Deliverable D4.4:

Cross-language and cross-domain annotation software

<table>
<thead>
<tr>
<th>Date:</th>
<th>Friday, 10. February 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s):</td>
<td>Miha Grčar (JSI)</td>
</tr>
<tr>
<td>Dissemination level:</td>
<td>PU</td>
</tr>
<tr>
<td>WP:</td>
<td>4</td>
</tr>
<tr>
<td>Version:</td>
<td>1.0</td>
</tr>
<tr>
<td>Keywords:</td>
<td>cross-language ontology querying, OntoBridge Portlet, domain ontologies, POSM, semantic annotation</td>
</tr>
<tr>
<td>Description:</td>
<td>In the context of this deliverable, we release an initial implementation of the OntoBridge Portlet (visual editor) and report on the devised semi-automatic cross-language semantic annotation infrastructure.</td>
</tr>
</tbody>
</table>
Document metadata

Quality assurors and contributors

<table>
<thead>
<tr>
<th>Quality assuror(s)</th>
<th>Nils Rune Bodsberg (SINTEF MET), Marcell Roth (UOM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contributor(s)</td>
<td>Miha Grčar (JSI), Janez Kranjc (JSI), Henry Michels (UOM), Patrick Maue (UOM), Fracois Tertre (BRGM), Nils Rune Bodsberg (SINTEF), Olaf Magnus Ostensen (NMA), Silviu Trasca (CS)</td>
</tr>
</tbody>
</table>

Version history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>24.10.2011</td>
<td>Initial TOC</td>
</tr>
<tr>
<td>0.2</td>
<td>04.12.2011</td>
<td>Initial content</td>
</tr>
<tr>
<td>0.3</td>
<td>22.12.2011</td>
<td>Responded to reviewers' comments</td>
</tr>
</tbody>
</table>
Executive Summary

This report is arranged into three conceptually different but tightly related parts.

In the first part (Section 2), we report on the cross-language ontology querying technology that participates in the semantic annotation process. The presented cross-language ontology querying technology is based on the work done in the SWING project.

In the second part, we present the software released as part of ENVISION D4.4 (Section 3). The main software “product” is the OntoBridge Portlet. The Portlet provides a graphical user interface and a set of machine learning algorithms that support the user in the annotation task. The accompanying software includes a JavaScript graph-drawing library and a .NET Web service containing the required data mining algorithms.

In the final part, we discuss a set of ENVISION domain and application ontologies and a set of POSMs (Procedure-Oriented Service Models; see ENVISION D4.3) of several Web services involved in the ENVISION use cases (see Section 3.2.2 and Appendix A).
Table of contents

EXECUTIVE SUMMARY .................................................................................................................. 3
TABLE OF CONTENTS .................................................................................................................. 4
1 INTRODUCTION ......................................................................................................................... 5
2 CROSS-LANGUAGE SEMANTIC ANNOTATIONS ........................................................................ 6
3 SOFTWARE AVAILABILITY AND USAGE INSTRUCTIONS .................................................... 16
4 USE CASE-SPECIFIC SEMANTIC RESOURCES ......................................................................... 22
5 CONCLUSIONS .......................................................................................................................... 27
APPENDIX A .................................................................................................................................... 28
REFERENCES .................................................................................................................................... 31
1 Introduction

This report is arranged into three conceptually different but tightly related parts.

In the first part (Section 2), we report on the cross-language ontology querying technology that participates in the semantic annotation process. The presented cross-language ontology querying technology is based on the work done in the SWING project (see SWING D4.3 [7]). Note that some of the content in Section 2 (i.e., the introduction and Section 2.1) was originally presented in SWING D4.3 and is given here for the reader’s convenience. However, in ENVISION, we introduce a new technique, called Stem Mapping Functions (SMF), which is more suitable for practical scenarios (Sections 2.2 and 2.3). SMF has two obvious advantages over the methods devised in SWING: (1) the translations can be obtained with a dictionary rather than a complex and expensive machine translation technology and (2) the ontology needs to be grounded in only one language.

In the second part, we present the software released as part of ENVISION D4.4 (Section 3). The main software “product” is the OntoBridge Portlet. The Portlet provides a graphical user interface and a set of machine learning algorithms that support the user in the annotation task. The accompanying software includes a JavaScript graph-drawing library and a .NET Web service containing the required data mining algorithms. The Web service was already discussed in ENVISION D4.2 (Section 3.2), therefore its description is omitted from this report.

Last but not least, we release a set of ENVISION domain and application ontologies and a set of POSMs (Procedure-Oriented Service Models; see ENVISION D4.3) of several Web services involved in the ENVISION use cases (see Section 3.2.2 and Appendix A). With this, we provide all the requirements for semantically annotating these services. Note, however, that at this time, we do not provide the corresponding semantic annotations.
2 Cross-language semantic annotations

In order to support efficient browse and search through resources and to enable efficient composition and execution of Web services, the resources need to be semantically annotated. In ENVISION, the semantic annotation is defined as a set of interlinked domain-ontology elements associated with the resource being annotated. For example, let us assume that our resource is a database table. We want to annotate its fields in order to provide compatibility with databases from other systems. Further on, let us assume that this table has a field called “employee_name” that contains employee names (as given in Figure 1, left side). On the other hand, we have a domain ontology containing knowledge and vocabulary about companies (an excerpt is given in Figure 1, right side). In order to state that our table field in fact contains employee names, we first create a placeholder for an instance of the domain-ontology concept Name and associate it with the field. We then create a placeholder for an instance of Person and link it to the placeholder for Name via the hasName relation. Finally, we create a placeholder for an instance of Company and link it to the placeholder for Person via the hasEmployee relation. Such annotation (shown in the middle in Figure 1) indeed holds the desired semantics: the annotated field contains names of people which some company employs (i.e. names of employees). Furthermore, it is possible to annotate the entire table; the semantics that should be attached to the table is “people who are employed somewhere” (see the dashed arrows in the figure).

Note that it is possible to exchange any of the placeholders with an actual instance representing a real-world entity. For example, the placeholder ?c could be replaced with an instance representing an actual company such as, for example, Microsoft ∈ Company. The annotation would then refer to “names of people employed at Microsoft”.

The annotation of a resource is a process in which the user (i.e. the domain expert) creates and interlinks domain-ontology instances and placeholders in order to create a semantic description for the resource in question. Formulating annotations in one of the ontology-description languages (e.g. WSML [1]) is not a trivial task and requires specific expertise.

For this reason, we have developed Visual OntoBridge (VOB) [2], a system that provides a graphical user interface and a set of machine learning algorithms that support the user in the annotation task. VOB allows the user to:

- Visualize the resource and the domain ontology (much like this is done in Figure 1)
- Create instance placeholders by clicking on the domain-ontology concepts
Interlink the placeholders and/or instances by “drawing” relations between them.

In addition, the user is able to enter a set of natural-language (Google-like) queries, according to which the system provides a set of non-trivial “building blocks” that can be used when defining the annotation.

The main purpose of Section 2 is to evaluate the developed querying facilities in a multilingual setting. We first discuss the ontology querying algorithm that we have developed in the European project SWING\(^1\). Further on we employ the developed algorithm in a multilingual setting and present experimental results of the cross-language ontology querying evaluation.

### 2.1 Semi-automatic annotation support

Establishing annotations manually is not a trivial task, especially if the domain ontology contains a large amount of entities and/or the user is not fully familiar with the conceptualizations in the domain ontology. VOB provides an advanced tool for querying the domain ontology with the purpose of finding the appropriate concepts and triples. A triple in this context represents two interlinked instance variables (e.g., \(\text{Company}\) hasEmployee \(\text{Person}\)) and serves as a more complex building block for defining semantic annotations.

The process of querying the domain ontology is as follows. The user enters a set of Google-like natural-language queries. The system then provides the user with two lists – the list of proposed concepts and the list of proposed triples. The user inspects the list of proposed concepts, from top to bottom, and selects relevant concepts. If the required concepts are not found at the top of the list, the user should consider reformulating the queries. Selecting a concept causes the triples to reorder: the triples containing at least one selected concept are pushed to the top of the list. The user then selects the required triples. The selected concepts and triples are transferred to the graphical annotation editor, where the user is able to revise and extend the annotation as required.

VOB employs text mining techniques, the PageRank algorithm, and consults a Web search engine to populate the two lists of recommended building blocks as described in the following subsections.

#### 2.1.1 Baseline algorithm

In [3], we presented and evaluated several term matching techniques that will serve as building blocks for automating the annotation process. To produce the two lists of recommendations as discussed in the previous section, it is possible to directly apply the term matching techniques. The algorithm is as follows:

1. Each concept and each possible domain-relation-range triple in the domain ontology is grounded through a Web search engine. Grounding a term means collecting a set of documents and assigning them to the term. In our case, the terms are the concept and relation labels in the domain ontology. With the ontology being grounded, it is possible to compare a natural-language query to the grounded domain-ontology entities. To ground a concept, the search engine\(^2\) is queried with the corresponding concept label. To ground a triple, on the other hand, the search engine is queried with the search term created by concatenating the label of the relation domain, the label of the relation, and the label of the relation range, respectively. The search terms are not put into quotes, meaning that the order

---

\(^1\) Semantic Web Service Interoperability for Geospatial Decision Making <http://www.swing-project.org/>

\(^2\) We use the Yahoo search engine through their API.
of words is not taken into account by the search engine. Each entity is grounded with 50 search result snippets.

2. The groundings are converted into TF-IDF bag-of-words vectors [8]. Each vector is labeled with the corresponding domain ontology entity (either a concept or a triple). These vectors constitute the training set (i.e. the set of labeled examples).

3. The training set is used to train the centroid classifier [4]. Each centroid is computed as the L2-normalized sum of the corresponding TF-IDF vectors.

4. The set of queries, provided by the user, is first grounded through a Web search engine. For each query, the corresponding centroid TF-IDF vector is computed. These TF-IDF vectors constitute the test set (i.e. the set of unlabeled examples).

5. Given a bag-of-words vector from the test set, the centroid classifier is employed to assign a classification score to each target class, that is, to each ontology entity (a concept or a triple). These scores are aggregated over the entire set of query vectors.

Given the set of bag-of-words vectors representing the user’s queries, the classifier is thus able to sort the domain ontology concepts and triples according to the relevance to the queries. This gives us the two required lists of annotation building blocks: the list of concepts and the list of triples.

2.1.2 Incorporating ontology structure

To establish the baseline discussed in the previous subsection, we treated the domain ontology as a flat list of entities. What we did not take into account is that these entities are in fact interlinked. This means that the domain ontology can be represented as a graph in which vertices are entities and edges represent links. In this section, we show how we can couple text similarity assessments with PageRank [5] to exploit the ontology structure for determining relevant ontology entities.

To employ PageRank, the domain ontology is first represented as a graph. This is achieved by the following recipe:

1. Represent each concept with a vertex.

2. Represent each triple $c_1$-$r$-$c_2 \in T$, where $T$ is the set of triples in the domain ontology, with two vertices: one representing $c_1$-$r$-$c_2$ and one representing the corresponding inverse relation, $c_2$-$r^{-1}$-$c_1$.

3. For each pair of concepts $c_1$, $c_2$ and for each relation $r$ such that $c_1$-$r$-$c_2 \in T$, do the following:
   
   - Connect the vertex representing $c_1$ to the vertex representing $c_1$-$r$-$c_2$ with a directed edge and weight it with $C(Q, c_1$-$r$-$c_2)$. Here, $Q = \{ q_1, q_2, q_3 \ldots \}$ is a set of natural-language queries and $C(Q, c_1$-$r$-$c_2)$ is computed as $\Sigma_{q \in Q} C(q, c_1$-$r$-$c_2)$. $C(a, b)$ refers to cosine similarity between the centroid of groundings of concept/relation $a$ and the centroid of groundings of concept/relation $b$. A centroid is computed by first converting the corresponding groundings to TF-IDF feature vectors and then computing the L2-normalized sum of these feature vectors.
   
   - Connect the vertex representing $c_1$-$r$-$c_2$ to the vertex representing $c_2$ with a directed edge and weight it with 1.
   
   - Connect the vertex representing $c_2$ to the vertex representing $c_2$-$r^{-1}$-$c_1$ with a directed edge and weight it with $C(Q, c_1$-$r$-$c_2)$.
• Connect the vertex representing $c_2-r^{-1}-c_1$ to the vertex representing $c_1$ with a directed edge and weight it with 1.

4. Represent each bag-of-words vector $q_i$ representing the test set $Q = \{q_1, q_2, q_3\ldots\}$ with a vertex. Note that the test set represents the query.

5. For each bag-of-words vector $q_i$ representing the query and each concept $c_j$, if $C(q_i, c_j) > 0$, draw a directed edge from $q_i$ to $c_j$ and weight it according to $C(q_i, c_j)$.

This process is illustrated in Figure 2. In the figure, $w_i$ represent the weights computed in Step 3 of the presented ontology-to-graph transformation process.

![Graph representation of ontologies](image)

**Figure 2. Representing ontologies as graphs.**

When the graph is created and properly weighted, we run PageRank to rank vertices (i.e. concepts and triples) according to the relevance to the query. The vertices representing the query are therefore used as the source vertices for PageRank. Note that a triple $c_1-r-C_2 \in T$ “accumulates” the ranking score in two different vertices: in the vertex representing $c_1-r-C_2$ and in the vertex representing $c_2-r^{-1}-c_1$. It is thus necessary to sum the ranking scores of these two vertices to obtain the ranking score of the corresponding triple.

Since every concept and every triple has been ranked by PageRank, we can now populate the two lists of annotation building blocks and present these to the user.

### 2.1.3 Evaluation of the semi-automatic annotation support

The presented approach was developed and successfully employed in SWING. In SWING, we were annotating geospatial Web services (called Web Feature Services or WFS [6]). More accurately, we were annotating their schemas (i.e. capability documents) to achieve semantic interoperability of services for the purpose of discovery, composition, and execution.

During the project, several WFS were annotated manually (by formulating annotations in the WSML language). For the purpose of evaluating the techniques discussed in Sections 2.1.1 and 2.1.2, we asked the domain experts at Bureau of Geological and Mining Research (BRGM, France) to provide us with natural-language queries with which they would hope to retrieve building blocks for these annotations.

We received input from three domain experts, each assigning queries to seven feature types (41 queries altogether by each of the participants). With respect to the annotations, we have identified 114 concepts and 96 triples relevant for annotating the feature types involved in the golden-standard acquisition process. Since the acquired golden standard thus contained both, the queries and the corresponding building blocks, we were able to assess the quality of the annotation algorithm by “measuring” the amount of golden-standard building blocks discovered in
the domain ontology, given a particular set of queries. We measured the area under the Receiver Operating Characteristic (ROC) curve\(^3\) to evaluate the lists produced by the algorithm.

We first evaluated the baseline algorithm discussed in Section 2.1.1 to establish the baselines and determine a setting in which the baseline algorithm performs best. Based on the experiments, we concluded that the concepts should be grounded with 50 documents each, so should the queries when used to rank the concepts. On the other hand, the triples should be grounded with only 10 documents each and the queries should not be grounded when used to rank the triples. Under these conditions, the baseline achieved 91.47% AUC (i.e. area under ROC curve) when ranking concepts, and 93.16% AUC when ranking triples. Note that the structure of the ontology was not taken into account here.

We further evaluated the graph-based algorithm discussed in Section 2.1.2. We showed that it indeed significantly outperforms the baseline algorithm. The most important parameter to tune was the PageRank damping factor. We experimented with the damping factor values 0.2, 0.4, 0.6, 0.8, and 0.9. The results are shown in Figure 3.

![Figure 3: Evaluation results for the two lists of proposed ontology entities.](image)

Through the evaluation of the graph-based algorithm, we learned the following:

- Grounding the queries helps to rank both, the concepts and the triples.
- The concepts, triples, and queries should be grounded with about 50 documents each.
- The damping factor should be set to 0.6 for the concepts and 0.8 for the triples. This means that we can either run PageRank twice or set the damping factor to 0.7 to increase the speed at the slight expense of quality on both sides.

The rewarding fact is that we managed to significantly beat the baselines. We increased the average AUC for 5.48% on concepts and for 3.18% on triples. This presents a big difference from the application point of view. Roughly speaking, the graph-based algorithms are twice as good as the baseline algorithm. Also, the user is able to interact with the system and reformulate queries to achieve even better results. To support this claim, we computed the average AUC by taking, for each annotation, only the most successful annotator into account (i.e. the annotator that formulated the query yielding the highest AUC). The average AUC on the triples rose to 98.15%.

---

This is yet again roughly twice as good as when considering all the annotators and their queries. The high AUC achieved in the evaluation process was also reflected in practice.

2.2 Cross-language experiments

The core idea of the machine-aided annotation in SWING and thus also in ENVISION is based on term matching through groundings obtained by a Web search engine.

The problem occurs when the query is in a different language than the ontology. Since the cosine similarity between two documents is based on the mutual term occurrence, it is not possible to compare two terms (their groundings, to be exact) that are not in the same language. If the two documents are in different languages, almost none of the relevant terms occur in both documents. This situation occurs when the user formulates a natural-language query in his/her native language that is different than that of the ontology.

In this section, we discuss how this problem can be resolved by using statistical machine translation (SMT\(^4\)) and/or translation dictionaries. In SWING, we were required to support English and French, and we also acquired the golden standard in both these two languages. There are at least two ways to apply SMT to breach the language barrier\(^5\). In the first, we use SMT to define a function that maps French word stems into English word stems. The French query is first grounded through a French search engine (in particular, we limit the search engine to French pages) and the French groundings are then transformed into English feature vectors by employing the stem mapping function as illustrated in Figure 4. Figure 5, on the other hand, illustrates the second SMT approach. Here, SMT is only used for translating the concept and relation labels to French and the French groundings are obtained by querying a French search engine. Having the labels translated, the user can view the ontology in French. Note that in both cases, we do not need to use a full-blown SMT as we are only translating words and phrases; we could use a dictionary coupled with a light-weight language model and/or a set of heuristics.

---

\(^4\) This should not be confused with the Service Model Translator (SMT) used in the ENVISION infrastructure.

\(^5\) A third option is presented in [7].
Since the computation and use of the stem mapping function requires more explanation, we discuss the details in the following subsection.

2.2.1 Stem mapping functions (SMF)

To compute the stem mapping function, we employ the following procedure:

1. The entire corpus of English groundings is converted into a set of TF-IDF vectors; this yields a bag-of-words space with the corresponding vocabulary. The vocabulary contains stems and the corresponding most frequent forms of all words contained in the TF-IDF vectors.

2. Each unigram from the vocabulary, more specifically its most frequent form, is translated into French by employing SMT and/or resorting to a dictionary. The translation process gives us a table that maps each English unigram to a French term (not necessarily a unigram). An example is given in Figure 6 (a).

3. A simple heuristics that converts all non-unigrams to unigrams is employed. If a term is not a unigram, it is tokenized, French stop words are removed, and only the first remaining unigram token\(^6\) is considered. The effect of this heuristics is shown in Figure 6 (b).

4. The table is inverted and stemming is applied on both sides; French unigrams are stemmed with the French stemmer, English unigrams with the English stemmer.

5. If a French stem maps into two different English stems, only the mapping that corresponds to the most frequent English stem is considered. The final mapping function is illustrated in Figure 6 (c).

<table>
<thead>
<tr>
<th>English Concept Names</th>
<th>French Concept Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>hallucinogenic</td>
<td>hallucinogène</td>
</tr>
<tr>
<td>abortion</td>
<td>avortement</td>
</tr>
<tr>
<td>doomsday</td>
<td>jour du jugement dernier</td>
</tr>
<tr>
<td>archnet</td>
<td>archnet</td>
</tr>
<tr>
<td>gba</td>
<td>gba</td>
</tr>
<tr>
<td>warfreak</td>
<td>warfreak</td>
</tr>
<tr>
<td>charisma</td>
<td>charisme</td>
</tr>
<tr>
<td>oilwatch</td>
<td>oilwatch</td>
</tr>
</tbody>
</table>

\(^6\) The intuition behind this is that French terms normally start with a noun, followed by one or more adjectives.
Figure 6. Computation of the stem mapping function, a step-by-step illustration.

The stem mapping function enables us to convert a French (query) document into the corresponding English TF-IDF vector that can be used in the querying process. The process is as follows:

1. The French document is tokenized and French stop words are removed.
2. Instead of employing the usual stemming process, each French unigram is first stemmed with the French stemmer and then mapped into the corresponding English stem as defined by the stem mapping function (i.e. mapping table). If the French stem is not found in the table, the corresponding French unigram is stemmed with the English stemmer instead. In this case, the overlap between the languages (i.e. words that are written in the same way and bear the same semantics) is exploited.
3. The English stems are grouped into n-grams as customary in the document preprocessing procedure. Note that this mostly yields English n-gram stems that are not found in the English bag-of-words vocabulary. The reason for this is that adjective-noun pairs are written in the opposite word order in French. In this work, we do not attempt to resolve this issue.
4. Each English unigram or n-gram stem is looked up in the BOW vocabulary. If it is found, its IDF value and index are retrieved and the corresponding TF-IDF weight is computed. On the other hand, if the stem is not found in the vocabulary, it is simply ignored (this is a normal behavior when projecting documents into a predefined BOW space; this situation may occur even when dealing with one single language). This step results in an English TF-IDF vector that can be used in the ontology querying process.

2.2.2 Cross-language querying evaluation

When dealing with one single language, we defined the baseline by classifying the query documents into a set of target classes representing the ontology entities (i.e. concepts and relation triples). We can do a similar thing in the cross-language scenario by employing SMT and/or translation dictionaries as explained in Section 2.2. We thus end up with two baselines,

---

7 This mapping will not be in effect as the French stem jour is mapped into the English stem day more frequently.
one based on the translation of labels (termed “BL-TOL”) and the other based on the stem mapping function (termed “BL-SMF”).

In addition to these two baselines, we define another baseline which avoids the translation process and simply employs the English stemmer when processing French query documents. In this setting, we rely on the overlap between the two languages in terms of words that are written in the same way in both languages and have the same meaning. We denote this baseline with “BL-O” as in “baseline based on overlap”.

We used the same evaluation setting as in the original experiments but this time we took the French natural-language queries into consideration. We first evaluated the three baselines presented earlier in this section. The results are shown in Figure 7.

From the figure, it is possible to conclude that BL-O is indeed outperformed by the other two baseline methods but even so exhibits a relatively high AUC score given its simplicity. This means that there is a significant overlap between words found in English groundings and those found in French query documents. As expected, there is a substantial improvement over BL-O when employing the stem mapping function (BL-SMF) or translating labels (BL-TOL) even in the baseline setting. These two baselines perform comparably well and constitute the baselines for the experiments that take the structure into account.

The experiments in which the structure was considered were conducted in the same way as described in Sections 2.1.2 and 2.1.3, but this time, we employed either the stem mapping function (termed “SMF”) or translation of labels (termed “TOL”) to overcome the language barrier in an attempt to respond to French queries. The results are shown in Figures 8 and 9.
Figure 8. Evaluation results for proposing concepts in a cross-language scenario.

Figure 9. Evaluation results for proposing triples in a cross-language scenario.

From the figures, it is evident that the structure-based approaches clearly outperform the baselines. The AUC on concepts is increased for 7.22% and the AUC on triples for 7.45%. The best results are achieved at the same damping factor values as in the monolingual setting (see Section 2.1.3), that is, at 0.6 for concepts and 0.8 for triples. Furthermore, we can see that TOL outperforms SMF in both tasks. However, the difference between the two is less than 1%. Note that SMF has two obvious advantages over TOL. Firstly, since only unigrams need to be translated in the SMF approach, the translations can be obtained with a dictionary rather than a full-blown SMT. On the other hand, since the labels can be rather complex phrases, using only a dictionary without a language model and/or heuristics is not possible in the TOL approach. Secondly, the ontology needs to be grounded in only one language (in our case, English). This makes the maintenance and storage of groundings more feasible. In the TOL approach, the ontology needs to be grounded in every language which is to be supported for ontology querying. From this perspective and given the small difference in performance between SMF and TOL, SMF can be advantageous over TOL in many practical settings.
2.3 Conclusions on the cross-language ontology querying

In order to support efficient browse and search through resources and to enable efficient composition and execution of Web services, the resources need to be semantically annotated, i.e. interlinked with a set of domain-ontology elements. Formulating annotations in one of the formal languages is not a trivial task and requires specific expertise. For this reason, we have developed a system that allows the user to pose natural-language queries to retrieve ontology elements that serve as building blocks.

We established the baselines by employing a typical text mining approach, training a classifier on a set of TF-IDF vectors. We further developed and evaluated a graph-based algorithm and managed to significantly outperform the baseline. We increased the average AUC (Area Under ROC Curve) for 5.48% on concepts (from 91.47% to 96.94%) and for 3.18% on triples (from 93.16% to 96.34%). Roughly speaking, the graph-based algorithm is twice as good as the baseline algorithm.

In addition, we evaluated the developed querying facilities in a multilingual setting. Specifically, we measured AUC for retrieving concepts and triples from an English ontology given natural-language queries in French. We designed and evaluated two relatively simple approaches termed TOL (Translation Of Labels) and SMF (Stem Mapping Function). Both approaches clearly outperformed the established baselines. The AUC on concepts was increased for 7.22% (from 89.10% to 96.32%) and the AUC on triples for 7.45% (from 88.23% to 95.68%). The difference in AUC between querying the ontology in its native language (English) and querying it in a foreign language (French) is relatively low. This means that not much has been lost in translation and that the developed approaches can be applied in many practical scenarios.

TOL outperformed SMF for less than 1%. However, SMF has two obvious advantages over TOL: (1) since only unigrams need to be translated, the translations can be obtained with a dictionary rather than a full-blown SMT, and (2) the ontology needs to be grounded in only one language. From this perspective, SMF can be advantageous over TOL in many practical settings.

3 Software availability and usage instructions

In the following sections, we describe the software that we release in the context of this report. The main software “product” is the OntoBridge Portlet which is presented in Section 3.2.1. The accompanying software includes a JavaScript graph-drawing library (see Section 3.2.2) and a .NET Web service containing the required data mining algorithms (see ENVISION D4.2, Section 3.2). The architecture of the OntoBridge Portlet, from a high-level perspective, is shown in Figure 10.
3.1 Availability

The software (released as part of ENVISION D4.4) is available at the following locations:

- OntoBridge Portlet download URL (WAR):
  http://kenai.com/projects/envision/downloads/download/OntoBridge-0.0.1-SNAPSHOT.war
- OntoBridge Portlet SVN URL (source):
  https://svn.kenai.com/svn/envision~portal/portlets/OntoBridge
- Graph drawing JavaScript library SVN URL (source):
  https://svn.kenai.com/svn/envision~portal/api/js-OnontologyVisualization
- Data mining C# Web service SVN URL (source):
  https://svn.kenai.com/svn/envision~runtime/services/OntoBridgeService

3.2 Usage instructions

In the following subsections, we give usage instructions for the OntoBridge Portlet and JavaScript graph-drawing library, respectively. On the other hand, the API of the .NET Web service providing data mining functionality is described in ENVISION D4.2, Section 3.2.

3.2.1 OntoBridge Portlet

The main software “product” that we release in the context of ENVISION D4.4 is the OntoBridge Portlet. In this section, we give brief instructions on how to use the OntoBridge Portlet. The instructions are given under the following two assumptions:

1. A POSM of an OGC or WSDL-compliant Web service has been selected in the Resource Manager Portlet. It represents the service or composition that we want to semantically annotate.
2. A domain ontology (or a set of domain ontologies), containing the appropriate real-world conceptualizations, has also been selected in the Resource Manager Portlet.

The process of annotating the selected resource with the conceptualizations from the selected domain ontology is as follows:

1. The user is initially presented with a blank [pop-up] window. By pressing the “Search” button, he issues a Google-like search query which allows him to select the central ontology concept. The user can also set the radius which specifies how much context around the central concept should be visualized.

2. The user clicks on “Annotation Window” to open the annotation window. In the annotation window, the resource is visualized as a set of interlinked objects from the corresponding POSM description. The service is represented with the central node and the other nodes represent its attributes (see Figure 11).

![Annotation Diagram]

Figure 11. Annotation window showing the resource as a graph.
3. The user double-clicks on a concept in the domain ontology (in the main application window) to create its instance placeholder (i.e., instance variable) in the annotation window. This is illustrated in Figure 12. The user creates as many placeholders as required for the annotation. It is of course possible to create several placeholders for the same concept. In this case, a sequential number is displayed in brackets as part of the label.

![Double-click on a concept in the domain ontology](image)

**Figure 12. Creating instance placeholders.**
4. To relate one instance placeholder to another, the user first selects “Connect” from the context menu of the first placeholder (the mouse pointer changes to crosshair) and then clicks on the second placeholder. The user is provided with the list of all possible relations between the two placeholders (more accurately: between the two corresponding instances). The selected relation is established between the two placeholders and visualized as an arrow. This is shown in Figure 13.

Figure 13. Relating two instance placeholders.
5. To define a complete annotation, the placeholders for the domain ontology instances need to be related to the schema objects. To establish these relations, the user first selects "Connect" from the context menu of a placeholder and then clicks on the appropriate schema object (or vice versa). This relates the placeholder to the schema object through the "annotate" relation (visualized as a red arrow). This is shown in Figure 14.

![Figure 14](image)

**Figure 14. Relating a resource element to an instance placeholder.**

6. By selecting “Save” in the annotation window, the semantic annotation is inserted into the corresponding POSM description and sent back to the Resource Manager. The semantic annotation for the example from Figure 14 is as follows:

```xml
?var1, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var2, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var3, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var4, "http://swing.uni-muenster.de/core/Swing/0.2#hasGeometry" hasValue
?var5, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var6, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var7, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var8, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var9, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
?var10, "http://swing.uni-muenster.de/core/Swing/0.2#hasProperty" hasValue
```
3.2.2 Graph drawing JavaScript library

In the context of this report, we release a JavaScript library for drawing and layouting graphs. This library is used in Visual OntoBridge to visualize ontologies and resources (i.e., Web service schemas). Ontologies are “converted” into graphs by following the procedure devised in SWING and presented in SWING D4.5 (Section 4.3). In this section, we give a short tutorial on how to use this library.

The library is a jQuery plugin, and as such can be instantiated on any block-level HTML element. In order to initialize the graph, the graph-drawing library, jQuery library, and jQuery UI library need to be included in the head of the HTML document:

```html
<script type="text/javascript" src="js/jquery-1.6.1.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.8.16.custom.min.js"></script>
<script type="text/javascript" src="js/graph.jquery.js"></script>
```

The body of the document requires an empty `div` tag that can be selected using the jQuery selectors. This can be achieved by providing a unique ID for the tag:

```html
<div id="graph"></div>
```

The graph can be initialized at any time using a JavaScript call (usually inside the jQuery document-ready function). The graph can be initialized with a set of custom options which provide the ability to style the appearance of the edges (e.g., width, color) and alter the basic functionalities of the graph itself (whether nodes can be dragged or not, whether can nodes
breach the boundaries set by the containing div element, whether custom context menus are enabled, etc.).

The list of options is as follows:

<table>
<thead>
<tr>
<th>Option name</th>
<th>Value type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linewidth</td>
<td>Integer</td>
<td>Width of lines representing edges.</td>
</tr>
<tr>
<td>Borderwidth</td>
<td>Integer</td>
<td>Width of borders of lines representing edges.</td>
</tr>
<tr>
<td>Linecolor</td>
<td>String</td>
<td>Color of lines representing edges.</td>
</tr>
<tr>
<td>Bordercolor</td>
<td>String</td>
<td>Color of border of lines representing edges.</td>
</tr>
<tr>
<td>selectedbordercolor</td>
<td>String</td>
<td>Color of selected lines.</td>
</tr>
<tr>
<td>selectedlinecolor</td>
<td>String</td>
<td>Color of borders of selected lines.</td>
</tr>
<tr>
<td>selectedborderwidth</td>
<td>Integer</td>
<td>Width of selected lines.</td>
</tr>
<tr>
<td>selectedlinewidth</td>
<td>Integer</td>
<td>Width of borders of selected lines.</td>
</tr>
<tr>
<td>Draggable</td>
<td>Boolean</td>
<td>If true, the user can drag and drop the nodes at will.</td>
</tr>
<tr>
<td>Directed</td>
<td>Boolean</td>
<td>Directed or undirected graph.</td>
</tr>
<tr>
<td>Contained</td>
<td>Boolean</td>
<td>If false, the nodes can be positioned outside the contained div element.</td>
</tr>
<tr>
<td>edgeLabels</td>
<td>Boolean</td>
<td>Display edge labels on edges when selected.</td>
</tr>
<tr>
<td>nodeEdgeLabels</td>
<td>Boolean</td>
<td>Display edge labels on nodes when selected.</td>
</tr>
<tr>
<td>edgeLabelsPersist</td>
<td>Boolean</td>
<td>Display edge labels on edges at all times.</td>
</tr>
<tr>
<td>Arrowonmiddle</td>
<td>Boolean</td>
<td>If true, the arrow determining the direction of an edge is rendered in the middle of the line, otherwise at the intersection of the line and the node.</td>
</tr>
<tr>
<td>Noselectedcolor</td>
<td>Boolean</td>
<td>If true, the regular styling is used for selected edges.</td>
</tr>
<tr>
<td>Contextmenu</td>
<td>Boolean</td>
<td>Enables custom context menus (requires the jQuery.contextmenu plugin).</td>
</tr>
</tbody>
</table>

We provide an example of the initialization of a graph with arrows on the intersections of lines and nodes and with labels on edges disabled:

```javascript
$("#graph").interactiveGraph({'arrowonmiddle':false,
                                 'edgeLabels':false});
```

The library offers several methods to add nodes to the graph. The simplest to use is the addNode method which requires the label of the node and its starting position. In this example, we add three nodes to the graph:
node1 = $('#graph').interactiveGraph('addNode', 'Node 1', 100, 200);
node2 = $('#graph').interactiveGraph('addNode', 'Node 2', 200, 100);
node3 = $('#graph').interactiveGraph('addNode', 'Node 3', 200, 200);

The simplest method for connecting nodes is the `connect` method which requires two node objects (returned by the `addNode` method) as input. In this example, we connect `Node 1` to `Node 2` and `Node 3`:

```javascript
$('#graph').interactiveGraph('connect', node1, node2, 'is connected to');
$('#graph').interactiveGraph('connect', node1, node3, 'is connected to');
```

To invoke the Fruchterman-Reingold layouting algorithm [9], the `forceLayout` method must be called:

```javascript
$('#graph').interactiveGraph('forceLayout');
```

Since the nodes are rendered as `div` elements, their appearance can be styled with Cascade Style Sheets (CSS). To display nodes with black borders and red background, the following code should be included in the CSS file:

```css
div#graph div.graph_node {
    text-align:left;
    float:left;
    background-color:#ffaaaa;
    border:1px solid black;
}
```

The following elements can be used for styling: `div.graph_node` (node appearance), `div.graph_edgeLabel` (the labels as they appear on the edges), `div.graph_node_hovered` (this should be used instead of the `hover` selector), and `div.graph_node span` (used for styling edge labels on nodes).

![Figure 15. Graph visualized by the code in this tutorial.](image-url)
4 Use case-specific semantic resources

In the context of this report, we release several UC-specific resources, i.e., domain ontologies and POSM models of several UC-related Web services.

4.1 Domain ontologies

The ontologies for two of the three ENVISION use cases (i.e., landslide use case and oil spill use case) are available at the following address:


The domain knowledge is organized hierarchically, forming a dependency structure. DOLCE is the basis for all other ontologies, extending into Observation, Patterns, and Geographic Space. The UC-related ontologies (termed “Scenario-specific Domain Ontologies”) and the ontologies describing the relevant OGC and WSMO models (termed “Application Ontologies”) are built on top of these ontologies. A high-level view on the ENVISION knowledge model is given in Figure 16.

![Figure 16. High-level view on the ENVISION knowledge model.](image)

The role of these ontologies in the annotation process is two-fold. On one hand, they are used to model the resource (white nodes in Figure 14) and on the other, they are used to semantically describe the resource and its attributes (yellow nodes in Figure 14). The “two worlds” are joined via the “annotate” relation (conceptually illustrated with the red arrow in Figure 14).

A complete list of the ENVISION ontologies (and their dependencies) available at Month 24 is given in Appendix A.

4.2 Web services and POSMs

In the annotation process, an OGC/WSDL service is first translated into its POSM representation. POSM schemas are essentially RDF documents, containing elements from the application ontologies. As already explained, the semantic annotation (consisting of the domain-ontology elements) is attached to the POSM elements through the “annotate” relation. More information about this process can be found in ENVISION D4.2, Section 2.

As part of this report, we release a set of ENVISION domain and application ontologies (Section 4.1) and a portlet for creating semantic annotations (Section 3.2.1). In addition, we release a set of POSM models corresponding to relevant Web services involved in the ENVISION use cases.
With this, we provide all the prerequisites for semantically annotating these services. Note, however, that at this time, we do not provide the corresponding semantic annotations. The annotations will be created in collaboration with the service providers as the project progresses.

### 4.2.1 Landslide use case

The landslide use case description can be found at the following locations:

- **Additional information:** [http://pad.ifgi.de/envision-workflow-landslide](http://pad.ifgi.de/envision-workflow-landslide)

The following services participate in the landslide use case scenario:

- **SOS** providing the precipitation data for the area RD23
- **WCS** providing the Digital Elevation Model (DEM) for Guadeloupe
- **WFS** providing the "elements at risk" (e.g., roads) for the area RD23\(^8\)
- **WMS/WFS** providing the geology of Guadeloupe
- **WPS** simulating the main water cycle in a catchment basin (precipitation, evapotranspiration, infiltration, underground flow)
- **WPS** estimating the probability of a landslide; the computation is based on a 2D static mechanical analysis, using the method of slices, carried out on every topographical profile (cross-section of a hill slope) over the study area
- **WPS** representing the run-out model, assessing the probability of a given point being reached by a landslide\(^6\)
- **WPS** estimating the probability of a linear element or its sections being damaged by a landslide\(^8\)

The POSM models for the following OGC Web services related to the landslide use case are currently available:

- **SOS** providing the precipitation data for the area RD23
  
  
  
  **POSM:** [http://first-vm2.ijs.si/envision/res/Precipitation.rdf](http://first-vm2.ijs.si/envision/res/Precipitation.rdf)

- **WCS** providing the Digital Elevation Model (DEM) for Guadeloupe
  
  **POSM:** [http://first-vm2.ijs.si/envision/res/SRTM3.rdf](http://first-vm2.ijs.si/envision/res/SRTM3.rdf)
  
  **Annotated POSM:** [http://first-vm2.ijs.si/envision/res/SRTM3_annotated.rdf](http://first-vm2.ijs.si/envision/res/SRTM3_annotated.rdf)

- **WMS/WFS** providing the geology of Guadeloupe
  
  **Service:** [http://mapdmzrec.brgm.fr/cgi-bin/mapserv54?map=/carto/envision/mapFiles/GLP_Geology.map](http://mapdmzrec.brgm.fr/cgi-bin/mapserv54?map=/carto/envision/mapFiles/GLP_Geology.map)
  
  **Geology (DescribeFeatureType):** [http://mapdmzrec.brgm.fr/cgi-bin/mapserv54?map=/carto/envision/mapFiles/GLP_Geology.map&SERVICE=wfs&VERSION=1.0.0&REQUEST=DescribeFeatureType&typename=GEOL50K](http://mapdmzrec.brgm.fr/cgi-bin/mapserv54?map=/carto/envision/mapFiles/GLP_Geology.map&SERVICE=wfs&VERSION=1.0.0&REQUEST=DescribeFeatureType&typename=GEOL50K)
  
  **POSM:** [http://first-vm2.ijs.si/envision/res/Geol50k.rdf](http://first-vm2.ijs.si/envision/res/Geol50k.rdf)

---

\(^8\) Web service still under development at this time.
4.2.2 Oil spill use case

The oil spill use case description can be found at the following locations:

- Additional information: [http://pad.ifgi.de/envision-workflow-oilspill](http://pad.ifgi.de/envision-workflow-oilspill)

Due to the unavailability of the services participating in the oil spill process, the POSM models for the oil spill use case are not available at the moment. The following services participate in the oil spill use case scenario:

- WPS predicting the drift of an oil spill
  (Available as WSDL: [http://giv-wfs.uni-muenster.de/pod-ifgi/services?wsdl](http://giv-wfs.uni-muenster.de/pod-ifgi/services?wsdl))
- WPS predicting the bio-accumulation and lethality among the cod exposed to an oil spill
- WCS providing the sea depth data
- WFS providing the coastline data for the Norwegian mainland

These services are expected to become available later in the course of the project.

4.2.3 Flood monitoring use case

Due to a late start of the corresponding WP, this use case is underdeveloped at the moment. The information on its services, models, and domain knowledge resources will be provided at a later stage.

5 Conclusions

In the context of this report, we released an initial implementation of the OntoBridge Portlet (visual editor) and reported on the devised semi-automatic cross-language semantic annotation infrastructure.

To enable cross-language ontology querying, we introduced a new technique, called Stem Mapping Functions (SMF). SMF has two obvious advantages over the methods devised in SWING: (1) the translations can be obtained with a dictionary rather than a complex and expensive machine translation technology and (2) the ontology needs to be grounded in only one language.
The released OntoBridge Portlet provides a graphical user interface and a set of machine learning algorithms that support the user in the annotation task. The accompanying software includes a JavaScript graph-drawing library and a .NET Web service containing the required data mining algorithms.

Last but not least, the released semantic resources and models satisfy all the requirements for creating semantic annotations for the services that participate in the ENVISION use cases.

**Appendix A**

This appendix gives a complete list of the ENVISION ontologies (and their dependencies) available at Month 24.

**Global Ontologies**

**DOLCE**

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Namespace</th>
<th>Location</th>
</tr>
</thead>
</table>

**Geographic Space**

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Namespace</th>
<th>Location</th>
</tr>
</thead>
</table>
## Geographic Systems

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Namespace</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>geographic_units</td>
<td><a href="http://purl.org/ifgi/geounits">http://purl.org/ifgi/geounits</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/global/Geographic-Space/geographic-units/0.2/geographic-units.rdf">http://wsmls.googlecode.com/svn/trunk/global/Geographic-Space/geographic-units/0.2/geographic-units.rdf</a></td>
</tr>
</tbody>
</table>

## Observation

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Namespace</th>
<th>Location</th>
</tr>
</thead>
</table>

## Patterns

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Namespace</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role</td>
<td><a href="http://purl.org/ifgi/patterns/role">http://purl.org/ifgi/patterns/role</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/global/Patterns/Role/0.1/Role.rdf">http://wsmls.googlecode.com/svn/trunk/global/Patterns/Role/0.1/Role.rdf</a></td>
</tr>
<tr>
<td>System</td>
<td><a href="http://purl.org/ifgi/patterns/system">http://purl.org/ifgi/patterns/system</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/global/Patterns/System/0.1/System.rdf">http://wsmls.googlecode.com/svn/trunk/global/Patterns/System/0.1/System.rdf</a></td>
</tr>
</tbody>
</table>

## Scenario-specific Domain Ontologies

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Namespace</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>OilSpill</td>
<td><a href="http://purl.org/ifgi/oilspill">http://purl.org/ifgi/oilspill</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/local/oilspill/0.1/oilspill.rdf">http://wsmls.googlecode.com/svn/trunk/local/oilspill/0.1/oilspill.rdf</a></td>
</tr>
<tr>
<td>Ontology</td>
<td>Namespace</td>
<td>Location</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GML</td>
<td><a href="http://purl.org/ifgi/gml/0.2">http://purl.org/ifgi/gml/0.2</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/OGC/GML/0.4/GML.rdf">http://wsmls.googlecode.com/svn/trunk/application/OGC/GML/0.4/GML.rdf</a></td>
</tr>
<tr>
<td>OWS</td>
<td><a href="http://purl.org/ifgi/ows/0.2">http://purl.org/ifgi/ows/0.2</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/OGC/OWS/0.2/OWS.rdf">http://wsmls.googlecode.com/svn/trunk/application/OGC/OWS/0.2/OWS.rdf</a></td>
</tr>
<tr>
<td>SML</td>
<td><a href="http://purl.org/ifgi/sml">http://purl.org/ifgi/sml</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/OGC/SML/0.2/SML.rdf">http://wsmls.googlecode.com/svn/trunk/application/OGC/SML/0.2/SML.rdf</a></td>
</tr>
<tr>
<td>SWE</td>
<td><a href="http://purl.org/ifgi/swe">http://purl.org/ifgi/swe</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/OGC/SWE/0.2/SWE.rdf">http://wsmls.googlecode.com/svn/trunk/application/OGC/SWE/0.2/SWE.rdf</a></td>
</tr>
<tr>
<td>WFS</td>
<td><a href="http://purl.org/ifgi/wfs">http://purl.org/ifgi/wfs</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/OGC/WFS/0.2/WFS.rdf">http://wsmls.googlecode.com/svn/trunk/application/OGC/WFS/0.2/WFS.rdf</a></td>
</tr>
</tbody>
</table>

**Application Ontologies**

**OGC**

*WMSO*
<table>
<thead>
<tr>
<th>Ontology</th>
<th>Namespace</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSM</td>
<td><a href="http://purl.org/ifgi/posm">http://purl.org/ifgi/posm</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/WSMO/POSM/0.2/POSM.rdf">http://wsmls.googlecode.com/svn/trunk/application/WSMO/POSM/0.2/POSM.rdf</a></td>
</tr>
<tr>
<td>WSL</td>
<td><a href="http://purl.org/ifgi/wsl">http://purl.org/ifgi/wsl</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/WSMO/WSL/0.1/WSL.rdf">http://wsmls.googlecode.com/svn/trunk/application/WSMO/WSL/0.1/WSL.rdf</a></td>
</tr>
<tr>
<td>Annotation</td>
<td><a href="http://purl.org/ifgi/annotations">http://purl.org/ifgi/annotations</a></td>
<td><a href="http://wsmls.googlecode.com/svn/trunk/application/Annotation/Annotation.rdf">http://wsmls.googlecode.com/svn/trunk/application/Annotation/Annotation.rdf</a></td>
</tr>
</tbody>
</table>

**References**


