"University of Turku Technical Reports, No. 4 — August 2015"

CURRENT STATE OF ONTOLOGY MATCHING

A Survey of Ontology and Schema Matching

Lauri Mukkala | Jukka Arvo | Teijo Lehtonen | Timo Knuutila

Lauri Mukkala,
University of Turku, Technology Research Center,
20014 Turun yliopisto, Finland,
lauri.mukkala@utu.fi

Jukka Arvo,
University of Turku, Technology Research Center,
20014 Turun yliopisto, Finland,
jukka.arvo@utu.fi

Teijo Lehtonen,
University of Turku, Technology Research Center,
20014 Turun yliopisto, Finland,
teijo.lehtonen@utu.fi

Timo Knuutila,
University of Turku, Technology Research Center,
20014 Turun yliopisto, Finland,
timo.knuutila@utu.fi

www.trc.utu.fi

ISSN: 2341-8028 | ISBN: 978-951-29-6191-7

Technology Research Center

Turun yliopisto
University of Turku

Abstract

Ontology matching is an important task when data from multiple data sources is integrated. Problems of ontology matching have been studied widely in the research literature and many different solutions and approaches have been proposed also in commercial software tools. In this survey, well-known approaches of ontology matching, and its subtype schema matching, are reviewed and compared. The aim of this report is to summarize the knowledge about the state-of-the-art solutions from the research literature, discuss how the methods work on different application domains, and analyze pros and cons of different open source and academic tools in the commercial world.

Keywords

Ontology matching, schema matching

Contents

1 Introduction

Modern society depends on the access to a wide range of information. However, a direct data access is not usually sufficient to make the data usable because different data sources contain heterogeneous data. Wide variety of data formats creates an interoperability problem that needs to be solved in order to enable practical data utilization. The interoperability problems caused by distributed data sources are well-known, but not all of the known issues have been solved comprehensively. This paper examines ontology matching[1], which are common tasks to integrate data from multiple heterogeneous sources.

A *Schema* is a definition that formally represents construction of data structures, defines data constraints, and states valid data structures and values [1]. XML schema and database schema are examples of schemas. Schemas contain elements that are called *schema elements*. One schema element typically describes one column in database. When data sources are combined, schema elements are used by a matching task. *Ontologies* enable unambiguous identification of elements in opaque heterogeneous data sources and relationships between these elements. Ontology matching is the process of determining correspondences between concepts. Ontology matching tools have generally been developed to operate on data source schemas. A subset of ontology matching using schemas can also be called *schema matching* [2, 3]. The remaining text of this survey uses ontology matching as an umbrella term and schema matching when matching is done particularly with schemas.

Ontology matching is a basic, but critical task of generating correspondence mappings for semantically equal elements between ontologies [4]. Ontology matching is typically used when data is physically or virtually combined from multiple heterogeneous and distributed data sources. Typically in data source integrations, data structure schemas are matched pair wisely by using a target schema or intermediate schemas that are combined at the end. Ontology matching is an important step in data source integration tools, data warehouses and reporting systems, which all typically rely on multiple data sources.

Currently, ontology matching requires manual user assistance via graphical user interfaces, which can be seen as the easiest way to perform ontology matching when the application only uses a couple of data sources. However, when the number of data sources increase, the task becomes time consuming. A diverse range of approaches that try to automate the ontology matching have been proposed to alleviate the issues of manual work requirements. Most currently available matching tools try to find semantic, structural or linguistic correspondences between schema elements of the data sources, but there is no single method that has been proven to work in all application domains.

1 http://www.ontologymatching.org/

During the past few decades, many diverse approaches and methods to ontology matching have been proposed in e.g. [5, 6, 7, 8, 9, 10, 11]. Literature reviews of schema matching ([4, 12, 13]) and ontology matching ([14, 2]) has also been published on multiple occasions in the past. A taxonomy used to group matchers in this paper is combined from other surveys to enable easier comparisons between this survey and the previous taxonomies. The query phrase used was:

"ontology matching" OR "schema matching" OR "ontology alignment"

All of the literature queries were conducted in February 2015 using ACM[2], IEEExplore[3] and CiteSeerX[4]. Other sources (e.g. Google Scholar[5]) were also used when it was necessery to find more information from specific subject.

This survey introduces general properties of current matching techniques, updates the knowledge about the state-of-the-art methods, and makes new discoveries. The paper analyzes pros and cons of different methods and discusses what applications different methods have in the commercial world. In section 2 different approaches to ontology matching are introduced. Section 3 reviews a wide range of state-of-the-art matcher and section 4 compares and analyzes approaches reviewed in section 3 and presents conclusions.

2 https://dl.acm.org/
3 http://ieeexplore.ieee.org/Xplore/home.jsp
4 http://citeseerx.ist.psu.edu/
5 https://scholar.google.fi/

2 Approaches and methods to ontology matching

Many different solutions have been proposed to ontology matching, because ontology matching is a critical and time consuming task. Nevertheless, all of the problems of ontology matching have not been solved and there is no single method that works well in all situations. Therefore, most of the developed matchers are combinations of multiple approaches and are hence called complex matchers. Complex matcher can be further divided into two groups, *hybrid* and *composite matchers*, depending on how the results of elementary matchers are combined. If the elementary matchers are used one after another in a sequence, the matcher is called hybrid matcher. However, if the elementary matchers are used parallel and their results are combined at the end, the matcher is called composite matcher. Hybrid approaches and composite approaches both use the contribution of each elementary matcher. Each elementary matcher in a complex matcher typically returns relatedness value, called semantic distance, which is used to calculate a weighted sum. By comparing the weighted sums to predefined threshold values, the matches of elements are determined. The threshold values can usually be adjusted by the user.

When methods for data integration projects are selected, input and output processing methods need to be considered [4]. Different schema and ontology matching methods vary greatly on what they can take as an input. Some methods, for example, only work with database models, such as ER models (Entity-Relationship model, used to describe data and its relations), while some algorithms only support XML files. The input given to the schema matcher is an important aspect of schema and ontology matching process, because it determines what kind of structures can be used by the matching algorithms. When schema or ontology matching methods are chosen by their processing properties, one important question is whether exact or approximated matches are required. The difference between the exact and approximated method is that the approximated method stops after one match is found, whereas the exact method tries to find all matching elements. In some cases, when processing speed is important, approximated matches can be sufficient. However, as mapping is usually done once per set of schemas or ontologies, more accurate matches can be produced because processing time is not that critical. Depending on the use case, it has to be considered if a matching method can use external aids such as different thesauri. The thesauri data base size or the need for Internet connection can cause limitations.

Schema and ontology matching approaches can be divided into two main groups as shown in figure 1. The first group are methods that compute matches in element-level and the second group are methods that compute matches in structure-level [13].

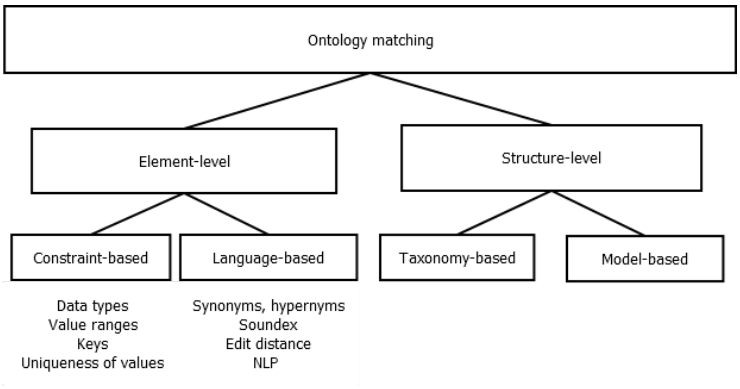

Figure 1: Taxonomy of ontology and schema matching.

In the *element-level*, matches are computed by determining which elements match between two schemas or ontologies. As examples, column's name, description field's value, type similarity, or linguistic feature can be used for elemental-level matching. *Structure-level* matching focuses on combinations of elements that usually appear together in the schema or in the ontology. However, structure-level approaches can also use hierarchies to compute matches. In this survey, methods are first divided into element-level and structure-level and then into smaller groups for finer granularity taxonomy.

2.1 Element-level approaches

In element-level approaches, schemas or ontologies are matched by analyzing meaning and content of the elements. Because matching is done by ignoring relations of elements with each other, element-level approaches are especially useful when matching schemas or ontologies that are opaque or data about the schemas or ontologies is incomplete [4]. The two main ways to implement element-level approaches are language-based matching and constraint-based matching.

2.1.1 Language-based matching

Language-based matching, sometimes called linguistic matching, uses names and text properties to find semantical matches between schemas or ontologies. The simplest form of language-based matching is to use element names or descriptions. If the names or descriptions of two elements match fully or partially (e.g. CName→CustomerName), the elements are usually semantically equal. Advanced methods use synonyms, hypernyms or similarity algorithms such as Soundex ([15]) to analyze how words sound. Edit distances ([16]), which calculate how many edits are needed to make two words equal or natural language processing techniques ([9, 8]) can also be used to improve matching. Additional methods that can be used in ontology matching can be found from approximate string matching papers (e.g. [16]).

Using synonyms (words that mean exactly or nearly the same, e.g. dog and hound) and hypernyms (is-a relations, e.g. dog is an animal) can be an efficient way to solve simple matching problems, but these methods require the usage of thesauri or dictionaries which can be a problem when Internet connection is not

available. Words that are equal, but have different meanings are called homonyms. The homonyms are a problem because they create false matches. The issue of false matches can be alleviated by using supplementary methods such as structure-level approaches or constraint-based matching. Problematic cases also appear when multiple match candidates are found. As an example, *address* matches for both *home address* and *business address.* The problems of multiple matches are easy to solve with manual interaction from a user, but hard to solve automatically. One solution to automatic multi matching is to use supervised learning to teach recognition of true matches (e.g. [5]).

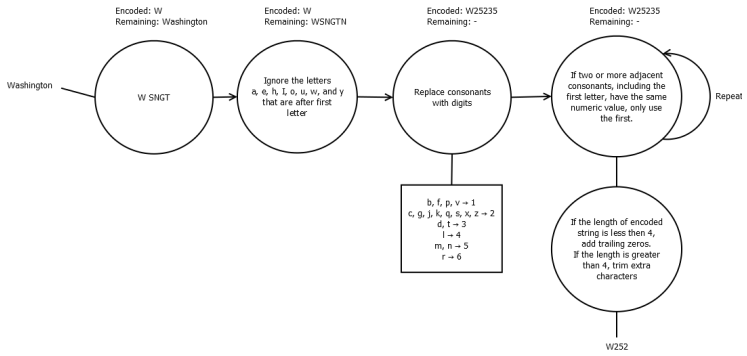

Figure 2: An example of Soudex algorithm's process

A more advanced way to use language-based matching is to use Soundex or edit distance. *Soundex* is a phonetic algorithm that uses sounds, as pronounced in English, to encode words into one letter and three numbers. Soundex algorithm works in three steps (figure 2). The first step is to take the first letter and remove all vocals and the letters h and w. The second step is to replace the remaining consonants with the corresponding values. The third and final step is to trim the encoded string iteratively until it is one character and three numbers long. Trailing zeros are added if necessary. The Soundex algorithm can be used as a supplementary method, because the method is very error-prone. It has been reported that only about 30% of matches are correct with the Soundex method [15]. Due to unreliability, many improvements over the original Soundex method have been introduced in subsequent phonetic algorithms.

It is often necessary to use additional techniques from natural language processing when language-based matching is used because basic forms of words appear infrequently. The most popular operations of natural language processing techniques are tokenization, lemmatization, and elimination [4]. In tokenization, words are split into tokens by adding punctuations and blank characters to words. The tokens can then be matched. Lemmatization is used to find basic forms of words to avoid problems caused by words that are in plural form. Elimination is a technique that removes articles, prepositions, conjunctions or otherwise useless parts of data elements.

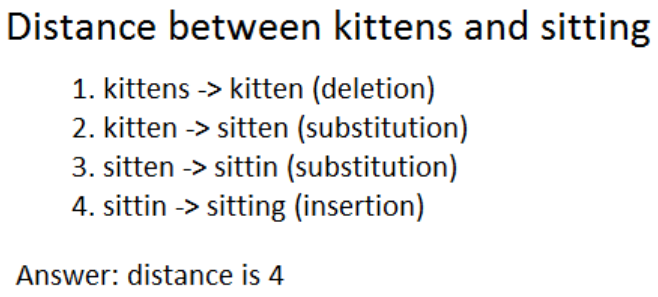

Figure 3: Example of Levenshtein distance.

Edit distances is another useful language-based matching method [16]. One popular algorithm that uses edit distances is the *Levenshtein distance* algorithm which measures the difference between two strings. The difference is calculated by determining the minimum number of single-character edits (insertions, deletions and substitutions) that are required to transform two words equal [16]. However, methods that use edit distances have problems to handle short words. As an example the word *cat* has edit distance of 3 to *dog*, but the words are not likely that match with each other. Another example is shown in figure 3 where the edit distance between *kittens* and *sitting* is calculated. First the algorithm uses deletion to remove *s* from *kittens*. Secondly, substitution is used to replace first *k* with *s* and then *e* with *i*. Thirdly, insertion is used to insert *g* at the end. It requires four modification to transform *kittens* into *sitting* which means that the edit distance between the two is four.. Both the Soundex method and edit distances perform poorly alone, but are beneficial as supplementary methods when using structural-level approaches.

2.1.2 Constraint-based matching

Constraint-based matching uses internal structures of schemas and ontologies to compute matches. The internal structures can be definitions of elements (data types), uniqueness of attributes (also known as cardinality), and foreign or primary keys. The constraint-based matching can be used to narrow down the list of possible matches in situations where schemas or ontologies are so opaque that the utilization of direct approaches are difficult. The use of constraint-based approach typically leads to multiple matches and therefore these methods should not be used alone. However, the constraint-based matching methods are useful when they are used together with other algorithms. If multiple possible match candidates are found, the correct match can be determined by examining value ranges or data types.

Data types and range comparison of values are one way to match schemas and ontologies by using constraint-based matching. If multiple match candidates are found, the amount of candidates can be reduced by checking data types. If the datatype of one element is integer while the datatype of the second is char, the two elements are not likely to be a match. However, small variations in data types can be misleading. If two matched elements have different precisions, ranges, or lengths, constraint-based matching might not find the match. The problem can be partially solved by using statistical analysis. By calculating means or other statistical measures, matching elements can be found regardless of small variances.

2.2 Structure-level approaches

Structure-level approaches match elements by finding similarities in structure-level [17, 18]. While element-level approaches ignore relations and only focuses on the actual elements, methods using structure-level approaches try to find relationships by analyzing relations that the single elements and the element groups have. One popular method is to analyze parent-children relationships (e.g. [8]). After potential matching structures have been found by using element-level algorithms, the matches can be confirmed with linguistic matching, constraint-based matching, or other supplementary techniques. The most popular choices to implement structure-level approaches are taxonomy-based algorithms, repositories of structures and model-based algorithms [4].

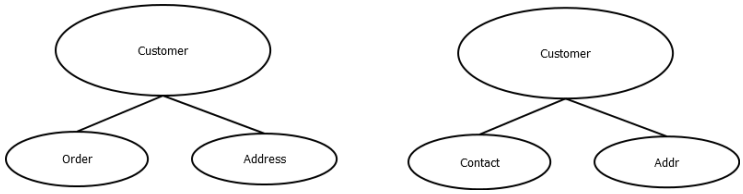

Figure 4: Two simple examples of two schemas. Database tables are presented as circles.

Taxonomy-based algorithms, also known as graph-based algorithms, use specialization relations of the elements. The first step is to construct taxonomy graphs by using hierarchical *is-a* links. After that similar elements are identified from the graph by comparing positions along paths. If two elements have similar parents, they are identified as possible matches. The possible matches can be further analyzed with supplementary techniques. For example, in figure 4, *address* and *addr* are identified as a possible match because both elements have a parent called *customer*.

Model-based algorithms focus on semantic interpretations of schemas and ontologies and try to solve the matching problem with formal semantics [8]. One commonly used reasoning technique is Boolean satisfiability that is also known as propositional satisfiability or SAT (abbreviation of the word satisfiability). In the Boolean satisfiability method, graphs are translated into a propositional formula that typically has form: Axioms $\rightarrow$ rel(context$_1$, context$_2$). The propositional formula is a statement expression that results Boolean values, which are used in matching. The problem with Boolean satisfiability method is that the algorithm cannot handle binary predicates due to the limitations of propositional language used for codifying matching problems [4].

Repositories of structures can be used to improve the quality of matching results [19]. The repository structure stores previously computed matches. When new matches are identified, the matching results are first compared to stored matches in the repository. The stored matches do not necessarily have to be exact. In most cases it is sufficient that structures are partially matching because data sources have differences and complete matches are rare. By using the repository, the number of false matches can be reduced and the matching results can be made more accurate. To ensure computational efficiency, repository structures should not scan the whole repository to identify matches. Suggested alternatives to accelerate finding potential matches from the repository are to use names of root elements, number of nodes, or

maximal path length [4].

3 Academic and open source tools and algorithms

In this section different academic and open source tools and algorithms are reviewed. Reviewed tools were chosen from other surveys and newest articles. Criteria was that a chosen tool had to be referenced in at least two latest articles to be relevant enough. Tools in this section are divided into two groups: those that are ontology matching tools and those that are purely schema matching tools.

3.1 Ontology matching tools and algorithms

AgreementMaker[1] is an ontology-based matching tool that has an extensible and modular architecture [11]. The tool has a multi-purpose user interface which helps to evaluate matching results. The AgreementMaker tool is meant to be a generic solution to a wide range of application domains. The built-in algorithms work in two steps. The first step computes similarity scores while the second step aligns elements based on the results from the first step. As an example, labels, comments, instances and structures are used for the computation.

Anchor-PROMPT [20] is an extension of PROMPT [21], also known as SMART. Anchor-PROMPT is an ontology merging and matching tool that combines multiple matching methods to one hybrid matcher. The method inputs two ontologies in graph form and computes matches by using a set of linguistic-based methods such as edit distance [20]. The algorithm also refines matches by analyzing similar positions on the graph and appearance frequencies.

Automatic Semantic Matching of Ontologies with Verification (ASMOV) is an automatic ontology-based matching tool [22]. The tool target is to integrate bioinformatical data and the algorithm works in two steps. The first step calculates similarity scores and the second step verifies the matches that are computed in the first step. The first step uses multiple language-based (e.g. Levenshtein distance) and structure-based methods (e.g. weighted sum of the range similarities). Multiple similarity scores are computed and summed together by using weighted average. ASMOV is able to use several thesauri depending on the application domain.

Falcon[2] is an automatic ontology-based matching tool that uses divide-and-conquer approach [23]. The Falcon tool is designed for matching large ontologies that consists of thousands of entities. The matching algorithm that the Falcon tool uses has three phases. The first phase is to partition ontologies into small clusters

1 http://agreementmaker.org/

2 http://ws.nju.edu.cn/falcon-ao/

based on structures. The second phase is to match blocks that are constructed from the clusters and the last phase is to discover matches. Falcon uses a greedy selection algorithm to discovering matches.

Naïve Ontology Mapping (NOM) [24] and its successor **Quick Ontology Mapping (QOM)** [25] are matching systems that combine multiple elementary matchers. The NOM and QOM tools have similarities to the COMA tool as they all offer a platform for matching tools with extendable libraries. NOM and QOM work on ontology-level by using sibling concepts (elements that are considered siblings) and sub-concepts (parent-child relations).

Owl Lite Aligner (OLA)[3] uses all ontology components including classes, properties, names and constraints [26]. OLA is based on a family of distance based algorithms which analyze the distances between two elements of the input structures. The OLA algorithm iterates until no improvements are produced. Elements that have short distances are considered as matches. The OLA solution considers the matching problem as an optimization problem.

3.2 Schema matching tools and algorithms

Artemis is a module designed to be used with data source mediator system called MOMIS [27]. The basic idea behind the Artemis module is to calculate affinity values for the name and structure using a thesaurus. Affinity based methods analyze elements and define affinity values for the elements. The affinity value identifies semantic relationships between the elements. Artemis calculates affinity from names and structures and combines the affinity values into global affinity coefficient. After that, hierarchical clustering produces affinity tree where match candidates are then selected. In the last phase elements belonging to a candidate cluster are unified into a global view.

Clio is a semi-automatic schema matcher developed under IBM Research [28]. The used matching algorithm works in two phases [15]. In the first phase suitable schema reader is used to read a schema and to translate it into an internal representation. After that a correspondence algorithm is used to identify elements that match together. Clio is designed to be modular which allows adding and removing plug-ins to alter schema matching results. The Clio tool has also a graphical user interface to correct and augment correspondences.

Cupid is a mapping tool that discovers mappings between schema elements based on their names, data types, constraints, and schema structures [9]. Cupid matcher has been developed to be a general-purpose solution that tries to combine the pros of different matchers. The Cupid tool uses automated linguistic-based matching, works in both element-level and structure-level, uses internal structures, and exploits keys and constraints. The matching process works in three phases. The first phase is called linguistic matching where names and data types are matched with the help of thesaurus. The second phase is called structural matching. In the structural matching phase elements are matched based on their similarity. In third phase, final alignments are chosen.

COMA (Combination of Matching Algorithms)[4] is a schema matching system platform that does schema matching by reusing results from previous match operations and by combining results from multiple matchers [10, 19, 29]. The

3 http://ola.gforge.inria.fr/

4 http://dbs.uni-leipzig.de/Research/coma.html

COMA tool is distributed with a library of matching algorithms, a framework for combining results, and a platform for evaluating the effectiveness of the different matchers. COMA supports multiple schema types such as XML schemas and relational schemas, and the tool can be extended by including libraries. The COMA system represents schemas as directed acyclic graphs where elements are represented by graph nodes. The match operations take two schemas as input and output the matched pairs and corresponding similarity values. If the results contain multiple matches, the final matching choice can be selected automatically or it can be left for user to decide.

Duplicate-based Matching of Schemas (DUMAS)[5] is an instance-based matching tool that compares names of the elements, properties of the underlying data instances, and the structure in which the elements are contained [30]. Usually instance-based methods are said to be working vertically, meaning that elements are analyzed and matched individually column vise. DUMAS, however, works horizontally by trying to find duplicate rows or tuples. Once few duplicates have been found, matching elements are deduced from matching values.

Line Segment Detector (LSD) is a multi-strategy matching tool that uses machine-learning techniques to find schema mappings semi-automatically [31, 6]. The system trains itself from semantic mappings of training sets that the user generates from data sources. The user input and the data sources are then used to train a set of learners. LSD uses multiple learners, because each learner uses different types of methods to focus on schemas or the element data. LSD uses XML as source data format and DTD (Document Type Definition, used to define document types) as schema data format. In LSD, schema matching problem is defined as a problem of matching the target schema and the source schemas. It has been reported that LSD can obtain accuracy of 71-92% [31].

The functionality of the LSD system consists of two phases. The first phase is training where user manually specifies the initial element mappings. Here, data is extracted and the training examples are created to the learners. The output from the first phase is the internal classification models for learners. In the second phase, which is the actual matching, data from the source is extracted, learners are applied, and constraint handler is used to output the target schema. If the mapping results do not satisfy the user, feedback can be given and a new set of training mappings is produced for the LSD algorithm.

OpenII Harmony[6] is a schema matching tool that combines multiple language-based matching algorithms and a graphical user interface [32]. Processed schemas are first translated into a canonical graph representation and then linguistic pre-processing is applied (e.g., tokenization, stop-word removal, and stemming) before match computation starts. Multiple matching algorithms are used to vote whether the two elements are matching. The OpenII Harmony tool is designed to be used with large schemas and it can efficiently handle schemas of 10^4 elements. OpenII Harmony differs from other language-based tools as it can also analyze textual documentation of the elements. Quality of the matching results can be increased with the help of human input via graphical interface.

SemInt (Semantic Integrator) is a tool for identifying attribute correspondences in heterogeneous databases using neural networks [5, 33, 34]. The SemInt method

5 http://hpi.de/naumann/projects/repeatability/algorithms/dumas-duplicate-based-matching-of-schemas.html

6 http://openii.sourceforge.net/index.php?act=tools&page=harmony

matches schema elements by using a classifier to categorize the elements according to descriptions and values, and then using a neural network to do the actual matching. SemInt does not use pre-programmed knowledge and the knowledge is learned from meta-data of schemas that makes the method more adjustable to new situations. The data clusters used to train the neural network are discovered by using self-organizing map ([35]) and statistical analysis to categorize elements based on mean of fields, variance, coefficient of variation, or groupings of numeric fields. After the elements have been classified in to clusters, back-propagation learning algorithms are used to train the neural network to determine similar matching cases in the future.

Similarity Flooding is a versatile graph-based matching algorithm [7]. The algorithm takes two graphs as input and maps corresponding nodes of the graphs as output. The algorithm is based on the assumption that if any of the two elements in given models are identified as similar, then the similarity of their adjacent elements should be increased. Algorithm manipulates Open Information Model[7] (OIM) specification with an iterative fixed-point computation which means that from iteration to iteration the depth is increased and a similarity measure is computed until the fix-point is reached. Similarity flooding method uses a language-based method that compares prefixes and suffixes of the elements to compute initial matching. After the initial mapping algorithm is computed, iterative phase is started where similarities are compared. As the result, an alignment between the nodes of input graphs is produced. The similarity flooding algorithm is semi-automatic as it expects the user to verify and adjust the results

S-Match[8] is a semantic matching algorithm that uses a treelike structure to match nodes [8, 36]. Matches are determined by analyzing meanings of elements rather than the labels. The meanings are typically codified into propositional formulas from names of elements. The S-Match tool is designed to be highly modular hybrid matcher where elementary matcher can be easily configured depending on the use case. The used matching methods are mainly based on element-level approach, but some structure-level plug-ins are contained in the S-Match library.

7 http://sqlmag.com/database-administration/open-information-model

8 http://semanticmatching.org/s-match.html

4 Discussion

Ontology matching is a critical task in many application domains that allows combining data from multiple heterogeneous data sources. In this survey different approaches, methods, tools and algorithms for ontology matching have been surveyed. In this section pros and cons of different approaches in the most typical application domains are discussed. Also future research directions are presented.

Name	Approaches	Input types	Available online
Anchor-PROMPT	Ontology-based		No
AgreementMaker	Ontology-based	XML, RDFS, OWL, N3	Yes
Artemis	Language-based	Relational, OO, ER	No
ASMOV	Ontology-based	OWL	No
Clio	Semantic translation	Relational and semi-structured (e.g. XML and DTD)	No
COMA	Element-level, structure-level	XML, relational	Yes
Cupid	Element-level, structure-level	XML, relational	No
DUMAS	Element-level, duplicate-based	CSV files	Yes
Falcon	Ontology-based	RDFS, OWL	Yes
LSD	Element-level, structure-level	XML	No
NOM and QOM	Ontology-based	RDF, OWL models	No
OLA	Ontology-based		Yes
OpenII Harmony	Language-based	XML Schema, SQL DDL, OWL, Excel spreadsheets	Yes
SemInt	Constraint-based	Relational, files	No
Similarity Flooding	Language-based, graph-based	Directed labeled graphs	No
S-Match	Language-based	Tree-like structures, e.g., XML	Yes

Table 1: Summary of the reviewed matching tools. The table contains information of used matching approaches, accepted input types and whether an implementation of the tool is publicly available.

All of the reviewed matching tools are summarized in the table 4.1. It became clear that many of the academically developed matching tools are no longer active as most of them were developed over ten years ago. However, the algorithms presented in the early academic matching tools have contributed to the development of the next generation matching tools that are more dynamic and can be extended with

plug-ins.

The lack of product-like commercial matching tools can be due to the fact that companies who combine multiple data sources develop own matching tools in-house or use outsourced developer. Companies specialized in data integrations currently develop case-specific solutions which are time consuming. However, as most of the projects are billed based on the used working hours, the used time is not an issue for the data integration company, but customers project costs increase. Commercially available data integration tools were not studied in this survey as in-depth information about the tools was not available.

Even though the problem of implementing efficient ontology matching is difficult to solve, a wide range of solutions has been presented during the past couple of decades. In the past, the ontology matching problem was solved with application specific approaches, but nowaday more generic and dynamic methods are being developed.

The two most obvious application domains that need ontology matching are data warehouses and E-commerce. The need for data integrations first appeared when data warehouses became popular in the 1990s. Data warehouses are repositories where data from several sources is combined, stored, and typically revision controlled for better usability. Once all the data is stored in the same server, the need for multiple connections becomes obsolete and data access times become shorter. Ontology matching is useful in the combination phase of building data warehouses as matched data can be described in various ways. Semantics of the elements is important when data is integrated into data warehouses, because source data often uses different data types and need to be combined into one representation. In addition, structure-level approaches and methods using constraints and data types are useful with data warehouses as data type transformations are easy to implement in these methods. Structure-level approaches seem to be most effective because even though data elements may have different naming conventions, the data is most likely structured similarly.

E-commerce is more recent application area of ontology matching than data warehouses. During the last ten years the number of messages traveling between companies has increased exponentially. Nowadays companies frequently send and receive information describing, e.g., transactions from trading partners and subcontractors. Used syntax and data formats vary greatly between different data systems and companies. The most used data types are EDI (electronic data interchange) and XML. However, custom message structures are also quite popular, which increases the need for automatic converters. The number of elements in one message can be so large that manual message matching takes a lot of time. The amount of different data structures and how to normalize and harmonize messages together are the two main challenges in creating automatic matching tools for E-commerce. Tools for this kind of matching would require a graphical user interface as manual modifications are inevitable. The best approach for E-commerce seems to be to use element-level matching of structures where individual elements may have variations but the field names of the structures are typically similar.

Most of the studied tools used multiple element-level matching approaches and language-based matching methods in one form or another. For the future work we will most likely focus on structure-level approaches. The usage of structure repositories combined with neural networks and learning algorithms is also an interesting development direction as it is less studied.

Acknowledgements

The research has been carried out during the MARIN2 project (Mobile Mixed Reality Applications for Professional Use) funded by Tekes (The Finnish Funding Agency for Innovation) in collaboration with partners; Defour, Destia, Granlund, Infrakit, Integration House, Lloyd's Register, Nextfour Group, Meyer Turku, BuildingSMART Finland, Machine Technology Center Turku and Turku Science Park. The authors are from Technology Research Center, University of Turku, Finland.

Bibliography

[1] Hector Garcia-Molina. *Database systems: the complete book*. Pearson Education India, 2008.

[2] Pavel Shvaiko and Jérôme Euzenat. "Ontology matching: state of the art and future challenges". In: *Knowledge and Data Engineering, IEEE Transactions on* 25.1 (2013), pp. 158–176.

[3] Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and Alma Gómez-Rodríguez. "Ontology matching: A literature review". In: *Expert Systems with Applications* 42.2 (2015), pp. 949–971.

[4] Pavel Shvaiko and Jérôme Euzenat. "A survey of schema-based matching approaches". In: *Journal on Data Semantics IV*. Springer, 2005, pp. 146–171.

[5] Wen-Syan Li and Chris Clifton. "Semantic integration in heterogeneous databases using neural networks". In: *VLDB*. Vol. 94. 1994, pp. 12–15.

[6] AnHai Doan, Pedro Domingos, and Alon Y Levy. "Learning Source Description for Data Integration." In: *WebDB (Informal Proceedings)*. 2000, pp. 81–86.

[7] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. "Similarity flooding: A versatile graph matching algorithm and its application to schema matching". In: *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE. 2002, pp. 117–128.

[8] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. *S-Match: an algorithm and an implementation of semantic matching*. Springer, 2004.

[9] Jayant Madhavan, Philip A Bernstein, and Erhard Rahm. "Generic schema matching with cupid". In: *VLDB*. Vol. 1. 2001, pp. 49–58.

[10] Hong-Hai Do and Erhard Rahm. "COMA: a system for flexible combination of schema matching approaches". In: *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment. 2002, pp. 610–621.

[11] Isabel F Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. "AgreementMaker: efficient matching for large real-world schemas and ontologies". In: *Proceedings of the VLDB Endowment* 2.2 (2009), pp. 1586–1589.

[12] Philip A Bernstein, Jayant Madhavan, and Erhard Rahm. "Generic schema matching, ten years later". In: *Proceedings of the VLDB Endowment* 4.11 (2011), pp. 695–701.

[13] Erhard Rahm and Philip A Bernstein. "A survey of approaches to automatic schema matching". In: *the VLDB Journal* 10.4 (2001), pp. 334–350.

[14] Holger Wache et al. "Ontology-based integration of information-a survey of existing approaches". In: *IJCAI-01 workshop: ontologies and information sharing*. Vol. 2001. Citeseer. 2001, pp. 108–117.

[15] JY Mortimer and JA Salathiel. "'Soundex'codes of surnames provide confidentiality and accuracy in a national HIV database." In: *Communicable disease report. CDR review* 5.12 (1995), R183–6.

[16] Patrick AV Hall and Geoff R Dowling. "Approximate string matching". In: *ACM computing surveys (CSUR)* 12.4 (1980), pp. 381–402.

[17] Balázs Villányi and Péter Martinek. "Towards a novel approach of structural schema matching". In: *Computational Intelligence and Informatics (CINTI), 2012 IEEE 13th International Symposium on*. IEEE. 2012, pp. 103–107.

[18] Yuan Yang, Mengdong Chen, and Bin Gao. "An effective content-based schema matching algorithm". In: *Future Information Technology and Management Engineering, 2008. FITME'08. International Seminar on*. IEEE. 2008, pp. 7–11.

[19] Hong-Hai Do and Erhard Rahm. "Matching large schemas: Approaches and evaluation". In: *Information Systems* 32.6 (2007), pp. 857–885.

[20] Natalya F Noy and Mark A 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)*. 2001, pp. 63–70.

[21] Natalya Fridman Noy and Mark A Musen. "Algorithm and tool for automated ontology merging and alignment". In: *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831*. 2000.

[22] Yves R Jean-Mary, E Patrick Shironoshita, and Mansur R Kabuka. "Ontology matching with semantic verification". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 7.3 (2009), pp. 235–251.

[23] Wei Hu, Yuzhong Qu, and Gong Cheng. "Matching large ontologies: A divide-and-conquer approach". In: *Data & Knowledge Engineering* 67.1 (2008), pp. 140–160.

[24] Marc Ehrig and York Sure. "Ontology mapping–an integrated approach". In: *The Semantic Web: Research and Applications*. Springer, 2004, pp. 76–91.

[25] Marc Ehrig and Steffen Staab. "QOM–quick ontology mapping". In: *The Semantic Web–ISWC 2004*. Springer, 2004, pp. 683–697.

[26] Jérôme Euzenat, Petko Valtchev, et al. "Similarity-based ontology alignment in OWL-lite". In: *ECAI*. Vol. 16. 2004, p. 333.

[27] Sonia Bergamaschi, Silvana Castano, and Maurizio Vincini. "Semantic integration of semistructured and structured data sources". In: *ACM Sigmod Record* 28.1 (1999), pp. 54–59.

[28] Mauricio A Hernández et al. "Creating nested mappings with Clio". In: *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE. 2007, pp. 1487–1488.

[29] David Aumueller et al. "Schema and ontology matching with COMA++". In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM. 2005, pp. 906–908.

[30] Alexander Bilke and Felix Naumann. "Schema matching using duplicates". In: *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. IEEE. 2005, pp. 69–80.

[31] AnHai Doan, Pedro Domingos, and Alon Y Halevy. "Reconciling schemas of disparate data sources: A machine-learning approach". In: *ACM Sigmod Record*. Vol. 30. 2. ACM. 2001, pp. 509–520.

[32] Len Seligman et al. "Openii: an open source information integration toolkit". In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM. 2010, pp. 1057–1060.

[33] Wen-Syan Li and Chris Clifton. "SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks". In: *Data & Knowledge Engineering* 33.1 (2000), pp. 49–84.

[34] Wen-Syan Li, Chris Clifton, and Shu-Yao Liu. "Database integration using neural networks: Implementation and experiences". In: *Knowledge and Information Systems* 2.1 (2000), pp. 73–96.

[35] Teuvo Kohonen. *Self-organizing maps*. Vol. 30. Springer Science & Business Media, 2001.

[36] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. "Semantic schema matching". In: *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*. Springer, 2005, pp. 347–365.