D5.5.1 – The Europeana Geoparser and Gazetteer

Documentation and final prototype

co-funded by the European Union

The project is co-funded by the European Union, through the ***e*Content*plus*** programme

http://ec.europa.eu/econtentplus

Österreichische Nationalbibliothek

EuropeanaConnect is coordinated by the Austrian National Library

ECP-2008-DILI-528001

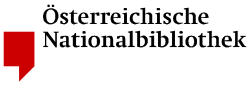
EuropeanaConnect

D5.5.1 – The Europeana Geoparser and Gazetteer

Deliverable number/name	*D 5.5.1*
Dissemination level	*PU*
Delivery date	*3/5/2011*
Status	*Final*
Author(s)	*Nuno Freire (BNP, IST) André Soares (IST)*

e*Content*plus

This project is funded under the *e*Content*plus* programme,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.



EuropeanaConnect is coordinated by the Austrian National Library

Executive Summary

Task 5.5 in EuropeanaConnect aimed to build data oriented services to support geographical information based user interfaces in the Europeana portal. Very few objects contain structured geographical metadata, however. This kind of information mainly exists in metadata records as unstructured natural language texts, which are difficult to use in information systems. This task pursued two objectives:

- To support the organization and maintenance of geographic data, by providing storage, data integration, and search services for geographical information.
- To enrich the object descriptions by analysing geographic and time references in existing metadata records.

This document describes the specification, design, implementation, evaluation and usage, of two services developed to achieve the above objectives: the Gazetteer and the Geoparser.

Table of Contents

1. Introduction

Aspects of the Europeana user interface, some developed in EuropeanaConnect task 3.3 – Spatio-temporal access channels for Europeana, make use of both time related and geographical metadata, providing a new visual access channel to the digital objects. However, very few objects contain explicit geographical metadata. Therefore, specific techniques must be employed to enrich the object descriptions by analysing geographic and time references in existing metadata records.

This task improved and adapted, for Europeana, the geographical services developed within the context of the DIGMAP and TELplus projects. Namely, two services have been the focus of this task:

- The Gazetteer: a service for supporting the organization and maintenance of geographic data, by providing storage, data integration, and search services;
- The Geoparser: a service that uses information extraction techniques to automatically identify names of places and time periods that are mentioned in unstructured text. It works together with the Gazetteer to be able to assign coordinates and dates with the mentioned names of places and periods.

The target users of these services are Europeana, aggregators and data providers, which need to enrich the object descriptions by analyzing geographic or temporal references in existing metadata records.

This document will follow with the description of both systems, covering their specification, design, implementation, evaluation and usage.

2. Gazetteer

Data representing the geographical aspects of our world can nowadays be stored and processed efficiently, and many of the decisions people make today depend on the details of our immediate surroundings, requiring information about specific places on the Earth's surface. This information, labeled *geographical information,* helps distinguish places and helps to make decisions pertaining to specific locations.

A Gazetteer consists of a list of geographic names, together with their geographic locations and other descriptive information. Mainly it serves as a storage and search service for geographical information and also as an important tool to provide geographical context on the Web.

The main target user of the Gazetteer is the Geoparser service, also described in this document, but other services and casual users can also use it, as a search service for geographical information.

The remainder of this section provides a detailed description of the Gazetteer service, namely its system specifications, restrictions, design and implementation.

2.1. Functional Specification

2.1.1. Product Position Statement

For	*Europeana Geoparser service, casual users*
Who	Geoparser – Needs to access geographical features to enrich metadata records. Casual Users – Can use to obtain geographic information Other web services – Can use to obtain geographic information
Product name	Gazetteer
That	Provides storage and a browsing service for geographical features, that can be accessed by users or other services.

2.1.2. Stakeholders

Besides Europeana, the Gazetteer may be of interest also for casual users and other services.

Name:	**Europeana – Geoparser service**
Description	The Geoparser service developed for Europeana.
Responsibilities	To enrich the metadata records received from data providers, with structured geographical coordinates and dates.

Name:	**Casual Users / Other web services**
Description	Random users (human and machine)
Responsibilities	To access and obtain geographical information for their own intents and purposes.

2.1.3. System Qualities

Quality	**Description**
Usability	The Gazetteer is accessible through a basic search interface, for browsing geographic features.
Reliability	The system should recover from failed requests. The system must be ready to respond under stress and adverse conditions. The system should be able to return the previous valid state and continue ongoing operations in case of hardware or software failure.
Performance	No measurable performance requirements were identified for the Gazetteer.
User interfaces	Basic search interface.
Interfaces to external systems	The Geoparser makes use of the Gazetteer, as its source of geographical data and historical periods' data.
Software interfaces	An XML over HTTP interface to access the Gazetteer service.

2.1.4. System Constraints

Constraint	**Description**
Operating systems	Any supporting the Java virtual machine
Programming language	Java
Build tool	Maven 2
Application server	*Apache Tomcat 6.x*

2.1.5. Licensing Requirements

Available under dual licensing:

- European Union Public License – *EUPL v.1.1*
- GNU General Public License – *GPLv3*

2.1.6. Applicable Standards

ADL-GP[1] – Alexandria Digital Library Gazetteer Protocol

ADL-FTT[2] – Alexandria Digital Library Feature Type Thesaurus

2.1.7. Use cases

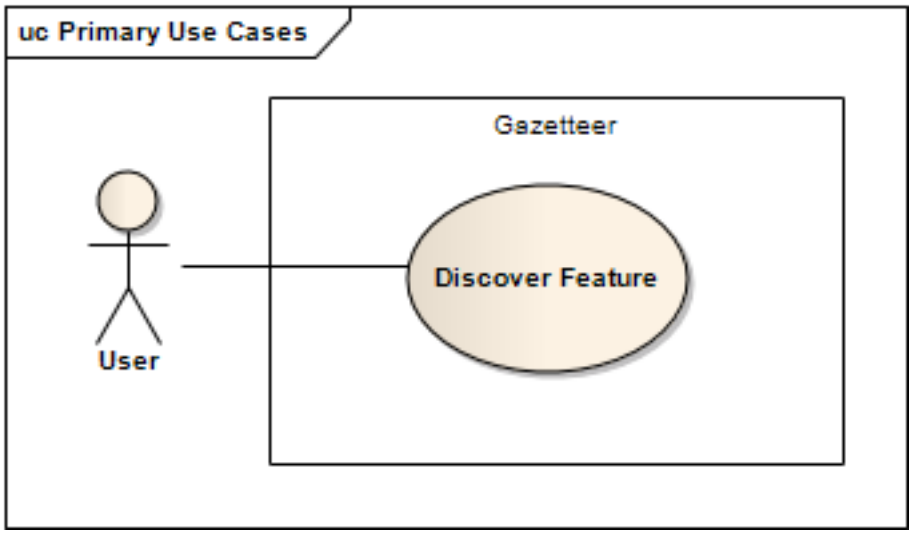


Figure 1 – Gazetteer's Use Cases

	Discover Feature
Description	Search and access a specific feature or set of features based on specified attributes.
Actors	Client application or Human User
Preconditions	None
Flow of events	1. The use case begins when the User (application or human) submits a request either through the user interface or the ADL-GP interface. 2. In case of the user interface:

[1] http://www.alexandria.ucsb.edu/gazetteer/protocol/
[2] http://www.alexandria.ucsb.edu/gazetteer/FeatureTypes/ver070302/

	a. The user submits a feature name through the user interface for search purposes; b. A search request is built and sent to the Gazetteer system; c. The request is then processed and the result list is sent to the user interface for presentation; d. The user can than browse throw the result list or perform another search; e. If the chooses to access a feature, he is presented with a descriptive page of the feature with its contents structured into groups. 3. In the case of the ADL-GP interface: a. A search request is sent to the gazetteer system; b. The request is then processed by the system, a response is built with the list of results from the search and is returned; 4. The use case ends.

2.2. Gazetteer Design

This section describes the main design aspects of the Gazetteer software.

2.2.1. Component Overview

The Gazetteer consists of three software components: the *Gazetteer User Interface*, the *Gazetteer Service* and the *Repository* for storing the features.

The *Gazetteer Service* component is to be built as a Java jar file. This component provides an implementation of the Gazetteer Use Cases, and it can be embedded in other software applications. It serves as a simple service providing the basic functionalities of an International Organization for Standardization (ISO) and Open Geospatial Consortium (OGC) defined gazetteer service.

The *Gazetteer User Interface* component is built as a Java Web application. This component makes all the Gazetteer Use Cases available online. It is dependent on the *Gazetteer Service* component.

The *Repository* is a relational database containing the geographic features managed by the gazetteer service.

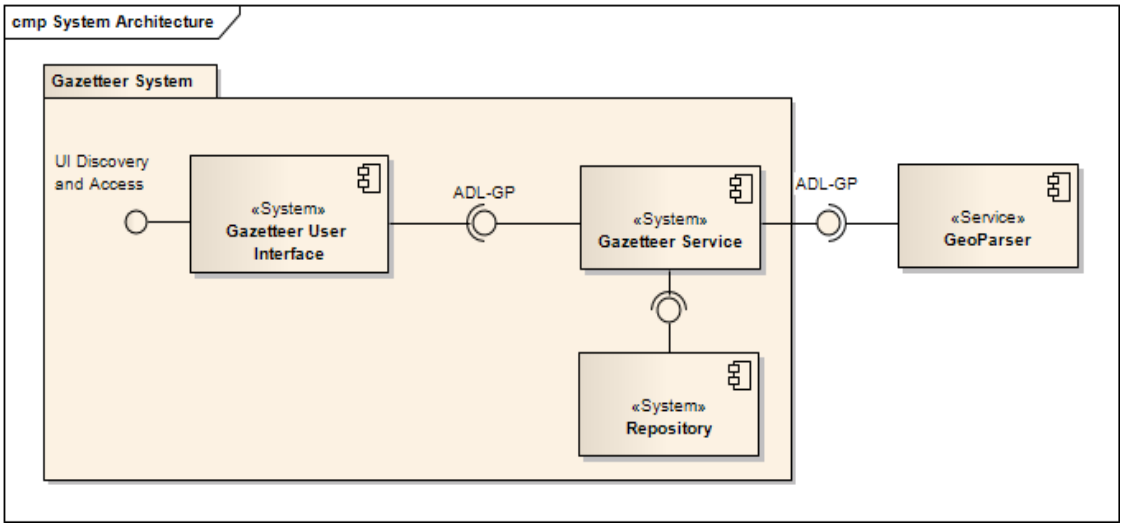


Figure 2 – Gazetteer Components Overview

2.2.2. Component Architecture

The architecture diagram shows the components that make up the system. As described in the previous section, the system is decomposed in three main software components, the *Gazetteer User Interface*, the *Gazetteer Service* and the *Repository*. The *Gazetteer User Interface* is comprised by the Public UI, while the *Gazetteer Service* is comprised of two internal components: Search Support and Storage Manager. This *Gazetteer Service* is setup as a minimal service, providing the basic functionalities established by ISO and OGC standards.

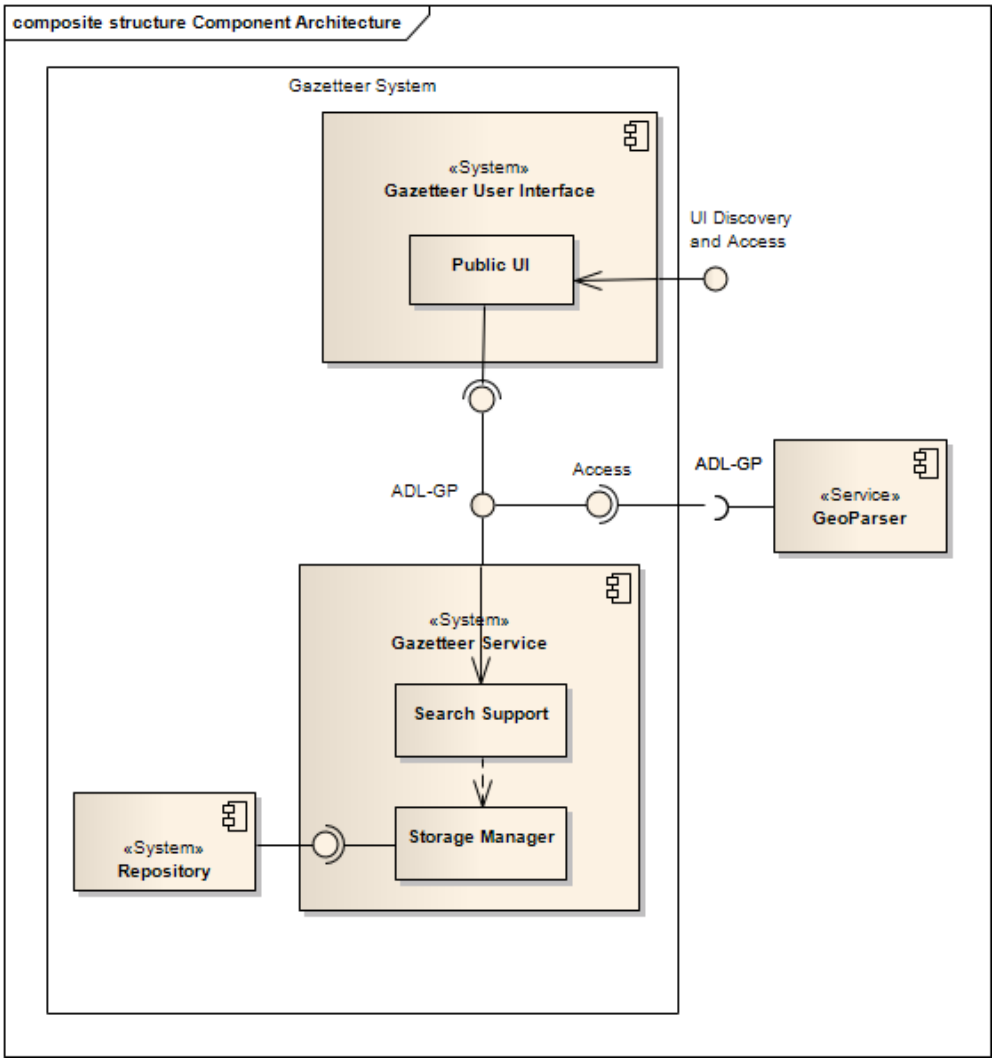


Figure 3 – The high level architecture for the gazetteer system

Component	Description
Public User (interface)	This is the human interface for the Gazetteer. It is responsible for providing operations that require user interaction searching for Features in the Gazetteer.
Administrative User (Interface)	The Gazetteer administration interface. It provides the authorized user with operations for managing the existing content in the Gazetteer Repository.
Search Support	Component responsible for handling the search requests made either through the UI or through the REST interface and processing them.
Storage Manager	Handles all operations regarding access or altering data into the Gazetteer Repository.

2.3. Gazetteer Implementation

2.3.1. Source Code

The source code of the Gazetteer is structured as two main Maven projects:

- gazetteer – Multi-module project. One of the modules (gazetteer-dist) builds a jar containing all necessary modules for integration in other projects.
- gazetteerWebapp – Builds into a web application that makes the gazetteer functionality available on the Internet, via a REST web services interface and a human interface.

Both projects are available on Europeana Labs at:

- gazetteer - https://europeanalabs.eu/svn/contrib/gazetteer
- gazetteerWebapp - https://europeanalabs.eu/svn/contrib/gazetteerWebapp

2.3.1.1. Using the Gazetteer

The Gazetteer is divided into a multi-module project. Since its purpose is to serve as a stand-alone service for browsing geographic features either through human interface or through XML requests over HTTP, it requires deployment as a web-service to function properly. This means it requires deploying the gazetteerWebapp project.

If you desire to fetch the Gazetteer in its entirety, there is a module (*gazetteer-dist*) that builds a jar containing the complete project, in case you want to construct your own interface for it.

2.3.1.2. Deploying the gazetteer web-app

The gazetteerWebapp is a web application that can be deployed in a J2EE application server. It was tested on Apache Tomcat 6.x.

The war file is built using Maven, and then deployed according to the specific application server's instructions.

Regarding configuration, the web-service runs an install file *gaz.install.xml* that builds the gazetteer classes. It should be specified in the deployment descriptor file WEB-INF/web.xml. The only requirement that may need to be configured is the location of the two databases used *(gaz_record*, which contains the xml feature data; and *gaz_index*, which contains indexes for search performance) and the username and password to access them.

These entries are:

```
<bean id="eu.digmap.gaz.database" class="eu.digmap.gaz.database.DatabaseImpl">
<constructor-arg>
      <bean class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
            <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
            <property name="url" value="jdbc:mysql://...:3306/gaz_record"/>
            <property name="username" value="..."/>
            <property name="password" value="..."/>
            ...
```

```
        </bean>
</constructor-arg>
```

And:

```
<bean id="eu.digmap.gaz.index" class="eu.digmap.gaz.database.DatabaseIndexImpl">
<constructor-arg>
        <bean class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
                <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://...:3306/gaz_index"/>
                <property name="username" value="..."/>
                <property name="password" value="..."/>
                ...
        </bean>
</constructor-arg>
```

2.3.2. ADL-GP interface usage

The Gazetteer can be accessed via an XML over HTTP request interface and the request/response protocol is the ADL-GP protocol.

Requests may be invoked using HTTP POST requests to the gazetteer base URL with the required parameters, and all text encoded in UTF-8.

The current deployment of the Gazetteer, for demonstration purposes, has the base URL: http://europeana-geo.isti.cnr.it/gazetteer (/services/gp is the servlet entry point for receiving ADL-GP XML queries).

The base request format is shown in the following figure:

```
<?xml version="1.0" encoding="UTF-8"?>
<gazetteer-service
    xmlns="http://www.alexandria.ucsb.edu/gazetteer"
    version="1.2">
  <query-request>
    <gazetteer-query>
      ...
    </gazetteer-query>
    <report-format>standard</report-format>
  </query-request>
</gazetteer-service>
```

Figure 4 – Base format for a Gazetteer feature query

The query types can then be specified.

2.3.2.1. Query types

The following section describes the types of queries the system supports and some examples.

A. **Identifier Query**
This simple query type fetches the feature, if existing, which matches with the provided internal identifier as argument.

```
<identifier-query identifier="http://sws.geonames.org/299042/" />
```

B. **Name Query**
This query type fetches the features whose name, being it primary or alternative, matches with the provided text as argument, depending on the matching operator provided as argument. Those different types of operators may be:

a. **"equals"** – A name, in its entirety, matches the exact text;

b. **"contains-all-words"** – A name must contain all words specified in the text, in no particular order;

c. **"contains-any-words"** – A name must contain at least one of the words specified in the text;

d. **"contains-phrase"** – A name must contain the exact sequence of words specified in the text

```
<name-query operator="equals" text="lisboa" />
```

C. **Class Query**
This query type fetches the features whose classification matches the term provided. The thesaurus used for search purposes is the ADL Feature Type Thesaurus.

```
<class-query thesaurus="ADL Feature Type Thesaurus" term="streams" />
```

D. **Relationship Query**
This query type fetches the features that share a specified relation type with a specified feature internal identifier. The relations covered are:

a. **partOf –** feature is contained within the area of target feature;

b. **contains –** feature contains target feature;

c. **adjacentTo –** feature has a frontier or border with target feature;

d. **inCountry –** feature is part of target country feature;

e. **inContinent –** feature is part of target continent feature;

```
<relationship-query relation="inCountry"
   target-identifier="http://sws.geonames.org/2990440/"/>
```

These individual query types may be combined with one another in order to form more complex queries.

```
<and>
  <class-query thesaurus="ADL Feature Type Thesaurus" term="streams" />
  <name-query operator="equals" text="Danube" />
</and>
```

2.3.2.2. Query Response

The XML response for any query request consists of a list of all search matches to the query parameters. The following figure depicts its basic format:

```
<?xml version="1.0" encoding="UTF-8"?>
<gazetteer-service xmlns="http://www.alexandria.ucsb.edu/gazetteer">
    <query-response>
        <standard-reports>
            <adlgp:gazetteer-standard-report xmlns:adlgp="http://www.alexandria.ucsb.edu/gazetteer"
                                    xmlns:fn="http://www.w3.org/2005/02/xpath-functions"
                                    xmlns:gaz="http://www.digmap.eu/gazetteer/version1.0#"
                                    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
                ....
            </adlgp:gazetteer-standard-report>
            ....
        </standard-reports>
    </query-response>
</gazetteer-service>
```

Figure 5 – Base format for a gazetteer query-response

An example response follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<gazetteer-service xmlns="http://www.alexandria.ucsb.edu/gazetteer">
<query-response>
    <standard-reports>
    <adlgp:gazetteer-standard-report xmlns:adlgp="http://www.alexandria.ucsb.edu/gazetteer"
                                     xmlns:fn="http://www.w3.org/2005/02/xpath-functions"
                                     xmlns:gaz="http://www.digmap.eu/gazetteer/version1.0#"
                                     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <adlgp:identifier>http://sws.geonames.org/1511497/</adlgp:identifier>
        <adlgp:place-status>current</adlgp:place-status>
        <adlgp:display-name xml:lang="">Anzha</adlgp:display-name>
        <adlgp:names>
            <adlgp:name primary="false" status="current" xml:lang="">Anzha</adlgp:name>
            <adlgp:name primary="false" status="current" xml:lang="no">Anzja</adlgp:name>
            <adlgp:name primary="false" status="current" xml:lang="ru">????</adlgp:name>
        </adlgp:names>
        <gaz:countryCode>RU</gaz:countryCode>
        <adlgp:bounding-box>
            <Envelope xmlns="http://www.opengis.net/gml">
                <lowerCorner>
                    <coord><X>55.34</X><Y>95.13</Y></coord>
                </lowerCorner>
                <upperCorner>
                    <coord><X>55.34</X><Y>95.13</Y></coord>
                </upperCorner>
            </Envelope>
        </adlgp:bounding-box>
        <adlgp:footprints>
            <adlgp:footprint adlgp:primary="true">
                <Point xmlns="http://www.opengis.net/gml">
                    <coord><X>55.34</X><Y>95.13</Y></coord>
                </Point>
            </adlgp:footprint>
            <adlgp:footprint>
                <gaz:CSquares>1805:495:393</gaz:CSquares>
            </adlgp:footprint>
            <adlgp:footprint>
                <gaz:GeoHash>y1gdudrdcetn</gaz:GeoHash>
            </adlgp:footprint>
            <adlgp:footprint>
                <gaz:WKT>POINT (55.34 95.13)</gaz:WKT>
            </adlgp:footprint>
        </adlgp:footprints>
        <adlgp:classes>
            <adlgp:class primary="true" thesaurus="ADL Feature Type Thesaurus">streams</adlgp:class>
            <adlgp:class primary="false" thesaurus="Geonames Feature Type Thesaurus">H</adlgp:class>
        </adlgp:classes>
        <adlgp:relationships>
            <adlgp:relationship relation="part of" target-identifier="http://sws.geonames.org/1502020/" target-name=""/>
            <adlgp:relationship relation="in country" target-identifier="http://sws.geonames.org/2017370/" target-name=""/>
        </adlgp:relationships>
    </adlgp:gazetteer-standard-report>
    </standard-reports>
</query-response>
</gazetteer-service>
```

Figure 6 – Example search result response

For each standard report entry, there are these main attributes:

- The feature's internal identifier (*adlgp:identifier*), which corresponds to the source provider's identifier;
- The display name (*adlgp:display-name*), which is the primary name associated with the feature;
- A list of alternative names (*adlgp:names*), where each one may have a language code, corresponding to the source language for that name;
- A country code (*gaz:countryCode*) that identifies the pertaining country of that feature;
- Geographic representations of the feature (t, Bounding Box, …);
- The feature types that classify the feature (*adlgp:classes*):
    - For each one, there is an attribute that determines if that class is the primary type and another attribute that identifies the thesaurus of origin for that class;
- A list of relationships with other features (*adlgp:relationships*):

- For each one, there is an attribute that identifies the relationship type and another that identifies the target feature identifier;

- Some other elements may be present, depending on the feature's provenience, such as Postal Codes, Population, external information articles (Wikipedia[3], DBpedia[4]), etc.

2.3.3. Human interface usage

Besides the web-service interface, there is a basic human interface for searching geographic features.

Figure 7 – Main page

A user can then perform a quick search for a feature name, for which the following page is produced:

[3] http://www.wikipedia.org/
[4] http://www.dbpedia.org/

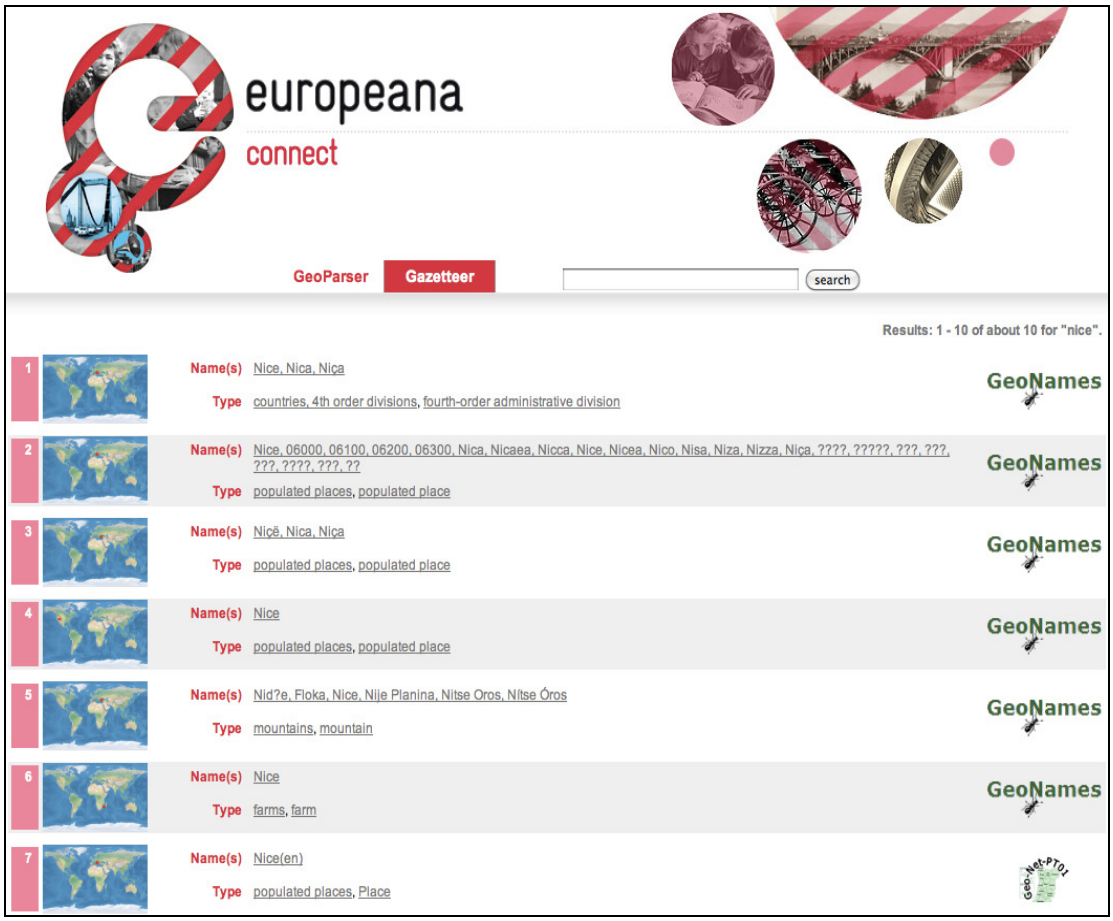


Figure 8 – Excerpt of a search result list for the word "Nice"

In this page one can already visualize, for each search match, their:

- Feature name;
- Feature type;
- Graphic approx. location;
- Provenience;

From this point, the user can either choose one of the results to browse within or perform another search.

When choosing one of the features, the user is presented with the following description page:

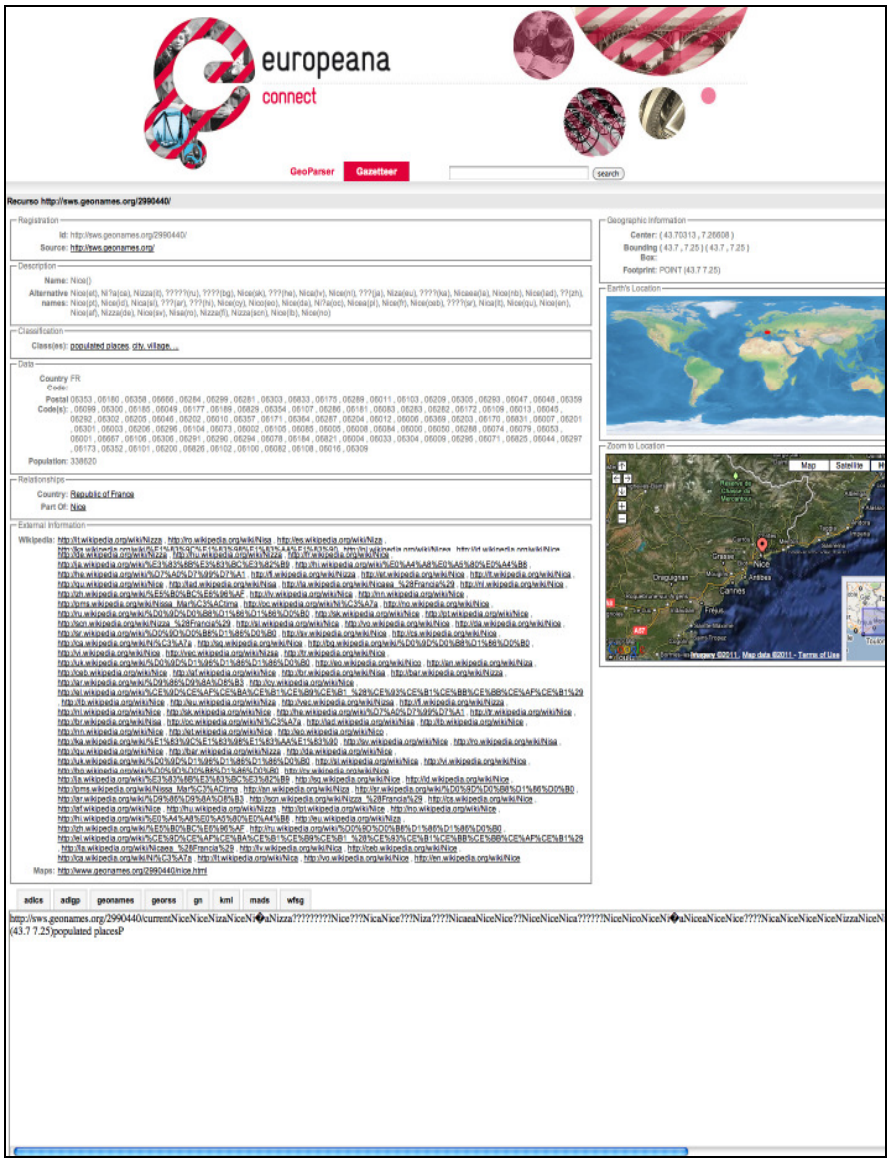


Figure 9 – Description page of a selected feature

In this description page, one can see the structured information of a feature, divided into groups:

- Registration – Contains information pertaining to the internal organizing of the feature:
    - ID – The feature identifier;
    - Source – The data provider
- Description – Contains information that identifies the feature:
    - Name – The primary name associated with the feature;
    - Alternative Names: A list of the feature's alternative names which may have in brackets "()" the language code in which the feature is named;
- Classification – Contains a list of classifying terms for the feature;
- Data – Various information that describes the feature:

    - Country Code – A two-letter code that identifies a country name, as defined by the ISO 3166[5];
    - Postal Code(s) – One or more series of letters and/or digits appended to a postal address for the purpose of sorting mail [may not be present in all features];
    - Population – The approx. total population residing in the feature area [may not be present in all features];
- Relationships – Contains information regarding connections between this feature and others, regarding containment, proximity, among others;
- External Information – Contains links to other information articles or maps that add more information regarding the feature and/or provide its contents in another language [may not be present in all features];
- Geographic information – Contains positional information about the feature in the form of geographic coordinates;

Below all this information are structured representations of the feature in several data types, such as the ADL Content Standard[6], Geonames ontology[7] or KML[8].

[5] http://www.iso.org/iso/country_codes.htm
[6] http://www.alexandria.ucsb.edu/gazetteer/ContentStandard/version3.2/GCS3.2-guide.htm
[7] http://www.geonames.org/ontology/documentation.html
[8] http://code.google.com/apis/kml/

3. Geoparser

Unstructured text, or semi-structured text (e.g. metadata records), may contain mentions to places and historical periods that are not directly usable by software applications. Geoparsing consists in automatically extracting structured information about places and time periods from these textual resources.

The Geoparser is a web service that uses information extraction techniques to automatically identify names of places and time periods that are mentioned in unstructured text. It works together with the Gazetteer to be able to assign coordinates and dates with the mentioned places and periods.

The target users of the Geoparser are Europeana and aggregators and data providers which need to enrich the object descriptions by analyzing geographic or temporal references in existing metadata records.

In this section a description of the Geoparser is provided, covering its specification, design, implementation, usage and evaluation.

3.1. Functional specifications

This section presents the results of the functional requirements for the Geoparser.

3.1.1. Product Position Statement

For	*Europeana, aggregators and data providers*
Who	Who need to enrich the object descriptions by analyzing geographic or temporal references in existing metadata records
Product name	Geoparser
That	It takes textual resources or metadata records as input. It identifies the mentioned names of places and time periods, and assigns them geographical coordinates or dates.

3.1.2. Stakeholders

Besides Europeana, the Geoparser may be of interest also for aggregators and data providers who want to enrich their own data.

Name:	**Europeana - content ingestion**
Description	The content ingestion team of Europeana
Responsibilities	To enrich the metadata records received from data providers, with structured geographical coordinates and dates.

Name:	**Aggregators**
Description	Aggregators of digital objects for Europeana
Responsibilities	To enrich the metadata records and textual resources with structured geographical coordinates and dates.

Name:	**Data providers**
Description	Owners or hosts of digital objects
Responsibilities	To enrich the metadata records and textual resources with structured geographical coordinates and dates.

3.1.3. System Qualities

Quality	**Description**
Usability	The Geoparser is not intended for human use.
Reliability	The system should recover from failed requests. The system must be ready to respond under stress and adverse conditions. The system should be able to return the previous valid state and continue ongoing operations in case of hardware or software failure.
Performance	No measurable performance requirements were identified for the Geoparser.
User interfaces	The Geoparser is not intended for human use. No user interface is necessary.
Interfaces to external systems	The Geoparser makes use of the Gazetteer, as its source of geographical data and historical periods' data. It uses the web service interface of the Gazetteer.
Software interfaces	An XML over HTTP interface (REST) to access the Geoparsing service. An application programmable interface (API) for embedding the Geoparser in other applications.

3.1.4. System Constraints

Constraint	Description
Operating systems	Any OS supporting the Java virtual machine
Programming language	Java 6
Build tool	Maven 2
Application server	*Apache Tomcat 6.x*

3.1.5. Licensing Requirements

Available under dual licensing:

- European Union Public Licence – EUPL v.1.1
- GNU General Public License – GPLv3

3.1.6. Applicable Standards

The metadata format supported for geoparsing is Europeana Semantic Elements v3.3[9].

Geographic coordinates are encoded in signed decimal degrees, as used in the DCMI Point Scheme[10].

Dates are encoded according to the XML schema Date and Time format[11].

3.1.7. Use cases

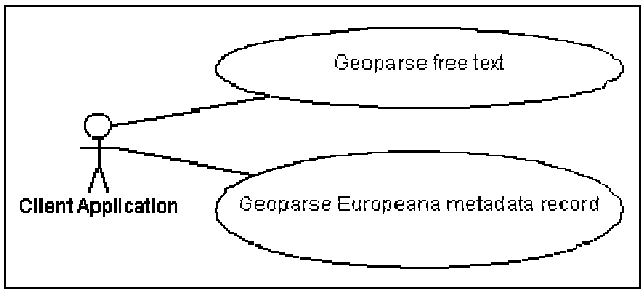


Figure 10 – Geoparser's use cases

[9] http://group.europeana.eu/web/guest/technical-requirements/
[10] http://dublincore.org/documents/dcmi-point/
[11] http://www.w3.org/TR/2001/REC-xmlschema-2-20010502

	Geoparse Europeana Metadata Record
Description	Recognizing geographic or temporal references in descriptive fields of the Europeana metadata records, and assignment of coordinates and dates to the recognized references.
Actors	Client application: a software client application
Preconditions	None
Flow of events	1. The use case begins when the Client application submits a Geoparsing request containing a metadata record. 2. The text is extracted from the descriptive metadata elements of the record. 3. The geoparser passes the text to a named entity recognition library that recognizes the geographic and temporal references. 4. The geographic references are searched in the Gazetteer, and coordinates are associated with the references. 5. The temporal references are searched in the Gazetteer, and dates are associated with the references. 6. The response with the identified references is sent to the Client application. 7. The use case ends.
	Geoparse Free Text
Description	Recognizing geographic or temporal references in free text, and assignment of coordinates and dates to the recognized references.
Actors	Client application: a software client application
Preconditions	None
Flow of events	1. The use case begins when the Client application submits a Geoparsing request containing the source text. 2. The Geoparser passes the text to a named entity recognition library that recognizes the geographic and temporal references. 3. The geographic references are searched in the Gazetteer, and coordinates are associated with the references. 4. The temporal references are searched in the Gazetteer, and dates are associated with the references. 5. The response with the identified references is sent to the Client application. 6. The use case ends.

3.2. Geoparser Design

This section presents the main design aspects of the Geoparser software.

3.2.1. Components overview

The Geoparser consists of two software components: the *Geoparser* and the *GeoparserServiceREST*.

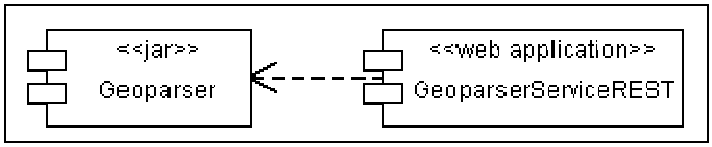


Figure 11 – Geoparser components

The *Geoparser* component is built as a Java jar file. This component provides an implementation of the Geoparser Use Cases, and it can be embedded in other software applications where the REST interface overhead is not desirable.

The *GeoparserServiceREST* component is built as a Java Web application. This component makes all the Geoparser Use Cases available online through a REST Web service. It is dependent on the *Geoparser* component.

3.2.2. The Geoparser component

The Geoparser component implements all the Use Cases for the Geoparser. It relies on named entity recognition (NER) to locate named entities in unstructured text. The most effective NER techniques are language dependent, therefore the most effective geoparsing requires the language of the text to be known by the Geoparser. Language specific tools will be provided to Europeana in WP2 of EuropeanaConnect, and will include NER tools. In order to allow the above tools to be used in the Geoparser in the future, the Geoparser's design defines interfaces for the geoparsing requirements for NER and language detection. This will allow the language specific tools provided from WP2 to be used within the Geoparser and improve its results.

The class diagram in Figure 12 shows the main classes and interfaces of the Geoparser component, and Figure 13 shows the class hierarchy of the supported named entities. The description of each class and interface is provided in the table that follows.

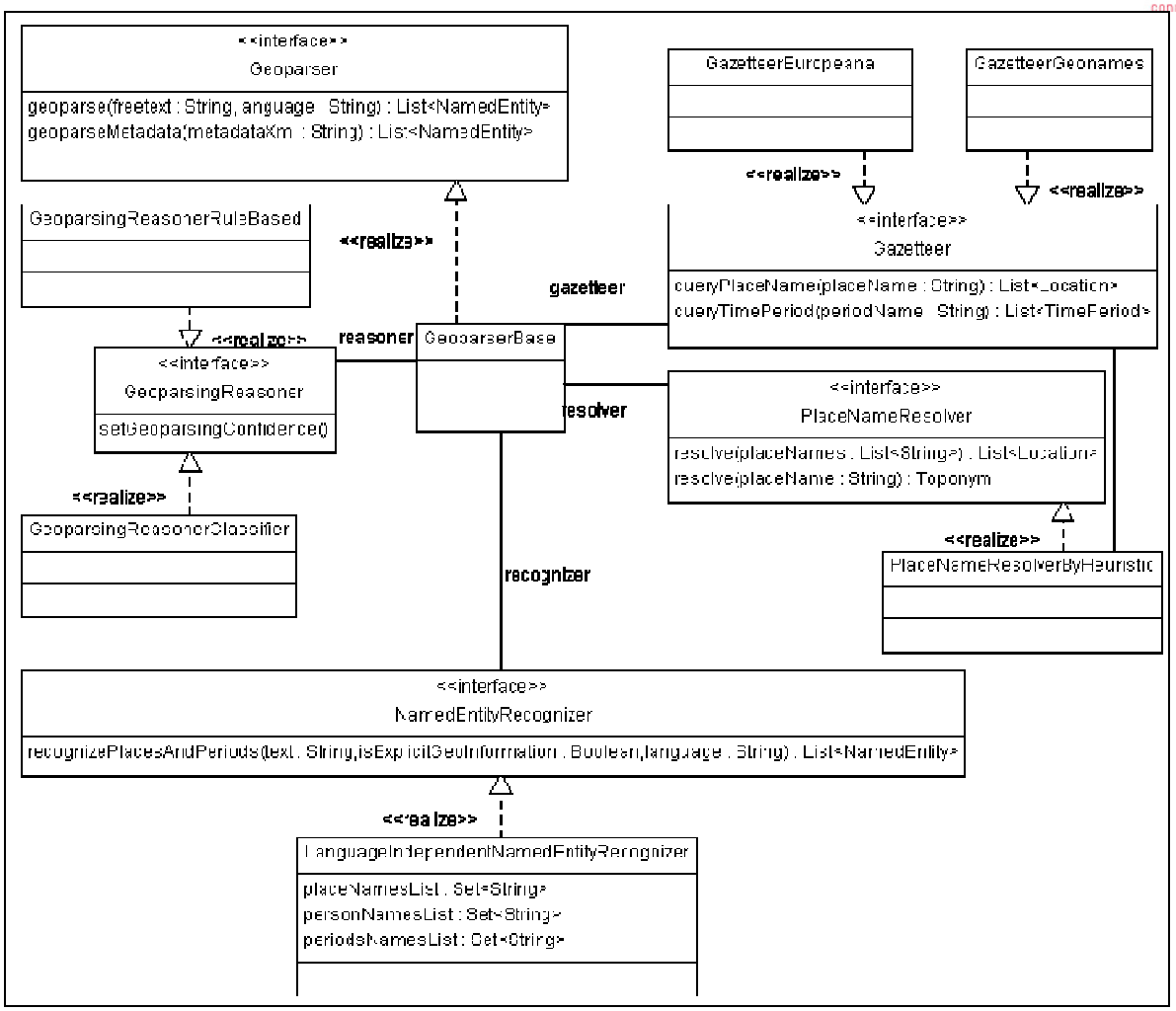


Figure 12 – High level design of the Geoparser

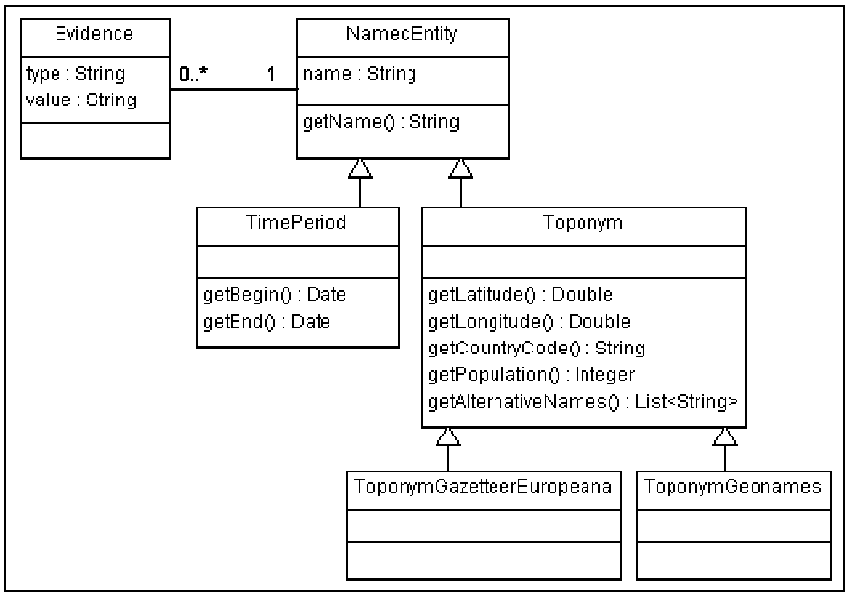


Figure 13 – Named entities class hierarchy

Class/Interface	Description
Geoparser (interface)	This is the interface for the Geoparser. It specifies the methods for implementing all the Geoparser Use Cases.
GeoparserBase	The base implementation of the Geoparser interface. It requires an implementation for each of the NamedEntityRecognizer, Gazetteer, PlaceNameResolver, and GeoparsingReasoner interfaces. It orchestrates them to execute the geoparsing methods.
NamedEntityRecognizer (interface)	Interface for named entity recognition implementations. For Geoparsing, implementations may only output Locations and Time Periods. Other entity types may be in the output, and, depending on the implementation of other interfaces of the Geoparser, they may be used for some purpose or simply be ignored.
LanguageIndependent-NamedEntityRecognizer	An implementation of a NamedEntityRecognizer that relies only on techniques for NER that are language independent, such as capitalization of words, word position in the sentence, lists of names of the entities, etc.
PlaceNameResolver (interface)	Interface for implementations of resolvers of name places. These resolvers take as input place names and try to geo-reference them. When a resolution request contains more than one place name, the resolver should assume that the place names are related in some way (for example, they were found in the same paragraph or the same metadata record). It should try to use that information to find the most likely resolution for the place names.
Gazetteer (interface)	An interface for data sources of geo referenced place names and time periods.
GazetteerEuropeana	An implementation of the Gazetteer interface, which uses the Europeana Gazetteer as a data source.
GazetteerGeonames	An implementation of the Gazetteer interface, which uses the Geonames web service a data source. Time periods are not part of this data set.

PlaceNameResolverByHeuristic	An implementation of a PlaceNameResolver based on heuristics. It uses a Gazetteer to query the place names and obtain all the candidate locations for each place name. It implements two different heuristics: one for resolving one single place name, and one for resolving several place names together. When resolving a single place name the heuristic chooses the best candidate based on: • Minimum edit distance of the place name to the name, or one of the alternative names of the candidates. • The population of the candidate. The higher the population, the most likely it is to be the best candidate. When resolving several place names, the heuristic chooses the best candidate based on: • Minimum edit distance of the place name to the name, or one of the alternative names of the candidates. • The population of the candidate. The higher the population, the most likely it is to be the best candidate. • The minimum geographic distance to the candidates of the other place names. • The minimum geographic area obtained when making a polygon with the coordinates of the candidates of the other place names.
GeoparsingReasoner (interface)	Interface for implementations of the final reasoning of the geoparsing process. Reasoners have access to all named entities that have been recognized and also to all evidence that was gathered to support the recognition and resolution of the entity. Reasoners may exclude entities from the result or assign them a confidence value.

GeoparsingReasonerRuleBased	A rule based implementation of a GeoparsingReasoner. The rules assign scores to each type of evidence, and calculate the total score of the evidences found. Based on the score, the named entity may be excluded from the geoparsing response (if the score is too low) or the entity is included in the result (with a confidence value assigned by the reasoner).
GeoparsingReasonerClassifier	A machine learning based implementation of a GeoparsingReasoner. It is based on the Random Forest[12] classification algorithm, and was trained from the training set used for the evaluation of the Geoparser (described in Section 3.4).
NamedEntity	A named entity found and processed in a geoparsing request.
TimePeriod	A type of named entity that refers to a time period. May be a named time period, such as *XIX century*, or a date, such as *25th of December 1975*.
Toponym	A type of named entity that refers to a location. These entities are found by the NamedEntityRecognizer and processed during the rest of the geoparsing processing.
ToponymGazetteerEuropeana	A specialization of the Toponym class, adapted to the GazetteerEuropeana.
ToponymGeonames	A specialization of the Toponym class, adapted to the GazetteerGeonames.
Evidence	A name-value pair containing any information that may be relevant for reasoning on the correct recognition and resolution of the entity.

3.2.3. The GeoparserServiceREST component

The GeoparserServiceREST component is built as a Java Web application. This component provides a REST interface to the Geoparser, therefore it is dependent on the Geoparser component.

[12] Breiman, Leo (2001). "Random Forests". Machine Learning 45 (1): 5–32. doi:10.1023/A:1010933404324

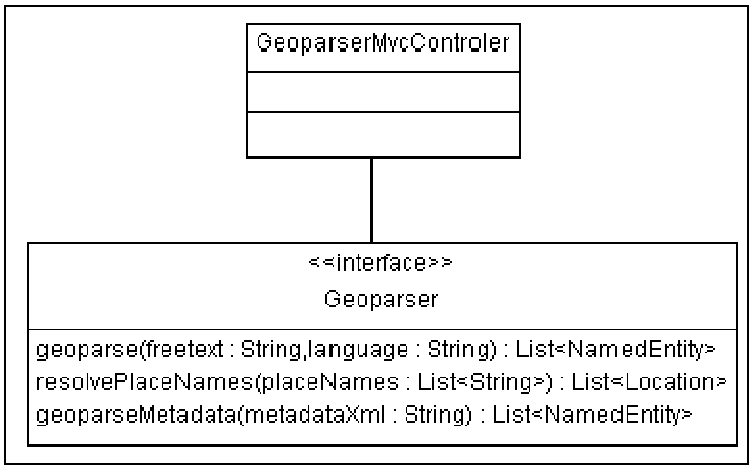


Figure 14 – The GeoparserServiceREST component

Class/Interface	Description
GeoparserServiceREST	This class implements the controller of the Geoparser as in the Model-View-Controler pattern. It allows all methods of the Geoparser component to be exposed as a REST web service.

3.3. Implementation

This section describes the implementation aspects of the Geoparser prototype. It describes where the software can be found, how to deploy it, and how it can be used by other applications either by integrating the Geoparser library or by using the REST web service.

3.3.1. Source code

The Source Code of the Geoparser is structured as two Maven projects:

- geoparserLib – Builds into a jar for integration in other projects.
- geoparserService – Builds into a web application that makes the geoparser functionality available on the Internet, via a REST web services interface.

Both projects are available on Europeana Labs at:

- geoparserLib – https://europeanalabs.eu/svn/contrib/geoparser
- geoparserService – https://europeanalabs.eu/svn/contrib/geoparserService

3.3.2. Using geoparserLib

The geoparserLib may be integrated into other Maven projects by adding the following dependency:

```
<dependency>
        <groupId>europeana</groupId>
        <artifactId>geoparserLib</artifactId>
        <version>1.0</version>
</dependency>
```

The Geoparser requires a directory on the file system, which will be used for storing files needed by the Geoparser. This working directory should already exist on the file system when running the Geoparser, and its location is configured when requesting a Geoparser instance.

The Geoparser is initialized via the following static method invocation:

```
europeana.geoparser.GeoparserFactory.newInstance(File workingDirectory, String europeanaGazetteerBaseURL)
```

The second parameter is the URL for accessing the web services interface of the Gazetteer[13]. Please refer to the Gazetteer's documentation for more information.

Refer to the europeana.geoparser.Geoparser java interface documentation for using the Geoparser functionality.

[13] At the time of writing of this document the base URL is http://europeana-geo.isti.cnr.it/gazetteer/services/gp

3.3.3. Deploying the geoparserService

The geoparserService is a web application that can be deployed in a J2EE application server. It was tested on Apache Tomcat 6.x.

The war file is built using Maven, and then deployed according to the specific application server's instructions.

Regarding configuration, the GeoparserService requires only the address of the Gazetteer. It should be specified in the deployment descriptor file WEB-INF/web.xml, as an initialization parameter of the controller servlet:

```
<servlet>
        <servlet-name>Controller</servlet-name>
        <servlet-class>europeana.geoparser.service.ControllerServlet</servlet-class>
        <init-param>
                <param-name>gazetteerBaseUrl</param-name>
                <param-value>http://europeana-geo.isti.cnr.it/gazetteer/services/gp</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
```

No further configuration is necessary since it automatically configures and creates the working directory for the GeoparserLib to a location inside the web application directory: "WEB-INF/tmp".

3.3.4. The REST interface of the GeoparserService

Applications may use the Geoparser over a REST interface instead of integrating the GeoparserLib jar.

The Geoparser REST interface makes available two types of requests: one for geoparsing free text; and another for geoparsing Europeana metadata records.

Both requests may be invoked using HTTP POST of GET requests to the geoparsing base URL with the required parameters, and all text encoded in UTF-8.

The current deployment of the Geoparser, for demonstration purposes, has the base URL: http://europeana-geo.isti.cnr.it/geoparser.

3.3.4.1. Geoparsing free text

URL:

- <base_URL>/freeText

Parameters:

- "freeText" – The input text to geoparse encoded in UTF-8 (required)
- "language" – The language of the text (optional)
- "stylesheet" – A URL pointing to a XSLT stylesheet. The geoparser will add a stylesheet processing instruction to the XML response (optional)

The XML response for this request list all entities recognized in the input text. It also presents the input text with annotations on the recognized entities. An example response follows:

```
- <geoparsingResult>
  - <entities>
      <PLACE entityURI="http://sws.geonames.org/6255148/" confidence="0.5475" latitude="48.69096"
      longitude="9.14062">Europe</PLACE>
      <PLACE entityURI="http://sws.geonames.org/2267057/" confidence="0.51096493" latitude="38.71667"
      longitude="-9.13333">Lisbon</PLACE>
      <PLACE entityURI="http://sws.geonames.org/3117735/" confidence="0.6050296" latitude="40.4165"
      longitude="-3.70256">Madrid</PLACE>
      <PLACE entityURI="http://sws.geonames.org/2988507/" confidence="0.504204" latitude="48.85341"
      longitude="2.3488">Paris</PLACE>
      <TIME entityURI="http://www.digmap.eu/time_periods#1942" confidence="0.6" begin="1801-01-01T00:00:00Z"
      end="1900-01-01T00:00:00Z">19th century</TIME>
    </entities>
  - <annotatedText>
      Example: History of Architecture in
      <PLACE entityURI="http://sws.geonames.org/6255148/" confidence="0.5475" latitude="48.69096"
      longitude="9.14062">Europe</PLACE>
      : the cases of
      <PLACE entityURI="http://sws.geonames.org/2267057/" confidence="0.51096493" latitude="38.71667"
      longitude="-9.13333">Lisbon</PLACE>
      ,
      <PLACE entityURI="http://sws.geonames.org/3117735/" confidence="0.6050296" latitude="40.4165"
      longitude="-3.70256">Madrid</PLACE>
      and
      <PLACE entityURI="http://sws.geonames.org/2988507/" confidence="0.504204" latitude="48.85341"
      longitude="2.3488">Paris</PLACE>
      of the
      <TIME entityURI="http://www.digmap.eu/time_periods#1942" confidence="0.6" begin="1801-01-01T00:00:00Z"
      end="1900-01-01T00:00:00Z">19th century</TIME>
      .
    </annotatedText>
  </geoparsingResult>
```

For the PLACE entities, the response contains the following data:

- entityURI – The URI that uniquely identifies the place in the Europeana Gazetteer.
- latitude – The latitude of the geographic centre of the place. In decimal degrees (WGS 84).
- longitude - The latitude of the geographic centre of the place. In decimal degrees (WGS 84).
- Name – The main name of the place.
- confidence – The confidence of the geoparsing result for the entity. A value from 0 (zero) to 1. It indicates the probability that the entity is indeed a place and that it was resolved to the correct place.

For the TIME entities, the response contains the following data:

- entityURI – The URI that uniquely identifies the time period in the Europeana Gazetteer.
- begin – The date when the time period begins. In ISO 8601.
- end – The date when the time period ends. In ISO 8601.

- Name – The main name of the time period place, or the date found in the source text.
- confidence – The confidence of the geoparsing result for the entity. A value from 0 (zero) to 1. It indicates the probability that the entity is indeed a time period and that it was resolved to the correct time period.

3.3.4.2. Geoparsing a metadata record

URL:

- <base_URL>/metadata

Parameters:

- "metadata" – The input metadata record to geoparse. The record should be encoded in XML according to the ESE schema (required).
- "stylesheet" – A URL pointing to a XSLT stylesheet. The geoparser will add a stylesheet processing instruction to the XML response (optional).

The XML response for this request list all entities recognized in the input metadata record. It also presents the metadata record with annotations on the recognized entities in the respective elements. An example response follows:

```
<geoparsingResult>
  <entities>
    <PLACE entityURI="http://sws.geonames.org/660013/" confidence="0.885" latitude="64.0" longitude="26.0">Finland</PLACE>
    <PLACE entityURI="http://sws.geonames.org/830706/" confidence="0.7125" latitude="60.5" longitude="26.0">Östra
    Nyland</PLACE>
    <PLACE entityURI="http://sws.geonames.org/660561/" confidence="0.59657896" latitude="60.4"
    longitude="25.66667">Borgå</PLACE>
  </entities>
  <annotatedRecord>
    <dc:title>Herrbacka skogsmark och areal</dc:title>
    <dcterms:issued>2008-03-11</dcterms:issued>
    <dcterms:spatial>
      <PLACE entityURI="http://sws.geonames.org/660013/" confidence="0.885" latitude="64.0"
      longitude="26.0">Finland</PLACE>
    </dcterms:spatial>
    <dcterms:spatial>
      <PLACE entityURI="http://sws.geonames.org/830706/" confidence="0.7125" latitude="60.5" longitude="26.0">Östra
      Nyland</PLACE>
    </dcterms:spatial>
    <dcterms:spatial>
      <PLACE entityURI="http://sws.geonames.org/660561/" confidence="0.59657896" latitude="60.4"
      longitude="25.66667">Borgå</PLACE>
    </dcterms:spatial>
    <dc:description>Herrbacka skogsmark och areal. Odaterad.</dc:description>
    <dcterms:isPartOf>Stensböle gårds arkiv</dcterms:isPartOf>
    <dcterms:isPartOf>SLSA 1070</dcterms:isPartOf>
    <dc:language>swe</dc:language>
    <dc:type>map</dc:type>
    <dc:format>image/jpeg</dc:format>
    <europeana:uri>
      http://www.europeana.eu/resolve/record/01201/3C2F001170A54C7A789861F17CD554F1E7359EFC
    </europeana:uri>
    <europeana:object>
      http://www.sls.fi/databasen/slsa1070/slsa1070_k14.jpg
    </europeana:object>
  </annotatedRecord>
</geoparsingResult>
```

The response contains the same data on the PLACE and TIME entities, as the free text geoparsing response.

The geoparsing of metadata records is only performed on the relevant fields. For example, the text within creator elements is not analysed, because any reference two places in these elements would be related to the creator and not to the digital object itself. The text within the following elements is considered for geoparsing:

- dcterms:temporal (geoparsed only for TIME entities)
- dcterms:spacial (geoparsed only for PLACE entities)
- dc: coverage
- dc:title
- dcterms:alternative
- dc:subject
- dc:description
- dcterms:tableOfContents

3.3.5. Connecting to the Gazetteer

Only the most relevant data about the recognized entities is provided in the Geoparser response. The complete data is made available by the Gazetteer, which can be queried using the entity URI given by the Geoparser. Please refer to the Gazetteer's documentation for more information.

3.4. Evaluation of the results

The results of the Geoparser were evaluated by two methods. The first method, by cross validation, allowed to evaluate if the geoparsing reasoning, created by machine learning, was generalizable to other data besides the evaluation collection. The second evaluation method was to compare the results against those obtained by another existing Geoparsing service, the Yahoo! Placemaker[14].

An evaluation of the Geoparser was performed on a selected collection of ESE metadata records. This collection was created by selecting records that contained place names that exist on the Gazetteer data set. In order to have heterogeneous data, only up to 20 records were selected from each data provider. This allowed the collection to have diverse languages and metadata with different characteristics. A total of 752 records were selected for the evaluation collection.

The evaluation collection was geoparsed by the two Geoparsers and the results of both were inspected and annotated by a person. The manual annotation was sometimes uncertain, because the metadata record may not contain enough information to enable a person to do the annotations. Some sentences with place names may be too small and no other information is available in the metadata record to support a decision, for example. So annotation was performed according to the following criterion:

- Place names and named time periods were annotated with the identifier (an URI) of the time period in the Gazetteer.
- Dates were annotated with a start and end date of the period they represent.
- If the annotator was not sure if a mentioned entity was a place, he would annotate it as *unknown*. These annotations were not considered for the evaluation of the results.
- If the annotator was sure that a mentioned entity was a place, but could not know for sure how to disambiguate it, it would be annotated as *location_unknown.* These annotations were used only for testing the entity recognition phase of geoparsing, and where not considered for the evaluation of the geoparsing results.
- If for any place name, the annotator was able to identify the country of the place but was not able to disambiguate between alternatives within that country, he would use its own judgement to choose the most likely one (the most populated, for example). Although these annotations seam uncertain, they mimic the desired behaviour for the Geoparser, so they were considered for the evaluation of the results.
- If a place name was not being used to refer to a place (for example if it is an ambiguous word, or is the name of a person) it would be annotated as *not_location.* These annotations were considered for the evaluation of the precision of geoparsing.

[14] http://developer.yahoo.com/geo/placemaker/

- Similarly to places, if a time period name was not being used to refer to a time period, it would be annotated as *not_time,* and were considered for the evaluation of the precision of geoparsing.

In total, the evaluation collection contains 3811 annotated entity names, of which 2823 are place names and 988 are time periods or dates.

3.4.1. Evaluation of the machine learning based geoparsing reasoner

The reasoning component of the Geoparser was trained on the evaluation collection. In order to evaluate if it was not over fitted to the evaluation collection, we wanted to estimate how accurately it would perform in practice. For this purpose, a cross-validation test was performed.

The cross-validation test involves partitioning the evaluation collection into complementary subcollections, training the random tree on one subset, and validating its results on the other subset. Ten rounds of cross-validation were performed using different partitions, and the validation results were averaged over the rounds.

For this evaluation, the following measurements were taken:

- Precision: the percentage of correctly identified entities in all entities found.
- Recall: the percentage of entities found compared to all existing entities.
- F_1-measure: the weighted harmonic mean of precision and recall.
- $F_{0,5}$-measure: the weighted harmonic mean of precision and recall, where precision is weighted twice as much as recall.

The measurements were taken at five levels of minimum confidence given by the Geoparser. They are shown in Figure 15 and Figure 16.

The results obtained allowed us to accept the machine learning based geoparsing reasoning with confidence that its output is applicable to other data besides the evaluation collection.

The evaluation results also provide solid guidance for external applications that, when using the Geoparser, need to choose a minimum confidence level, since to obtain very high levels of precision, recall may lower considerably. Applications should choose the appropriate confidence level for their own objectives.

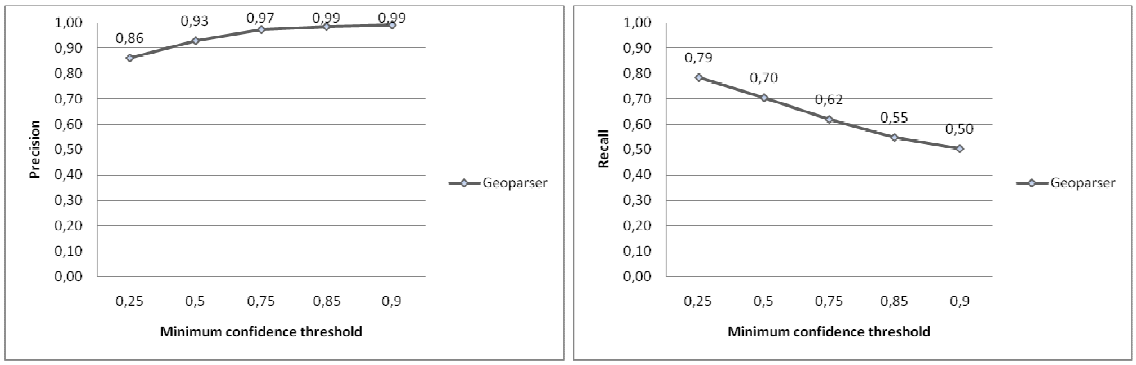


Figure 15 – Measured precision and recall by cross-validation

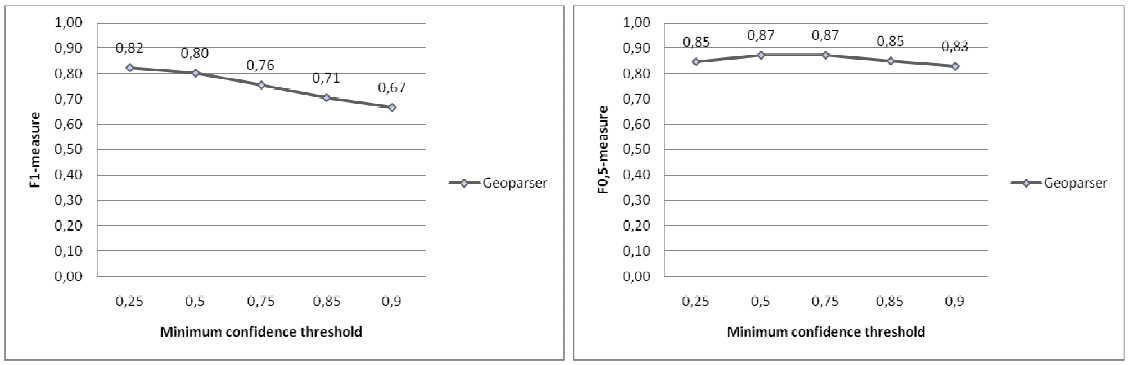


Figure 16 – F_1-measure and $F_{0,5}$-measure results by cross-validation

3.4.2. Comparative evaluation of the Geoparser

To compare the results of the Geoparser against alternative solutions we have chosen to evaluate and compare the geoparsing results of our solution against those obtained by using the the Yahoo! Placemaker geoparser. This comparison was only possible to perform for places, since Placemaker does not recognize time periods or dates.

For this comparison, the evaluation collection was geoparsed by both systems and the results compared against those off the manual annotations. Also in this evaluation, the values of precision, recall, F_1-measure and $F_{0,5}$-measure where calculated.

Analysis of the results shows that the Geoparser generally performed better on any confidence level than Placemaker. F-measure and recall values where always higher for the Geoparser, and in precision the Geoparser was only slightly lower on very low confidence levels of 0,25.

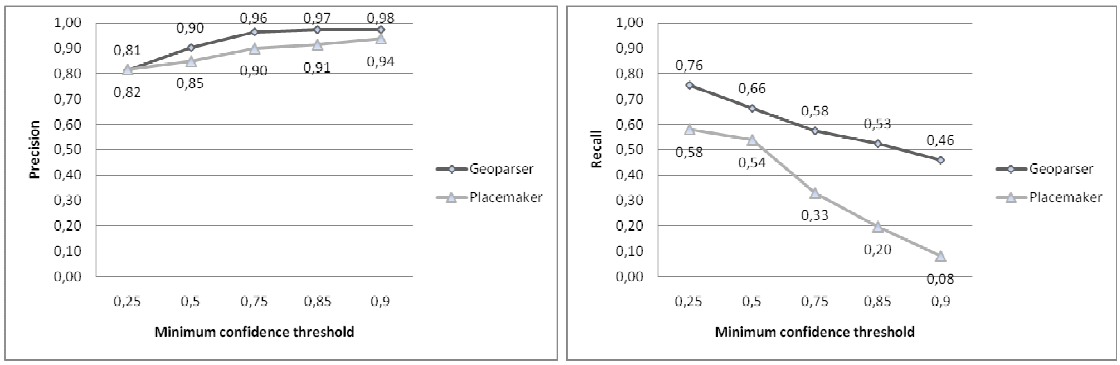


Figure 17 – Precision and recall comparison between the Geoparser and the Yahoo! Placemaker

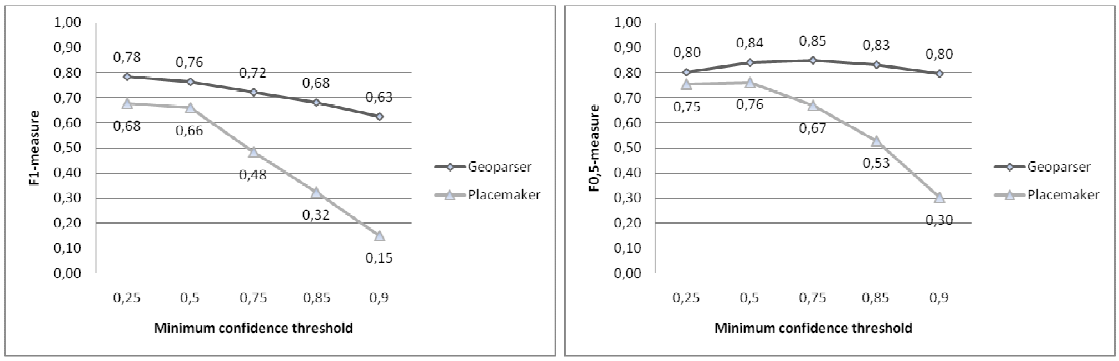


Figure 18 – F-measure comparison between the Geoparser and the Yahoo! Placemaker

3.4.3. Evaluation for time periods

Evaluation of the geoparsing results for time periods was performed by comparing the results of the Geoparser on the evaluation collection with the annotations done manually.

The evaluation collection turned out to be poor for this purpose, however. It contained only a small amount of named time periods, and the great majority of references to time periods where dates. For this reason, we measured precision, recall and F-measure at a minimum confidence of 0,9 (Table 1).

Confidence level	Precision	Recall	F_1-measure	$F_{0,5}$-measure
0,90	0,975	0,91	0,941	0,916

Table 1 – Precision and recall evaluation result for time periods

Both precision and recall were high. However due to the limitations of the evaluation collection, we consider the obtained values to be just an indicative.

4. Conclusions

This document described the specification, design, implementation, usage and evaluation of two services developed to support other geographical and time based services in Europeana. It is our belief that the developed and delivered systems are well implemented and satisfy the corresponding requirements.

Next steps should now consist in its evaluation by Europeana, from which we expect to have to revise small details (for which the IST will be available to assist until the end of the project).

Meanwhile, it was proposed to provide a Search/Retrieval via URL (SRU)[15] interface for the Gazetteer. Since we believe this to be a straightforward implementation, it was taken into consideration for an extra possible delivery of the Gazetteer's software until the end of the project (where we also might include other improvements suggested after Europeana's evaluation).

A pending issue is a suggested extension to the Geoparser to include a batch mode for processing records, in order to accommodate processing entire datasets in a single execution. This should be taken into account in a future development of the system.

[15] http://www.loc.gov/standards/sru/