MAMA: A Novel Approach to Ontology Mapping

Shi-Kuo Chang¹, Francesco Colace², Massimo De Santo², Emilio Zegarra¹, YongJun Qie¹

¹Department of Computer Science, University of Pittsburgh Pittsburgh, PA 15260 USA
²DIEII - Department of Electronic and Information Engineering, University of Salerno, Italy
{chang, ezegarra}@cs.pitt.edu, yongjun@pitt.edu,{fcolace, desanto}@unisa.it

Abstract

Ontology is a key factor for enabling interoperability across heterogeneous systems, services and users. One of the most challenging tasks in its use is ontology construction. However, experts usually represent the same knowledge domain by the use of different ontologies that can differ in structure, concepts, relations and attributes. Therefore, ontology could lose its main feature: allowing the semantic interoperability among actors working on the same knowledge domain. An effective solution to this problem is the introduction of methods for finding mapping among the various components of ontologies. In this paper, a novel approach to the ontology mapping is proposed. The proposed approach, named MAMA, investigates the possibility of finding overlaps among different ontologies that describe the same knowledge domain through the combined use of syntactic, semantic and topological similarity indexes. The indexes are combined to define the degree of similarity between the various components of ontologies by introducing a combining rule. This rule is adaptive and automatically emphasizes the contributions of those indexes that provide better results in certain operative conditions. The proposed approach has been tested on standard datasets and the obtained results are very promising. Moreover, we are currently exploring the application of the ontology mapping approach to software component reuse.

1. Introduction

Distributed systems are by their nature heterogeneous and so the various players in the system need tools that allow them to share services and resources in it. In this complex scenario, an effective solution is the adoption of the ontology formalism. The term ‘ontology’ was first used in the computer science field by Gruber referring to an explicit specification of a conceptualization [13]. So an ontology is defined as a formal, explicit specification of a shared conceptualization [11] and plays an important role in the domains of software engineering, artificial intelligence, knowledge management, information integration and semantic web [2]. Ontology is playing a significant role in information systems, semantic web and knowledge-based systems, where ‘ontology’ refers to “the representation of meaning of terms in vocabularies and the relationships between those terms” [1]. The role of ontology, then, is to represent the main components of a knowledge domain according to the view that a generic user has about it. However, for the same knowledge domain each user may create a different ontology, which differs in structure, concepts, attributes and relationships from the other ones. In other words, an ontology, while describing the same domain of knowledge, may be different from another because both contain the point of view of a particular user, and how it was built. Therefore, ontology could lose its main feature: allowing the semantic interoperability among actors working on the same knowledge domain [12].

An effective solution to this problem is the introduction of methods for finding mapping among the various components of ontologies. The main approaches to the ontology mapping can be grouped into three categories according to the pursued approach, namely: the lexical category, the semantic category and the structural category. In the literature, many approaches in these three categories have been developed, but none is appropriate for ontology mapping. Each of the proposed approach, in fact, shows good performance in certain situations, while exhibiting weaknesses in others. An approach in the structural category, for example, works well when ontologies shows similar structure, but is unable to recognize semantic similarities related the name of the concepts.

In this paper, a novel methodology for ontology mapping is proposed. The starting point of the proposed methodology is the assumption that the previous approaches can be complementary and so they can reach better results when working together. The contributions of various approaches are combined by an adaptively weighted rule in order to emphasize those that work better in the various scenarios. In Section 2 the proposed ontology mapping approach, named MAMA, will be described. In Section 3 the initial experimental results obtained by the use of the proposed approach on standard datasets will be discussed. In Section 4 we explore the application of ontology mapping to software component reuse, which is an important issue in software engineering. Some initial experimental results in software component reuse by ontology mapping will be presented.
2. The Proposed Ontology Mapping Approach

What is an ontology? This question could seem trivial but it is still difficult to give a unique definition. An ontology’s formal definition could be the following [14]:

\[ O = \{ C, H_c, H_r, A, I \} \]

- \( C \) is the set of concepts in a domain
- \( H_c \) is the set of generic relations among concepts
- \( H_r \) is the set of the hierarchical relations among concepts
- \( A \) is the set of axioms
- \( I \) is the set of concepts’ instances

Keeping in mind the above definition, ontology mapping is an operation that can be described as follows: given two ontologies \( A \) and \( B \), mapping one ontology to another means that for each concept (node) in ontology \( A \), a corresponding concept (node) in ontology \( B \) is found, which has the same or similar semantics in ontology \( B \) and vice versa [15].

According to this definition, an ontology mapping function can be defined as follows: mapping one ontology to another means that for each concept (node) in ontology \( A \), a corresponding concept (node) in ontology \( B \) is found, which has the same or similar semantics in ontology \( B \) and vice versa [15].

Therefore the mapping process could be seen as a classification task. In particular, the proposed algorithm classifies the similarities among the classes, the properties and the relationships and then creates a new ontology that will be the common layer capable of bridging the various ontologies.

As previously said, various indexes are introduced in order to calculate the similar components among the ontologies. Some of them are adopted without change from the literature, while others have been improved and modified [3]-[10]. The introduced indexes are:

**Editing Distance (ED):** This index is so defined:

\[ \text{sim}_{\text{ed}}(x, y) = \max \left( 0, \frac{\min(|x|, |y|) - \text{ed}(x, y)}{\min(|x|, |y|)} \right) \in [0,1] \]

It aims to calculate the likelihood between two words that label concepts in the ontology.

**Trigram Function (TF):** This function aims to measure the number of similar trigrams that are in the words that label the concepts in the ontologies.

\[ \text{TF}(x, y) = \frac{1}{1 + |\text{tri}(x)| + |\text{tri}(y)| - 2 \cdot |\text{tri}(x) \cap \text{tri}(y)|} \in [0,1] \]

The function \( \text{tri}(x) \) gives the set of trigrams that are in the words that label the concepts in the ontologies.

**Semantic similarity index (SS):** This index is so defined

\[ \text{SS}(w_1, w_2) = 1 - \frac{1}{\text{sim}_j(w_1, w_2)} \in [0,1] \]

where

\[ \text{sim}_j(w_1, w_2) = 2 \cdot \log P(\text{Super}(c_1, c_2)) - (\log P(c_1) + \log P(c_2)) \]

This index aims to compare from a semantic point of view two words according to the taxonomy defined in Wordnet [16]. In particular this index measures the distance of the words in the taxonomy defined in Wordnet.

**Granularity (GR):** This index aims to measure the mutual position of the words representing the concepts of the ontology in the WordNet. This index is so defined:

\[ \text{GR}(c_1, c_2) = \min(\text{Dens}(c_1) \cdot \text{path}(c_1, p), \text{Dens}(c_2) \cdot \text{path}(c_2, p)) \max(\text{Dens}(c_1) \cdot \text{path}(c_1, p), \text{Dens}(c_2) \cdot \text{path}(c_2, p)) \in [0,1] \]
where dens(c) is the function representing the density of the concept c. This function is defined as \( E(c)/E \) where E is the ratio between the number’s arc of the concept and the numbers of its parents while \( E(c) \) is the number of the sibling of the concept c. The function \( path(c_1, p) \) is the shortest path from \( c_1 \) to p that is the first parent common to \( c_2 \).

**Attribute Index:** This index aims to measure the numbers of similar attributes between the two nodes. In particular it is so defined:

\[
\text{sim}_{\text{att}} = \frac{|X \cap Y|}{|X| + a(x,y) \cdot |X|/|Y| + (1 - a(x,y)) \cdot |Y|/|X|}
\]

**Synonym Index (SI):** This index aims to verify if in Wordnet there are synonyms of the word related to the concept in an ontology that label a concept in another ontology. This index can assume value 0 or 1.

**Derived Index (DE):** This index aims to find in WordNet an adjective, representing a node of an ontology, derived from the label of a concept that is in the other ontology. This index can assume value 0 or 1.

**Property Similarity Index (ISP):** This index has the aim to verify the equality between the nodes evaluating their properties. In particular the following indexes are introduced:

- Equality Indexes among superclasses (ISC): this index verifies if the superclasses of the comparing classes are similar. This index can assume value 0 or 1.
- Equality indexes among equivalent classes (IEC): this index compares all the classes that are equivalent to the comparing classes. This index assumes value 1 if all the classes are equivalent and 0 otherwise.
- Equivalent Indexes of inherited classes (IIC): this index assumes value 1 if all the inherited classes of the comparing nodes are similar. Also in this case this index assumes value 1 if all the inherited classes are equivalent and 0 otherwise.
- Equivalent Indexes of disjoint classes (IDC): this index evaluates the similarity among the disjoint classes of the comparing nodes. This index assumes values 1 if all the disjoint nodes are similar, and 0 otherwise.

The full similarity index (ISP) is obtained by the following formula:

\[
\text{ISP} = \text{ISC} \times \text{IEC} \times \text{IIC} \times \text{IDC}
\]

This index can assume value 0 or 1.

**Similarity Index for entities (ISI):** This index evaluates if the entities derived from the comparing nodes are equal. The comparison is made by evaluating both the number of entities and their typology. This index can assume value 0 or 1.

**Acronym (AC):** This index aims to verify if in the two comparing nodes one word is the acronym of the other. If it is true this index is 1, otherwise it is 0.

**Fingerprint Index (IM):** This index verifies if the word that describes a comparing node is in the other word that describes the other nodes. If the word is contained in the other one this index is 1, otherwise it is 0.

**Abbreviation (AB):** This index measures if a word that describes a node is an abbreviation of the other that describes the other comparing node. If the word is an abbreviation of the other one this index is 1, otherwise it is 0.

**Label (LA):** This index measures if the two labels of the comparing nodes are equal. Also, in this case the index assumes value 1 if the nodes have the same label, and 0 otherwise.

The introduced indexes can be grouped in three sets:

**Syntactic indexes:** These indexes aim to detect the syntactical similarities among the various components of the ontology. The following indexes are syntactic indexes:

- Editing Distance (ED)
- Trigram Function (TF)
- Acronym (AC)
- Fingerprint (IM)
- Abbreviation (AB)
- Label (LA)
- Attributes (ATT)

**Semantic indexes:** These indexes aim to compare the ontologies from a semantic point of view. As previously said these indexes can use structured knowledge as in the WordNet. The set of semantic indexes includes:

- Semantic Similarity (SS)
- Granularity (GR)
- Synonym Index (SI)
- Derived (DE)
- Label (LA)

**Structural indexes:** The indexes belonging to this set aim to compare the ontologies from a structural point of view. The indexes of this set are:
The three sets are so combined in order to map the various nodes that are in the ontologies:

\[\text{Mapping}(X,Y) = \theta \ast \text{IndSin}(X,Y) + \sigma \ast \text{IndSem}(X,Y) + \omega \ast \text{IndStr}(X,Y)\]

where:

\[\theta + \sigma + \omega = 1\]

In particular:

\[\theta = \frac{\text{IndSin}(X,Y)}{\text{IndSin}(X,Y) + \text{IndSem}(X,Y) + \text{IndStr}(X,Y)}\]

\[\sigma = \frac{\text{IndSem}(X,Y)}{\text{IndSin}(X,Y) + \text{IndSem}(X,Y) + \text{IndStr}(X,Y)}\]

\[\omega = \frac{\text{IndStr}(X,Y)}{\text{IndSin}(X,Y) + \text{IndSem}(X,Y) + \text{IndStr}(X,Y)}\]

and

\[\text{IndSin}(X,Y) = 0.5 \ast (\alpha \ast ED + \beta \ast TF) + 0.5 \ast (\max(AC,IM,AB))\]

\[\alpha = ED/(ED + TF) \ast \beta = TF/(ED + TF)\]

\[\text{IndSem}(X,Y) = 0.5 \ast (\gamma \ast SS + \delta \ast GR) + 0.5 \ast (\max(SI,DE,LA))\]

\[\gamma = SS/(SS + GR) \ast \delta = GR/(SS + GR)\]

\[\text{IndStr}(X,Y) = 0.5 \ast (\text{ATT}) + 0.5 \ast (\max(ISI,ISP))\]

After this first step, the mapping among the various nodes that are in the N ontologies is obtained. The second step is the mapping among the relations. So the following index is introduced:

\[\text{IndRel}(x,y) = \min(\text{Mapping}(A,C),\text{Mapping}(B,D),\text{RO}(x,y))\]

where x and y are the comparing relations while A and B are their domains and C and D are their co-domains. In particular,

\[\text{RO}(R_1,R_2) = \sqrt{CM(d(R_1),d(R_2)) \ast CM(r(R_1),r(R_2))}\]

and

\[\text{CM}(C_1,C_2) = \frac{|\text{UC}(C_1,H_1) \cap \text{UC}(C_2,H_2)|}{|\text{UC}(C_1,H_1) \cup \text{UC}(C_2,H_2)|}\]

In this case H_1 is the taxonomy related to the concept C_1 while H_2 is the taxonomy related to the concept C_2. The function UC (Upward Cotopy) is so defined:

\[\text{UC}(C_i,H) = \{c_j \in C \mid H(C_i,C_j)\}\]

The last step is the mapping among the attributes. This task is accomplished by the introduction of this index:

\[\text{IndAttr}(x,y) = \max\left(\text{IndSin}(x,y),\min\left(M\text{apping}(A,C),\text{equal}\left(\text{type}_{\text{source}},\text{type}_{\text{target}}\right)\right)\right)\]

After this phase the mapping process among the ontologies is obtained.

### 3. Experiment Setup and Results

In order to test the performance of the MAMA approach an experimental setup on standard datasets has been developed. In particular the experimental approach adopted was the same one developed in the SEALS project. The SEALS Project has developed a reference infrastructure known as the SEALS Platform to facilitate the formal evaluation of semantic technologies. This allows both large-scale evaluations to be run as well as ad-hoc evaluations by individuals or organizations. The SEALS evaluation setup aims at evaluating the competence of matching systems with respect to different evaluation criteria. The evaluation will focus on demonstrating the feasibility and benefits of automating matching evaluation. In this evaluation a limited set of criteria has been considered:

- Conformance: standard precision and recall, restricted semantic precision and recall, coherence

The evaluation setup contains three different scenarios, where the tools are evaluated according to common datasets and criteria:

- Scenario 1: Test data: Benchmark. Criteria: conformance with expected results
- Scenario 2: Test data: Anatomy. Criteria: conformance with expected results
- Scenario 3: Test data: Conference. Criteria: conformance with expected results and alignment coherence

The datasets were selected based on the existence of reliable reference alignments and experiences with using the datasets in evaluations:

- Systematic benchmark: the goal of this benchmark series is to identify the areas in which
each matching algorithm is strong or weak. The test is based on one particular ontology dedicated to a very narrow domain of bibliography and a number of alternative ontologies on the same domain for which alignments are provided.

- Conference: collection of conference organization ontologies. This effort was expected to materialize in alignments as well as in interesting individual correspondences (‘nuggets’), aggregated statistical observations and/or implicit design patterns.
- Anatomy: the anatomy real world case is about matching the Adult Mouse Anatomy (2744 classes) and the NCI Thesaurus (3304 classes) describing the human anatomy.

So in order to evaluate the performance of the proposed approach the following indexes, suggested by the SEALS project, have been adopted:

\[
\text{Precision} = \frac{\#\text{Correct\_Mappings}}{\#\text{Correct\_Mappings} + \#\text{Wrong\_Mappings}} \\
\text{Recall} = \frac{\#\text{Correct\_Mappings}}{\#\text{Correct\_Mappings} + \#\text{Missed\_Mappings}} \\
\text{F-Measure} = \frac{(b^2 + 1) \times \text{Precision} \times \text{Recall}}{b^2 \times \text{Precision} + \text{Recall}}
\]

In the F-Measure evaluation index the parameter b has been set to 1 in order give the same importance to the precision and recall parameters as suggested. The results obtained by the use of the MAMA approach has been compared with the same ones obtained by other methodologies developed in the literature. The experimental results are shown in Figure 1.
In order to measure the overall performance of the system, the average value of $F_{\text{Measure}}$ parameter was evaluated (Figure 2).

![Figure 2. The average F-measure.](image)

The MAMA approach shows good results for each dataset, although it does not always achieve the best result. On the other hand, it is interesting to underline that the MAMA approach shows the best value of average $F_{\text{Measure}}$. In fact, some approaches work well only for some datasets under certain conditions, while the MAMA approach is general purpose. As depicted in Figure 4 only six approaches, along with the MAMA approach, can be applied for each dataset and condition. The MAMA approach, as previously said, handles the ontology mapping problem in a very general way and therefore is able to adapt itself to the various cases.

4. Component Reuse based upon Ontology Mapping

Ontology mapping, which is an important part of ontology integration, can promote sharing and communication among different ontologies. The incorporation of ontology into software engineering can improve the reuse of software assets effectively [18]. In recent years, it has become less likely to develop complete new software systems from scratch. It becomes very important to develop software by adapting or combining existing reusable components [17]. We observe that requirement specification can provide a data source for ontology model and also the vital link for the combination of software engineering and ontology [19]. With this insight, a software component reuse approach based on ontology mapping is formulated in Figure 3.

![Figure 3. Software component reuse approach based on ontology mapping.](image)

CROM Algorithm: Through the above analysis, we find that the mapping between ontology nodes and reusable components is the key to realize the CROM (Component Reuse through Ontology Mapping) algorithm. We can construct the mapping by using requirement engineering. With requirement engineering, we will decompose requirement specification into several fragments and every fragment of requirement contains several functional points. Each functional point contains the input and output which are designed to match the requirement. The requirement and the corresponding input/output are the basis for the design and implementation of software components. Each node of ontology model is related to each fragment of requirement specification, and each fragment is related to several functional points so that each node of ontology model is also related to several functional points. This unique approach to construct the mapping between ontology node and reusable components by the functional point is the heart of the CROM algorithm. In general, the mapping operation is illustrated in Figure 4.

Ontology nodes and reusable components have $N:N$ relationship. Every ontology node may correspond to several reusable components, and every reusable component may correspond to several ontology nodes.
Even though the function description of customer requirement is the same, the attributes are often different, so it is difficult for each software component to completely meet different requirements of different ontology nodes. We need to calculate the matching degree between software component and ontology node. We consider the matching degree between a concept of the ontology node C and a reusable software component S to be a number P(C, S) between 0 and 1, with 0 representing unmatched and 1 representing completely matched. The formula is: \( P(C, S) = \frac{f(S)}{f(C)} \), with \( f(S) \) representing the matched functional points number of the current reusable software component, and \( f(C) \) representing the total functional points number of the current ontology node.

![Figure 4. The mapping between ontology node and reusable components.](image)

**Experimental Design:** The experiment is divided into two parts. The first part is to construct the application domain ontology model and the mapping between the application domain ontology nodes and the reusable software components. The data source is the application domain requirement specification and several sets of reusable software components.

Firstly, we need to construct the application domain ontology model. We adopt the ontological concept to divide the application domain requirement specification into different application fragments. For example, the general equipment application may contain four fragments: equipment, purchase, storage, and organization. Every fragment of the requirement contains the whole functional points and input/output and then we will construct the ontology on the application domain by processing different fragments describing the application domain and by mining them into common vocabularies as domain-specific concepts. So every application domain ontology node has several corresponding functional points and input/output. For example, equipment maintenance node may contain new, updating, and delete three functional points.

Secondly, we construct the mapping between application domain ontology nodes and reusable software components based on the functional points. We will match the functional points and input/output between every node and the reusable components by traversing the application domain ontology and then calculating \( P(C, S) \). If the number \( P(C, S) \) is greater than 0, the reusable component satisfies matching conditions. Because the reusable components are limited, we cannot guarantee all of the application domain ontology nodes have corresponding components that can be matched. Those nodes will be matched later during the software development and then we can put those components into the reusable components libraries.

The second part is to realize the reuse of the software components based on the ontology mapping technology and the result of the first part. The data source is the sub-application domain requirement specification. For example, `engine equipment` is a sub-application domain of general equipment.

Firstly, we need to construct the sub-application domain ontology model. We can accomplish this step according to the first step of the first part.

Secondly, we need to realize the mapping between sub-application domain ontology and application domain ontology. We can calculate the mapping index according to the MAMA approach. The final result shows that 45 nodes are equal, 9 nodes are similar, and 3 nodes are not similar. According to the software components reuse approach based on the ontology mapping, it is highly likely that the reusable components that match the 45 application domain ontology nodes also match the 45 sub-application domain ontology nodes one by one.

Finally, we will check the mapping result between the sub-application domain ontology nodes and the reusable software components by the functional points. We can calculate the number \( P(C, S) \) between sub-application domain ontology nodes and the reusable components according to the second step of the first part, and then calculate \( r_1 \), defined as the ratio of reused components over total number of components, to measure the rate of component reuse for the sub-application domain. For those components that are not directly reusable because of different attributes, but with high \( P(C, S) \) value, we can adjust the \( P(C, S) \) value according to results of the previous step, which was calculated by using MAMA to map the attribute concepts. Then we can calculate \( r_2 \), the revised rate of component reuse for the sub-application domain. To compare \( r_1 \) and \( r_2 \), the results are shown in Figure 5.
Through this experiment, we conclude that we can increase the percentage of the components reuse by matching attribute names through ontology mapping. In other words, we can systematically change the name of variables in a program so that it can be reused in another sub-application domain.

5. Conclusion

This paper presented a novel approach for ontology mapping. It uses many existing and/or improved indexes and a novel way to combine them. Experimental results demonstrate the effectiveness of the approach approach. Future research topics include the continuous improvement of the various indexes and the definition of a merging approach.

As a practical application of this methodology, the software component reuse approach based on ontology mapping emphasizes the application of the ontology mapping technology and the reuse of software components. In software engineering, ontology and ontology mapping technology can be very useful to promote the sharing and reuse of the domain knowledge. Through the mapping between the ontology nodes and reusable software components, we can achieve the reuse of the software assets from the ontology model to the software components. Further research topics include the refinement of mapping techniques between the ontology nodes and software components.

References
