State of the Art: Automatic Ontology Matching

Feiyu Lin

Information Engineering Research Group
Department of Electronic and Computer Engineering
School of Engineering, Jönköping university
Jönköping, SWEDEN

ISSN 1404-0018
Research Report 2007:1

Abstract

This report describes the theoretical foundations and relevant background in the domain of automatic ontology matching. It aims to show the range of ontology matching, matching strategies, and an overview of ontology matching systems. The measures for evaluating ontology matching are discussed. This report also summarizes interesting research questions in the field.

Keywords

Ontologies, Ontology Matching

Table of Contents

1 Introduction

This report describes the theoretical foundations and relevant background in the domain of automatic ontology matching. It aims to show the range of ontology matching, matching strategies, and an overview of ontology matching tools. The measures for evaluating ontology matching are discussed. This report also summarizes interesting research questions in the field.

1.1 Background

When people or machines must communicate between themselves, they need a shared understanding of the same concepts. An ontology can be used to solve this problem. Gruber [28] defined an ontology as:

An ontology is a formal, explicit specification of a shared conceptualization.

Due to an increased awareness of potential ontology applications in industry, public administration and academia, a growing number of ontologies is created by different organizations and individuals. Although these ontologies are developed for various application purposes and areas, they often contain overlapping information, but these different ontologies cannot easily be used together in a new application. Furthermore, ontology users or engineers do not only use their own ontologies, but also want to integrate or adapt other ontologies, or even apply multiple ontologies to solve a problem. In this context, it is necessary to find ways to integrate various ontologies and enable cooperation between them.

1.2 Method

This report is based on a literature study of research papers and books in different topics in the context of ontologies. The report brings up both background information as found in most textbook literature as well as recent findings and research.

1.3 Delimitations

As already mentioned, this report is delimited in selecting only parts of the general topics discussed for detailed excursion. The selection is based on what is deemed as relevant for the topic of the research. In particular, this report contributes to a Ph.D. project focusing on improvement to ontology matching strategies and evaluation. Ontology matching involves a large number of fields, e.g., machine learning, database schema, linguistics, etc. In this report, we do not evaluate the ontology

matching frameworks, but instead compare them and treat them as ontology matching. The available measures for evaluating ontology matching are discussed.

1.4 Disposition

This report is organized as follows. In Chapter 2, we discuss the scope of ontology matching, what ontology matching is, its input and output, usage categories of ontology matching and applications . In Chapter 3, the strategies used in ontology matching are presented. Chapter 4 describes the current existing systems and their comparison. Chapter 5 discusses the evaluation methods. Finally, we summarize and list interesting research questions in the field.

2 Range of Ontology Matching

In this chapter, first we will describe our scope of ontology matching. Secondly, we define what ontology matching is. Then we will look into different ways of using ontology matching and applications.

2.1 Ontology Matching Scope

Ontologies are considered as an important contribution used for solving the data heterogeneity problem. However, ontologies themselves can also be heterogeneous [20], e.g., the ontology can be expressed in different languages, e.g., OWL, RDFS, OKBC, KIF, etc. Different languages use different syntax, different logical representation, different semantics of primitives, and language expressivity. Even using the same ontology language does not solve heterogeneity problems. An ontology on motor-vehicles, for example, may include the concept "motor-bike", whereas another ontology on the same subject may ignore it. Klein [32] categorized possible mismatches of the ontologies heterogeneity by two levels:

- Language level
  - **Syntax.** Different languages have different syntax. For example, in RDFS the definition of class *Chair* is $< rdfs : ClassID = "Chair" >$. In LOOM, $(defconcept\ Chair)$ is expressing the same class.
  - **Logical representation.** Differences in logical representation occur when syntactically different, but the statement are logically equivalent. For example, the way present disjointness in OWL Lite is $Class(owl : Nothing\ complete\ A\ B)$, but also valid in OWL DL as $DisjiontClasses(A\ B)$.
  - **Semantics of primitives.** The same syntactically construct can have different meanings (semantics) in different language. For example, there are several interpretations of $A\ equalTo\ B$.
  - **Language expressivity.** Some language can express things that the other language can not. For example, negation can be expressed in one language but not in another.
- Ontology level
  - **Conceptualization.** Using the same linguistic terms describe different concepts (e.g., the concept "employee" can have different meaning in the ontologies). Using different modeling conventions and levels of granularity describe concepts (e.g., one ontology model "car" but not "truck", the other ontology model "car" that includes "truck".).

- **Terminological.** Using different terms to describe the same concepts (e.g., one ontology uses concept ”car”, while the other ontology use ”automobile”), homonym term (e.g., ”conductor” has a different meaning in music than in electric engineering).
- **Style of modeling.** Using different modeling paradigms to present concepts (e.g., one model uses temporal presentations based on interval logic while another uses a representation based on points).
- **Encoding.** Values in the ontologies can be encoded in different formats (e.g., a date represented as dd/mm/yyyy or mm-dd-yy).

As the translation between ontology languages can be considered as an independent issue [19], it is recommendable to translate different ontologies into the same language before comparing them on ontology level. Currently, most ontology matching systems are focusing on the ontology level.

2.2 Terminology

The terms mapping, matching and alignment are frequently used in work about combining ontologies. Keet summarizes different definitions about these terms [31]. Based on recent studies about combining ontologies, the terms used in this report are defined as follows [31] [51]:

Ontology merging: combine two ontologies from the same subject area into a new ontology.

Ontology integration: combine two ontologies from different subject areas into a new ontology.

Ontology alignment: identify correspondences between the source ontologies.

Ontology mapping: find equal parts in different source ontologies.

Ontology matching: find similar parts in the source ontologies or finding translation rules between ontologies.

2.3 Usage Categories of Ontology Matching and Applications

Choi categories the usage of ontology matching into three categories [9]:

1. Matching between an integrated global ontology and local ontologies. In this category, the global ontology provides a shared vocabulary, the matching maps a concept found in one ontology into a view, or query over other ontologies.

The main advantage is that it is easier to define mapping and find mapping rules because of the shared vocabulary in the global ontology. However, in this matching, the global ontology is needed which is very difficult to maintain in a highly dynamic environment. The traditional applications (e.g., information integration or schema integration) require determining ontology matching before running the system. For information integration systems, ontology matching interprets the relationship between the mediated schema and local schemas (e.g., [38], [12]).

2. Matching between local ontologies. In this category, the matching is transforming the source ontology entities into the target ontology entities based on semantic relations. This approach is more relevant for highly dynamic environments. However, comparing global ontology and local ontology matching, it is not easier to find mapping because of the lack of common vocabularies. The dynamic applications (e.g., agents, peer-to-peer, web services) require running time ontology matching. For example, [6] allows agents in peer-to-peer networks to communicate to other agents based on dynamic ontology mapping. [25] and [52] are web services application examples, ontology matching is used for finding new services to complete a request.

3. Matching in ontology merging. In this category, matching is used to try to find similarities and conflicts entities between the ontologies to be merged. Managing and maintaining the different versions of ontologies can also be the applications of ontology matching. Some application examples in these categories are Prompt [48] suit for Protege editor and Chimera [8] tool for Ontolingua.

2.4 Ontology Matching Input and Output

Shvaiko and Euzenat define the matching process is as five parameters [57] (see in Figure 1).

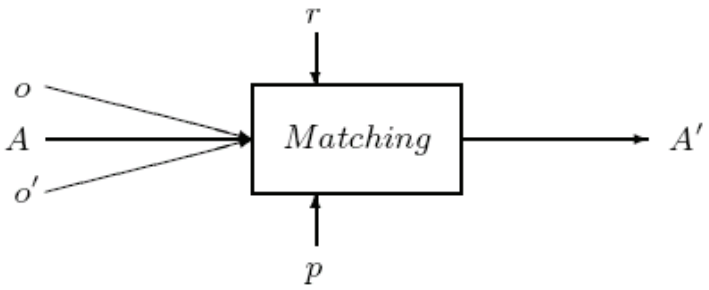

Figure 1: The matching process

The processing is generating a new alignment A' from the input consisting of

the ontology *o* and *o´*. However, other parameters can be involved in the matching process, like the use of input alignment *A*, which is to be completed by the processing, the matching parameters *p* (e.g., weight, thresholds), and the resources (e.g., lexicons).

Currently, the matching systems consider the matching of ontologies expressed in the same languages. Different elements of ontologies as input will be analysed in the different approaches, for example, GLUE uses taxonomies and instances [13], OLA uses many elements (e.g., classes, properties, constraints, taxonomy, instances) [22], ASCO uses as much available information in the ontologies as possible [63] (e.g., concepts, relations, structure, even apply TF-IDF to calculate similarity value between descriptions of the concepts or relations). Some systems are schema-based which can be viewed as a special ontology restrained relationship. In these cases, the input is a data model. For example, MOMIS (Mediator Environment for Multiple Information Sources) [5] uses local schemas.

Shvaiko and Euzenat separate different output types of ontology matching [57]:

- One-to-one or one-to-many correspondence between ontology entities.
- Expression of correspondence between ontology entities can be in the range 0 to 1.
- The relationship between entities in most systems is expressed as equivalence (=). Some systems (e.g., [26]) can provide more expressive result, like equivalence $(=)$, more general $(\supseteq)$, less general $(\subseteq)$, disjointness $(\perp)$), while the special *idk* (I do not know) expresses none of the relations.

3 Ontology Matching Strategies

In the following we consider an ontology to be a 4-tuple $< C, R, I, A >$, where C is a set of concepts (also called class, entity type, concept), R is a set of relations (also called property, attribute, feature, slot), I is a set of instances (also called individual) and A is a set of axioms. Ontology entities mean 4-tuple (e.g., concept, relation, instance, axiom) in the ontology. To find the correspondence between ontology entities, the similarity between entities need to be calculated. In this chapter, we first describe different strategies (e.g., string similarity, synonyms, structure similarity and based on instances) for getting similarity between entities used in current ontology matching systems. In addition, we introduce some methods to integrate similarities in section 3.5.

3.1 String Similarity

String similarity calculates the string distance metrics to determine the matching of entity names. String Distance is a non-negative integer that measures the distance between two strings. Before actually comparing string distance, some linguistic technologies (e.g., removing stop-word, stemming) are performed. Linguistic technologies transform each term to a standard form that can be easily recognised. Stop-word [4] means that some appear frequently words in the text with lack of indexing consequence. Indexing is the process of associating one or more keywords with each document in the information retrieval. For example, words like *the, this* and *of* in English, they appear often in the sentence but less value of indexing. Stemming is trying to remove certain surface marking words to root form. For example, words like *fishes* original form is *fish*.

Cohen et al. provide a good survey of the different methods to calculate string distance from edit-distance like functions (e.g., Levenstein distance, Monger-Elkan distance, Jaro-Winkler distance) to token-based distance functions (e.g., Jaccard similarity, TF-IDF or cosine similarity) and hybird methods (e.g., Level2JaroWinkle, SlimTFIDF, JaroWinklerTFIDF) [10]. These methods follow a big range of approaches and have been designed according to different criteria and perspectives, such as statistics, artificial intelligence, information retrieval, and databases. Cohen et al. introduce the most used methods in [10].

3.1.1 Edit-distance

In order to transform two compared strings to the same, a sequence of edit operations (e.g., character insertion, deletion and substitution) are performed. Edit-distance is the minimum cost of edit operations to convert string *s* to *t.* Each operation will be assigned a cost. One example is Levenstein distance which assigns all edit opera-

tions to 1. Monger-Elkan distance assigns lower cost to insertions or deletions than Levenstein distance, scaling to interval $[0, 1]$.

Jaro metric has similar metric but it is not based on edit distance. Jaro metric is based on the number and order of the common characters between two strings.

Given strings: $s = a_1, \dots a_K$, and $t = b_1, \dots b_L$, a character a_i in s is said to be common with t if there is a $b_j = a_i$ in t such that $i - H \leq j \leq i + H$, where $H = \frac{min(|s|,|t|)}{2}$. Let $s^{'} = a_1^{'} \dots a_{K'}^{'}$, be the characters in s which are common with t (in the same order they appear in s) and let $t^{'} = b_1^{'} \dots b_{L'}^{'}$, be analogous. A transposition for $s^{'}, t^{'}$ is a position i that $a_i^{'} \neq b_i^{'}$. Let $T_{s',t'}$ be half the number of transpositions for s' and t'. The Jaro similarity metric for s and t is

$$Jaro(s,t) = \frac{1}{3} \cdot \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s' - T_{s',t'}|}{|s'|} \right) \tag{1}$$

Jaro-Winkler is a variant of Jaro. The length P is the longest common prefix of s and t. Letting $P' = max(P, 4)$,

$$Jaro - Winkler(s,t) = Jaro(s,t) + \frac{P'}{10} \cdot (1 - Jaro(s,t)) \tag{2}$$

The Jaro and Jaro-Winkler metrics are suitable for short strings (e.g., personal first or last names).

3.1.2 Token-based Distance

The strings are compared as multisets of tokens. Jaccard distance computes the similarity between the sets of words S and T as

$$Jaccard = \frac{|S \bigcap T|}{|S \bigcup T|} \tag{3}$$

TF-IDF (Term Frequency - Inverse Document Frequency) [54] is widely used in information community and it is defined as

$$TF - IDF(S,T) = \sum_{\omega \in S \cap T} V(\omega, S) \cdot V(\omega, T) \tag{4}$$

Where $TF_{\omega,S}$ is the frequency of word ω in S, N is the size of the corpus and IDF_ω is the inverse of the fraction of names in the corpus that contains ω,

$$V'(\omega, S) = log(TF_{\omega,S} + 1) \cdot log(IDF_\omega) \tag{5}$$

$$V(\omega, S) = \frac{V'(\omega, S)}{\sqrt{\sum_{\omega'} V'(\omega, S)^2}} \tag{6}$$

3.1.3 Hybrid Distance

Level2JaroWinkler, Level2Levenstein, SlimTFIDF, and JaroWinklerTFIDF are hybrid-distance functions. First s and t are broken into substrings $s = a_1, \ldots a_K$, and $t = b_1, \ldots b_L$. Second a secondary distance function (e.g., Monge-Elkan, Jaro, Jaro-Winkler) operates over the $s = a_1, \ldots a_K$, and $t = b_1, \ldots b_L$. These are called level two distance functions. If we replace the exact token matches and use TF-IDF with approximate tokens based on the level two distance functions, it is called Soft-TFIDF (e.g., SlimTFIDF, JaroWinklerTFIDF).

3.1.4 Tool for String Similarity

A comparison of string distances methods available in four tools (SecondString [56], Simetrics [58], the Alignment API [64] and SimPack [59]) can found in [21] and is summerized in table 1:

Table 1: String distances methods available in four tools (Source: [21], Chapter 4)

Simetrics	SecondString	AlignAPI	SimPack
	n-grams	n-grams	
Levenshtein	Levenshtein	Levenshtein	Levenshtein
Jaro	Jaro	Jaro	
Jaro-Winkler	Jaro-Winkler	Jaro-Winkler	
Neddleman-Wunch	Neddleman-Wunch	Neddleman-Wunch	
		Smoa	
Smith-Waterman			
Monge-Elkan	Monge-Elkan		
Gotoh			
Matching coefficient			
Jaccard	Jaccard		Jaccard
Dice coefficient	TF-IDF		Dice coefficient TF-IDF
Cityblocks			Cityblocks
Euclidean			Euclidean
Cosine			Cosine
Overlap			Overlap
Soundex			Soundex

3.2 Synonyms

Synonyms (with the help of dictionary or thesaurus) can help to solve the problem of using different terms in the ontologies for the same concept. For example, an ontology might use “diagram”, the other could use “graph”referring to the same concern.

The WordNet [70] thesauri can support improving the similarity measure. WordNet is a large lexical database of English. It is based on psycholinguistic theories to define word meaning and model not only word meaning associations but also meaning-meaning associations [23]. WordNet tries to focus on the word meanings instead of word forms, though inflection morphology is also considered. WordNet consists of three databases, one for nouns, one for verbs and a third for adjectives and adverbs. WordNet consists of a set of synonyms “synsets”. Synsets provide different inter relationships such as synonymy and antonymy, hypernymy and hyponymy (superconcept and subconcept), meronymy and holonymy (Part-Of and Has-a). Figure 2 [27] shows part of noun hierarchy about term concerning a person, his (her) components, his (her) substances, and his (her) family organization. Since it is a very reduced view of the noun hierarchy, we can only see some relations such as meronymy, antonymy and hyponymy.

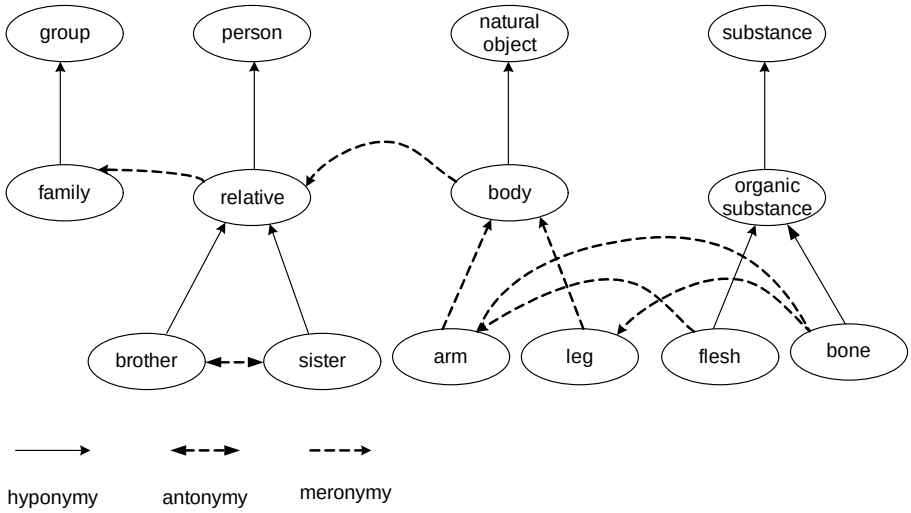

Figure 2: A partial view of the category of nouns of WordNet. (Source: [27], Chapter 2)

Su summarizes synsets’ relations associated with nouns, verbs, adjectives and adverbs [61]. Table 2, 4 and 3 show synsets’ relation with a brief definition and an example.

Rodriguez presents an approach to determine similar entities based on WordNet. For example, it considers hypernym/hyponym, holonym/meronyms relations [53]. The similarity measure based on the normaliztion of Tversky’s model and set theory

Table 2: Noun relations in WordNet. (Source: [61], Chapter 3)

Relation	Definition	Example
Hypernym	synset which is the more general class of another synset	$breakfast \rightarrow meal$
Hyponym	synset which is a particular kind of another synset	$meal \rightarrow lunch$
Holonym	synsets which is the whole of which another synset is part	$flower \rightarrow plant$
Meronyms	synsets which the parts of another synset	$bumper \rightarrow car$
Antonyms	synsets which are opposite in meaning	$man \leftrightarrow woman$

Table 3: Verb relations in WordNet. (Source: [61], Chapter 3)

Relation	Definition	Example
A-value-of	adjective synset which represents a value for a nominal target synset	$slow \rightarrow speed$
Antonyms	synsets which are opposite in meaning	$quickly \leftrightarrow slowly$

functions of intersection $|A \cap b|$ and difference $|A/B|$ as follows.

$$S(a,b) = \frac{|A \cap b|}{|A \cap b| + \alpha(a,b)\,|A/B| + (1-\alpha(a,b))\,|B/A|} \tag{7}$$

Where a and b are entity classes, A and B are the description sets of a and b (i.e., synonym sets, is-a or part-whole relations), α is a function that defines the relative importance of the non-common characteristics. For is-a hierarchy, α is expressed in term of the depth of the entity classes.

3.3 Structure Similarity

Structure similarity is usually based on different intuition of ontologies structures, such as is-a hierarchy, sibling concepts, relation and graph nodes.

3.3.1 Is-a Hierarchy

It is based on *is-a (taxonomy)* hierarchy of the ontology. The hypothesis is that if the direct super-concepts and/or the direct sub-concepts of two concepts are similar,

Table 4: Adjective and adverb relations in WordNet. (Source: [61], Chapter 3)

Relation	Definition	Example
Hypernym	synset which is the more general class of another synset	$fly \rightarrow travel$
Troponym	synset which is one particular way to perform another synset	$walk \rightarrow stroll$
Entails	synset which is entailed by another synset	$snore \rightarrow sleep$
Antonyms	synsets which are opposite in meaning	$increase \leftrightarrow decrease$

the two compared concepts may be also similar, for example [1], [11] and [39]. Here comes concepts similarity definition in [1]:

Definition *For $i \in \{1, 2\}$, let c_i be a concept in the hierarchy of concept Hc_i. Let $Pred(c_i)$, $Succ(c_i)$ be respectively the set of direct super-concepts of c_i in Hc_i, the set of direct sub-concepts of c_i in Hc_i. $SameSet(S_1, S_2)$ is the set of elements in S_1 which are similar with any element is S_2. $UnionSet(S_1, S_2)$ is the set of all of elements in S_1 combining with elements of S_2 that are not similar with any element is S_2. P_{Pred}, P_{Succ} is the proportions of the concepts in the sets $Pred$, $Succ$, respectively, then*

$$P_{Pred}(c_i, c_j) = \frac{|SamePred(c_i, c_j)|}{|UnionPred(c_i, c_j)|} \tag{8}$$

$$P_{Succ}(c_i, c_j) = \frac{|SameSucc(c_i, c_j)|}{|UnionSucc(c_i, c_j)|} \tag{9}$$

The direct *is-a (taxonomy)* hierarchy can be extended to the path of an ontology [1]. Now the intuition is that if the path from the root concepts to the concept A in the first ontology contains similar concepts with the path from the root concept to the concept A in the second ontology, concepts A and B may be similar.

Definition *Let $Path(c_i)$ be the path from the root to the class c_i in the hierarchy of concept Hc_i. $Path(c_i)$ is a set of classes along the path. The similar proportion between two paths of c_i and c_j is*

$$P_{Path}(c_i, c_j) = \frac{|SamePath(c_i, c_j)|}{|UnionPath(c_i, c_j)|} \tag{10}$$

Maedche and Staab [39] try to compare two taxonomies. The taxonomic overlap is determined by concepts semantic cotopy (i.e., all its super and sub concepts).

3.3.2 Sibling Concepts

The hypothesis is that if slibling concepts of two concepts are similar, the two compared concepts may be also similar [1].

Definition *For $i \in \{1, 2\}$, let c_i be a concept in the hierarchy of concept Hc_i. Let $Sibl(c_i)$, be the set of sibling concepts of c_i in Hc_i. $SameSet(S_1, S_2$ is the set of elements in S_1 which are similar with any element is S_2. $UnionSet(S_1, S_2$ is the set of all of elements in S_1 combining with elements of S_2 that are not similar with any element is S_2. P_{Pred} is the proportions of the concepts in the sets $Pred$, then*

$$P_{Sibl}(c_i, c_j) = \frac{|SameSibl(c_i, c_j)|}{|UnionSibl(c_i, c_j)|} \tag{11}$$

3.3.3 Relation

The hypothesis is that if the relations and related classes are similar, the two compared concepts may be also similar. Maedche and Staab propose computation entities similarity based on their relations' overlap (how similar their domain and range concepts are) [39]. For example, in ontology 1, relation *"located at"* is specifying the domain and range corresponding to *("hotel", "area").* In ontology 2, relation *"located at"* is specifying the domain and range corresponding to *("hotel", "city").* If relation *"located at"* and *"hotel"* are considered similar, it infers that *"area"* and *"city"* is similar. This approach can be extended to a set of classes and a set of relations. If a set of relations in the first ontology which is similar with the other set of relations in the second ontology, it is possible that two classes (domains or range of relations in the two sets) are similar. The relation overlap is determined by concepts upwards cotopy (i.e., all its super concepts).

3.3.4 Graph Nodes

The intuition is that two nodes are similar if their neighbors are also similar. The similarity flooding [42] matching algorithm uses graphs to find corresponding nodes in the graphs based on a fix-point computation. The algorithm takes two graphs (schemas, catalogs, or other data structures are converted into labeled graphs) as input. These graphs are used in an iterative fix-point computation to find out the mapping between corresponding nodes of the graphs, it relies on labels of arcs. Figure 3 illustrates the similarity flooding algorithm as follows:

1. Construct the pairwise connectivity graph (the dashed frame in Figure 3). Pairwise connectivity graph (PCG) definition is: $((x, y), p, (, y^{'})) \in PCG(A, B) \Longleftrightarrow (x, p, x^{'}) \in A$ and $(y, p, y^{'}) \in B$.

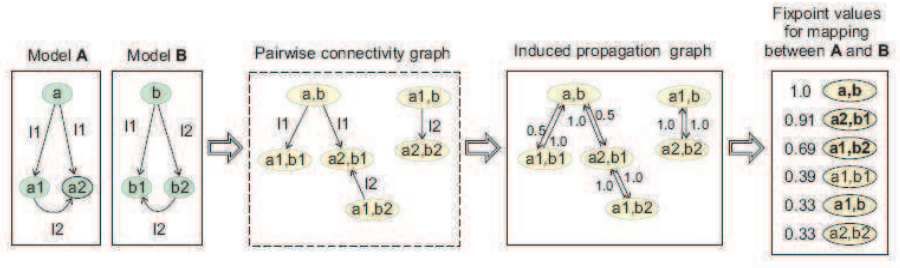

Figure 3: Example illustrating the Similarity Flooding Algorithm (Source: [42])

2. Assign propagation coefficients in PCG, see the induced propagation graph frame in Figure 3. The coefficient is $1/n$ and n is the out of edge in PCG.

3. Assign σ^0 to each node and calculate σ value using the formula below:

$$\begin{aligned}\sigma^{i+1}(x,y) =& \sigma^i(x,y) + \\ & \sum_{(a_u,p,x)\in A,(b_u,p,y)\in B} \sigma^i(a_u,b_u)\cdot\omega((a_u,b_u),(x,y)) + \\ & \sum_{(x,p,a_u)\in A,(y,p,b_u)\in B} \sigma^i(a_u,b_u)\cdot\omega((a_u,b_u),(x,y))\end{aligned} \tag{12}$$

4. Iteratively calculate σ until the change between σ^n and σ^{n-1} is less than threshold ϵ. For example, after iterations we can get fix-point values as shown in Figure 3.

The other fix-point algorithm example is OLA (OWL Lite Aligner) [22]. It considers the entities and structure of ontology (i.e., class (C), property (P), property instance (A), property restriction labels (L), taxonomy (e.g., subclass (S))). The distances of the input structures are expressed in a set of equations:

$$\begin{aligned}Sim_c(c,c') =& \pi_L^C sim_L(\lambda(c),\lambda(c')) \\ & + \pi_I^C sim_o(I(c),I'(c')) \\ & + \pi_S^C sim_C(\lambda(c),S'(c')) \\ & + \pi_A^C sim_P(A(c),A'(c'))\end{aligned} \tag{13}$$

To find the minimum distance between the concepts in the ontologies, it iterates the fix-point algorithm until the results are closer.

Noy and Musen use fix-point to combine matchers in approach PROMPTDIFF [46]. Based on results of set of heuristic matchers, the fix-point invokes the matchers repeatedly, feeding the results of one matcher into the others, until they produce no more changes in the diff.

3.4 Based on Instances

When two ontologies have the same or similar instances (also called individuals), we can find corresponding concepts. It can be performed by checking similarities between instances. If the similarity level of two instances reaches the threshold, then the two instances can be regarded as matched. For example, the *person number* will not change even if people maybe play different roles in different ontologies. The matching can be performed based on instances comparisons.

To identify the similarity level of two instances, string similarity methods (see 3.1) and cosine similarity based on TF-IDF [54] are common measurements. Formal concept analysis (FCA) is the other technique to identify concepts and instances. For example, FCA-Merge [60] is a method for comparing ontologies that have a set of shared instances or a shared set of documents annotated with concepts from source ontologies. Based on this information, FCA-Merge uses mathematical techniques from FCA to produce a lattice of concepts which relates concepts from the source ontologies.

Machine learning (e.g., [13], [68]) or clustering-based method [41] approaches integrate these different similarity measurements for instance matching.

Machine learning can be based on (i) shared attributes of objects/records, (ii) profile-based object matching and can correlate disjoint attributes to improve matching accuracy [13]. Wang et al. propose machine learning based on ontology hierarchy and object properties [68]. For example, if two instances from different ontologies are identified as instances of concept *Student* and *GraduateStudent* respectively, then they are more likely to be the same than two instances with one identified as *Student* and another as *Professor* based on string similarity methods or TF-IDF. If we take a look ontology hierarchy, *Student* and *Professor* (both are sub concept of *Person*) are defined as disjoint concepts, a student instance could never be matched with a professor instance, even if they have very high string-based similarity. Object properties allow users to connect relations between instances. For example, a property *writtenBy* is used by a publisher to relate publications to their author instances, while inverse property *write* is used by a professor to link his own instance with his publications.

Machine learning can be separated in two phases [21]. The first phase is the learning or training phase. In this phase, the training data is created, for example, manually matching two ontologies and the system learns a matcher from this data. Learning phase can be processed online, so the system can continue learning, or offline in the case of that speed is not relevant but its accuracy is. To be adaptive for dynamic situations, a stochastic model and SVM classifier can be applied (e.g., [68], [7]). In The second phase, the learnt matcher is used for matching new ontologies.

There are several well-known machine learning methods used in ontology matching illustrated in [21] such as:

- **Naive Bayesian learning**. "A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions" [35]. The Bayes decision rule is the rule that selects the category with minimum conditional risk [65]. In the case of minimum-error-rate classification, the rule will select the category with the maximum posterior probability. Suppose there are k classes, $c_1, c_1, \ldots, c_k$. Given a feature vector x, the minimum-error-rate rule will assign it to class c_j if

$$Prob(c_j|x) > Prob(c_i|x) \qquad \text{for all } i \neq j \tag{14}$$

  Here, the posterior probability is used as the discriminant function. An alternative criterion for minimum-error-rate classification is to choose class c_j so that

$$Prob(x|c_j)Prob(c_j) > Prob(x|c_i)Prob(c_i) \qquad \text{for all } i \neq j \tag{15}$$

  which is derived from well-known Bayes theorem:

$$Prob(x|c) = \frac{Prob(x|c)Prob(c)}{Prob(x)} \tag{16}$$

  Examples use naive Bayesian classifier for ontology matching are [62], [13].

- **WHIRL (Word-based Heterogeneous Information Representation Language) learner.** Whirl is an extension of conventional relational databases to perform soft joins based on the similarity of textual identifiers. Doan et al. use the Whirl learner to classify an input instance based on the labels of its nearest neighbors in the training set [13]. It uses the TF-IDF similarity measure commonly employed in information retrieval.

- **Neural networks**. Artificial neural networks are made up of nodes and weighted connections between them. The commonest type of artificial neural network consists of three groups, or layers, of nodes: a layer of "input" nodes is connected to a layer of "hidden" nodes, which is connected to a layer of "output" nodes. Neural network has adapted to ontology matching, for example, it is used t discover correspondences among attributes via category and classification [36]. Authors use neural network to learn matching parameters such as matcher weights to generate features and similarities [15].

- **Decision trees**. Decision trees [66] is used as predictive model which maps observations about an item to conclusions about the item's target value. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. Decision trees are constructed in order to help with making decisions. For example, decision trees are used for discovering correspondences entities in [71] and for learning thresholds in [15].

- **Stacked generalization**. Stacked generalization is an approach to combine multiple learning algorithms [69]. This approach is adapted by GLUE [13] to aggregate different learners such as Naive Bayesian learner, Whirl learner.

- **Bayesian networks**. Bayesian networks is a probabilistic approach. In approach [50], the source and target ontologies are translated into Bayesian networks, the concept mapping between the two ontologies are treated as evidential reasoning between the two translated Bayesian networks. The other work [43] uses Bayesian networks to enhance existing matches between ontology concepts.

3.5 Similarity Aggregation

Once the similarity between ontology entities are available based on different strategies (e.g., string similarity, synonyms, structure similarity, based on instances), aggregating similarities algorithms are needed to combine matchers. Euzenat and Shvaiko summarize similarity aggregation measures as follows [21]:

Definition (Triangular norm). *A triangular norm T is a function from D* $\times$ *D* $\rightarrow$ *D (where D is a set ordered by* $\leq$ *and provided with an upper bound* $\top$*) satisfying the following conditions:*

$T(x, \top) = x$ (boundary condition)

$x \leq y \Rightarrow T(x, z) \leq T(y, z)$ (monotonicity)

$T(x, y) = T(y, x)$ (commutativity)

$T(x, T(y, z)) = T(T(x, y), z)$ (associativity)

Triangular norms are suitable to combine the highest score from all aggregated values. Triangular norms tend to express the dependencies between the values of the different dimensions.

Definition (Weighted product). *Let o be a set of objects which can be analysed in n dimensions. The weighted product between two such objects is as follows:*

$\forall x, x' \in o, \delta(x, x') = \prod_{i=1}^{n} \delta(x_i, x'_i)^{w_i}$

such that $\delta(x_i, x'_i)^{w_i}$ *is the dissimilarity of the pair of objects along the* i^{th} *dimension and* w_i *is the weight of dimension i.*

Weighted product is another triangular norm. It has the drawback that if one of the dimensions has a measure of 0, then the result is also 0.

Definition (Minkowski distance). *Let o be a set of objects which can be analysed in n dimensions, the Minkowski distance between two such objects is as follows:*

$\forall x, x' \in o, \delta(x, x') = \sqrt[p]{\sum_{i=1}^{n} \delta(x_i, x'_i)^p}$

where $\delta(x_i, x_i')$ is the dissimilarity of the pair of objects along the i^{th} dimension.

Minkowski distance is suitable to independent dimensions and try to balance the values between dimensions.

Definition (Weighted sum). *Let o be a set of objects which can be analysed in n dimensions, the weighted sum between two such objects is as follows:*

$\forall x, x' \in o, \delta(x, x') = \sum_{i=1}^{n} w_i \times \delta(x_i, x_i^{'})$

where $\delta(x_i, x_i')$ is the dissimilarity of the pair of objects along the i^{th} dimension and w_i is the weight of dimension i.

Weighted sum considers that the different important values can be aggregated.

Definition (Fuzzy aggregation operator). *A fuzzy aggregation operator f is a function from $D^n \rightarrow D$ (with D being a set ordered by $\leq$ and provided with an upper bound $\top$) satisfying $\forall x, x_1, \ldots, x_n, y, y_1, \ldots, y_n, \in D$ the following conditions:*

$f(x, \ldots x) = x$ *(idempotency)*

$\forall x_i, y_i, x_i \leq y_i \Rightarrow f(x_1 \ldots x_n) \leq f(y_1 \ldots y_n)$ *(increasing monotonicity)*

f is a continuous function (continuity)

Fuzzy aggregation is used for aggregating the results of competing algorithms.

Definition (Weighted average). *Let o be a set of objects which can be analysed in n dimensions. The weighted average between two such objects is as follows:*

$\forall x, x' \in o, \delta(x, x') = \frac{\sum_{i=1}^{n} w_i \times \delta(x_i, x_i^{'})}{\sum_{i=1}^{n} w_i}$

such that $\delta(x_i, x_i')$ is the dissimilarity of the pair of objects along the i^{th} dimension and w_i is the weight of dimension i.

Weighted average is often used as a fuzzy aggregate.

Definition (Ordered Weighted average). *An ordered weighted average operator f is a function from $D^n \rightarrow D$ (with D being a set ordered by $\leq$ and provided with an upper bound $\top$) satisfying $\forall x, x_1, \ldots, x_n \in D$, such that:*

$f(x, \ldots x) = \sum_{i=1}^{n} w_i \times x_i^{'}$

where

$w_1, \ldots w_n$ *is a set of weights in* $[01]$ *such that* $\sum_{i=1}^{n} w_i = 1$*;*

$x_i^{'}$ *is the i-th largest element of* $(x_i, \ldots x_n)$*.*

Ordered weighted average allows to give more importance to the highest (or lowest) value.

4 Ontology Matching Systems Overview

This chapter will illustrate several ontology matching systems or tools. There are some surveys or comparisons about ontology matching systems in [30], [44], [19], [37], [9], [21]. We will compare the systems or tools based on the ontology strategies discussed in chapter 3 and their input and output. Finally, a table is presented for a summary of ontology matching systems.

4.1 PROMPT (Stanford Medical Informatics)

PROMPT [48] is a semi-automatic tool and a plug-in for the open-source ontology editor PROTEGE. It determines string similarity and analyzes the structure of an ontology. It provides guidance for the user for merging ontologies. It suggests the possible mapping and determines the conflicts in the ontology and proposes solutions for theses conflicts. PROMPT consists of several tools (see Figure 4). iPROMPT [45] is an interactive ontology merging tool. AnchorPROMT [46] uses graph-based mappings to provide additional information for iPROMPT. PROMPT-Diff [46] compares different ontology versions by combining matchers in a fixed point manner. PROMPTFactor is a tool for extrating a part of an ontology.

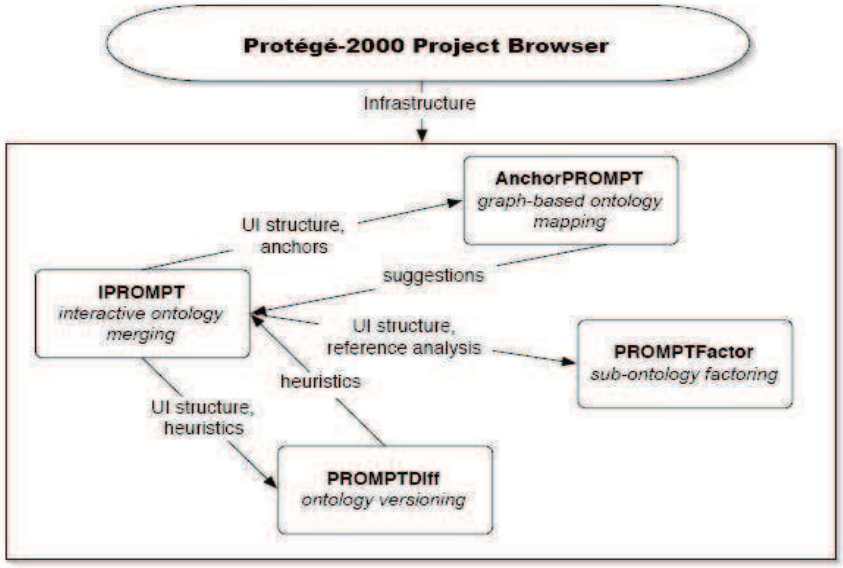

Figure 4: The PROMPT infrastructure. (Source: [48])

Input of PROMPT is two ontologies with languages OKBC or OWL. Output is the suggestion of mapping and a merging ontology (based on user choices). The main strategies of PROMPT are based on string similarity and structure. For example,

Anchor-Prompt is that if two pairs of terms from the source ontologies are similar and there are paths connecting the terms, then the elements in those paths are often similar as well.

4.2 SAMBO (Linköping Universitet)

SAMBO (System for Aligning and Merging Biomedical Ontologies) [33] is a system that assists the user in aligning and merging two biomedical ontologies. The user performs an alignment process with the help of alignment suggestions proposed by the system. The system carries out the actual merging and derives the logical consequences of the merge operations.

Input of SAMBO is two ontologies with languages DAML+OIL or OWL. Output is the suggestion of mapping and a merging ontology (based on user choices). The main strategies of SAMBO are including (combinations of) string similarity, synonyms (based on WordNet and domain knowledge UMLS (Unified Medical Language Systems) [67]) structure-based strategies and algorithms based on machine learning. Figure 5 shows the alignment strategy.

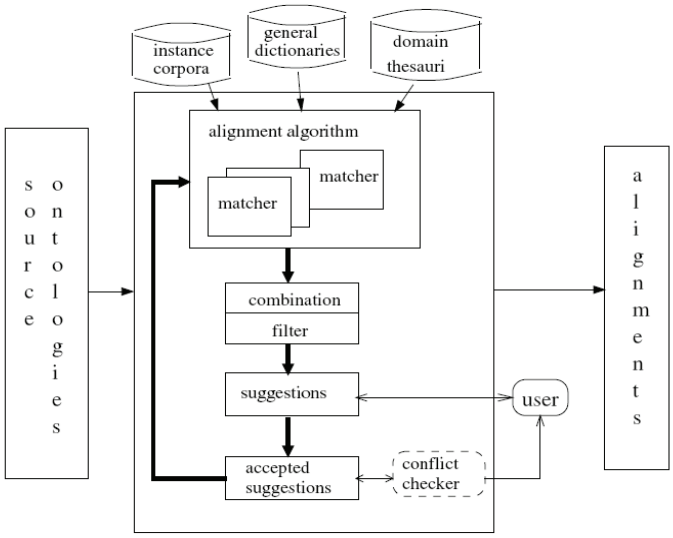

Figure 5: The SAMBO Alignment strategy. (Source: [33])

4.3 FCA-Merge (University of Karlsruhe)

FCA-Merge [60] is a method for merging ontologies based on mathematical techniques from FCA (Formal Concept Analysis, [24]). FCA-Merge is a bottom-up tech-

nique for merging ontologies based on a set of documents. It consists of three steps (see Figure 6, namely (i)instance extraction, (ii) concept lattice computation and (iii) the generation of the merged ontology based on the concept lattice).

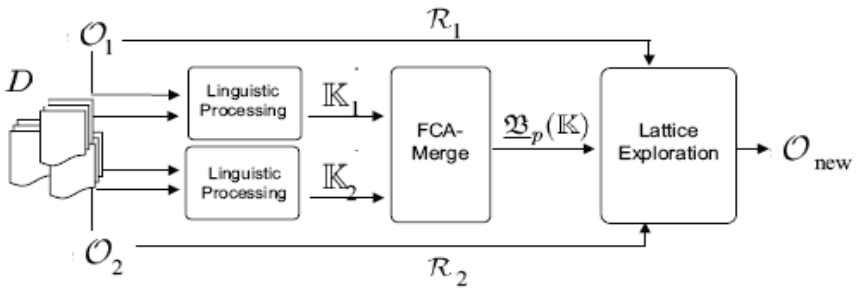

Figure 6: The FCA-Merge process. (Source: [60])

Input of FCA-Merge is two ontologies and a set of documents that are relevant to both ontologies. Output is a merged ontology. The strategies of FCA-Merge are based on string similarity, FCA and instances, structure.

4.4 GLUE (University of Washington)

GLUE [13] is a system that employs machine learning techniques to find mappings. Given two ontologies, for each concept in one ontology GLUE finds the most similar concept in the other ontology. Figure 7 shows GLUE's architecture. It consists of *Distribution Estimator, Similarity Estimator,* and *Relaxation Labeler.* The *Distribution Estimator* takes as input two taxonomies and instances. Then it applies multiple machine learners and exploits information in concept instances and taxonomic structure of ontologies. It uses a probabilistic model to combine results of different learners. Next, GLUE feeds the above results into the *Similarity Estimator*, which applies a user-supplied similarity function to compute a similarity value for each pair of concepts to generate similarity matrix. The *Relaxation Labeler* module then takes the similarity matrix, together with domain-specific constraints and heuristic knowledge, and finds mappings.

Input of GLUE is two ontologies, where ontology is seen as a taxonomy of concepts. Output is (1-1) correspondences between the taxonomies of two given ontologies: for each concept node in one taxonomy, find the most similar concept node in the other taxonomy. The strategies of GLUE are based on string similarity, taxonomic structure of ontologies and machine learning techniques (e.g., naive Bayesian classifier, Whirl and stacked generalization, see Sect. 3.4) to exploit instances.

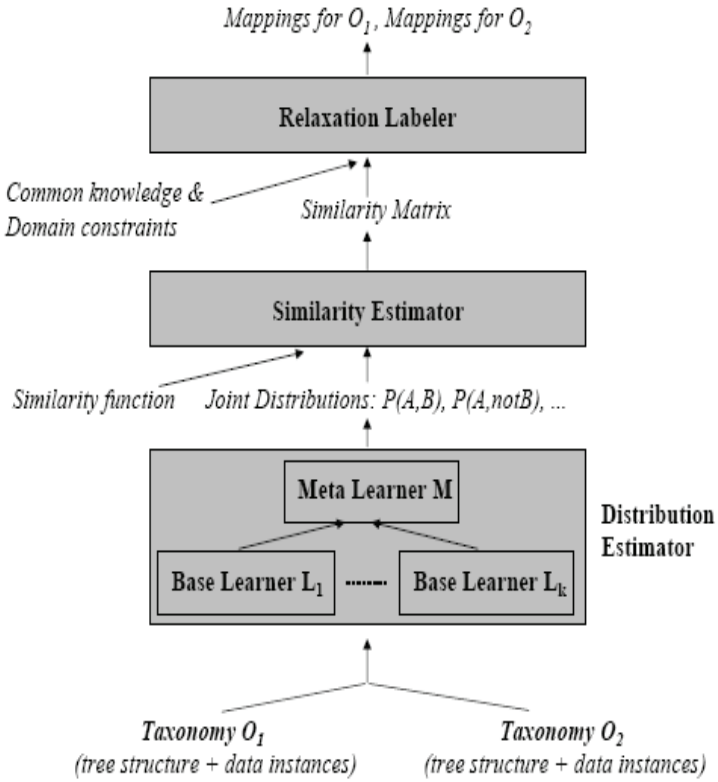

Figure 7: The GLUE Architecture. (Source: [13])

4.5 OLA (INRIA Rhone-Alpes and University of Montreal)

OLA (OWL Lite Aligner) [22] is a system that is designed with the idea of balancing the contribution of each elements of ontologies. It first transforms the input ontologies to graph structures and marks the relationships between entities. The similarity between nodes in the graph structures will depend on the category of nodes (e.g., class, property) considered and all the features of this category (e.g., superclasses, properties).

Input of OLA is two OWL ontologies. OLA uses many elements (e.g., classes, properties, constraints, taxonomy, instances) in the ontologies. Output is one-to-many correspondences. The strategies of OLA are based on string similarity, synonyms, structure and instances. The fix-point algorithm is used to aggregated the results see Figure 8 and Sect. 3.3.4.

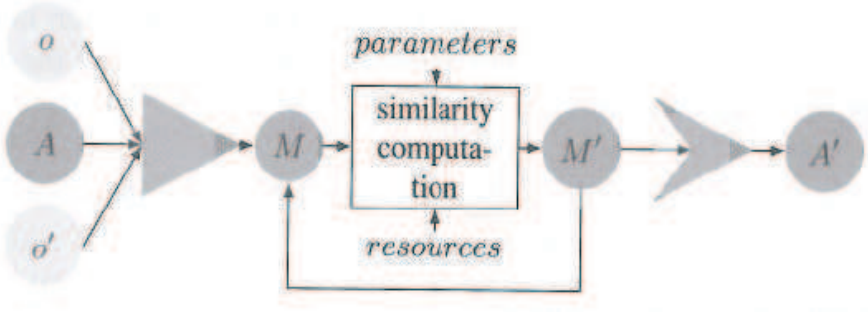

Figure 8: The OLA architecture. (Source: [21] p. 127)

4.6 ASCO (INRIA Sophia-Antipolis)

ASCO [1] [2] is a system that automatically discovers pairs of corresponding elements in two ontologies. ASCO has three phases. In the first phase, the system normalizes terms and expressions. Different string similarity methods (e.g., JaroWinkler metric, Monger-Elkan metric, see Sect. 3.1) are used to compare the terms. TF-IDF is used to calculate similarity value between descriptions of the concepts or relations and WordNet is integrated. The second phase is structure matching. It uses an iterative fixed point computation algorithm that propagates similarities to the neighbours (subclasses, superclasses and siblings). In the final phase, the results from the above two phases are aggregated through a weighted sum, and the final similarities correspondence are selected by a threshold.

Input of ASCO is two OWL or RDFS ontologies. ASCO uses many elements (e.g., concepts, relations, structure, even apply TF-IDF to calculate similarity value between descriptions of the concepts or relations) in the ontologies. Output is one-to-one or one-to-n correspondences. The strategies of ASCO are based on string similarity, structure and synonyms (based on WordNet and EuroWordNet [17]).

4.7 QOM (University of Karlsruhe)

QOM (Quick Ontology Mapping) [14] is an approach that improves the efficiency of NOM (Naive Ontology Mapping) [16]. The idea is that the loss of quality (compared to a standard baseline) is marginal, but the improvement in efficiency can significant that it allows for mapping large-size ontologies. The run-time complexity of QOM is $O(n \cdot log(n))$, while NOM is $O(n^2 \cdot log^2(n))$, AnchorPROMT [46] is $O(n^2 \cdot log^2(n))$ and GLUE [13] is $O(n^2)$. Figure 9 illustrates its six steps. To make an efficient mapping algorithm, several measures are used in the processing. For example, in the second step, it uses heuristics to lower the number of candidate mappings; in the third step, it avoids the complete pairwise comparison of trees in favor top-down strategy; in the fourth step, it applies sigmoid function which emphasizes high indi-

vidual similarities and de-emphasizes low individual similarities; in the fifth step, it uses a threshold to discard spurious evidence of similarity.

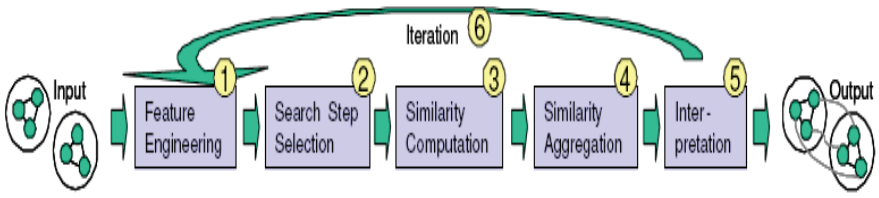

Figure 9: The QOM mapping process. (Source: [14])

Input of QOM is two OWL or RDFS ontologies. QOM uses many elements (e.g., classes, properties, instances) in the ontologies. Output is one-to-one or one-to-none correspondences. The strategies of QOM are based on string similarity, structure and instances.

4.8 S-Match (University of Trento)

S-Match [26] is an approach that takes two graph-like structures and finds a mapping between the nodes of graphs that correspond semantically to each other. S-Match determines determine semantic relations by analyzing the meaning which is codified in the elements and the structures of schemas. Labels at nodes are automatically translated into propositional formulas. Then the matching problem has been translate intopositional validity problem, which can then be efficiently resolved using (sound and complete) state of the art propositional satisfiability (SAT) deciders, e.g., [55].

Input of S-Match is two graph-like structures. Output is the semantic relations (e.g., equivalence $(=)$, more general $(\supseteq)$, less general $(\subseteq)$, disjointness $(\perp)$), while the special *idk* (I do not know) expresses none of the relations. The strategies of S-Match are based on string similarity, structure (SAT) and synonyms (based on WordNet).

4.9 IF-Map (University of Southampton and University of Edinburgh)

IF-Map (Information-Flow-based Map) [29] is a fully automatic approach to ontology matching based on Barwise-Seligman [3] theory of information flow. Given two local ontologies with instances, IF-Map generates a *logic infomorphism* - a mapping between local ontologies and reference ontology which without instances. Figure 10 shows the IF-Map architecture. It consists four steps: (i) acquisition ontology,

(ii) translate ontology into Prolog clauses, (iii) find *logic infomorphism* (iv) display results.

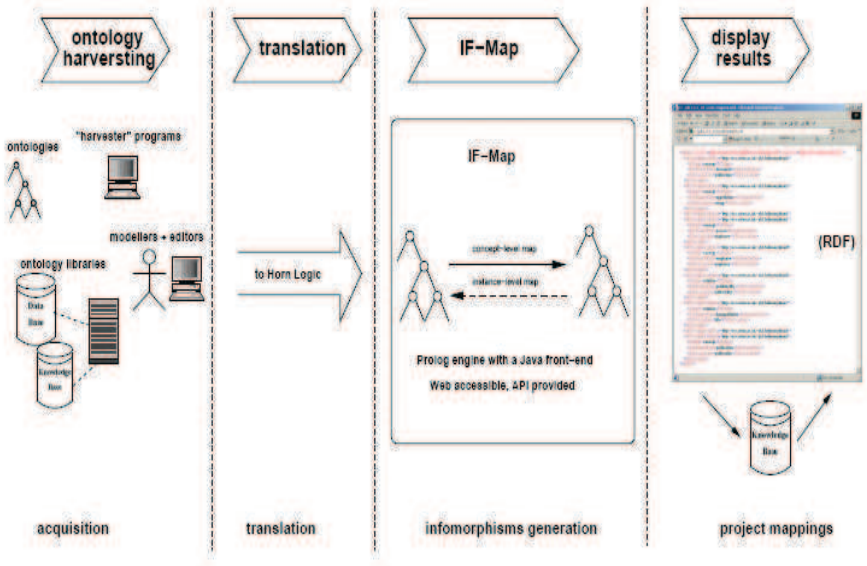

Figure 10: The IF-Map architecture. (Source: [29])

Input of IF-Map is two local KIF or RDF ontologies and one reference ontology. Output is concept-to-concept and relation-to-relation correspondence. The strategies of IF-Map are based on string similarity, structure (check *is-a* hierarchy in both directions), Barwise-Seligman theory and Horn logic.

4.10 Momis (University of Modena and Reggio Emilia)

Momis (Mediator Environment for Multiple Information Source) [5] is an approach that creates a global virtual view (GVV) of the local sources. Figure 11 shows the process for building the GVV for a set of Web pages in five steps:

1. Local source schemata extraction. Wrappers generate schemas for the local sources and translate them into the common language ODL_{I3} (extension of Object Definition language [49]).

2. Local source annotation with WordNet. The integration designer chooses a meaning for each element of a local source schema, according to the WordNet lexical ontology.

3. Common thesaurus generation. It describes relationships of inter-schema and intra-schema knowledge about classes and attributes of the source schemata.

4. GVV generation. It generates a global schema and sets of mappings with local schemata by using the common thesaurus and the local schema descriptions.

5. GVV annotation. It semi-automatically generates mapping between local schemas and global schema by exploiting the annotated local schemas.

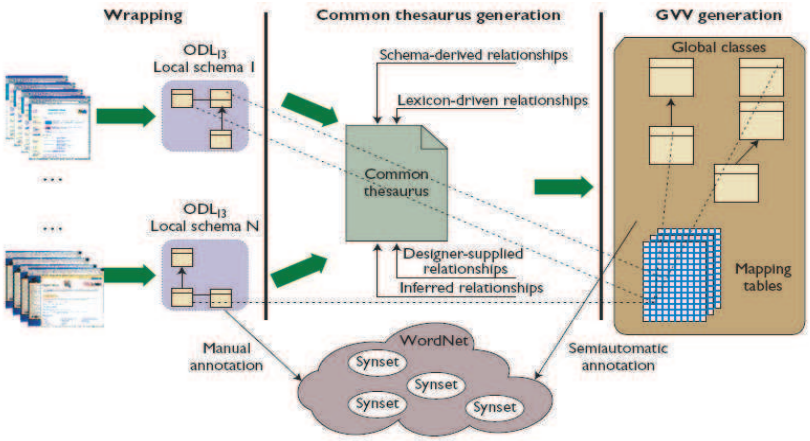

Figure 11: The Momis architecture. (Source: [5])

Input of Momis is information sources (e.g., database, Webpages and documents). Output is a global virtual view (merged ontology from the local data schemas). The strategies of Momis are based on string similarity, synonyms (based on WordNet) and structure.

4.11 Summary

See Table 5 for a summary of ontology matching systems.

	PROMPT	SAMBO	FCA-Merge	GLUE	OLA	ASCO	QOM	S-Match	IF-Map	Momis
Input	Two ontologies (OWL or OKBC)	Two ontologies (DAML + OIL or OWL)	Two ontologies and documents	Two ontologies (taxonomy)	Two ontologies (OWL)	Two ontologies (OWL or RDFS)	Two ontologies (OWL or RDFS)	Two graph-like structure	Two ontologies (KIM or RDFS) and one reference ontology	Information sources
Output	One merged ontology	One merged ontology	One merged ontology	1:1 correspondence	1:N correspondence	1:1 or 1:N correspondence	1:1 or 1:none correspondence	Semantic relations	1:1 correspondence	A global virtual view
String Similarity	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Synonyms	No	Yes	No	No	Yes	Yes	No	Yes	No	Yes
Structure Similarity	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Instances	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No
Aggregation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 5: A summary of ontology matching systems

5 Evaluation

Currently, there exist a number of ontology matching systems. However, these systems are developed for various purposes and using different strategies. Noy and Musen propose a framework for evaluating ontology-mapping tools based on a variety in underlying assumptions and requirements [47]. In order to compare different systems, surveys (e.g., [9], [37]) summarize and evaluate several tools. However, this evaluation focuses on functionality, user interaction and mapping strategies, but does not deal with matching quality.

To evaluate the increasing number of ontology matching methods and their qualities, OAEI (Ontology Alignment Evaluation Initiative) started arranging evaluation campaigns yearly from 2004. We will focus on OAEI evaluation here.

5.1 Evaluation Input

The input of evaluation are two ontologies written in the OWL-DL language. The different elements of ontologies, e.g., concepts, instance and relations can be aligned. The output is a *:* (* noted none) equivalence alignment of named entities. For example, one entity of one ontology can (e.g., injective, total, one-to-one) map to entity/entities of the other ontology. That means non constraint on the alignment.

Two benchmarks are proposed to be evaluated: a competence benchmark and a comparison benchmark [20]. Competence benchmarks aim to distinguish the performance of a special system regarding a set of well-defined tasks which are isolated special characteristics. Competence benchmarks help the system designers to evaluate their systems to localize with the stable system. Comparison benchmarks aim to compare the performance of different systems on a defined task or application. It aims to improving the whole field instead of individual systems.

5.2 How to Measure Evaluation Results

OAEI proposed different evaluation measures, from machine-focused (e.g., compliance and performance measures) to user-focused, from general to task-specific measures. However, user-focused measures need interaction of users which is not easy to get any objective evaluations. Task-specific measures need to set up different task compare profiles with respect for certain tasks. It is difficult to determine the evaluation value of the alignment process independently, so the current evaluations focus more on compliance and performance measures.

5.2.1 Compliance Measures

Compliance measures aim to evaluate the quality of the output provided by a system compared to a reference output. Even the reference output is not always available, not always useful and not always consensual. The compliance measures consist of *Precision, Recall, Fallout, F-measure, Overall, etc.*. Here are their definitions [20]:

Definition (Precision).*Given a reference alignment R, the precision of some alignment A is given by*

$$P(A,R) = \frac{(R \bigcap A)}{|A|}. \tag{17}$$

It measures a valid possibility for ex-post evaluations.

Definition (Recall). *Given a reference alignment R, the recall of some alignment A is given by*

$$R(A,R) = \frac{(R \bigcap A)}{|A|}. \tag{18}$$

Definition (Fallout). *Given a reference alignment R, the fallout of some alignment A is given by*

$$F(A,R) = \frac{|A| - |A \bigcap R|}{|A|} = \frac{|A \setminus R|}{|A|}. \tag{19}$$

It measures the percentage of retrieved pairs which are false positive.

Definition (F-measure). *Given a reference alignment R and a number between 0 and 1, the F-measure of some alignment A is given by*

$$M_\alpha(A,R) = \frac{P(A,R) \times R(A,R)}{(1-\alpha) \times P(A,R) + \alpha \times P(A,R)}. \tag{20}$$

It is used to aggregate the result of precision and recall.

Definition (Overall). *Given a reference alignment R, the overall of some alignment A is given by*

$$O(A,R) = R(A,R) \times \left(2 - \frac{1}{P(A,R)}\right). \tag{21}$$

It can also be defined as:

$$O(A,R) = \frac{(A \bigcup R) - (A \bigcap R)}{|R|}. \tag{22}$$

It measures the effort required to fix the given alignment.

5.2.2 Performance Measures

Performance measures compare important features of the algorithms (e.g., speed, memory and complexity). However, performance measures depend on the evaluation environment and ontology management system. It is difficult to get objective evaluations.

5.3 Which Frameworks Can Be Used in Evaluation

OAEI provides two frameworks that can be used in evaluation: alignment and evaluation frameworks. The participants can adapt their systems and implement Alignment API [18] to generate the results. The Alignment API supplies many services and new algorithms can be added through the interfaces. The evaluation framework compares the alignment results to generate evaluation results. The Alignment API provides methods to get evaluation measures such as precision, recall, overall, fall-out, f-measure directly.

5.4 KitAMO Evaluation Framework

Lambrix and Tan present KitAMO framework for evaluating ontology alignment strategies. KitAMO provides an integrated system for comparing evaluation of alignment strategies and their combinations [34]. The performance of the strategies or their combinations are compared. The output of KitAMO is the number of the correct, wrong and inferred suggestions in a table.

6 Conclusion and Research Questions

Ontology matching is a quite new area but developing faster. For example, before the year 2000, 16 publications devoted to matching at various conferences are collected on the Ontology Matching website [40]. There are 15 publications in 2001 and the number grows to 53 in 2005. In this report we try to present the state-of-the-art in ontology matching. We have discussed why we need ontology matching (Chapter 1), what ontology matching is and its applications (Chapter 2). Different strategies used in ontology matching are presented in Chapter 3. Some current ontology matching systems are compared in Chapter 4. The evaluation measures for ontology matching are discussed in Chapter 5.

Ontology matching involves a large number of fields, e.g., machine learning, database schema, linguistics, etc. Different strategies used in ontology matching (see Chapter 3) are based on these fields. However, the matching systems still need improvement. For example, how to improve the performance such as time consumption, how to improve accurate similarity.

In section 3.5, we discuss some aggregation algorithms to combine matching results. However, most algorithms are based on weight which is manually defined. One research question can be: how to combine matchers? For example, how to set weight for matcher in the different applications. Is there another way to combine matchers instead of based on weight?

Until now, there are a few ontology evaluation tools available. OAEI propose different measures to evaluate ontology matching (see Sect. 5). However, it does not support to evaluate the combination of matchers. How to evaluate the combination of matchers and give suggestion for the combination will be one interesting research question.

From the above discussion, we summarizes the following research questions:

1. How to combine different matchers? For example, how to set weight for matchers in different applications. Is there another way to combine matchers instead of based on weight?

2. How to evaluate the combination of matchers and give suggestion for the combination?

3. How to improve current ontology matching strategies (e.g., system performance, accurate similarity)?

References

[1] Than-Le Bach, Rose Dieng-Kuntz, and Fabien Gandon. On ontology matching problems (for building a corporate semantic web in a multi-communities organization). In *Proc. 6th International Conference on Enterprise Information Systems (ICEIS)*, pages 236–243, Porto (PT), 2004.

[2] Thanh-Le Bach and Rose Dieng-Kuntz. Measuring similarity of elements in OWL ontologies. In *Proc. AAAI Workshop on Contexts and Ontologies (C&O)*, pages 96–99, Pittsburgh (PA US), 2005.

[3] Jon Barwise and Jerry Seligman. *Information flow: the logic of distributed systems*, volume 44 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge (UK), 1997.

[4] Richard K. Belew. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press, 2001.

[5] Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, and Maurizio Vincini. Synthesizing an integrated ontology. *IEEE Internet Computing*, 07:42–51, 2003.

[6] Paolo Besana, Dave Robertson, and Michael Rovatsos. Exploiting interaction contexts in p2p ontology mapping. In *P2PKM*, 2005.

[7] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures, 2003.

[8] Chimera, http://www.ksl.stanford.edu/software/chimaera/.

[9] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41, September 2006.

[10] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks, 2003.

[11] R. Dieng and S. Hug. Comparison of personal ontologies represented through conceptual graphs. In *Proceedings of ECAI 1998*, pages 341–345, 1998.

[12] Anhai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Mach. Learn.*, 50(3):279–301, 2003.

[13] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.

[14] Marc Ehrig and Steffen Staab. Efficiency of ontology mapping approaches. In *International Workshop on Semantic Intelligent Middleware for the Web and the Grid at ECAI 04*, Valencia, Spain, AUG 2004.

[15] Marc Ehrig, Steffen Staab, and York Sure. Bootstrapping ontology alignment methods with APFEL. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 186–200, Galway (IE), 2005.

[16] Marc Ehrig and York Sure. Ontology mapping – an integrated approach. In *Proc. 1st European Semantic Web Symposium (ESWS)*, volume 3053 of *Lecture notes in computer science*, pages 76–91, Hersounisous (GR), May 2004.

[17] EuroWordNet. http://www.illc.uva.nl/eurowordnet/.

[18] Jérôme Euzenat. An api for ontology alignment. 2004.

[19] Jérôme Euzenat, Thanh Le Bach, Jesus Barrasa, Paolo Bouquet, Jan De Bo, Rose Dieng, Marc Ehrig, Manfred Hauswirth, Mustafa Jarrar, Ruben Lara, Diana Maynard, Amedeo Napoli, Giorgos Stamou, Heiner Stuckenschmidt, Pavel Shvaiko, Sergio Tessaris, Sven Van Acker, and Ilya Zaihrayeu. State of the art on ontology alignment. Technical report, NoE Knowledge Web project delivable, 2004.

[20] Jérôme Euzenat, Raul Garcia Castro, and Marc Ehrig. D2.2.2: Specification of a benchmarking methodology for alignment techniques. Technical report, NoE Knowledge Web project delivable, 2004.

[21] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[22] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in owl-lite. In *15th ECAI*, Valencia (ES), 2004.

[23] R. Ferrer-i-Cancho. The structure of syntactic dependency networks: insights from recent advances in network theory. In Levickij V. and Altmman G., editors, *Problems of quantitative linguistics*, pages 60–75. 2005.

[24] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997. Translator-C. Franzke.

[25] Fausto Giunchiglia, Fiona McNeill, and Mikalai Yatskevich. Web service composition via semantic matching of interaction specifications. Technical Report DIT-06-080, University of Trento, 2006.

[26] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX, 2007.

[27] Asuncion Gomez-Perez, Mariano Fernandez-Lopez, and Oscar Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web, 1st Edition*. Springer-Verlag, Heidelberg, 2004.

[28] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[29] Yannis Kalfoglou and Marco Schorlemmer. IF-Map: an ontology mapping method based on information flow theory. *Journal on Data Semantics*, I:98–127, 2003.

[30] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[31] C. Maria Keet. Aspects of ontology integration, 2004.

[32] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions, 2001.

[33] Patrick Lambrix and He Tan. Sambo-a system for aligning and merging biomedical ontologies. *Web Semant.*, 4(3):196–206, 2006.

[34] Patrick Lambrix and He Tan. A tool for evaluating ontology alignment strategies. *Data Semantics*, 182-202, 2007.

[35] Naive Bayesian learning. http://en.wikipedia.org/wiki/naive_bayes_classifier.

[36] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proc. 10th International Conference on Very Large Data Bases (VLDB)*, pages 1–12, Santiago (CL), 1994.

[37] François Scharffe Jos de Bruijn Francisco Martin-Recuerda Dimitar Manov Marc Ehrig Livia Predoiu, Cristina Feier. D4.2.2 state-of-the-art survey on ontology merging and aligning v2. Sekt eu-ist integrated project, SEKT EU-IST Integrated Project, 2005.

[38] Alexander Maedche, Boris Motik, Ljiljana Stojanovic, Rudi Studer, and Raphael Volz. Ontologies for enterprise knowledge management. *IEEE Intelligent Systems*, 18(2):26–33, 2003.

[39] Alexander Maedche and Steffen Staab. *Measuring Similarity between Ontologies*. 2002.

[40] Ontology Matching. http://www.ontologymatching.org/.

[41] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[42] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm. In *18th International Conference on Data Engineering (ICDE)*, San Jose, CA US, 2002.

[43] Prasenjit Mitra, Natalya Noy, and Anuj Jaiswal. Ontology mapping discovery with uncertainty. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 537–547, Galway (IE), 2005.

[44] Natalya Noy. Semantic integration: A survey of ontology-based approaches. *ACM SIGMOD Record*, 33(4):65–70, 2004.

[45] Natalya Noy and Mark Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th National Conference on Artificial Intelligence (AAAI)*, pages 450–455, Austin (TX US), 2000.

[46] Natalya Noy and Mark Musen. PromptDiff: A fixed-point algorithm for comparing ontology versions. In *Proc. 18th National Conference on Artificial Intelligence (AAAI)*, pages 744–750, Edmonton (CA), 2002.

[47] Natalya F. Noy and Mark A. Musen. Evaluating ontology-mapping tools: Requirements and experience. In *OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management*, Siguenza, Spain, 2002.

[48] Natalya F. Noy and Mark A. Musen. The prompt suite: Interactive tools for ontology merging and mapping. Technical report, Stanford University, CA, USA, 2003.

[49] ODL_I3. http://www.service-architecture.com/database/articles/odmg_3_0.html.

[50] Rong Pan, Zhongli Ding, Yang Yu, and Yun Peng. A Bayesian network approach to ontology mapping. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 563–577, Hiroshima (JP), 2005.

[51] H. Sofia Pinto, Asuncion Gomez-Perez, and Joao P. Martins. Some issues on ontology integration. *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden*, 1999.

[52] Dave Robertson, Fausto Giunchiglia, Frank van Harmelen, Maurizio Marchese, Marta Sabou, Marco Schorlemmer, Nigel Shadbolt, Ronnie Siebes, Carles Sierra, Chris Walton, Srinandan Dasmahapatra, Dave Dupplaw, Paul Lewis,

Mikalai Yatskevich, Spyros Kotoulas, Adrian Perreau de Pinninck, and Antonis Loizou. Open knowledge semantic webs through peer-to-peer interaction. Technical Report DIT-06-034, University of Trento, 2006.

[53] M.A. Rodriguez and M.J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456, 2003.

[54] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

[55] SAT4J: A satisfiability library for Java. http://www.sat4j.org/.

[56] SecondString. http://secondstring.sourceforge.net/.

[57] Pavel Shvaiko and Jerome Euzenat. A survey of schema-based matching approaches. *Data Semantics*, 2005.

[58] Simetrics. http://www.dcs.shef.ac.uk/ sam/stringmetrics.html.

[59] SimPack. http://www.ifi.unizh.ch/ddis/simpack.html.

[60] Gerd Stumme and Alexander Maedche. Fca-merge: Bottom-up merging of ontologies, 2001.

[61] Xiaomeng Su. *Semantic Enrichment for Ontology Mapping*. PhD thesis, Dept. of Computer and Information Science, Norwegian University of Science and Technology, 2004.

[62] He Tan, Vaida Jakoniene, Patrick Lambrix, Johan Aberg, and Nahid Shahmehri. Alignment of Biomedical Ontologies using Life Science Literature. In *Proceedings of Workshop on Knowledge Discovery in Life Science Literature, Singapore (9th–12th April 2006)*, 2006.

[63] Le Bach Thanh, Dieng-Kuntz Rose, and Gandon Fabien. Ontology matching: A machine learning approach for building a corporate semantic web in a multi-communities organization, April 14-17 2004.

[64] the Alignment API. http://alignapi.gforge.inria.fr/.

[65] Bayes theorem. http://www.cise.ufl.edu/ fu/lecture/learn/bayes-fu.html.

[66] Decision tree. http://en.wikipedia.org/wiki/decision_tree_learning.

[67] UMLS. http://umlsks.nlm.nih.gov/kss/servlet/turbine/template/admineet.vm.

[68] Chao Wang, Jie Lu, and Guangquan Zhang. Integration of ontology data through learning instance matching, 2006.

[69] David Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[70] WordNet. http://wordnet.princeton.edu/.

[71] Li Xu and David Embley. Discovering direct and indirect matches for schema elements. In *Proc. 8th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 39–46, Kyoto (JP), 2003.