# Deliverable 3.2.3: Fusing of Geospatial Relations

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due Date of Deliverable</td>
<td>Month 31, 30/06/2015</td>
</tr>
<tr>
<td>Actual Submission Date</td>
<td>Month 32, 29/07/2015</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>34</td>
</tr>
<tr>
<td>Filename</td>
<td>D3.2.3_Fusing_of_geospatial_relations.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 functionality regarding fusion with geospatial relations, as well as several other enhancements and extensions performed on FAGI-gis during the third year of the project.

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.
## 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>09/07/2015</td>
<td>First version and section outline</td>
<td>Giorgos Giannopoulos</td>
</tr>
<tr>
<td>0.2</td>
<td>13/07/2015</td>
<td>Description of fusing with spatial relations</td>
<td>Giorgos Giannopoulos, Nick Vitsas</td>
</tr>
<tr>
<td>0.3</td>
<td>15/07/2015</td>
<td>Description of learning mechanisms</td>
<td>Giorgos Giannopoulos, Nikos Karagiannakis</td>
</tr>
<tr>
<td>0.4</td>
<td>17/07/2015</td>
<td>Description of FAGI's evolution</td>
<td>Giorgos Giannopoulos</td>
</tr>
<tr>
<td>0.5</td>
<td>20/07/2015</td>
<td>Demonstration of the tool</td>
<td>Giorgos Giannopoulos</td>
</tr>
<tr>
<td>0.6</td>
<td>21/07/2015</td>
<td>Version prepared for the project's peer-review</td>
<td>Giorgos Giannopoulos, Dimitrios Skoutas, Spiros Athanasiou</td>
</tr>
<tr>
<td>0.7</td>
<td>25/07/2015</td>
<td>Internal Review</td>
<td>Andreas Both</td>
</tr>
<tr>
<td>0.8</td>
<td>28/07/2015</td>
<td>Additions on optimizations</td>
<td>Nick Vitsas</td>
</tr>
<tr>
<td>0.9</td>
<td>29/07/2015</td>
<td>Additions and improvements based on internal review comments</td>
<td>Dimitrios Skoutas, Spiros Athanasiou, Giorgos Giannopoulos</td>
</tr>
<tr>
<td>1.0</td>
<td>29/07/2015</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 during the third year of the project to efficiently support a wide range of fusion actions on geospatial RDF data.

FAGI-gis is a software, implemented within GeoKnow, for fusing geospatial RDF data. It receives as input two datasets (through SPARQL endpoints) 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. 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)\(^1\). In its final version, FAGI-gis supports both individual and batch fusion actions, as well as semi-automatic property matching, link creation and fusion actions recommendation.

During the third year of development, our main focus was on exploiting spatial relations and batch handling between linked, but also unlinked, entities. This aim was closely interconnected with the Unister (commercial use case) requirements for richer batch fusion functionality. To this end, we extended FAGI-gis to three directions exploiting, in different ways, geospatial relations between entities:

(a) **Clustering of links between entities.** With this functionality, we essentially create groups of pairs of linked entities, that are interlinked in a similar manner w.r.t. the geospatial dimension. This way, we allow users to perform batch fusion actions on different groups of linked entities. For example, a user may select to handle differently those pairs of linked entities that have a very small spatial distance, as opposed to others that have greater distance.

(b) **Link recommendation on neighbourhoods of unlinked entities.** With this feature, apart from fusion functionality, FAGI-gis also offers the capability to create new links between entities, based on spatial relations and non-spatial metadata similarities, between entities within spatial neighbourhoods.

(c) **Learning mechanisms that exploit geospatial relations and historical fusion actions.** We introduce two learning mechanisms for producing (i) OSM categories (classes) recommendations for fused geospatial entities (realized as \texttt{rdf:type} properties in the fused dataset) and (ii) fusion recommendations for pairs of linked entities based on past fusion actions. Both mechanisms exploit geospatial relations and non-spatial metadata between training entities (annotated geospatial entities in OSM or already fused pairs of entities in FAGI-gis) and new entities in order to produce recommendations for new pairs of linked entities.

Further, significant effort has been put on improving and facilitating the map-based user interaction. Thus, we added map-based operations for transforming (e.g. moving or rescaling) geometries on the map and fusion functionality that facilitates the selection of proper namespaces and properties to be fused. Also, following requirements of Unister, we put emphasis on allowing the user to execute several types of batch fusion actions, through the FAGI interface, including: (a) Clustering of similarly linked entities and specializing batch fusion actions per cluster; (b) Manually (or by defining Bounding Boxes) selecting individual linked pairs of entities to perform specific batch fusion actions; (c) Batch loading of remaining entities’ properties, upon the fusion of matched 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.

\(^1\) Terms “property” and “metadata” will be used interchangeably within this document.
In Chapter 2, we describe the functionality extensions performed on FAGI-gis w.r.t. exploiting geospatial relations for batch fusion actions and for link creation in neighbourhoods of unlinked entities.

In Chapter 3, we describe the learning mechanisms that were incorporated into FAGI, and demonstrate the class recommendation and fusion recommendation functionality.

In Chapter 4, we briefly present the evolution of the tool through the three years of the project and then we demonstrate its usage, showcasing the wealth of fusion options offered to the end user.
## 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>OSM</td>
<td>OpenStreetMap (<a href="https://www.openstreetmap.org">https://www.openstreetmap.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 FRAMEWORK ARCHITECTURE .................................................................................. 8
   1.2 PROGRESS OF THE WORK ...................................................................................... 10
2. **FUSING WITH SPATIAL RELATIONS** ......................................................................... 12
   2.1 LINK CLUSTERING .................................................................................................... 12
   2.2 NEIGHBOURHOOD LINK DISCOVERY ................................................................. 13
   2.3 BATCH FUSION ACTIONS ....................................................................................... 15
3. **LEARNING MECHANISMS** ....................................................................................... 17
   3.1 FUSION RECOMMENDATION .................................................................................. 17
      3.1.1 Features describing the link between entities .................................................... 17
      3.1.2 Features describing geometric characteristics of individual entities .......... 18
      3.1.3 Features describing geometric relations between entities .............................. 18
      3.1.4 Features describing topological relations between entities ............................ 18
   3.2 OSM CATEGORIES RECOMMENDATION ............................................................... 19
      3.2.1 Learning OSM category recommendations ...................................................... 19
4. **EXTENDED FAGI-GIS** ............................................................................................. 21
   4.1 OVERVIEW OF VERSION 1 ..................................................................................... 21
   4.2 OVERVIEW OF VERSION 2 ..................................................................................... 22
   4.3 OVERVIEW OF VERSION 3 ..................................................................................... 22
      4.3.1 Software description .......................................................................................... 24
         4.3.1.1 Implementation information ....................................................................... 24
         4.3.1.2 Supported data sources and formats .......................................................... 25
      4.3.2 Installation/Building instructions ....................................................................... 25
      4.3.3 FAGI-gis for map based, user interactive fusion ........................................... 25
         4.3.3.1 Single fusion actions ................................................................................ 27
         4.3.3.2 Batch fusion actions ................................................................................ 28
      4.3.4 Licensing ........................................................................................................ 29
      4.3.5 Future work ..................................................................................................... 29
4.3.3.1 Single fusion actions ...................................................................................... 27
5. **CONCLUSIONS** ......................................................................................................... 31
6. **REFERENCES** ........................................................................................................... 32
7. **APPENDIX** ............................................................................................................... 33
   7.1 FAGI-GIS BATCH VERSION DEPLOYMENT ......................................................... 33
List of Figures

Figure 1: Fusion Framework Architecture ................................................................. 9
Figure 2: FAGI-gis in the LD Lifecycle ................................................................. 11
Figure 3: Differently linked spatial entities ......................................................... 12
Figure 4: Unlinked Entities Fetching ................................................................. 14
Figure 5: Candidate links and link creation ......................................................... 15
Figure 6: Batch fusion mode: a single entity pair is selected to represent all the linked entities; every fusion action that is selected in the fusion table (left bottom panel) is propagated to all pairs of linked entities ................................................................. 16
Figure 7: Category recommendations for fused entities ........................................ 19
Figure 8: Desktop GUI of FAGI-gis v1.0 .................................................................. 21
Figure 9: Web Interface for FAGI-gis version 2.0 .................................................. 22
Figure 10: Web Interface for FAGI-gis version 3.0 .................................................. 23
Figure 11: Configuration panels ............................................................................ 25
Figure 12: Link filtering panel ................................................................................ 26
Figure 13: Property matching panel and linked entities visualized on the map ............ 27
Figure 14: Fusion panel ......................................................................................... 28
Figure 15: Fused entity and its properties ............................................................. 28
Figure 16: Link recommendations for an area of unlinked entities ............................ 29

List of Tables

Table 1: Feature matrix for FAGI-gis versions 1 and 2 ........................................... 23
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 shortly describe which features-subcomponents of FAGI have been implemented during each of the three years of the project.

1.1 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 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 architecture is slightly changed compared to the first two years of the project, as described in the basic components presentation that 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.

- **Transformation Component.** This component became obsolete, since the Vocabulary Transformations component was incorporated into the Geospatial Preprocessing component, while the Dataset Quality Assessment Module became obsolete, since this functionality is covered in other Tasks of WP3 (e.g. by Crocus tool [CROCUS]).

- **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 to produce the geometric similarity scores for fusion, 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 unlinked entities, it suggests creation of new links between entities. The Batch Fusion Module supports handling several batch fusion actions, either in the level of entities or in the level of subsets of properties of an entity. The Fusion Recommendation and the Class Recommendation Modules exploit the learning mechanisms to recommend fusion actions for pairs of linked entities and OSM categories as OWL Classes for fused geospatial entities. The Link CLustering Module, allows the clustering of pairs of linked entities in order to be handled with different batch fusion actions.

- **User Interface Component.** This component supports the interaction of the end user with the rest, underlying components. Through this component, the user can select and apply fusion strategies on (parts of) the datasets, perform clustering and batch fusion actions, create new links and fuse previously unlinked entities and enrich the fused entities with OWL Classes based on OSM categories.

- **Learning Component.** This component handles: (a) 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; (b) The training of recommendation models on OSM data, for recommending OSM categories as OWL classes for the fused entities.

---

**Figure 1: Fusion Framework Architecture**
1.2 Progress of the Work

Each component of Figure 1 is coloured based on the framework’s development phase they correspond to. 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 Transformation 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 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.

During the third year of the project, we implemented geospatial relations based fusion functionality, including link discovery through neighbourhoods, clustering of linked entities, several facilities for batch fusion actions, and learning mechanisms for category (class) recommendation and fusion actions recommendation. Further, we re-designed the web-based user interface, facilitating the execution of map-based, individual or batch geometry transformations and fusion actions. Finally, we added SPARQL filtering functionality, so that the user is able to precisely filter the types of linked entities she wishes to fuse or unlinked entities she wishes to interlink.

FAGI’s main functionality regards fusing of geospatial RDF data. Thus, it mainly handles the “Interlinking/Fusing” stage of the Linked Data Lifecycle, as depicted in Figure 2. However, through these years, the tool has expanded beyond its initial purpose, partially addressing, as a result, functionality from other aspects of the Lifecycle. Therefore, it allows Revision/Authoring of the examined RDF data, by allowing the user to correct their geometries (e.g. rescale, move), to rename their properties, etc. Further, it also supports Browsing and Exploration of the data, since it visualizes linked and unlinked geospatial entities on a map and allows the user to query spatial relations between these entities.
The development of FAGI-gis continues beyond Deliverable 3.2.3 and until the end of the project. Our ongoing work involves further efficiency optimizations on the map-based, user interactive version of the tool, extended evaluation of its functionality (and the recommendation mechanisms) on several geospatial RDF datasets and extensions on fusion functionality for several domain specific datasets (e.g. Unister administrative areas data).
2. Fusing with spatial relations

In this section, we describe the functionality extensions performed on FAGI-gis, w.r.t. exploiting spatial relations to perform fusion. The facilities we implemented expand the initial goals described in the DoW, in two ways: we exploit spatial relations (i) for both linked and unlinked entities and (ii) for performing both fusion and link discovery. First, we present the clustering mechanism that allows the identification of groups of linked entities that are interlinked in similar way. Then, we describe the link discovery on neighbourhoods of unlinked entities that utilizes the distance between entities, as well as similarity of non-spatial metadata. Finally, we present the wide range of operations that facilitate batch fusion actions.

2.1 Link clustering

FAGI-gis supports clustering of links between linked entities. Through this process, it allows the identification of pairs of linked entities that are interlinked in “similar ways”. In our case, we consider the line segment (or the vector) that connects the centroids of two linked entities and we examine two characteristics of it: (a) its length and (b) its direction.

The intuition behind (a) is that different distances between linked entities might imply different provenance of the respective spatial entities and, thus, the need for different fusion actions. For example, consider the linked entities presented in Figure 3. The entities enclosed in the yellow box have very small distances with each other, while, on the other hand, in the black and red boxes, the distances between linked entities are much larger. This might imply that each group of entities have been produced in a different way and, thus, the geocoding errors are larger there. So, while in the case of the yellow-box-enclosed entities the user can select any of the two entities (green polygon or blue linestring) of each pair of linked entities as a final geometry for the fused entity, this does not apply for, e.g., the entities in the black box. There, the user needs to examine whether the blue linestrings or the green polygons are actually the geometries with the correct coordinates, and possibly select a complex fusion action (e.g. shift geometry and keep).

Figure 3: Differently linked spatial entities
Similarly, the intuition behind (b) is that different directions of the line segment that connects two interlinked entities might denote different provenance of the information. So, again, two linked entities within the black box (where green polygons consistently lie on south of their corresponding blue linestrings) probably require different fusion actions than a pair of entities within the red box (where green polygons consistently lie on east of their corresponding blue linestrings).

In order to cluster links, we adopt the widely used k-Means clustering algorithm, and specifically, we use its implementation provided by the Weka library [WEKA]. Our clustering items are the vectors that connect the centroids of two interlinked entities. Without loss of generality, we always define the beginning of the vector as the centroid of the entity from dataset A and the end as the centroid of the entity from dataset B. In order to represent these vectors as feature vectors, we need to quantify their length and their direction. The former is straightforward, thus, the length feature for each vector is the distance between the two centroids, normalized by the maximum length found for all vectors (links available for clustering). In order to quantify the direction we consider the unit vector of each one of the link vectors, that is we normalize each of them by its own length. Each produced unit vector essentially represents the direction of the initial link vector, with unitary length. Then, we consider the coordinates x’ and y’ of each unit vector as the features that represent its direction in the two-dimensional space. Eventually, we construct the item’s feature vector (l, x’, y’), where all three features are normalized within the interval [0, 1]. The item vectors from each link-vector are used as input in the clustering algorithm.

The k-Means algorithm takes as input the desirable number of clusters to be created. Upon that, it performs an initial random assignment of items to clusters and then it applies an iterative process where it corrects the assignment by optimizing an objective function that quantifies the quality of the clusters at each stage. The user is able to select the clustering criterion to apply (length, directions), as well as the number of clusters to be produced (in the range of 2 to 10 clusters). If the user selects ‘0’ clusters, then the algorithm performs all possible clusterings (w.r.t. number of produced clusters) and selects the optimal one, i.e. the one which contains the most qualitative clusters.

Finally, the user is able to preview the created clusters of pairs of linked entities in two ways. Initially, all linked entities are previewed in the map, exactly as they were before performing the clustering. However, the links are coloured differently, according to the cluster they belong to. Moreover, when the user selects one of the clusters from a list, then only the linked entities belonging to the cluster are previewed in the map. Finally, the user is able to proceed in batch fusion actions per cluster, as described in Section 2.3.

2.2 Neighbourhood link discovery

Link discovery is another important feature that was added in FAGI-gis. Of course, the tool does not perform interlinking in the classic sense, as this is a job for other well performing and widely used tools, such as LIMES [LIMES] and SILK [SILK]. In our case, we consider the scenario where, for possibly several reasons, not all links between two datasets have been found. For example, there might be the case where the user, at a previous interlinking step with a third party tool, has applied very strict interlinking criteria in order to find links between entities. That is, she may have set a very small distance threshold between two entities to be considered identical, or she might have required similarity in too many non-spatial properties (name, label, address name, etc.). So, many links might have not been identified in the first place, either because of large discrepancies in the coordinates of the entities, or because non-spatial metadata are represented in different ways between the two datasets, leading to low similarity values. Given the above, the link discovery functionality that is supported in FAGI-gis acts complementary to external interlinking tools.

In order to examine unlinked entities, the user is able to draw a bounding box and fetch all included unlinked entities (Figure 4). The first, basic option is to manually create a link between two
entities she considers identical. Note that a link can be created only between entities from different datasets.

Figure 4: Unlinked Entities Fetching

The advanced functionality is realized as link recommendation. The user is able to select a single entity from either dataset and ask the system to recommend candidate entities to be linked with. To do so, the user has to input only a distance threshold, that is the radius of the circle-neighbourhood around the entity where candidate entities will be searched in. Upon that, the search for candidate entities is performed in two steps: (i) A geospatial query is performed that returns candidate entities whose geometries lie within the defined circle. (ii) The candidate entities are ranked based on the similarity of their non-spatial property values, as explained next.

Let the set $S$ of candidate entities $\{e_i\}$ to be linked with the selected entity $e_0$. For each entity $e_i$, each property with literal value $p_{ij}$ is examined. If a property takes the same value in a large percentage of the entities $e_i$, then it is discarded as non-useful for our purpose. For example, in our scenario, a property that would contain the name of the city where a building lies would be useless, since all examined buildings within a distance of a selected building would most probably take the same value. Next, for the rest of the properties, we perform the process described by the following pseudocode:

1. For each entity $e_i$
   For each property $p_{ij}$
   For each property $p_{0j}$
   $\text{Sim}_j(p_{ij}, p_{0j}) = \text{Jaccard similarity of words of the two property values}$

2. For entity $e_i$
   Average the similarity $\text{Sim}_j$ for all its matching properties with the properties of $e_0$ into $\text{Sim}_j^{\text{average}}$ score

3. Recommend to the user candidates $e_i$ for which $\text{Sim}_j^{\text{average}} > \text{threshold } t_c$

Essentially, the algorithm compares all properties of a candidate entity with the properties of the selected entity, only considering properties with literal values. The comparison is performed by extracting the words of the literal of each property and then applying Levenshtein distance on pairs
of words. If the distance of two words is over a threshold $t_w$, then the words are considered identical for the Jaccard distance that is applied next. The Jaccard distance $\text{Sim}_j(p_{ij}, p_{Oj})$ provides the similarity of two properties and the average Jaccard distance $\text{Sim}_{\text{average}}$ provides the similarity between a candidate entity and the selected entity, w.r.t. the similarity of their properties.

The user is finally presented with a set of candidate links, connecting the selected entity's geometry with geometries from the other dataset, on the map. The line segments for the candidate links are rendered with different color and shape, so that the user can distinguish between them and the actual links. The user is then able to select one and only one candidate link per entity, to validate as actual link. Upon that, she can further perform fusion actions on the newly linked pair of entities, in exactly the same way as for the initial, actual links. Also, the newly created links can be output to a file and used to enrich the pre-existing links set.

Finally, the whole process described above can be automated for a neighborhood of entities. In this scenario, the user is not required to select entities one by one in order to get link recommendations; instead, she can simply draw a bounding box and fetch unlinked entities and link recommendations for all the retrieved entities. This option starts an iterative process where, for the retrieved through the bounding box entities, each entity from dataset A is compared to each entity from dataset B, in exactly the same way as described above.

![Figure 5: Candidate links and link creation](image_url)

2.3 Batch fusion actions

Next, we present the set of batch fusion actions that can be applied through FAGI-gis. We note here that the map-based, user interactive version of the tool was initially designed to support mainly single fusion actions, while batch fusion functionality was mainly supported by the command line version (see Deliverable 3.2.2 [GeoKnowD322]). The rationale behind this decision was based on the nature of the problem we solve: while fusing entities and deciding which geometries and metadata to keep, change or merge, a user would probably examine their attributes case by case and would not heavily rely on batch and/or automatic fusion actions. However, after discussing real world cases with Unister (commercial use case), we revised our initial planning and started integrating batch fusion functionality in the map-based version of the tool too. Unister's scenarios include interlinking geospatial entities where the geometries in one of
the datasets might have been massively shifted in the same way. So, in such cases, identifying groups of geometries with similar shifts and being able to transform their geometries in the same way would be of high value.

The most straightforward batch fusion option is performing the same fusion action for all loaded linked entities. To do so, the user has to activate the “Batch” mode on the top right of the screen, and then select an arbitrary pair of linked geometries. Any geometry transformation or fusion action on either spatial or non-spatial properties is then propagated to all available pairs of linked entities. For example, if a user selects a pair of geometries, shifts and rotates geometry B on the map and selects fusion action “Keep B”, then these transformations are applied to geometries B in all pairs of entities and the final, fused entities include as geometry the respective geometry B from each linked entities pair.

Further, the user is able to specifically select which entity pairs will be considered for batch fusion, either by selecting them “by hand”, or by defining a bounding box that includes all pairs of linked entities to be batch-fused. The same applies for individual clusters created as described in Section 2.1: each cluster can be handled as whole, w.r.t. specific fusion actions.

Finally, batch operations can also be executed at the level of sets of properties. As described in Deliverable 3.2.2, the user is presented with two lists of properties, from which she selects which properties of dataset A match with properties from dataset B. Upon this selection of matched properties takes place, fusion actions are applied on them. Apart from these properties, the user is able to select batch copying of the non-matched properties (either from dataset A or from dataset B) into the final, fused dataset.
3. Learning Mechanisms

In this section, we present the learning mechanisms that allow FAGI-gis to: (a) train on previous fusion actions and recommend fusion actions for new pairs of linked entities; and (b) train on OSM data (spatial entities annotated with OSM categories) and recommend classes for fused entities.

3.1 Fusion recommendation

The aim of this functionality is to assist the user in the fusion selection process by recommending possible fusion actions, based on historical fusion actions that have been performed. Our goal is not to fully automate the fusion process, since we believe this is an unrealistic option for the specific problem. This is due to the fact that the fusion process takes into account not only strict geometrical/geographic criteria of the entities to be fused, but also non-spatial metadata of the entities, comparison of the vector representations of the entities (actual geometries) with raster data (underlying map layer), knowledge about the provenance of the two linked entities, etc. All this wealth of metadata makes the fusion process a fuzzy task, where strict fusion rules, based on the metadata of the entities to be fused, cannot be defined. Further, a huge number of training data (past fusion actions) as well as exhaustive fine-tuning of learning algorithms would be required. Given the above, the ideal choice for the specific setting would be a recommendation mechanism that assists the user's decision on fusion actions rather than enforcing it.

To this end, we adopt the widely used Support Vector Machines (SVM) algorithm for classification and we adjust it to our setting. The algorithm, as a model based approach, includes two steps: a training step, where a classification model is created and a classification step, where the created model is utilized to classify new entities. The input of the training step of the algorithm is a set of training entities and their labels, that is the classes in which the entities are assigned. In our case, a training item is a pair of linked entities and a label is the geospatial fusion action that has been selected for this pair.

To produce the classification model, the algorithm maps the training entities into a multidimensional feature space and aims at finding the optimal hyperplanes that discriminate the entities belonging to different labels. To do so, one needs to represent a training item with proper features that capture the item’s relation to the label it belongs to. We describe the features we defined later on in this section. The optimality of the hyperplane depends (among other tuning parameters) on the selected parameter C, which adjusts the trade-off between misclassified training entities and optimal discrimination of correctly classified entities.

The produced model is essentially a weight vector that gives different weights to different features that represent the entities, thus, assigning different importance to different attributes of the items to be classified. Then, this model is applied to new items in order to assign a label to them. Using multiclass classification, scores are assigned to all candidate labels, thus, ranking the relevance of each item to each label.

As we mentioned, in our problem setting, an item is a link between two spatial entities. So, in order to represent the item as feature vector, we need to define features that describe both the link itself and the individual entities that are linked to each other. Specifically, we defined and implemented the training features described next.

3.1.1 Features describing the link between entities

These features represent the link that connects the geometries of the two entities. Here, we exploit the two features that are defined in Section 2.1 for clustering:
3.1.2 Features describing geometric characteristics of individual entities

These features describe individual attributes of the geometries that are interlinked. Each feature occupies two slots in the feature vector, one for each of the two geometries.

- **Type of geometry.** This is a boolean feature that occupies 8 slots in the feature vector, one for each geometry type we consider: *Linestring*, *Polygon*, *CircularString*, *Point*, *MultiLineString*, *MultiPolygon*, *MultiPoint*, *GeometryCollection*.
- **Number of points.** This is one normalized real-valued feature counting the number of points included in the geometry.
- **Area of geometry.** This is one normalized real-valued feature counting the occupied area of the geometry.

3.1.3 Features describing geometric relations between entities

These features describe quantitative relations of the characteristics of the geometries as compared to each other:

- **Sum of points of the two geometries.** This is a boolean feature, so several positions in the feature vector are used to represent it. Each position represents a different range. In total, we define (based on observing statistics on the frequencies of entities in OSM subsets having certain numbers of points) 11 ranges.
- **Sum of areas.** This is a boolean feature, so several positions in the feature vector are used to represent it. Each position represents a different range. In total, we define (based on observing statistics on the frequencies of entities in OSM subsets) 25 ranges.
- **Percentage of Area Difference,** using the formula \( \frac{\text{Area}_A - \text{Area}_B}{\text{Area}_{\max(A,B)}} \). Similarly, this is a boolean feature, with several positions in the feature vector representing various ranges.
- **Percentage of Points Difference.** This feature is identical with the previous one, with the difference that it refers to the points of the two geometries.
- **Mean Edge Length.** The average edge length in both geometries. Similarly, this is a boolean feature, with several positions in the feature vector representing various ranges.
- **Percentage of Edge Variance Difference,** using the formula \( \frac{\text{EdgeV}_A - \text{EdgeV}_B}{\text{EdgeV}_{\max(A,B)}} \). Similarly, this is a boolean feature, with several positions in the feature vector representing various ranges.

3.1.4 Features describing topological relations between entities

These features describe topological relations between the two geometries, e.g. whether their geometries are disjoint or one intersects with the other. Specifically, we implement, as boolean features, the following topological relations: *geometriesCross*, *geometriesTouch*, *geometriesIntersect*, *geometryAWithinB*, *geometryBWithinA*, *geometryAContainsB*, *geometryBContainsA*.

The learning mechanism works as follows: Each fusion action is logged, that is all the necessary information (entities and link characteristics, geometry fusion action (label)) are kept in order to gather training data. As soon as a sufficient number of training fusion actions is gathered,
a classification model is trained. After this point, the training dataset continues to grow with each new fusion action and a new model is periodically trained, in order to include the most recent actions. After the first model is trained, each time the user selects a pair of linked entities, the system classifies the pair and returns the most fitting label (geometry fusion action) for it. This fusion action is selected as the default action in the geometry actions list of the fusion panel.

3.2 OSM categories recommendation

This feature allows the enrichment of the fused dataset with rdf:type triples that annotate the fused entities with OSM categories (as OWL Classes). For this functionality, we exploited the already implemented infrastructure of Task 3.3 for learning classification models on OSM data, described in Deliverable [GeoKnowD332].

The mechanism exploits classification models, already trained in OSM data. As soon as a pair of linked entities is selected for fusion in FAGI-gís, each of the two individual geometries is given as input in the classification model, which produces a set of OSM category recommendations for each of the two geometries. These recommendations are presented to the user in a separate list, under the fusion panel (Figure 7), and the user decides which of these categories to keep as OWL classes for the final, fused entity.

![Figure 7: Category recommendations for fused entities](image)

Next, we briefly describe the training and classification mechanism. Detailed description can be found in [GeoKnowD332].

3.2.1 Learning OSM category recommendations

For producing category recommendations, we use the same classification algorithm as in Section 3.1, SVM for (multiclass) classification. The training items are the geometries of OSM spatial entities, where the labels are the OSM categories these entities are annotated with. The training features that represent the spatial entities are based on solely geometric characteristics of the entities and are given below:

- **Geometry type.** This is a boolean feature, so six positions in the feature vector are assigned to it, for the six distinct geometry types that we can identify: Point, LineString, Polygon, LinearRing, Circle and Rectangle.

- **Number of geometry points.** Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector that are used to represent it. Each position represents a different range. In total, we define (based on a statistical analysis of the frequencies of entities having certain numbers of points) 13 ranges: [1-10], (10-20], (20-30], (30-40], (40-50], (50-75], (75-100], (100-150], (150-200], (200-300], (300-500], (500-1000], (1000-...]. So, according to the number of points of an entity’s geometry, the proper position is set to 1, while the rest positions are set to 0.

- **Area of geometry.** Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector that are used to represent the various ranges of areas we consider. We define intuitively (and considering that in this problem
setting we are mainly interested in entities that are buildings) 24 ranges with increasing length in square meters, until the area of 4000m$^2$, where we consider the 25$^{th}$ range of (4000-...).

- **Mean edge length.** Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector representing different ranges of the mean length of the geometry's edges. In our case, we define 23 ranges, starting from length less than 2 meters and ending with length more than 200 meters.

- **Variance of edge lengths.** Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector representing different variations of a geometry's edges from the mean value. In our case, we define 36 ranges.
4. 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 evolution of the software through the three years of the project. Then, we describe the current version of the software demonstrating, step by step, the usage of the web version.

4.1 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.

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.

4.2 Overview of 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. Finally, a web interface (Figure 9) 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 1 summarizes the additions and enhancements performed on FAGI-gis during the second year of development.

Figure 9: Web Interface for FAGI-gis version 2.0

4.3 Overview of version 3

The third version of FAGI-gis significantly improved the user interface, re-arranging the several panels and realizing a map-centric interface, where the entities are the main items to be visualized as geometries in the map (Figure 10). The user experience is also enhanced, by allowing the user to perform several kinds of geometry transformations, graphically on the map.
Further, according to the requirements of the DoW, several facilities for fusing with geospatial relations were added: clustering and batch handling linked entities, link discovery (recommendation) for neighborhoods of unlinked entities and batch handling of fusion actions in the level of both entities and properties. Finally, learning mechanisms were incorporated into the tool that allow the recommendation of fusion actions and of OSM categories for annotation of the fused entities. The enhancements/extensions performed during the third year of development are summarized in Table 1.

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Version 1</th>
<th>Version 2</th>
<th>Version 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fusion of spatial metadata</td>
<td>7 fusion actions</td>
<td>8 fusion actions</td>
<td>7 fusion actions +</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>map based transformations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of geometries</td>
</tr>
<tr>
<td>Fusion of non-spatial metadata</td>
<td>-</td>
<td>7 fusion actions</td>
<td>10 fusion actions</td>
</tr>
<tr>
<td>Fusion of multiple properties (m-to-n fusion)</td>
<td>-</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Fusion of property chains</td>
<td>-</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Property matching</td>
<td>-</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Link rejection</td>
<td>Supported</td>
<td>Supported</td>
<td>Obsolete</td>
</tr>
<tr>
<td>Geospatial Vocabulary support</td>
<td>GeoSPARQL</td>
<td>GeoSPARQL, Basic Geo</td>
<td>GeoSPARQL, Basic Geo</td>
</tr>
<tr>
<td>Modes</td>
<td>User Interactive</td>
<td>Batch processing, 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>
<td>Map centric web GUI, Command Line Interface</td>
</tr>
<tr>
<td>GeoKnow Generator Integration</td>
<td>-</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Geometries fusion (runtimes)</td>
<td>baseline</td>
<td>1-2 orders of magnitude faster than baseline</td>
<td>1-2 orders of magnitude faster than baseline</td>
</tr>
</tbody>
</table>
In the following sections, we provide implementation information and demonstrate the usage of the two versions of FAGI-gis.

4.3.1 Software description

4.3.1.1 Implementation information

FAGI-gis is implemented in Java and Javascript, 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.

- **LIBLINEAR [LIBL]** is a linear classifier for data with millions of instances and features and a library for solving large-scale regularized linear classification and regression. It supports (among others) two modes of multiclass classification: (a) one-vs-the rest, (b) Crammer & Singer. FAGI-gis incorporates LIBLINEAR in and utilizes the multiclass classification for recommending classes for fused entities.

- **GeoTools**. GeoTools [GeoTools] is an open source (LGPL) Java library, which provides standards compliant methods for geospatial data management comparable to those implemented in Geographical Information Systems (GIS).
- **Java Topology Suite (JTS).** The JTS Topology Suite [JTS] is an API of 2D geospatial predicates and functions, conforming to the OGC Simple Features Specification for SQL. JTS is open source (under the LGPL license) and provides a complete implementation of fundamental 2D geospatial algorithms written in Java.

- **Weka.** Weka [WEKA] is a Java library that implements several machine learning algorithms for data mining tasks. We used Weka's K-Means implementation for clustering links between entities.

### 4.3.1.2 Supported data sources and formats

**FAGI-gis** supports importing data from a 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. Basic Geo and GeoSPARQL vocabulary (with WKT serialization) are supported for representing geospatial features. **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.

### 4.3.2 Installation/Building instructions

**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 v07.2 and PostgreSQL v9.3.5 w/ PostGIS v2.1. 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.

Next, we demonstrate the usage of the map based, user interactive version of **FAGI-gis** which was the component we focused our efforts during the third year. The deployment of the batch version was described in Deliverable 3.2.2 and is also provided in Appendix Section 7.1 for completeness.

### 4.3.3 FAGI-gis for map based, user interactive fusion

Next, we demonstrate the usage of the software through the graphical user interface. The user needs to first input the connection information regarding the SPARQL endpoints containing the two datasets and the local Virtuoso and PostGIS databases (Figure 11).

![Figure 11: Configuration panels](image-url)
Upon that, the user loads the links file containing the pairs of interlinked entities. FAGI-gis allows the user to filter the links based on the entities they connect and using SPARQL. Specifically, the user is able to: (a) Choose specific classes 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) and (b) Write her own SPARQL query for limiting and specializing the set of linked entities to be loaded (Figure 12).

![Link filtering panel](image)

**Figure 12: Link filtering panel**

The next step regards property matching. Following the rules described in Deliverable 3.2.2 [GeoKnowD322], 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 yellow colour the properties of the other list (dataset) that are found to match. The final selection of the matching is performed by the user. 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.

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 13). The user can select one or more pairs of linked entities from the map (by clicking on the line segments that represent links). Next, we distinguish between single and batch fusion functionality.
4.3.3.1 Single fusion actions

When the user clicks on a link, the fusion table appears in the “Property Matching” panel (Figure 14). The user is able, for the specific pair of linked entities, to also consider other properties from the properties lists, to be fused. The user selects proper fusion actions for each pair of matched properties and then clicks on “Fuse” button. Especially for the geometries of the linked entities, the user may very easily transform them (move, rotate, re-scale) through the rendered polygons in the map interface. Then, the new fused geometry, along with all fused metadata are output to the final, fused graph. The fused geometry is rendered with a different color and the user can preview the triples written in the fused graph by clicking on the fused geometry (Figure 15).

As mentioned in previous sections, several auxiliary facilities support the described fusion process:

- If enough past fusion actions have been performed, and the system has been trained on them, the default geometry fusion action is the one that is recommended by the learning system as the most suitable one for the selected pair of linked entities.
- Under the fusion table, a list of OSM categories are recommended and the user may select any of them to be added as OWL Classes that characterize the fused entity.
- For the properties that have not been matched between the two entities, the user may select a batch copy of the triples, either from entity A or from entity B, into the fused graph, in order to characterize the fused entity.
4.3.3.2 Batch fusion actions

As described in Section 2.1, the user is able to cluster pairs of linked entities and form clusters of similarly linked entities. Then, she may select one of the clusters and perform batch fusion actions on all pairs of entities of the cluster. The process is exactly the same as described in the previous subsection. Further, the user may select manually, or by defining a bounding box, a set of pairs of linked entities and, again, perform batch fusion actions on all of them.
As far as link discovery is concerned, the user can load, by defining a bounding box, unlinked entities from an arbitrary area on the map. Upon that, she is able: (a) To select a single spatial entity and ask for link recommendations, that is find spatial entities in the neighborhood of the initial entity that might correspond to the initial entity; (b) To ask for link recommendations for the whole bounding box of unlinked entities, where the system searches for links between all contained, unlinked entities. Figure 16 demonstrates such a case where candidate links have been drawn on the map for unlinked entities. We note that the user has to validate a candidate link in order to be able to further exploit it for fused the respective entities. Finally, only one link can be validated per entity; upon that, all other candidate links are deleted for the respective entity.

![Figure 16: Link recommendations for an area of unlinked entities](image)

### 4.3.4 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.

### 4.3.5 Future work

Although this deliverable describes the work performed in Task 3.2 until M32 of the project, the development of FAGI-gis continues until the end of the project. During this third development iteration, and until M32, we mainly focused on increasing and enhancing the fusion functionality and on improving the user interaction with the tool, as described in the previous sections. Further improvements in the efficiency of the tool were postponed, marked with lower priority since efficiency and scalability reached a satisfactory level during the second development iteration (Deliverable 3.2.2).

After achieving a satisfactory level of functionality too, our next steps now include further improving the runtimes and scalability of FAGI-gis. To this end, we aim at further optimizing the ways the underlying RDF stores are queried by the tool and minimizing the magnitude of data that are transferred through the several fusion stages. Also, replacing several utilized libraries with
newer and more efficient versions and transferring computation functionality to in memory processes will also contribute to our goal.

Another direction towards improving FAGI-gis is adapting its functionality to several diverse usage scenarios. For example, we intend to examine the modifications and extensions required so that FAGI-gis can effectively handle spatial entities such as roads, administrative areas, large parks/forests, rivers, etc.

Finally, we aim to evaluate the effectiveness of the learning mechanism for recommending fusion actions and OSM categories and fine tune them for the specific problem setting.
5. Conclusions

In this deliverable, we presented the extensions and enhancements that were performed on FAGI-gis during the third 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 significantly enhancing its efficiency. Specifically, we extended and refined the initial set of fusion actions so that both spatial and non-spatial 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.

During the third year, we implemented fusion functionality on geospatial relations, such as clustering of pairs of linked entities and link discovery on neighbourhoods of unlinked entities. Several batch fusion facilities have been implemented both in the level of entities and properties. Moreover, learning mechanisms for recommending fusion actions and OSM categories for fused entities have been implemented. Finally, several facilities that improve the user interaction have been added, such as filtering of linked entities through SPARQL, map based transformations of geometries and enhancements on the visualization of (linked, unlinked and fused) entities and of clusters of entities.

Our next steps involve: (a) further improving the efficiency of the tool, (b) adapting and fine tuning FAGI-gis for several diverse datasets and (c) assessing and fine tuning the implemented recommendation mechanisms. To this end, the development of the tool will continue until the end of the GeoKnow project.
6. References


[LIBL] LIBLINEAR -- A Library for Large Linear Classification. Available at: http://www.csie.ntu.edu.tw/~cjlin/liblinear/


7. Appendix

7.1 FAGI-gis batch version deployment

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.

```
# Input and output parameters

## 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.

# 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
#wordnet_dir = /home/nick/Downloads/WordNet-3.0/dict

# Specify parameters according to the examples for your specific dataset.
# Examples given below assume an existing PostgreSQL+PostGIS database.
#linksFile = ${links_file}
#virtuosoAllowedDir = ${virt_allowed_dir}
#wordnetDir = ${wordnet_dir}
#outputGraph = ${out_graph}

# PostGIS Configuration Properties
#pg_DatabaseName = postgis1
#pg_User = postgres
#pg_Password = 1111
#pg_Import = true

pg_DatabaseName = ${pg_name}
p_User = postgres = ${pg_user}
p_Password = ${pg_pass}
p_Import = ${pg_import}
```

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).

```
# Virtuoso Configuration Properties
#vi_URL = localhost:1111
```
Finally, the selected fusion action and the respective required parameters are provided.

## 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}