Matching Ontologies in Open Networked Systems: Techniques and Applications

Silvana Castano, Alfio Ferrara, and Stefano Montanelli

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

Abstract. In open networked systems a varying number of nodes interact each other just on the basis of their own independent ontologies and of knowledge discovery requests submitted to the network. Ontology matching techniques are essential to enable knowledge discovery and sharing in order to determine mappings between semantically related concepts of different ontologies. In this paper, we describe the H-Match algorithm and related techniques for performing matching of independent ontologies in open networked systems. A key feature of H-Match is that it can be dynamically configured for adaptation to the semantic complexity of the ontologies to be compared, where the number and type of ontology features that can be exploited during the matching process is not known in advance as it is embedded in the current knowledge request. Furthermore, this number can vary, also for the same ontologies, each time a new matching execution comes into play triggered by a knowledge request. We describe how H-Match enforces this capabilities through a combination of syntactic and semantic techniques as well as through a set of four matching models, namely *surface*, *shallow*, *deep*, and *intensive*. Then, we describe the application of H-Match and its implementation for knowledge discovery in the framework of the Helios peer-based system. Finally, we present experimental results of using H-Match on different test cases, along with a discussion on precision and recall.

1 Introduction

Open networked systems like Peer-to-Peer networks and Grids are becoming more and more semantics-enriched infrastructures enabling to share and create knowledge and to enforce semantic collaboration among the involved parties. For example, basic P2P networks adopting simple filenames for data sharing have been evolving to schema-based P2P networks, capable of supporting the exchange of complex resources like documents and services described by using metadata or thematic ontologies [1, 2]. P2P scientific collaboration networks have recently emerged to take advantage of the inherent properties of P2P networks in order to enforce scientific data sharing and to obtain better performance, flexible and efficient use of resources and system resilience [3, 4].

S. Spaccapietra et al. (Eds.): Journal on Data Semantics V, LNCS 3870, pp. 25–63, 2006.

In such systems, it is widely recognized that the use of ontologies plays a crucial role for providing a semantic description of the resources to be shared and for enhancing resource discovery through expressive queries [5]. A key feature of open networked systems is that the network organization can vary at any moment and a unique global ontology committed by all the parties is not a viable solution. Rather, a networked system is characterized by a multitude of independent *peer ontologies* autonomously made available by each node joining the system. Consequently, for knowledge discovery and sharing, a varying number of nodes interact each other just on the basis of their own independent ontologies and of knowledge discovery requests submitted to the network. In this context, appropriate ontology matching techniques are required to determine whether and how concepts of different ontologies are semantically related each other [6, 7]. The problem of schema and ontology matching has been investigated in the literature and a number of approaches and tools have been proposed in the area of data and knowledge management [7, 8, 9, 10, 11, 12]. A reference survey on schema matching is given in [13] while ontology matching is surveyed according to different classification frameworks in [14, 15, 16, 17].

Existing ontology matching approaches address a number of general requirements which remain very important in open networked systems. A first general requirement is the applicability to different ontology specification languages, with special attention to recent standards of the Semantic Web like OWL [18]. A further general requirement is the capability of coping with different levels of detail and design choices in describing the knowledge of interest using a certain language. In addition, the capability of considering different constructs used in ontology languages is required for matching purposes.

In addition, new peculiar requirements must be taken into account in conceiving ontology matching techniques for open networked systems. These requirements are originated by the dynamic behavior of peers in such a scenario. A first peculiar requirement is that the number and type of ontology features that can be exploited during the matching process is not known in advance as it is embedded in the current knowledge request. Furthermore, this number can vary, also for the same ontologies, each time a new matching execution comes into play triggered by a knowledge discovery request. Moreover, design principles of ontology matching techniques must be driven by i) the necessity of satisfying matching requests that are dynamically posed by peers on the basis of unexpected needs that can vary continuously, and ii) by the necessity of addressing all general and peculiar matching requirements as a whole.

In this paper, we present the H-Match algorithm and related techniques for matching independent ontologies in open networked systems. H-Match has been developed in the framework of the Helios peer-based system, where it is used to enable knowledge discovery and sharing [19, 20]. A key feature of H-Match is that it can be dynamically configured for adaptation to the semantic complexity of the ontologies to be compared, using a combination of syntactic and semantic techniques. This feature is achieved by means of four matching models, namely *surface*, *shallow*, *deep*, and *intensive* defined with the goal of providing a wide spectrum of metrics suited for dealing with many different matching scenarios. Another distinguishing feature of H-Match is that the matching configuration is selected in an automated way according to a matching policy embedded in the incoming request.

In developing H-Match, we started from the schema matching functionalities of the Artemis integration system [21]. From Artemis we borrowed the thesaurus-based approach for name affinity management, and we made a number of extensions for matching linguistic features of ontology elements to provide a fully-automated approach. Furthermore, we have moved from the notion of structural affinity, typical of schema elements based on attributes, to the notion of contextual affinity, typical of ontology elements, based on semantic relations with explicit semantics, with consequent development of suitable techniques for contextual affinity. Moreover, we have introduced the notion of matching model and of configurability of the matching process through matching models. Finally, we want to remark that H-Match implements an automated ontology matching approach, since it has been conceived to enable the knowledge discovery process in open networked systems without any manual intervention. On the contrary, Artemis enforces a semi-automated approach to schema matching being targeted to support the schema unification process in data integration systems with expected interaction with the designer.

Motivating and Running Example. In Figure 1, we show a graphical representation [1] of two simple ontologies, namely Apple-q and Apple-o[2]. They will be used as running example throughout the paper to show how the H-Match techniques work. The Apple-q ontology specifies that the concept Apple is a fruit and a kind of food. Apples, in this ontology, origin from Italy. The Apple-o ontology describes two kinds of products, namely Edible_fruits and Computer, where Mobile_Computer is a kind of computers. For fruits (i.e., Banana, Grape, and Pineapple), we have information about the provenance (i.e., Brazil for banana and pineapple, and Italy for grape). In this latter ontology, the concept Apple denotes a brand of computers like IBM, and it is located in USA. One of the challenging goals of ontology matching in this example is to capture the difference between the two Apple concepts, even if they have the same name. We have to evaluate whether the matching techniques are able to capture this difference on the basis of the different context that characterizes the two apple concepts in their respective ontologies. Other relevant matchings that should be found by the matching techniques are the ones between Food and Fruit of Apple-q and the concept of Edible_fruits in Apple-o, as well as between the two Italy concepts which denote the same region in the two ontologies. In the remaining of the paper, we will use this

[1] This graphical representation is based on H-MODEL, the formalism adopted by H-Match for internal representation of ontologies for matching (see Section 2.1).

[2] The OWL specification of the two ontologies is provided at http://islab.dico.unimi.it/-ontologies/apple-q.owl and http://islab.dico.unimi.it/ontologies/apple-o.owl, respectively.

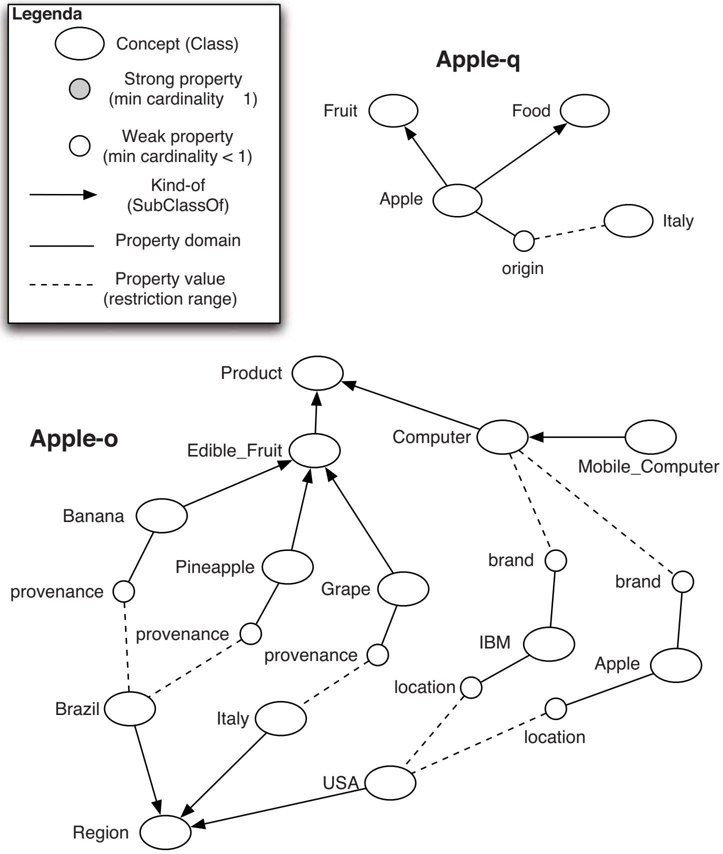


Fig. 1. H-Model graphical representation of Apple-q and Apple-o

example as the running example to show step by step how the matching process works.

Organization of the Paper. The paper is organized as follows. In Section 2, we give the foundations of the proposed ontology matching techniques. In Section 3, we describe the H-Match algorithm and related matching techniques with running examples. In Section 4, we describe the application of H-Match and its implementation for dynamic knowledge discovery in the framework of our open networked system Helios. In Section 5, we provide experimental results of applying H-Match and related matching techniques to Semantic Web ontologies test cases, by discussing the obtained results in terms of precision and accuracy. In Section 6, we make a critical comparison of H-Match with related work in the field of ontology matching. Finally, in Section 7, we give our concluding remarks.

2 Foundations of H-Match

We define *ontology matching* as a process that takes two ontologies as input and returns the mappings that identify corresponding concepts in the two ontologies, namely the concepts with the same or the closest intended meaning. We define a *mapping* as a correspondence between a concept of the first ontology and one or more concepts of the second ontology [3]. Ontology mappings are established after an analysis of the similarity of the concepts in the compared ontologies. In H-Match, we perform similarity analysis through affinity metrics to determine a measure of semantic affinity in the range $[0, 1]$. A threshold-based mechanism is enforced to set the minimum level of semantic affinity required to consider two concepts as matching concepts. With H-Match, it is possible to determine *one-to-one mappings* and *one-to-many mappings*. In a one-to-one mapping, a concept of the first ontology is associated with only one concept of the second ontology, namely the matching concept with the highest value of semantic affinity (also called *best-matching* concept). In a one-to-many mapping, a concept of the first ontology is associated with a set of concepts of the second ontology, namely all the selected matching (also called *best-k matching* concepts).

2.1 Ontology Representation

In H-Match, an ontology is seen as a set of concepts, properties, and semantic relations. For the sake of internal representation of ontology specification languages, and in particular for Semantic Web languages like OWL, we rely on a reference model, called H-Model. H-Model, as many other tools for ontology matching [4], provides a graph-based representation of ontologies in terms of concepts, properties, and semantic relations. In Figure 1, we show an example of H-Model representation of OWL ontologies. In this representation, the graph nodes denote concepts and properties (that in the example represent classes and properties of OWL), while the edges denote the semantic relations between concepts (that in the example represent the equivalence and subclass relations in OWL as well as properties domain and range derived by OWL restrictions). For a more detailed description of H-Model and supported ontology specification languages, the reader can refer to [20].

2.2 Semantic Complexity

The notion of semantic complexity has been introduced in [10] to describe different levels of complexity at which an ontology can be seen for matching purposes.

[3] In this respect, other approaches like [8, 10, 22], consider properties as first-class objects and, therefore, they find mappings also between them. We take an approach similar to [7, 12, 23] where mappings are found for concepts and properties are still matched but for the purpose of evaluating concept similarity.

[4] The state of the art ontology matching tools are analyzed with respect to the supported model for ontology representation in the related work of Section 6.

At each level, the focus is on the different types of constructs of the ontology specification. The four matching models defined in H-MATCH, namely *surface*, *shallow*, *deep*, and *intensive*, allow the matching process to enforce different levels of semantic complexity depending on the ontologies to be matched. In particular, the surface model is suitable for matching ontologies at the entity level, because it considers only names of ontology elements. The shallow model is suitable for ontologies that are semantic nets in that both names and concept properties are taken into account for matching. The deep and the intensive models are adequate for semantic complexity of the Description Logics languages, because they also take into account semantic relations and property values, respectively. H-MATCH performs matching by considering schema-level information of ontology descriptions. In other words, H-MATCH is focused the terminological box level of the description logics languages. This has been a design choice in developing H-MATCH, in order to support knowledge discovery in the HELIOS open networked systems [5].

2.3 Linguistic Features

Linguistic features refer to names of ontology elements and their meaning. To capture the meaning of names for ontology matching, we borrow from ARTEMIS [21] the idea of relying on a thesaurus of terms and weighted terminological relationships among them. In H-MATCH the thesaurus is automatically derived from the lexical system WordNet [24], to provide a common reference basis for all the peers of the system and to achieve a uniform interpretation of linguistic features as much as possible. To this end, we have introduced the following extensions to the ARTEMIS procedure motivated by the use of WordNet:

- Full use of the relations among synsets in WordNet, including not only synonymy and hypernymy/hyponymy, but also other relations provided by WordNet like meronymy and coordinate terms. Thesaurus construction can be configured in order to select the syntactic category to be taken into account. In the case of verbs, adjectives, and adverbs, the corresponding specific relations (e.g., troponymy for verbs) are considered.
- Automated management of compound terms not included in WordNet. In fact, names appearing in real ontologies often are formed by two or more terms originating compound terms that are not retrieved in WordNet.

The thesaurus is structured as a graph, where the nodes represent terms and the edges represent terminological relationships. Terms can be basic or compound. *Basic terms* are all those terms that are included in WordNet, composed by one or multiple tokens. *Compound terms* are all those terms composed by more than one token that are not included in WordNet. Terminological relationships represented in the thesaurus are SYN, BT, NT, and RT. SYN (synonymy)

[5] In Section 6, we discuss how currently available matching techniques of H-MATCH can be taken into account for the purpose of considering also instance-level information of ontology specifications.

denotes that two terms have the same meaning. BT (broader term) (resp., NT (narrower term)) denotes that a term has a more (resp., less) general meaning than another term. Finally, RT (related terms) denotes that two terms have a generic positive relationship. These terminological relationships are derived from the relations defined in WordNet during the thesaurus construction process, as described in Section 3. As in Artemis, a weight W_{tr} is associated with each terminological relationship $tr \in \{\mathsf{SYN}, \mathsf{BT/NT}, \mathsf{RT}\}$ in the thesaurus. Such a weight expresses the implication of the terminological relationship for semantic affinity. Different types of relationships have different implications for semantic affinity, with $W_{\mathsf{SYN}} \geq W_{\mathsf{BT/NT}} \geq W_{\mathsf{RT}}$. In fact, synonymy is generally considered a more precise indicator of affinity than hierarchical relationships, consequently $W_{\mathsf{SYN}} \geq W_{\mathsf{BT/NT}}$. The lowest weight is associated with RT since it denotes a more generic relationship than the hierarchical relationships BT/NT.

2.4 Contextual Features

Contextual features of a concept c refer both to the properties and to the concepts directly related to c through a semantic relation in an ontology. The importance of considering contexts was already pointed out in [25] for matching heterogeneous information. It becomes mandatory for ontology matching especially in distributed contexts, where the meaning of a concept if often determined by the context where it is downhearted [17]. In Section 6, we provide a comparison of the state of the art ontology matching tools with respect to the contextual features that are supported.

Given a concept c, we denote by $P(c)$ the set of properties of c, and by $C(c)$ the set of *adjacents* of c, namely concepts that participate in a semantic relation with c, respectively. The context of a concept in H-Match is defined as the union of the properties and of the adjacents of c, that is, $Ctx(c) = P(c) \cup C(c)$. In H-Match, we distinguish between strong and weak properties. A strong property sp is a mandatory property with minimal cardinality 1. A weak property wp is an optional property with minimal cardinality 0. In H-Match we distinguish four semantic relations sr between two concepts c and c', namely same-as, kind-of, part-of, and associates. The same-as relation denotes that c and c' are equivalent, while the kind-of and part-of relations denote that c and c' are related by a specialization relation and a composition relation, respectively. Finally, the associates relation denotes that c and c' are related by a generic positive semantic association. In Figure 1, examples of how OWL constructs are mapped on such semantic relations are given.

Like linguistic features, also contextual features are weighted in H-Match. In particular, we associate a weight W_{sp} to strong properties, and a weight W_{wp} to weak properties, with $W_{\mathsf{sp}} \geq W_{\mathsf{wp}}$ to capture the different importance each kind of property has in characterizing the concept. In fact, strong properties are mandatory properties related to a concept and they are considered more relevant in contributing to concept description. Weak properties are optional for the concept in describing its structure, and, as such, are less important in featuring the concept than strong properties. Each semantic relation has associated a weight W_{sr} which expresses the strength of the connection expressed by the relation on the involved concepts. The greater the weight associated with a semantic relation, the higher the strength of the semantic connection between concepts. For this reason, we define $W_{\mathsf{same-as}} \geq W_{\mathsf{kind-of}} \geq W_{\mathsf{part-of}} \geq W_{\mathsf{associates}}$.

3 Matching Ontologies with H-Match

H-Match combines a measure of linguistic affinity and a measure of contextual affinity in order to evaluate a comprehensive measure of semantic affinity between ontology concepts. The linguistic affinity provides a measure of similarity between the ontology concepts by considering their linguistic features, while the contextual affinity provides a measure of similarity by taking into account their contextual features. Four matching models, namely, *surface*, *shallow*, *deep*, and *intensive*, are defined for dynamically configuring H-Match for its adaptation to the semantic complexity of the ontologies to be compared. The H-Match matching process is shown in Figure 2. The process starts with the computation of the linguistic affinity among the ontology concepts, which is common to all matching models. Then a matching model is chosen. The context of concepts is

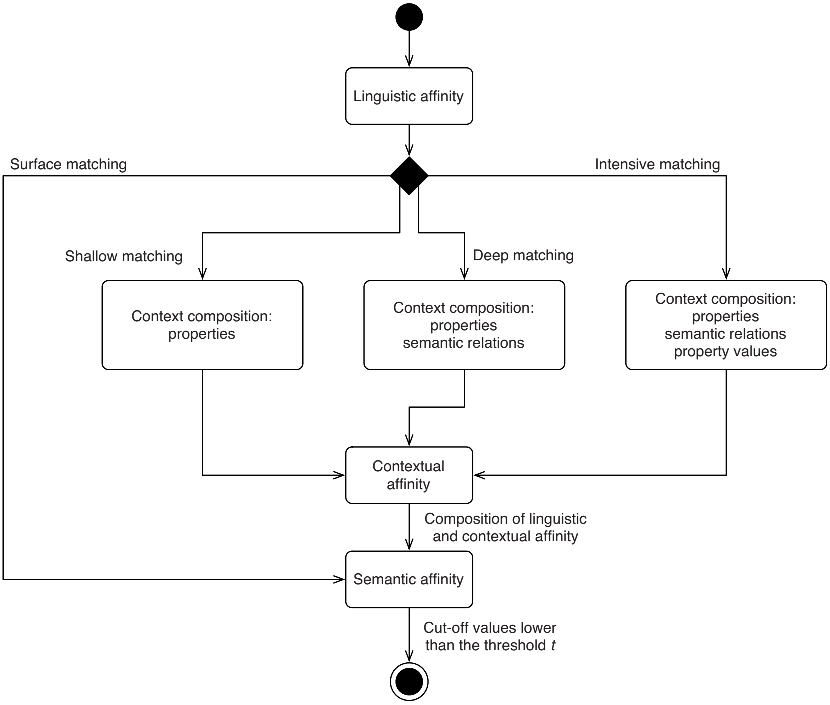

Linguistic affinity
Surface matching
Intensive matching
Shallow matching
Deep matching
Context composition: properties
Context composition: properties semantic relations
Context composition: properties semantic relations property values
Contextual affinity
Composition of linguistic and contextual affinity
Semantic affinity
Cut-off values lower than the threshold t

Fig. 2. The matching process of H-Match

then composed according to the selected matching model and the corresponding contextual affinity is computed. Finally, the linguistic and the contextual affinity values are combined to produce the final comprehensive semantic affinity value. Matching concepts are then selected by cutting off the concepts whose semantic affinity is below the threshold.

After describing the H-Match WordNet-based procedure for thesaurus construction and the basic functions for matching terms, datatypes, and relations, we describe how each model of H-Match works.

3.1 Thesaurus Construction

The H-Match thesaurus construction is performed in two steps. First the entries for both basic and compound terms are defined. Then, the terminological relationships holding among the term entries are defined. Given the set T of terms used as names of ontology elements, the construction procedure inserts into the thesaurus an entry for each basic term $bt_i \in T$, for each compound term $ct_i \in T$, and for each constituent token of ct_i. If a token is a compound term itself, the procedure is recursively iterated until all the compound terms are analysed and corresponding basic term entries are defined. Regarding terminological relationships, the first step is devoted to define terminological relationships for compound terms. Our approach relies on the idea that in a typical compound term ct, one of its constituent tokens denotes the central concept represented by ct, while the remaining tokens denote a specification of such a central concept [26]. In particular for English, we follow the heuristics that the last token bt_n appearing on the right side of a compound term ct composed by n tokens denotes the central concept, and that each remaining token bt_i, $i = 1 \ldots n-1$ we encounter going from the left side to the right side of ct denotes a qualification of the meaning of bt_n. On this basis, a NT relationship is defined between ct and bt_n and a RT relationship is defined between ct and each remaining token bt_i, $i = 1 \ldots n-1$. Finally, we define the terminological relationships SYN, BT, NT and RT between basic term entries on the basis of the relations among synsets that are provided by WordNet. In particular: a WordNet synonymy is represented through a SYN terminological relationship; a WordNet hypernymy (resp., hyponymy) relation is represented through a BT (resp., NT) terminological relationship; meronymy and coordinate terms relations of WordNet are represented through a RT relationship in thesaurus [6].

Example. We consider the example of Figure 1. The first step in the thesaurus construction is to extract from Apple-q and Apple-o the names of concepts and properties and to determine the thesaurus entries. Most names are single terms and are already present in WordNet. An entry for them is thus defined in the

[6] The examples and the thesaurus description provided in the paper are referred to nouns for the sake of clarity. In the case of verbs, the corresponding specific relations in WordNet, such as troponymy, are considered and mapped onto the thesaurus terminological relationships following analogous rules.

thesaurus. There are only two compound terms, Edible_Fruit and Mobile_Computer. The first one is retrieved in WordNet and therefore is considered as a basic term and inserted as an entry in the thesaurus. The second one is not retrieved in WordNet. For this reason, we split it into two tokens (i.e., Mobile and Computer); then we insert in the thesaurus two new entries, one for Mobile_Computer and one for Mobile, while an entry for Computer is already present in thesaurus being already a concept name.

The second step is to determine the terminological relationships among the thesaurus entries. First of all, we consider the compound term Mobile_Computer and we insert a NT relationship between Mobile_Computer and Computer, in order to denote that mobile computers are a specialization of computers. Moreover, we insert also a RT relationship between Mobile_Computer and Mobile, according to the approach described above. Then, WordNet is exploited for deriving all the other relationships that are reported in Table 1. Finally, at the end of the thesaurus construction phase, a weight is associated with each terminological relationship in the thesaurus. The weights of terminological relationships used for thesaurus construction are 1.0 for SYN, 0.8 for BT/NT, and 0.5 for RT. Such weights have been maintained from Artemis where they have been defined after extensive experimentation on several schema matching and integration cases. We performed experimentations using them also on several ontology matching

Apple	SYN	**Apple**
Apple	NT	**Edible_Fruit**
Banana	SYN	**Banana**
Banana	NT	**Edible_Fruit**
Brand	SYN	**Brand**
Brazil	SYN	**Brazil**
Computer	SYN	**Computer**
Computer	BT	**Mobile_Computer**
Edible_Fruit	SYN	**Edible_Fruit**
Edible_Fruit	BT	**Apple**
Edible_Fruit	BT	**Banana**
Edible_Fruit	BT	**Grape**
Edible_Fruit	BT	**Pineapple**
Edible_Fruit	NT	**Fruit**
Food	SYN	**Food**
Fruit	SYN	**Fruit**
Fruit	BT	**Edible_Fruit**
Fruit	NT	**Product**
Grape	SYN	**Grape**
Grape	NT	**Edible_Fruit**

IBM	SYN	**IBM**
Italy	SYN	**Italy**
Location	SYN	**Location**
Location	NT	**Region**
Mobile	SYN	**Mobile**
Mobile	RT	**Mobile_Computer**
Mobile_Computer	SYN	**Mobile_Computer**
Mobile_Computer	NT	**Computer**
Mobile_Computer	RT	**Mobile**
Origin	SYN	**Origin**
Origin	BT	**Provenance**
Pineapple	SYN	**Pineapple**
Pineapple	NT	**Edible_Fruit**
Product	SYN	**Product**
Product	BT	**Fruit**
Provenance	SYN	**Provenance**
Provenance	NT	**Origin**
Region	SYN	**Region**
Region	BT	**Location**
USA	SYN	**USA**

Table 1. Example of thesaurus entries for the running example

cases and we have seen that they work well also for ontology matching. Consequently, we maintain them as default weights also in the H-Match thesaurus construction procedure.

3.2 Basic Matching Functions

In this section, we describe the basic matching functions that are used in order to evaluate the similarity/compatibility of terms, datatypes, properties and semantic relations, respectively.

Term Affinity Function. The term affinity function $\mathcal{A}(t, t') \rightarrow [0, 1]$ evaluates the affinity between two terms t and t' based on the thesaurus. The term affinity function is borrowed from Artemis and it is reported here for the sake of clarity. $\mathcal{A}(t, t')$ of two terms t and t' is equal to the value of the highest-strength path of terminological relationships between them in Th if at least one path exists, and is zero otherwise. A path strength is computed by multiplying the weights associated with each terminological relationship involved in the path, that is:

$$\mathcal{A}(t, t') = \begin{cases} \max_{i=1\dots k} \{W_{t \rightarrow_i^n t'}\} & \text{if } k \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where: k is the number of paths between t and t' in Th; $t \rightarrow_i^n t'$ denotes the ith path of length $n \geq 1$; $W_{t \rightarrow_i^n t'} = W_{1_{tr}} \cdot W_{2_{tr}} \cdot \ \ldots \ \cdot W_{n_{tr}}$ is the weight associated with the ith path, and $W_{j_{tr}}, j = 1, 2, \ldots, n$ denotes the weight associated with the jth terminological relationship in the path.

Datatype Compatibility Function. The datatype compatibility function $\mathcal{T}(dt, dt') \rightarrow [0, 1]$ is defined to evaluate the compatibility of data types of two concept properties according to a pre-defined set CR of compatibility rules. $\mathcal{T}(dt, dt')$ of two data types dt and dt' returns 1 if dt and dt' are compatible according to CR, and 0 otherwise, that is:

$$\mathcal{T}(dt, dt') = \begin{cases} 1 & \text{iff } \exists \text{ a compatibility rule for } dt, dt' \text{ in } CR \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

For instance, with reference to XML Schema datatypes (which are relevant for OWL ontology matching), examples of compatibility rules that have been defined are: xsd:integer ⇔ xsd:int, xsd:integer ⇔ xsd:float, xsd:decimal ⇔ xsd:float, xsd:short ⇔ xsd:int.

Property and Semantic Relation Closeness Function. The closeness function $\mathcal{C}(e, e') \rightarrow [0, 1]$ is defined to calculate a measure of the distance between two elements e and e' of concept contexts. Depending on the way concept contexts are defined in each respective ontology, e and e' can be either two properties, or two semantic relations, or a semantic relation and a property, respectively.

$\mathcal{C}(e, e')$ exploits the weights associated with context elements and returns a value in the range [0,1] proportional to the absolute value of the complement of the difference between the weights associated with the elements, that is:

$$\mathcal{C}(e, e') = 1- \mid W_e - W_{e'} \mid \tag{3}$$

where W_e and $W_{e'}$ are the weights associated with e and e', respectively. For any pairs of elements e and e', the highest value (i.e., 1.0) is obtained when weights of e and e' coincide. The higher the difference between W_e and $W_{e'}$ the lower the closeness value of e and e'.

3.3 Matching Models

The matching models have been conceived to span from surface to intensive matching. Each model calculates a semantic affinity value $SA_{c,c'}$ of two concepts c and c' which expresses their level of matching. $SA_{c,c'}$ is produced by considering linguistic and/or contextual features of concept descriptions. In a given matching model, the relevance of the linguistic and the contextual features of c and c' for matching can be established, by properly setting the linguistic affinity weight $W_{la} \in [0, 1]$ in the semantic affinity evaluation process.

Before describing matching models, we make a general consideration for the surface, shallow, deep, and intensive models. In the evaluation of the contextual affinity, a special case occurs when both the concepts have an empty context. To deal with this, three strategies are possible: i) *NULL-value strategy*: the empty contexts can be considered to have a semantics analogous to the one of the NULL value in relational databases. In this strategy, the contextual affinity is set to undetermined to capture this semantics; ii) *worst-case strategy*: since the concepts do not have elements in their contexts, the contextual affinity value is set to 0 for them, to express that no matching elements have been found in their contexts; iii) *best-case strategy*: since the concepts do not have elements in their contexts, the contextual affinity value is set to 1 for them, to express that two empty contexts are considered to fully match. In implementing H-Match, we have decided to adopt the worst-case strategy (ii) in order to avoid to produce semantic affinity values either too much optimistic (iii) or semantic affinity values that are based only on linguistic affinity (i), without accounting for the information about the presence of empty contexts.

Surface Matching. The surface matching is defined to take into account only the linguistic features of concept descriptions. Surface matching addresses the requirement of dealing with high-level, poorly structured ontological descriptions. Given two concepts c and c', surface matching provides a measure $SA_{c,c'}$ of their semantic affinity determined only on the basis of their names using the term affinity function (1), that is:

$$SA_{c,c'} \equiv \mathcal{A}(n_c, n_{c'}) \tag{4}$$

where n_c and $n_{c'}$ are the names of c and c', respectively. When the surface model is selected, the W_{la} weight is automatically considered to be 1 by H-Match.

Example. All the semantic affinity values computed using the surface matching on the running example ontologies Apple-q and Apple-o are shown in Table 3, referring to the thesaurus shown in Table 1. For instance, the semantic affinity value of Apple and Grape is $0.8 \cdot 0.8 = 0.64$ as NT relationship is defined between Apple and Edible_Fruit and between Grape and Edible_Fruit, whose weight is 0.8. The other semantic values are calculated in an analogous way. According to this model, we note that the semantic affinity of the two Apple concepts is 1 as their names coincide.

Apple-q/Apple-o	Apple	Banana	Brazil	Computer	Edible_Fruit	Grape	IBM	Italy	Mobile_Computer	Pineapple	Product	Region	USA
Apple	1.0	0.64	-	-	0.8	0.64	-	-	-	0.64	0.512	-	-
Food	-	-	-	-	-	-	-	-	-	-	-	-	-
Fruit	0.64	0.64	-	-	0.8	0.64	-	-	-	0.64	0.8	-	-
Italy	-	-	-	-	-	-	-	1.0	-	-	-	-	-

Table 2. Surface matching results for the running example

Shallow Matching. The shallow matching is defined to take into account concept names and concept properties. With this model, we want a more accurate level of matching, by taking into account not only the linguistic features but also information about the presence of properties and about their cardinality constraints. For property comparison, each property $p_i \in P(c)$ is matched against all properties $p_j \in P(c')$ using (1) and (3), and the best matching value $m(p_i)$ is considered for the evaluation of $SA_{c,c'}$, as follows:

$$m(p_i) = \max\{\mathcal{A}(n_{p_i}, n_{p_j}) \cdot \mathcal{C}(p_i, p_j)\}, \forall p_j \in P(c') \tag{5}$$

where n_{p_i} and n_{p_j} denote the names of p_i and p_j, respectively. $SA_{c,c'}$ is evaluated by the shallow matching as the weighted sum of the linguistic affinity of c and c', calculated using (1), and of their contextual affinity, calculated as the average of the property best matching values computed using (5), that is:

$$SA_{c,c'} = W_{la} \cdot \mathcal{A}(n_c, n_{c'}) + (1 - W_{la}) \cdot \frac{\sum_{i=1}^{|P(c)|} m(p_i)}{\mid P(c) \mid} \tag{6}$$

Example. The matching models from shallow to intensive have been applied to the running example using the following weights values for contextual features:

strong properties and weak properties have been associated with the weights $W_{\mathsf{sp}} = 1.0$ and $W_{\mathsf{wp}} = 0.5$, respectively. For semantic relations, $W_{\mathsf{same-as}} = 1.0$, $W_{\mathsf{kind-of}} = 0.8$, $W_{\mathsf{part-of}} = 0.5$, and $W_{\mathsf{associates}} = 0.3$. These are the default weights in H-Match. For their definition, we followed a method similar to the one used for weighting terminological relationships: we defined specific values for each weight and then we tested them on several real cases, by choosing as default values those that exhibited best behavior in most cases. Furthermore, for the shallow matching and the other two remaining models, we have configured H-Match with a W_{la} value of 0.5, which is used as a default in H-Match as it guarantees an equilibrated balancing of linguistic and contextual affinity in the matching process.

All the results produced for the running example using the shallow matching model are shown in Table 3. As an example of semantic affinity evaluation using the shallow matching model, we continue to consider the case of matching concept Apple of Apple-q with the concept Grape of Apple-o. Both the concepts have a weak property in their contexts, i.e., origin for Apple and provenance for Grape. By applying the term affinity function (1) and the closeness function (3), the matching value of these two properties is evaluated as follows $\mathcal{A}(\mathsf{origin,provenance}) \cdot \mathcal{C}(\mathsf{weak\_property,weak\_property}) = 0.8 \cdot 1.0 = 0.8$. Since these properties are the only elements in the contexts of both Grape and Apple, this is also the best matching value. According to this, the semantic affinity between Apple and Grape is computed as $(0.5 \cdot 0.64) + (0.5 \cdot 0.8) = 0.72$. As we can see,

Apple-q/Apple-o	Apple	Banana	Brazil	Computer	Edible_Fruit	Grape	IBM	Italy	Mobile_Computer	Pineapple	Product	Region	USA
Apple	0.5	0.72	-	-	0.4	0.72	-	-	-	0.72	0.256	-	-
Food	-	-	-	-	-	-	-	-	-	-	-	-	-
Fruit	0.32	0.32	-	-	0.4	0.32	-	-	-	0.32	0.4	-	-
Italy	-	-	-	-	-	-	-	0.5	-	-	-	-	-

Table 3. Shallow matching results for the running example

with this model the semantic affinity value of Apple and Grape is increased with respect to the previous value obtained using the surface model. This because we have taken into account also the presence of matching properties in their context. We also note that the two apple concepts are now less matching than before, because we are able to capture differences due to the context properties.

Deep Matching. The deep matching model is defined to take into account concept names and the whole context of concepts, that is, both properties and semantic relations. Each element $e_i \in Ctx(c)$ (i.e., a property or an adjacent) is compared against all elements $e_j \in Ctx(c')$ using (1) and (3) and the best matching value $m(e_i)$ is considered for the evaluation of $SA_{c,c'}$, as follows:

$$m(e_i) = \max\{\mathcal{A}(n_{e_i}, n_{e_j}) \cdot \mathcal{C}(e_i, e_j)\}, \forall e_j \in Ctx(c') \tag{7}$$

where n_{e_i} and n_{e_j} denote the names of e_i and of e_j, respectively. With the deep matching model, $SA_{c,c'}$ is evaluated as the weighted sum of the linguistic affinity of c and c', calculated using (1), and of their contextual affinity, calculated as the average matching value for the elements of the context of c using (7), that is:

$$SA_{c,c'} = W_{la} \cdot \mathcal{A}(n_c, n_{c'}) + (1 - W_{la}) \cdot \frac{\sum_{i=1}^{|Ctx(c)|} m(e_i)}{\mid Ctx(c) \mid} \tag{8}$$

Example. All the results produced by the deep matching on the running example ontologies are shown in Table 4. Considering the contexts of Apple in Apple-q and of Grape in Apple-o, H-Match searches in the context of Grape for the best matching element for each element of the context of Apple. It is simple to verify that the best matching element for Fruit is Edible_Fruit, and that the best matching element for origin is provenance, while Food has not any matching element in the context of Grape. By applying the term affinity function (1) and the closeness function (3), we obtain that the best matching value is 0.8 both for Fruit and for origin. Then, considering that the context of Apple is composed by 3 elements, the contextual affinity is given by $\frac{0.8+0.8}{3} = 0.53$. The linguistic affinity value between Apple and Grape is 0.64, so that the final semantic affinity is calculated as $(0.5 \cdot 0.64) + (0.5 \cdot 0.53) = 0.59$. The main advantage of the deep model in the example is that it emphasizes the difference between Banana, Pineapple, Grape, and Apple in Apple-o that is due to the fact that the former three are fruits in the ontology, while the last one is a computer brand. This is evident by taking into account the results obtained for Fruit in Apple-q which has a high semantic affinity value with Banana, Pineapple, and Grape, and a low affinity value with Apple in Apple-o.

Intensive Matching. The intensive matching model is defined to take into account concept names, the whole context of concepts, and also property values, in order to exhibit the highest accuracy in semantic affinity evaluation. In fact, by adopting the intensive model not only the presence and cardinality of properties, but also their values are considered to produce the resulting semantic affinity value. Given two concepts c and c', the intensive matching calculates a comprehensive matching value for the elements of the context of c such as in (7) as well as a matching value $v(p_i)$ for each property $p_i \in P(c)$. The matching value $v(p_i)$ is calculated as the highest value obtained by composing the affinity of the name n_{p_i} and the value v_{p_i} of p_i with the name n_{p_j} and the value v_{p_j} of each property

Apple-q/Apple-o	Apple	Banana	Brazil	Computer	Edible_Fruit	Grape	IBM
Apple	0.5	0.59	-	-	0.53	0.59	-
Food	-	0.4	-	-	0.32	0.4	-
Fruit	0.32	0.72	-	-	0.72	0.72	-
Italy	-	-	-	-	-	-	-

Apple-q/Apple-o	Italy	Mobile_Computer	Pineapple	Product	Region	USA
Apple	-	-	0.59	0.386	-	-
Food	-	-	0.4	0.4	-	-
Fruit	-	-	0.72	0.8	-	-
Italy	0.5	-	-	-	-	-

Table 4. Deep matching results for the running example

$p_j \in P(c')$, respectively. For property values comparison, we exploit the term affinity function (1) if the property value is the name of a referenced concept, and the datatype compatibility function (2) if the property value is a datatype, that is:

$$v(p_i) = \begin{cases} \max\{\mathcal{A}(n_{p_i}, n_{p_j}) \cdot \mathcal{A}(v_{p_i}, v_{p_j})\}, \forall p_j \in P(c') \text{ iff } v_{p_i} \text{ is a reference name} \\ \max\{\mathcal{A}(n_{p_i}, n_{p_j}) \cdot \mathcal{T}(v_{p_i}, v_{p_j})\}, \forall p_j \in P(c') \text{ iff } v_{p_i} \text{ is a datatype} \end{cases} \tag{9}$$

$SA_{c,c'}$ is evaluated by the intensive matching as the weighted sum of the linguistic affinity of c and c', calculated using (1), and of their contextual affinity, calculated as the average of the matching values for the elements of the context of c using (7) and for the property values calculated using (9), that is:

$$SA_{c,c'} = W_{la} \cdot \mathcal{A}(n_c, n_{c'}) + (1 - W_{la}) \cdot \frac{\sum_{i=1}^{|Ctx(c)|} m(e_i) + \sum_{j=1}^{|P(c)|} v(p_j)}{|Ctx(c)| + |P(c)|} \tag{10}$$

Example. All the results produced by the intensive matching for concepts of the running example ontologies are shown in Table 5. Let us consider again Apple in Apple-q and Grape in Apple-o of the running example. Using the intensive matching, the contextual affinity of these two concepts must consider also the value of the properties origin and provenance, i.e., the concept Italy. As shown above, the affinity between these two property values is given by the following formula: $\mathcal{A}(\text{origin,provenance}) \cdot \mathcal{A}(\text{Italy,Italy}) = 0.8 \cdot 1.0 = 0.8$. The context of Apple is composed by 3 elements, but, in the intensive model, we have to sum to this number also the number of properties, that is 1. For this reason, the contextual affinity of Apple and Grape is given by $\frac{0.8+0.8+0.8}{4} = 0.6$. The linguistic affinity value between Apple and Grape is 0.64, so that the final semantic affinity is calculated as $(0.5 \cdot 0.64) + (0.5 \cdot 0.6) = 0.62$. The main advantage of the intensive model with respect to the deep model, is that we now capture the difference between Banana, Pineapple and Grape. In fact, all these concepts have in common with Apple of Apple-q the fact that they are fruits, but Grape has a higher semantic affinity with Apple because they both origin from Italy, while Banana and Pineapple origin from Brazil.

Apple-q/Apple-o	Apple	Banana	Brazil	Computer	Edible_Fruit	Grape	IBM	Italy	Mobile_Computer	Pineapple	Product	Region	USA
Apple	0.5	0.52	-	-	0.5	0.62	-	-	-	0.52	0.356	-	-
Food	-	0.4	-	-	0.32	0.4	-	-	-	0.4	0.4	-	-
Fruit	0.32	0.72	-	-	0.72	0.72	-	-	-	0.72	0.8	-	-
Italy	-	-	-	-	-	-	-	0.5	-	-	-	-	-

Table 5. Intensive matching results for the running example

3.4 Matching Policies

The set of parameters to configure the current execution of H-Match for a given matching case is called *matching policy*. A matching policy is a 4-tuple of the form $\langle model, W_{la}, t, mapping \rangle$, where:

- *model* $\in${surface, shallow, deep, intensive} denotes the matching model to be used for H-Match execution;
- $W_{la} \in [0,1]$ denotes the linguistic affinity weight to be used for setting the relevance of the linguistic affinity, and, consequently, the one of the contextual affinity;
- $t \in (0,1]$ denotes the matching threshold value to be used in order to cut-off from the results the matching concepts having a low value of semantic affinity, and thus considered poorly relevant;
- *mapping* $\in${one-to-one, one-to-many} denotes the kind of mapping to be determined at the end of the matching process.

In the context of open networked systems, each node can specify its own policy directly within the knowledge discovery request, in order to force the configuration of H-Match at the destination node as described in Section 4.

3.5 Considerations on the Running Example

The main challenging issue in the running example that we have presented above is to capture the difference between the meaning of the concept Apple in Apple-q and the meaning of Apple in Apple-o. The two concepts have the same name, but in Apple-q apple is a kind of fruit, while in Apple-o it is a brand of computer. For this reason when we compare Apple-q and Apple-o, the fruit Apple is expected to be more similar to Banana, Pineapple, and Grape than to Apple in Apple-o. Moreover, we expect to have a higher similarity between Apple and Grape, since they both origin from Italy, while Banana and Pineapple origin from Brazil. In order to achieve these goals, the matching based on the linguistic features alone would fail. In fact with the surface matching, we obtain that the best matching for Apple in Apple-q is Apple in Apple-o, due to the synonymy between their names. However, the surface matching enriches the information provided in the thesaurus, because it captures a semantic affinity between Apple and the other fruits even if they do not have any terminological relationship between their names in the thesaurus. A first refinement of the results is given by the shallow matching model. In this model, we are able to disambiguate the meaning of the apple concepts, by detecting also a high affinity between Apple and the other fruits. With the shallow model, we do not capture the affinity between the concept Fruit and the different types of fruits in Apple-o because we do not have any property in the context of Fruit. With the deep model, we consider also semantic relations. This gives us the possibility to strengthen the affinity between Fruit and Banana, Pineapple, and Grape, by exploiting the semantic relation that holds between Fruit and Apple in Apple-q. The best matching concepts for Apple are still Banana, Pineapple, and Grape, although we do not capture the higher affinity between Apple and Grape. This goal is achieved by means of the intensive matching model, because it captures the fact that these concepts have two similar properties (i.e., origin and provenance) and that these properties have the same value (i.e., Italy). The example shows how the matching models of H-Match can be used to adapt the algorithm to the specific features of the ontologies to be matched. A further discussion on the applicability of the different matching models is given in Section 4.

4 Application of H-Match to Knowledge Discovery in Open Networked Systems

In this section, we present the query-based approach to knowledge discovery and sharing we developed in the framework of the Helios open system which relies on H-Match for ontology matching. Subsequently, we discuss the design principles that we followed for the implementation of H-Match in Helios.

4.1 Query-Based Knowledge Discovery

In Helios, independent peers with equal role and capabilities cooperate by sharing their information resources (e.g., data, documents) described through peer ontologies. Each node provides its own ontology describing the information resources to be shared and interacts with the other members of the system by sending *probe queries*. A probe query provides an ontological description of target concept(s) of interest for the peer. The Helios probe query template is reported in Figure 3 and it is composed of the following clauses:

- *Find:* list of target concept(s) names.
- *With:* (optional) list of properties of the target concept(s).
- *Where:* (optional) list of conditions to be verified by the property values, and/or (optional) list of concepts related to the target by a semantic relation.
- *Matching policy:* (optional) specification of the H-Match configuration requested for the evaluation of the query.

Probe query template

Find	target concept name [, ...]
[*With*	⟨property name⟩ [, ...]]
[*Where*	condition,
	⟨related concept, semantic relation name⟩ [, ...]]
[*Matching policy*	⟨ model, W_{la}, t, mapping ⟩]

Fig. 3. The reference probe query template

The answer to a probe query is list of concepts that match the target. As described in Figure 4, the structure of the Helios answer template contains the following clauses:

- *Concept:* name of the matching concept.
- *Properties:* (optional) list of properties of the matching concept.
- *Adjacents:* (optional) list of concepts related to the matching concept by a semantic relation.
- *Matching:* set of pairs ⟨target concept, affinity value⟩, specifying the target concept with which the matching concept matches, together with the corresponding affinity value.
- *Matching policy:* (optional) the matching policy adopted for the evaluation of the query.

Probe answer template

{*Concept*	matching concept name
[*Properties*	⟨property name⟩ [, ...]]
[*Adjacents*	⟨related concept, semantic relation name⟩ [, ...]]
Matching	⟨target concept, affinity value⟩[, ...]
[*Matching policy*	⟨ model, W_{la}, t, mapping ⟩]}

Fig. 4. The reference probe answer template

If a peer is interested in discovering nodes capable of providing information resources semantically related to a given target, it composes and submits to the system a probe query according to the query template of Figure 3. Receiving a probe query, a peer invokes the H-Match algorithm to compare such a request against the concepts contained in its peer ontology in order to identify whether there are concepts matching the target. In particular, the *Find*, *With*, and *Where* clauses are used to derive a H-Model description of the target concept(s), while the matching policy to apply is derived from the *Matching policy* clause of the probe query, if specified [7]. As a result, for each target concept of the probe query, H-Match returns a (possibly empty) ranked list of matching concepts semantically related to the target (that is, those concepts whose semantic affinity value exceeds the threshold specified in the adopted matching policy). Depending on the kind of mapping specified in the policy, this ranked list can contain either one single concept (one-to-one mapping policy) which is the best-matching concept for the target, or a set of concepts (one-to-many policy), which are all best-k matching concepts for the target. Finally, the results of H-Match are organized according to the probe answer template of Figure 4, and such an answer is replied back to the requesting peer. Collecting query replies from answering peers, the requesting peer evaluates the results and decides whether to further interact with those peers found to be relevant in order to access the specific information resources. A discussion on how the access to information resource data takes place once the knowledge discovery process has been completed is out of the scope of this paper. For further details, the reader can refer to [20].

4.2 H-Match Implementation Design

The H-Match algorithm and related techniques have been implemented in C++ in the framework of the Helios project. In order to acquire the OWL ontology descriptions in H-Model, we exploit the OWL APIs commonly used to this end also by other ontology matching tools [10]. In this section, we discuss the design principles that we followed for implementing H-Match.

Creation and management of the thesaurus. This functionality exploits the WordNet C library, distributed by the WordNet group at the Princeton University [8]. The library provides the basic functionalities to access the lexical database and to find the relations holding among terms. Our implementation extracts from the ontologies to be matched the names of the elements, defines the thesaurus entries, and exploits WordNet for the definition of the terminological relationships among them, as described in Section 3. The thesaurus is represented as a graph where the nodes are the entries derived from entity names and the edges are the weighted terminological relationships. A specific problem that is addressed by the thesaurus management functionality is related to terms that are not included in WordNet (e.g., some acronyms). We have taken into account three main options:

[7] If the matching policy is not specified in the query, the receiving peer can autonomously select the policy to apply according to internal criteria (e.g., workload, bandwidth).

[8] http://wordnet.princeton.edu/

i) off-line manual extension of the WordNet-based thesaurus with the main terms of the peer ontology domain not included in WordNet; ii) semi-automated extension of the WordNet-based thesaurus with the main terms of the peer ontology domain not included in WordNet by referring to a domain-specific vocabulary, if available; iii) use of syntax-based techniques (e.g., string similarity) to recognize and manage these terms. Currently, we enforce the first option, by providing some basic editing functionalities for off-line thesaurus extension.

Policy management. This functionality has the aim of managing the matching policies for the H-Match configuration. The matching policies are managed through the configure method of the class HMatch.

Representation and storage of ontology concepts. This functionality is based on the idea to represent an ontology as a set of C++ objects, called ConceptVectors. Each concept vector represents a concept in the ontology, together with its context in terms of properties and semantic relations. In our implementation, each H-Match execution is characterized by two set of concept vectors, called target and ontology, respectively. The first set contains the concepts that are the target of the matching process, while the second set contains the concepts that are the basis of the matching process.

Affinity functions. This functionality provides the functions that are used for evaluating both the linguistic and the contextual affinity between two concepts, working on concept vectors. In particular, for each target concept, the matching process calculates the semantic affinity value with the concepts that compose the ontology.

A main portion of the UML class diagram of the C++ classes implementing the H-Match functionalities is shown in Figure 5. The HMatch class is used for representing the process of matching two ontologies by means of the H-Match algorithm. The class is configured by means of a configure function that sets the value associated with the weight of the linguistic affinity (WLA) and the threshold from a MatchingPolicy object. Each HMatch instance is then associated with an instance of the LinguisticAffinity object, which is used to represent the $\mathcal{A}(n, n')$ function described in (1). This association gives HMatch the capability to evaluate the linguistic affinity holding between two concepts. The LA method returns the linguistic affinity between two ConceptVector as a float number in the range [0,1]. In a similar way, the CA function calculates the contextual affinity between two concepts and requires a third parameter that specifies the matching model to be used. Finally, the comprehensive semantic affinity value is provided by means of the SA method that invokes LA and CA, respectively. The evalResults method is exploited for calculating the semantic affinity value for each target conceptand each concept of the ontology. The complete set of results are then stored, together with WLA, threshold, and model used, in the Results object. Each result is a triple of the form ⟨query concept, ontology concept, semantic affinity

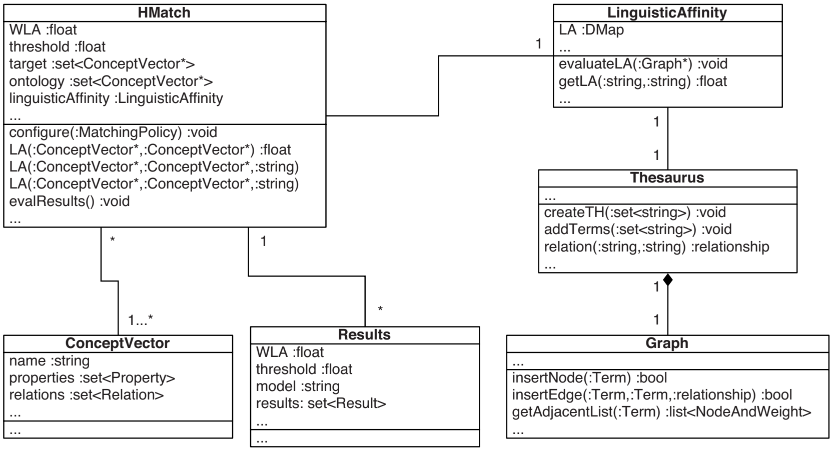

HMatch
WLA :float
threshold :float
target :set<ConceptVector*>
ontology :set<ConceptVector*>
linguisticAffinity :LinguisticAffinity
...
configure(:MatchingPolicy) :void
LA(:ConceptVector*,:ConceptVector*) :float
LA(:ConceptVector*,:ConceptVector*,:string)
LA(:ConceptVector*,:ConceptVector*,:string)
evalResults() :void
...
LinguisticAffinity
LA :DMap
...
evaluateLA(:Graph*) :void
getLA(:string,:string) :float
...
Thesaurus
...
createTH(:set<string>) :void
addTerms(:set<string>) :void
relation(:string,:string) :relationship
...
ConceptVector
name :string
properties :set<Property>
relations :set<Relation>
...
...
Results
WLA :float
threshold :float
model :string
results: set<Result>
...
...
Graph
...
insertNode(:Term) :bool
insertEdge(:Term,:Term,:relationship) :bool
getAdjacentList(:Term) :list<NodeAndWeight>
...
1
1
1
*
1...*
1
*
1
1

Fig. 5. A significant portion of the H-Match UML class diagram

value⟩. The thesaurus of terminological relationships is represented by the Thesaurus object, which is composed by a Graph. The graph is implemented by means of adjacency lists in the class Graph. The class Thesaurus supports basically two main tasks: i) the creation of the thesaurus given a set of strings, and ii) the mapping between the relations found in WordNet and the terminological relationships supported by H-Match, as described in Section 3. A further functionality of the Thesaurus is given by the addTerms method that is used for updating a predefined thesaurus with new term(s) and terminological relationship(s) entries. This function is adopted in the open networked scenario for extending the thesaurus that has been pre-calculated for a peer ontology. In particular, we have a WordNet pre-processing based on the contents of a given peer ontology which is performed off-line. At the time the matching is requested, the extension of the thesaurus is performed with new terms/terminological relationships of the target ontology. Usually the target ontology (i.e., a probe query) contains a limited number of concepts and the extension process can be performed on-line easily. Finally, the thesaurus is associated with a LinguisticAffinity object that exploits a Dijkstra-based algorithm for evaluating the $\mathcal{A}(n, n')$ function described in (1) over the thesaurus graph. These results are then stored in a map that associates a float value in the range [0,1] with each pair of terms in the thesaurus. The map is then refreshed to reflect the thesaurus extensions.

4.3 H-Match **Optimization**

Let n_c be the number of target concepts in the probe query and m_c be the number of concepts in the peer ontology. We developed some optimizations for H-Match to avoid to perform $n_c \times m_c$ matchings. This problem has been addressed by other approaches proposed in literature. For example, the QOM tool [27] proposes several strategies for reducing the number of matching performed by the matchmaker starting from an initial set of candidate mappings. Similarly, we want to reduce the number of atomic matchings required for a probe query by contemporary guaranteeing a high level of accuracy of the results. The idea is to match a target concept $\overline{c}$ of the probe query only against the concepts of the ontology that have a high probability to have a high semantic affinity with $\overline{c}$. To this end, we observed that the problem is analogous to the problem of finding resources over the Web using the PageRank algorithm [28]. In PageRank, the *importance* of a page p is based both on the number of other pages that point to p and on the importance of those other pages [29]. A ranking scheme is then exploited for matching a query against the pages that have a high importance in the rank. For H-Match optimization, we have defined an algorithm, called ConceptRank, that is based on the idea of building a ranking scheme of concepts in a peer ontology and of exploiting it for matching a target concept *only against* the concepts that have a high level of importance in the rank. The rank of a concept is determined by taking into account the number and the weight of ontology relations that point to it. In this step, the difference with respect to PageRank is that the relevance of an edge is given by the weight associated with it. The importance of a concept c is a measure proportional to the number of concepts c_i that have a semantic relation with c and to the importance of c_i. Probe queries are processed by matching each target concept against the peer ontology concepts in the ranking scheme starting from the most important ones and by stopping the matching process when a given number TF of atomic matchings produce a semantic affinity value under the matching threshold. The number TF is called *tolerance factor* and is determined experimentally in order to obtain the best balance between the required reduction of atomic matchings and the quality of the results obtained for a probe query. This process has to take into account the fact that, in the ranking scheme, the most important concepts are the ones that are mostly referenced by the other concepts. For this reason, the most important concepts in the rank have generally the most generic meaning in the ontology. When a probe query is searching for a concept with a specific meaning, the top ranked concepts (i.e., the ones with the most generic meaning) could not be the best answers for the query. In order to address this problem, we have defined two main strategies:

- *Ranking-based strategy.* A target concept $\overline{c}$ is matched against the concepts c_i in the rank starting from the top ranked concept c_0. The matching process ends when the number of atomic matchings that do not provide a semantic affinity value exceeding the threshold is higher than or equal to the tolerance factor TF. A variation of this strategy, called *Surface Ranking-based Strategy*, avoids to match $\overline{c}$ against the peer ontology concepts with the most generic meaning through a revision of the ranking scheme obtained by exploiting the surface matching model.
- *Graph-based Strategy.* In this strategy, we use the ranking scheme just to find the most important concept c_0 that is seen as the concept that has the highest probability to be relevant for the probe query. Then, the concepts that have

to be matched against $\overline{c}$ are chosen by visiting the ontology graph starting from c_0. The process is iterated until the number of atomic matchings that do not provide a semantic affinity value exceeding the threshold is higher than or equal to the tolerance factor TF. Also in this case, a variation, called *Surface Graph-based Strategy*, has been developed for the same purpose of the previous one.

We have compared these strategies in order to determine their impact on the matching process. The results of this experimentation show how the first strategy guarantees a higher level of performance and a lower level of accuracy than the last one. Moreover, the graph-based strategy is more adequate to obtain the most relevant results. The optimization strategies are available in the HELIOS query processing module to pre-configure the peer capabilities at the initialization time.

5 Experimental Results

We analyze the behavior of H-MATCH by performing different tests devoted i) to evaluate the matching models with respect to performance and quality of results in terms of precision and recall, and ii) to compare the H-MATCH results with the results produced by selected ontology matching tools available on the Web.

5.1 Experimental Evaluation of H-MATCH Models

In order to evaluate the effectiveness of the four H-MATCH models, two kind of tests have been executed: i) a *quality test*, where the experiments were devoted to examine precision and recall of the models on real case studies; and ii) a *performance test*, where the H-MATCH models have been analyzed with respect to computation time and scalability in manifold scenarios with different level of complexity.

Quality Test. For what concern the H-MATCH quality test, we have considered two OWL ontologies from the publication domain (i.e., Ka [9], Portal [10]), and we asked to users (i.e., students of our Ontologies and Semantic Web course) to manually define a set of target ontologies (i.e., twenty target ontologies) related to the publication domain, too. Each target ontology describes publication-related concepts with some properties, semantic relations, and property values according to the domain knowledge and experience of the ontology creator. To avoid to be influenced, target ontologies have been composed without considering Ka and Portal ontology contents. Note that the choice of manually constructing small-size target ontologies for extensive experimentation is motivated by the fact that H-MATCH is used for knowledge discovery in the HELIOS open networked system. In such a context, the typical problem is to match a probe query embedding a

[9] http://protege.stanford.edu/plugins/owl/owl-library/ka.owl

[10] http://www.aktors.org/ontology/portal

small-size ontology describing target concept(s) against a large ontology. After the target ontology definition, we asked users to consider Ka and Portal ontologies, and to map each concept of the target ontologies they have defined on one or more concepts belonging to the two reference ontologies according to their intuitive understanding of concept similarity. In this way, we have collected a set of 1:1 mappings between the target ontologies and the reference ontology concepts. The goal of the quality test is to evaluate the effectiveness of H-Match matching process by verifying the overlapping between the results produced by the different H-Match models and the manual mappings. Moreover, we intend to analyze the impact of different threshold values t and linguistic affinity weights W_{la} on the level of overlapping. To this end, we use *precision* and *recall*, which are the measures commonly adopted for matching evaluation [10] derived from the classical definitions of Information Retrieval [30]. In particular, *precision* is defined as the ratio of the number of relevant matching concepts automatically found by H-Match to the total number of matching concepts automatically found. *Recall* is defined as the ratio of the number of relevant matching concepts automatically found by H-Match to the total number of matching concepts (i.e., mappings) manually defined.

In Figure 6, we show the precision of the H-Match matching models with a linguistic affinity weight which varies from $W_{la} = 0.1$ to $W_{la} = 0.8$ and the threshold $t = 0.6$. These values show the number of manual mappings which

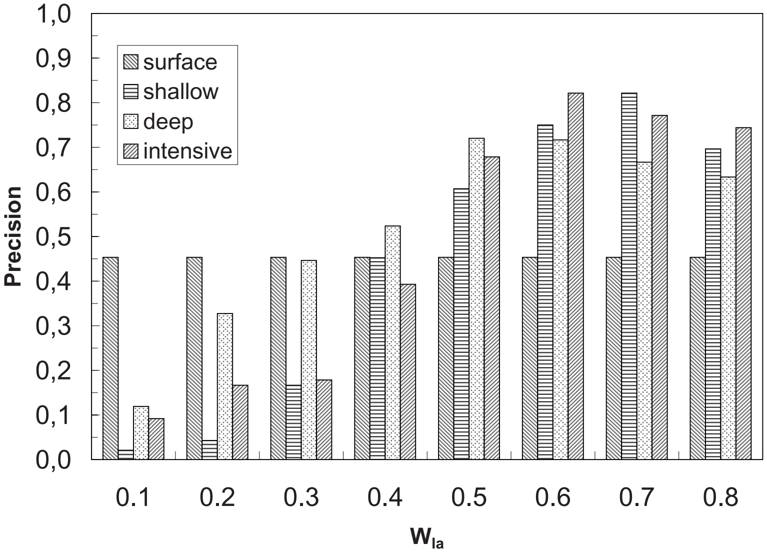


Fig. 6. The precision of the H-Match matching models with $t = 0.6$

have been correctly identified by H-Match with respect to the total number of results provided by the algorithm. We observe that the surface matching is not affected by the variation of the W_{la} parameter and can ensure a precision equal to 45%. This is due to the fact that the surface matching always works with the parameter $W_{la} = 1$ and contextual features are not considered in the matching evaluation. For what concern the other matching models, H-Match can guarantee a high level of precision in correspondence of high W_{la} values. In particular, we observe that with $W_{la} \geq 0.4$, H-Match can achieve a precision of 82%.

In Figure 7, we measure the H-Match recall in correspondence of different linguistic affinity weights and with the threshold $t = 0.3$. The results show the number of manual mappings which have been correctly identified with respect to the total number of mappings manually defined. Independently from the values of W_{la}, we observe that the recall of each matching model is over the 80%, and the deep matching model can achieve a recall equal to 99.1% when $W_{la} = 0.4$.

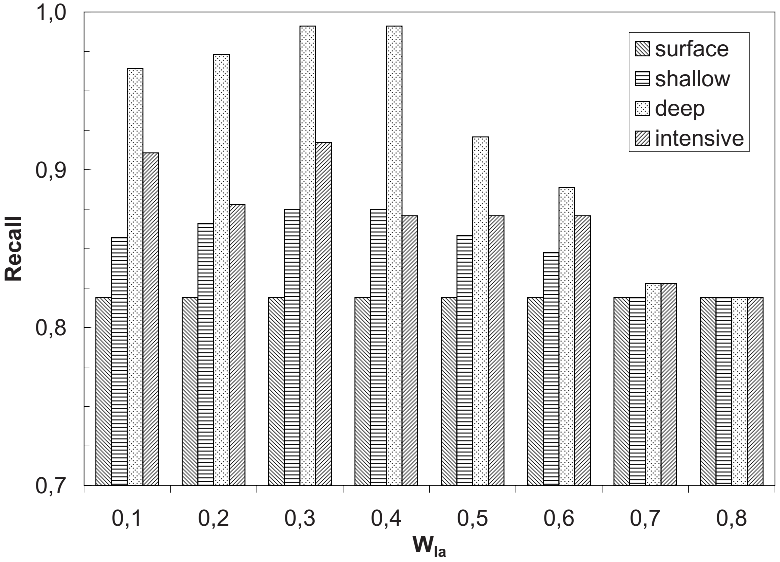


Fig. 7. The recall of the H-Match matching models with $t = 0.3$

From these figures, we observe that the choice of the correct value for the threshold t and for the linguistic affinity weight W_{la} depends on the goal of the matching case. If we are interested in very precise results, we have to set high values (i.e., ≥ 0.6) both for threshold and for linguistic affinity weight. On the opposite, if we are interested in accuracy of results, lower values (i.e., $0.3 \leq t, W_{la} \leq 0.5$) for both parameters work better. In both cases, deep and intensive matching are to be preferable and can provide better results than surface and shallow models. We want to stress that H-Match has been implemented to be interactively configured, in order to suit the matching process to the requirements of the specific matching case.

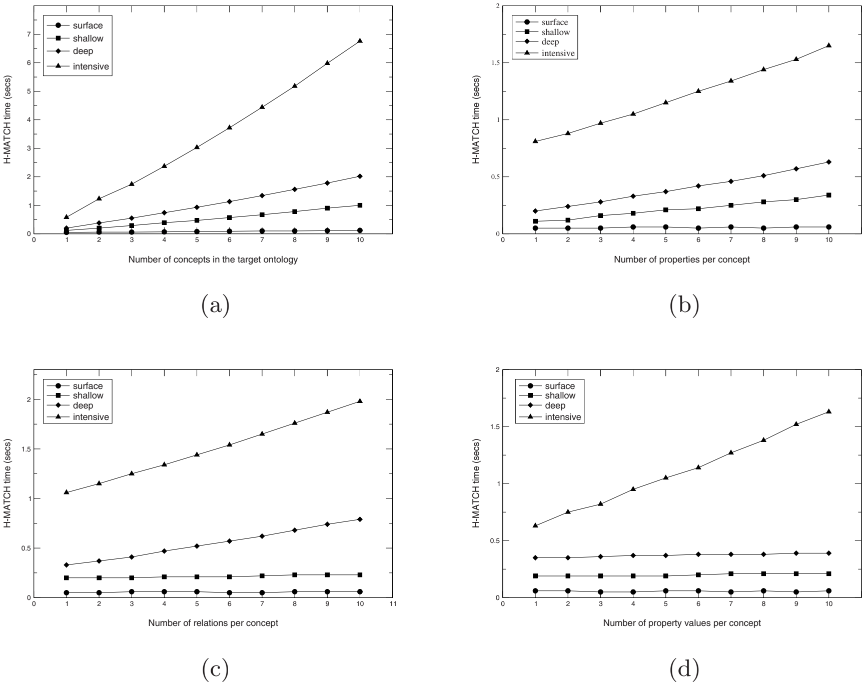
Performance Test. The semantic affinity evaluation performed by H-Match is based on the comparison of all the concepts and corresponding contexts contained in the two ontologies to be considered. For this reason, the computation time is affected by the number of elements to be compared and by the adopted matching model. The goal of the performance test is to focus the attention on the performance differences between the four matching models for the aspects related to the contextual affinity evaluation. To this end, we analyze the matching models when comparing a large reference ontology with a great number of small target ontologies which differ in complexity of concept contexts. In this test, the linguistic affinity is computed only once as it is common to all matching models and, as such, it is not considered in discriminating between the different matching models [11]. The test is characterized by the following features:

- The reference ontology is the W3C Wine ontology [12]. We have selected this well known ontology because it provides a relevant number of concepts (more than one hundred concepts) with an adequate richness in terms of properties, semantic relations, and property values per concept (an average of five properties, two semantic relations, and six property values per concept, respectively).
- The target ontologies used in the comparison with the reference Wine ontology are randomly composed and vary according to four different complexity measures: number of concepts in the ontology and number of properties, semantic relations, and property values per concept, respectively.
- The test has been executed on a Dual Xeon machine at 2.80 GHZ with 1GB RAM and SCSI disks.

The diagrams reported in Figure 8 show the computation time of the H-Match models by varying one complexity measure while keeping fixed the other three measures. In Figure 8(a), we have measured the computation time required for comparing the reference ontology with a target ontology which varies in the number of concepts and with a fixed number of properties, semantic relations, and property values per concept. In the remaining diagrams, we have fixed the number of target ontology concepts and the complexity measure which varies

[11] However, based on the test cases performed on real examples, such as those presented in Section 5.2, where the linguistic affinity is calculated each time, we found that the average time required for matching (including LA computation) is in the range of 0.6 seconds for quite complex ontologies (ontologies with more than sixty concepts to one hundred and more concepts)

[12] http://www.w3.org/TR/2003/WD-owl-guide-20030210/wine.owl



(a) (b) (c) (d)

Fig. 8. Comparison of the H-Match models

is the number of properties (Figure 8(b)), semantic relations (Figure 8(c)), and property values (Figure 8(d)), respectively.

We stress that, in the worst case, all the models, including the intensive matching, observe a linear growth of their computation times in correspondence of the increase of the elements to be evaluated. Furthermore, the surface, shallow, and deep matching are also scalable, in that the semantic affinity evaluation restricted to concept names, properties, and semantic adjacents is a very efficient task. On the contrary, the intensive matching computation times are higher and not comparable with the corresponding values of the other matching models. This means that the semantic affinity evaluation of property values has a great impact on the matching performances. Anyway, we observe that there is a real difference between the computation times of Figure 8(a) and the remaining diagrams: the real impact on matching performances is due to the number of concepts to be processed in the target ontology rather than to the variations in concept context definition.

Considerations. Performance and quality tests show that the choice of the most suitable matching model is a key factor for obtaining relevant matching results. This depends on the level of detail of the ontology descriptions to be compared as well as on the expected degree of precision and recall of the results. Furthermore, the appropriate configuration of the threshold and of the linguistic affinity weight have an impact on the quality of H-Match results. When adopting H-Match, the trade-off between high performances, high precision, and high recall has to be evaluated. In scenarios where a rapid response time is required (e.g., open networked systems with discovery queries), some lacks in matching precision and recall can be admitted in turn of high performances during the semantic affinity evaluation. On the opposite, when computation time is not a critical constraint, we can apply the matching model which best suits the particular application scenario. In Table 6, we summarize the main features of the matching models and their corresponding suggested scenarios. The surface model is useful when only concept names are to be considered. It requires few computational resources since neither concept properties nor semantic relations are considered. This model is well suited, for example, to perform an initial ontology comparison to decide whether it is worth to perform a deeper analysis. If the ontology is constituted mainly by concepts with a few number of properties and hierarchical relation among concepts, the shallow and deep model allow a good degree of precision without requiring great amount of computational resources. In presence of an articulated ontology, with rich resource descriptions and where relations among concepts are described through property values, the intensive model guarantees the most precise and accurate results, although being the most expensive in term of computation.

	Surface	**Shallow/Deep**	**Intensive**
Ontological description	Poorly structured ontologies with very simple resource description	Schematic ontologies with taxonomic resource description	Articulated ontologies with rich resource description
Kind of matching	Linguistic-driven matching	Linguistic and context-driven matching	Linguistic, value, and context-driven matching
Advantages	High performances	More accurate characterization of matching concepts	High precision and recall

Table 6. Applicability of the matching models

5.2 Comparative Evaluation with Other Tools

In this test, we compare the H-Match algorithm with similar ontology matching tools and we analyze the results produced by applying them to the same test case. The tools considered are *FOAM* [13] (*Framework for Ontology Alignment*

[13] http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/

and Mapping) and *OLA* [14] (*OWL Lite Alignment*). The choice of these tools has been led by the following considerations:

- The ontology mapping approach proposed by these tools is similar to H-Match [15].
- A prototype of these tools is free for download or is available through a Web Service.
- Different test cases and associated experimental results in term of precision, recall, and F1 measure are available on the respective Web site.

Two different experimentations have been performed to compare H-Match with FOAM and OLA, respectively. The goal of each test is to compare two OWL ontologies in order to identify the pairs of matching elements. A list of expected mappings is also specified for evaluating the results provided by the tools in terms of precision, recall, and F1 measure. For what concern precision and recall we refer to the definitions provided in Section 5.1, while *F1 measure* is derived from the classical definition of Information Retrieval [30] and it is defined as follows:

$$F1 = \frac{2pr}{p + r} \tag{11}$$

where p and r represent precision and recall measures, respectively. For each test, we execute H-Match under many different matching policies varying the adopted matching model, the linguistic affinity weight W_{la}, the threshold t, and the kind of mapping determined (i.e., one-to-one, one-to-many). For each test, we report the H-Match policy that provides the best results in terms of precision, recall, and F1 measure and we compare such results with the corresponding results produced by the observed tool.

Comparison with FOAM. The test case adopted in this comparison has been selected from the FOAM Web site and regards the animal domain. In Table 7, we show the results produced by FOAM and H-Match, respectively. The FOAM measurements are obtained by submitting the test case to the FOAM Alignment Web Service [16]. The point of maximum precision of H-Match is obtained with the matching policy ⟨intensive, 0.8, 0.7, one-to-one⟩, while the points of maximum recall and F1 measure are obtained with the policy ⟨deep, 0.5, 0.3, one-to-one⟩. We can observe that FOAM has better results than H-Match for what regards precision, while recall and F1 measure are a little bit higher for H-Match. In general, these results confirm the impression we stressed in the experiments of Section 5.1: deep and intensive matching can ensure appreciable results in terms of precision and recall when W_{la} and t are properly set. In particular, precision increases with high values both for linguistic affinity weight and threshold (e.g., $W_{la} = 0.8$ and $t = 0.7$), while low values for both these

[14] http://www.iro.umontreal.ca/∼owlola/

[15] FOAM and OLA are also discussed in Section 6 where an analytical comparison with H-Match is provided.

[16] http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/service.htm

	Precision	**Recall**	**F1 measure**
FOAM			
Point of maximum precision	1.0	0.58	0.74
Point of maximum recall	0.76	0.67	0.71
Point of maximum F1	1.0	0.58	0.74
H-MATCH			
Point of maximum precision	0.86	0.67	0.75
Point of maximum recall	0.78	0.78	0.78
Point of maximum F1	0.78	0.78	0.78

Table 7. Comparison of H-Match and FOAM results

parameters (e.g., $W_{la} = 0.5$ and $t = 0.3$) are required for obtaining high values of recall. Finally, we note that the one-to-one mapping strategy is always associated to maximum measures of H-Match. Such a mapping strategy allows to obtain more balanced results of precision and recall, thus it positively affects the F1 measures.

Comparison with OLA. The test case adopted in this comparison has been selected from the *Ontology Alignment Contest* [17] and regards the comparison of a reference bibliographic ontology [18] with the more complex Karlsruhe ontology [19]. In Table 8, the results of the comparison between H-Match and OLA are summarized. Precision, recall, and F1 measure related to OLA on this test case are obtained from [8] where the authors indicate that the tool configuration has the intention to led to the best alignment with respect to precision. For what concern H-Match, the points of maximum precision, recall, and F1 measure are all obtained when the adopted matching policy is ⟨intensive, 0.8, 0.8, one-to-one⟩.

	Precision	**Recall**	**F1 measure**
OLA			
	0.5	0.31	0.38
H-MATCH			
Point of maximum precision	0.82	0.74	0.78
Point of maximum recall	0.82	0.74	0.78
Point of maximum F1	0.82	0.74	0.78

Table 8. Comparison of H-Match and OLA results

[17] http://co4.inrialpes.fr/align/Contest/

[18] http://co4.inrialpes.fr/align/Contest/101/onto.rdf

[19] http://co4.inrialpes.fr/align/Contest/303/onto.rdf

We note that H-Match can provide appreciable results both in terms of precision and recall. With respect to our considerations in Section 5.1, we note that the H-Match configuration for the point of maximum recall is anomalous. In this test case, this is due to the fact that the expected mappings are strongly dependent from linguistic features. This means that when the linguistic affinity weight is high (e.g., $W_{la} = 0.8$), even if the threshold has a high value (e.g., $t = 0.8$), the accuracy of H-Match can achieve relevant results.

6 Related Work

In this section, we perform a comparative analysis of the state of the art tools for ontology matching, providing also a critical comparison with H-Match. We perform the comparison in the light of three main criteria: i) the ontology representation formalism adopted by the tools; ii) the semantic complexity supported by the matching process of each tool; iii) the mechanism adopted by each tool for the composition of different similarity measures. For comparison we have selected *PROMPT* [12], *FOAM/QOM* [10, 27, 31], *OLA* [8], *S-Match* [11, 23], *GLUE* [7], and *COMA++* [22]. PROMPT is a framework for multiple ontology management; FOAM/QOM is a tool to full- or semi-automatically align two or more ontologies; OLA is an ontology alignment tool tailored to OWL Lite ontologies; S-Match is a semantic matchmaker tailored to graph-like structures; GLUE is an approach based on machine learning techniques for schema and ontology matching purposes; COMA++ is a combined framework for schema and ontology matching.

Ontology Representation Formalism. In Table 9, we compare the different selected tools with respect to the internal formalism adopted for the ontology representation and with the ontology languages supported. In particular, we show which OWL dialect is supported by each tool and which OWL features are considered in the matching process. The tools presented in Table 9 generally adopt an internal representation, either a tree-based or a graph-based model, of the ontology contents, which in several cases is defined to capture the features of OWL. S-Match and GLUE do not refer explicitly to OWL, but their reference model is compatible with OWL Lite. With respect to the ontology representation formalism, H-Match is in line with the state of the art systems, in that it refers to a graph-based model and provides direct support for OWL.

Features of the Matching Process. Table 10 shows the comparison of the tools with respect to the ontology elements, the linguistic features, and the contextual features that characterize the matching process in each tool. With respect to linguistic features, we compare the tools by taking into account the techniques adopted for determining linguistic similarities and to the level of linguistic analysis provided by each tool. Contextual features refer to the number and type of semantic relations among concepts that are used by each tool in

	OWL Dialect	Internal representation
PROMPT	OWL Lite OWL DL (partial) OWL Full (partial)	Frame-based Graph-based
FOAM/QOM	OWL Lite OWL DL	Karlsruhe Ontology Model [27] (Graph-based)
OLA	OWL Lite	OL-Graph (Graph-based)
S-Match	OWL Lite (no specific support for OWL)	Tree-based
GLUE	OWL Lite (no specific support for OWL)	Tree-based
COMA++	OWL Lite	Graph-based
H-MATCH	OWL Lite OWL DL (partial) OWL Full (partial)	H-Model (Graph-based)

Table 9. Comparison on ontology representation

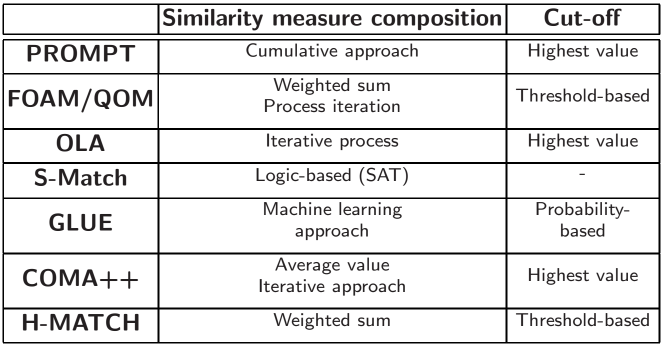
order to determine the contextual similarity between two concepts. Regarding ontology elements that can be matched, all the tools perform the matching process by taking into account concepts and properties. Instances are considered in PROMPT, FOAM/QOM, OLA, GLUE, and COMA++. In the case of H-Match, the choice of considering concepts and properties is motivated by the fact that the algorithm is conceived for matching knowledge requests in the context of open networked systems, where probe queries are used with the aim of acquiring new knowledge from other peers. In fact, a probe query provides basically the schema-level description of one or more concepts of interest. However, H-Match can be easily extended to consider also instance level information in the ontology matching process. This can be achieved by exploiting a combination of linguistic and property value similarity measures already available in the intensive matching model. Considering linguistic features, we distinguish between tools that exploit an external dictionary or a thesaurus taking into account terminological relationships and domain knowledge for linguistic matching, and tools that rely on the syntactic features of labels and identifiers through string matching. PROMPT and GLUE do not adopt any external support for the linguistic analysis. FOAM and COMA++ adopt domain specific vocabularies that can be used for refining the matching results obtained by syntactic analysis. H-Match, as COMA++, S-Match, and OLA, adopts a language-based approach, in particular for the preprocessing of compound terms. S-Match, OLA, and H-Match are similar also with respect to the intensive use of WordNet for the linguistic matching. With respect to contextual features, all the tools take into account the semantic relations holding between concepts. S-Match adopts a logic-based approach that exploits the relations between concepts for automatic reasoning purposes. All the tools refer to property domain and range as well as to taxonomic relations among concepts. FOAM/QOM takes also into account property

	Ontology elements	Linguistic features		Contextual features
		Type of matching	External dictionary	
PROMPT	Concepts Properties Instances	Syntactic (String matching)	-	Property domain Property range Kind-of (Among concepts)
FOAM/QOM	Concepts Properties Instances	Syntactic (String matching)	Domain specific dictionary	Property domain Property range Property hierarchy Same-as Kind-of Individual identity
OLA	Concepts Properties Instances	Syntactic (String matching) Language-based (tokenization, compound terms)	WordNet	Property domain Property range Kind-of
S-Match	Concepts	Syntactic (String matching) Language-based (lemmatization, tokenization, compound terms)	WordNet	Same-as Kind-of Mismatch Overlapping
GLUE	Concepts Properties Instances	-	-	Property domain Property range Same-as Kind-of
COMA++	Concepts Properties Instances	Syntactic (String similarity) Language-based (tokenization)	Domain specific dictionary	Property domain Property range Same-as Kind-of
H-MATCH	Concepts Properties	Language-based (Tokenization, compound terms)	WordNet	Property domain Property range Same-as Kind-of Part-of Associates

Table 10. Comparison on semantic complexity

hierarchies and identity relations among instances. H-Match takes into account property domain and range, and the semantic relations of same-as, kind-of, part-of, and associates in the context of concepts. In particular, the part-of and associates relations are useful to deal with object-oriented ontology specifications, where these relations are typically used. H-Match and FOAM/QOM adopt a similar strategy in discarding some kind of contextual features in order to increase the matching performance. The main difference between the two is related to the way the strategy is enforced. In H-Match, the strategy is chosen at run-time, by selecting the most adequate matching model that has to be adopted at a given invocation time.

Similarity Measures Composition. In Table 11, we compare the different approaches with respect to: i) the mechanism adopted for deriving a comprehensive similarity value out of the different similarity measures and ii) the mechanism adopted for cutting off the useless results. With respect to the similarity measures composition, PROMPT, OLA, and GLUE, even if in different ways, adopt a cumulative and iterative strategy for deriving the comprehensive degree of similarity between two ontology elements. In S-Match the structural matching is seen as a logic proof based on the previously determined relations among concept labels. COMA++ and FOAM are frameworks based on the idea of combining different measures of similarity. In COMA++ the similarity measures are composed in a comprehensive measure by evaluating their average value. In FOAM, the strategy is to perform a weighted sum of the different similarity measures, where the factors are functionally computed. H-Match adopts a weighted sum between linguistic and contextual measures of similarity, where the weights associated with linguistic and contextual affinity are constant. The problem of cutting off the results that are not used to determine mappings is typical of the approaches that provide a measure of similarity between ontology elements. S-Match is based on the idea to discover a semantic relation between two elements of different ontologies, so that there is no need of a cutting off mechanism. In GLUE, mappings represent the probability that an instance of a given element



	Similarity measure composition	**Cut-off**
PROMPT	Cumulative approach	Highest value
FOAM/QOM	Weighted sum Process iteration	Threshold-based
OLA	Iterative process	Highest value
S-Match	Logic-based (SAT)	-
GLUE	Machine learning approach	Probability-based
COMA++	Average value Iterative approach	Highest value
H-MATCH	Weighted sum	Threshold-based

Table 11. Comparison on similarity measure composition

is an instance also of another element. GLUE chooses the most probable mapping, that is the mapping between the elements with the highest probability to represent the same real object in the domain. In PROMPT and OLA, the matching value used for determine the mappings is the highest value that is computed between two elements by means of a cumulative strategy. FOAM/QOM, COMA++, and H-Match provide a measure of similarity between ontology elements in the range $[0, 1]$. In COMA++, mappings are determined between the elements with the highest matching values. In FOAM and H-Match, the cut-off mechanism is based on a threshold that is set as a parameter of the algorithm.

7 Concluding Remarks

In this paper, we have presented the H-Match algorithm and related techniques for matching of independent ontologies in open networked systems. H-Match has been implemented and used for probe query processing in the framework of the Helios networked system for supporting dynamic knowledge discovery and ontology-addressable content retrieval in peer-based systems [19, 20]. The novel contributions and distinguishing features of H-Match can be summarized as follows:

- Fully-automated matching process that can be used as: i) a matchmaker engine of a peer for matching probe queries against peer ontologies for knowledge discovery as occurs in the Helios open networked system; ii) a conventional ontology matching tool for the alignment of two independent ontologies;
- Capability of satisfying as a whole both general requirements of ontology matching per se and peculiar requirements of matching in open networked context. This is achieved by providing a wide spectrum of metrics suited for dealing with many different matching scenarios where the number and type of ontology features that can be exploited during the matching process is not known in advance;
- Capability of being dynamically configured for adaptation to the semantic complexity of the ontologies to be compared. This is achieved by automatically selecting the matching configuration according to a policy embedded in the incoming request which can vary, also for the same ontology, each time a new request is submitted to the system.

Future work will regard the enrichment of the ontology features supported by H-Match; the semantic query routing; the semantic community definition. With respect to the first issue, we will consider also instance-level information of ontology specifications in the matching process of H-Match, starting from current intensive matching techniques. Other two research issues are related to the use of H-Match for enforcing semantic query routing and for semantic community formation, which are hot research topics in open networked systems. Some initial results on these issues are presented in [32, 33].

Acknowledgments. The authors would like to thank the anonymous referees for their detailed and insightful comments. A special acknowledgment is due to Gianpaolo Racca for his invaluable collaboration to the development of H-MATCH in the framework of the HELIOS project. This paper has been partially funded by "Wide-scalE, Broadband, MIddleware for Network Distributed Services (WEB-MINDS)" FIRB Project funded by the Italian Ministry of Education, University, and Research, and by NoE INTEROP, IST Project n. 508011 - 6th EU Framework Programme.

References

[1] Broekstra, J., et al.: A Metadata Model for Semantics-Based Peer-to-Peer Systems. In: Proc. of the 1st WWW Int. Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID 2003), Budapest, Hungary (2003)

[2] Nejdl, W., et al.: EDUTELLA: a P2P Networking Infrastructure Based on RDF. In: Proc. of the 11th Int. World Wide Web Conference (WWW 2002), Honolulu, Hawaii, USA (2002)

[3] Mitre, J., Navarro-Moldes, L.: P2P Architecture for Scientific Collaboration. In: Proc. of the 13th Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2004), Modena, Italy, IEEE Computer Society (2004) 95–100

[4] Iamnitchi, A., Ripeanu, M., Foster, I.T.: Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. In: Proc. of the 1st Int. Workshop on Peer-to-Peer Systems IPTPS 2002, Cambridge, MA, USA (2002) 232–241

[5] Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Logical Foundations of Peer-To-Peer Data Integration. In: Proc. of the 23rd ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS 2004), Paris, France (2004) 241–251

[6] Motik, B., Maedche, A., Volz, R.: A Conceptual Modeling Approach for Semantics-Driven Enterprise Applications. In Springer, ed.: Proc. of Confederated Int. Conferences DOA, CoopIS and ODBASE 2002, Irvine, California, USA (2002) 1082–1099

[7] Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to Map between Ontologies on the Semantic Web. In: Proc. of the 11th Int. World Wide Web Conference (WWW 2002), Honolulu, Hawaii, USA (2002) 662–673

[8] Euzenat, J., Loup, D., Touzani, M., Valtchev, P.: Ontology Alignment with OLA. In: Proc. of the 3rd ISWC Workshop on Evaluation of Ontology-based Tools (EON 2004), Hiroshima, Japan (2004)

[9] Do, H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: Proc. of 28th Int. Conference on Very Large Databases (VLDB 2002), Hong Kong, China (2002)

[10] Ehrig, M., Sure, Y.: Ontology Mapping - An Integrated Approach. In: Proc. of the 1st European Semantic Web Symposium, Heraklion, Greece, Springer Verlag (2004) 76–91

[11] Giunchiglia, F., Shvaiko, P.: Semantic Matching. Knowledge engineering review **18** (2003) 265–280

[12] Noy, N.F., Musen, M.A.: The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. International Journal of Human-Computer Studies **59** (2003) 983–1024

[13] Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. VLDB Journal **10** (2001) 334–350

[14] Shvaiko, P., Euzenat, J.: A Survey of Schema-based Matching Approaches. Journal on Data Semantics (JoDS) (2005)

[15] Noy, N.F.: Semantic Integration: a Survey of Ontology-based Approaches. SIGMOD Record Special Issue on Semantic Integration (2004)

[16] Kalfoglou, Y., Schorlemmer, M.: Ontology Mapping: the State of the Art. The Knowledge Engineering Review **18** (2003)

[17] INTEROP - Network of Excellence: State of the Art and State of the Practice Including Initial Possible Research Orientations. Deliverable D8.1, NoE INTEROP - IST Project n. 508011 - 6th EU Framework Programme (2004)

[18] Smith, M.K., Welty, C., McGuinness, D.L., (eds.): OWL Web Ontology Language Guide (2004) World Wide Web Consortium (W3C), http://www.w3.org/TR/owl-guide/.

[19] Castano, S., Ferrara, A., Montanelli, S., Zucchelli, D.: HELIOS: a General Framework for Ontology-based Knowledge Sharing and Evolution in P2P Systems. In: Proc. of the 2nd DEXA Int. Workshop on Web Semantics (WEBS 2003), Prague, Czech Republic, IEEE Computer Society (2003)

[20] Castano, S., Ferrara, A., Montanelli, S.: Dynamic Knowledge Discovery in Open, Distributed and Multi-Ontology Systems: Techniques and Applications. In: Web Semantics and Ontology. Idea Group (2005) To Appear.

[21] Castano, S., De Antonellis, V., De Capitani Di Vimercati, S.: Global Viewing of Heterogeneous Data Sources. IEEE Transactions on Knowledge and Data Engineering **13** (2001) 277–297

[22] Aumueller, D., Do, H., Massmann, S., Rahm, E.: Schema and Ontology Matching with COMA++. In: Proc. of SIGMOD 2005 - Software Demonstration, Baltimore, USA (2005)

[23] Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: an algorithm and an implementation of semantic matching. In: Semantic Interoperability and Integration, Schloss Dagstuhl, Germany (2005)

[24] Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM (CACM) **38** (1995) 39–41

[25] Ouksel, A.M., Naiman, C.F.: Coordinating Context Building in Heterogeneous Information Systems. Journal of Intelligent Information Systems **3** (1994) 151–183

[26] Lauer, M.: Designing Statistical Language Learners: Experiments on Noun Compounds. In: Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL 1995), Cambridge, Massachusetts, USA (1995) 47–54

[27] Ehrig, M., Staab, S.: QOM - Quick Ontology Mapping. In: Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004), Hiroshima, Japan (2004)

[28] Page, L., Brin, S., Motwani, R., Winograd, T.: The Pagerank Citation Ranking: Bringing Order to the Web. Technical report, Computer Science Department, Stanford University (1998)

[29] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., Raghavan, S.: Searching the web. ACM Transactions on Internet Technology (2001)

[30] Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley (1989)

[31] Ehrig, M., Haase, P., Stojanovic, N., Hefke, M.: Similarity for Ontologies - A Comprehensive Framework. In: Proc. of the 13th European Conference on Information Systems, Regensburg, Germany (2005)

[32] Castano, S., Ferrara, A., Montanelli, S., Pagani, E., Rossi, G.P., Tebaldi, S.: On Combining a Semantic Engine and Flexible Network Policies for P2P Knowledge Sharing Networks. In: Proc of the 1st DEXA Workshop on Grid and Peer-to-Peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems (GLOBE 2004), Zaragoza, Spain, IEEE Computer Society (2004) 529–535

[33] Castano, S., Montanelli, S.: Semantic Self-Formation of Communities of Peers. In: Proc. of the ESWC Workshop on Ontologies in Peer-to-Peer Communities, Heraklion, Greece (2005)