The HMatch 2.0 Suite for Ontology Matchmaking *

S. Castano, A. Ferrara, D. Lorusso, and S. Montanelli

Università degli Studi di Milano
DICo - Via Comelico, 39, 20135 Milano - Italy
{castano,ferrara,lorusso,montanelli}@dico.unimi.it

Abstract. In this paper, we present the HMatch 2.0 suite for a flexible and tailored ontology matchmaking, by focusing on the architectural features and on the evaluation results. Applications of HMatch 2.0 are also discussed, with special regard for the ontology evolution issues in the frame of the BOEMIE research project.

1 Introduction

The increasing complexity of knowledge-intensive applications such as data integration, semantic search, semantic web services, peer-to-peer systems, social networks, demands for sophisticated and flexible ontology matchmaking systems, to appropriately manage and compare independent heterogeneous ontologies adopted by the various parties for knowledge representation and discovery. In particular, the capability of finding mappings between semantically related elements of two different ontologies according to different notions of similarity is a key feature for effective ontology matchmaking [1–3]. In this paper, we present the HMatch 2.0 suite for a flexible and tailored ontology matchmaking, by focusing on the architectural features and on the evaluation results and application issues also in the frame of the BOEMIE research project for the ontology evolution [1]. HMatch 2.0 has been designed to provide: i) a comprehensive suite of components for ontology matchmaking, that can be invoked alone or in combination to fit a wide range of matching requirements arising in different matching scenarios and applications; ii) a family of matching techniques for each different ontology matching component, to perform the matching process in the most suitable way, according to different understandings of the notion of semantic similarity; iii) an open architecture to ensure a high level of flexibility in combining the various matching components and to support a service-oriented interaction with knowledge intensive applications. With respect to our previous work on ontology matching [4, 5],the main contribution of this paper regards the new component-based architecture of HMatch 2.0 and the new functionalities of

* This paper has been partially funded by the BOEMIE Project, FP6-027538, 6th EU Framework Programme and by the ESTEEM PRIN project funded by the Italian Ministry of Education, University, and Research.

[1] http://www.boemie.org

instance matching and mapping composition that have been introduced in the framework of the BOEMIE project to support multimedia ontology evolution. The paper is organized as follows. In Section 2, we introduce the architecture of HMatch 2.0. Section 3 describes the components available for concept and instance matching. In Section 4, we discuss main issues on the evaluation of HMatch 2.0 while, in Section 5, we discuss expected application scenarios and current application to ontology evolution in BOEMIE. Finally, in Section 6, we provide our concluding remarks.

2 Architecture of HMatch 2.0

HMatch 2.0 is designed with the goal of providing a comprehensive framework for ontology matchmaking. In particular, HMatch 2.0 is composed by several matching components that can be used alone or in combination. Each component has the goal of evaluating a different type of matching under different understandings of similarity and by using different kind of matching techniques.

2.1 Component Interactions

HMatch 2.0 components and their interactions are shown in Figure 1.

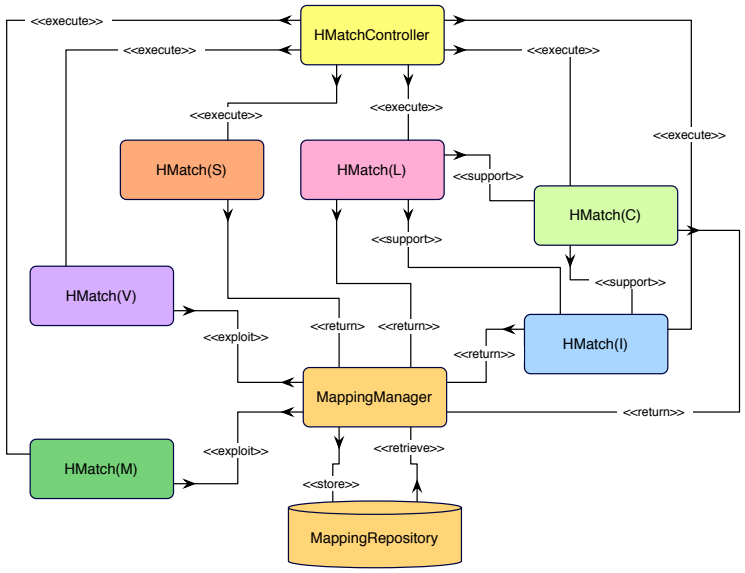

Fig. 1. High-level view of the HMatch 2.0 components and their interactions

The HMatchController is responsible of managing the configuration of HMatch 2.0 and the execution of the matching workflow. It also provides the interface

to all the matching components and to the mapping manager. HMatch($\mathcal{L}$) is the component devoted to enforce the linguistic matching. It is used both for providing to the designer the mappings derived from the linguistic analysis and as a support for the other modules which rely on linguistic affinity for implementing their matching task (i.e., HMatch($\mathcal{C}$) and HMatch($\mathcal{I}$)). HMatch($\mathcal{C}$) performs contextual matching and implements techniques for comparing ontology elements based on their contexts. HMatch($\mathcal{I}$) performs instance matching by interacting with HMatch($\mathcal{C}$) in order to establish the mappings at the schema level and with HMatch($\mathcal{L}$) to exploit the linguistic matching techniques for instance matching. HMatch($\mathcal{S}$) performs structural matching to evaluate the structural similarity between two ontologies. Since it is dependent only from the structure of the ontologies to be matched, it works on the graph structure of the ontologies without interactions with other components. HMatch($\mathcal{M}$) component provides all the functionalities required for merging sets of mappings by means of mapping operations such as intersection, union, product and transitive closure. HMatch($\mathcal{V}$) is used for mapping validation and inference. This component takes in input an initial set of mappings which can be calculated by means of any other HMatch 2.0 component. The MappingManager is responsible for the storage, release, and post-processing of mappings produced by the other modules. A mapping m produced as output of a matching process is a 5-tuple of the form:

$$m = \langle E_1, E_2, \mathcal{R}, \mathcal{V}, \mathcal{S} \rangle$$

where, E_1 and E_2 denote two ontology elements (i.e., concepts, properties, individuals), $\mathcal{R}$ denotes a semantic relation (e.g., $\equiv$, $\sqsubseteq$) holding between E_1 and E_2, $\mathcal{V} \in [0, 1]$ is a confidence value associated with $\mathcal{R}$, and $\mathcal{S} \in [0, 1]$ denotes the level of similarity between E_1 and E_2 determined by the matching component. The confidence value $\mathcal{V}$ denotes the level of trust associated with m and it is differently computed by each matching component (e.g., in HMatch($\mathcal{C}$), $\mathcal{V}$ is determined by considering the number of matching elements in the context of two ontology concepts).

2.2 Matching process

The workflow of the matching process of HMatch 2.0 is shown in Figure 2. When two ontologies are submitted to the matching process, the first step is the configuration of the HMatch 2.0 execution. In this step, the designer can choose the components of HMatch 2.0 to be activated during the matching process and can set the set of parameters required in the matching process. HMatch 2.0 is highly configurable and almost every parameter intervening in the matching process can be set by the designer. However, HMatch 2.0 is provided with default values for each parameter, in order to simplify the configuration step. If any of the activated components requires linguistic similarity analysis, the linguistic affinity (LA) is calculated between the names of ontology concepts and properties. Then, each selected matching component is executed. As a result, one or more sets of mappings are produced. The designer can stop the process and store the

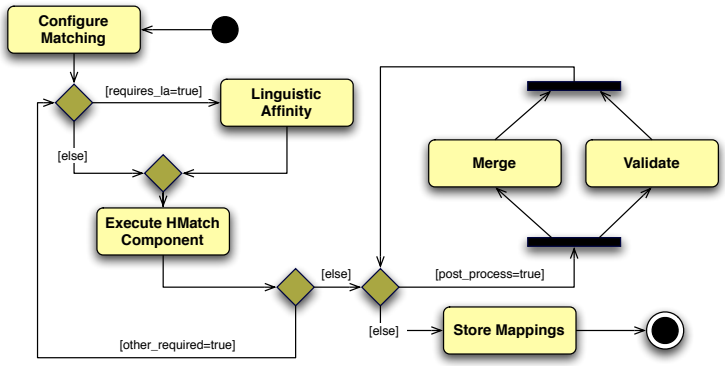

Fig. 2. Workflow of the matching process

mappings or he can perform a post-processing activity on the resulting mappings. During the mapping post-processing step, each set of mappings can be validated and new mappings can be inferred. Moreover, the different set of mappings obtained according to different similarity matchings can be merged into a comprehensive and coherent mapping set. The validation and merge step can be iterated as many times as required by the designer.

3 Ontology Matchmaking Functionalities

The HMatch 2.0 suite provides components for performing matching both at schema and instance level, namely HMatch($\mathcal{L}$), HMatch($\mathcal{C}$), HMatch($\mathcal{S}$), HMatch($\mathcal{I}$), and HMatch($\mathcal{M}$).

3.1 HMatch($\mathcal{L}$): Linguistic Matching

The Linguistic Matching component determines the level of semantic correspondence between terms used as names of ontology elements (i.e., concepts, properties, individuals). The degree of such a correspondence is calculated through a Linguistic Affinity function (LA) which returns a value in the range [0,1]. LA is used as the fundamental technique for establishing the similarity between two atomic ontology elements to be matched in all the situations when linguistic analysis is required. In HMatch 2.0, LA can be evaluated by means of three different approaches:

- **Syntactic**: using a string matching algorithm (i.e., QGram, i_Sub).
- **Semantic**: using a thesaurus or a lexical system (i.e., WordNet) of terms and terminological relationships and a notion of weighted distance between terms.
- **Combined**: using a function that combines syntactic and semantic measures.

3.2 HMatch($\mathcal{C}$): Contextual Matching

HMatch($\mathcal{C}$) determines the level of semantic affinity between concepts of two independent ontologies. A threshold-based mechanism is enforced to set the minimum level of semantic affinity required to consider two concepts as matching concepts. Given two concepts c and c', HMatch($\mathcal{C}$) calculates a semantic affinity value $SA(c, c')$ in the range $[0, 1]$ as the linear combination of a linguistic affinity value $LA(c, c')$ produced by exploiting HMatch($\mathcal{L}$) and a contextual affinity value $CA(c, c')$. The contextual affinity function of HMatch 2.0 provides a measure of similarity by taking into account the contextual features of the ontology concepts c and c'. The context can be differently composed to consider different levels of semantic complexity, and four matching models, namely, *surface*, *shallow*, *deep*, and *intensive*, are defined to this end. The context of a concept can include properties, semantic relations with other concepts, and property values. In the surface matching, only the linguistic affinity between the concept names of c and c' is considered to determine concept similarity. A detailed description of there matching models and of their evaluation is given in [4].

3.3 HMatch($\mathcal{S}$): Structural Matching

The goal of HMatch($\mathcal{S}$) is to evaluate the degree of similarity between two ontologies on the basis of their structure, by considering the RDF graph associated with the OWL representation of the two ontologies to be compared. More specifically, HMatch($\mathcal{S}$) is conceived to work only with structural information and without taking into account linguistic and contextual features of the two ontologies. This goal is important when the two ontologies do not provide significant linguistic information or the names used for ontology elements are missing or misleading. Another advantage of HMatch($\mathcal{S}$) is the fact that it allows to define mappings also for the elements which are not featured by a name, such as the so-called *anonymous classes* (i.e., quantified and cardinality restrictions, collections).

Goal of HMatch($\mathcal{S}$), in other terms, is to capture the *structural role* of an ontology element, that is the combination of information about the position of the element with respect to other elements in the ontology, the number and features of RDF triples in which it is involved, and the position within a RDF triple (i.e., subject, predicate, or object). In particular, HMatch($\mathcal{S}$) takes into account the following features for each element:

- The type of a feature, i.e., classes, properties, instances, anonymous classes.
- The number of RDF triples involving an element.
- The position within a triple (i.e., subject, predicate, or object).
- The type of the elements involved into the same RDF triples of a given element.
- The language constructs involved in the same RDF triples of a given element.

3.4 HMatch($\mathcal{I}$): Instance Matching

The goal of the instance matching component is to determine instances that represent the same real object or event and to help in disambiguating multiple

explanations of the same entity [6]. HMatch($\mathcal{I}$) evaluates the degree of similarity among different individuals by considering those assertions which provide a description of the individuals features. Consequently, the similarity of role filler values as well as the similarity of their direct types is evaluated. When two instances are compared, their similarity is proportional to the number of similar roles and role fillers they share. Moreover, for the similarity evaluation we associate a different weight with different properties of the instances, to capture the fact that some properties are more important than others in denoting the real object denoted by an instance, because they are relevant for object identification. For example, the name of a person is more important than his age for the goal of identifying the person.

The approach adopted in HMatch 2.0 is based on the idea of considering roles (referred also as properties) as connections between individuals and propagating similarity values through them. Each specification of an individual of the ABoxes is represented by means of a tree. In order to evaluate the degree of similarity of two individuals, the procedure computes a measure of similarity between datatype values and propagates these similarity degrees to the individuals of the higher level by combining the similarity among their role fillers. To this end, HMatch($\mathcal{I}$) provides a set of specific techniques devoted to the evaluation of similarity between datatype values. A function called *datatype role filler matching* is responsible of selecting the most suitable matching technique for each pair of datatype role fillers, according to the semantic meaning of the roles and to the datatype category.

3.5 HMatch($\mathcal{M}$): Mapping Analysis and Combination

Using HMatch 2.0 it is possible to collect several sets of mappings produced by the different components available both for concept and for instance matching. The different sets of mappings can be produced against the same ontologies or against a collection of different ontologies. Even in case of mappings collected against the same ontologies, the correspondences among ontology elements can be different moving from one set of mapping to another, because the different components analyze different features of the ontology elements. In order to obtain a comprehensive evaluation of the level of matching between two ontologies, HMatch 2.0 provides operations for combining different sets of mappings into a unique set and for dealing with the cardinality of a mapping set. Operations supported by HMatch($\mathcal{M}$) are defined over two mapping sets and are *intersection*, *union*, *product*, and *transitive closure*. Intersection and union are used to combine together the results obtained by different matching components against the same ontologies. From a conceptual point of view, the intersection has the goal of selecting those ontology elements that are similar with respect to more than one matching component at the same time, while union has the goal of selecting elements which are similar with respect to at least one matching component. Combining structural similarity with linguistic similarity, for example, is useful in case of elements labeled with different terms or terms that are not retrieved to be similar by linguistic matching techniques. On the other side, product and

transitive closure are used for combining together results obtained from different ontologies. The semantics of mapping operation is described in [7].

4 Evaluation results

In general, the evaluation of ontology matchmaking tools is based on the idea of using a benchmark constituted by several heterogeneous ontologies to be matched and a set of manually defined results, that is a set of expected mappings. Then, the matching tool to be evaluated is executed against the ontologies in the benchmark, in order to obtain a set of automatically retrieved mappings. On the basis of these two sets of mappings, conventional metrics are employed for the evaluation of the tool, namely precision, recall, and F-measure [8].

The evaluation of HMatch 2.0 has been performed over the 2006 and 2007 benchmarks of the Ontology Alignment Evaluation Initiative (OAEI), an international ontology matching contest held with the goal of comparing different ontology matching tools [5] [2]. We performed the evaluation of each component separately as well as the evaluation of HMatch($\mathcal{M}$)by applying the union and intersection operations. A summary view of these result is reported in Table 1. A detailed description of these results is given in [7, 5].

	HMatch($\mathcal{C}$)	HMatch($\mathcal{S}$)	HMatch($\mathcal{C}$) $\cap$ HMatch($\mathcal{S}$)	HMatch($\mathcal{C}$) $\cup$ HMatch($\mathcal{S}$)
Precision	0.92	0.81	0.99	0.81
Recall	0.60	0.72	0.35	0.76
F-Measure	0.73	0.77	0.51	0.79

Table 1. Evaluation results of HMatch 2.0

As a general remark, we note that the intersection is useful to increase precision (up to a very high level of 0.99), while union is useful to increase recall. This is because the general behavior of HMatch 2.0 is to find a quite low number of results, but very correct. Then, if we apply intersection, we reduce again the number of results, but we increase the probability to have them correct. On the opposite, if we take the union, we increase the number of results by affecting precision. More in detail, the results show that the union provides the best balance between precision and recall. The general conclusion is that intersection is supposed to be used when the precision of the results is much more important than the number of results retrieved. In all the other cases, union is the best solution in order to combine different components of HMatch 2.0.

5 Application scenarios

The HMatch 2.0 ontology matching suite provides a highly configurable matching environment where the various components can be dynamically selected accord-

[2] http://oaei.ontologymatching.org/2007/

ing to the application context that is considered for a given matching case. With respect to the complexity of the ontologies to be matched, we note that HMatch 2.0 is intended to work with OWL ontologies and it is compatible with all the OWL dialects (i.e., OWL Lite, OWL DL, OWL Full). By taking into account the elements that affects the matching process, the level of semantic complexity supported by HMatch 2.0 is $\mathcal{ALC}$. In Figure 3, we summarize the applicability of each HMatch 2.0 component in terms of i) the ontological features that are considered, and ii) the suggested application context.

	HMatch($\mathcal{L}$)	HMatch($\mathcal{C}$)	HMatch($\mathcal{S}$)	HMatch($\mathcal{I}$)
Ontological description	Poorly structured ontologies, with very simple concept descriptions (e.g., Web directories, glossaries)	Ontologies with semantically rich concept descriptions (e.g., OWL ontologies, DL specifications)	Ontologies with "nameless" concept descriptions (e.g., XML documents, Ontologies with generally labeled elements)	Ontologies with instances (e.g., Semantic Web ontologies)
Application context	Oriented to soft chema integration	Oriented to stable schema integration	Oriented to semi-structured document integration	Oriented to data/information integration

Fig. 3. Applicability of the HMatch 2.0 components

HMatch($\mathcal{L}$) is suited to work with poorly structured ontologies, such as Web directories, taxonomies, and glossaries, where properties, semantic relations, and instances are not available. Moreover, HMatch($\mathcal{L}$) is also used with richer ontology descriptions to perform an initial comparison and to provide a linguistic analysis as input for the execution of other HMatch 2.0 components. In general, HMatch($\mathcal{L}$) is suggested for generic matching scenarios where the linguistic component is relevant and/or is self-explanatory. In particular, soft schema integration is a typical application context where HMatch($\mathcal{L}$) is invoked to identify the corresponding labels used in different schemas of web sources. As a further application context, HMatch($\mathcal{L}$) can be used for supporting social tagging and classification of Web resources (i.e., *folksonomy*). In this respect, HMatch($\mathcal{L}$) has the role to suggest to the user the "right tag" for a given resource according to linguistic-based rather than popularity-based metrics. When semantically rich ontology descriptions are provided (e.g., OWL ontologies, DL specifications), HMatch($\mathcal{C}$) is the suggested component to use due to the high level of matching granularity enforced through its matching models (i.e., surface, shallow, deep, intensive). For this reason, HMatch($\mathcal{C}$) is suited for general-purpose matching scenarios where it can be properly combined with HMatch($\mathcal{L}$) to obtain stable integrated representation of the information. For instance, in schema integration applications, HMatch($\mathcal{C}$) can be used to refine the results of the linguistic matching and to provide a more accurate matching evaluation by taking into account contextual features of schema elements to be integrated. HMatch($\mathcal{S}$) is suggested when the ontology descriptions contain "nameless" concept descrip-

tions, that is ontologies with anonymous classes or with non-meaningful element names. For this reason, HMatch($\mathcal{S}$) is suited for those application contexts where the meaningfulness of the terminological part is not guaranteed and/or is not a core requirement. As an example, HMatch($\mathcal{S}$) can be adopted for semi-structured document integration (e.g., XML documents) where tag labels are often automatically generated by applications thus making ineffective the adoption of linguistic-based matching components. For what concern HMatch($\mathcal{I}$), it is suited to work with ontology instances (ABoxes). In particular, HMatch($\mathcal{I}$) is suggested for those application contexts where extensional matching is applicable, such as data/information integration.

We note that in real matching scenarios more than one HMatch 2.0 component can be executed according to the specific features of the ontology descriptions to be matched. In this respect, the results produced by each component can be combined by invoking HMatch($\mathcal{M}$) in order to return a single comprehensive set of matching results. Moreover, the combination of different HMatch 2.0 components can contribute to increase the quality of the matching results. For instance, intersection and union of HMatch($\mathcal{C}$) and HMatch($\mathcal{S}$) results provide better performance in terms of precision and recall, respectively, as discussed in Section 4.

5.1 A practical application to ontology evolution

The HMatch 2.0 ontology matching suite is actually adopted in the framework of the BOEMIE project where it is exploited for supporting multimedia ontology evolution. In BOEMIE, a novel methodology for ontology evolution is defined to enhance traditional approaches and to provide methods and techniques for evolving a domain ontology through acquisition of semantic information from multimedia sources such as image, video, and audio [9]. The BOEMIE methodology is characterized by the use of a reasoning-based engine with the role of providing a semantic interpretation of the extracted information and by the use of an ontology matching engine. In particular, HMatch 2.0 is used as a comprehensive matching service to support the BOEMIE evolution activities and tasks as shown in Table 2. According to the interpretation results, ontology evolution is performed i) through *population* by inserting new instances in the ontology, and ii) through *enrichment* by inserting new concepts and relation types in the ontology. Coordination activities are also defined in the BOEMIE methodology to log changes and to manage the different versions of the ontology produced with evolution.

6 Concluding Remarks

In this document, we have described the architecture and the main matching components implemented in the HMatch 2.0 ontology matchmaking suite. Instance matching is a challenging task and HMatch 2.0 is one of the few ontology

Activity	Task	HMatch 2.0 component	Goal
Population	Instance grouping	HMatch($\mathcal{I}$)	To group together instances referred to the same individual in the domain
Enrichment	Concept enhancement	HMatch($\mathcal{L}$), HMatch($\mathcal{C}$)	To suggest names for new concepts and properties
Coordination	Alignment	HMatch($\mathcal{C}$), HMatch($\mathcal{S}$)	To align a new version of the domain ontology with other external knowledge sources

Table 2. Usage of HMatch 2.0 components in BOEMIE

matching tools with instance matching functionalities. Moreover, we have introduced the idea of composing mappings obtained by applying different matching components. In such a way, the domain expert is capable of collecting separate mappings sets on the ground of different application purposes and of deriving a comprehensive similarity view of ontology elements. Our future work will be devoted (i) to increasing the recall results obtained with HMatch($\mathcal{C}$) starting from experimental data and working on the combination between HMatch($\mathcal{C}$) and HMatch($\mathcal{S}$), and (ii) to formalize the semantics of mapping operations by taking into account the different types of mappings and their relations.

References

1. Aumueller, D., Do, H., Massmann, S., Rahm, E.: Schema and Ontology Matching with COMA++. In: Proc. of SIGMOD 2005 - Software Demonstration, Baltimore, USA (2005)
2. Jian, N., Hu, W., Cheng, G., Qu, Y.: Falcon-AO: Aligning Ontologies with Falcon. In: Proc. of K-CAP Workshop on Integrating Ontologies, Banff, Canada (2005)
3. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag (2007)
4. Castano, S., Ferrara, A., Montanelli, S.: Matching Ontologies in Open Networked Systems: Techniques and Applications. Journal on Data Semantics (JoDS) **V** (2006)
5. Castano, S., Ferrara, A., Messa, G.: ISLab HMatch Results for OAEI 2006. In: Proc. of ISWC Int. Workshop on Ontology Matching, Athens, Georgia, USA (2006)
6. Gu, L., Baxter, R.A., Vickers, D., Rainsford, C.: Record Linkage: Current Practice and Future Directions. Technical Report 03/83, CSIRO Mathematical and Information Sciences (2003)
7. Bruno, S., Castano, S., Ferrara, A., Lorusso, D., Messa, G., Montanelli, S.: Ontology Coordination Tools: Version 2. Technical Report D4.7, BOEMIE Project, FP6-027538, 6th EU Framework Programme (2007)
8. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley (1989)
9. Castano, S., Espinosa, S., Ferrara, A., Karkaletsis, V., Kaya, A., Melzer, S., Möller, R., Montanelli, S., Petasis, G.: Ontology Dynamics with Multimedia Information: The BOEMIE Evolution Methodology. In: Proc. of the ESWC Int. Workshop on Ontology Dynamics (IWOD 07), Innsbruck, Austria (2007)