Figures 4.10 and 4.11 show an example with student grades where the subject names are demoted into the row via Format, grades are Folded together,and then Split to separate the subject from the grade. Fold and UnFold are adapted from the restructuring operators of SchemaSQL [25], and are discussed in more detail in the full paper [37].

Users have the ability to undo incorrect transforms. After the user is satisfied with the sequence of transforms, Potter's Wheel can compile it into a transformation, and export it as either a C or Perl program, or a Potter's Wheel macro.



		Stewart,Bob
Anna	Davis	
		Dole,Jerry
Joan	Marsh	

		Bob Stewart
Anna	Davis	
		Jerry Dole
Joan	Marsh	

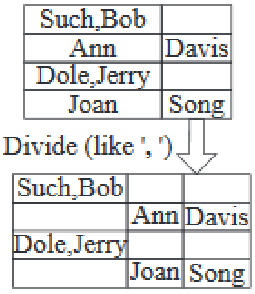
		Bob	Stewart
Anna	Davis		
		Jerry	Dole
Joan	Marsh		

Bob	Stewart
Anna	Davis
Jerry	Dole
Joan	Marsh

Figure 4.8: Using Format, Merge and Split to clean name format differences

4.2.3 WinPure Clean

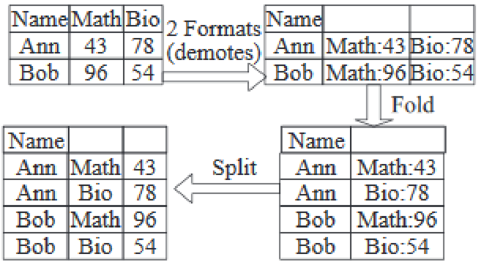
WinPure [19] is a worldwide provider of data quality solutions, is simple to use and not only by experts. Businesses around the world use WinPure software to improve the quality of their information.



Such,Bob	
Ann	Davis
Dole,Jerry	
Joan	Song

Such,Bob		
	Ann	Davis
Dole,Jerry		
	Joan	Song

Figure 4.9: Divide-ing to separate various name formats



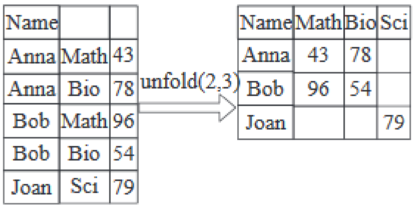
Name	Math	Bio
Ann	43	78
Bob	96	54

Name		
Ann	Math:43	Bio:78
Bob	Math:96	Bio:54

Name	
Ann	Math:43
Ann	Bio:78
Bob	Math:96
Bob	Bio:54

Name		
Ann	Math	43
Ann	Bio	78
Bob	Math	96
Bob	Bio	54

Figure 4.10: Fold-ing to fix higher-order variations



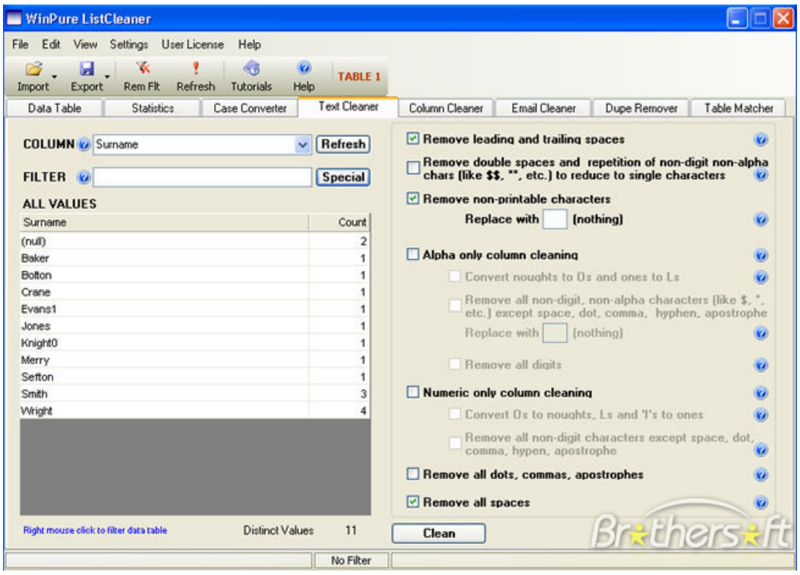
Name		
Anna	Math	43
Anna	Bio	78
Bob	Math	96
Bob	Bio	54
Joan	Sci	79

Name	Math	Bio	Sci
Anna	43	78	
Bob	96	54	
Joan			79

Figure 4.11: Unfold-ing into three columns

It supports operations like filtering of rows, text cleaning (Fig. 4.12), update missing values, merging duplicate groups (Fig. 4.13), deduplication (Fig. 4.14) and exports data in different formats such as CSV, Excel and in other databases like MySQL, SQL Server.

It is a stand alone desktop application, runs on windows and it was designed using .NET technologies.



COLUMN: Surname

FILTER:

ALL VALUES

Surname	Count
(null)	2
Baker	1
Bolton	1
Crane	1
Evans1	1
Jones	1
Knight0	1
Merry	1
Sefton	1
Smith	3
Wright	4

Right mouse click to filter data table

Distinct Values 11

No Filter

☑ Remove leading and trailing spaces

□ Remove double spaces and repetition of non-digit non-alpha chars (like $$, "", etc.) to reduce to single characters

☑ Remove non-printable characters
Replace with □ (nothing)

□ Alpha only column cleaning
□ Convert noughts to Os and ones to Ls
□ Remove all non-digit, non-alpha characters (like $, *, etc.) except space, dot, comma, hyphen, apostrophe
Replace with □ (nothing)
□ Remove all digits

□ Numeric only column cleaning
□ Convert Os to noughts, Ls and 'I's to ones
□ Remove all non-digit characters except space, dot, comma, hypen, apostrophe

□ Remove all dots, commas, apostrophes

☑ Remove all spaces

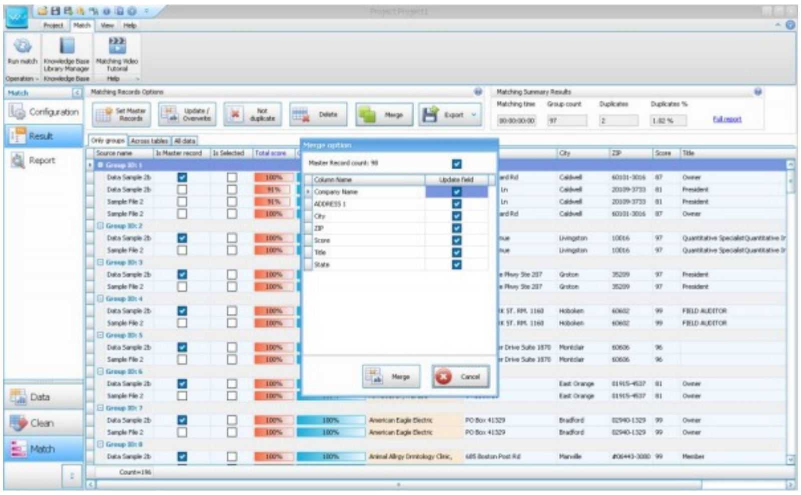
Figure 4.12: WinPure Clean

4.2.4 OpenRefine

OpenRefine[4] is a free and open-source desktop application for cleaning and explore large datasets that opens in the browser as a Local Webserver, so data is safe and it doesn't get uploaded to the Google server. Data exploration is performed through filtering and faceting exploration. OpenRefine can support several of data formats, such as RDF, CSV, TSV, JSON, EXCEL, XML and Google Docs. It gives the opportunity to the user to get a preview of the data before create a project for apply data cleaning and other transformations to the data (Fig. 4.15).

User can specify how to view the data (e.g. in plain text, in tabular format). Also, can select a facet and view in a widget the distinct values and the number of their frequencies (Fig. 4.16). The user can then select a value from the widget and filter rows that contain the selected value. Users can select and edit values in each column (Fig. 4.17). Also, it supports data clustering which groups values in each facet that have specific characteristics, such as same prefix (Fig. 4.18). Data is always visible in each step of editing and user can change the type of each column (e.g. from string to number type), rearrange, hide and delete them.

[4] https://casci.umd.edu/wp-content/uploads/2013/12/OpenRefine-tutorial-v1.5.pdf



Matching Summary Results

Matching time	Group count	Duplicates	Duplicates %
00:00:00:00	97	2	1.02 %

Full report

Master Record count: 98 ☑

	Column Name	Update field
▸	Company Name	☑
	ADDRESS 1	☑
	City	☑
	ZIP	☑
	Score	☑
	Title	☑
	State	☑

Source name	Is Master record	Is Selected	Total score				City	ZIP	Score	Title
Group ID: 1										
Data Sample 2b	☑	□	100%			ard Rd	Caldwell	60101-3016	87	Owner
Data Sample 2b	□	□	91%			Ln	Caldwell	20109-3733	81	President
Sample File 2	□	□	91%			Ln	Caldwell	20109-3733	81	President
Sample File 2	□	□	100%			ard Rd	Caldwell	60101-3016	87	Owner
Group ID: 2										
Data Sample 2b	☑	□	100%			nue	Livingston	10016	97	Quantitative SpecialistQuantitative Ir
Sample File 2	□	□	100%			nue	Livingston	10016	97	Quantitative SpecialistQuantitative Ir
Group ID: 3										
Data Sample 2b	☑	□	100%			e Pkwy Ste 207	Groton	35209	97	President
Sample File 2	□	□	100%			e Pkwy Ste 207	Groton	35209	97	President
Group ID: 4										
Data Sample 2b	☑	□	100%			K ST. RM. 1160	Hoboken	60602	99	FIELD AUDITOR
Sample File 2	□	□	100%			K ST. RM. 1160	Hoboken	60602	99	FIELD AUDITOR
Group ID: 5										
Data Sample 2b	☑	□	100%			r Drive Suite 1870	Montclair	60606	96	
Sample File 2	□	□	100%			r Drive Suite 1870	Montclair	60606	96	
Group ID: 6										
Data Sample 2b	☑	□	100%				East Orange	01915-4537	81	Owner
Sample File 2	□	□	100%				East Orange	01915-4537	81	Owner
Group ID: 7										
Data Sample 2b	☑	□	100%	100%	American Eagle Electric	PO Box 41329	Bradford	02940-1329	99	Owner
Sample File 2	□	□	100%	100%	American Eagle Electric	PO Box 41329	Bradford	02940-1329	99	Owner
Group ID: 8										
Data Sample 2b	☑	□	100%	100%	Animal Allrgy Drmtology Clinic,	685 Boston Post Rd	Marville	#06443-3000	99	Member

Count=196

Figure 4.13: WinPure data merging

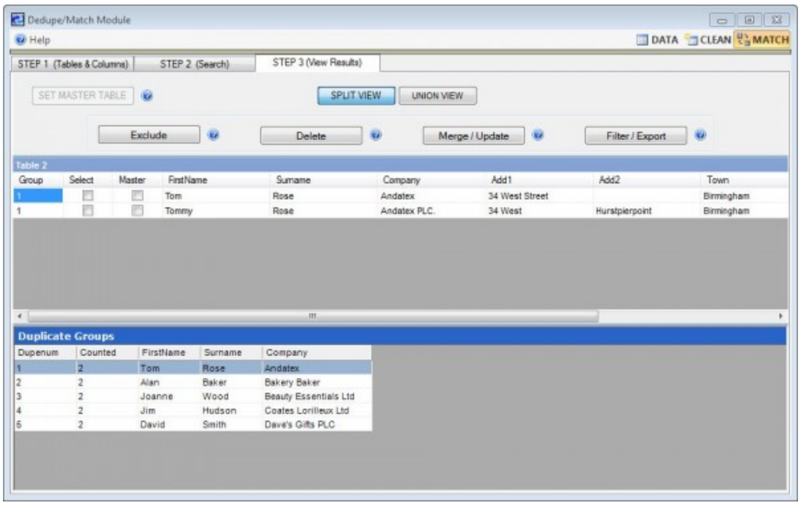


Table 2

Group	Select	Master	FirstName	Surname	Company	Add1	Add2	Town
1	□	□	Tom	Rose	Andatex	34 West Street		Birmingham
1	□	□	Tommy	Rose	Andatex PLC.	34 West	Hurstpierpoint	Birmingham

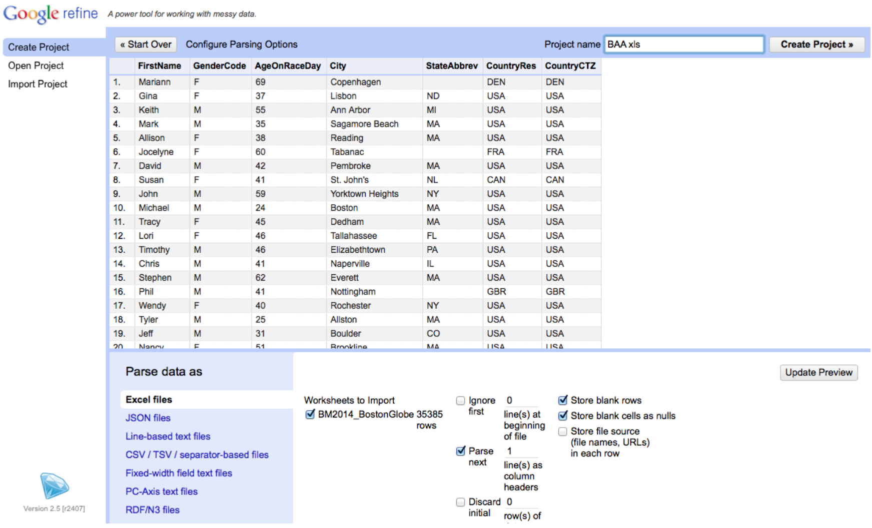
Duplicate Groups

Dupenum	Counted	FirstName	Surname	Company
1	2	Tom	Rose	Andatex
2	2	Alan	Baker	Bakery Baker
3	2	Joanne	Wood	Beauty Essentials Ltd
4	2	Jim	Hudson	Coates Lorilleux Ltd
5	2	David	Smith	Dave's Gifts PLC

Figure 4.14: WinPure deduplication

It supports also filling of empty cells, remove blank rows, remove duplicates (Fig. 4.19), undo/redo transformations, cells transformations and addendum of columns based on other columns with expressions that a user can type (Fig. 4.20).

Finally, the user can export the transformed data in the supported by the system file formats, or export the whole project in zip format (Fig. 4.21).



Project name BAA xls

	FirstName	GenderCode	AgeOnRaceDay	City	StateAbbrev	CountryRes	CountryCTZ
1.	Mariann	F	69	Copenhagen		DEN	DEN
2.	Gina	F	37	Lisbon	ND	USA	USA
3.	Keith	M	55	Ann Arbor	MI	USA	USA
4.	Mark	M	35	Sagamore Beach	MA	USA	USA
5.	Allison	F	38	Reading	MA	USA	USA
6.	Jocelyne	F	60	Tabanac		FRA	FRA
7.	David	M	42	Pembroke	MA	USA	USA
8.	Susan	F	41	St. John's	NL	CAN	CAN
9.	John	M	59	Yorktown Heights	NY	USA	USA
10.	Michael	M	24	Boston	MA	USA	USA
11.	Tracy	F	45	Dedham	MA	USA	USA
12.	Lori	F	46	Tallahassee	FL	USA	USA
13.	Timothy	M	46	Elizabethtown	PA	USA	USA
14.	Chris	M	41	Naperville	IL	USA	USA
15.	Stephen	M	62	Everett	MA	USA	USA
16.	Phil	M	41	Nottingham		GBR	GBR
17.	Wendy	F	40	Rochester	NY	USA	USA
18.	Tyler	M	25	Allston	MA	USA	USA
19.	Jeff	M	31	Boulder	CO	USA	USA
20.	Nancy	F	51	Brookline	MA	USA	USA

Parse data as

- Excel files
- JSON files
- Line-based text files
- CSV / TSV / separator-based files
- Fixed-width field text files
- PC-Axis text files
- RDF/N3 files

Worksheets to Import
☑ BM2014_BostonGlobe 35385 rows

□ Ignore first 0 line(s) at beginning of file
☑ Parse next 1 line(s) as column headers
□ Discard initial 0 row(s) of

☑ Store blank rows
☑ Store blank cells as nulls
□ Store file source (file names, URLs) in each row

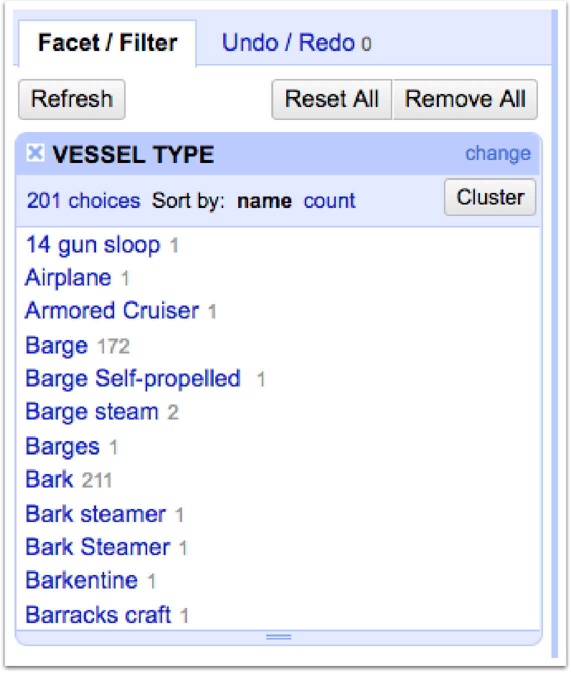
Figure 4.15: Create a new project in OpenRefine

4.2.5 Karma

Karma [18] is an open source, available tool to model open data[5]- and it was started by the University of Southern California. It lets the user to publish the generated model using different formats: RDF, R2RML or JSON. Apart from this, Karma allows the user to define some predefined ontologies and to get a list of suggested semantic types for a given column header. Karma also learns through experience. If the user selects a specific semantic type for a column, then that semantic type would be suggested when user loads data that looks very similar with previous modeled information. Karma has the following options for the columns:

- **Set Semantic Type** - allows the user to select a vocabulary and a property for that specific node column, other than the ones suggested by default by the application (Fig. 4.22).
- **Add Column** - allows the user to add a new column to the existing table
- **Rename** - rename the specific column header

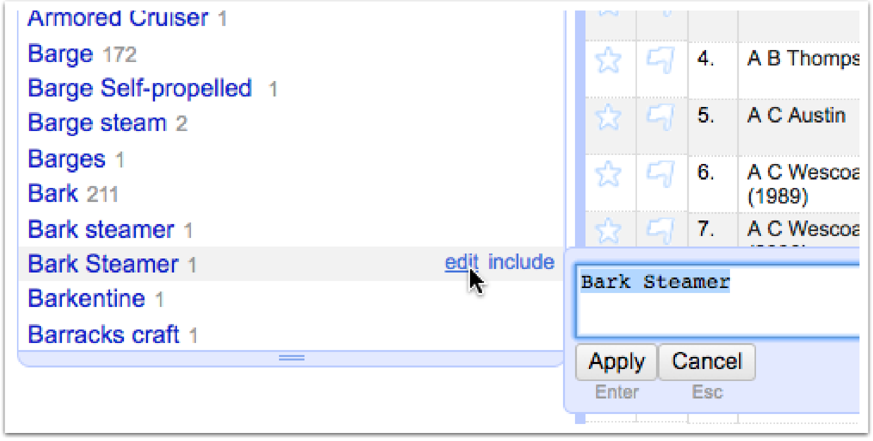
[5] Open Data Handbook: http://opendatahandbook.org/guide/en/what-is-open-data



201 choices Sort by: name count

14 gun sloop	1
Airplane	1
Armored Cruiser	1
Barge	172
Barge Self-propelled	1
Barge steam	2
Barges	1
Bark	211
Bark steamer	1
Bark Steamer	1
Barkentine	1
Barracks craft	1

Figure 4.16: Distinct values of a facet in OpenRefine

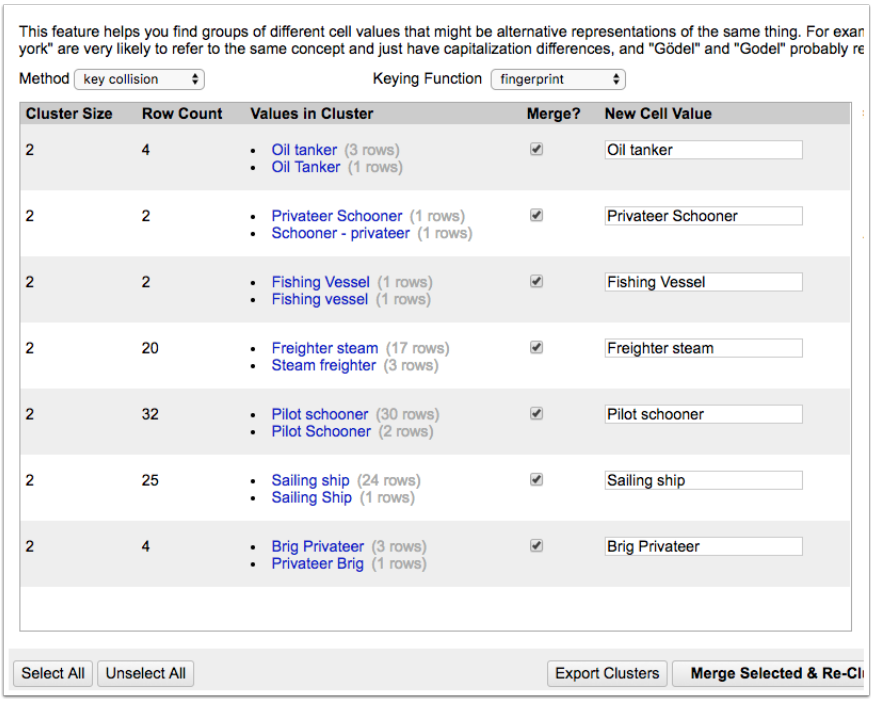


Armored Cruiser	1
Barge	172
Barge Self-propelled	1
Barge steam	2
Barges	1
Bark	211
Bark steamer	1
Bark Steamer	1
Barkentine	1
Barracks craft	1

4.	A B Thomps
5.	A C Austin
6.	A C Wescoa (1989)
7.	A C Wescoa

```
Bark Steamer
```

Figure 4.17: Edit values in OpenRefine



This feature helps you find groups of different cell values that might be alternative representations of the same thing. For exan
york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably re

Method key collision Keying Function fingerprint

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	4	• Oil tanker (3 rows) • Oil Tanker (1 rows)	☑	Oil tanker
2	2	• Privateer Schooner (1 rows) • Schooner - privateer (1 rows)	☑	Privateer Schooner
2	2	• Fishing Vessel (1 rows) • Fishing vessel (1 rows)	☑	Fishing Vessel
2	20	• Freighter steam (17 rows) • Steam freighter (3 rows)	☑	Freighter steam
2	32	• Pilot schooner (30 rows) • Pilot Schooner (2 rows)	☑	Pilot schooner
2	25	• Sailing ship (24 rows) • Sailing Ship (1 rows)	☑	Sailing ship
2	4	• Brig Privateer (3 rows) • Privateer Brig (1 rows)	☑	Brig Privateer

Figure 4.18: Data clustering in OpenRefine

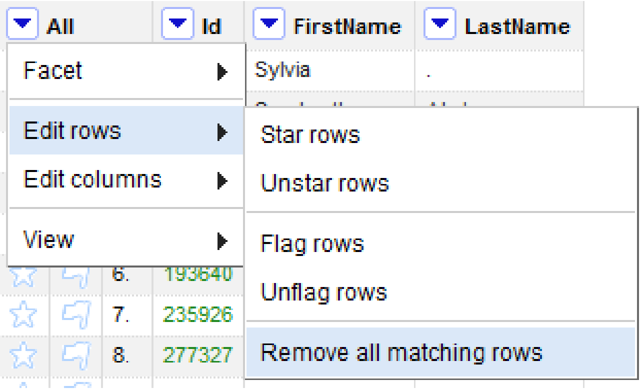
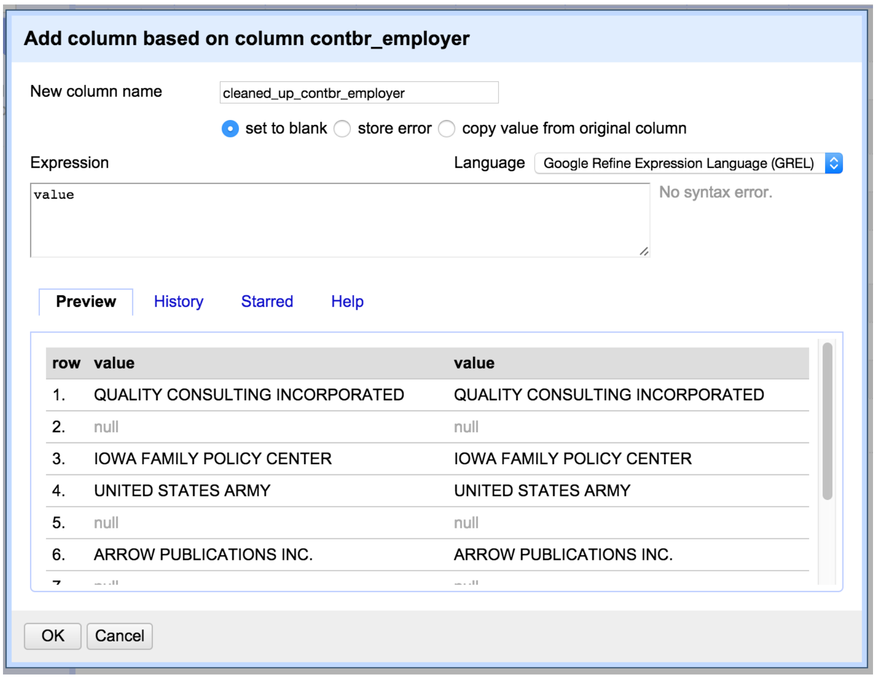


Figure 4.19: Remove duplicates in OpenRefine



Add column based on column contbr_employer

New column name: cleaned_up_contbr_employer

● set to blank ○ store error ○ copy value from original column

Expression: value

Language: Google Refine Expression Language (GREL)

No syntax error.

Preview | History | Starred | Help

row	value	value
1.	QUALITY CONSULTING INCORPORATED	QUALITY CONSULTING INCORPORATED
2.	null	null
3.	IOWA FAMILY POLICY CENTER	IOWA FAMILY POLICY CENTER
4.	UNITED STATES ARMY	UNITED STATES ARMY
5.	null	null
6.	ARROW PUBLICATIONS INC.	ARROW PUBLICATIONS INC.

Figure 4.20: Add a new column in OpenRefine

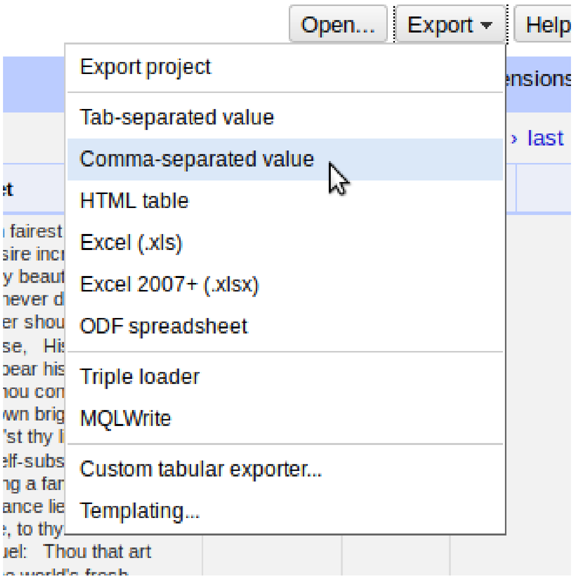


Figure 4.21: Export data in OpenRefine

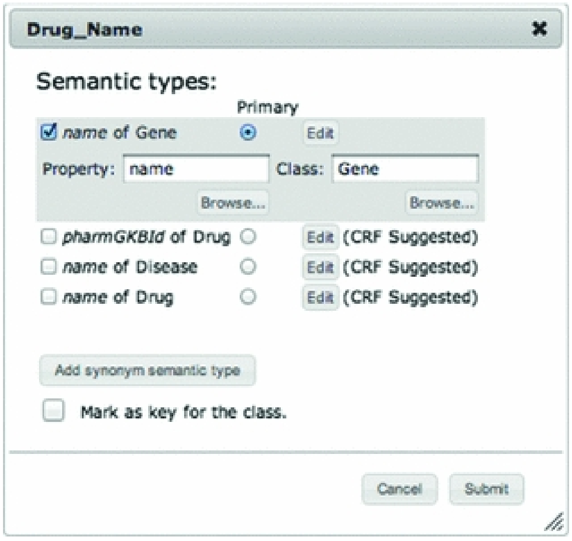
- **Split Values** - using a regex or a list of characters create a new column/or update an existing one with values from one column, each value splitted by that

- **Add Row** - add a new row at the end of the table

- **Extract Entities** - gives the user the possibility to extract data, places or persons from one of the columns

The person that would want to modify the data from a column must provide a piece of Python code that would allow him to modify the values from that column (split, merge columns, transform the data).

One of the problems with open data, is that it needs to be cleaned. When people that are responsible with inserting data might be in a hurry, some of the values from the table might be written wrong; like *"Bucharest"* might be written *"Buchraest"* (the letters *'a'* and *'r'* are transposed) or *"Paris"* and *"Pari"* (one letter is missing). For correcting this type of possible errors, Karma has a feature that indicates which string is close related to a reference one, using string similarity algorithms. The strings that are used for generating the suggestions, are the values from the same column that contains the value that would get a recommendation. The string similarity algorithms implemented in the Karma Integration Tool are: (a) *Levenshtein* [26], (b) *Damerau-Levenshtein* [10], (c) *Jaro Jaro* [22], (d) *Jaro-Winkler* [31], (e) *Cosine Similarity* [20], (f) *Jaccard* [16]. Each one of them is good to use depending on what data they are run on (Fig. 4.23).



Drug_Name ×

Semantic types:

Primary

☑ *name* of Gene ● Edit

Property: name Class: Gene

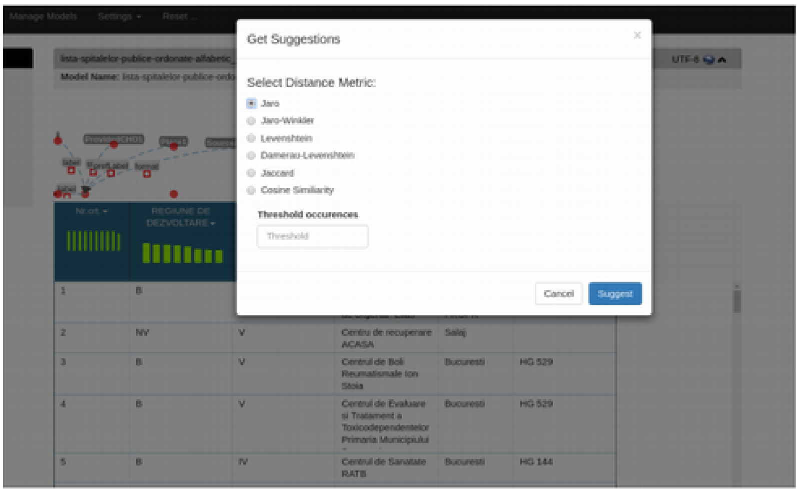
□ *pharmGKBId* of Drug ○ Edit (CRF Suggested)

□ *name* of Disease ○ Edit (CRF Suggested)

□ *name* of Drug ○ Edit (CRF Suggested)

□ Mark as key for the class.

Figure 4.22: Karma set semantic type


Manage Models
Settings
Reset ...
lista-spitalelor-publice-ordonate-alfabetic_
Model Name: lista-spitalelor-publice-ordo
UTF-8
label
prefLabel
format
×
Cancel
Suggest

Get Suggestions

Select Distance Metric:

● Jaro
○ Jaro-Winkler
○ Levenshtein
○ Damerau-Levenshtein
○ Jaccard
○ Cosine Similarity

Threshold occurences

Threshold

Nr.crt.	REGIUNE DE DEZVOLTARE				
1	B				
2	NV	V	Centru de recuperare ACASA	Salaj	
3	B	V	Centrul de Boli Reumatismale Ion Stoia	Bucuresti	HG 529
4	B	V	Centrul de Evaluare si Tratament a Toxicodependentelor Primaria Municipiului	Bucuresti	HG 529
5	B	IV	Centrul de Sanatate RATB	Bucuresti	HG 144

Figure 4.23: Karma string similarity algorithms

4.3 Executive Summary

To sum up, the existence of anomalies in real-world data motivates the development and application of data cleansing methods. Existing data cleansing approaches mostly focus on the transformation of data, the elimination of duplicates, syntax and lexical errors detection. Moreover, the current approaches offer methods for create and edit attributes using a number of functions. Also, exist methods for hide, delete and reorder attributes. Users can see the transformations in each step of editing and export the data after applying transformations in different file formats, such as Excel, Csv, RDF and others.

OpenRefine data cleansing and transformation tool, supports operations for specifying the order of the facets, defining derived attributes and editing existing ones. Also, OpenRefine supports the notion of project, where one can open, edit and export the datasets in different file formats.

In addition to the above, with `Facetize` user can create hierarchies of values and numeric intervals. Also, user can execute SPARQL queries to retrieve the data from a source, delete rows that have specific values using functions and reapply the same range of transformations.

Chapter 5

The Target Data Structure

We need a kind of "canonical data structure", a structure appropriate for representing the contents of the input CSV file as well as the outcome of the various transformations. This canonical data structure is essentially a multidimensional structure where each dimension can have hierarchically organized values. This structure can be loaded to Hippalus (if expressed in RDF using a particular RDF Schema) and thus it is appropriate for enabling PFS over the transformed dataset.

Below we define formally this structure, for being able to define formally the defined transformation. We shall refer to this structure with the term *materialized faceted taxonomy* (as it was introduced in [46]).

5.1 Materialized Faceted Taxonomy

A *terminology* $\mathcal{T}$ is any finite set of *terms* where the notion of *term* is general and able to capture both categorical and numerical values. A *taxonomy* is a pair $(\mathcal{T}, \leq)$ where $\mathcal{T}$ is a terminology, and $\leq$ a subsumption relation over $\mathcal{T}$ (whose semantics are that of the partial orders: reflexive, antisymmetric and transitive). If $\leq = \emptyset$ then the taxonomy is flat, i.e. just a terminology.

A *faceted taxonomy* with k facets is a set $\{F_1, ..., F_k\}$ where each F_i is a pair $name_i, (\mathcal{T}_i, \leq_i)$ where $name_i$ is a string (the name of the facet) and $(\mathcal{T}_i, \leq_i)$ is the taxonomy of the values that correspond to this facet (note that the terminologies of the facets are distinct, and can be separated by prefixing each term with the name of the facet).

Let Obj be any denumerable set of objects (corresponding to the real world objects at hand). An *interpretation of a terminology* $\mathcal{T}$ is any function $I : \mathcal{T} \rightarrow 2^{Obj}$. If $\mathcal{F}$ is a faceted taxonomy and I is an interpretation of $\mathcal{T} = \bigcup_{i=1,k} \mathcal{T}_i$ then a *materialized faceted taxonomy* is this pair $(\mathcal{F}$,$I)$. The *model* of $(\mathcal{T}, \leq)$ induced by I is denoted by $\bar{I}$, and is defined by considering the semantics, specifically it is defined as $\bar{I}(t) = \cup\{I(t') \mid t' \leq t\}$. Consequently, this is how the model of a materialized ontology is defined, and thus how exploration and query answering should be done (over the model of the materialized faceted taxonomy).

The *description* of an object o with respect to an interpretation I is defined as $D_I(o) = \{\ t \in \mathcal{T} \mid o \in I(t)\}$. Analogously, The *description* of an object o with respect to $\bar{I}$ is defined as $D_{\bar{I}}(o) \equiv \bar{D}_I(o) = \{\ t \in \mathcal{T} \mid o \in \bar{I}(t)\} = \cup_{t \in D_I(o)}(\{t\} \cup B^+(t))$.

All the above notions and notations are also shown in the upper part of Table 5.1.

5.2 A CSV file as a Materialized Faceted Taxonomy

A classical CSV can be seen as a flat materialized ontology ($\mathcal{F}$,I). Specifically each row corresponds to the description of one object so the rows of the file specify the set Obj. Each object is described by attributes values of the corresponding line. This means that each column heading A_i of the CSV file corresponds to a facet F_i having as name the name of A_i whose distinct values, i.e. all values that occur in that column in the file, constitute the terminology T_i. In other words each row of the CSV file represents the description $D_I(o)$ of an object o.

Consequently we can say that the contents of a classical CSV file can be captured with the notion of materialized flat faceted taxonomy. The same holds for the answers of SPARQL SELECT queries.

5.3 FS Interaction

A materialized faceted taxonomy can be explored by end users through faceted search (and PFS). The lower part of Table 5.1 defines formally the interaction of FDT, i.e. the notion of *restriction* and *zoom-in/out points*. However our objective here is not to describe the interaction. Details are given in the FDT book [38], PFS [45] and an extension for RDF datasets is described in [44].

In brief, one area in the left bar is allocated for each facet and the user can expand and see the terms of the desired facet each accompanied by a natural number signifying how many objects (of the current state) have this value. The objects of the current state (i.e. the focus) are shown in the main area of the window. By clicking on such facet term, the objects of the focus are filtered to those that have that value, and the applicable facets and their terms and counts are recomputed. FS is essentially a session-based method where the user through simple clicks can form complex filterings and during the interaction he is getting an overview of the information space. Apart from clicks (which correspond to hard constraints) the user can right click on a term and enact a preference action, i.e. an action that ranks the focus (the objects of the current state).

MATERIALIZED FACETED TAXONOMIES		
Name	**Notation**	**Definition**
terminology	$\mathcal{T}$	a set of *terms* (can capture categorical/numeric values)
subsumption	$\leq$	a partial order (reflexive, transitive and antisymmetric)
taxonomy	$(\mathcal{T}, \leq)$	$\mathcal{T}$ is a terminology, $\leq$ a subsumption relation over $\mathcal{T}$
broaders of t	$B^+(t)$	$\{ t' \mid t < t'\}$
narrowers of t	$N^+(t)$	$\{ t' \mid t' < t\}$
direct broaders of t	$\mathcal{B}(t)$	$minimal_<(B^+(t))$
direct narr. of t	$N(t)$	$maximal_<(N^+(t))$
Top element	$\top_i$	$\top_i = maximal_{\leq}(\mathcal{T}_i)$
faceted taxonomy	$\mathcal{F} = \{F_1, ..., F_k\}$	$F_i = (\mathcal{T}_i, \leq_i)$, for $i = 1, ..., k$ and all $\mathcal{T}_i$ are disjoint
object domain	Obj	any denumerable set of objects
interpretation of $\mathcal{T}$	I	any function $I : \mathcal{T} \rightarrow 2^{Obj}$
materialized faceted taxonomy	$(\mathcal{F}, I)$	$\mathcal{F}$ is a faceted taxonomy $\{F_1, ..., F_k\}$ and I is an interpretation of $\mathcal{T} = \bigcup_{i=1,k} \mathcal{T}_i$
ordering of two interpretations	$I \sqsubseteq I'$	$I(t) \subseteq I'(t)$ for each $t \in \mathcal{T}$
model of $(\mathcal{T}, \leq)$ induced by I	$\bar{I}$	$\bar{I}(t) = \cup\{I(t') \mid t' \leq t\}$
Descr. of o wrt I	$D_I(o)$	$D_I(o) = \{ t \in \mathcal{T} \mid o \in I(t)\}$
Descr. of o wrt $\bar{I}$	$D_{\bar{I}}(o) \equiv \bar{D}_I(o)$	$\{ t \in \mathcal{T} \mid o \in \bar{I}(t)\} =$ $\cup_{t \in D_I(o)}(\{t\} \cup B^+(t))$
FDT-INTERACTION: BASIC NOTIONS AND NOTATIONS		
focus	ctx	any subset of T such that $ctx = minimal(ctx)$
projection on f. i	ctx_i	$ctx \cap T_i$
Kinds of zoom points w.r.t. a facet i while being at ctx		
zoom points	$AZ_i(ctx)$	$\{ t \in T_i \mid \bar{I}(ctx) \cap \bar{I}(t) \neq \emptyset\}$
zoom-in points	$Z_i^+(ctx)$	$AZ_i(ctx) \cap N^+(ctx_i)$
immediate zoom-in points	$Z_i(ctx)$	$maximal(Z_i^+(ctx)) =$ $AZ_i(ctx) \cap N(ctx_i)$
Restriction over an object set $A \subseteq Obj$		
reduced interpretation	I_A	$I_A(t) = I(t) \cap A$
reduced terminology	T_A	$\{ t \in T \mid \bar{I}_A(t) \neq \emptyset\} =$ $\{ t \in T \mid \bar{I}(t) \cap A \neq \emptyset\} =$ $\cup_{o \in A} B^+(D_I(o))$

Table 5.1: Faceted Information Sources and FDT Interaction

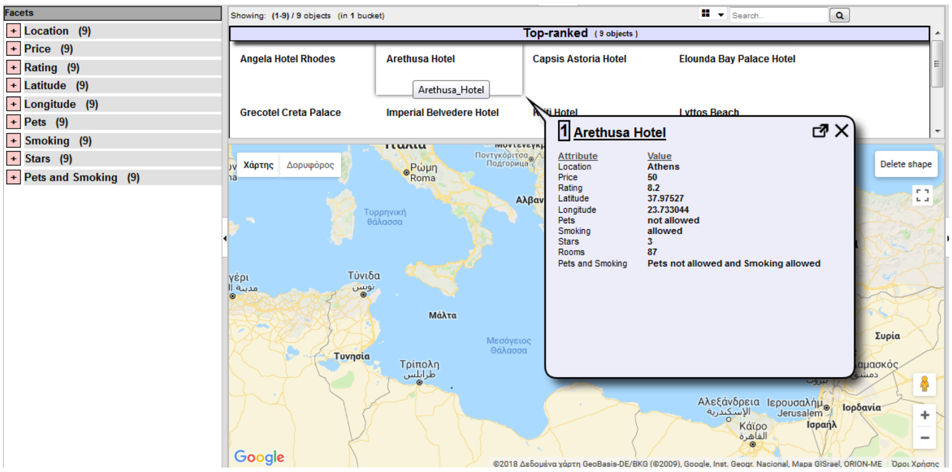
Chapter 6

The Supported Transformations

Let ($\mathcal{F}$,I) be the materialized faceted (flat) taxonomy defined by a CSV file. The objective of the transformation is to produce a new one that is more convenient for exploration. The transformations are those that the presentation designer has issued.

6.1 Visibility

A user can specify which facets would be visible. Nevertheless, the values for these attributes in each object's card are visible, but they will not appear in the Hippalus' facet menu. For example in Figure 6.1, property *Rooms* has been defined hidden and shows the value of property *Rooms* in *Arethusa Hotel*'s card, but this property is not appeared in facets' menu.



1 Arethusa Hotel

Attribute	Value
Location	Athens
Price	50
Rating	8.2
Latitude	37.97527
Longitude	23.733044
Pets	not allowed
Smoking	allowed
Stars	3
Rooms	87
Pets and Smoking	Pets not allowed and Smoking allowed

Figure 6.1: Hidden property *Rooms*

6.2 Facet Type

A user can specify the type of each facet F_i of a faceted taxonomy F. The supported types are *integer*, *float*, *string*, *boolean* and $identifier$. If the type of facet is defined $identifier$, then all values in that facet should be distinct, i.e. no value should appear more than once and the entities will take as names their corresponding values in this facet. In our scenario described in Chapter 1, property *Name* has been defined as $identifier$ and Figure 1.1(b) shows the objects with names their corresponding values in this facet. To set the type of this facet as $identifier$, we left click on facet *Name* and select options *Facet Type...* and $Identifier$ from menus (Fig. 6.2). Moreover, a user can specify that a particular pair of facets contains geographical coordinates. In such cases the user can mark these facets as *Longitude* and *Latitude*. This tagging allows systems (like [27]) to display the objects also in a map. For example, in our mentioned scenario, to set property *Longitude* as contains geographic information for longitude, we left click on facet *Longitude* and select options *Specify Geographic Information...* and *Longitude* from menus (Fig. 6.3). Respectively, to set property *Latitude* as contains geographic information for latitude, we left click on facet *Latitude* and select options *Specify Geographic Information...* and *Latitude* from menus (Fig. 6.4).

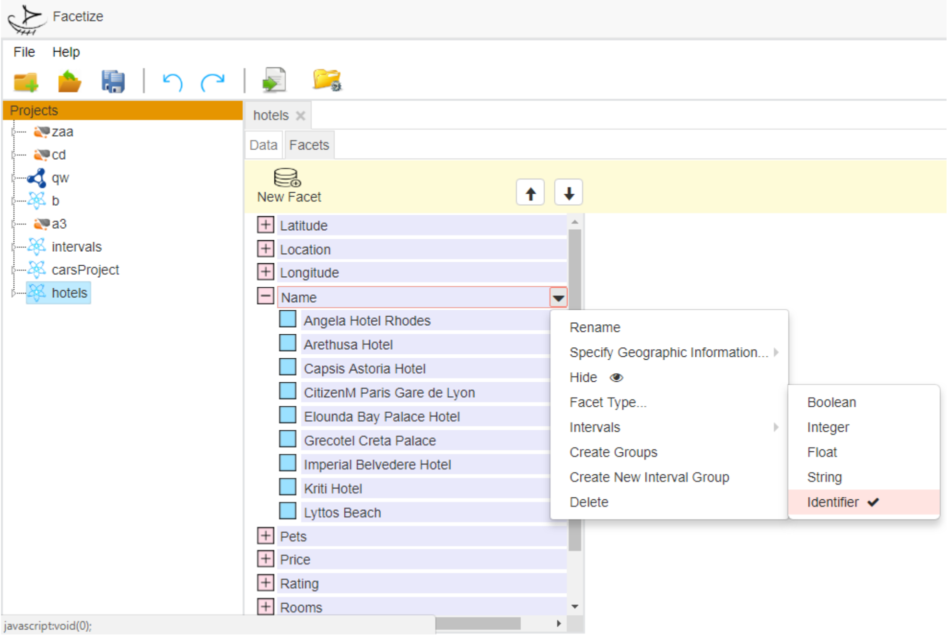


Figure 6.2: Specify property *Name* as type $Identifier$.

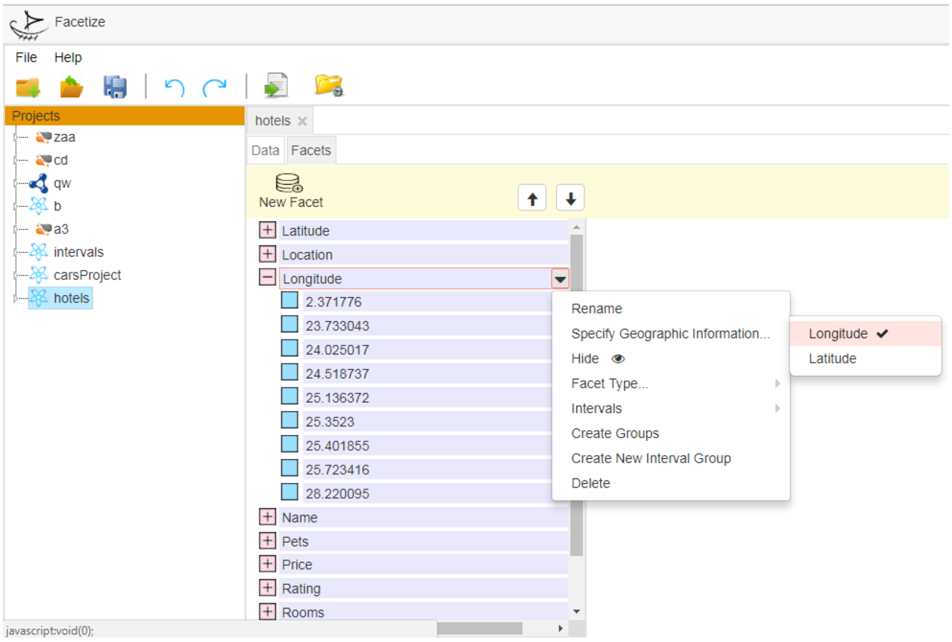


Figure 6.3: Specify property *Longitude* as contains geographic information for longitude.

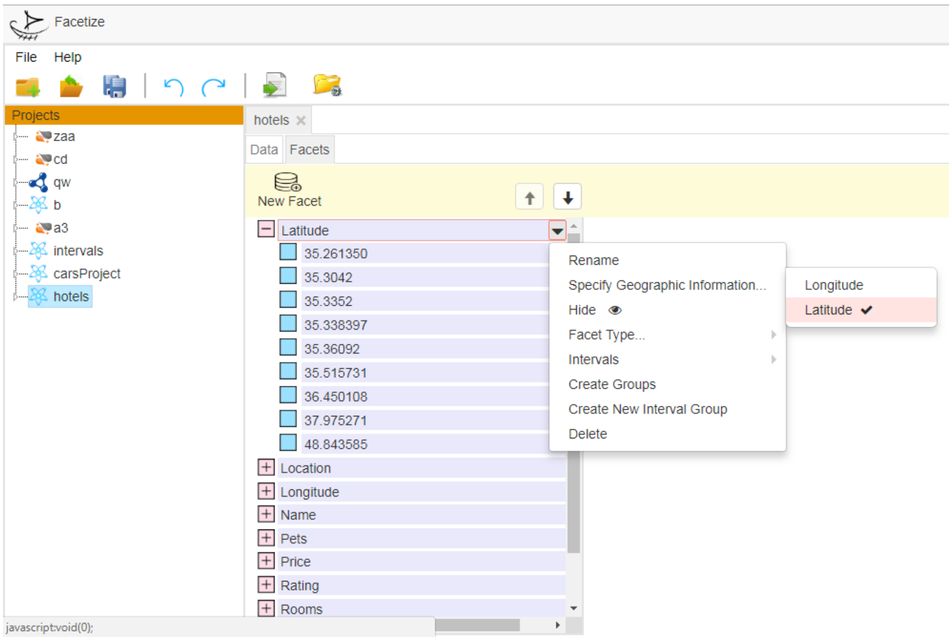


Figure 6.4: Specify property *Latitude* as contains geographic information for latitude.

6.3 Ordering of Facets

The objective is to define a linear order $>_{facets}$ over the set of facets F. This corresponds to the order of appearance of facets in the GUI. In general facet ranking can be done using various methods (e.g. see [44] for a survey). Our objective here is to enable the user to define the desired order.

6.4 Defining Hierarchies

Let $(T, \leq)$ be a taxonomy. In many cases $\leq = \emptyset$. The objective is to organize the values in T hierarchically to avoid cluttering the GUI in case of big terminologies, and to aid the thinning process of faceted search.

The output is $(T \cup T_n, \leq')$ where T are the existing terms (and leaves of the taxonomy) and T_n is the set of the new terms, which are defined as broaders of the terms in T. Below we distinguish taxonomies over categorical value and taxonomies over numerical values.

6.4.1 Hierarchies of Categorical Values

The user defines T_n manually. For each term t in T, the user can specify another term (a new one or one from the existing terminology T) as parent of t. For example, consider that a facet with name *Country* contains the distinct values: *Heraklion*, *Patra*, *Chania*, *Crete*. The user can set as parent of the terms *Heraklion* and *Chania*, the existing term *Crete*; he can also define a new term *islands* and set it parent of *Crete*.

6.4.2 Hierarchies of Numerical Values: Intervals

The user can define intervals that group the values of a facet. To define K intervals, the user can define K new terms (T_n) each corresponding to one interval. Each one can have a name, and its two bounds which can be closed or open. This feature is very useful for defining price intervals, and intervals of measurements in general. For example, from our running example in the introductory chapter, if the user wants to create an interval hierarchy in facet *Price* with values from 22 to 69, we left click on this facet and select option *Create New Interval Group* from menu (Fig. 6.5). Then, a (bootstrap.js) modal dialogue appears and we specify low value 22 and high value 69 for this interval (Fig. 6.6).

6.5 Derived Facets

The user can define new facets using various functions that the system provides. The values of new derived facets can be based on values of existing facets. For example, consider a new facet Fn, whose terminology is defined as: Tn =

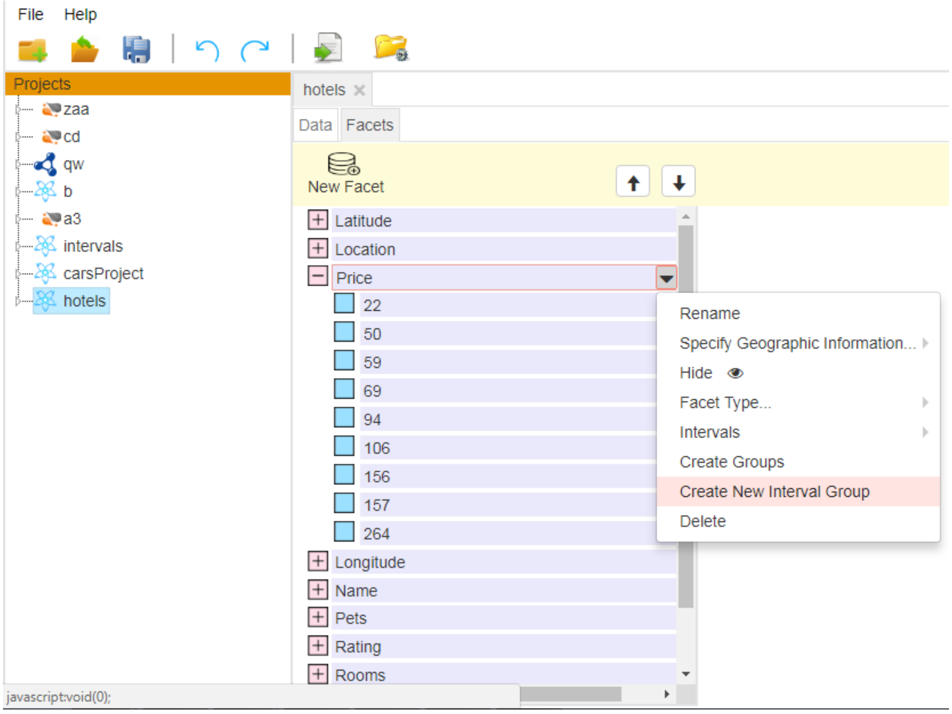
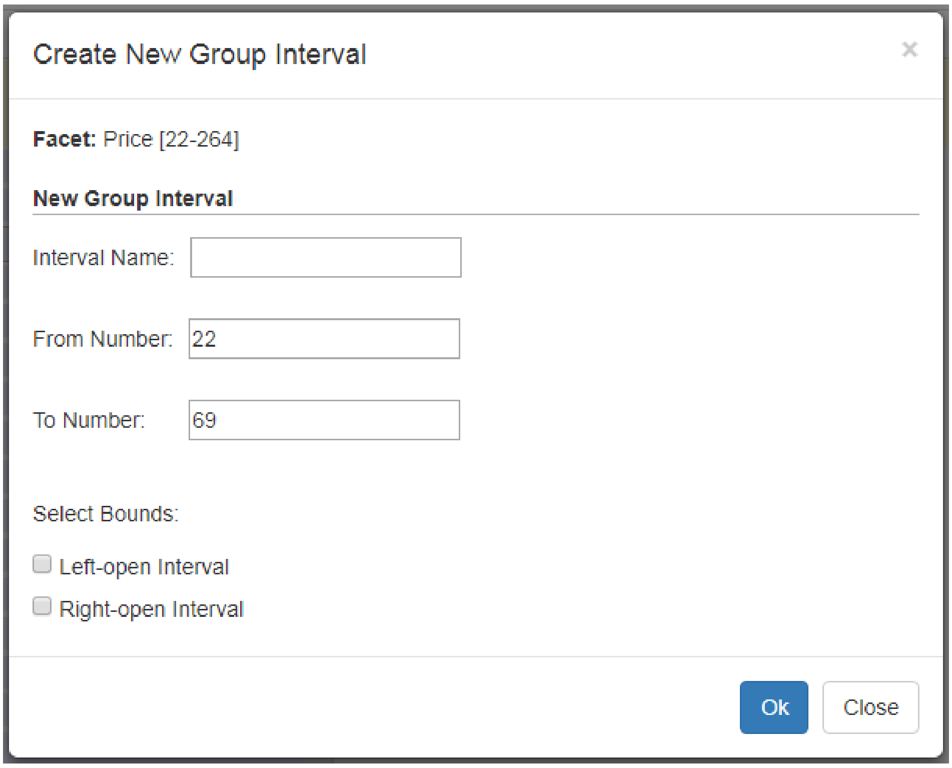


Figure 6.5: Specify interval hierarchy in facet *Price*.

`concat("math", facetAM)`. In this case, $facetAM$ is an existing facet that contains student identifier numbers and *concat* is a system function that joins the values that receives as parameters in one string value. Then, the new terminology Tn will contain values of the form:
`mathstudentId1,...,mathstudentIdk`,
where `studentId1,...,studentIdk` are the distinct values of $facetAM$.

6.6 Linear, Logarithmic and User Defined Intervals

Supposing a facet contains number values, then a user can specify linear, logarithmic or user defined intervals. In case of linear and logarithmic intervals, the user can specify via `Facetize` the number of uniformly distributed intervals that he wants to group the values of the facet and Hippalus will create and show them in facets' menu. For example, consider a facet with number values: 8000, 9000, 12000, 17000, 19000, 22190. If a user specifies 3 logarithmic divisions via our system, then they will be created logarithmic intervals: [7990, 11240), [11240, 15800), [15800, 22190] in Hippalus. In case of user defined intervals, the user can specify via `Facetize` the bounds of the intervals, which can be closed or open, as well as a name for the interval. For example, consider a facet with number values: 1, 2, 3, 4, 5, 6. If a user specify a user defined interval with name “Low values”, that



Create New Group Interval

Facet: Price [22-264]

New Group Interval

Interval Name:

From Number: 22

To Number: 69

Select Bounds:

□ Left-open Interval

□ Right-open Interval

Figure 6.6: Modal for create interval hierarchy.

has low value: 3 and high value: 6, with right-open and left-closed bounds, then it will be created interval “Low values” in Hippalus that contains values 3, 4, 5.

6.7 Synopsis of Supported Data Transformations

To sum up, with `Facetize` users can delete specific rows from a dataset, insert and edit a row. Also, they can create hierarchies of values, specify a type for a facet or geographic coordinates for a pair of facets. It supports operations for edit distinct values of a facet, hide, reorder and delete a facet. Also they can create new facets using functions for define expressions. Moreover, users can specify linear, logarithmic and user defined intervals for facets that contain numerical values.

Chapter 7

The System `Facetize`

Here we discuss the implementation of `Facetize` (in §7.1), the supported transformations (in §7.2), the notion of project (in §7.3), an indicative interaction example (in §7.4), and then (in §7.5) how it tackles the error correction requirements.

7.1 Implementation

`Facetize` is a Web Application, based on the architecture of client-server. Figure 7.1, shows the implementation architecture of our system. The presentation layer (*front end*) is implemented using HTML, CSS for rendering elements appropriately on screen, and JavaScript libraries such as `Bootstrap.js` to create responsive content. The data access layer (*back end*), is implemented using Java Servlets technologies and Jena Semantic Web framework[1]. To run the editor the user should have installed at his computer Java Runtime Environment (JRE) and a Java Servlet Container. Users can run from command line the WAR file of `Facetize` (e.g. java -jar *name_of_Java_Servlet_Container*.jar Facetize.war). Finally, it can be accessed by loading http://localhost:8080 in browser.

7.2 Supported Transformations

In brief, `Facetize` supports all transformation requirements described in §3.1. As regard hierarchies, the user is able to move values in existing hierarchies, and add intermediate terms. In case of string values, the user is able to use functions to define expressions for creating groups of values with same prefix and values that start with the same range of letters for aiding the exploration. For numerical facets, the specification of intervals is supported and the user can define linear intervals, logarithmic intervals, and intervals with specific bounds.

[1] http://jena.apache.org/

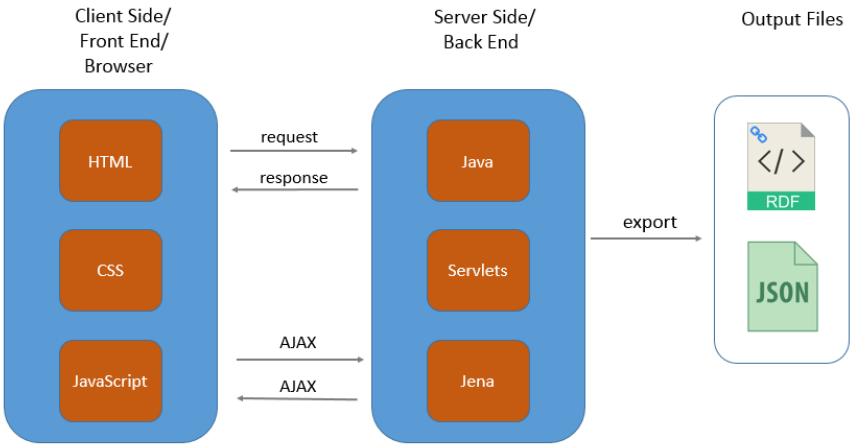


Figure 7.1: Image with the implementation architecture of system Facetize

7.3 The Notion of Project

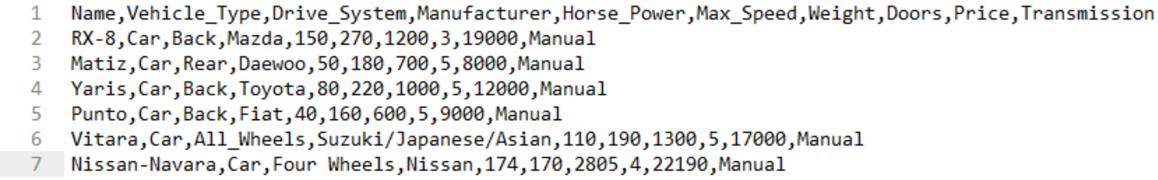
Since Facetize can be considered as an authoring environment for producing explorable datasets, Facetize supports the notion of project, allowing the user to gradually configure the desired presentation and also to update (refresh) the underlying dataset without losing the transformations that they have been defined. In brief it supports all requirements described in §3.2.

Each project has a name, an input dataset and a configuration of the transformations that should be applied for producing an output dataset that is suitable for exploration by Hippalus. The structure of the project is described next. In brief, Facetize supports three types of project: (a) Project with Single File Dataset, (b) Project with Multiple Files, (c) Project with SPARQL Query.

7.3.1 Project with Single File Dataset

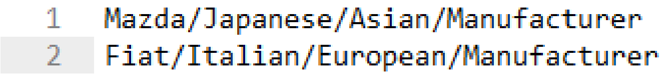
The dataset files that this type of project supports are TSV, CSV and JSON. Figure 7.2, shows a CSV dataset file that contains information for 6 entities, 10 properties and an internal hierarchy. More specifically, the first row in a CSV or TSV dataset file is the header row. Each column in this row must contain a Property name (the name of the corresponding facet).

Hierarchies can be specified (a) internally, (b) externally, or (c) by both methods. According to method (a), the user can specify the complete path from the instance to its root inside dataset file, e.g. Suzuki/Japanese/Asian, (Figure 7.2). According to method (b), the hierarchy is specified through a configuration file that contains hierarchical information about some or all the values of the tabular data file. Figure 7.3 shows a plain text file that contains two external hierarchies. Specifically, each line contains hierarchical information about one (non literal) value. The syntax of a single line is: name + path from the specific value to the root of the hierarchy separated with slashes (/), e.g. Mazda/Japanese/Asian/-Manufacturer, where Manufacturer is the name of the facet that the mentioned hierarchy belongs. The order of rows is not important and the import of a configuration file in each project is optional, since the user can create the desired hierarchies though the editor.



```
Name,Vehicle_Type,Drive_System,Manufacturer,Horse_Power,Max_Speed,Weight,Doors,Price,Transmission
RX-8,Car,Back,Mazda,150,270,1200,3,19000,Manual
Matiz,Car,Rear,Daewoo,50,180,700,5,8000,Manual
Yaris,Car,Back,Toyota,80,220,1000,5,12000,Manual
Punto,Car,Back,Fiat,40,160,600,5,9000,Manual
Vitara,Car,All_Wheels,Suzuki/Japanese/Asian,110,190,1300,5,17000,Manual
Nissan-Navara,Car,Four Wheels,Nissan,174,170,2805,4,22190,Manual
```

Figure 7.2: CSV dataset file with internal hierarchies



```
Mazda/Japanese/Asian/Manufacturer
Fiat/Italian/European/Manufacturer
```

Figure 7.3: Plain text file with external defined hierarchies

7.3.2 Project with Multiple Files

Facetize supports input from multiple CSV or TSV files, located at a specified folder. The files that such project can contain are:

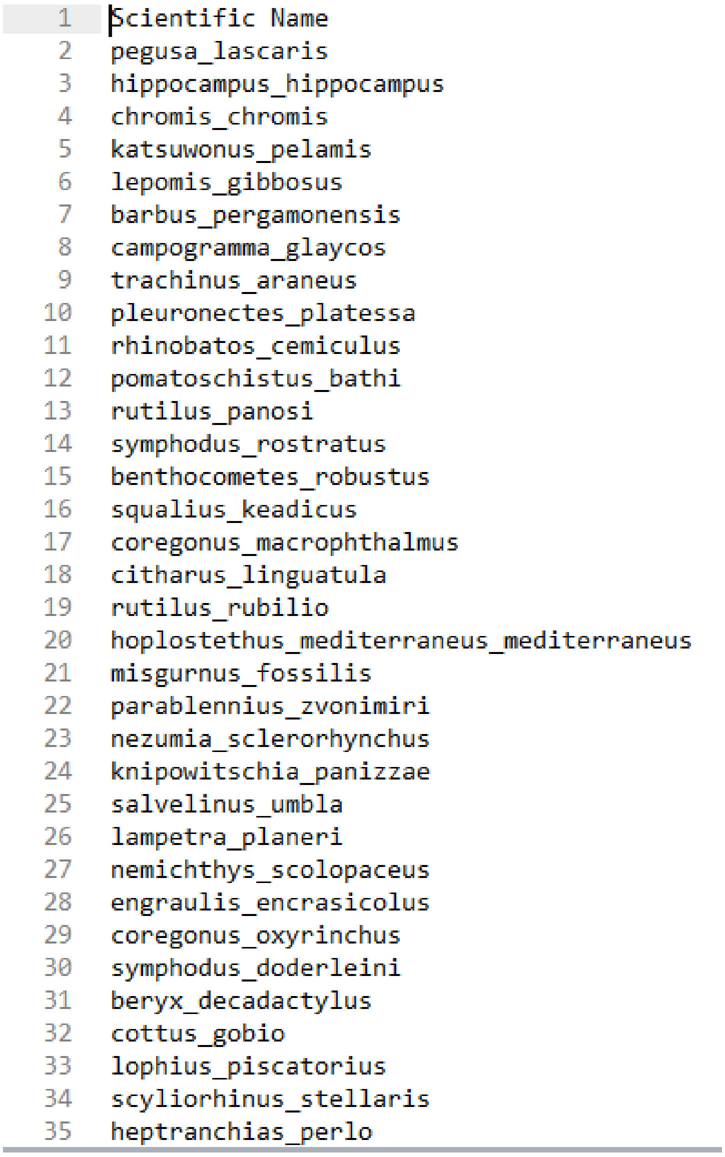
1. **Object Id File**

   The object id file is specified explicitly by the user and is the one that holds all the ids of the objects of this project. They are actually the names of the entities. It is a CSV/TSV file, with a single column. The header row provides a human meaningful name, e.g. Scientific Name, Product No, etc, which is skipped by the translator, because Hippalus sees only unique id values. It does not need to know what these ids stand for.

2. **Dimension (Property) Files**

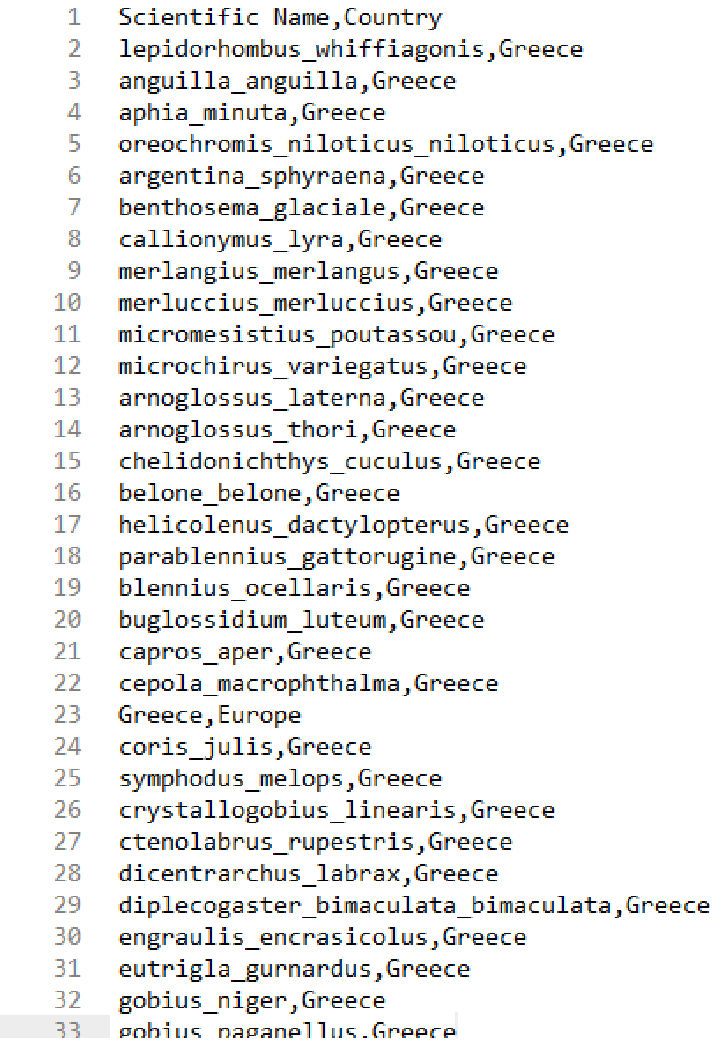
   A dimension file contains information about a single property. It is a good practice that the file name is the same with the property name, e.g. length.csv for the length property, etc. These files have 2 columns: the first is for the object id and it is the column of object id file that mentioned above and the second for the value that the specified entity has for this property. The hierarchies can be straightforwardly represented. We just have to put the child at the first column and its parent at the second column. The hierarchies can be expressed anywhere in the file (at the beginning, end, or mixed). The translator will detect that a property is hierarchical when a first column item is not a valid object id, so the current property is hierarchical. For example, in line 23, in Figure 7.5, is defined an hierarchy.

This structure is convenient for enriching the dataset with more dimensions. For example, Figure 7.4 shows an object id file that contains distinct names for fish species, and Figure 7.5 shows the content of a property file for dimension *Country*.



```
Scientific Name
pegusa_lascaris
hippocampus_hippocampus
chromis_chromis
katsuwonus_pelamis
lepomis_gibbosus
barbus_pergamonensis
campogramma_glaycos
trachinus_araneus
pleuronectes_platessa
rhinobatos_cemiculus
pomatoschistus_bathi
rutilus_panosi
symphodus_rostratus
benthocometes_robustus
squalius_keadicus
coregonus_macrophthalmus
citharus_linguatula
rutilus_rubilio
hoplostethus_mediterraneus_mediterraneus
misgurnus_fossilis
parablennius_zvonimiri
nezumia_sclerorhynchus
knipowitschia_panizzae
salvelinus_umbla
lampetra_planeri
nemichthys_scolopaceus
engraulis_encrasicolus
coregonus_oxyrinchus
symphodus_doderleini
beryx_decadactylus
cottus_gobio
lophius_piscatorius
scyliorhinus_stellaris
heptranchias_perlo
```

Figure 7.4: Object Id file



```
Scientific Name,Country
lepidorhombus_whiffiagonis,Greece
anguilla_anguilla,Greece
aphia_minuta,Greece
oreochromis_niloticus_niloticus,Greece
argentina_sphyraena,Greece
benthosema_glaciale,Greece
callionymus_lyra,Greece
merlangius_merlangus,Greece
merluccius_merluccius,Greece
micromesistius_poutassou,Greece
microchirus_variegatus,Greece
arnoglossus_laterna,Greece
arnoglossus_thori,Greece
chelidonichthys_cuculus,Greece
belone_belone,Greece
helicolenus_dactylopterus,Greece
parablennius_gattorugine,Greece
blennius_ocellaris,Greece
buglossidium_luteum,Greece
capros_aper,Greece
cepola_macrophthalma,Greece
Greece,Europe
coris_julis,Greece
symphodus_melops,Greece
crystallogobius_linearis,Greece
ctenolabrus_rupestris,Greece
dicentrarchus_labrax,Greece
diplecogaster_bimaculata_bimaculata,Greece
engraulis_encrasicolus,Greece
eutrigla_gurnardus,Greece
gobius_niger,Greece
gobius_paganellus,Greece
```

Figure 7.5: CSV property file

7.3.3 Project with SPARQL Query

For exploiting the wealth of Linked Data and the available SPARQL endpoints, Facetize allows the user to specify the address of a SPARQL endpoint and the SPARQL SELECT query to be sent. For example a query with select clause of the form SELECT ?Speed ?Price ?Weight will lead the tool to create 3 properties: Speed, Price and Weight. Moreover the user can save queries and endpoints in the favourites list. Then, the user can re-run a query or delete it from the list. Also note that if the SPARQL endpoint supports SPARQL-LD [12, 13], then that implies that Facetize projects can be defined by extracting information from RDF dumps and HTML pages with RDFa. For example, the query in Figure 7.6 returns all co-authors together with their publications.

```
SELECT DISTINCT ?authorName ?paper WHERE {
SERVICE <http://users.ics.forth.gr/~fafalios/> {
?p <http://purl.org/dc/terms/creator> ?author
FILTER(?author != <http://dblp.l3s.de/d2r/resource/authors/Pavlos_Fafalios>) }
SERVICE ?author {
?author <http://xmlns.com/foaf/0.1/name> ?authorName .
?paper <http://purl.org/dc/elements/1.1/creator> ?author.
}
}
```

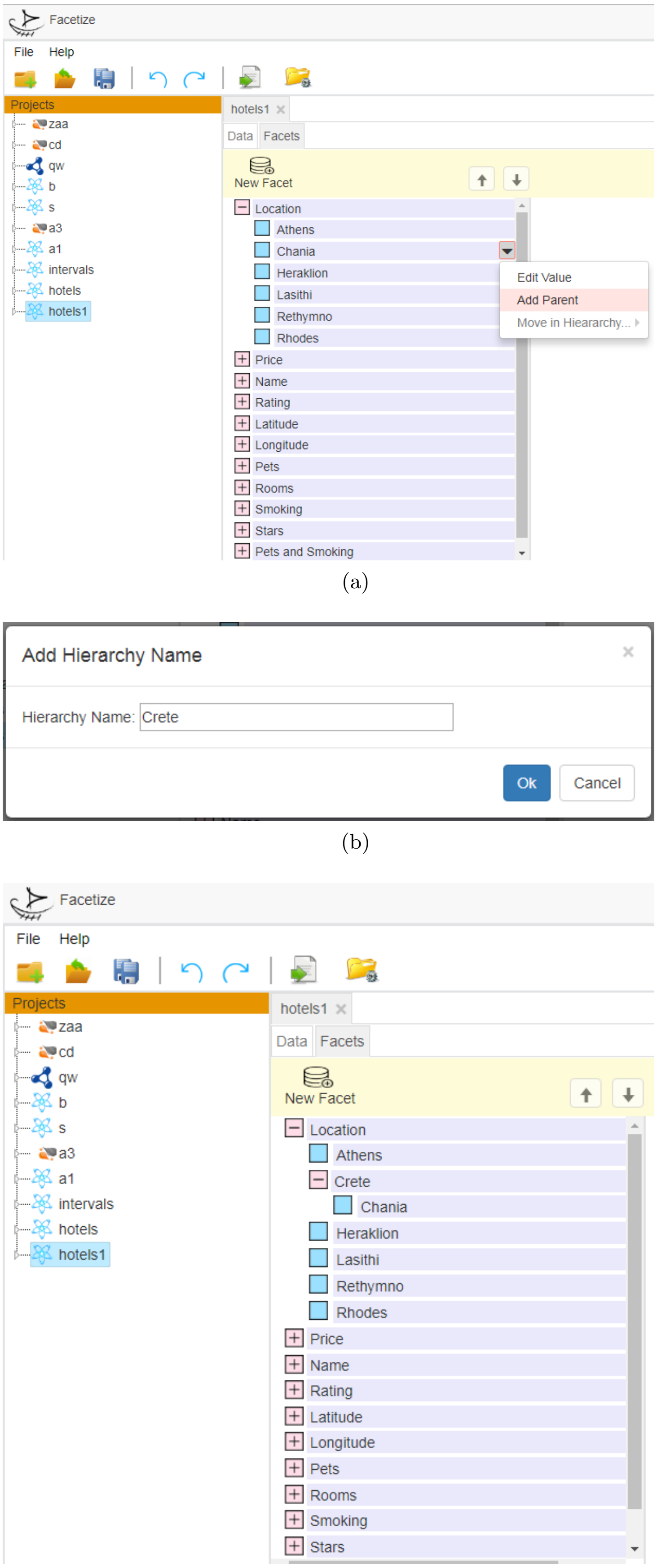
Figure 7.6: Example of a SPARQL query that reads and queries RDF data embedded in a Web page (as RDFa) at query execution time.

7.4 Interaction Example

Here we sketch a scenario for demonstrating some parts of the functionality

7.4.1 Defining Hierachies

Here we sketch a scenario demonstrating how a user can organize the hierarchically the values of a facet. Figure 7.7.a shows the list of facets of the running example described in the introductory chapter. Notice that facet *Location* contains 6 distinct values. By left clicking on the term *Chania*, and selecting the option *Add Parent* from the menu, a (bootstrap.js) modal dialogue appears with a text field for adding the parent value (see Fig. 7.7.b). If for example the user inserts the value *Crete*, the hierarchy Chania/Crete will be created in facet *Location* (as shown in Fig. 7.7.c).



(a)

Add Hierarchy Name

Hierarchy Name: Crete

(b)

(c)

Figure 7.7: (a) Click on a term and open menu, (b) Set parent *Crete* for term *Chania* in the hierarchy, (c) Hierarchy Chania/Crete has been created

7.4.2 Refreshing the Dataset of an existing Project

Every time the user applies a transformation to the dataset of a Facetize project, that transformation is saved in JSON format in a file in the project's folder. For each transformation, its type, as well as related information, are saved, for enabling reapplying the transformation in the future.

For example, for the transformations of the introductory chapter: i) Properties *Longitude* and *Latitude*, must be marked as properties that contain geographic information for longitude and latitude respectively, and ii) The property *Rooms* should be hidden or deleted from facet's list (if we choice to hide it), the log file keeps the following information:

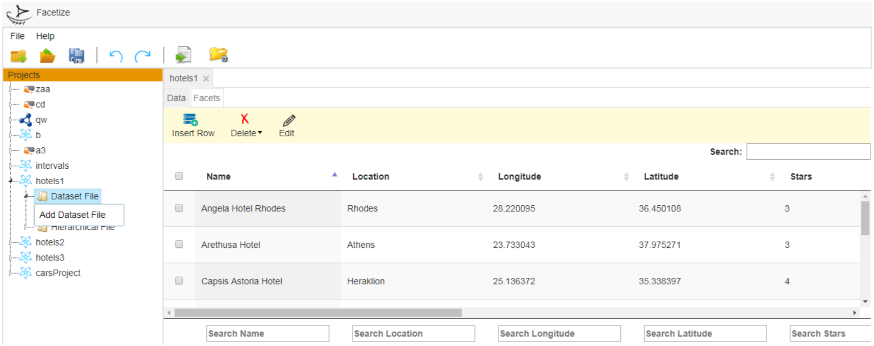
```
{"facetName": "Longitude",
"geoInfo": "Longitude",
"transformationType": "geoInfo"},

{"facetName": "Latitude",
"geoInfo": "Latitude",
"transformationType": "geoInfo"},

{"facetName": "Rooms",
"visibility": "hidden",
"transformationType": "facet-visibility"
}
```

The actions that are not logged are the following: deletions, additions and edits of particular rows, edits of values, deletions and additions of facets, additions of new dataset files and rename of a facet. The user can re-apply one or several transformations, and delete transformations from the history. In case a transformation cannot be applied, the user is notified with a message.

To refresh the dataset of an existing project with a new one, the user has to right click on the *Dataset File* folder of the project, on the left sidebar with the projects and select option *Add Dataset File* (Fig. 7.8). Then he has to browse on the disc and select the new file (Fig. 7.9). If the project contains data transformations from previous dataset, the system will ask the user if he wants to apply the same range of transformations in the new dataset with a dialog box (Fig. 7.10). If the user selects *Apply All* in the dialog box, then the system will inform the user with a message which of the transformations were applied successfully and which not (Fig. 7.11).

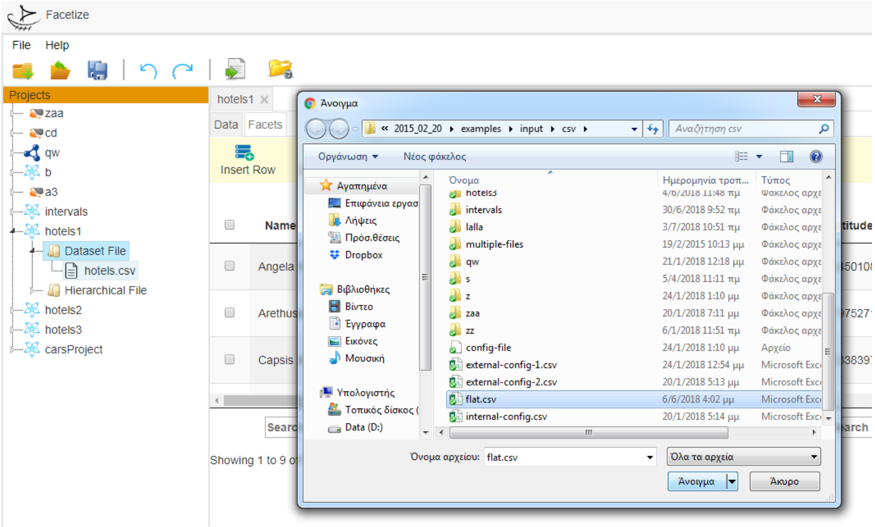
7.5 Errors

Facetize supports a variety of transformation requirements, as mentioned in Section 3.1. In terms of the data error types that were mentioned in Section 4.1,



	Name	Location	Longitude	Latitude	Stars
□	Angela Hotel Rhodes	Rhodes	28.220095	36.450108	3
□	Arethusa Hotel	Athens	23.733043	37.975271	3
□	Capsis Astoria Hotel	Heraklion	25.136372	35.338397	4
	Search Name	Search Location	Search Longitude	Search Latitude	Search Stars

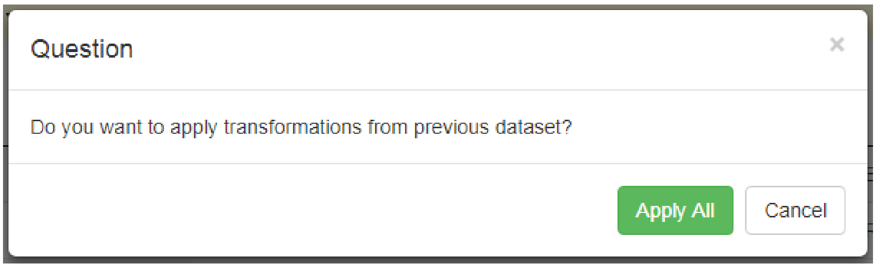
Figure 7.8: Add new dataset file.



Όνομα	Ημερομηνία τροπ...	Τύπος
hotels3	4/6/2018 11:48 πμ	Φάκελος αρχε
intervals	30/6/2018 9:52 πμ	Φάκελος αρχε
lalla	3/7/2018 10:51 πμ	Φάκελος αρχε
multiple-files	19/2/2015 10:13 μμ	Φάκελος αρχε
qw	21/1/2018 12:18 μμ	Φάκελος αρχε
s	5/4/2018 11:11 πμ	Φάκελος αρχε
z	24/1/2018 1:10 μμ	Φάκελος αρχε
zaa	20/1/2018 7:11 μμ	Φάκελος αρχε
zz	6/1/2018 11:51 πμ	Φάκελος αρχε
config-file	24/1/2018 1:10 μμ	Αρχείο
external-config-1.csv	24/1/2018 12:54 μμ	Microsoft Exc
external-config-2.csv	20/1/2018 5:13 μμ	Microsoft Exc
flat.csv	6/6/2018 4:02 μμ	Microsoft Exc
internal-config.csv	20/1/2018 5:14 μμ	Microsoft Exc

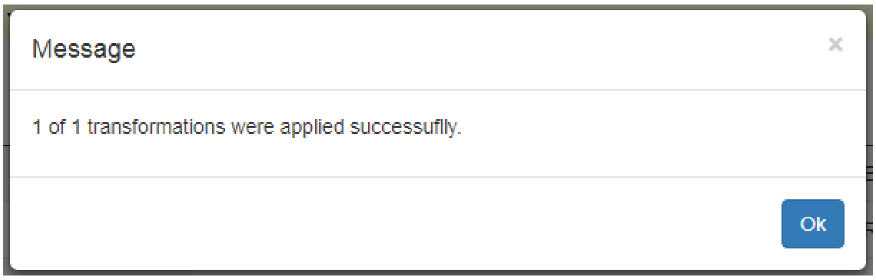
Όνομα αρχείου: flat.csv

Figure 7.9: Select file from disc.


Question

Do you want to apply transformations from previous dataset?

Figure 7.10: Apply transformations dialog box.


Message

1 of 1 transformations were applied successuflly.

Figure 7.11: Message for applied transformations.

`Facetize` assists the user to detect and remove them Specifically, in case of *Rule*, *Pattern violations* and *Outliers* errors in the dataset, the user can edit the values that cause the inconsistencies and replace them with other values. In case of *Duplicates*, the user can delete or edit duplicate rows. For example, consider from our running example in the introductory chapter that a user wants to replace value *Iraklio* in property *Location* with value *Heraklion* (Fig.7.12). By left clicking on the term *Iraklio*, and select the option *Edit Value* from the menu (Fig.7.13), a (bootstrap.js) modal dialogue appears with a text field for replace it with the value *Heraklion* (see Fig. 7.14, 7.15).

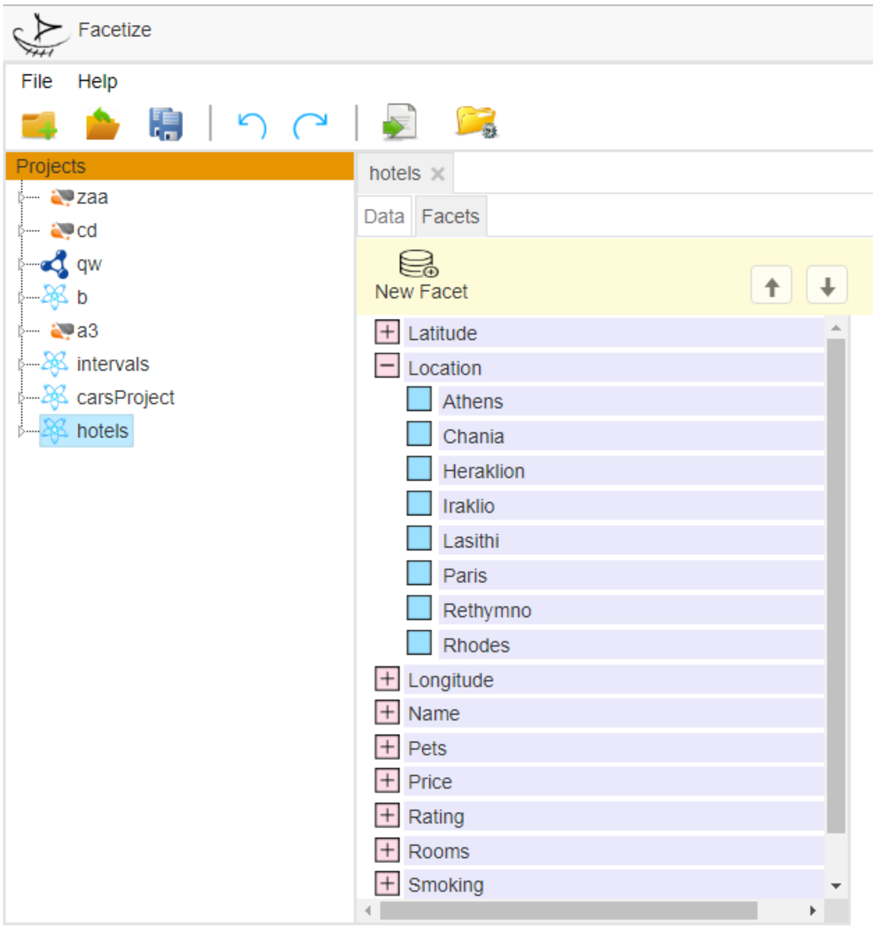


Figure 7.12: Facet *Location* with value *Iraklio*.

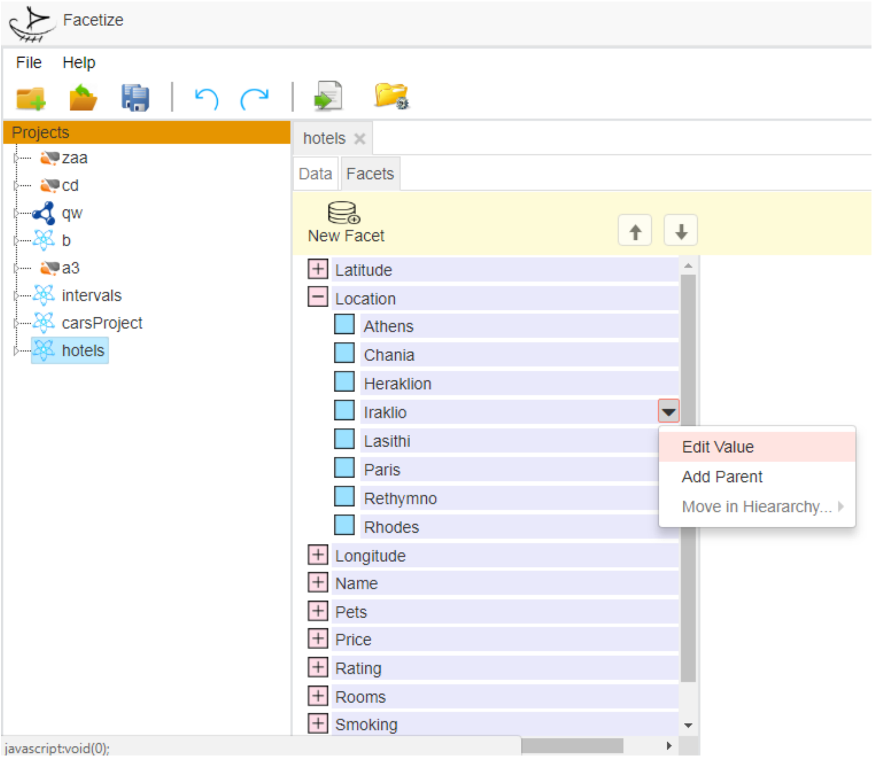
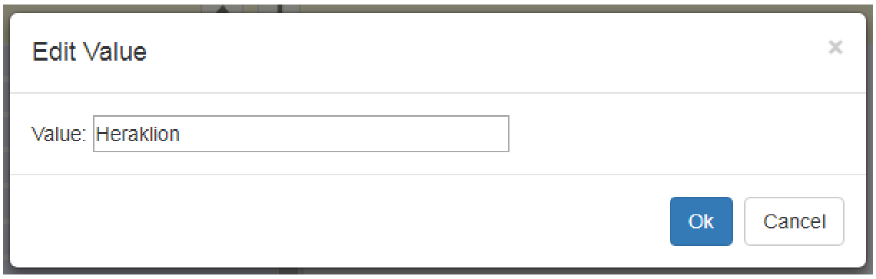


Figure 7.13: Click option *Edit Value* from menu.


Edit Value

Value: Heraklion

Figure 7.14: Replace with value *Heraklion*.

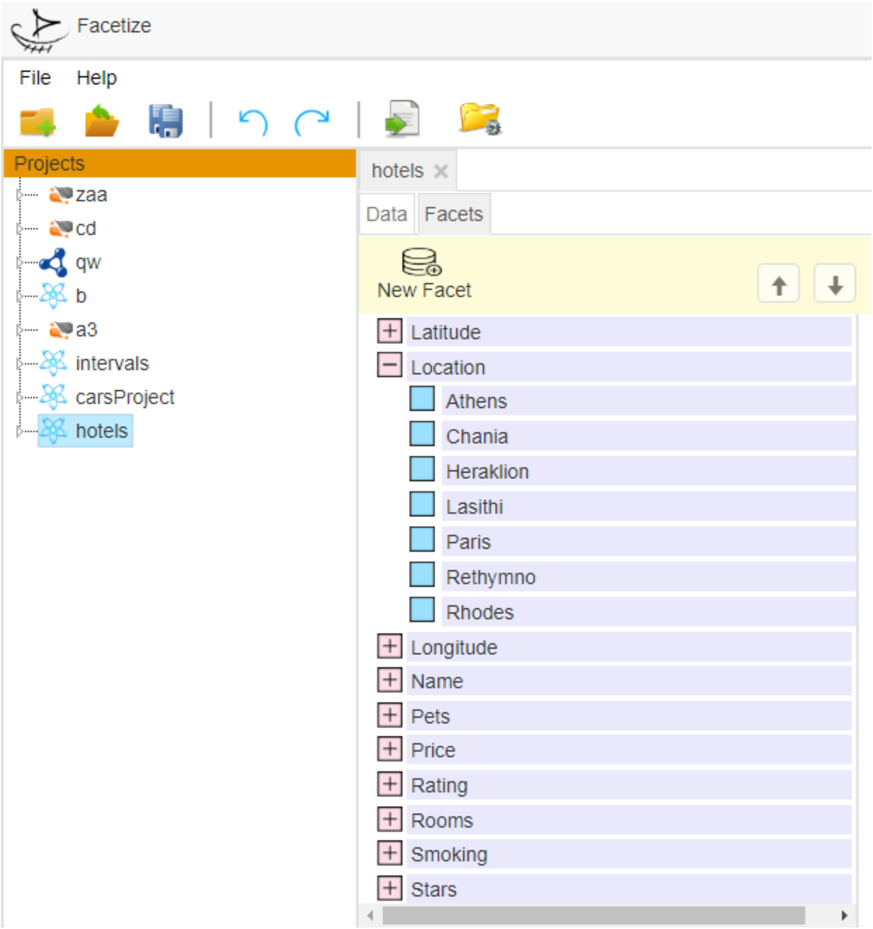


Figure 7.15: Facet *Location* without value *Iraklio*.

Chapter 8

Evaluation

Section 8.1 compares the functionality of Facetize with the functionality of related systems. Section 8.2 discusses the scalability and efficiency of Facetize. Section 8.3 reports the results of an evaluation with users. Finally, Section 8.4 discusses issues related to the improvements of Facetize in the future.

8.1 Comparing with the Functionality of Related Systems

Table 8.2, shows the features of each data cleansing system that was mentioned in Section 4.2. We observe that one distinctive feature of Facetize is that it supports the creation of hierarchies and numeric intervals and these features are crucial for managing the complexity of large datasets, and producing datasets that can be easily explored. Moreover, Facetize can fetch data directly from SPARQL endpoints, making it appropriate for dynamic datasets. Finally, users have the ability to reapply the same range of transformations to an existing project and this is very convenient in case they want to refresh the dataset of a project (with a new version of the dataset) and would like to apply the same transformations.

8.1.1 Comparison using the Running Example

We tried to carry out scenario of the introductory section by ourselves, without the involvement of any other users, using the tools mentioned in Table 8.2, i.e. DataPreparator, Potters Wheel, WinPure Clean, Karma, OpenRefine, Facetize. We wanted to check what transformations were feasible to perform and how much time it took to make them. Of course none of these tools (apart from Facetize) can produce an RDF file that is directly loaded to Hippalus, however we wanted to test the rest aspects of the scenario.

The results are shown in Table 8.1, where we can see which of the 10 requirements (listed in the introductory section) were possible to perform with each tool how much time it took us to carry out them. As we can see, with DataPreparator

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Time
WinPure Clean											-
Data Preparator					✓	✓					1 m
Potters Wheel						✓	✓				2 m
Karma		✓			✓		✓				3 m
Open - Refine	✓	✓			✓	✓	✓			✓	6 m
Facetize	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	7 m

Table 8.1: Table with requirements that were done by each tool.

we managed to perform requirements 5, 6 in 1 (one) minute, with Potters Wheel we managed to perform requirements 6, 7 in 2 (two) minutes, with WinPure Clean we did not manage to satisfy any of the requirements, with Karma we managed to perform requirements 2, 5, 7 in 3 (three) minutes, and finally, with OpenRefine we managed to perform requirements 1, 2, 5, 6, 7, 10 in 6 (six) minutes, i.e. it is the tool that covered most of the requirements. With Facetize we managed to perform all requirements in 7 (seven) minutes.

8.2 Scalability and Efficiency

Here, we discuss issues related to the efficiency and scalability of Facetize. The measurements that are reported below and in Table 8.3, were performed using a PC with CPU of a base clock speed of 2.30 GHz and 3,00 GB RAM. The loading of a project with a dataset file of 720 rows and 7 columns takes around 1 second (1,158 ms). In Table 8.3, we see the measurements of time that it took to load projects with datasets of different number of rows that contain 37 columns. As we can see, the more rows a dataset file contains, the more time Facetize needs to load it and execute different operations. This happens because at the beginning, Facetize has to read the file, to create the structures with properties, entities and hierarchies. Then, if no error occurs, it scans the structures to create the table and list with facets in the *Data* and *Facets* tab of Facetize project. Just indicatively, the loading of a dataset file with 10,000 of rows and 18 columns takes much more time (specifically 84,005 ms), due to the time required to parse the file and create the internal structure that is required. In case there are more rows in the dataset, the error: *"Uncaught RangeError: Maximum call stack size exceeded..."* is be thrown in browser from jQuery[1] library when tries to create the table with the data. This is caused due to many calls to a function that is responsible for rendering the rows in the table with the checkboxes. The times this function will be called is analogous to the number of rows. A possible solution for this problem in the future, would be to create the table in server-side or to use another javascript library for data table creation.

The rest operations are quite fast. For editing distinct values, the system scans

[1] https://jquery.com/

Table 8.2: Features of data cleansing systems

Category	Feature	**DataPreparator**	**Potters Wheel**	**WinPure Clean**	**Karma**	**OpenRefine**	**Facetize**
Import/Export	Import/Export text files, EXCEL	✓		✓		✓	✓
	Import/Export Data from databases	✓		✓			
	Import/Export RDF files				✓	✓	✓
	Export R2RML/JSON files				✓		
	Execute SPARQL queries to retreive data from a source						✓
Data Visualization	Ability to view data in plain text as well as in tabular format					✓	
Data cleaning: Rows	Deletion of rows					✓	✓
	Removal of duplicate rows			✓		✓	
	Addition of row				✓	✓	✓
	Removal of empty rows					✓	✓
Data cleaning: Values	Deletion of records containing missing values	✓					✓
	Character removal, text replacement, date conversion	✓					
	Value editing				✓	✓	✓
	Impute missing values	✓		✓		✓	
	Creation of value hierarchies						✓
	Creation of intervals for arithmetic values						✓
	Correction of bad values using string similarity metrics				✓		
	Clustering of the values in a property that contain specific characters (e.g. same prefix)					✓	✓
	Delete/move attributes	✓					✓
Data cleaning: Columns	Filtering			✓		✓	✓
	Renaming of Columns				✓	✓	✓
	Set column types				✓	✓	✓
	Creation of new columns using expression					✓	✓
	Split of columns		✓		✓		
	Deletion of columns		✓			✓	✓
	Rearrangement of columns					✓	✓
	Hiding of columns					✓	✓
	Addition/merging of columns		✓		✓		
	Split records into columns	✓	✓				
Editor Facilities	Undo/redo transformations		✓			✓	✓
	Display of the distinct values of a property and number of their occurrences					✓	
	Ability to reapply same range of transformations						✓
Other Export Functions	Export macro or a C program, or a Perl program		✓				
	Extraction of Entities				✓		
	Export of Statistics, Table, File, Database, Visualize	✓					

the structure with the properties to find out the column that contains the value to edit and then, it read all entities for replacing the delected. Finally, it creates again the facets' list. In the scenario of Chapter 1, the time it takes to replace the value *Iraklio* with value *Heraklion* is 137 ms. The operations for deleting, creating, and moving items in an existing hierarchy, require to scan all hierarchies that exist in a project. The required time is analogous to the number of hierarchies. In our scenario, the time it takes to create the hierarchy *Heraklion/Crete* is 19 ms.

Rows	Columns	Time
1,000	37	12,268 ms
2,000	37	21,154 ms
3,000	37	29,480 ms
4,000	37	35,938 ms
5,000	37	55,594 ms
6,000	37	75,476 ms
7,000	37	80,665 ms
8,000	37	82,004 ms
9,000	37	87,543 ms
10,000	37	90,369 ms

Table 8.3: Table with measurements of time to load projects with datasets of different number of rows and 37 columns.

8.3 Task-based Evaluation with Users

We conducted a task-based evaluation with users with the following objectives: (a) to get general and specific feedback from users, (b) to test the usability of `Facetize`. The second objective refers to the effectiveness and the overall satisfaction of the users while interacting with the system. More specifically, we need to test: i) how fast a user who has never used the system before, can accomplish basic tasks (*Ease of learning*), ii) how often users make errors while using the system, how serious the errors are, and how users recover from the errors (*Error frequency and severity*), iii) to what extend the users were satisfied by this application. Finally we want to collect comments and feedback for improving the approach and the system in the future (*Subjective satisfaction*).

For the purposes of evaluation, we used the scenario described in Chapter 1 with the dataset of hotels of Figure 1.1(a). We prepared a simple text tutorial of 45 slides, a questionnaire and a file with the description of the aforementioned scenario. The tutorial is available in the Web[2]. We invited by email various persons to participate in the evaluation voluntarily. The users were asked to carry out the tasks and to fill the questionnaire. It was stated to them clearly, that they should not rush up. The participation to this evaluation was optional. Twenty persons (20), eventually participated. The number was sufficient for our purposes, since according to [15] 20 evaluators are enough for getting more than 95% of

[2] `http://ophelia.ics.forth.gr/svn/Thesis/KokolakiThesis/2_Thesis/tutorial.pdf`.

the usability problems of a user interface. In numbers, the participants were 11 (55%) female and 9 (45%) male, with ages ranging from 18 to 64 years. As regards occupation and skills, users have studied Computer Science. In detail, 5 (25%) were undergraduate students, 12 (60%) of them postgraduate students and 3 (15%) computer engineers and researchers.

8.3.1 Questionnaire

We used a questionnaire that users had to fill it in, and then send it back to us by email. The questionnaire is shown below. Moreover, after each question, we show the results of the survey in the form of percentages written in bold.

Questions:

1) How many mistakes did you make ?

   □ No mistake (**20%**)

   □ 1-3 mistakes (**45%**)

   □ 3-6 mistakes (**15%**)

   □ 6-9 mistakes (**5%**)

   □ More than 9 mistakes (**15%**)

2) How much time you spent for carrying out the task?

   □ 6-7 minutes (**0%**)

   □ 7-10 minutes (**20%**)

   □ 10-15 minutes (**25%**)

   □ 15-20 minutes (**45%**)

   □ More than 20 minutes (**10%**)

3) Have you ever used data transforming systems like Facetize?

   □ Yes, I have used systems like Facetize. (**25%**)

   □ No, I have not used any system like Facetize. (**75%**)

4) What was the final outcome of the scenario?

   □ I successfully completed the entire scenario, and the data are displayed exactly as it was requested. (**70%**)

   □ I did the whole script, except for some requirements that i did not perform successfully and did not receive all the correct results from Hippalus. List the numbers (in the order in which they appear) of the requirements you did not successfully execute. (**20%**)

□ I quitted the task in the requirement with number...... (fill in the number of requirement you were trying to perform or whatever you did). (**10%**)

5) How would you rate `Facetize` as a data transformation system?

   □ Very Useful (**40%**)

   □ Useful (**60%**)

   □ Little Useful (**0%**)

   □ Not Useful (**0%**)

6) Your feedback is important. Please use the textbox below for reporting problems that you encountered, other comments, or suggestions for future improvements (e.g. usability problems, etc). *(textbox)*

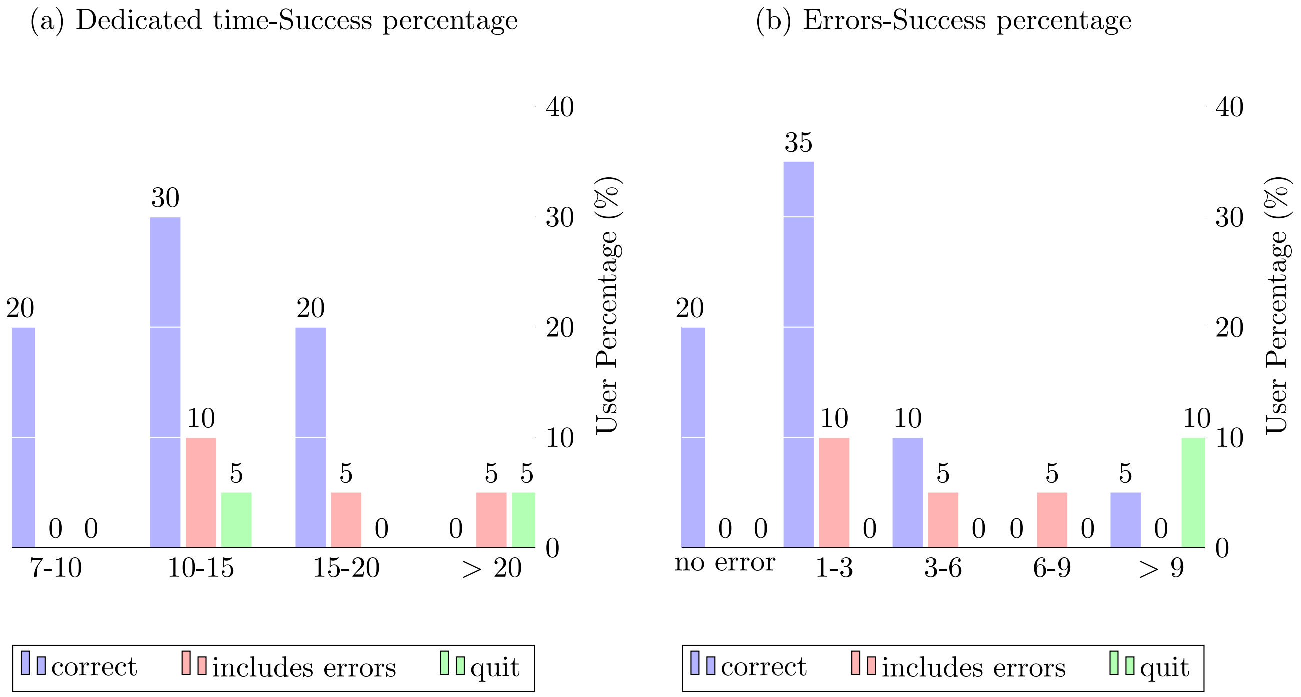
7) Question for participant's sex and age.

8.3.2 Results of the Evaluation

As regards the final outcome of the scenario, the results are satisfactory, since 14 (70%) users managed to complete the task successfully, 4 (20%) executed the scenario, except from some requirements that they probably did not perform successfully, and only 2 (10%) quitted the task. For those users who did not get the right results, or quitted the process, they did not understand the system and used it wrongly. As regards the overall rating, 12 (60%) users rated the system *Useful*, while (40%) *Very Useful*. Hence, all participants (100%) were positive.

Plots (a)-(d) of Figure 8.1 and Figure 8.2 provide additional information for further analyzing the results. In particular, from plot (a) Dedicated time-Success percentage, we can see that users that spent 10-15 minutes were the most successful. Those who spent more than 20 minutes were not very successful; probably they followed a wrong path or they had difficulties with the system. In contrast, those who spend 7-10 and 15-20 minutes had 20% success. From plot (b) Errors-Success percentage, we can see that the users who made 1-3 errors were the most successful; 35% success. Moreover, those who made no error had 20% success, 3-6 errors had 10% success and those who made more than 9 errors had 5% success. From plot (c) Categories-Success percentage, we can see that postgraduate students had 40% success, undergraduate students had 25% success and ICS employees had 5% success. From plot (d) Age-Success percentage, we can see that users from 25 to 34 years old achieved the highest success percentage (65%).

Users stated useful suggestions and comments for improving the usability of our system in the future. For example, many stated that had difficulties to define an expression for create a new facet and it would be useful to exist autocomplete when the user type a name of a facet or a function. Moreover, they pointed out that it would be useful to exist the ability to select many values at the same time, for add them in the same hierarchy when create one. Other suggestions are the


(a) Dedicated time-Success percentage

(b) Errors-Success percentage

Figure 8.1: (a) Dedicated time-Success percentage, and (b) Errors-Success percentage

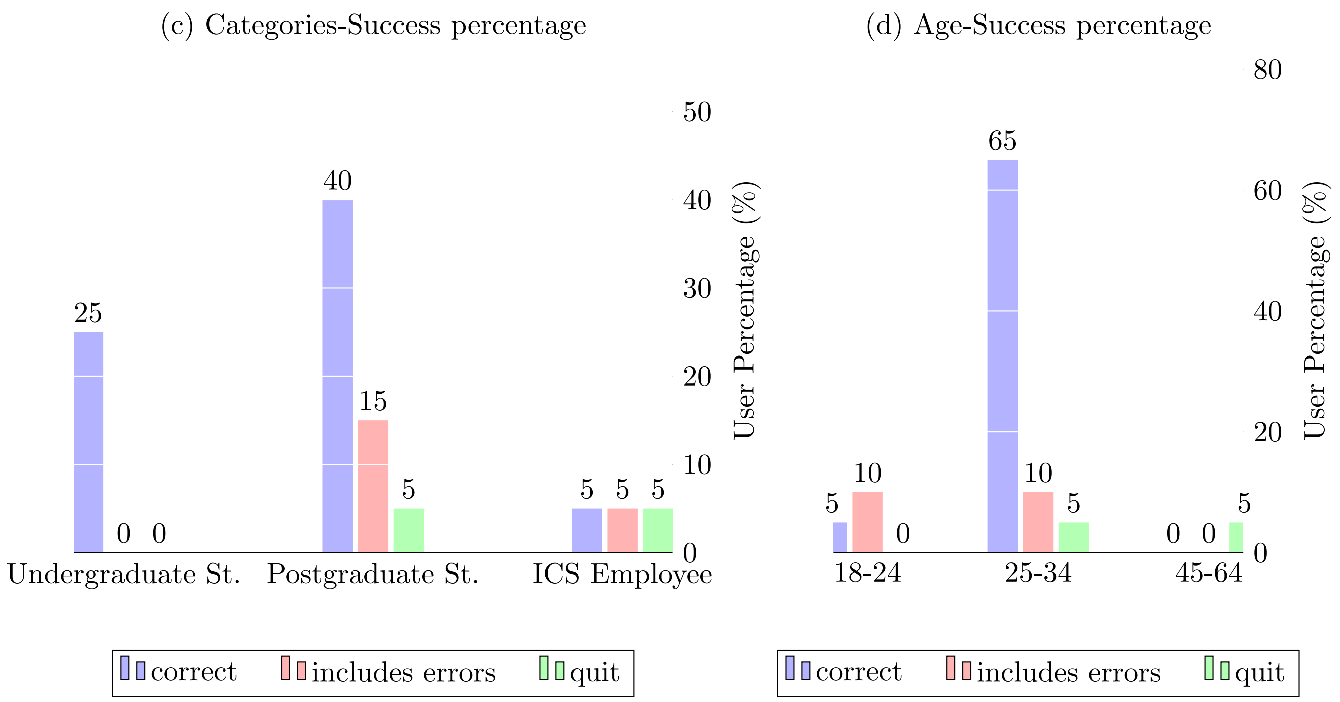
files that are being exported from our system, to be uploaded automatically in the Hippalus for exploration, with a button click via `Facetize` and to exist the ability to edit the values of an existing facet with an expression definition, using functions, as those that are used for create a new facet. Also, they stated to appear the number of occurrences of each value in a facet.

8.4 Possible Improvements

Here, we discuss issues related to the improvements and extensions of `Facetize` in the future.

8.4.1 Null Values in the Dataset

In case a user wants to investigate the rows for which a certain column is empty, one possible solution would be to exist a dropdown menu for each column. Then, when a user choose an option from that menu to get a facet pane with two choices: *false* and *true*. Choice *true* will contain the number of rows in that facet that are blank. By left clicking on that choice, it will select rows with empty cells on that facet. Respectively, the option *false* will filter rows that do not contain empty cells on that facet. Moreover, if the user wants to search for rows with blank cells (not



(c) Categories-Success percentage

(d) Age-Success percentage

Figure 8.2: (c) Categories-Success percentage and (d) Age-Success percentage

in a specific column), it could exist another dropdown menu that filter rows that contain blank cells.

8.4.2 Applicability over Very Large Datasets

Here, we discuss issues in order to improve the efficiency of `Facetize` in the future. In case exist a dataset with large amount of rows, it could scan the dataset file one time to select the facets and their distinct values. Then, the user could apply the desired transformations in the dataset (except from edits of individual rows) and after that to scan one more time the dataset file and to create the output file. Using that approach, our system will not keep tuples of dataset's rows in the main memory and will not create the table with the dataset's rows.

8.4.3 Format Support

`Facetize` could support other data file formats to export the transformed data. For example, it could export the data in CSV, TSV, JSON, XML, HTML. Also, the users could add their own export template in case they want to export the data in a specific format for loading from other systems for exploration.

8.4.4 Enrich a Facet with Restrictions

In case the users want to create a new facet, it could exist the ability to enrich a facet with restrictions regarding their values. For example, a facet with name "age" must contain positive integer numbers. In this way, users will create facets that contain valid values.

8.4.5 Integrity Constraints

Another extension is to enrich a project with integrity constraints. For instance, the valid range of values of each facet could be specified through the editor. Such constraints concern both numerically valued facets (e.g. person age) as well as categorical ones (e.g. location names). Having specified such constraints, the system could check whether the values in the dataset respect these constraints and then it could inform the user accordingly. This is especially important for projects whose dataset evolves over time: whenever the user refreshes the dataset of a project, the system could check whether these constraints are satisfied or are not. This extension requires an extension of the notion of project, specifically each facet should be associated with constraints that the user is able to formulate through the `Facetize` editor.

Chapter 9

Conclusion and Future Work

To conclude, there is a demand for systems that help users with no particular technical background to clean and apply transformations on plain datasets for turning them easily explorable by end users. In this thesis we investigate the hypothesis that the familiar interaction paradigm faceted search systems can be useful for such a task. However, a straightforward loading of such datasets in a faceted search system will not always result to a satisfying solution, since additional tasks are usually required. This includes deciding (a) the parts of the dataset that should be explorable, (b) the attributes that should visible and their order, (c) the transformations and/or enrichments that should be done, (d) the groupings (hierarchical or not) of the values that should be made, (e) the addition of derived attributes, and others. For this reason in this thesis we presented the design and implementation of an editor, called `Facetize`, that allows the user to carry out these tasks in a user-friendly and guided manner,without presupposing any technical background about the data representation language or the query language.

`Facetize` can be construed as an authoring environment for setting up exploration services over static files (represented in various formats) or results produced by querying SPARQL endpoints. It is worth noting that the authoring environment, supports the notion of project, allowing the user to gradually configure the desired presentation and also to update (refresh) the underlying dataset without losing the transformations that they have been defined.

We evaluated the `Facetize` prototype with users over real data. The results of the evaluation with users were positive: all participants (100%) were positive, as regards the overall rating. Moreover, most of the users (90%) managed to complete the task, while 70% of them managed to complete the task successfully. In addition, this evaluation allowed us to identify useful extensions for improving the usability of our system in the future.

`Facetize` offers remarkable innovations in the area of data transforming. As we mentioned in this thesis, `Facetize` supports the creation of hierarchies and numerical intervals and it can fetch data directly from SPARQL endpoints, making it appropriate for dynamic datasets. In addition, users have the ability to reapply the same range of transformations to an existing project, which is very convenient in case they want to refresh the dataset of a project and would like to apply the same transformations.

As regards the suggestions of users of the evaluation for system improvements, they pointed out to incorporate autocomplete in names of functions and facets when define an expression for create a new facet. Moreover, to exist the ability to select many values at the same time for add them in the same hierarchy when create a new one and the files that are being exported from `Facetize` to be uploaded automatically in Hippalus for exploration. Also, users to be able to edit the values of a facet using expression definition and to appear the number of occurrences of each value in a facet. Directions that are worth further work and research include methods for predicting the missing values. Such functionality is helpful in cases where there are a lot of missing values that we would like to fill in the dataset at hand, and various models could be employed for that purpose (e.g. dependence tree, Naive Bayes model, etc). Another direction is to device methods that can spot possible errors or misspellings by exploiting string similarity-based clustering methods.

Chapter 10

Appendix

10.1 Logging of Transformations

The transformations of each project are saved and can be reapplied to a new version of the dataset. Such transformations are the following: (a) set facet type, (b) set order of facets, (c) create hierarchy, (d) create intervals (linear, log, user defined), (e) specify geographic information for facets, (f) set visibility of facets.

Each transformation is saved on disc in JSON format. For example, for the transformations of our running example, the log file keeps the following information:

```
{
"facetName": "Longitude",
"geoInfo": "Longitude",
"transformationType": "geoInfo"
}, {
"facetName": "Latitude",
"geoInfo": "Latitude",
"transformationType": "geoInfo"
}, {
"facetName": "Rooms",
"visibility": "hidden",
"transformationType": "facet-visibility"
}, {
"facetName": "Price",
"transformationType": "facet-order",
"order": 4
}, {
"facetName": "Price",
"transformationType": "facet-order",
"order": 3
}, {
```

```
"facetName": "Price",
"transformationType": "facet-order",
"order": 2
}, {
"facetName": "Price",
"transformationType": "facet-order",
"order": 1
}, {
"facetName": "Location",
"transformationType": "facet-order",
"order": 1
}, {
"facetName": "Location",
"transformationType": "facet-order",
"order": 0
}, {
"facetName": "Name",
"transformationType": "facet-order",
"order": 3
}, {
"facetName": "Name",
"transformationType": "facet-order",
"order": 2
}, {
"facetName": "Name",
"transformationType": "facet-order",
"order": 1
}, {
"facetName": "Price",
"transformationType": "facet-order",
"order": 2
}, {
"facetName": "Price",
"transformationType": "facet-order",
"order": 1
}, {
"facetName": "Rating",
"transformationType": "facet-order",
"order": 5
}, {
"facetName": "Rating",
"transformationType": "facet-order",
"order": 4
}, {
```

```
"facetName": "Rating",
"transformationType": "facet-order",
"order": 3
}, {
"facetName": "Rating",
"transformationType": "facet-order",
"order": 2
}, {
"facetName": "Rating",
"transformationType": "facet-order",
"order": 3
}, {
"hierarchy": "Chania\/Crete\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Heraklion\/Crete\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Lasithi\/Crete\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Rethymno\/Crete\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Chania\/Crete\/Islands\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Heraklion\/Crete\/Islands\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Lasithi\/Crete\/Islands\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Rethymno\/Crete\/Islands\/Location",
"transformationType": "hierarchy"
}, {
"hierarchy": "Rhodes\/Islands\/Location",
"transformationType": "hierarchy"
}, {
"facetName": "Rooms",
"visibility": "visible",
"transformationType": "facet-visibility"
}, {
"newType": "id",
"facetName": "Name",
"transformationType": "facet-type"
}, {
"hierarchies": ["106\/106-115\/Price", "115\/106-115\/Price"],
"transformationType": "hierarchies"
}, {
"hierarchies": ["156\/156-264\/Price", "157\/156-264\/Price",
"264\/156-264\/Price"], "transformationType": "hierarchies"
}, {
"hierarchies": ["22\/22-69\/Price", "50\/22-69\/Price",
"59\/22-69\/Price", "69\/22-69\/Price"],
"transformationType": "hierarchies"
}
```

Bibliography

[1] Ziawasch Abedjan, Cuneyt G Akcora, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker. Temporal rules discovery for web data cleaning. *Proceedings of the VLDB Endowment*, 9(4):336–347, 2015.

[2] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. Detecting data errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment*, 9(12):993–1004, 2016.

[3] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: a survey. *The VLDB Journal—The International Journal on Very Large Data Bases*, 24(4):557–581, 2015.

[4] Ziawasch Abedjan, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker. Dataxformer: A robust transformation discovery system. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*, pages 1134–1145. IEEE, 2016.

[5] Abhinay B Angadi, Akshata B Angadi, and Karuna C Gull. International journal of advanced research in computer science and software engineering. *International Journal*, 3(6), 2013.

[6] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[7] Xu Chu, Ihab F Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 458–469. IEEE, 2013.

[8] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1247–1261. ACM, 2015.

[9] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. Nadeef: a commodity data cleaning

system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 541–552. ACM, 2013.

[10] Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[11] Tamraparni Dasu and Ji Meng Loh. Statistical distortion: Consequences of data cleaning. *Proceedings of the VLDB Endowment*, 5(11):1674–1683, 2012.

[12] Pavlos Fafalios and Yannis Tzitzikas. Sparql-ld: a sparql extension for fetching and querying linked data. In *International Semantic Web Conference (Posters & Demos)*, 2015.

[13] Pavlos Fafalios, Thanos Yannakis, and Yannis Tzitzikas. Querying the web of data with sparql-ld. In *International Conference on Theory and Practice of Digital Libraries*, pages 175–187. Springer, 2016.

[14] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Wenyuan Yu. Towards certain fixes with editing rules and master data. *The VLDB Journal*, 21(2):213–238, 2012.

[15] Laura Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, 2003.

[16] Nicholas V Findler and Jan Van Leeuwen. A family of similarity measures between two strings. *IEEE transactions on pattern analysis and machine intelligence*, (1):116–118, 1979.

[17] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. Mapping and cleaning. In *IEEE 30th International Conference on Data Engineering (ICDE), Chicago, Ill., March 31-April 4, 2014/Cruz, Isabel F.[edit.]; et al.*, pages 232–243, 2014.

[18] Shubham Gupta, Pedro Szekely, Craig A Knoblock, Aman Goel, Mohsen Taheriyan, and Maria Muslea. Karma: A system for mapping structured sources into the semantic web. In *Extended Semantic Web Conference*, pages 430–434. Springer, 2012.

[19] Hamed Ibrahim Housien, Zhang Zuping, and Zainab Qays Abdulhadi. A comparison study of data scrubbing algorithms and frameworks in data warehousing. *International Journal of Computer Applications*, 68(25), 2013.

[20] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[21] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Fundamentals of data warehouses*. Springer Science & Business Media, 2013.

[22] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[23] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011.

[24] Zuhair Khayyat, Ihab F Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. Bigdansing: A system for big data cleansing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1215–1230. ACM, 2015.

[25] Laks VS Lakshmanan, Fereidoon Sadri, and Iyer N Subramanian. Schemasql-a language for interoperability in relational multi-database systems. In *VLDB*, volume 96, pages 239–250. Citeseer, 1996.

[26] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[27] Panagiotis Lionakis and Yannis Tzitzikas. Pfsgeo: Preference-enriched faceted search for geographical data. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 125–143. Springer, 2017.

[28] Jonathan I Maletic and Andrian Marcus. Data cleansing. In *Data Mining and Knowledge Discovery Handbook*, pages 21–36. Springer, 2005.

[29] Michalis Mountantonakis and Yannis Tzitzikas. High performance methods for linked open data connectivity analytics. *Information (2078-2489)*, 9(6), 2018.

[30] Heiko Müller and Johann-Christph Freytag. *Problems, methods, and challenges in comprehensive data cleansing.* Professoren des Inst. Für Informatik, 2005.

[31] Andriy Nikolov, Victoria Uren, Enrico Motta, and Anne De Roeck. Integration of semantically annotated data by the knofuss architecture. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 265–274. Springer, 2008.

[32] Panagiotis Papadakos and Yannis Tzitzikas. Hippalus: Preference-enriched faceted exploration. In *EDBT/ICDT Workshops*, volume 172, 2014.

[33] Panagiotis Papadakos and Yannis Tzitzikas. Comparing the effectiveness of intentional preferences versus preferences over specific choices: a user study. *International Journal of Information and Decision Sciences*, 8(4):378–403, 2016.

[34] Alexandros Papangelis, Panagiotis Papadakos, Margarita Kotti, Yannis Stylianou, Yannis Tzitzikas, and Dimitris Plexousakis. Ld-sds: Towards an expressive spoken dialogue system based on linked-data. In *Search Oriented Conversational AI, SCAI 17 Workshop (co-located with ICTIR 17)*, 2017.

[35] Nataliya Prokoshyna, Jaroslaw Szlichta, Fei Chiang, Renée J Miller, and Divesh Srivastava. Combining quantitative and logical data cleaning. *Proceedings of the VLDB Endowment*, 9(4):300–311, 2015.

[36] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[37] Vijayshankar Raman and Joseph M Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, volume 1, pages 381–390, 2001.

[38] Giovanni Maria Sacco and Yannis Tzitzikas. *Dynamic taxonomies and faceted search: theory, practice, and experience*, volume 25. Springer Science & Business Media, 2009.

[39] Kofi Adu-Manu Sarpong and John Kingsley Arthur. A review of data cleansing concepts-achievable goals and limitations. *International Journal of Computer Applications*, 76(7), 2013.

[40] Michael Stonebraker, Daniel Bruckner, Ihab F Ilyas, George Beskales, Mitch Cherniack, Stanley B Zdonik, Alexander Pagan, and Shan Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.

[41] Daniel Tunkelang. Faceted search. *Synthesis lectures on information concepts, retrieval, and services*, 1(1):1–80, 2009.

[42] Y. Tzitzikas and E. Dimitrakis. Preference-enriched faceted search for voting aid applications. *IEEE Transactions on Emerging Topics in Computing*, PP(99):1–1, 2016.

[43] Yannis Tzitzikas, Nicolas Bailly, Panagiotis Papadakos, Nikos Minadakis, and George Nikitakis. Using preference-enriched faceted search for species identification. *International Journal of Metadata, Semantics and Ontologies*, 11(3):165–179, 2016.

[44] Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. Faceted exploration of RDF/S datasets: a survey. *Journal of Intelligent Information Systems*, 2016.

[45] Yannis Tzitzikas and Panagiotis Papadakos. Interactive exploration of multidimensional and hierarchical information spaces with real-time preference elicitation. *Fundamenta Informaticae*, 20:1–42, 2012.

[46] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. Mediators over taxonomy-based information sources. *The VLDB Journal*, 14(1):112–136, 2005.

[47] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. S ee db: efficient data-driven visualization recommendations to support visual analytics. *Proceedings of the VLDB Endowment*, 8(13):2182–2193, 2015.

[48] Kwok-Bun Yue. A realistic data cleansing and preparation project. *Journal of Information Systems Education*, 23(2), 2012.