A Survey of Schema-based Matching Approaches*

Pavel Shvaiko[1] and Jérôme Euzenat[2]

[1] University of Trento, Povo, Trento, Italy,
pavel@dit.unitn.it
[2] INRIA, Rhône-Alpes, France,
Jerome.Euzenat@inrialpes.fr

Abstract. Schema and ontology matching is a critical problem in many application domains, such as semantic web, schema/ontology integration, data warehouses, e-commerce, etc. Many different matching solutions have been proposed so far. In this paper we present a new classification of schema-based matching techniques that builds on the top of state of the art in both schema and ontology matching. Some innovations are in introducing new criteria which are based on (i) general properties of matching techniques, (ii) interpretation of input information, and (iii) the kind of input information. In particular, we distinguish between approximate and exact techniques at schema-level; and syntactic, semantic, and external techniques at element- and structure-level. Based on the classification proposed we overview some of the recent schema/ontology matching systems pointing which part of the solution space they cover. The proposed classification provides a common conceptual basis, and, hence, can be used for comparing different existing schema/ontology matching techniques and systems as well as for designing new ones, taking advantages of state of the art solutions.

1 Introduction

Matching is a critical operation in many application domains, such as semantic web, schema/ontology integration, data warehouses, e-commerce, query mediation, etc. It takes as input two schemas/ontologies, each consisting of a set of discrete entities (e.g., tables, XML elements, classes, properties, rules, predicates), and determines as output the relationships (e.g., equivalence, subsumption) holding between these entities.

Many diverse solutions to the matching problem have been proposed so far, e.g., [2, 5, 21, 23, 40, 46, 49, 50, 52, 57, 76]. Good surveys through the recent years are provided in [39, 62, 75]; while the major contributions of the last decades are presented in [3, 41, 42, 66]. The survey of [39] focuses on current state of the art in ontology matching. Authors review recent approaches, techniques and tools. The survey of [75] concentrates on approaches to ontology-based information integration and discusses general matching approaches that are used in information integration systems. However, none of the above mentioned surveys provide a comparative review of the existing ontology matching techniques and systems. On the contrary, the survey of [62] is devoted to a classification of database schema matching approaches and a comparative

* For more information on the topic (e.g., tutorials, relevant events), please visit the Ontology Matching web-site at www.OntologyMatching.org

review of matching systems. Notice that these three surveys address the matching problem from different perspectives (artificial intelligence, information systems, databases) and analyze disjoint sets of systems.

This paper aims at considering the above mentioned works together, taking in to account some novel schema/ontology matching approaches, and at providing a common conceptual basis for their analysis. Although, there is a difference between schema and ontology matching problems (see next section for details), we believe that techniques developed for each of them can be of a mutual benefit. Thus, we bring together and discuss systematically recent approaches and systems developed in schema and ontology matching domains. We present a new classification of schema/ontology matching techniques that builds on the work of [62] augmented in [29, 67] and [28]. Some innovations are in introducing new criteria which are based on (i) general properties of matching techniques, (ii) interpretation of input information, and (iii) the kind of input information. In particular, we distinguish between approximate and exact techniques at schema-level; and syntactic, external, and semantic techniques at element- and structure-level. Based on the classification proposed we provide a comparative review of the recent schema/ontology matching systems pointing which part of the solution space they cover. In this paper we focus only on schema-based solutions, i.e., matching systems exploiting only schema-level information, not instance data.

The rest of the paper is organized as follows. Section 2 provides, via an example, the basic motivations and definitions to the schema/ontology matching problem. Section 3 discusses possible matching dimensions. Section 4 introduces a classification of elementary automatic schema-based matching techniques and discusses in detail possible alternatives. Section 5 provides a vision on classifying matching systems. Section 6 overviews some of the recent schema/ontology matching solutions in light of the classification proposed pointing which part of the solution space they cover. Section 7 reports some conclusions and discusses the future work.

2 The Matching Problem

2.1 Motivating Example

To motivate the matching problem, let us use two simple XML schemas that are shown in Figure 1 and exemplify one of the possible situations which arise, for example, when resolving a schema integration task.

Suppose an e-commerce company needs to finalize a corporate acquisition of another company. To complete the acquisition we have to integrate databases of the two companies. The documents of both companies are stored according to XML schemas S and S' respectively. Numbers in boxes are the unique identifiers of the XML elements. A first step in integrating the schemas is to identify candidates to be merged or to have taxonomic relationships under an integrated schema. This step refers to a process of schema matching. For example, the elements with labels Office_Products in S and in S' are the candidates to be merged, while the element with label Digital_Cameras in S' should be subsumed by the element with label Photo_and_Cameras in S. Once the correspondences between two schemas have been determined, the next step has to generate

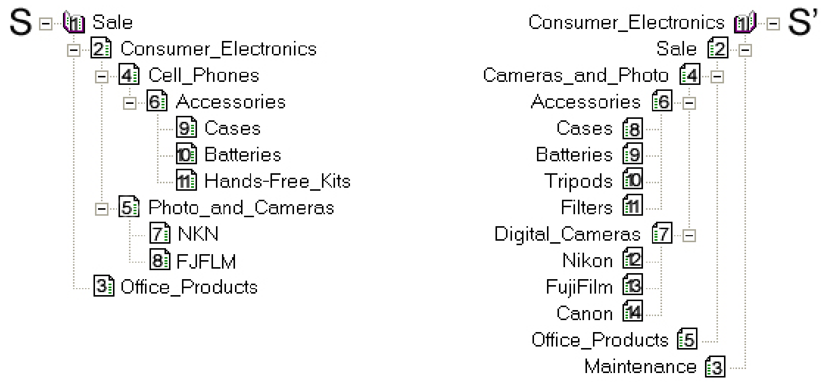

Fig. 1. Two XML schemas

query expressions that automatically translate data instances of these schemas under an integrated schema.

2.2 Schema Matching vs Ontology Matching

Many different kinds of structures can be considered as data/conceptual models: description logic terminologies, relational database schemas, XML-schemas, catalogs and directories, entity-relationship models, conceptual graphs, UML diagrams, etc. Most of the work on matching has been carried out among (i) database schemas in the world of information integration, (ii) XML-schemas and catalogs on the web, and (iii) ontologies in knowledge representation. Different parties, in general, have their own steadfast preferences for storing data. Therefore, when coordinating/integrating data among their information sources, it is often the case that we need to match between various data/conceptual models they are sticked to. In this respect, there are some important differences and commonalities between schema and ontology matching. The key points are:

- Database schemas often do not provide explicit semantics for their data. Semantics is usually specified explicitly at design-time, and frequently is not becoming a part of a database specification, therefore it is not available [56]. Ontologies are logical systems that themselves obey some formal semantics, e.g., we can interpret ontology definitions as a set of logical axioms.
- Ontologies and schemas are similar in the sense that (i) they both provide a vocabulary of terms that describes a domain of interest and (ii) they both constrain the meaning of terms used in the vocabulary [37, 70].
- Schemas and ontologies are found in such environments as the Semantic Web, and quite often in practice, it is the case that we need to match them.

On the one side, schema matching is usually performed with the help of techniques trying to guess the meaning encoded in the schemas. On the other side, ontology matching systems (primarily) try to exploit knowledge explicitly encoded in the ontologies.

In real-world applications, schemas/ontologies usually have both well defined and obscure labels (terms), and contexts they occur, therefore, solutions from both problems would be mutually beneficial.

2.3 Problem Statement

Following the work in [11, 27], we define a *mapping element* as a 5-uple: $\langle id, e, e', n, R\rangle$, where

- id is a unique identifier of the given mapping element;
- e and e' are the entities (e.g., tables, XML elements, properties, classes) of the first and the second schema/ontology respectively;
- n is a *confidence measure* in some mathematical structure (typically in the [0,1] range) holding for the correspondence between the entities e and e';
- R is a relation (e.g., *equivalence* (=); *more general* ($\sqsupseteq$); *disjointness* ($\perp$); *overlapping* ($\sqcap$)) holding between the entities e and e'.

An *alignment* is a set of mapping elements. The *matching* operation determines the alignment (A') for a pair of schemas/ontologies (o and o'). There are some other parameters which can extend the definition of the matching process, namely: (i) the use of an input alignment (A) which is to be completed by the process; (ii) the matching parameters, p (e.g., weights, thresholds); and (iii) external resources used by the matching process, r (e.g., thesauri); see Figure 2.

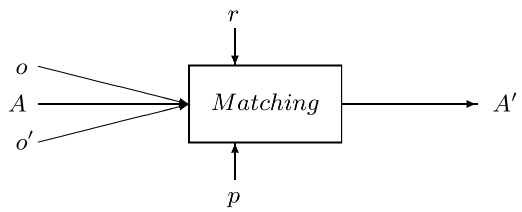

Fig. 2. The matching process

For example, in Figure 1, according to some matching algorithm based on linguistic and structure analysis, the confidence measure (for the fact that the equivalence relation holds) between entities with labels Photo_and_Cameras in S and Cameras_and_Photo in S' could be 0.67. Suppose that this matching algorithm uses a threshold of 0.55 for determining the resulting alignment, i.e., the algorithm considers all the pairs of entities with a confidence measure higher than 0.55 as correct mapping elements. Thus, our hypothetical matching algorithm should return to the user the following mapping element: $\langle id_{5,4}, Photo\_and\_Cameras, Cameras\_and\_Photo, 0.67, =\rangle$. However, the relation between the same pair of entities, according to another matching algorithm which is able to determine that both entities mean the same thing, could be exactly the equivalence relation (without computing the confidence measure). Thus, returning to the user $\langle id_{5,4}, Photo\_and\_Cameras, Cameras\_and\_Photo, n/a, =\rangle$.

2.4 Applications

Matching is an important operation in traditional applications, such as information integration, data warehousing, distributed query processing, etc. Typically, these applications are characterized by structural data/conceptual models, and are based on a design time matching operation, thereby determining alignment (e.g., manually or semi-automatically) as a prerequisite of running the system.

There is an emerging line of applications which can be characterized by their dynamics (e.g., agents, peer-to-peer systems, web services). Such applications, on the contrary to traditional ones, require a run time matching operation and take advantage of more ”explicit” conceptual models.

Below, we first discuss an example of a traditional application, namely, catalog integration. Then, we focus on emergent applications, namely, peer-to-peer (P2P) databases, agent communication, and web services integration.

Catalog Integration. In B2B applications, trade partners store their products in electronic catalogs. Catalogs are tree-like structures, namely concept hierarchies with attributes. Typical examples of catalogs are product directories of www.amazon.com, www.ebay.com, etc. In order for a private company to participate in the marketplace (e.g., eBay), it is used to determine correspondences between entries of its catalogs and entries of a single catalog of a marketplace. This process of mapping entries among catalogs is referred to the catalog matching problem, see [12]. Having identified the correspondences between the entries of the catalogs, they are further analyzed in order to generate query expressions that automatically translate data instances between the catalogs, see, for example, [74]. Finally, having aligned the catalogs, users of a marketplace have a unified access to the products which are on sale.

P2P Databases. P2P networks are characterized by an extreme flexibility and dynamics. Peers may appear and disappear on the network, their databases are autonomous in their language, contents, how they can change their schemas, and so on. Since peers are autonomous, they might use different terminology, even if they refer to the same domain of interest. Thus, in order to establish (meaningful) information exchange between peers, one of the steps is to identify and characterize relationships between their schemas. Having identified the relationships between schemas, next step is to use these relationships for the purpose of query answering, for example, using techniques applied in data integration systems, namely Local-as-View (LAV), Global-as-View (GAV), or Global-Local-as-View (GLAV) [43]. However, P2P applications pose additional requirements on matching algorithms. In P2P settings an assumption that all the peers rely on one global schema, as in data integration, can not be made, because the global schema may need to be updated any time the system evolves, see [36]. Thus, if in the case of data integration schema matching operation can be performed at design time, in P2P applications peers need coordinating their databases on the fly, therefore requiring a run time schema matching operation.

Agent Communication. Agents are computer entities characterized by autonomy and capacity of interaction. They communicate through speech-act inspired languages which determine the ”envelope” of the messages and enable agents to position them within a particular interaction context. The actual content of messages is expressed in knowledge representation languages and often refer to some ontology. As a consequence, when two autonomous and independently designed agents meet, they have the possibility of exchanging messages, but little chance to understand each others if they do not share the same content language and ontology. Thus, it is necessary to provide the possibility for these agents to match their ontologies in order to either translate their messages or integrate bridge axioms in their own models, see [73]. One solution to this problem is to have an ontology alignment protocol that can be interleaved with any other agent interaction protocol and which could be triggered upon receiving a message expressed in an alien ontology. As a consequence, agents meeting each other for the first time and using different ontologies would be able to negotiate the matching of terms in their respective ontologies and to translate the content of the message they exchange with the help of the alignment.

Web Services Integration. Web services are processes that expose their interface to the web so that users can invoke them. Semantic web services provide a richer and more precise way to describe the services through the use of knowledge representation languages and ontologies. Web service discovery and integration is the process of finding a web service able to deliver a particular service and composing several services in order to achieve a particular goal, see [59]. However, semantic web services descriptions have no reasons to be expressed by reference to exactly the same ontologies. Henceforth, both for finding the adequate service and for interfacing services it will be necessary to establish the correspondences between the terms of the descriptions. This can be provided through matching the corresponding ontologies. For instance, if some service provides its output description in some ontology and another service uses a second ontology for describing its input, matching both ontologies will be used for (i) checking that what is delivered by the first service matches what is expected by the second one, (ii) verifying preconditions of the second service, and (iii) generating a mediator able to transform the output of the first service in order to be input to the second one.

In some of the above mentioned applications (e.g., two agents meeting or looking for the web services integration) there are no instances given beforehand. Thus, it is necessary to perform matching without them, based only on schema-level information.

3 The Matching Dimensions

There are many independent dimensions along which algorithms can be classified. As from Figure 2, we may classify them according to (i) input of the algorithms, (ii) characteristics of the matching process, and (iii) output of the algorithms. Let us discuss them in turn.

Input dimensions. These dimensions concern the kind of input on which algorithms operate. As a first dimension, algorithms can be classified depending on the data / conceptual models in which ontologies or schemas are expressed. For example, the Artemis

[13] system supports the relational, OO, and ER models; Cupid [46] supports XML and relational models; QOM [26] supports RDF and OWL models. A second possible dimension depends on the kind of data that the algorithms exploit: different approaches exploit different information of the input data/conceptual models, some of them rely only on schema-level information (e.g., Cupid [46], COMA [21]), others rely only on instance data (e.g., GLUE [23]), or exploit both, schema- and instance-level information (e.g., QOM [26]). Even with the same data models, matching systems do not always use all available constructs, e.g., S-Match [34] when dealing with attributes discards information about datatypes (e.g., string, integer), and uses only the attributes names. In general, some algorithms focus on the labels assigned to the entities, some consider their internal structure and the type of their attributes, and some others consider their relations with other entities (see next section for details).

Process dimensions. A classification of the matching process could be based on its general properties, as soon as we restrict ourselves to formal algorithms. In particular, it depends on the *approximate* or *exact* nature of its computation. Exact algorithms compute the absolute solution to a problem; approximate algorithms sacrifice exactness to performance (e.g., [26]). All the techniques discussed in the remainder of the paper can be either approximate or exact. Another dimension for analyzing the matching algorithms is based on the way they interpret the input data. We identify three large classes based on the intrinsic input, external resources, or some semantic theory of the considered entities. We call these three classes *syntactic*, *external*, and *semantic* respectively; and discuss them in detail in the next section.

Output dimensions. Apart from the information that matching systems exploit and how they manipulate it, the other important class of dimensions concerns the form of the result they produce. The form of the alignment might be of importance: is it a one-to-one correspondence between the schema/ontology entities? Has it to be a final mapping element? Is any relation suitable?

Other significant distinctions in the output results have been indicated in [32]. One dimension concerns whether systems deliver a graded answer, e.g., that the correspondence holds with 98% confidence or 4/5 probability; or an all-or-nothing answer, e.g., that the correspondence definitely holds or not. In some approaches correspondences between schema/ontology entities are determined using distance measures. This is used for providing an alignment expressing equivalence between these entities in which the actual distance is the ground for generating a confidence measure in each correspondence, usually in [0,1] range, see, for example, [29, 46]. Another dimension concerns the kind of relations between entities a system can provide. Most of the systems focus on equivalence (=), while a few other are able to provide a more expressive result (e.g., equivalence, subsumption ($\sqsubseteq$), incompatibility ($\perp$), see for details [12, 33]).

There are many dimensions that can be taken into account when attempting at classifying matching methods. In the next section we present a classification of elementary techniques that draws simultaneously on several such criteria.

4 A retained classification of elementary schema-based matching approaches

In this section we discuss only schema-based elementary matchers. We address issues of their combination in the next section. Therefore, only schema/ontology information is considered, not instance data[1]. The exact/approximate opposition has not been used because each of the methods described below can be implemented as exact or approximate algorithm, depending on the goals of the matching system. To ground and ensure a comprehensive coverage for our classification we have analyzed state of the art approaches used for schema-based matching. The references section reports a partial list of works which have been scrutinized pointing to (some of) the most important contributions. We have used the following guidelines for building our classification:

Exhaustivity. The extension of categories dividing a particular category must cover its extension (i.e., their aggregation should give the complete extension of the category);

Disjointness. In order to have a proper tree, the categories dividing one category should be pairwise disjoint by construction;

Homogeneity. In addition, the criterion used for further dividing one category should be of the same nature (i.e., should come from the same dimension). This usually helps guaranteeing disjointness;

Saturation. Classes of concrete matching techniques should be as specific and discriminative as possible in order to provide a fine grained distinction between possible alternatives. These classes have been identified following a *saturation* principle: they have been added/modified till the saturation was reached, namely taking into account new techniques did not require introducing new classes or modifying them.

Notice that disjointness and exhaustivity of the categories ensures stability of the classification, namely new techniques will not occur in between two categories. Classes of matching techniques represent the state of the art. Obviously, with appearance of new techniques, they might be extended and further detailed.

As indicated in introduction, we build on the previous work of classifying automated schema matching approaches of [62]. The classification of [62] distinguishes between *elementary* (individual) matchers and *combinations* of matchers. Elementary matchers comprise *instance-based* and *schema-based*, *element-* and *structure-level*, *linguistic-* and *constrained-based* matching techniques. Also *cardinality* and *auxiliary information* (e.g., thesauri, global schemas) can be taken into account.

For classifying elementary schema-based matching techniques, we introduce two synthetic classifications (see Figure 3), based on what we have found the most salient properties of the matching dimensions. These two classifications are presented as two trees sharing their leaves. The leaves represent classes of elementary matching techniques and their concrete examples. Two synthetic classifications are:

[1] Prominent solutions of instance-based schema/ontology matching as well as possible extensions of the instance-based part of the classification of [62] can be found in [23] and [40] respectively.

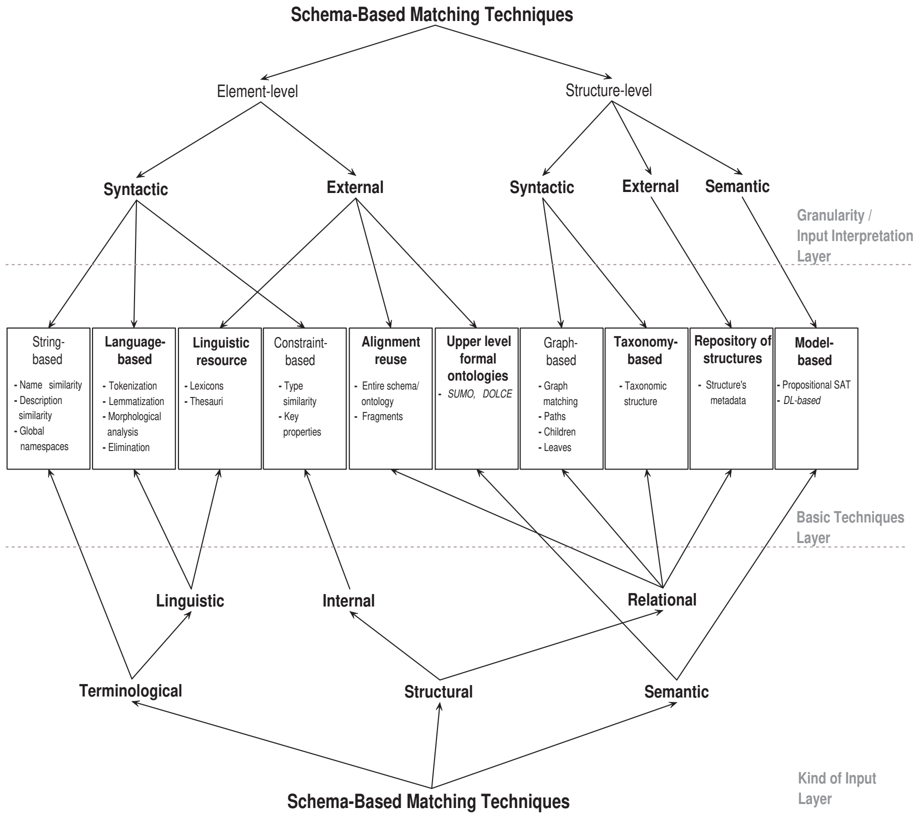

Fig. 3. A retained classification of elementary schema-based matching approaches

– *Granularity/Input Interpretation* classification is based on (i) granularity of match, i.e., element- or structure-level, and then (ii) on how the techniques generally interpret the input information;
– *Kind of Input* classification is based on the kind of input which is used by elementary matching techniques.

The overall classification of Figure 3 can be read both in descending (focusing on how the techniques interpret the input information) and ascending (focusing on the kind of manipulated objects) manner in order to reach the *Basic Techniques* layer. Let us discuss in turn *Granularity/Input Interpretation*, *Basic Techniques*, *Kind of Input* layers together with supporting arguments for the categories/classes introduced at each layer.

Elementary matchers are distinguished by the *Granularity/Input interpretation* layer according to the following classification criteria:

– *Element-level vs structure-level.* Element-level matching techniques compute mapping elements by analyzing entities in isolation, ignoring their relations with other entities. Structure-level techniques compute mapping elements by analyzing how entities appear together in a structure. This criterion is the same as first introduced in [62].

- *Syntactic vs external vs semantic.* The key characteristic of the syntactic techniques is that they interpret the input in function of its sole structure following some clearly stated algorithm. External are the techniques exploiting auxiliary (external) resources of a domain and common knowledge in order to interpret the input. These resources might be human input or some thesaurus expressing the relationships between terms. The key characteristic of the semantic techniques is that they use some formal semantics (e.g., model-theoretic semantics) to interpret the input and justify their results. In case of a semantic based matching system, exact algorithms are complete (i.e., they guarantee a discovery of all the possible mappings) while approximate algorithms tend to be incomplete.

To emphasize the differences with the initial classification of [62], the new categories/classes are marked in bold face. In particular, in the *Granularity/Input Interpretation* layer we detail further (with respect to [62]), the element- and structure-level of matching by introducing the *syntactic* vs *semantic* vs *external* criteria. The reasons of having these three categories are as follows. Our initial criterion was to distinguish between *internal* and *external* techniques. By *internal* we mean techniques exploiting information which comes only with the input schemas/ontologies. *External* techniques are as defined above. *Internal* techniques can be further detailed by distinguishing between *syntactic* and *semantic* interpretation of input, also as defined above. However, only limited, the same distinction can be introduced for the *external* techniques. In fact, we can qualify some oracles (e.g., WordNet [53], DOLCE [31]) as syntactic or semantic, but not a user's input. Thus, we do not detail *external* techniques any further and we omit in Figure 3 the theoretical category of *internal* techniques (as opposed to *external*). Notice, that we also omit in further discussions element-level semantic techniques, since semantics is usually given in a structure, and, hence, there are no element-level semantic techniques.

Distinctions between classes of elementary matching techniques in the *Basic Techniques* layer of our classification are motivated by the way a matching technique interprets the input information in each concrete case. In particular, a label can be interpreted as a string (a sequence of letters from an alphabet) or as a word or a phrase in some natural language, a hierarchy can be considered as a graph (a set of nodes related by edges) or a taxonomy (a set of concepts having a set-theoretic interpretation organized by a relation which preserves inclusion). Thus, we introduce the following classes of elementary schema/ontology matching techniques at the element-level: *string-based*, *language-based*, *based on linguistic resources*, *constraint-based*, *alignment reuse*, and *based on upper level ontologies*. At the structure-level we distinguish between: *graph-based*, *taxonomy-based*, *based on repositories of structures*, and *model-based techniques*.

The *Kind of Input* layer classification is concerned with the type of input considered by a particular technique:

- The first level is categorized depending on which kind of data the algorithms work on: strings (*terminological*), structure (*structural*) or models (*semantics*). The two first ones are found in the ontology descriptions, the last one requires some semantic interpretation of the ontology and usually uses some semantically compliant reasoner to deduce the correspondences.

– The second level of this classification decomposes further these categories if necessary: *terminological* methods can be *string-based* (considering the terms as sequences of characters) or based on the interpretation of these terms as linguistic objects (*linguistic*). The structural methods category is split into two types of methods: those which consider the *internal* structure of entities (e.g., attributes and their types) and those which consider the relation of entities with other entities (*relational*).

Notice that following the above mentioned guidelines for building a classification the *terminological* category should be divided into *linguistic* and *non-linguistic* techniques. However, since *non-linguistic* techniques are all *string-based*, this category has been discarded.

We discuss below the main classes of the *Basic Techniques* layer (also indicating in which matching systems they are exploited) according to the above classification in more detail. The order follows that of the *Granularity/Input Interpretation* classification and these techniques are divided in two sections concerning element-level techniques (§4.1) and structure-level techniques (§4.2). Finally, in Figure 3, techniques which are marked in italic (techniques based on upper level ontologies and DL-based techniques) have not been implemented in any matching system yet. However, we are arguing why their appearance seems reasonable in the near future.

4.1 Element-level techniques

String-based techniques are often used in order to match names and name descriptions of schema/ontology entities. These techniques consider strings as sequences of letters in an alphabet. They are typically based on the following intuition: the more similar the strings, the more likely they denote the same concepts. A comparison of different string matching techniques, from *distance* like functions to *token-based distance* functions can be found in [16]. Usually, distance functions map a pair of strings to a real number, where a smaller value of the real number indicates a greater similarity between the strings. Some examples of string-based techniques which are extensively used in matching systems are *prefix*, *suffix*, *edit distance*, and *n-gram*.

– *Prefix*. This test takes as input two strings and checks whether the first string starts with the second one. *Prefix* is efficient in matching cognate strings and similar acronyms (e.g., int and integer), see, for example [21, 34, 46, 50]. This test can be transformed in a smoother distance by measuring the relative size of the prefix and the ratio.
– *Suffix*. This test takes as input two strings and checks whether the first string ends with the second one (e.g., phone and telephone), see, for example [21, 34, 46, 50].
– *Edit distance*. This distance takes as input two strings and computes the edit distance between the strings. That is, the number of *insertions*, *deletions*, and *substitutions* of characters required to transform one string into another, normalized by the length of the longest string. For example, the edit distance between NKN and Nikon is 0.4. Some of matching systems exploiting the given technique are [21, 34, 57].

– *N-gram*. This test takes as input two strings and computes the number of common n-grams (i.e., sequences of n characters) between them. For example, $trigram(3)$ for the string nikon are nik, iko, kon. Thus, the distance between nkon and nikon would be $1/3$. Some of matching systems exploiting the given test are [21, 34].

Language-based techniques consider names as words in some natural language (e.g., English). They are based on Natural Language Processing (NLP) techniques exploiting morphological properties of the input words.

– *Tokenization*. Names of entities are parsed into sequences of *tokens* by a tokenizer which recognizes punctuation, cases, blank characters, digits, etc. (e.g., Hands-Free_Kits → ⟨hands, free, kits⟩, see, for example [33]).
– *Lemmatization*. The strings underlying tokens are morphologically analyzed in order to find all their possible basic forms (e.g., Kits → Kit), see, for example [33].
– *Elimination*. The tokens that are articles, prepositions, conjunctions, and so on, are marked (by some matching algorithms, e.g., [46]) to be discarded.

Usually, the above mentioned techniques are applied to names of entities before running string-based or lexicon-based techniques in order to improve their results. However, we consider these language-based techniques as a separate class of matching techniques, since they can be naturally extended, for example, in a distance computation (by comparing the resulting strings or sets of strings).

Constraint-based techniques are algorithms which deal with the internal constraints being applied to the definitions of entities, such as types, cardinality of attributes, and keys. We omit here a discussion of matching keys as these techniques appear in our classification without changes with respect to the original publication [62]. However, we provide a different perspective on matching datatypes and cardinalities.

– *Datatypes comparison* involves comparing the various attributes of a class with regard to the datatypes of their value. Contrary to objects that require interpretations, the datatypes can be considered objectively and it is possible to determine how a datatype is close to another (ideally this can be based on the interpretation of datatypes as sets of values and the set-theoretic comparison of these datatypes, see [71, 72]). For instance, the datatype day can be considered closer to the datatype workingday than the datatype integer. This technique is used in [30].
– *Multiplicity comparison* attribute values can be collected by a particular construction (set, list, multiset) on which cardinality constraints are applied. Again, it is possible to compare the so constructed datatypes by comparing (i) the datatypes on which they are constructed and (ii) the cardinality that are applied to them. For instance, a set of between 2 and 3 children is closer to a set of 3 people than a set of 10-12 flowers (if children are people). This technique is used in [30].

Linguistic resources such as common knowledge or domain specific thesauri are used in order to match words (in this case names of schema/ontology entities are considered as words of a natural language) based on linguistic relations between them (e.g., synonyms, hyponyms).

– *Common knowledge thesauri*. The approach is to use common knowledge thesauri to obtain meaning of terms used in schemas/ontologies. For example, WordNet [53] is an electronic lexical database for English (and other languages), where various *senses* (possible meanings of a word or expression) of words are put together into sets of synonyms. Relations between schema/ontology entities can be computed in terms of bindings between WordNet senses, see, for instance [12, 33]. For example, in Figure 1, a sense-based matcher may learn from WordNet (with a prior morphological preprocessing of labels performed) that Camera in S is a hypernym for Digital Camera in S', and, therefore conclude that entity Digital_Cameras in S' should be subsumed by the entity Photo_and_Cameras in S. Another type of matchers exploiting thesauri is based on their structural properties, e.g., WordNet hierarchies. In particular, hierarchy-based matchers measure the distance, for example, by counting the number of arcs traversed, between two concepts in a given hierarchy, see [35]. Several other distance measures for thesauri have been proposed in the literature, e.g., [61, 64].
– *Domain specific thesauri*. These thesauri usually store some specific domain knowledge, which is not available in the common knowledge thesauri, (e.g., proper names) as entries with synonym, hypernym and other relations. For example, in Figure 1, entities NKN in S and Nikon in S' are treated by a matcher as synonyms from a domain thesaurus look up: syn key - "NKN:Nikon = syn", see, for instance [46].

Alignment reuse techniques represent an alternative way of exploiting external resources, which contain in this case alignments of previously matched schemas/ontologies. For instance, when we need to match schema/ontology o' and o'', given the alignments between o and o', and between o and o'' from the external resource, storing previous match operations results. The alignment reuse is motivated by the intuition that many schemas/ontologies to be matched are similar to already matched schemas/ontologies, especially if they are describing the same application domain. These techniques are particularly promising when dealing with large schemas/ontologies consisting of hundreds and thousands of entities. In these cases, first, large match problems are decomposed into smaller sub-problems, thus generating a set of schema/ontology fragments matching problems. Then, reusing previous match results can be more effectively applied at the level of schema/ontology fragments compared to entire schemas/ontologies. The approach was first introduced in [62], and later was implemented as two matchers, i.e., (i) reuse alignments of entire schemas/ontologies, or (ii) their fragments, see, for details [2, 21, 63].

Upper level formal ontologies can be also used as external sources of common knowledge. Examples are the Suggested Upper Merged Ontology (SUMO) [55] and Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [31]. The key characteristic of these ontologies is that they are logic-based systems, and therefore, matching techniques exploiting them can be based on the analysis of interpretations. Thus, these are semantic techniques. For the moment, we are not aware of any matching systems which use these kind of techniques. However, it is quite reasonable to assume that this will happen in the near future. In fact, for example, the DOLCE ontology aims at providing a formal specification (axiomatic theory) for the top level part of WordNet. Therefore, systems exploiting WordNet now in their matching process might also consider using DOLCE as a potential extension.

4.2 Structure-level techniques

Graph-based techniques are graph algorithms which consider the input as labeled graphs. The applications (e.g., database schemas, taxonomies, or ontologies) are viewed as graph-like structures containing terms and their inter-relationships. Usually, the similarity comparison between a pair of nodes from the two schemas/ontologies is based on the analysis of their positions within the graphs. The intuition behind is that, if two nodes from two schemas/ontologies are similar, their neighbors might also be somehow similar. Below, we present some particular matchers representing this intuition.

- *Graph matching*. There have been done a lot of work on graph (tree) matching in graph theory and also with respect to schema/ontology matching applications, see, for example, [65, 77]. Matching graphs is a combinatorial problem that can be computationally expensive. It is usually solved by approximate methods. In schema/ontology matching, the problem is encoded as an optimization problem (finding the graph matching minimizing some distance like the dissimilarity between matched objects) which is further resolved with the help of a graph matching algorithm. This optimization problem is solved through a fix-point algorithm (improving gradually an approximate solution until no improvement is made). Examples of such algorithms are [50] and [30]. Some other (particular) matchers handling DAGs and trees are *children*, *leaves*, and *relations*.
- *Children*. The (structural) similarity between inner nodes of the graphs is computed based on similarity of their children nodes, that is, two non-leaf schema elements are structurally similar if their immediate children sets are highly similar. A more complex version of this matcher is implemented in [21].
- *Leaves*. The (structural) similarity between inner nodes of the graphs is computed based on similarity of leaf nodes, that is, two non-leaf schema elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not, see, for example [21, 46].
- *Relations*. The similarity computation between nodes can also be based on their relations. For example, in one of the possible ontology representations of schemas of Figure 1, if class Photo_and_Cameras relates to class NKN by relation hasBrand in one ontology, and if class Digital_Cameras relates to class Nikon by relation hasMarque in the other ontology, then knowing that classes Photo_and_Cameras and Digital_Cameras are similar, and also relations hasBrand and hasMarque are similar, we can infer that NKN and Nikon may be similar too, see [48].

Taxonomy-based techniques are also graph algorithms which consider only the specialization relation. The intuition behind taxonomic techniques is that *is-a* links connect terms that are already similar (being a subset or superset of each other), therefore their neighbors may be also somehow similar. This intuition can be exploited in several different ways:

- *Bounded path matching*. Bounded path matchers take two paths with links between classes defined by the hierarchical relations, compare terms and their positions along these paths, and identify similar terms, see, for instance [57]. For example, in Figure 1, given that element Digital_Cameras in S' should be subsumed by the element Photo_and_Cameras in S, a matcher would suggest FJFLM in S and FujiFilm in S' as an appropriate match.
- *Super(sub)-concepts rules*. These matchers are based on rules capturing the above stated intuition. For example, if super-concepts are the same, the actual concepts are similar to each other. If sub-concepts are the same, the compared concepts are also similar, see, for example [19, 26].

Repository of structures stores schemas/ontologies and their fragments together with pairwise similarities (e.g., coefficients in the [0 1] range) between them. Notice, that unlike the alignment reuse, repository of structures stores only similarities between schemas/ontologies, not alignments. In the following, to simplify the presentation, we call schemas/ontologies or their fragments as structures. When new structures are to be matched, they are first checked for similarity to the structures which are already available in the repository. The goal is to identify structures which are sufficiently similar to be worth matching in more detail, or reusing already existing alignments. Thus, avoiding the match operation over the dissimilar structures. Obviously, the determination of similarity between structures should be computationally cheaper than matching them in full detail. The approach of [63], in order to match two structures, proposes to use some metadata describing these structures, such as structure name, root name, number of nodes, maximal path length, etc. Then, these indicators are analyzed and are aggregated into a single coefficient, which estimates similarity between them. For example, schema $S1$ might be found as an appropriate match to schema $S2$ since they both have the same number of nodes.

Model-based algorithms handle the input based on its semantic interpretation (e.g., model-theoretic semantics). Thus, they are well grounded deductive methods. Examples are propositional satisfiability (SAT) and description logics (DL) reasoning techniques.

- *Propositional satisfiability (SAT)*. As from [12, 32, 33], the approach is to decompose the graph (tree) matching problem into the set of node matching problems. Then, each node matching problem, namely pairs of nodes with possible relations between them is translated into a propositional formula of form: $Axioms \rightarrow rel(context_1, context_2)$, and checked for *validity*. $Axioms$ encodes background knowledge (e.g., Digital_Cameras $\rightarrow$ Cameras codifies the fact that Digital_Cameras is less general than Cameras), which is used as premises to reason about relations *rel* (e.g., $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$) holding between the nodes $context_1$ and $context_2$ (e.g., node 7 in S and node 12 in S'). A propositional formula is valid iff its negation is unsatisfiable. The unsatisfiability is checked by using state of the art SAT solvers. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability, and therefore, they can be used for an exhaustive check of all the possible mappings.

– *DL-based techniques*. The SAT-based approach computes the satisfiability of theory merging both schemas/ontologies along an alignment. Propositional language used for codifying matching problems into propositional unsatisfiability problems is limited in its expressivity, namely it allows for handling only unary predicates. Thus, it can not handle, for example, binary predicates, such as properties or roles. However, the same procedure can be carried within description logics (expressing properties). In description logics, the relations (e.g., $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$) can be expressed in function of subsumption. In fact, first merging two ontologies (after renaming) and then testing each pair of concepts and roles for subsumption is enough for aligning terms with the same interpretation (or with a subset of the interpretations of the others). For instance, suppose that we have one ontology introducing classes company, employee and micro-company as a company with at most 5 employees, and another ontology introducing classes firm, associate and SME as a firm with at most 10 associates. If we know that all associates are employees and we already have established that firm is equivalent to company, then we can deduce that a micro-company is a SME. However, we are not aware of existence of any schema/ontology matching systems supporting DL-based techniques for the moment.

There are examples in the literature of DL-based techniques used in relevant to schema/ontology matching applications. For example, in spatio-temporal database integration scenario, as first motivated in [60] and later developed in [68] the inter-schema mapping elements are initially proposed by the integrated schema designer and are encoded together with input schemas in $\mathcal{ALCRP}(\mathcal{S}_2 \oplus \mathcal{T})$ language. Then, DL reasoning services are used to check the satisfiability of the two source schemas and the set of inter-schema mapping elements. If some objects are found unsatisfied, then the inter-schema mapping elements should be reconsidered.

Another example, is when DL-based techniques are used in query processing scenario [52]. The approach assumes that mapping elements between pre-existing domain ontologies are already specified in a declarative manner (e.g., manually). User queries are rewritten in terms of pre-existing ontologies and are expressed in Classic [10], and further evaluated against real-world repositories, which are also subscribed to the pre-existing ontologies. An earlier approach for query answering by terminological reasoning is described in [4].

Finally, a very similar problem to schema/ontology matching is addressed within the system developed for matchmaking in electronic marketplaces [18]. Demand D and supply S requests are translated from natural language sentences into Classic [10]. The approach assumes the existence of a pre-defined domain ontology T, which is also encoded in Classic. Matchmaking between a supply S and a demand D is performed with respect to the pre-defined domain ontology T. Reasoning is performed with the help of the NeoClassic reasoner in order to determine the *exact match* ($T \models (D \sqsubseteq S)$) and ($T \models (S \sqsubseteq D)$), *potential match* (if $D \sqcap S$ is satisfiable in T), and *nearly miss* (if $D \sqcap S$ is unsatisfiable in T). The system also provides a logically based matching results rank operation.

5 On classifying matching systems

As the previous section indicates, elementary matchers rely on a particular kind of input information, therefore they have different applicability and value with respect to different schema/ontology matching tasks. State of the art matching systems are not made of a single elementary matcher. They usually combine them: elementary matchers can be used in sequence (called *hybrid* matchers in [62]), examples are [5, 46], or in parallel (also called *composite* matchers [62]) combining the results (e.g., taking the average, maximum) of independently executed matchers, see, for instance [21, 23, 26]. Finding a better classification here is rather difficult.

The distinction between the sequential and parallel composition is useful from an architectural perspective. However, it does not show how the systems can be distinguished in the matter of considering the alignment and the matching task, thus representing an user-centric perspective. Below, we provide a vision of a classification of matching systems with respect to this point:

– *Alignments as solutions.* This category covers purely algorithmic techniques that consider that an alignment is a solution to the matching problem. It could be characterized as a (continuous or discrete) optimization problem, see, for example [30, 50].
– *Alignments as theorems.* Systems of this category rely on semantics and require the alignment to satisfy it. This category, strictly speaking, is a sub-category of *alignments as solutions* (the problem is expressed in semantic terms). However, it is sufficiently autonomous for being singled out, see, for example [32, 33].
– *Alignments as likeness clues.* This category refers to the algorithms which aim at producing reasonable indications for a user to select the alignment, although using the same techniques (e.g., string-based) as systems from the *alignments as solutions* category, see, for example [21, 46].

6 Review of state of the art matching systems

We now look at some recent schema-based state of the art matching systems in light of the classification presented in Figure 3 and criteria highlighted in Section 5.

Similarity Flooding. The Similarity Flooding (SF) [50] approach utilizes a hybrid matching algorithm based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs; grounding on the OIM specification [15] the algorithm manipulates them in an iterative fix-point computation to produce an alignment between the nodes of the input graphs. The technique starts from string-based comparison (common prefix, suffix tests) of the vertices labels to obtain an initial alignment which is refined within the fix-point computation. The basic concept behind the SF algorithm is the similarity spreading from similar nodes to the adjacent neighbors through propagation coefficients. From iteration to iteration the spreading depth and a similarity measure are increasing till the fix-point is reached. The result of this step is a refined alignment which is further filtered to finalize the matching process. SF considers the alignment as a solution to a clearly stated optimization problem.

Artemis. Artemis (Analysis of Requirements: Tool Environment for Multiple Information Systems) [13] was designed as a module of the MOMIS mediator system [5] for creating global views. It performs affinity-based analysis and hierarchical clustering of source schema elements. Affinity-based analysis represents the matching step: in a hybrid manner it calculates the name, structural and global affinity coefficients exploiting a common thesaurus. The common thesaurus is built with the help of ODB-Tools, WordNet or manual input. It represents a set of intensional and extensional relationships which depict intra- and inter-schema knowledge about classes and attributes of the input schemas. Based on global affinity coefficients, a hierarchical clustering technique categorizes classes into groups at different levels of affinity. For each cluster it creates a set of global attributes - global class. Logical correspondence between the attributes of a global class and source schema attributes is determined through a mapping table. Artemis falls into the alignments as likeness clues category.

Cupid. Cupid [46] implements a hybrid matching algorithm comprising linguistic and structural schema matching techniques, and computes similarity coefficients with the assistance of a domain specific thesauri. Input schemas are encoded as graphs. Nodes represent schema elements and are traversed in a combined bottom-up and top-down manner. The matching algorithm consists of three phases and operates only with tree-structures to which non-tree cases are reduced. The first phase (linguistic matching) computes linguistic similarity coefficients between schema element names (labels) based on morphological normalization, categorization, string-based techniques (common prefix, suffix tests) and a thesauri look-up. The second phase (structural matching) computes structural similarity coefficients weighted by leaves which measure the similarity between contexts in which elementary schema elements occur. The third phase (mapping elements generation) computes weighted similarity coefficients and generates final alignment by choosing pairs of schema elements with weighted similarity coefficients which are higher than a threshold. Referring to [46], Cupid performs somewhat better overall, than the other hybrid matchers: Dike [58] and Artemis [13]. Cupid falls into the alignments as likeness clues category.

COMA. COMA (COmbination of MAtching algorithms) [21] is a composite schema matching tool. It provides an extensible library of matching algorithms; a framework for combining obtained results, and a platform for the evaluation of the effectiveness of the different matchers. Matching library is extensible, and as from [21] it contains 6 elementary matchers, 5 hybrid matchers, and one reuse-oriented matcher. Most of them implement string-based techniques (affix, n-gram, edit distance, etc.) as a background idea; others share techniques with Cupid (thesauri look-up, etc.); and reuse-oriented is a completely novel matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Schemas are internally encoded as DAGs, where elements are the paths. This aims at capturing contexts in which the elements occur. Distinct features of the COMA tool in respect to Cupid, are a more flexible architecture and a possibility of performing iterations in the matching process. Based on the comparative evaluations conducted in [20], COMA dominates Autoplex[6] and Automatch [7]; LSD [22] and GLUE [23]; SF [50], and SemInt [44] matching tools. COMA falls into the alignments as likeness clues category.

NOM. NOM (Naive Ontology Mapping) [26] adopts the idea of composite matching from COMA [21]. Some other innovations with respect to COMA, are in the set of elementary matchers based on rules, exploiting explicitly codified knowledge in ontologies, such as information about super- and sub-concepts, super- and sub-properties, etc. At present the system supports 17 rules. For example, rule (R5) states that if super-concepts are the same, the actual concepts are similar to each other. NOM also exploits a set of instance-based techniques, this topic is beyond scope of the paper. The system falls into the alignments as likeness clues category.

QOM. QOM (Quick Ontology Mapping) [25] is a successor of the NOM system [26]. The approach is based on the idea that the loss of quality in matching algorithms is marginal (to a standard baseline), however improvement in efficiency can be tremendous. This fact allows QOM to produce mapping elements fast, even for large-size ontologies. QOM is grounded on matching rules of NOM. However, for the purpose of efficiency the use of some rules have been restricted, e.g., R5. QOM avoids the complete pair-wise comparison of trees in favor of a (n incomplete) top-down strategy. Experimental study has shown that QOM is on a par with other state of the art algorithms concerning the quality of proposed alignment, while outperforming them with respect to efficiency. Also, QOM shows better quality results than approaches within the same complexity class. The system falls into the alignments as likeness clues category.

OLA. OLA (OWL Lite Aligner) [30] is designed with the idea of balancing the contribution of each component that compose an ontology (classes, properties, names, constraints, taxonomy, and even instances). As such it takes advantage of all the elementary matching techniques that have been considered in the previous sections, but the semantic ones. OLA is a family of distance based algorithms which converts definitions of distances based on all the input structures into a set of equations. These distances are almost linearly aggregated (they are linearly aggregated modulo local matches of entities). The algorithm then looks for the matching between the ontologies that minimizes the overall distance between them. For that purpose it starts with base distance measures computed from labels and concrete datatypes. Then, it iterates a fix-point algorithm until no improvement is produced. From that solution, an alignment is generated which satisfies some additional criterion (on the alignment obtained and the distance between aligned entities). As a system, OLA considers the alignment as a solution to a clearly stated optimization problem.

Anchor-PROMPT. Anchor-PROMPT [57] (an extension of PROMPT, also formerly known as SMART) is an ontology merging and alignment tool with a sophisticated prompt mechanism for possible matching terms. The anchor-PROMPT is a hybrid alignment algorithm which takes as input two ontologies, (internally represented as graphs) and a set of anchors-pairs of related terms, which are identified with the help of string-based techniques (edit-distance test), or defined by a user, or another matcher computing linguistic similarity, for example [49]. Then the algorithm refines them by analyzing the paths of the input ontologies limited by the anchors in order to determine terms frequently appearing in similar positions on similar paths. Finally, based on the frequencies and a user feedback, the algorithm determines matching candidates.

Anchor-PROMPT falls into the alignments as solutions and alignments as likeness clues categories.

S-Match. S-Match [32–34] is a schema-based matching system. It takes two graph-like structures (e.g., XML schemas or ontologies) and returns semantic relations (e.g., equivalence, subsumption) between the nodes of the graphs that correspond semantically to each other. The relations are determined by analyzing the *meaning* (concepts, not labels) which is codified in the elements and the structures of schemas/ontologies. In particular, labels at nodes, written in natural language, are translated into propositional formulas which explicitly codify the label's intended meaning. This allows for a translation of the matching problem into a propositional unsatisfiability problem, which can then be efficiently resolved using (sound and complete) state of the art propositional satisfiability deciders. S-Match was designed and developed as a platform for semantic matching, namely a highly modular system with the core of computing semantic relations where single components can be plugged, unplugged or suitably customized. It is a hybrid system with a composition at the element level. At present, S-Match libraries contain 13 element-level matchers, see [35], and 3 structure-level matchers (e.g., SAT4J [9]). S-Match falls into the alignments as theorems category.

Table 1. Characteristics of state of the art matching approaches

	Element-level		**Structure-level**	
	Syntactic	**External**	**Syntactic**	**Semantic**
SF [50]	string-based (2); data types; key properties	-	iterative fix-point computation	-
Artemis [13]	domain compatibility; language-based (1)	common thesaurus (CT): synonyms, broader terms, related terms	matching of neighbors via CT	-
Cupid [46]	string-based (2); language-based (2); data types; key properties	auxiliary thesauri: synonyms, hypernyms, abbreviations	tree matching weighted by leaves	-
COMA [21]	string-based (4); language-based (1); data types	auxiliary thesauri: synonyms, hypernyms, abbreviations; alignment reuse (2)	DAG (tree) matching with a bias towards leaf or children structures (2); paths	-
NOM [26] **FOAM/QOM** [25]	string-based (1); domains and ranges	application-specific vocabulary	matching of neighbors (2); taxonomic structure (4)	-
Anchor-PROMPT [57]	string-based (1); domains and ranges	-	bounded paths matching (arbitrary links); bounded paths matching (processing *is-a* links separately)	-
OLA [30]	string-based (3); language-based (1); data types	WordNet(1)	iterative fix-point computation; matching of neighbors; taxonomic structure	-
S-Match [33, 34]	string-based (5); language-based (3);	WordNet: sense-based (2), gloss-based (6)	-	propositional SAT (2)

Table 1 summarizes how the matching systems cover the solution space in terms of the proposed classification. Numbers in brackets specify how many matchers of a particular type a system supports. For example, S-Match supports 5 string-based element-level syntactic matchers (prefix, suffix, edit distance, n-gram, and text corpus, see [34]), OLA has one element-level external matcher based on WordNet. Table 1 also testifies that schema/ontology matching research was mainly focused on syntactic and external techniques so far. Semantic techniques have been exploited only by S-Match [33].

Having considered some of the recent schema-based matching systems, it is important to notice that the matching operation typically constitutes only one of the steps towards the ultimate goal of, e.g., schema/ontology integration, web services integration or meta data management. To this end, we would like to mention some existing infrastructures, which use matching as one of its integral components. Some examples are Chimaera [49], OntoMerge [24], Rondo [51], MAFRA [47], Protoplasm [8]. The goal of such infrastructures is to enable a user with a possibility of performing such high-level tasks, e.g., given a product request expressed in terms of the catalog $C1$, return the products satisfying the request from the marketplaces $MP1$ and $MP2$. Moreover, use matching component $M5$, and translate instances by using component $T3$.

7 Conclusions

This paper presents a new classification of schema-based matching approaches, which improves the previous work on the topic. We have introduced new criteria which are based on (i) general properties of matching techniques, i.e., we distinguish between approximate and exact techniques; (ii) interpretation of input information, i.e., we distinguish between syntactic, external, and semantic techniques at element- and structure-level; and (iii) the kind of input information, i.e., we distinguish between terminological, structural, and semantic techniques. We have reviewed some of the recent schema/ontology matching systems in light of the classification proposed pointing which part of the solution space they cover. Analysis of state of the art systems discussed has shown, that most of them exploit only syntactic and external techniques, following the input interpretation classification; or terminological and structural techniques, following the kind of input classification; and only one uses semantic techniques, following both classifications. However, the category of semantic techniques was identified only recently as a part of the solution space; its methods provide sound and complete results, and, hence it represents a promising area for the future investigations.

The proposed classification provides a common conceptual basis, and, hence, can be used for comparing (analytically) different existing schema/ontology matching systems as well as for designing a new one, taking advantages of state of the art solutions. As the paper shows, the solution space is quite large and there exists a variety of matching techniques. In some cases it is difficult to draw conclusions from the classifications of systems. A complementary approach is to compare matching systems experimentally, with the help of benchmarks which measure the quality of the alignment (e.g., computing precision, recall, overall indicators) and the performance of systems (e.g., measuring execution time, main memory indicators). We have started working on such an approach and we have found useful our classifications for designing systematic benchmarks, e.g., by discarding features (one by one) from schemas/ontologies with respect to the classifications we had (namely, what class of basic techniques deals with what feature), see for preliminary results the I3CON initiative[2], Ontology Alignment Contest[3] [69], and Ontology Alignment Evaluation Initiative[4].

Future work proceeds in at least three directions. The first direction aims at taking into account some novel matching approaches which exploit schema-level information, e.g., [1, 14, 38, 45, 54]. As it has already been mentioned, in some applications (e.g., agents communication, web services integration) there are no instances given beforehand, and therefore, schema-based matching is an only solution for such cases. However, in the other applications (e.g., schema/ontology integration), instances are given beforehand, and therefore, instance-based approaches should be considered as a part of the solution space. Thus, the second direction of the future work aims at extending our classification by taking into account instance-based approaches, e.g., [17, 23, 40]. Finally, the third direction aims at conducting an extensive evaluation of matching systems by systematic benchmarks and by case studies on the industrial size problems.

Acknowledgements: This work has been partly supported by the Knowledge Web European network of excellence (IST-2004-507482).

References

1. Z. Aleksovski, W. ten Kate, and F. van Harmelen. Semantic coordination: a new approximation method and its application in the music domain. In *Proceedings of the Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC)*, 2004.
2. D. Aumüller, H. H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. In *Proceedings of the International Conference on Management of Data (SIGMOD), Software Demonstration*, 2005.
3. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
4. H. W. Beck, S. K. Gala, and S. B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 572–581, 1989.
5. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, (28(1)):54–59, 1999.
6. J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 108–122, 2001.
7. J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 452–466, 2002.
8. P. Bernstein, S. Melnik, M. Petropoulos, and C. Quix. Industrial-strength schema matching. *SIGMOD Record*, (33(4)):38–43, 2004.
9. D. Le Berre. A satisfiability library for Java. http://www.sat4j.org/.
10. A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. CLASSIC: A structural data model for objects. *SIGMOD Record*, 18(2):58–67, 1989.

[2] http://www.atl.external.lmco.com/projects/ontology/i3con.html
[3] http://oaei.inrialpes.fr/2004/Contest/
[4] http://oaei.inrialpes.fr/2005/

11. P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou, and S. Tessaris. D2.2.1: Specification of a common framework for characterizing alignment. Technical report, NoE Knowledge Web project delivable, 2004. http://knowledgeweb.semanticweb.org/.
12. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: A new approach and an application. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 130–145, 2003.
13. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, (13(2)):277–297, 2001.
14. S. Castano, A. Ferrara, S. Montanelli, and G. Racca. Semantic information interoperability in open networked systems. In *Proceedings of the International Conference on Semantics of a Networked World (ICSNW), in cooperation with ACM SIGMOD*, pages 215–230, 2004.
15. Meta Data Coalition. Open information model, version 1.0. http://mdcinfo/oim/oim10.html, August 1999.
16. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *Proceedings of the workshop on Data Cleaning and Object Consolidation at the International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.
17. R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 383–394, 2004.
18. T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. In *Proceedings of the World Wide Web Conference (WWW)*, pages 321–330, 2003.
19. R. Dieng and S. Hug. Comparison of ”personal ontologies” represented through conceptual graphs. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 341–345, 1998.
20. H. H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proceedings of the workshop on Web and Databases*, 2002.
21. H. H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 610–621, 2001.
22. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 509–520, 2001.
23. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 662–673, 2003.
24. D. Dou, D. McDermott, and P. Qi. Ontology translation on the Semantic Web. *Journal on Data Semantics (JoDS)*, II:35–57, 2005.
25. M. Ehrig and S. Staab. QOM: Quick ontology mapping. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 683–697, 2004.
26. M. Ehrig and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of the European Semantic Web Symposium (ESWS)*, pages 76–91, 2004.
27. J. Euzenat. An API for ontology alignment. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 698–712, 2004.
28. J. Euzenat, J. Barrasa, P. Bouquet, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. van Acker, I. Zaihrayeu, and T. L. Bach. D2.2.3: State of the art on ontology alignment. Technical report, NoE Knowledge Web project delivable, 2004. http://knowledgeweb.semanticweb.org/.

29. J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *Proceedings of the Semantic Integration workshop at the International Semantic Web Conference (ISWC)*, 2003.
30. J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 333–337, 2004.
31. A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening WordNet with DOLCE. *AI Magazine*, (24(3)):13–24, 2003.
32. F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review Journal (KER)*, (18(3)):265–280, 2003.
33. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Proceedings of the European Semantic Web Symposium (ESWS)*, pages 61–75, 2004.
34. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Semantic schema matching. Technical Report DIT-05-014, University of Trento, 2005.
35. F. Giunchiglia and M. Yatskevich. Element level semantic matching. In *Proceedings of Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC)*, 2004.
36. F. Giunchiglia and I. Zaihrayeu. Making peer databases interact - a vision for an architecture supporting data coordination. In *Proceedings of the International workshop on Cooperative Information Agents (CIA)*, pages 18–35, 2002.
37. N. Guarino. The role of ontologies for the Semantic Web (and beyond). Technical report, Laboratory for Applied Ontology, Institute for Cognitive Sciences and Technology (ISTC-CNR), 2004.
38. B. He and K. C.-C. Chang. A holistic paradigm for large scale schema matching. *SIGMOD Record*, 33(4):20–25, 2004.
39. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review Journal (KER)*, (18(1)):1–31, 2003.
40. J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 205–216, 2003.
41. V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The International Journal on Very Large Data Bases (VLDB)*, 5(4):276–304, 1996.
42. J. A. Larson, S. B. Navathe, and R. Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.
43. M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.
44. W. S. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 1–12, 1994.
45. J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 57–68, 2005.
46. J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 49–58, 2001.
47. A. Maedche, B. Motik, N. Silva, and R. Volz. MAFRA - A MApping FRAmework for Distributed Ontologies. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 235–250, 2002.
48. A. Maedche and S. Staab. Measuring similarity between ontologies. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 251–263, 2002.

49. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 483–493, 2000.
50. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 117–128, 2002.
51. S. Melnik, E. Rahm, and P. Bernstein. Rondo: A programming platform for generic model management. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 193–204, 2003.
52. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 14–25, 1996.
53. A. G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, (38(11)):39–41, 1995.
54. P. Mitra, N. Noy, and A. Jaiswal. OMEN: A probabilistic ontology mapping tool. In *Proceedings of the Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC)*, 2004.
55. I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*, pages 2–9, 2001.
56. N. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 2002.
57. N. Noy and M. Musen. Anchor-PROMPT: using non-local context for semantic matching. In *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 63–70, 2001.
58. L. Palopoli, G. Terracina, and D. Ursino. The system DIKE: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *ADBIS-DASFAA, Matfyzpress*, pages 108–117, 2000.
59. M. Paolucci, T. Kawmura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 333–347, 2002.
60. C. Parent and S. Spaccapietra. Database integration: the key to data interoperability. In M. P. Papazoglou, S. Spaccapietra, and Z. Tari, editors, *Advances in Object-Oriented Data Modeling*. The MIT Press, 2000.
61. R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, (19(1)):17–30, 1989.
62. E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *The International Journal on Very Large Data Bases (VLDB)*, (10(4)):334–350, 2001.
63. E. Rahm, H. H. Do, and S. Maßmann. Matching large XML schemas. *SIGMOD Record*, 33(4):26–31, 2004.
64. P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 448–453, 1995.
65. D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 39–52, 2002.
66. A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
67. P. Shvaiko. A classification of schema-based matching approaches. In *Proceedings of the Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC)*, 2004.

68. A. Sotnykova, C. Vangenot, N. Cullot, N. Bennacer, and M.-A. Aufaure. Semantic mappings in description logics for spatio-temporal database schema integration. *Journal on Data Semantics (JoDS), Special Issue on Semantic-based Geographical Information Systems*, III, 2005.
69. Y. Sure, O. Corcho, J. Euzenat, and T. Hughes. *Evaluation of Ontology-based Tools*. Proceedings of the International Workshop on Evaluation of Ontology-based Tools (EON), 2004. http://CEUR-WS.org/Vol-128/.
70. M. Uschold and M. Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.
71. P. Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1, 1999.
72. P. Valtchev and J. Euzenat. Dissimilarity measure for collections of objects and values. *Lecture Notes in Computer Science*, 1280:259–272, 1997.
73. R. van Eijk, F. de Boer, W. van de Hoek, and J. J. Meyer. On dynamically generated ontology translators in agent communication. *International Journal of Intelligent System*, 16:587–607, 2001.
74. Y. Velegrakis, R. J. Miller, and J. Mylopoulos. Representing and querying data transformations. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 81–92, 2005.
75. H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 108–117, 2001.
76. L. Xu and D. W. Embley. Using domain ontologies to discover direct and indirect matches for schema elements. In *Proceedings of the Semantic Integration workshop at the International Semantic Web Conference (ISWC)*, 2003.
77. K. Zhang and D. Shasha. Approximate tree pattern matching. In A. Apostolico and Z. Galil, editors, *Pattern matching in strings, trees, and arrays*, pages 341–371. Oxford University, 1997.