# GeoKnow – Making the Web an Exploratory for Geospatial Knowledge

**Project Number:** 318159  
**Start Date of Project:** 01/12/2012  
**Duration:** 36 months

## Deliverable 3.2.2: Fusing of Geospatial Metadata

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due Date of Deliverable</td>
<td>Month 23, 31/10/2014</td>
</tr>
<tr>
<td>Actual Submission Date</td>
<td>Month 24, 18/11/2014</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP 3, Spatial Knowledge Aggregation, Fusing &amp; Quality Assessment</td>
</tr>
<tr>
<td>Task</td>
<td>T3.2</td>
</tr>
<tr>
<td>Type</td>
<td>Report</td>
</tr>
<tr>
<td>Approval Status</td>
<td>Final</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Number of Pages</td>
<td>47</td>
</tr>
<tr>
<td>Filename</td>
<td>D3.2.2_Fusing_of_geospatial_metadata.pdf</td>
</tr>
</tbody>
</table>

**Abstract:** This document describes FAGI-gis, the software implemented in GeoKnow for spatial knowledge fusing. The document focuses on the scalability improvements and the functionality additions that were performed on FAGI-gis during the second year of the project. Also, it describes the benchmarking results that quantify the efficiency improvements compared to the first year’s version of the software.

The information in this document reflects only the author’s views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided “as is” without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.

---

Project funded by the European Commission within the Seventh Framework Programme (2007 – 2013)
History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>08/10/2014</td>
<td>First version and section outline</td>
<td>Giorgos Giannopoulos</td>
</tr>
<tr>
<td>0.2</td>
<td>14/10/2014</td>
<td>Description of spatial metadata fusion</td>
<td>Giorgos Giannopoulos</td>
</tr>
<tr>
<td>0.3</td>
<td>17/10/2014</td>
<td>Description of software versions</td>
<td>Nick Vitsas, Giorgos Giannopoulos</td>
</tr>
<tr>
<td>0.4</td>
<td>23/10/2014</td>
<td>Description of benchmark and evaluation</td>
<td>Giorgos Giannopoulos, Nikos Karagiannakis</td>
</tr>
<tr>
<td>0.5</td>
<td>24/10/2014</td>
<td>Description of optimizations</td>
<td>Nick Vitsas</td>
</tr>
<tr>
<td>0.6</td>
<td>24/10/2014</td>
<td>First complete draft</td>
<td>Giorgos Giannopoulos</td>
</tr>
<tr>
<td>0.7</td>
<td>31/10/2014</td>
<td>Additions on optimizations and evaluation sections</td>
<td>Nikos Karagiannakis, Nick Vitsas</td>
</tr>
<tr>
<td>0.8</td>
<td>03/10/2014</td>
<td>Version prepared for the project’s peer-review</td>
<td>Dimitrios Skoutas, Spiros Athanasiou</td>
</tr>
<tr>
<td>0.9</td>
<td>14/11/2014</td>
<td>Additions and improvements based on internal review comments</td>
<td>Giorgos Giannopoulos</td>
</tr>
<tr>
<td>1.0</td>
<td>18/11/2014</td>
<td>Final version</td>
<td>Giorgos Giannopoulos, Spiros Athanasiou</td>
</tr>
</tbody>
</table>

Author List

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Name</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATHENA</td>
<td>Spiros Athanasiou</td>
<td><a href="mailto:spathan@imis.athena-innovation.gr">spathan@imis.athena-innovation.gr</a></td>
</tr>
<tr>
<td>ATHENA</td>
<td>Giorgos Giannopoulos</td>
<td><a href="mailto:giann@imis.athena-innovation.gr">giann@imis.athena-innovation.gr</a></td>
</tr>
<tr>
<td>ATHENA</td>
<td>Nikos Karagiannakis</td>
<td><a href="mailto:nkaragiannakis@imis.athena-innovation.gr">nkaragiannakis@imis.athena-innovation.gr</a></td>
</tr>
<tr>
<td>ATHENA</td>
<td>Dimitris Skoutas</td>
<td><a href="mailto:dskoutas@imis.athena-innovation.gr">dskoutas@imis.athena-innovation.gr</a></td>
</tr>
<tr>
<td>ATHENA</td>
<td>Nick Vitsas</td>
<td><a href="mailto:nickvitsas@imis.athena-innovation.gr">nickvitsas@imis.athena-innovation.gr</a></td>
</tr>
</tbody>
</table>
Executive Summary

This document describes the extensions and enhancements that were performed on FAGI-gis to support the efficient fusion of geospatial metadata, in addition to several other improvements performed during the second year of the project to improve the scalability and usability of the software.

FAGI-gis is a software, implemented within GeoKnow, for fusing geospatial RDF data. It receives as input two datasets and a set of links that interlink entities between the datasets and produces a new dataset where each pair of linked entities is fused into a single entity. The fusion is performed for each pair of matched properties between two linked entities, according to a selected fusion action, and considers both spatial and non-spatial properties (metadata).

In this deliverable we present the improvements that were performed on FAGI-gis. First, we describe how non-spatial metadata are mapped between linked entities and we present the extended set of fusion actions that are supported both on spatial and non-spatial properties. Then, we describe the optimizations we implemented on FAGI-gis that significantly increased its efficiency and scalability. Finally, we evaluate the efficiency of FAGI-gis and compare it to the first version of the software.

The software can be used in two modes of operation:

(a) a batch processing mode, which takes as input a configuration file, specifying the two datasets to be fused, the links that have been already identified among the entities in those datasets, as well as parameters of the fusion action(s) to be applied, and fuses large numbers of linked entities according to these specified parameters

(b) a user-interactive mode, via a Web user interface, where the linked entities are visualized on a map and the user can filter them and perform separate fusion actions on the level of their properties.

The layout of the document is the following.

In Chapter 1, we provide a short description of the FAGI framework for Fusion and Aggregation for Geospatial Information.

In Chapter 2, we describe the metadata fusion functionality we implemented that includes (a) fusion of non-geospatial metadata, (b) semi-automatic mapping of properties between linked entities and (c) extended set of fusion actions, such as concatenation and m-to-n properties fusion.

In Chapter 3, we present the optimizations we performed on FAGI-gis that increase its efficiency by orders of magnitude and describe the two versions of the software for batch processing and for interactive, online processing.

In Chapter 4, we present the evaluation of FAGI-gis with respect to efficiency and compare its runtimes with the previous version of the software, demonstrating the significant increase in performance.

1 Terms “property” and “metadata” will be used interchangeably within this document.
# Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRS</td>
<td>Coordinate Reference System</td>
</tr>
<tr>
<td>DBMS</td>
<td>DataBase Management System</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographical Information Systems</td>
</tr>
<tr>
<td>GML</td>
<td>Geography Markup Language</td>
</tr>
<tr>
<td>OGC</td>
<td>Open Geospatial Consortium (<a href="http://www.opengeospatial.org/">http://www.opengeospatial.org/</a>)</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SPARQL</td>
<td>SPARQL Protocol and RDF Query Language</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>WGS84</td>
<td>World Geodetic System (EPSG:4326)</td>
</tr>
<tr>
<td>WKT</td>
<td>Well Known Text (as defined by ISO 19125)</td>
</tr>
</tbody>
</table>
Table of Contents

1. INTRODUCTION .................................................................................................................. 8
   1.1 CHALLENGES .................................................................................................................. 8
   1.2 FRAMEWORK ARCHITECTURE ....................................................................................... 10
   1.3 PROGRESS OF THE WORK ............................................................................................. 11
2. FUSING OF GEOSPATIAL METADATA ............................................................................. 13
   2.1 PROPERTY MATCHING ..................................................................................................... 13
      2.1.1 Semantic and lexical similarity ............................................................................... 13
      2.1.2 Text similarity .......................................................................................................... 15
      2.1.3 Type similarity .......................................................................................................... 16
      2.1.4 Combined property similarity ................................................................................. 17
   2.2 FUSION ACTIONS ............................................................................................................ 17
      2.2.1 Fusion of m-to-n matched properties ..................................................................... 17
      2.2.2 Fusion of property chains ....................................................................................... 18
      2.2.3 Extended fusion actions ......................................................................................... 18
3. EXTENDED FAGI-GIS ......................................................................................................... 21
   3.1 OVERVIEW OF VERSION 1 .............................................................................................. 21
   3.2 OPTIMIZATIONS ............................................................................................................. 22
      3.2.1 Virtuoso communication ........................................................................................... 22
      3.2.2 Dataset joins for metadata fetching ......................................................................... 22
      3.2.3 PostGIS optimizations ........................................................................................... 23
   3.3 FAGI-GIS VERSION 2 ..................................................................................................... 24
      3.3.1 Software description ................................................................................................ 24
         3.3.1.1 Implementation information .............................................................................. 24
         3.3.1.2 Components ....................................................................................................... 25
         3.3.1.3 Supported data sources and formats ................................................................... 27
      3.3.2 Installation/Building instructions .............................................................................. 28
      3.3.3 FAGI-gis for batch processing ............................................................................... 28
      3.3.4 FAGI-gis for map based, user interactive fusion .................................................... 29
      3.3.5 Licensing ................................................................................................................ 32
4. BENCHMARKING ................................................................................................................ 33
   4.1 DATASET .......................................................................................................................... 33
   4.2 BENCHMARK DESCRIPTION .......................................................................................... 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3 VERSION COMPARISON</td>
<td>35</td>
</tr>
<tr>
<td>5. CONCLUSIONS</td>
<td>38</td>
</tr>
<tr>
<td>6. REFERENCES</td>
<td>39</td>
</tr>
<tr>
<td>7. APPENDIX</td>
<td>40</td>
</tr>
<tr>
<td>7.1 WIKIMAPIA XML SAMPLE</td>
<td>40</td>
</tr>
<tr>
<td>7.2 WIKIMAPIA SAMPLE RDF FILE</td>
<td>46</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Fusion Framework Architecture ................................................................. 11
Figure 2: Property namings ..................................................................................... 14
Figure 3: Semantic relations in WordNet ................................................................. 14
Figure 4: Semantic similarity calculation ............................................................... 15
Figure 5: String similarity calculation ................................................................ 16
Figure 6: 2-to-1 property matching ...................................................................... 18
Figure 7: Property chains ..................................................................................... 18
Figure 8: Desktop GUI of FAGI-gis v1.0 ................................................................. 21
Figure 9: FAGI-gis major components ................................................................. 27
Figure 10: Configuration panels .......................................................................... 30
Figure 11: Link filtering panel ............................................................................. 30
Figure 12: Geospatial RDF entities on the map .................................................... 31
Figure 13: Property matching panel .................................................................... 32
Figure 14: Fusion panel ....................................................................................... 32

List of Tables

Table 1: FAGI fusion actions ............................................................................. 19
Table 2: Feature matrix for FAGI-gis versions 1 and 2 ..................................... 24
Table 3: Evaluation parameters .......................................................................... 34
Table 4: FAGI version comparison on 1000 links (chain=2) ......................... 35
Table 5: FAGI version comparison on 10000 links (chain=2) ....................... 36
Table 6: FAGI version comparison on 100000 links (chain=2) .................... 36
Table 7: FAGI version comparison on 1000 links (chain=1) ......................... 37
Table 8: FAGI version comparison on 10000 links (chain=1) ....................... 37
Table 9: FAGI version comparison on 100000 links (chain=1) .................... 37
1. Introduction

Fusion is the process of merging the descriptions (attributes, metadata, properties) of two or more resources that correspond to the same real world entity, to produce a richer, cleaner and universal description for the respective entity. Fusion constitutes the final part of the data integration process, which usually consists of three steps: schema integration, duplicate detection (interlinking) and fusion [BN08]. Although there have been several works on schema integration/mapping and interlinking on RDF data, fusion is still a field of ongoing research. Current approaches adopt state of the art, generic fusion techniques, without specializing them based on the characteristics of the RDF data model and schemata. Besides, none of the existing works propose methodologies for automating the fusion process on large RDF datasets.

In the frame of GeoKnow, we are working on filling this gap, by developing FAGI (Fusion and Aggregation of Geospatial Information), an integrated software for fusing RDF datasets, with emphasis on their geospatial aspects. FAGI is designed to incorporate all stages of the fusion process: vocabulary alignment, geospatial processing and transformations, property mapping, manual and semi-automatic fusion actions, as well as advanced facilities such as link discovery and rejection and topological relations exploitation.

In the following, we describe the main challenges with respect to fusing geospatial RDF data and the architecture of FAGI, as presented in deliverable D3.2.1 [GeoKnowD32]. Finally, we shortly describe which features-subcomponents of FAGI have been implemented during the first and the second year of the project.

1.1 Challenges

The problem of fusing geospatial RDF datasets differentiates, to some extent, from the individual settings of fusing geospatial data, or fusing RDF data. This results from the fact that both semantic and spatial knowledge is available and needs to be taken into account when performing the fusion process. A main difference lies in the following. In the case of RDF data, there is a clear and explicit separation between the two steps of interlinking and fusion: first, matching entities from the two input datasets are identified and linked together, and then their attributes are fused to produce a single entity representation. In the case of geospatial data, matching and fusing are rather intertwined (e.g., identifying control points that are used to drive the fusion of geometries). Thus, when dealing with RDF entity pairs from two compared datasets, along with their geospatial representations, the additional knowledge that these pairs correspond to the same real world object, and thus, their geometries correspond to each other, is a given. Still, the fusion process in this case needs to handle more complex sets of attributes, including strings or arithmetic values in conjunction with geometries of entities, which in certain scenarios requires transformations and/or combinations of existing geometries. Consequently, a comprehensive set of fusion strategies and actions needs to consider the following:

- Define effective scoring functions that meaningfully combine semantic properties and geospatial features of the entities. These scoring functions should utilize:
  - Available mappings on the schemas that characterize the interlinked entities.
  - Refined string similarity functions that can decompose the names and descriptions of the entities and apply individually fitting similarity metrics. For example, given the names "Joe's diner" and "8th elementary school of Athens" of a restaurant and a school respectively, numeric values ("8th") as well as special words ("diner", "school") should be considered when evaluating the similarity.
“elementary”) should be treated separately when calculating the similarity between interlinked entities.

- Select the most accurate geometry according to some criterion. For example, in some cases the most accurate geometry might be considered the most complex one (i.e. the one consisting of the most points), or the most recent one. In other cases, the most accurate geometry is given explicitly by the quality indicators of the datasets, if available.

- Select a combination of geometries that produces a richer geometry. This strategy may involve several actions/sub-strategies. We present some indicative examples:
  
  o Keep partial geometric properties from both geometries, to produce a new geometry of higher quality. For example, in an interlinked pair of geometries, if geometry A is a point with high quality coordinates and the other geometry is polygon B with less qualitative coordinates, then the fused geometry should be the polygon B, shifted so that its new centroid becomes the point A. In another case, where two points are interlinked, the final geometry could be produced by averaging the coordinates of the two points.
  
  o Select and keep both geometries separately. In this case, the geometries could indeed correspond to the same real world entity but represent slightly different aspects of this entity. For example, in the case of two interlinked entities, geometry A may represent the building of a school, while geometry B might represent both the school building and the school yard. In this case, a decision on how to handle these two entities should be made:
    
    ▪ Keep both geometries for the entity, exactly as they are in separate properties.
    ▪ Create a new, composite geometry object and use it as the geometry of the fused entity.
    ▪ Create a new entity and keep two entities corresponding to the semantics of the two separate geometries as result of the fusion.

- Reject the link between two entities (cleansing). In case the matching score of two interlinked geometries is too low, there might be the case that these two geometries have been falsely interlinked.

- Produce new links between entities. Even if there are no links between two entities, after examining clusters/neighborhoods of interlinked entities, and by applying traditional geospatial conflation techniques (space transformation, statistics on spatial properties, etc), new links between entities of the two datasets may be recognized. This makes the problem significantly more challenging, since it increases both the complexity of specifying the rules for the fusion process and its computational complexity.

- A further action that can be part of the fusion process is, apart from fusing the attributes of the matched entities, to produce an overall fused dataset that contains entities from both initial datasets. This entails a process similar to the one described above: by searching on neighborhoods of interlinked entities, and concluding that an entity of dataset B does not exist in dataset A, the entity can be created in A too, and be interlinked with the respective entity of dataset B.

- Produce new semantic, topological relations. Based on analysis of the geospatial relations between interlinked entities some basic geospatial relations between entities could be calculated and stored as semantic metadata of the entities. For example, properties that capture, for each entity, the orientation (e.g. north from) of its top-k closest neighbors could be created.

Another specificity of our problem setting, compared to previous works (see [GeoKnowD32]), is the fact that, due to the already existing links between entities, automatic conflation techniques that process the whole space of the datasets cannot be straightforwardly applied. On the other hand, these interlinked
entities can be utilized as control point pairs for partitioning the space and working on each partition separately.

Finally, a great challenge is to implement meaningful learning algorithms that will be able to train on user actions and recommend both fusion scores and preferred fusion strategies. A critical factor for the effectiveness of such algorithms is the proper selection of training features, i.e. features that describe interlinked entities/geometries pairs and correlate them to the user selected fusion strategy. The algorithms should function both offline and online. That is, the algorithms should be able to be trained offline on previous user actions, or online, as the user manually selects a fusion action for a (set of) interlinked entity pairs.

1.2 Framework Architecture

In this section, we describe the framework for fusing geospatial RDF data called FAGI. FAGI aims at providing a comprehensive solution for geospatial enrichment of knowledge bases, based on semantic technologies and open data sources. Its main functionality is to support the entire process of aligning RDF representations of geospatial data, fusing and aggregating geospatial triples through intuitive, step-by-step processes, model-learning for automatic fusion recommendations and map-based UIs. Our focus is to build the FAGI framework based on real world and diverse geospatial RDF datasets, so that we can capture the heterogeneity of data sources and provide an extended processing coverage to several use case scenarios.

Figure 1 presents the architecture of the framework. The basic components are as follows.

- **Storage Component**. This component is essentially an RDF store where data and metadata from all stages of the fusion process are stored and retrieved. This is where the input data are initially stored and the new, processed data, metadata from the Learning Component, as well as the final fused dataset(s) are written.

- **Preprocessing Component**. This is the component that performs an initial processing of the data, so that they are normalized w.r.t. the vocabularies they use (i.e. RDF vocabularies for representation of geospatial features, OGC standards for the encoding of geometries in the form of literals, coordinate reference systems). Also, this component gathers and organizes quality metadata of the datasets, which either exist within the datasets, or are manually input by the user.

- **Geospatial Processing Component**. This component is assigned with the whole process of geospatial indexing, topological relations calculation and geospatial transformations of the geometric features of the entities. Geospatial indexing is necessary for being able to perform efficiently the remaining two processing categories. The functionality of the other two modules is self-evident: Geospatial calculations are required in order for geometric similarity scores for fusion are produced, while geospatial transformation operations are necessary for the actual fusing process, when a geometry or even a whole set of geometries need to be transformed to produce a final, qualitative fused geometry.

- **Fusion Component**. This component performs the core fusion functionality of the framework. It includes property mapping for both spatial and non-spatial properties, calculation of similarity scores for geospatial features of entities and recommendation of fusion strategies. Furthermore, based on the calculated similarity scores (both on spatial and non-spatial properties of the interlinked entities), as well as on geospatial calculations on the neighborhoods of the interlinked entities, it suggests creation of new links between entities or rejection of existing links, as false positives.

- **User Interface Component**. This component supports the interaction of the end user with the rest, underlying components. Through this component, the user can configure and guide the preprocessing of the input datasets, select and apply fusion strategies on (parts of) the
datasets and provide feedback either by validating the fusion results or by validating the new
link creation or the link rejection recommendations.

- **Learning Component.** This component handles the training of machine learning models for
  automatic suggestion of fusion strategies for groups of triples, based on the individual
  characteristics of the interlinked geospatial entities, as well as the history of user actions w.r.t.
  selecting and validating fusion actions.

![Fusion Framework Architecture](image_url)

**Figure 1: Fusion Framework Architecture**

### 1.3 Progress of the Work

Each component of Figure 1 is coloured based on the framework’s development phase they correspond
too. During the first year of the project (green and green-orange coloured components), we focused on
creating the infrastructure for fusing different representations of geometry in RDF. This involves both
aligning the vocabularies, the geometry serializations and the coordinate reference system (CRS) of
geometries (Vocabulary Alignment Module) and implementing the infrastructure for indexing, comparing
and transforming geometries to be fused (Geospatial Processing Component). Of course, this development
iteration included also the development of parts of the general use Modules, such as the Link Rejection
Module, the Selection of Fusion Actions Interfaces, the Storage Component and the Fusion Actions
component, that correspond to the core components of the first iteration.

During the second year of the project (orange and green-orange coloured components), we revisited
the Geospatial processing Component, optimizing the whole workflow of obtaining and processing
metadata for linked entities. Specifically, we optimized the processes of querying the input SPARQL
endpoints, the processes for gradually obtaining, at each step, only the necessary metadata for fusion and
the actual fusion processes. Further, we replaced the desktop GUI with a web-based interface where spatial
entities are visualized on map layers, allowing the user to visually review the spatial relation between linked entities and, thus, facilitating the selection of proper fusion actions. Finally, we implemented the property mapping functionality that facilitates the fusion of non-spatial properties, by recommending the matching properties between two linked entities and we extended the set of available fusion actions.

Our goals for the third year include the constant improvement of the system, in terms of efficiency and scalability, the developments of the mechanism for learning and automatically recommending fusion actions and the implementation advanced fusion facilities (link discovery, relations fusion).
2. Fusing of geospatial metadata

The problem of fusing interlinked geospatial entities can be described as follows: given a set of pairs of linked entities \((e_1, e_2)\), for each pair: (a) identify which property(ies) of \(e_1\) correspond to which property(ies) of \(e_2\), that is describe the same metadata of \(e_1\) and \(e_2\), (b) for each pair of corresponding (matched properties) select a fusion action, namely a combination of the metadata content of the two properties to eventually keep as the content of the fused entity for the specific properties and (c) for each pair of linked entities, create a new set of triples that maintain a single, unified description of the entities with respect both to their resource description and to the final set of properties that are attached to it.

2.1 Property matching

The first step of fusion consists of finding matchings between the properties of the first entity and the ones of the second entity. Practically, this is an easier task for geospatial properties, since the number of geospatial vocabularies that are used to represent geospatial information is rather limited (see [GeoKnowD21], Section 3.1.2). Correspondences between geospatial properties from different vocabularies can be predefined, so that they are automatically matched by the software without any effort from the user. FAGI-gis already supports W3C Basic Geo and GeoSPARQL vocabularies (the latter tends to become a standard in representing geospatial information in RDF data). However, even if there is no provision by the software to automatically match geospatial properties, it is relatively straightforward for the user to manually perform the matching, due to the small number of vocabularies (and thus the candidate property namings) and the particular format of geospatial content (i.e. coordinates).

On the other hand, it is usually the case that RDF entities are characterized by a wealth of non-spatial properties, describing all kinds of metadata of the entities (e.g. name, type, size, year, age, cost, etc.) and relations with other entities. Thus, mapping non-spatial properties between linked entities constitutes the most challenging task in the specific problem. There is a wealth of proposed methods for solving the specific problem, that can be generalized to the problem of schema mapping. These include utilization of linguistic properties of the data, external thesauri, content, structure, statistics, pruning, partitioning and several other methods, that are summarized in [BMR11].

Since our core problem in this work is the actual fusion action (strategy\(^2\)) that will be chosen for a pair of matched properties, that is step (b) in the fusion process, we did not focus our efforts on thoroughly examining and implementing several approaches for automatic schema matching. On the contrary, we implemented a semi-automatic workflow, where the software combines three different techniques to identify potential property matchings, which are presented to the user so that she makes the final selection-validation of the proper matching(s). The implemented property matching techniques are presented next.

2.1.1 Semantic and lexical similarity

With this similarity function, we aim at comparing the distinctive part of the naming of each property, that is, the part of the property that comes after its namespace. For example, consider the following three RDF properties:

\[
\begin{align*}
A. & \quad \text{http://example.org/name} \\
B. & \quad \text{http://example.org/first_name} \\
C. & \quad \text{http://example.org/label}
\end{align*}
\]

\(^2\) The terms action and strategy will be used interchangeably in this deliverable.
Property’s (A) useful naming part is *name* while property’s (B) is *first_name* and property’s (C) is *label*. Our aim is to calculate the conceptual similarity between these specific parts of the properties’ namings.

In order to measure conceptual similarity, we exploit WordNet [Wnet], a lexical database that organizes conceptual-semantic relations between words. WordNet is a directed acyclic graph, where each vertex is a set of synonyms called *synset* and each directed edge denotes that the start node of the edge is a *hyponym* (a more specific synset) of the end node. Hence, two conceptual similarity relations between words are defined this way: (a) Two words have similar meaning (denoted as *similar_meaning*) (b) A word has a more general meaning than another word (denoted as *general_meaning*). To be able to extract such relations for word pairs, we use the Java WordNet Library [JWNL] which implements querying facilities on WordNet.

In the example of Figure 3, we can see how the terms *name* and *label* are interconnected in the WordNet graph: *name* is a generalization of *label* with a distance of two edges between the words.

In our scenario, our initial input is a set of interlinked pairs of entities. Each entity is connected by a set of properties that describe it. Our aim is to produce a semantic score for each pair of properties \((a,b)\). The process we follow is illustrated in Figure 4 and described next.

The first step is to extract the meaningful terms that will represent each property (step i). This includes removing the property namespace and splitting the remaining part in case there are more than one concatenated terms. This results to a set of terms for each property, denoted as *property_terms_set*. In our running example (Figure 2), *property_terms_set*(A)={*name*}, *property_terms_set*(B)={*first*, *name*} and *property_terms_set*(C)={*label*}.
The next step is to compare, for each pair of properties, pairs-wise, all terms that represent them. This comparison is performed through WordNet and JWNL, that return, for each pair of terms, whether they are synonyms or whether one is hyponym of the other and, in the latter case, their distance in terms of graph edges. Using this information, we produce a similarity score for each term pair between the two properties. Indicatively, we have: sim(name, name) = 1, sim(name, first) = 0 and sim(name, label) = 0.5.

At this stage, for each pair of properties, we have calculated all pairwise semantic similarities between the terms that represent them. The third step consists in utilizing these pairwise, term similarities, to extract a total semantic similarity score between the two properties. To do so, we first sum all term similarities and then we normalize the sum with the product of the size of the terms set of the two properties. For example, in the case of properties A and B, we have:

\[
sim(A,B) = \frac{\text{SUM}(\text{sim}(u_i,v_j))}{|\text{property_terms_set}(A)|*|\text{property_terms_set}(B)|} = \frac{0 + 1}{1*2} = 0.5
\]

Finally, each property-to-property similarity score is normalized by dividing with the maximum such score, in the fourth step. This ensures that semantic similarity scores between properties are normalized in the interval \([0,1]\), so that they can properly be combined with other types of similarity scores, as will be described next.

Terminology:
- (a,b): pair of properties
- (u,v): pair of terms
- sim(u,v): similarity between terms u and v
- sim(a,b): similarity between properties a and b
- synonym(u,v): denotes synonymy between u and v
- hyponym(u,v): denote that u is hyponym of v OR v is hyponym of u
- distance(u,v): number of edges between u and v
- property_terms_set(a): the set of terms that correspond to property a

i. Extract meaningful term(s) from the property naming
   Filter out property’s namespace
   Split remaining string to find potential separate terms
   Remove stop words \(\rightarrow\) property_terms_set

ii. For each property pair (a, b)
   For each term pair \((u_i,v_j)\) between property_terms_set(a) and property_terms_set(b)
   Query WordNet about the relation of \(u_i, v_j\)
   If synonym\((u_i,v_j)\) then \(\text{sim}(u_i,v_j) = 1\)
   If (hyponym\((u_i,v_j)\) AND distance\((u_i,v_j) < 4\)) then \(\text{sim}(u_i,v_j) = 1 - (\text{distance}(u_i,v_j)/4)\)
   If (hyponym\((u_i,v_j)\) AND distance\((u_i,v_j) \geq 4\)) then \(\text{sim}(u_i,v_j) = 0\)

iii. For each property pair (a, b)
   Calculate the sum of all pairwise similarities \(\text{sim}(u_i,v_j) \rightarrow \text{SUM}(\text{sim}(u_i,v_j))\)
   \(\text{sim}(a,b) = \frac{\text{SUM}(\text{sim}(u_i,v_j))}{|\text{property_terms_set}(a)|*|\text{property_terms_set}(b)|}\)

iv. Normalize all \(\text{sim}(a,b)\) by the maximum \(\text{sim}(a,b)\)

Figure 4: Semantic similarity calculation

2.1.2 Text similarity

Apart from conceptual similarity, we need also to take into account the string similarity between the terms representing two properties. The process we follow is described in Figure 5. The first step, which aims at finding a representative set of terms for each property, is almost identical to the one in Figure 4.
The only difference is that at the end of the step, the remaining terms are stemmed. Next, we apply the Jaccard similarity on the representative term sets of the two properties. Finally, we normalize all pairwise property similarities similarly to Figure 4.

Terminology:
- \((a,b)\): pair of properties
- \((u,v)\): pair of terms
- \(\sim(u,v)\): similarity between terms \(u\) and \(v\)
- \(\sim(a,b)\): similarity between properties \(a\) and \(b\)
- \(\text{synonym}(u,v)\): denotes synonymy between \(u\) and \(v\)
- \(\text{hyponym}(u,v)\): denote that \(u\) is hyponym of \(v\) OR \(v\) is hyponym of \(u\)
- \(\text{distance}(u,v)\): number of edges between \(u\) and \(v\)
- \(\text{property_terms_set}(a)\): the set of terms that correspond to property \(a\)

i. Extract meaningful term(s) from the property naming
   Filter out property's namespace
   Split remaining string to find potential separate terms
   Remove stop words
   Stem remaining terms\(\rightarrow\) property_terms_set

ii. For each property pair \((a, b)\)
   \[ \sim(a,b) = \text{Jaccard}(\text{property_terms_set}(a), \text{property_terms_set}(b)) \]

iii. Normalize all \(\sim(a_x,b_x)\) by the maximum \(\sim(a_x,b_x)\)

Figure 5: String similarity calculation

2.1.3 Type similarity

A final comparison between two entities considers the data type that characterizes their literal. The incentive behind this similarity metric is that, if two properties match each other (that is describe the same attribute of two linked entities) their objects should probably be of the same data type.

In order to compare data types of property objects (literals in this case), we define a set of rules in the form of regular expressions, which are matched against each property's literal. If two properties match the same regular expression, then their similarity equals 1; else it equals 0. The regular expressions we define consider the following data types:

- **Single word.** This rule examines whether the literal consists of a single word. Indicatively, the regular expression for this rule is presented below.
  \[ \w \]

- **Text.** This rule examines whether the literal consists of a phrase. Indicatively, the regular expression for this rule is presented below. The expression demands that a word is found that is followed by at least a whitespace and another word.
  \[ \w(\s\w)^+ \]

- **Date.** This rule examines whether the literal consists of a date. Indicatively, the regular expression for this rule is presented below. The expression matches dates of the form \"MM/DD/YYYY\", \"MM-DD-YYYY\", \"DD/MM/YYYY\", \"DD-MM-YYYY\".
  \[ [0-9]{2}(/|-)[0-9]{2}(/|-)[0-9]{4} \]
• **Integer**. This rule examines whether the literal consists of an integer value. The regular expression for this rule is presented below.
\d+

• **Real numbers**. This rule examines whether the literal consists of a real number value. The regular expression for this rule is presented below.
\[0-9]+\/[0-9]+\]

• **Coordinates**. This rule examines whether the literal consists of a date. Indicatively, the regular expression for this rule is presented below. The expression matches dates of the form “lat long”.
\([-+]?\d{1,2}(\.[.]\d+)?,\s*([-+]?\d{1,3}(\.[.]\d+))?\]

### 2.1.4 Combined property similarity

In order to produce a final score for each property pair, we consider an equally weighted sum of the above three similarity scores:

\[
sim_{\text{total}}(a,b) = \frac{\left(\text{sim}_{\text{semantic}}(a,b) + \text{sim}_{\text{text}}(a,b) + \text{sim}_{\text{type}}(a,b)\right)}{3}
\]

Since each of these scores are normalized in the interval \([0,1]\), the final score is also normalized in the same interval.

If the final similarity of a pair of properties scores above 0, then this property pair is recommended to the user as a potential matching pair. The user is eventually presented with the whole set of potential matchings and validates or rejects them.

### 2.2 Fusion actions

The first version of FAGI-gis focused on the fusion of geospatial properties. Namely, we implemented a set of fusion strategies that handle geometries and are presented in the following list:

• keep target,
• keep source,
• keep both,
• keep the geometry containing the most points (most complex geometry),
• keep the geometry containing the most points (most complex geometry) and shift it so that it has as centroid the centroid of the other geometry
• keep the average of two points
• keep one of two geometries, scaling it to some selected factor

Next, we present the extensions we implemented during the second year with respect to the fusion functionality on both spatial and non-spatial metadata.

#### 2.2.1 Fusion of m-to-n matched properties

An important facility that is incorporated into FAGI is the option to fuse m-to-n properties with a single fusion action. This functionality is useful in cases where an attribute of an entity is represented by more than one RDF properties. For example, the full name of a person might be represented by two separate properties: *first_name* and *last_name*, as presented in Figure 6: Dataset 1 uses two properties to store a person’s name, while Dataset2 only one. In this case, a specific fusion action needs to be defined that will be able to handle the two properties of Dataset1 as a single property. We implement this action by allowing the user to handle the content of the two properties in two ways: she can keep the content of one of the properties or a concatenation of the content of the properties to represent the total of properties.
from Dataset1. Then, she proceeds to fusion according to the standard fusion actions supported by the system. We denote this fusion action facility as *multi-fusion*.

**Figure 6: 2-to-1 property matching**

### 2.2.2 Fusion of property chains

There are cases when a set of properties of an RDF entity are closely related, in the sense that they describe several attributes of a more general concept that characterizes the entity. For example, the address of a person is an attribute of the person that consists of several sub-attributes that contain the actual information. In our example (Figure 7), node “Address1” functions as blank node, since it is just used as an intermediate node to gather together properties *street, number* and *zip code*, on Dataset1. On the other hand, Dataset 2 organizes the same information in a flat manner, connecting the respective properties directly with the entity. If we ignore this fact, and try to match properties between the two datasets considering only first level properties of the linked entities, then we will miss the matchings between the second level properties of Dataset1 and the first level properties of Dataset2. To overcome this issue, for each entity, we consider property chains (of maximum length 3) that connect the entity with some literal. Each property chain is represented as a sequence of concatenated property names that constitute the chain and is assigned as object the object-literal of the last property. This way, the user can match properties of various depths (with maximum depth of three properties) with respect to two linked entities. We denote this fusion action facility as *chain-fusion*.

**Figure 7: Property chains**

### 2.2.3 Extended fusion actions

During the second year of the project, the initial set of fusion actions was extended, providing more options w.r.t. fusing facilities but, most importantly, allowing the fusion of both spatial and non-spatial
properties in an integrated way. To this end, we extended the first three fusion actions (keep target, keep source, keep both) in order to cover both spatial and non-spatial properties.

Further, we implemented a new fusion action, concatenate, that has similar functionality to keep both. However, instead of keeping the two properties (with the respective literals) as they are, it keeps only one property and concatenates the two separate literals to be used as the object of the retained property. Since this property regards both spatial and non-spatial metadata, its functionality slightly differs between these two cases. In the case of two geospatial properties, the concatenation process results into creating a multipolygon from the two respective geometries; in any other case, the strings contained in the literals are simply concatenated so that the initial values are preserved exactly.

As far as non-spatial metadata are concerned, we defined one more fusion action that specializes on textual metadata: keep most complete, examines whether the literal of one property contains most words from the literal of the other property; in this case, the former literal is kept as the fusion result. To this end, we split the text contained in each literal into words and then apply stemming and stop-word removal on them, in order to avoid missing word matches due to their endings and to avoid comparing words of no use (e.g. articles and prepositions) respectively. This way, we essentially create, for a pair of properties \((A, B)\), their representative set of terms \(property\_terms\_set\) as described in Figure 5. We then compute the ratio of words of property \(B\) that are contained in \(property\_terms\_set(A)\) to the size of \(property\_terms\_set(B)\) (and vice versa). If this ratio exceeds a threshold, then the literal of property \(A\) is eventually kept.

The final set of fusion actions, including the specialized handling of m-to-n matchings and property chains, is presented in Table 1:

\begin{table}[h]
\begin{center}
\begin{tabular}{|l|l|l|}
\hline
\textbf{Action} & \textbf{Metadata type} & \textbf{Functionality} \\
\hline
keep target & both & Keeps the value of the first property. \\
keep source & both & Keeps the value of the first property. \\
keep both & both & Keeps both properties separately. \\
concatenate & both & Keeps only one property that contains the complete information from both initial properties. \\
keep the most complex geometry & spatial & Keeps the geometry that consists of the most points. \\
keep most complete & Non-spatial & Keeps the literal of the property, if it contains a large part of the literal of the other property. \\
keep the most complex geometry and shift it & spatial & Keeps the geometry that consists of the most points and shifts it so that it has as centroid the centroid of the other geometry. \\
keep the average of two points & spatial & Keeps a new point geometry that is calculated by the average of the two initial points. \\
keep one of two geometries, scaling it to some selected factor & spatial & Keeps one the geometries and rescales it. \\
multi-fusion & non-spatial & Allows the handling of multiple properties describing subattributes of a more general attribute of an entity to be handled as a singular property w.r.t. fusion. \\
chain-fusion & non-spatial & Allows considering for fusion properties that describe an entity but are not directly connected with the entity. \\
\hline
\end{tabular}
\end{center}
\end{table}
We note that multi-fusion and chain-fusion properties consider only non-spatial properties, since geospatial properties are specifically handled by FAGI: the most popular vocabularies (Basic Geo and GeoSPARQL) can be identified through encoded matching rules, so these two fusion facilities are inherently supported without the need to be applied by the user.

Finally, we note that this version of the software allows the user to separately select, apart from the content (literal) of the eventually kept geometry, also the name of the fused property; the user can select to keep the name of either of the initial properties or define a new one.
3. Extended FAGI-gis

In this section we present FAGI-gis, the software for fusing interlinked geospatial RDF entities. First, we give a brief overview of the previous version of the software. Then, we describe the optimizations we implemented to increase the efficiency and scalability of the software. Finally, we describe in detail the current version of the software, analysing its main components and demonstrating, step by step, the usage of the web version.

3.1 Overview of Version 1

The first version of FAGI-gis provided the infrastructure for (a) indexing geospatial features (b) computing similarities between geometries and (c) applying geometry fusion strategies, either by combining or by transforming geometries, implementing an initial set of functions that can be straightforwardly extended. That is, during the first year of development, we focused on implementing the infrastructure for efficiently fusing the geospatial metadata of linked RDF entities.

Since complex geospatial operations might become prohibitively time consuming, we based this initial implementation of the software on a PostgreSQL/PostGIS database, which provides efficient geospatial indexing and a wide range of efficient calculation and transformation functions. The implemented software provided a graphical user interface for executing the above functionality in a step by step process, as described next.

![Figure 8: Desktop GUI of FAGI-gis v1.0](image)

At first, the user is required to provide connection information for the PostGIS database and, then, input SPARQL endpoints and graph URIs for the source and target datasets. Then, the geospatial features are extracted from the triples and stored/indexed in PostgreSQL/PostGIS tables. At the final step of the process, the user loads the links between the interlinked entities of the two datasets and calls the scoring function, to assess similarities between linked entities. Then, according to the calculated scores, the user can select which entity pairs to fuse and which fusion strategy to follow. The first version implemented...
seven fusion actions (considering geospatial properties of the entities), as well as the underlying infrastructure, so that several more geospatial fusion strategies could be implemented through the proper interfaces. Non-spatial properties were treated trivially by keeping metadata from both entities. The geospatial fusion actions utilize the functionality of PostgreSQL/PostGIS, by calling the respective functions and working on indexed geospatial features. The final outcome is a set of fused geometries, for the respective linked entity pairs, that are written back to the Virtuoso RDF store.

3.2 Optimizations

The entire fusion process requires large moving of data between components of the program. Those include a relational database (Postgres/PostGIS) operating on tables, a quad store (Virtuoso) operating on RDF graphs and the program (FAGI-gis) operating on in-memory data structures. In our first approach, FAGI was placed in the centre of all data handling. This introduced the major bottleneck of constantly transferring and converting data between heterogeneous representations that each part of the program could understand. A major design shift was required in order to achieve better performance. In particular, we have revised the process to store the program input (namely, the provided links) in various intermediate forms that would benefit each part of the program individually. We used a relational table to store links in PostGIS, an RDF graph to store them in Virtuoso and Java data structures in FAGI. This approach did not only minimize the data transfers but also allowed the individual components to work more efficiently on the data. When large data transfers where required, we used bulk insert methods for both Postgres and Virtuoso stores. Those included loading through files and other special JDBC API methods. This allowed FAGI to take a controller role, where it manipulates data in order to facilitate other components in performing data fusion.

3.2.1 Virtuoso communication

Data analysis and manipulation is done on local graphs that contain data from the linked entities. This is done to improve efficiency but requires an expensive pre-process step of loading them into Virtuoso. Therefore, values to be inserted are stored in a file to enable fast bulk insertion through Virtuoso specific functions. Since we are working locally, using files is by far the fastest way of importing large amounts of data into Virtuoso.

3.2.2 Dataset joins for metadata fetching

In our setting, where fusion is performed on pairs of interlinked geospatial RDF entities, the input comprises the set of links connecting the respective entities, as well as the two RDF datasets where the linked entities are fully described (through several spatial and non-spatial properties). In the most general case, the two input datasets are on remote SPARQL endpoints. Thus it would be very time consuming to perform all necessary fusion tasks on them: querying the remote endpoints and constantly transferring data through the network. Because of this, we need to pre-fetch the minimum required information from the input datasets that allows us to fuse the interlinked entities. Essentially, we need to download all RDF properties that have as subject one of the interlinked entities. Further, since there are cases where an entity is described by a sequence of properties, we also need to fetch property sequences (chains) of some depth (e.g. of depth of three properties starting from the entity).

In order to perform this fetching effectively, FAGI-gis constructs locally an RDF graph on Virtuoso, that is composed of the triples contained in the links file. Then, it performs a join SPARQL query between the remote SPARQL endpoints and the local graph, in order to fetch all the metadata of the linked entities. The query is presented below, where <http://localhost:8890/DAV/links> corresponds to the local links graph and <http://localhost:8890/DAV/uni2> to the remote endpoint. The optional clauses retrieve possible properties of depth more than one with respect to the entities of interest.
Since no fusion action can be performed before analyzing the data, we perform a single query that fetches all non-spatial metadata from the remote graph and inserts them in the local quad store. The query joins triples from the links graph with triples from one of the sources. This ensures that only RDF data with a subject contained in the linked entities is evaluated. Filters are applied to exclude geospatial entities. We store the returned data in a new graph containing only the metadata of the source. This allows us to avoid the major overhead of first fetching the data in our application and then importing them to Virtuoso.

With the data resident in a new graph in our local Virtuoso instance we can perform schema matching faster and more robustly. This graph only contains triples from the linked entities and is used for the schema matching analysis and for efficient metadata fusion.

### 3.2.3 PostGIS optimizations

Apart from keeping the input links in a local Virtuoso RDF graph, we also create a links table in PostGIS. This way, we optimize PostGIS based queries on geometries fetching and transformations by limiting the processing to the minimum required metadata of the linked entities (see above for an example SPARQL query for fetching and inserting all metadata from a graph).

For example, the SQL query for executing the fusion action ShiftPolygonToPointTransformation in PostGIS is shown below:
The shown query operates only on the geometries of the linked entities. This is achieved by joining the geometry tables with the one containing the links. Specifically, we combine geometries from dataset A with geometries from dataset B and perform a shifting of geometry B to the centroid of geometry A. The intermediate results are stored as rows of a temporary table. This temporary table is lastly inner-joined with the links table in order to exclude entities which are not linked. The resulting table contains the linked entities and the fused geometry.

Notice that we perform geometry fusion by issuing a single query to PostGIS. This only required a single pre-process step to upload the data to the database.

3.3 FAGI-gis version 2

The second version of FAGI-gis significantly improved the efficiency and scalability of the tool, reducing the fusion runtimes by orders of magnitude. It also extended fusion functionality allowing advanced fusion actions for both spatial and non-spatial metadata, as described in detail in Section 2. Finally, a web interface was implemented to facilitate the fusion process by visualizing linked entities (both the initial ones and the ones resulting after fusion) on a map, allowing FAGI-gis to run in two modes: (a) Batch processing, where a configuration file is provided and the tool performs static fusion actions on the defined datasets and (b) User interactive, where the linked entities can be filtered and visualized on a map and the user can select specific/different fusion actions for different properties of one or more pair of linked entities. Table 2 summarizes the additions and enhancements performed on FAGI-gis during the second year of development.

Table 2: Feature matrix for FAGI-gis versions 1 and 2

<table>
<thead>
<tr>
<th>Feature</th>
<th>Version 1</th>
<th>Version 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fusion of spatial metadata</td>
<td>7 fusion actions</td>
<td>8 fusion actions</td>
</tr>
<tr>
<td>Fusion of non-spatial metadata</td>
<td>-</td>
<td>7 fusion actions</td>
</tr>
<tr>
<td>Fusion of multiple properties (m-to-n fusion)</td>
<td>-</td>
<td>Supported</td>
</tr>
<tr>
<td>Fusion of property chains</td>
<td>-</td>
<td>Supported</td>
</tr>
<tr>
<td>Property matching</td>
<td>-</td>
<td>Supported</td>
</tr>
<tr>
<td>Link rejection</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Geospatial Vocabulary support</td>
<td>GeoSPARQL</td>
<td>GeoSPARQL, Basic Geo</td>
</tr>
<tr>
<td>Modes</td>
<td>User Interactive</td>
<td>Batch processing, User Interactive</td>
</tr>
<tr>
<td>User Interfaces</td>
<td>Desktop GUI</td>
<td>Web GUI, Command Line Interface</td>
</tr>
<tr>
<td>GeoKnow Generator Integration</td>
<td>-</td>
<td>Supported</td>
</tr>
<tr>
<td>Geometries fusion (runtimes)</td>
<td>baseline</td>
<td>1-2 orders of magnitude faster</td>
</tr>
<tr>
<td>Geometries import (runtimes)</td>
<td>baseline</td>
<td>2 orders of magnitude faster</td>
</tr>
<tr>
<td>Non-spatial metadata fusion (runtimes)</td>
<td>baseline</td>
<td>1-2 orders of magnitude faster</td>
</tr>
</tbody>
</table>

In the following sections, we provide implementation information and demonstrate the usage of the two versions of FAGI-gis.

3.3.1 Software description

3.3.1.1 Implementation information

FAGI-gis is implemented in Java, and it provides a web-based, graphical user interface for executing all the steps of the fusion process, as well as a command line interface for executing batch fusion tasks. It uses Virtuoso RDF store as a back-end for storing initial datasets and outputting the final, transformed
datasets. Also, it uses PostgreSQL/PostGIS for spatial indexing and processing. **FAGI-gis** also uses the following external libraries:

- **Apache Jena**: Jena is a Java framework for building Semantic Web applications [Jena]. It provides a collection of tools and Java libraries for developing semantic web and linked-data apps, tools and servers.

- **Google Guava**: The Guava [Guava] project contains several of Google’s core libraries that we rely on in our Java-based projects: collections, caching, primitives support, concurrency libraries, common annotations, string processing, I/O, and so forth.

- **Java WordNet Library (JWNL)**: JWNL [JWNL] is a java API for accessing the WordNet relational dictionary. It also provides functionality beyond data access, such as relationship discovery and morphological processing.

- **WordNet**: WordNet [Wnet] is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

### 3.3.1.2 Components

**FAGI-gis** consists of six basic components, as depicted in Figure 9: the **Interface component** that provides the interface to the user to access the rest of the components (either through command line or through graphical interface), the **POSTGIS component** that handles the geospatial indexing and processing, the **CORE component** that handles the distinction of spatial and non-spatial properties, their loading into the database, scoring and fusing and the **TRANSFORMATIONS component** that implements the supported fusion actions through geometry transformations. Next, we describe in more detail the functionality of each component.

- **Web interface / Command Line Interface**:  
  **Command Line Interface**: This component takes as input a configuration file, containing information about the database configuration, Virtuoso SPARQL endpoints, graph names for the source and the fused dataset, the location of the links file with the interlinked entities and a static set of fusion actions to be performed in batch mode for all the input data.  
  **Web Interface**: This component provides a more detailed way for user interaction that is supported by map visualization of the linked entities to be fused (and the resulting fused entities). The user can select specific links and choose fusion actions upon them while getting feedback about the results. This interface handles in a more fine-grained way the fusion process, allowing the user to check the state and the quality of the initial data and results before she actually chooses a fusion action for the geometries or the non-spatial metadata. The CLI version on the other hand uses a pre-configured workflow for the fusing process.

- **UI Workers component**: This component stores the database configuration such as database name, username and password. It also keeps information about the input datasets, such as the SPARQL endpoint, the graph names and possible regular expressions provided by the user for the subjects. This component handles also the procedure of the transformations against a set of links and exports triples from the given datasets using their SPARQL endpoints and graph names and then imports them into the PostGIS database. Finally, it handles the scoring process for the given transformations against a set of links.

- **PostGIS**: This component is responsible for handling the initialization of a new PostGIS database and provides methods for importing RDF triples to the PostGIS database. Specifically, it establishes a connection to the specified database, loads endpoint/graph information into the appropriate dataset table, it loads the geospatial data regarding geometries into the appropriate geometry tables and, finally, it closes the database connection.
• **Core component**: The *Importer* provides the infrastructure of exporting the triples regarding geometries from the datasets of the input SPARQL endpoints and loading them into PostGIS database. It fetches all triples with a subject matching the subject regex parameter and excludes all geometry related properties. These triples are then imported into PostGIS. The *GeometryFuser* provides methods for obtaining RDF links and scoring the selected fusion strategy for every link, with an optional threshold from the user. This component is also responsible for applying the fusion transformation against the selected links. Finally, it keeps information about the links between RDF nodes from the two datasets obtained by a file containing the interlinked entities.

• **Virtuoso Importer component**: This component is responsible for importing all the fused geometries back to Virtuoso. The resulting, fused data can be integrated in the first original graph or be written into a new, separate graph. The fused data contain the fused spatial and non-spatial metadata. This component also provides a bulk loading process for inserting the data in Virtuoso. The batch processing mode is enabled in order to fuse a large amount of links quickly and efficiently.

• **Transformations component**: This component implements all the available fusion actions, using methods for fusing geometries and providing appropriate scoring based on the fusion strategy. The available fusion actions are presented in Section 2.2.3.
3.3.1.3 Supported data sources and formats

Currently, FAGI-gis supports importing data from a Virtuoso SPARQL endpoint where the actual endpoint and the graph URI of the dataset are required. The output is written into the underlying Virtuoso RDF store. Also, apart from the already supported GeoSPARQL vocabulary (with WKT serialization) for representing geospatial features, support for Basic Geo vocabulary was added in this version. FAGI-gis
allows the creation of a new, local Virtuoso graph for outputting the fused dataset or incorporating the fusion results on the initial, source dataset graph.

3.3.2 Installation/Building instructions

This initial version of FAGI-gis is publicly available, offering the entire source code as well as a .deb package that contains executable binaries [FAGI-gis]. The .deb package can be installed on any system supporting a deb package manager. The package requires that Java JRE (or SDK) 1.7 (or later) is installed and a Virtuoso store and a PostgreSQL/PostGIS database are set up on the machine where the software runs. Specifically, we use Virtuoso v07.00.3203 and PostgreSQL v9.3.5 w/ PostGIS v2. The batch processing version additionally requires that Virtuoso is set up with a user provided directory for file loading. This can be achieved by including the new directory in the DirsAllowed entry of the virtuoso.ini file. The web interface version is distributed as a standalone .war file which needs to be deployed in the webapps folder of an existing Tomcat installation. Lastly, the WordNet dictionary needs to be installed at a known location. FAGI-gis has been successfully tested in both MS Windows and Linux environments. The software’s code is organized in a Maven Project that can be directly loaded into an IDE (e.g. Eclipse, NetBeans) and be further developed.

3.3.3 FAGI-gis for batch processing

FAGI-gis for batch processing allows the offline configuration by the user of all the parameters required for the batch fusion of a set of linked entities, so that the whole fusion process can be executed in one step and produce a fused dataset where all pairs of entities are fused exactly the same way. The user is required to fill in a configuration file and then run FAGI-gis through the command line, using as input this file, as shown in the example command below:

```
java -jar fagis-gis.jar -c /path/config_file
```

Next, we describe the fields that must be filled by the user, based on an actual configuration file for FAGI-gis. First, the user is required to provide information about the path of the file containing the links interlinking the RDF entities and the output graph where the final fused dataset will be written. Also, the path to the WordNet dictionary that is utilized for property matching is required.

```
#############################
## Template for FAGI-gis configuration in order to fuse two datasets containing geospatial data.
## Specify parameters according to the examples for your specific dataset.
## Examples given below assume an existing PostgreSQL+PostGIS database.
#############################

# Input and output parameters

# Local paths used during processing (all paths are ABSOLUTE)  
# Modify virtuoso.ini and include a folder for bulk inserts  
# if the output graph is not specified, Source A is used
#linksFile = /home/user/links.nt  
#virtuosoAllowedDir = /home/user/bulk_inserts  
#outputGraph = http://localhost:8890/fused_dataset  
#wordnetDir = http://localhost:8890/fused_dataset  
#wordnetDir = http://localhost:8890/fused_dataset

linksFile = ${links_file}  
virtuosoAllowedDir = ${virt_allowed_dir}  
wordnetDir = ${wordnet_dir}
```

```
Next, connection and configuration information with respect to the local installations of Virtuoso and PostGIS are provided, along with information about the SPARQL endpoint of the two input datasets (e.g. endpoint and graph URIs).

```plaintext
# PostGIS Configuration Properties
pg_DatabaseName = postgis1
pg_User = postgres
pg_Password = 1111
pg_Import = true
pg_DatabaseName = ${pg_name}
pq_User = postgres = ${pg_user}
pq_Password = ${pg_pass}
pq_Import = ${pg_import}

# Virtuoso Configuration Properties
vi_URL = localhost:1111
vi_User = dba
vi_Password = dba
vi_URL = ${virt_url}
vi_User = ${virt_user}
vi_Password = ${virt_pass}

# Source A Configuration Properties
sa_Graph = http://localhost:8890/DAV/uni
sa_Endpoint = http://localhost:8890/sparql
sa_Graph = ${sa_graph}
sa_Endpoint = ${sa_endpoint}

# Source B Configuration Properties
sb_Graph = http://localhost:8890/DAV/wiki
sb_Endpoint = http://localhost:8890/sparql
sb_Graph = ${sb_graph}
sb_Endpoint = ${sb_endpoint}
```

Finally, the selected fusion action and the respective required parameters are provided.

```plaintext
# Geometry Fusion Configuration Properties
fuse_Transformation = Keep left
fuse_Threshold = 50
fuse_Transformation = ${fuse_trans}
fuse_Threshold = ${fuse_thresh}

# Metadata Fusion Configuration Properties
mfuse_Transformation = Keep Left Meta
mfuse_Transformation = ${mfuse_trans}

# extra params
Will change according to the fusion type
fuse_scale_factor = 2.0
fuse_scale_factor = ${scale_factor}
```

3.3.4 FAGI-gis for map based, user interactive fusion

Next, we demonstrate the usage of the software through the graphical user interface. Similarly to creating the configuration file for the batch processing version, the user needs to first input the connection information regarding the SPARQL endpoints containing the two datasets, the local Virtuoso and PostGIS databases and the input links file containing the pairs of interlinked entities (Figure 10).
Upon that, two processes are performed: (i) The RDF triples representing the links are loaded in an RDF graph in the local Virtuoso store and (ii) for each dataset, all possible classes that may characterize any of the linked entities are queried from the respective SPARQL endpoints and presented in two distinct lists. The user is able to (optionally) choose specific classes (Figure 11) from both datasets and filter the pairs of linked entities based on them (that is keep only linked entities that belong to the selected classes).

After this task is performed, the linked entities are visualized on the map of the interface through points or polygons (depending on the respective type of geometry they contain). Further, a straight line segment connects each pair of linked entities so that the user can explicitly see on the map the pairs of entities to be fused with each other (Figure 12).
The next step regards property matching. Following the rules described in Section 2.1, FAGI-gis first selects some sample linked entities pairs and tries to match the properties of the entities for each pair individually. Eventually, the total of the properties for all selected link pairs are presented to the user divided into two lists, one for each of the two input datasets (Figure 13). When the user selects a property from one list (dataset), the system marks with green colour the properties of the other list (dataset) that are found to match. The final selection of the matching is performed by the user, based on the functionality described in Section 2.2. The user is able to match a property from one list with one or more properties from the other list. Also, she is able to rename the final, fused property to be kept.
Eventually, after the final set of links to be considered for fusion is selected, and matchings of their properties are identified by the system and verified by the user, the actual fusion task takes place. The user can select one or more pairs of linked entities, either from the list of links or through the map (by clicking on the line segments that represent links). Then, the fusion panel pops-up (Figure 14), that allows to perform different fusion actions, as defined in Section 2.2.3, for each pair of properties corresponding to the linked entities. Upon that, the system executes the required transformations and transforms the results into RDF triples, either to be output in a new graph, or to replace some of the initial triples of one of the input datasets.

3.3.5 Licensing

The FAGI-gis software is provided as free software and its current version (including the Java source code and sample data) is available from [FAGI-gis]. It can be redistributed and/or modified under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3.0 of the License, or (optionally) any later version.
4. Benchmarking

In this section, we evaluate the performance of FAGI-gis with respect to the execution time of several sub-processes that take place throughout the whole fusion process. One of our major goals during the second year of development was to significantly increase the scalability of the tool, so that it can efficiently handle hundreds of thousands of linked entities. The benchmark we performed focuses on (a) measuring the time required by each of the basic sub-processes of FAGI-gis, as well as the total time required, (b) evaluating these results for different fusion actions and (c) similarly, measuring the performance for different input dataset sizes. Thus, the experiments are performed with the batch processing version of the tool, that is targeted in fusing large amounts of linked entities. Also, for our measurements, we consider only fusion actions on spatial entities, since these comprise the most demanding actions in terms of performance.

Next, we first describe the dataset used for the evaluation and the fusion sub-processes we evaluated and then we present the experimental results that compare the runtimes of the previous and the current version of FAGI-gis. The datasets used for the evaluation are available at [GeoBenchFAGI].

4.1 Dataset

The input used for the evaluation is two datasets in the form of graphs stored in Virtuoso. The source of our experimental dataset was Wikimapia [Wiki], a crowdsourced, open-content collaborative mapping initiative, where users can contribute mapping information and, also, download geospatial data through the API it provides [WikiAPI].

In particular, we considered a set of cities throughout the world (Athens, London, Leipzig, Berlin, New York) and downloaded the whole content provided by Wikimapia regarding the geospatial entities included in those geographical areas. Apart from geometries, Wikimapia provided a very rich set of metadata (non-spatial properties) for each entity (e.g. tags and categories describing the geospatial entities, topological relations with nearby entities, comments of the users, etc.). Section 7.1 of the Appendix presents a sample structured dump of Wikimapia describing a geospatial entity, where several properties of the entity are provided.

The aforementioned dumps were transformed into RDF triples in a straightforward way: (a) defining intermediate resources (functioning as blank nodes) where information was organized in more than one levels, (b) flattening the information of deep levels where possible in order to simplify the structure of the dataset and (c) transforming tags into OWL classes. A sample of the outcome RDF dataset is presented in Section 7.2.

Specifically, we developed a parsing tool to communicate with the Wikimapia API and construct appropriate n-triples from the data set. The tool takes as input a bounding box in the form of wgs84 coordinates (min long, min lat, max long, max lat). We chose five initial bounding boxes for each of the cities as mentioned above. The provided bounding box was then further divided by the tool in a grid of smaller bounding boxes in order to overcome the upper limit per area of the returned entities from Wikimapia API. For each place returned, we transformed all properties into RDF triples.

Every tag found is assigned to an OWL class and gets an appropriate label, corresponding to the textual description in the initial Wikimapia XML file. Each place becomes an instance of the classes provided by its tags. For the rest of the returned Wikimapia fields, we create a custom property in a uniform way for each field of the returned Wikimapia XML file. The properties resulting from the Wikimapia XML fields point to their literal values. For example, we construct properties about each place’s language id, Wikipedia link, URL link, title, description, edit info, location info, global administrative areas, available languages and geometry information. If these fields follow a deeper tree structure, we assign the properties at intermediate custom
nodes by concatenating the property with the place ID; these nodes function as blank nodes and connect the initial entity with a set of properties and the respective values.

Finally, some of the literal values had to be modified in order to follow some conventions for uploading into Virtuoso (e.g. adding/removing escape characters and quotes, deleting line separators, etc.). We provide a sample of the produced dataset in Section 7.2.

Upon that, we split the Wikimapia RDF dataset, duplicating the geometries and dividing them into the two datasets in the following way: for each polygon geometry, we created another point geometry located in the centroid of the polygon and then shifted the point by a random (but bounded) factor. The polygon was left in the first dataset where the point was transferred to the second dataset. The rest of the properties where distributed between the two datasets as follows: The first dataset consists of metadata containing the main information about the Wikimapia places and edit information about users, timestamps, deletion state and editors. The second dataset consists of metadata concerning basic info, location and language information.

This way, the two datasets essentially refer to the same Wikimapia entities, differing only in geometric and metadata information. Considering this, every entity that exists in both datasets is considered interlinked among the datasets.

### 4.2 Benchmark description

The evaluation process emphasizes on the comparison of the two versions of the tool. As mentioned above, there are five parameter types that are varied and upon which the two versions of FAGI-gis are compared. The values that are assigned to each of these parameters for the experiments are presented in Table 3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (in number of links between entities)</td>
<td>1000, 10000, 100000</td>
</tr>
<tr>
<td>Size of property chains considered as non-spatial metadata</td>
<td>1(no property chains), 2</td>
</tr>
<tr>
<td>Fusion action</td>
<td>Keep left, keep both, shift geometry</td>
</tr>
<tr>
<td>Sub-process of the fusion process (measured in sec)</td>
<td>Geometry loading, Geometry fusion, Geometry triples output, Metadata triples output, Total process</td>
</tr>
<tr>
<td>Include Virtuoso query execution</td>
<td>Yes, No</td>
</tr>
</tbody>
</table>

The software is tested against 1000, 10,000 and 100,000 links. We note that these numbers actually correspond to a much higher number of total triples to be fused. This number is represented by the size of each fused dataset in terms of triples. The second parameter is the size of property chains (sequences) that are considered as metadata of the linked entities. The available dataset had only property chains of depth 2, so our measurements were limited to chains of size 1(no property chains) and 2.

Further, we tested the systems on three different fusion actions regarding geospatial properties: keep left which is a simple fusion action where no special processing is required; shift geometry, which requires a spatial transformation to take place; keep both, which requires that a larger number of geometries are kept in the fused dataset.

The fourth parameter allows the evaluation of the efficiency of each sub-process of the fusion workflow. Specifically, it considers the steps of (i) Loading geometries into PostGIS, (ii) Fusing geometries in PostGIS, (iii) Outputting geometry triples, (iv) Outputting non-spatial properties and (v) all above steps
together. Finally, the fifth parameter isolates the query execution time on Virtuoso and provides runtime measurements with and without it, since query execution on Virtuoso can be considered as independent of the FAGI-gis implementation.

4.3 Version comparison

Next, we present the evaluation results on the two versions (FAGI-gis v1.0 and FAGI-gis v2.0) of the software to demonstrate the improvements that have been achieved in the second version. We present three tables corresponding to the three different values that are assigned to the parameter “size” (measured by number of links between spatial entities): 1000, 10000 and 100000. All columns present times in seconds except from the last column (triples in fused graph) that counts number of triples. The notation that is used in the tables is the following:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNF:</td>
<td>Did Not Finish after 1 hour of execution</td>
</tr>
<tr>
<td>QE:</td>
<td>counting query execution in virtuoso</td>
</tr>
<tr>
<td>WQE:</td>
<td>without counting query execution in virtuoso</td>
</tr>
<tr>
<td>/:</td>
<td>program did not reach there, because of DNF of previous process</td>
</tr>
</tbody>
</table>

The first two tables (Table 4 and Table 5) provide a direct comparison in runtimes of the previous and the current version of the software, in all sub-processes, for the case of including property chains of size 2. We can see that the new version is one to (more than) two orders of magnitude faster compared to the year-one version; that is a significant increase in efficiency. In the case of 1000 links, v2.0 achieves two orders of magnitude faster performance, even on total processing time; in the case of 10000 links, the increase in performance lies between one and two orders of magnitude (~30 times faster in the worst case of the Shift action).

Another observation is that loading geometries in PostGIS constituted a noticeable bottleneck of the system in the previous version. In v2.0 this has been significantly improved by reducing the runtime of this sub-process by more than two orders of magnitude. Finally, we can also see that query execution (which directly depends on the RDF store) is also an unavoidable bottleneck that consumes the largest part of runtime in the respective sub-processes (Geometry triples output, Metadata triples output).

### Table 4: FAGI version comparison on 1000 links (chain=2)

<table>
<thead>
<tr>
<th>1000 links</th>
<th>Geometry loading</th>
<th>Geometry fusion</th>
<th>Geometry triples output</th>
<th>Metadata triples output</th>
<th>Total Time (QE)</th>
<th>triples in fused graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>46.17</td>
<td>8.36</td>
<td>0.21</td>
<td>1642.88</td>
<td>21531</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>11.86</td>
<td>3.08</td>
<td>0.19</td>
<td>0.001</td>
<td>20.06</td>
<td>21531</td>
</tr>
<tr>
<td>Keep Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>32.22</td>
<td>7.02</td>
<td>0.5</td>
<td>1618.81</td>
<td>21531</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>11.86</td>
<td>0.12</td>
<td>0.03</td>
<td>0.001</td>
<td>16.13</td>
<td>21531</td>
</tr>
<tr>
<td>Keep Both</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>66.71</td>
<td>5.1</td>
<td>0.16</td>
<td>1660.09</td>
<td>23531</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>11.86</td>
<td>0.21</td>
<td>0.13</td>
<td>0.001</td>
<td>15.36</td>
<td>23531</td>
</tr>
</tbody>
</table>
Table 5: FAGI version comparison on 10000 links (chain=2)

<table>
<thead>
<tr>
<th>10000 links</th>
<th>Geometry loading</th>
<th>Geometry fusion</th>
<th>Geometry triples output</th>
<th>Metadata triples output</th>
<th>Total Time (QE)</th>
<th>triples in fused graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shift</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>596.61</td>
<td>64.42</td>
<td>0.94</td>
<td>734.87</td>
<td>134.91</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>11.86</td>
<td>7.18</td>
<td>0.76</td>
<td>0.002</td>
<td>76.22</td>
<td>0.96</td>
</tr>
</tbody>
</table>

| **Keep Left** |                  |                 |                         |                         |                 |                        |
| FAGI-gis v1.0 | 1500            | 340.55          | 65.87                   | 0.23                    | 685.35          | 121.49                 | 2591.77               | 222445                 |
| FAGI-gis v2.0 | 11.86           | 5.55            | 0.12                    | 0.002                   | 24.93           | 2.88                   | 42.46                 | 222445                 |

| **Keep Both** |                  |                 |                         |                         |                 |                        |
| FAGI-gis v1.0 | 1500            | 768.35          | 46.89                   | 1.01                    | 608.47          | 109.2                  | 2923.71               | 242445                 |
| FAGI-gis v2.0 | 11.86           | 6.9             | 0.56                    | 0.002                   | 67.69           | 0.83                   | 87.01                 | 242445                 |

Table 6 presents the respective measurements for the larger dataset of 100000 links. The difference here is that the previous version of the software could not complete the sub-process of fusing geometries after 1 hour of running, so it was terminated. On the other hand, the current version is able to perform fusion on a dataset of 100000 links in less than 5 minutes (Total Time) for all three tested fusion actions. Note also that, although the links are only 100000, the final triples of the fused datasets are ~5 Million. This fact showcases the improvement in the scalability of the software, since it is now able to handle millions of triples in a few minutes.

Table 6: FAGI version comparison on 100000 links (chain=2)

<table>
<thead>
<tr>
<th>100000 links</th>
<th>Geometry loading</th>
<th>Geometry fusion</th>
<th>Geometry triples output</th>
<th>Metadata triples output</th>
<th>Total Time (QE)</th>
<th>triples in fused graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shift</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>DNF</td>
<td>DNF</td>
<td>DNF</td>
<td>DNF</td>
<td>DNF      /</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>11.86</td>
<td>18.46</td>
<td>6.32</td>
<td>1.72</td>
<td>239.65</td>
<td>12.8</td>
</tr>
</tbody>
</table>

| **Keep Left** |                  |                 |                         |                         |                 |                        |
| FAGI-gis v1.0 | 1500            | DNF             | DNF                     | DNF                     | DNF             | DNF      / | 4936015               |
| FAGI-gis v2.0 | 11.86           | 8.88            | 2.69                    | 0.002                   | 272.04          | 10.65                  | 295.47                | 4936015               |

| **Keep Both** |                  |                 |                         |                         |                 |                        |
| FAGI-gis v1.0 | 1500            | DNF             | DNF                     | DNF                     | DNF             | DNF      / | 5136015               |
| FAGI-gis v2.0 | 11.86           | 19.16           | 8.94                    | 0.001                   | 212.26          | 8.19                   | 252.22                | 5136015               |

Below, we present the equivalent tables for the case where we do not consider any property chains and we retrieve only immediate properties of linked entities for non-spatial metadata. In this case, the total amount of produced triple decreases at least by 20%, and, in the largest dataset case (Table 9 against Table 6) the decrease is by 50%. Of course, the process that is mostly relieved by the reduction of metadata triples is the **Metadata triples output**. Given that this process consumes a significant percentage of the total time in **FAGI-gis v2.0**, the fact that the Total Time of fusion is reduces in this case is expected. The decrease is more apparent, again, in the largest dataset (Table 9 against Table 6), where Total Time of processing decreases from 40% to 55% percent, demonstrating a relatively linear behavior compared to number the total triples output in the fused graph.
Table 7: FAGI version comparison on 1000 links (chain=1)

<table>
<thead>
<tr>
<th>1000 links</th>
<th>Geometry loading</th>
<th>Geometry fusion</th>
<th>Geometry triples output</th>
<th>Metadata triples output</th>
<th>Total Time (QE)</th>
<th>triples in fused graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>45.4</td>
<td>14.92</td>
<td>0.16</td>
<td>43.24</td>
<td>9.34</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>1.2</td>
<td>0.19</td>
<td>0.06</td>
<td>1.47</td>
<td>0.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keep Left</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>30.69</td>
<td>12.5</td>
<td>0.02</td>
<td>39.84</td>
<td>7.84</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>0.27</td>
<td>0.08</td>
<td>0.009</td>
<td>1.99</td>
<td>0.14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keep Both</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>64.47</td>
<td>4.65</td>
<td>0.13</td>
<td>39.76</td>
<td>8.26</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>0.31</td>
<td>0.22</td>
<td>0.007</td>
<td>1.39</td>
<td>0.17</td>
</tr>
</tbody>
</table>

Table 8: FAGI version comparison on 10000 links (chain=1)

<table>
<thead>
<tr>
<th>10000 links</th>
<th>Geometry loading</th>
<th>Geometry fusion</th>
<th>Geometry triples output</th>
<th>Metadata triples output</th>
<th>Total Time (QE)</th>
<th>triples in fused graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>567.43</td>
<td>69.73</td>
<td>1.05</td>
<td>454.47</td>
<td>74.28</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>1.67</td>
<td>1.01</td>
<td>0.007</td>
<td>15.59</td>
<td>0.68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keep Left</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>333.02</td>
<td>64.46</td>
<td>0.23</td>
<td>572.58</td>
<td>66.15</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>0.69</td>
<td>0.34</td>
<td>0.006</td>
<td>13.39</td>
<td>0.66</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keep Both</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>752.61</td>
<td>48.34</td>
<td>1.06</td>
<td>557.69</td>
<td>63.23</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>0.92</td>
<td>1.11</td>
<td>0.007</td>
<td>13.91</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Table 9: FAGI version comparison on 100000 links (chain=1)

<table>
<thead>
<tr>
<th>100000 links</th>
<th>Geometry loading</th>
<th>Geometry fusion</th>
<th>Geometry triples output</th>
<th>Metadata triples output</th>
<th>Total Time (QE)</th>
<th>triples in fused graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>DNF</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>7.83</td>
<td>7.5</td>
<td>0.006</td>
<td>104.54</td>
<td>6.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keep Left</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>DNF</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>6.43</td>
<td>2.81</td>
<td>0.007</td>
<td>110.58</td>
<td>5.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keep Both</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FAGI-gis v1.0</td>
<td>1500</td>
<td>DNF</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>FAGI-gis v2.0</td>
<td>12</td>
<td>12.25</td>
<td>11.02</td>
<td>0.007</td>
<td>117.49</td>
<td>5.21</td>
</tr>
</tbody>
</table>
5. Conclusions

In this deliverable, we presented the extensions and enhancements that were performed on FAGI-gis during the second year of development. The first version of the software focused on implementing the baseline functionality for fusing geometries, implementing the initial infrastructure that supported indexing, transformations and fusion actions on geometries of linked entities.

During the second year, we built on this initial version, extending its functionality and dramatically enhancing its efficiency. Specifically, we extended and refined the initial set of fusion actions so that both spatial and non-spatial metadata, as well as groups of metadata can be handled per fusion action. Further, we performed several optimizations that increased the scalability and efficiency of the software by even more than two orders of magnitude in several fusion cases. Another important extension was the implementation of a map-based, web interface for facilitating the fusion actions through visualization and filtering of linked entities. Also, we presented an initial benchmark that aims at evaluating the performance of the tool in terms of efficiency and scalability and used it to demonstrate the increase in efficiency, compared to the first version of FAGI-gis.

Our next steps involve: (a) further extending current functionality by adding more fusion actions, (b) implementing an effective learning mechanism that can train models on past user fusion actions and produce fusion action recommendations for newly inserted linked entities, based on these models, (c) implementing advanced fusion functionality (e.g. link discovery and fusion by exploiting geospatial relations and neighbourhoods of geometries). Finally, another direction towards further increasing the scalability of the software would be the development of a distributed version of the software for (parallel) fusing of massive amounts of linked entities. This development could boost even further the scalability of the tool, allowing the efficient fusion of hundreds of millions of triples.
6. References


[GeoBenchFAGI] FAGI-gis benchmark. Available at: https://github.com/GeoKnow/GeoBenchLab/tree/master/FAGI-gisBench


[WikiAPI] Wikimapia API. Available at: http://wikimapia.org/api/
7. Appendix

7.1 Wikimapia XML Sample

```xml
<id>12961631</id>
<object_type>1</object_type>
<language_id>0</language_id>
<language_iso>en</language_iso>
<language_name>English</language_name>
<title>Greenburgh, New York</title>
<description>Greenburgh is a town in the western part of Westchester County, New York, United States. The population was 88,400 at the 2010 census. Within its borders can be found six villages: Ardsley, Dobbs Ferry, Elmsford, Hastings-on-Hudson, Irvington, and Tarrytown.</description>
<is_building/>
<is_region/>
<is_deleted/>
<tags>
  <tags_0>
    <id>2658</id>
    <title>township</title>
  </tags_0>
  <tags_1>
    <id>44733</id>
    <title>draw only border</title>
  </tags_1>
</tags>
<parent_id>0</parent_id>
<edit_info>
  <user_id>930156</user_id>
  <user_name>elbarto90se</user_name>
  <date>1351303974</date>
  <is_unbindable/></deletion_state/>
  <is_in_deletion_queue/><is_in_undeletion_queue/>
</edit_info>
<is_protected/>
<location>
  <lon>-73.8353455</lon>
  <lat>41.0309381</lat>
  <north>41.0870974</north>
  <south>40.9747789</south>
  <east>-73.772099</east>
  <west>-73.898592</west>
  <country>USA</country>
  <state>New York</state>
  <place>Scarsdale</place>
  <country_adm_id>233087</country_adm_id>
  <gadm>
    <gadm_0>
      <id>1000</id>
      <country>0</country>
      <level>0</level>
      <is_last_level>0</is_last_level>
      <name>World</name>
      <iso/>
      <type/>
    </gadm_0>
  </gadm>
</location>
```

Official Website: http://www.greenburghny.com


7.2 Wikimapia Sample RDF File

```
<http://geoknow.eu/geodata#place_12961631> <http://geoknow.eu/geodata#has_language_id> "0" .
<http://geoknow.eu/geodata#place_12961631> <http://geoknow.eu/geodata#has_language_iso> "en" .
<http://geoknow.eu/geodata#place_12961631> <http://geoknow.eu/geodata#has_urlhtml>
<http://geoknow.eu/geodata#place_12961631> <http://www.w3.org/2002/07/owl#Class>奥巴马
<http://www.w3.org/2000/01/rdf-schema#type>
<http://www.w3.org/2000/01/rdf-schema#Label> .
<http://geoknow.eu/geodata#place_12961631> <http://geoknow.eu/geodata#has_gadm>
<http://geoknow.eu/geodata#has_gadm
"233" .
```

---

D3.3.1 v. 1.0

Page 46
"United States".

"United States".

"United States".

"New York".

"New York".

"New York".

"Greenburgh - New York".

"Greenburgh - New York".

"Greenburgh - New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".

"New York".