GeoStream - Exploiting User-Generated Geospatial Content Streams

FP7-SME-2012-315631

Online Demo
D7.4

Deliverable lead contractor: MMV

Main contributors:

George Lamprianidis glampr@imis.athena-innovation.gr
George Papatheodorou papatheodorou@imis.athena-innovation.gr
Dimitris Skoutas dskoutas@imis.athena-innovation.gr
Shoaib Burq shoaib.burq@gmail.com
Dirk Morgenroth dm@mmm24.de

Due data: 31.03.2014

Abstract

This deliverable describes the features and the user interface of the online demo that has been developed so far in the project, including the UGCS content store and data mining processes.

Research Center "ATHENA," Greece
FU BERLIN, Germany
Fraunhofer, Germany
Michael Mueller Verlag, Germany
TALENT, Greece
WIGEOGIS, Austria

Table of Contents

Table of Figures

1 Introduction

This document describes the features and the user interface of the GEOSTREAM online demo, a Web application that showcases the geospatial content collection and processing methods. These methods, i.e. the Web sources used for data collection, and the data storage, retrieval, integration, and analysis, have been developed in WPs 1 and 2 and are documented in detail in the deliverables D1.2 and D2.1.

More specifically, the main features of the online demo include:

- specification of areas for data collection (described in Section 2);
- category mappings from the sources to a common schema (described in Section 3);
- entity matching across the various sources (described in Section 4);
- mining of regions of interest (described in Section 5);
- content search and faceted browsing (described in Section 6).

Finally, Section 7 provides information on implementation details, and Section 8 concludes the deliverable with a listing of the next steps.

2 Areas for Data Collection

The starting point for collecting and analysing crowdsourced geospatial content is the page listing the areas for data collection. This page can be accessed at the following URL:

http://dataminer.geocontentstream.eu/admin/areas

A screenshot is illustrated in Figure 1.

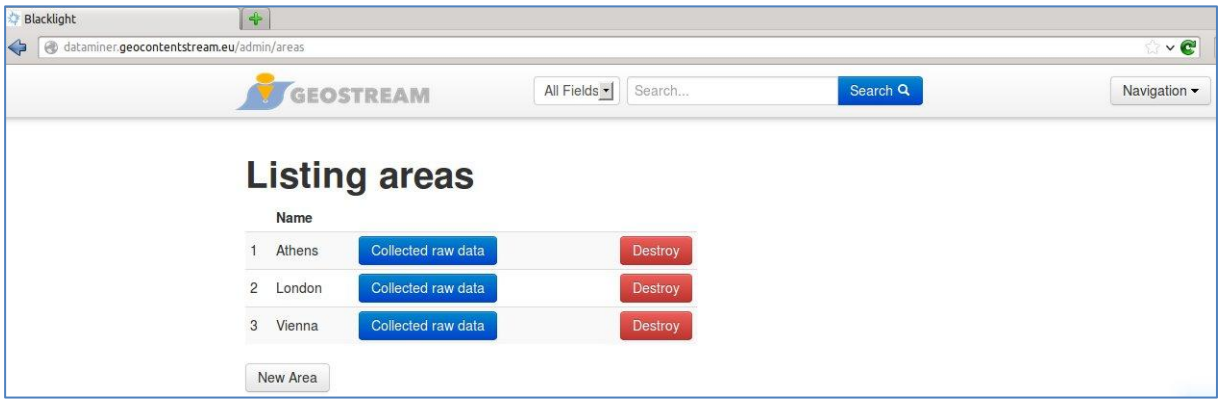

Figure 1: Areas for data collection.

This page lists the areas already specified for data collection (currently, Athens, London, and Vienna), and provides options for defining a new area, viewing an overview of the data collected for an area, as well as deleting the data for a specific area.

Specifying a new area for data collection is done by clicking on "New Area" and filling in the form shown in Figure 2.

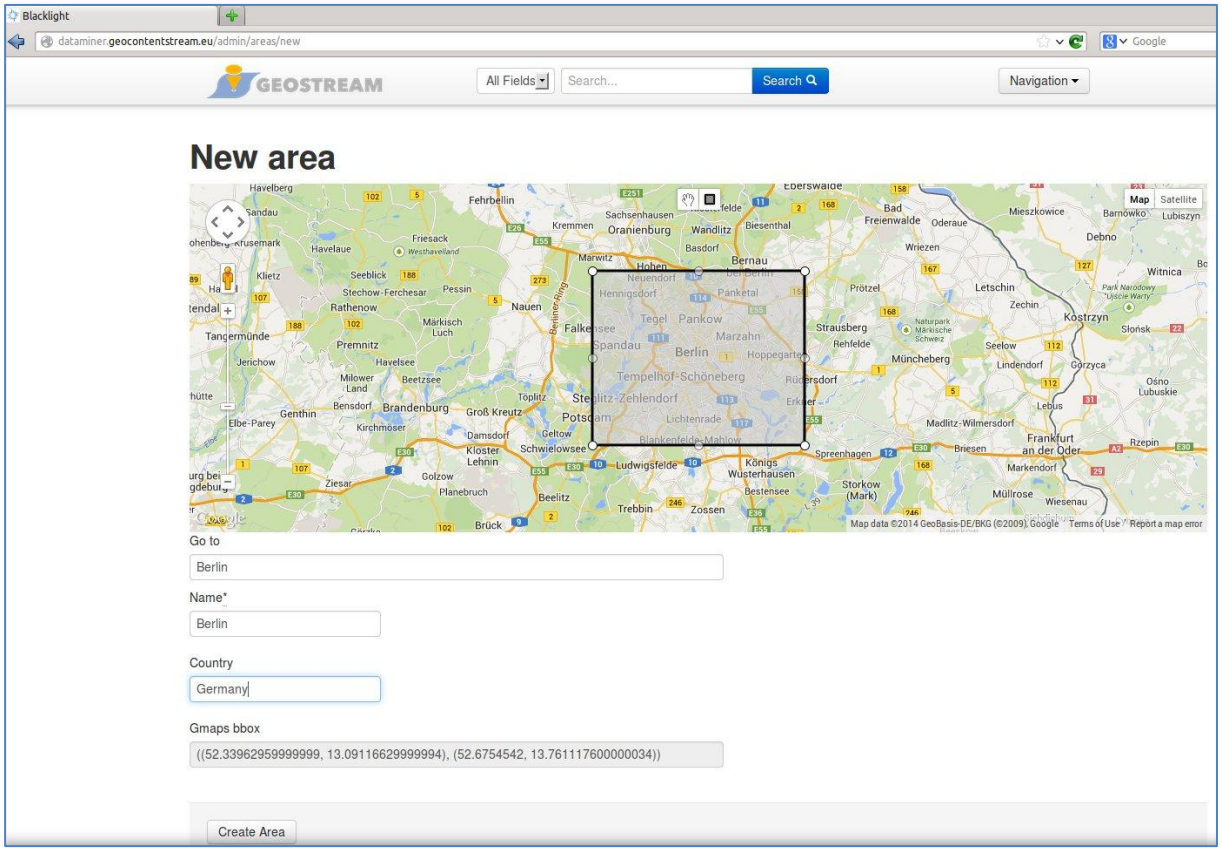

Figure 2: Creation of new area.

In particular, the user needs to provide a name for the new area, and can draw a rectangle on the map to specify the desired area. Clicking on "Create Area" stores this area (i.e., the coordinates of the bounding box) in the content store.

To get an overview of the data collection process for an area, the user clicks on "Collected raw data" on the page shown in Figure 1. This leads to a page illustrated in Figure 3, where the user can view counts of collected data on the map, as well as status information for each source, e.g. whether the collection process is currently running or not, and how many data have been collected. The sources used for data collection, and the details on how the data are stored, are provided in Deliverable D2.1.

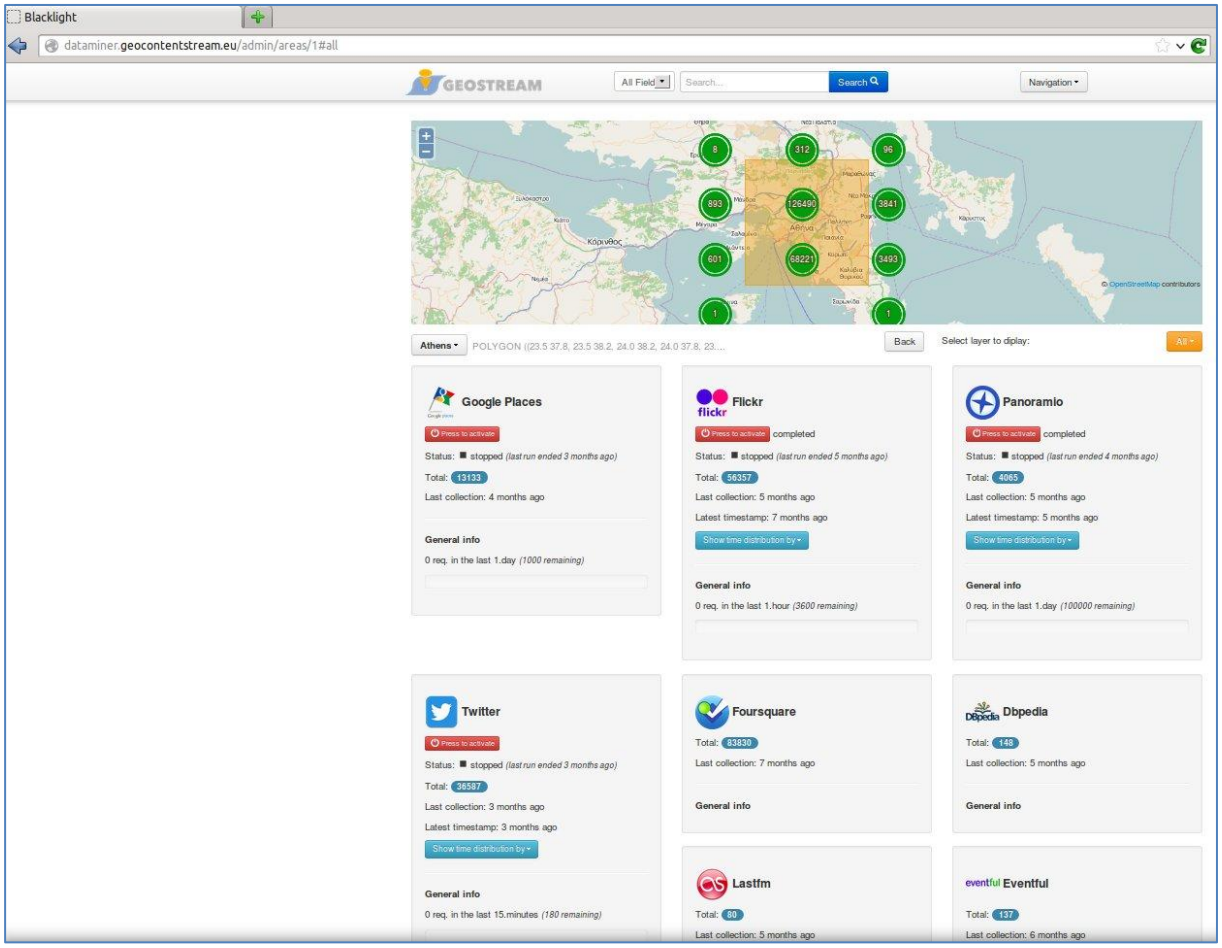

Figure 3: Overview of the data collection process.

3 Category Mappings

Once data for a specific area are collected, the first step towards integrating them is to reconcile the different classification schemes used by each source to a common category hierarchy. The details of this process are described in Deliverable D1.2.

The relevant section of the Web application comprises three views. The first can be accessed by selecting the option "Category Mappings" in the navigation menu and is illustrated in Figure 4. This view concerns the validation of the automatically computed mappings. That is, a list of mappings is displayed, showing the original category, the source of origin, and the category to which it has been mapped in the common schema. For each such mapping, a score computed by the matcher is given, indicating some degree of confidence for the match, and three options are provided: to accept, reject or modify the mapping, by clicking the corresponding buttons or using the drop down list to assign a different category. At the top of the list, there are also options to browse the mappings by the level of confidence or the status (accepted, rejected or pending).

The second view can be accessed by selecting the option "Match Statistics" in the drop down list at the top of the page. This view is shown in Figure 5, and refers to statistics on the results of the automatic matching. In particular, for each data source, it plots the number of identified mappings, indicating also their distribution w.r.t. the level of confidence.

Finally, a third view is accessed via the option "Category Distribution" and is illustrated in Figure 6. This comprises a list of pie charts that plot, for each source, the distribution of collected POIs for each top level category in the common category hierarchy.

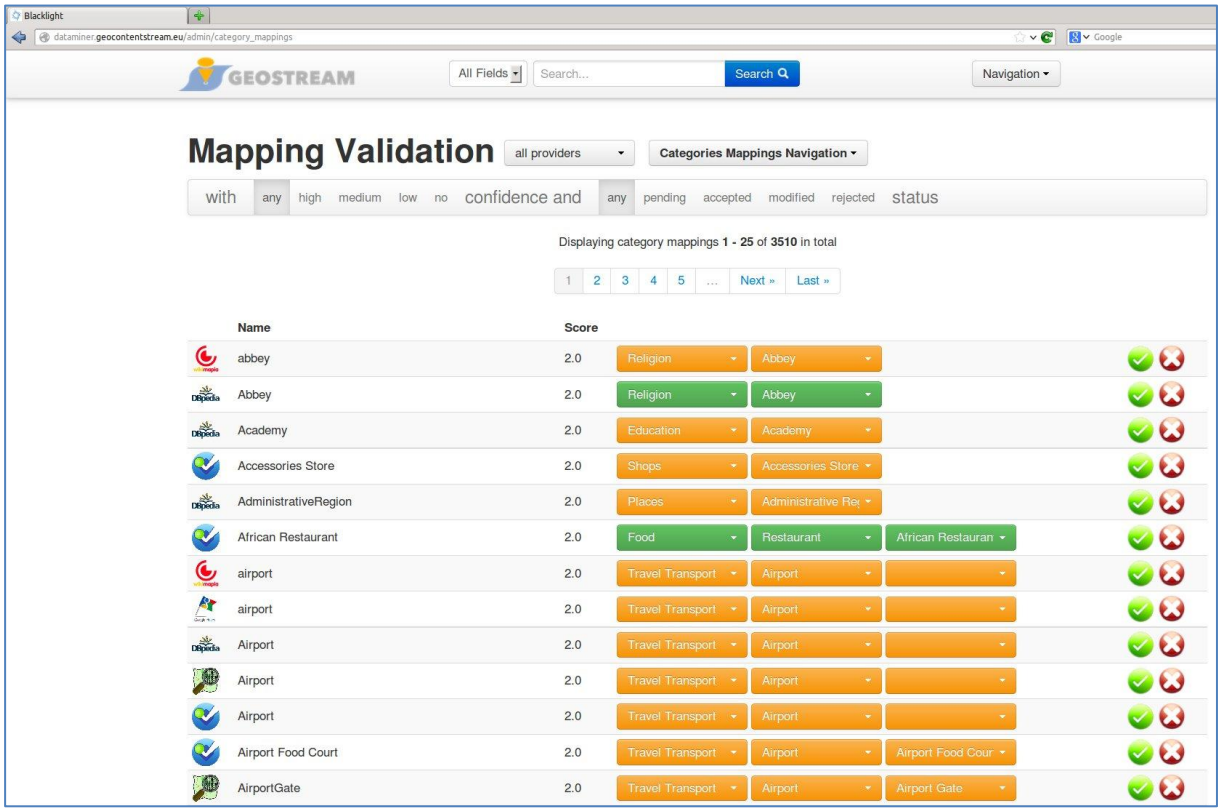

Figure 4: Validation of category mappings.

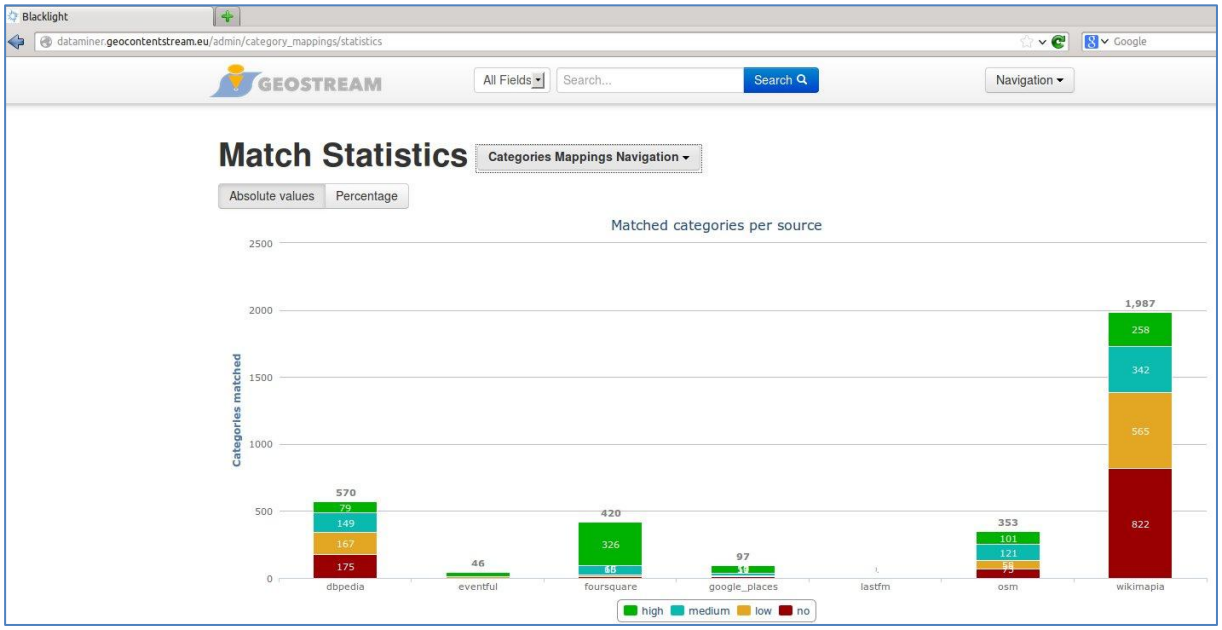

Figure 5: Category mapping statistics.

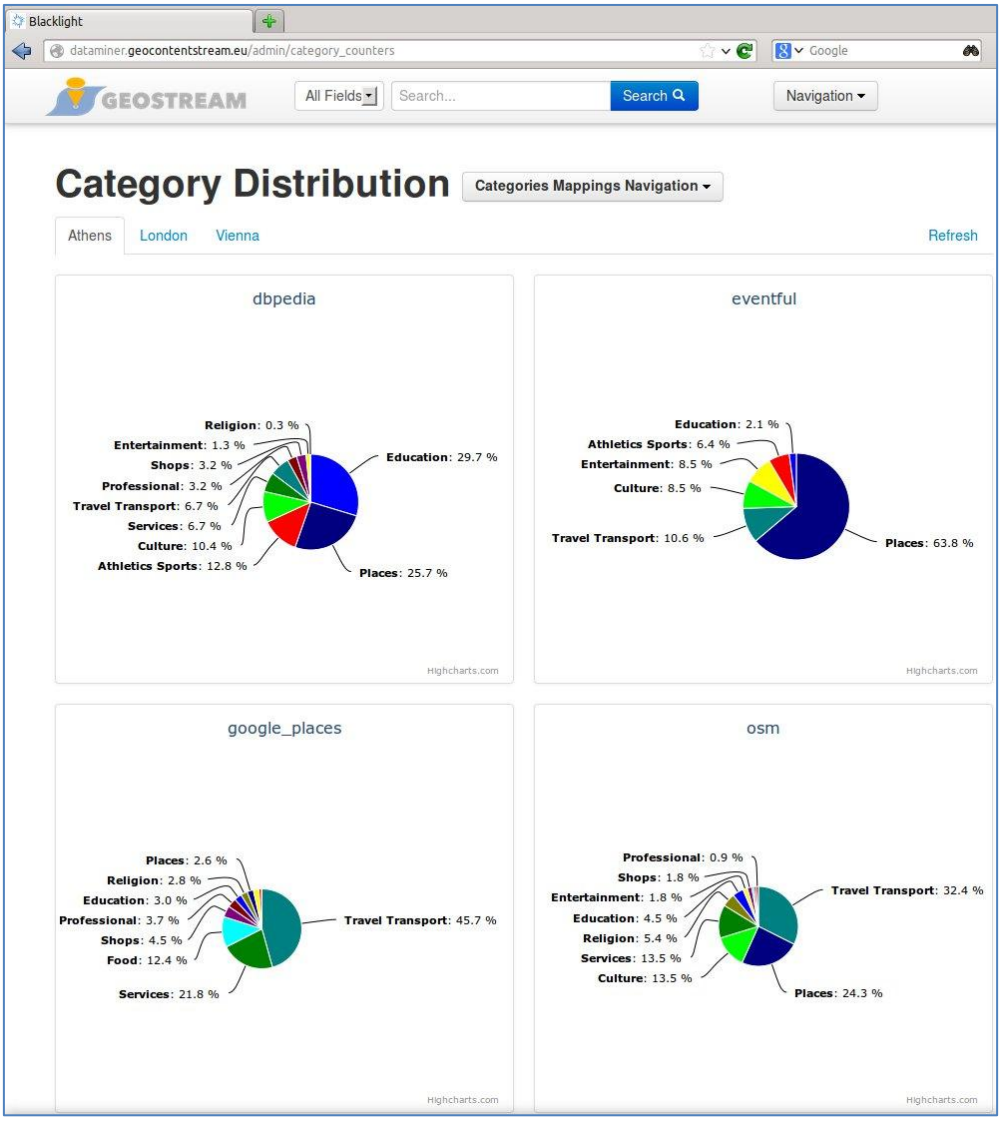

Figure 6: Data distribution per category.

4 Matched Entities

The second step of the data integration process is to identify duplicate entities across the various sources. This process is described in detail in Deliverable D1.2. The Web application provides two views related to this analysis. The first can be accessed through the navigation menu, selecting the option "Matched Entities", and is illustrated in Figure 7.

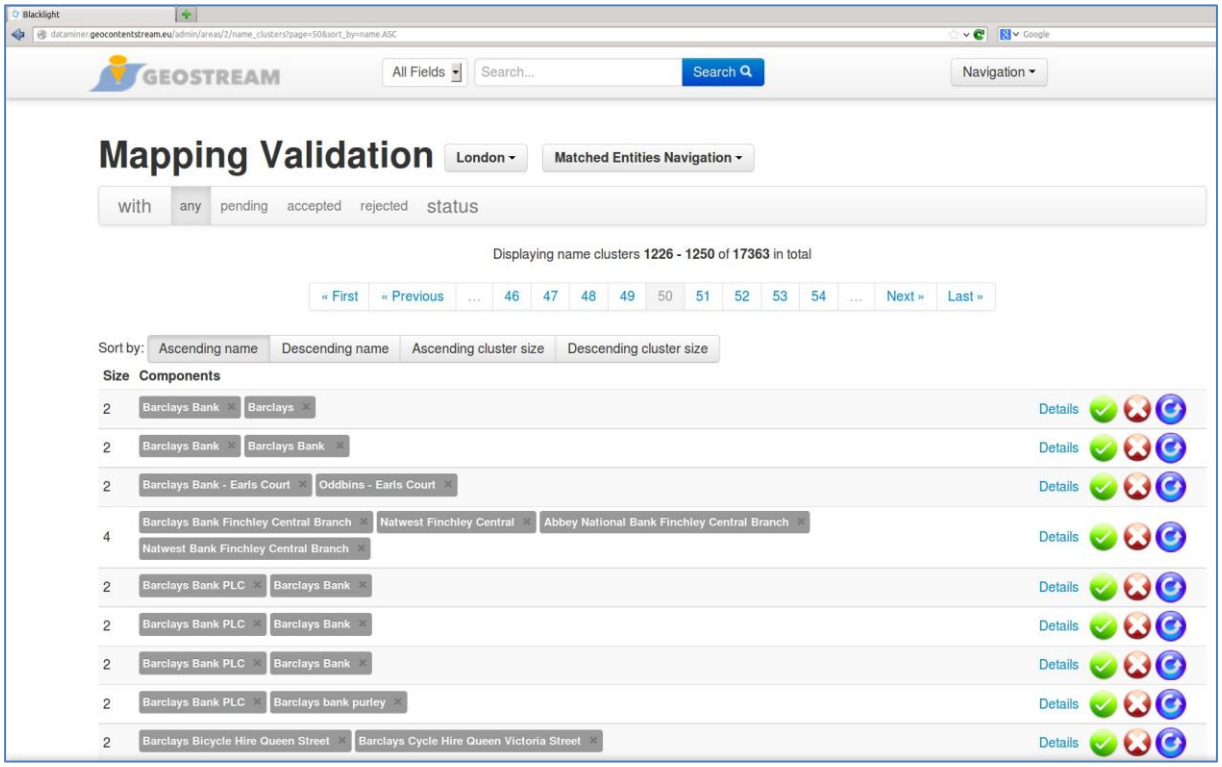

Figure 7: Validation of matched entities.

Furthermore, by clicking on "Details", a map is displayed showing the location of the matched entities (see Figure 8), to facilitate the decision on whether to accept or reject the match.

The second view can be accessed by selecting "Statistics" in the dropdown list at the top of the page. This view provides three plots (bar charts) showing: (a) the total number of POIs downloaded from each source; (b) the number of POIs, for each source, for which a match with POIs from other sources has been found (see Figure 9), and (c) the distribution of the number of matches between the various sources.

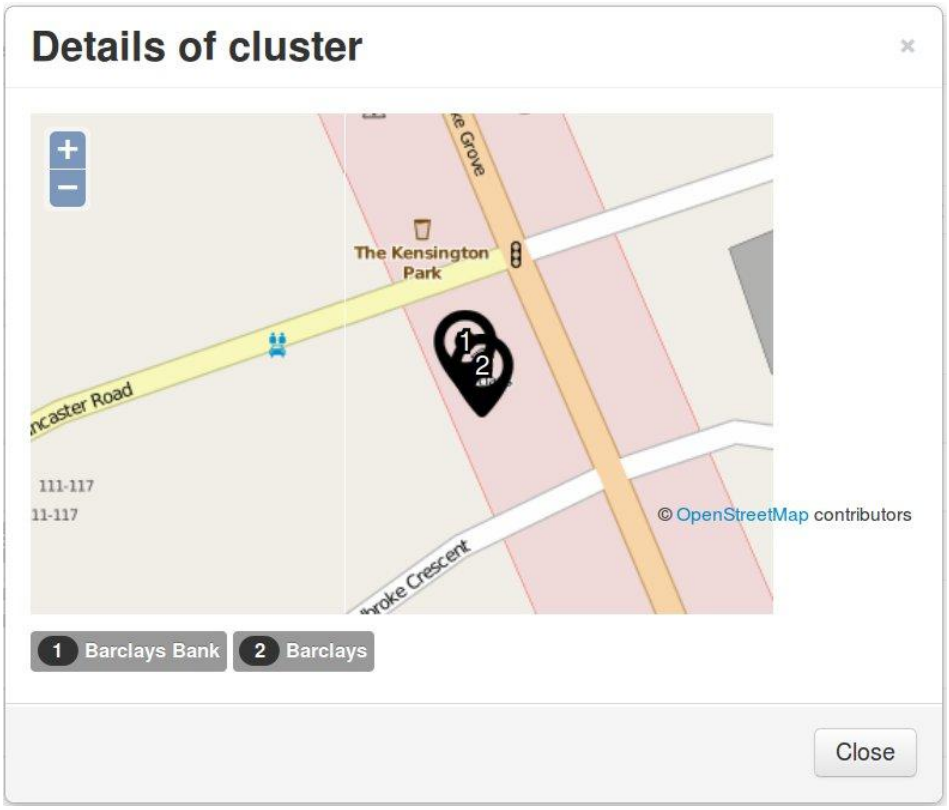

Figure 8: Inspect location of matched entities.

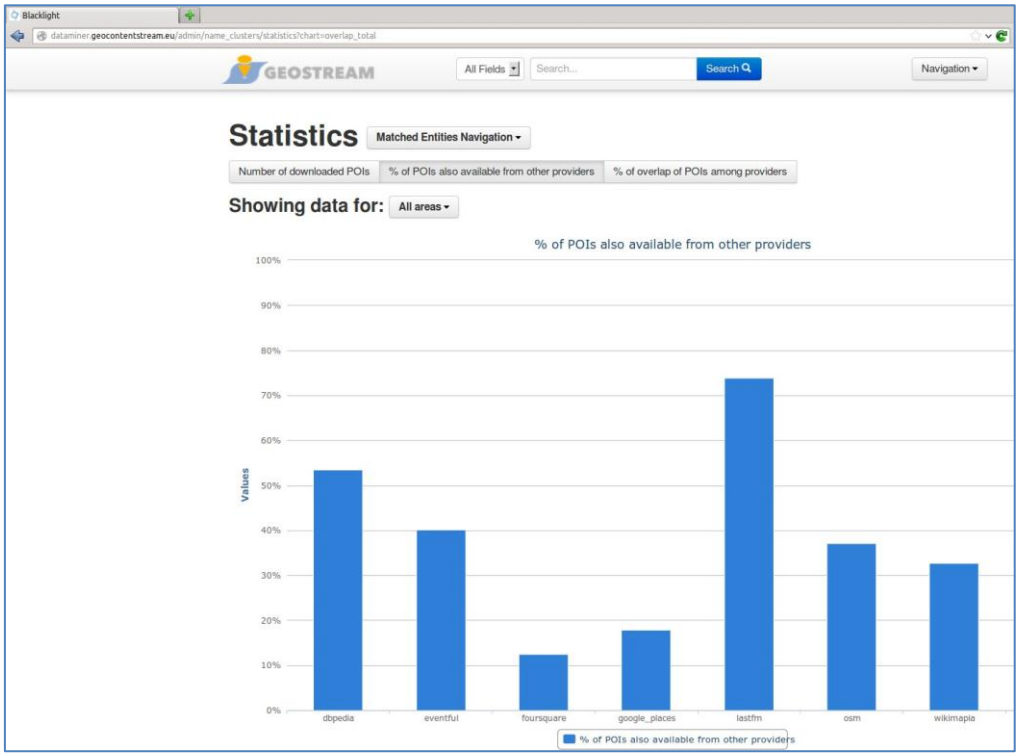

Figure 9: Percentage of POIs matched with other sources.

5 Regions of Interest

The final step of the data mining process is the spatial clustering of POIs per category to identify Regions of Interest, i.e. areas with high density of POIs of a specific category. The clustering algorithm is described in Deliverable D1.2. The results of this analysis can be inspected by selecting the option "Regions of Interest" from the navigation menu. An example is illustrated in Figure 10.

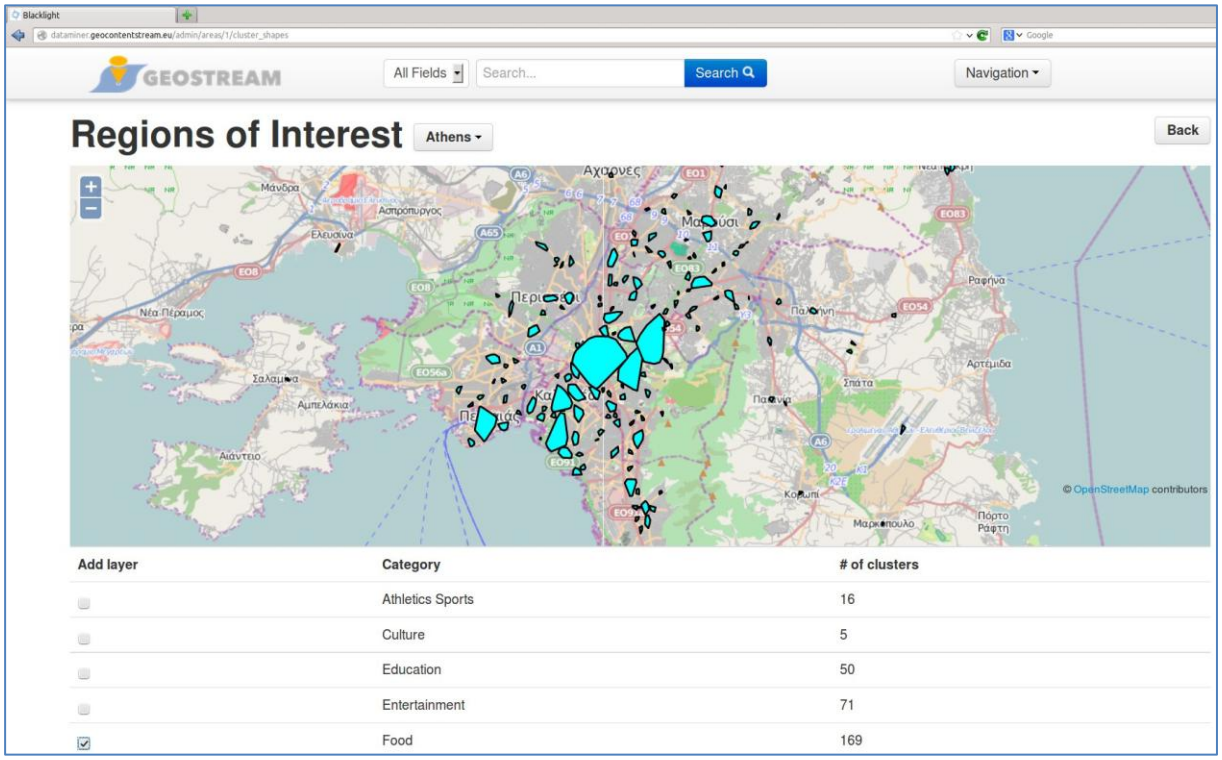

Figure 10: Regions of Interest per category.

6 Search and Browsing

In addition to inspecting the analysis results of the data collection, integration and mining process, as described in the previous sections, the Web application provides also search and browsing capabilities on the collected data. This is done by entering a search term in the text field provided at the top of the page and clicking the "Search" button. Figure 11 displays an example for the keyword "Wembley". To narrow down the search results, the user can filter the result list by selecting specific values for the facets displayed on the left of the page, i.e. to drill down to a specific source, category, etc. A map is also displayed, showing the locations of the search results.

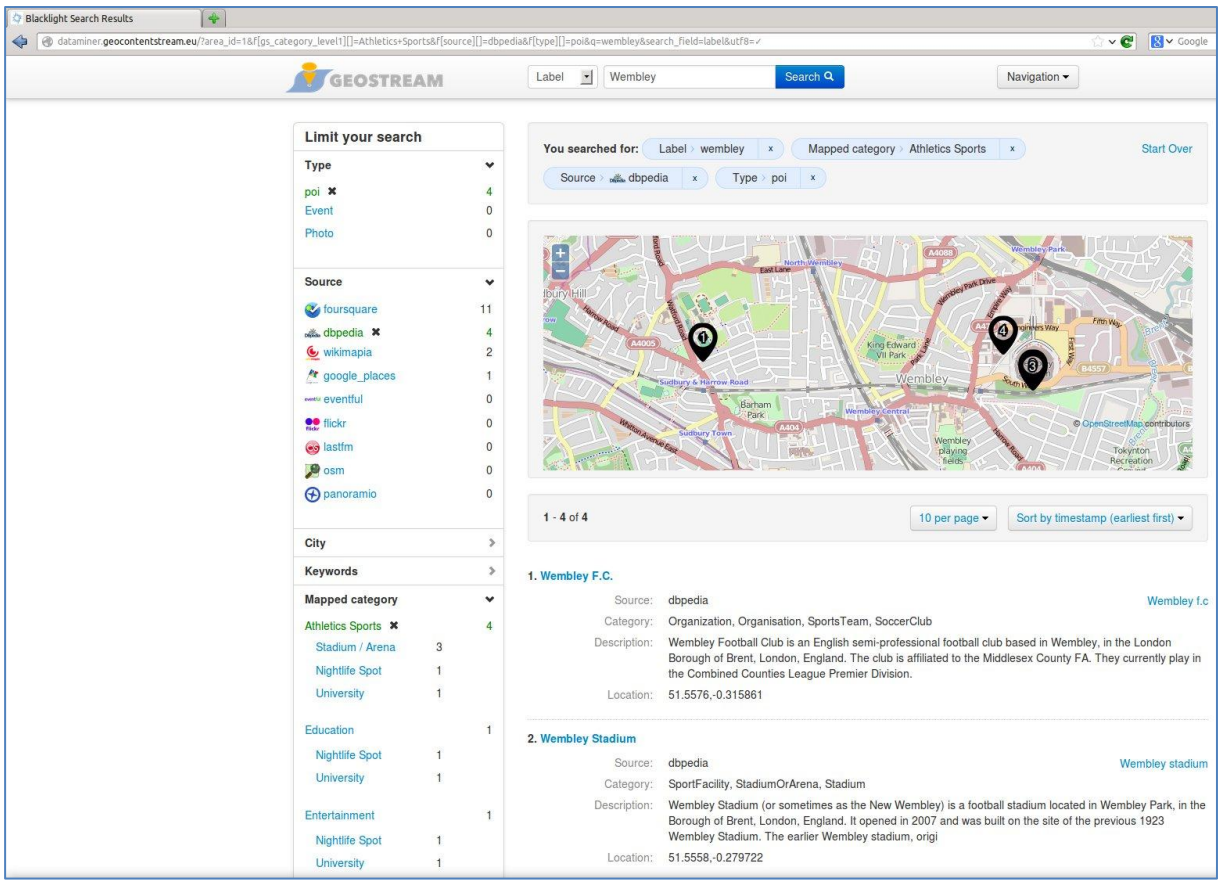

Figure 11: Search and browsing.

7 Implementation Details

The online demo is hosted at a Virtual Machine in ~okeanos (https://okeanos.grnet.gr/), a cloud service designed and developed by the Greek Research and Technology Network (GRNET) for the Greek Research and Academic Community.

The demo pages are served by a Web application written in the Ruby on Rails (version 4.0) framework. The data is stored in a PostgreSQL (version 9.3) database, with the PostGIS (version 2.1) extension to handle all spatial functionality.

The collected data have been downloaded from their respective providers by client applications written in Ruby (for Google Places, Flickr, Panoramio, Twitter) and Java (for FourSquare, DBpedia, LastFM, Eventful, Wikimapia, OSM).

A search form is integrated in the application's navigation bar providing full-text search on the entirety of the downloaded data. The search process is powered by Solr (version 4.4), an open source enterprise search platform which features powerful full-text search and faceted search.

On top of the collected raw data, a series of optimized clustering algorithms group the related POIs by similarity of name and/or by proximity, to present meaningful statistics about the providers. These algorithms are written in Java.

The results of the clustering are presented via charts rendered purely in JavaScript in the browser. Furthermore, the OpenLayers JavaScript (http://openlayers.org/) is extensively used to present spatial information, such as the location of POIs and the extent of the computed cluster in interactive web maps.

8 Next Steps

The current version of the online demo described in this deliverable corresponds to the work mainly done within the first period of the project. It is our goal to continue improving and enriching this demo throughout the duration of the project, adding more features as they become available (e.g. to include the authoring tools that will be developed in WP4). Also, the demo is currently focused on presenting in a visual and intuitive way the results of the data collection and mining process, which is mainly run separately in the background (e.g. by executing a jar file to compute matching entities). We plan to enhance the graphical interface to allow controlling the execution of these processes too. Finally, we are working on further testing and improving the various algorithms (e.g. for category mappings) to improve accuracy and performance, and also to make the system architecture more modular so that it is easier to maintain and extend individual modules independently to facilitate future usage and exploitation.