DELFT UNIVERSITY OF TECHNOLOGY

MASTERS THESIS

Reproducing state-of-the-art schema matching algorithms

Author:
Andra-Denis IONESCU

Supervisor:
Assistant Prof.dr. C. Lofi

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Web Information Systems Group
Software Technology

February 3, 2020

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

Electrical Engineering, Mathematics and Computer Science
Software Technology

Master of Science

Reproducing state-of-the-art schema matching algorithms

Author: Andra-Denis IONESCU
Student id: 4722485
Email: A.D.Ionescu@student.tudelft.nl

Schema matching has been a researched topic for over 20 years. Therefore, many schema matching solutions have been proposed to treat various problems such as: creating unified knowledge bases or mediation schema, data translation, data discovery, data curation. Such a wide variety of schema matching algorithms requires a benchmarking system that can evaluate to what extent one solution is appropriate for a given problem. However, creating the benchmark requires open source algorithms, which are not widely available in the data management community. One solution to this problem is reproducing the algorithms, although there is a reproducibility crisis which proves that the majority of existing research can not be reproduced. These circumstances have determined the goal of this research: conducting a reproducibility study on the state-of-the-art schema matching algorithms. This study supports the schema matching development and emphasizes the issues regarding the ability to reproduce the algorithms or the results. Moreover, we implement the selected algorithms and benchmark them in an industry case study.

Thesis Committee:

Chair:	Prof.dr.ir. G.J.P.M. Houben, TU Delft
University supervisor:	Assistant Prof.dr. C. Lofi, TU Delft
Committee Member:	Prof.dr. A. van Deursen, TU Delft
Committee Member:	Assistant Prof.dr. A. Katsifodimos, TU Delft

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We live in the digital era where content is produced every day [Dong and Srivastava, 2013] due to the rapid expansion of technologies and the high accessibility of data in multiple domains such as social media, marketing, online shops, banking, enterprise and even government [Clifton et al., 2004]. Nowadays, producing information is a natural process and similarly, consuming it. Moreover, a significant amount of disciplines require data in order to generate insights, statistics, and reports. The real challenge begins when the necessary information to deliver such work is located in different sources, as missing data can severely impact the final outcome. Having data in multiple sources constitutes a problem, because one does not know which attributes represent the same concept in order to extract the proper data. Moreover, even if all the sources have been connected, data integration is a very laborious process requiring great knowledge about data.

For many years, integrating data has been done manually by experts in the field [Rahm and Bernstein, 2001]. However, this task has become increasingly difficult, because the volume of data has been rapidly growing [Palopoli, Sacca, and Ursino, 1998]. The solution for integrating multiple sources has been researched for many years and the core task is named **schema matching**. Schema matching has been proposed as a solution for more than two decades. As such, we can notice the emergence of developing systems and methodologies that aim for an automated solution where the user's input is as minimal as possible.

Two decades of development of schema matching systems have led to a wide variety of approaches, some treating specific domains and scenarios [Cortez et al., 2015, Mitra, Wiederhold, and Jannink, 1999, Doan, Domingos, and Levy, 2000], while some are considered generic and can be applied in any domain and any type of data [Castano and De Antonellis, 2001, Palopoli, Sacca, and Ursino, 1998].

Given the diversity of domains tackled by schema matching systems, choosing the proper algorithm for a specific scenario has become a challenging task. Therefore, to facilitate the task of choosing the appropriate algorithm we propose creating a *benchmark* that can indicate which algorithm performs the best under certain circumstances. However, creating such benchmark requires open source algorithms that can be evaluated and compared under different conditions. Currently, the data management community lacks open source schema matching algorithms. As such, creating a benchmark requires reproducing the algorithms.

The literature indicates that the evaluation of schema matching solutions is either performed by comparing the reported results in the papers [Bernstein, Madhavan, and Rahm, 2011], or by running the binaries corresponding to the algorithms that are provided upon request [Madhavan, Bernstein, and Rahm, 2001]. However, the majority of research does not mention the provenance of the algorithms. Thus, no study reports attempts to reproduce the work of other authors before comparing the results.

Investigating this deficiency, we have discovered that the issue of reproducibility has become a concern in various scientific domains. In 2016, a survey on 1576 researchers reported that 70% of them failed to reproduce other scientist's research, while 50% failed to reproduce their own study [Baker, 2016]. Moreover, the research is facing a reproducibility crisis in many domains [Baker, 2016]. The health sciences address the issue of reproducibility as a lack of source code, environment variables, code packaging procedures, and data [Peng, 2011]. To solve the issue, ACM SIGMOD Reproducibility[1] has been trying to develop the culture of sharing the necessary materials and it incentives the authors by awarding the most reproducible papers.

Under these circumstances, we formulate the **goal of this thesis**: analyzing the current state of schema matching research classified according to the problems tackled and proposed solutions. Furthermore, we assess the degree to which the algorithms are reproducible. More precisely, our **contributions** are the following:

1. developing an extensive literature study on schema matching with the focus on the problems and the methodology employed to solve them;
2. creating a comprehensive taxonomy of schema matching techniques based on the data elements used and the proposed matching algorithms;
3. conducting a reproducibility study of the most representative schema matching papers;
4. evaluating the algorithms proposed in the representative papers on synthetic datasets
5. benchmarking the algorithms in an industry use case.

Developing an extensive literature study helps us understand what schema matching is, what problems it solves and how. The literature survey includes two decades of schema matching development and it shows the problem spectrum classified into three large categories that are influenced by the time and popularity of the database solutions in that period.

Consequently, the literature survey constitutes the basis for constructing the taxonomy, which details all the solutions proposed in the literature survey. The variety of solutions is classified according to the data attributes: either schema level, instance-level or both. Moreover, the taxonomy presents how the user influences the algorithms and what are the major categories of algorithms employed by each schema matching solution. Furthermore, the taxonomy helps us identify the high-level schema matching categories for which we select one representative paper. The papers will undergo a reproducibility study that will determine the state of schema matching literature and the degree of implementation.

Finally, concluding the reproducibility study, the selected papers will be evaluated on synthetic data and real-life data. The results will determine how well the papers can be reproduced and how well they perform in real-life scenarios, where data integration represents a daily struggle.

The next chapters address each of the contributions: in Chapter 2 we present the literature survey, in Chapter 3 we construct and detail the taxonomy, in Chapter 4 we discuss the concept of reproducibility, perform an analysis over the papers and present the implementation design decision. In Chapter 5 we evaluate the algorithms and present the results. Finally, we conclude the research and propose future improvements in Chapter 6.

[1] http://db-reproducibility.seas.harvard.edu

Chapter 2

Literature review

2.1 What is schema matching?

Schema *represents the definition of a relational database. It is a representation of the information from an application or user view. A schema contains information about the tables, columns, data types and constraints.* [Batini, Lenzerini, and Navathe, 1986].

Schema matching *represents the process of identifying correspondences between the elements of one or multiple schemas* [Madhavan et al., 2005].

Schema matching consists of a collection of *match* operations that consume different types of data and outputs the correspondences between elements. The match operation can be a simple similarity measure or it can be expressed by complex algorithms that involve machine learning, deep learning, graph operations or a combination of them. The data consumed by the algorithms are accessible in multiple formats, such as XML, DTD, and relational data, either as schema information, data instances or both. The **scope of the thesis** consists of having one or more match operations that consume only relational data and its elements, such as column and table names, data types, data instances. Finally, the output of a schema matching algorithm is a list of pairs containing the correspondences between elements. To illustrate this, Figure 2.1 depicts a minimal example of the input and output of a schema matching algorithm.

Considering the above definition, this chapter presents a literature review of the schema matching techniques, classified according to the problem they solve. The goal of the literature review is to assess the current state of schema matching research: the problems solved, the solution proposed and how the data is used.

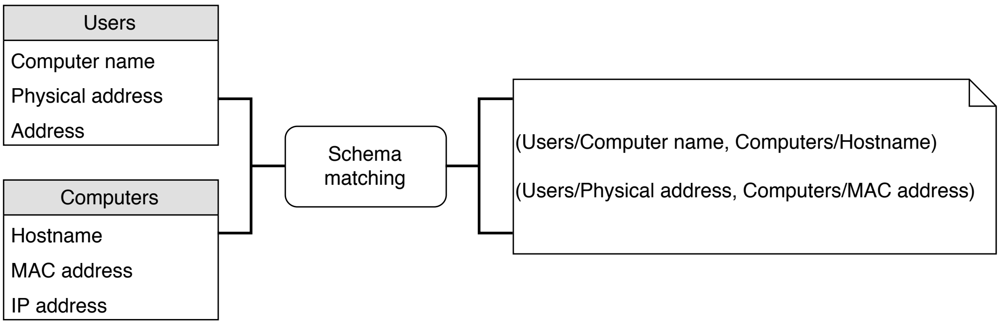


FIGURE 2.1: The figure illustrates two tables *Users* and *Computers* that represent the input for a schema matching algorithm. The output produced by the algorithm is produced in the left part as a list of pairs.

2.2 Paper selection

The starting point of the literature review is the survey of Rahm and Bernstein, 2001, which presents six early research papers on schema matching. We further used the survey to find other state-of-the-art papers by inspecting the works that cited the survey. Therefore, we used Google Scholar[1], which produced initially 4306 results. We filtered the results by searching for the words *schema matching* contained in the title, keywords or the body of the papers, which reduced the initial number of results to 3870. We further proceeded with the filtering and chose only the papers from the most reputable conferences such as: the International Conference on Very Large Data Bases (VLDB), Association for Computing Machinery's Special Interest Group on Management of Data (ACM SIGMOD), International Conference on Data Engineering (ICDE), International Conference on Extending Database Technology (EDBT), Conference on Innovative Data Systems Research (CIDR), Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining (ACM SIGKDD), Association for Computing Machinery's Transactions on Database Systems (ACM TODS) and Transactions on Knowledge and Data Engineering (TKDE).

Table 2.1 contains the number of search results for each conference and the number of selected papers. The paper selection followed a set of criteria devised as follows:

- number of citations
  As the study from Rahm and Thor, 2005 indicates, SIGMOD registered the highest average number of citations for a period of four years. The average approached 100 citations and as a result of the fact that we investigate the research from early ages, we considered appropriate to eliminate all the papers that have less than 100 citations, except for the papers published one year ago.

- domain membership
  The papers that met the above criterion were manually inspected, as they had to relate to the scope of the thesis. Many papers were describing the general field of data integration, with a few mentions to the schema matching or they were describing other domains where schema matching can be applied. However, the scope of the literature review is to investigate the state-of-the-art for schema matching and how the algorithms and methodologies described utilizing the data, therefore we discarded all the unrelated research.

- ability to consume the same type of dataset
  The scope of the thesis as mentioned in the first section (Section 2.1) consists of having match operations that consume only relational data. Therefore, each algorithm has to consume the same type of dataset. Thus, ontology mapping represents one of the discarded topics, although schema matching can be employed to solve the issue. Similarly, many studies were focused on web query matching using web tables or other data mined from the deep web, which does not constitute the scope of the thesis.

Finally, the total number of state-of-the-art studies on schema matching that conforms to the aforementioned criteria has been reduced to 20.

[1] https://scholar.google.com/

TABLE 2.1: The table indicates the conferences used to filter the search results from Google Scholar, the number of search results for each conference, the number of papers that have more than 100 citations and the number of selected papers after the manual inspection. The last row in the table shows the total number of papers for each column.

Conference	Search results	> 100 citations	Selected papers
ACM SIGMOD	67	26	5
VLDB	92	24	4
ICDE	12	4	4
CIDR	7	5	1
SIGKDD	4	1	0
EDBT	20	0	0
ACM TODS	7	1	0
TKDE	23	4	0
Number of papers mentioned in the survey			6
Total	232	65	20

2.3 Findings

As the starting point of the literature review is the survey from 2001 used in the paper selection (Section 2.2), the schema matching journey starts from the early beginning, even before the year 2000 (Figure 2.2). Examining the literature from 1994 until 2018, we observed that the problems solved by schema matching can be categorized into three groups as follows:

- **Multiple databases era** - The period of multiple databases starts in 1994 and ends in 2001 in our literature review. In this time frame, schema matching is employed to solve problems that concern multiple databases such as federated databases, creating unified knowledge bases [Heimbigner and McLeod, 1985] and introducing the data integration problem together with data translation [Halevy, 2001].

- **General approach** - The period between 2001 and 2013 introduces the general approach of schema matching. The researchers are not concerned with one type of problem anymore and they believe they should focus on the general view and develop performing algorithms independently of the problem.

- **Big data era** - From 2013, big data started to gain more and more popularity[2]. Thus, schema matching development adapted and it tries to solve the problems concerning data discovery and data curation.

Following this classification, the next sections present the findings of the selected literature in chronological order.

[2] https://gumroad.com/l/LaUj

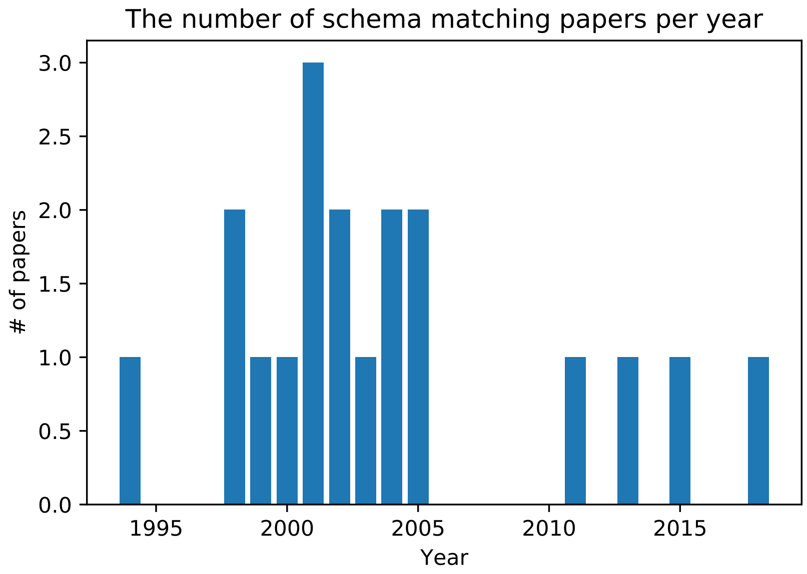


FIGURE 2.2: The figure illustrates the distribution of papers according to the publishing year.

2.3.1 Multiple databases era

In 1994, the *SemInt* framework [Li and Clifton, 1994] was proposed to solve the problem of **developing federated databases**[Sheth and Larson, 1990]. The approach used the meta-data information and data instances from the schema level and instance level and aspired to solve the problem in a semi-automated manner. The match operation consists of examining the structure information to classify it according to the data types and constraints and discovering patterns in data instances by applying string-based algorithms and computing statistics (average, variance, coefficient of variation) on numerical data. The algorithm uses machine learning to understand the patterns in attributes, while the output is transferred to a neural network in order to determine the mappings. Furthermore, the framework reuses the results as an external knowledge representation for future mappings. Although the processing is automated, the system is semi-automated because it allows user intervention to provide thresholds and confirm the results.

Four years later, in 1998, *TranScm* [Milo and Zohar, 1998] is developed to solve the problem of **data translation** by producing a data representation that can be easily integrated with the conventional translation languages. The system consumes data from the schema level and uses graphs to represent it. The match operation uses the graphs to find matches by applying different rules that manipulate the data using string-based approaches or constraints. The system is semi-automated and it involves the user as a mediator in the cases where a component has multiple matches and the system can not decide between them and to provide rules.

Another semi-automated framework developed in 1998 wants to solve the problem of **constructing a global dictionary** to support the integration of federated data systems. *Dike* [Palopoli, Sacca, and Ursino, 1998] uses information from the schema level and performs a clustering operation on the data before other computations are employed. The match operation consists of transforming the schemas from each cluster into a high-level abstraction until a sufficiently abstract model results. The match is supported by the user who provides match candidates as examples in the initial phases of the algorithm. Moreover, the user is also involved in the validation and verification phase required after the semantic operations such as synonymy, inclusion or homonymy. Finally, the system represents a semi-automatic manner due to user intervention.

In 1999, another semi-automated framework is developed to tackle the problem of **creating a unified knowledge base** from multiple independent heterogeneous sources. *SKAT* [Mitra, Wiederhold, and Jannink, 1999] uses ontologies and schema information to find matches and processes the elements using a language-based and string-based approach. The framework relies heavily on the input from an expert even from early phases and its functionality is similar to an active learning system. Thus, the expert provides examples of matches and mismatches between elements of ontologies and accepts or rejects the new system generated matches. Finally, the system updates itself and the cycle is reloaded.

In 2000, the *LSD* [Doan, Domingos, and Levy, 2000] framework was proposed to support the problem of **data integration**. It consumes data from schema level represented as a tree and data instances. The schema level data is used as high-level training examples manually provided by the user, while the data instances are used to train several machine learning classifiers. Once the classifiers are trained, they are used to output confidence scores on each new source schema. Next, a meta learner is used to combine all the confidence scores and returns a prediction list for each schema element. Finally, a prediction combiner assigns the correct label based on heuristics.

In 2001, the *ARTEMIS* [Castano and De Antonellis, 2001] system is proposed as a solution for **constructing global views** [Castano and De Antonellis, 1999], a problem also tackled in 1998 with the Dike framework [Palopoli, Sacca, and Ursino, 1998]. Another similarity between the two frameworks is the type of data used, as both use schema-level elements. Moreover, both approaches use clustering in their implementation, but with different outcomes. Unlike Dike, which uses clustering before other computations, ARTEMIS uses hierarchical clustering after a preprocessing phase based on the string and language-based approaches. Similar to the other semi-automated frameworks, ARTEMIS needs user intervention for identical reasons such as setting thresholds and validating results.

Another framework developed in 2001, addresses the problem of **data integration and translation**. The data integration component of the *Clio project* [Miller et al., 2001, Popa et al., 2002] uses schema information to perform attribute classification. Moreover, the framework uses external knowledge such as dictionaries and thesauri and the user intervention as a verification step for the generated matches. The rest of the framework is concerned with the operations necessary for data translation.

2.3.2 General approach

Generic schema matching algorithms are introduced in 2001, starting with the *Cupid* framework [Madhavan, Bernstein, and Rahm, 2001] that offers a **general approach** to a variety of problems that involve schema matching. It uses the structure information from schema-level as well as the elements and it consumes the data in the form of a tree structure after applying multiple language and string-based operations. Finally, the outcome results after applying structural matching on the tree, which is highly dependent on constraints. The framework relies heavily on the existence of a thesaurus for the linguistic matching and the human input does not differ from the rest of the frameworks, as the user has to specify different parameters or initial mappings.

Another example of generic algorithms is *COMA* [Do and Rahm, 2002], developed in 2002. Similar to *Cupid*, COMA uses schema level elements represented internally as a direct acyclic graph. The match is performed using either simple (string, linguistic, constraint or semantic-based), hybrid (string, constraints or graph-based) or reuse-oriented (using existing matches) algorithms. The user is heavily involved in the process by selecting matchers or matching strategies, as well as providing feedback on the resulted matches. Moreover, the framework represents a suitable environment for testing different algorithms and evaluating their individual or combined performances.

The third generic schema matching algorithm is also developed in 2002, under the name of *Similarity flooding* [Melnik, Garcia-Molina, and Rahm, 2002] and, similar to the other generic frameworks, it uses schema level information represented as a graph. Unlike *Cupid* that applies linguistic operations on the graph structure, *Similarity flooding* applies string-based operations. The unique characteristic of the framework is represented by the matching algorithm that spreads the similarity from the two corresponding nodes to the adjacent nodes. Applied over multiple iterations, the similarity is propagated into the graph. Finally, using filtering heuristics, a subset of the mappings is extracted and inspected by a human expert and the "accuracy" of the algorithm is measured by counting the number of user interventions.

In 2003, a new approach for schema matching is proposed, named *data interpretation* [Kang and Naughton, 2003]. Although the algorithm uses instance-based data, it explores the uninterpreted element and structure matching techniques. The motivation behind the methodology is the existence of hidden dependencies between data that helps to match the columns with opaque names. The algorithm first performs a pair-wise comparison between all the attributes and constructs a dependency graph, which is processed using graph matching algorithms.

Although ontology mapping does not constitute the research topic of the study, *S-Match* [Giunchiglia, Shvaiko, and Yatskevich, 2004] computes semantic matches exclusively between trees, thus it transforms the schemas in hierarchies or ontologies. The underlying algorithm uses only schema level information that suffers linguistic-based transformations. The match is accomplished by computing the relations between labels based on string approaches using external thesauri and computing the relations between nodes using *propositional satisfiability* [Serafini et al., 2003]. One remarkable feature of the framework is the lack of any user interaction in any phase of the algorithm, unlike the rest of the frameworks that have been discussed so far.

Besides the development of a framework where the user intervention is not required, the year 2004 also introduces the first framework that solves **complex matches** such as *{address - street, city}* and not only 1:1 correspondences (*{surname - lastname}*). *iMap* [Dhamankar et al., 2004] uses both schema level and instance level information. The match is performed as a search operation in the entire space of the match candidates and is based not only on the data type (text, numeric) but also on the semantics of the data. To restrict the number of searches or prune the match candidates, the framework exploits external information to enrich explainability. Additional to the external information, the framework offers the possibility of adding different search algorithms and match candidates initially provided by the user. Finally, the resulted mappings are the outcome of machine learning and statistical algorithms and they are manually inspected by the user in the process of validation and verification.

Moreover, in 2005, a framework [Madhavan et al., 2005] that can solve a variety of schema matching problems can only leverage the schema information from only one particular domain at a time. The framework is based on an improved version of LSD [Doan, Domingos, and Halevy, 2001] and it inherits the majority of the matching operations. However, there are several differences between the two, such as: using different approaches to generate training data, as it constructs a corpus of schemas and mappings used to infer the matches between two other new schemas. Moreover, it uses improved algorithms to train the data, using the corpus to derive statistics that would be used as constraints and an updated algorithm to generate the matches based on heuristics.

In the same year, another approach to schema matching is developed. DUMAS [Bilke and Naumann, 2005] relies entirely on the data instances to find matches using duplicate detection algorithms. Moreover, it is the first paper that classifies the matching strategies using the terminology *horizontal and vertical matching*. The strategy of the underlying algorithm uses the vertical match as it inspects the rows and uses the values to find duplicates, by exploiting a collection of fuzzy matchers, such as string-based similarity measures, combined using the composite strategy. To improve performance, the data might suffer language-based transformations, such as removing the stopwords and applying stemming. The algorithm does not require human intervention and the mappings can be extracted as a subset of any K values from the generated set.

In 2011, a purely instance-based framework [Zhang et al., 2011] for schema matching is developed. The framework does not require external information such as ontologies or thesauri and its implementation relies on data distributions, computed using the similarity metric introduced in Zhang et al., 2010, named Earth Mover's Distance. It uses the metric as an underlying methodology for clustering the columns, it transforms the clusters into complete graphs and it applies an integer program to solve the graph and output the correlated attributes. As the majority of the frameworks, it requires the user input for setting certain thresholds.

2.3.3 Big data era

In 2013, as **part of a data curation framework**, the matching component of *Data Tamer* [Stonebraker et al., 2013] uses information from web sites that is transformed into class entities using different means of extraction. Besides this type of data, it also uses external ontologies and thesauri to augment the initial data. The match is performed on schema level and instance level data and it applies different algorithms and consolidates the results using a composite matcher. The algorithms include semantic and syntactic procedures such as string-based, language-based and statistical operations. The user has a similar function as he previously had in the other frameworks presented and he needs to provide thresholds, matching examples and validate the matches generated by the system.

In 2015, the schema matching matter shifts the focus from data representation and similarity functions to the usage of external information. Therefore, *Barcelos* [Cortez et al., 2015] solved the **data discovery** problem by proposing a method of assisting the matching algorithms with **information from within the organisation** instead of external knowledge bases (ontologies, thesauri). The framework examines and mines the repositories containing spreadsheets from within the organization because they include information carefully generated using different meaningful queries on the data. Therefore, the columns from the spreadsheets are mapping candidates for the columns of the databases. The match is realized using a heuristic based on Jaccard containment [Chaudhuri, Ganti, and Kaushik, 2006].

Using different external knowledge has been a preferred method to infer the semantics, thus *SemProp* [Fernandez et al., 2018b] part of the *Aurum*[Fernandez et al., 2018a] framework, that solved the problem of **data discovery in enterprises**, introduces *coherent groups*, a new structure to semantically represent a group of words. The framework uses both schema level and instance level data and follows the approach involving both syntactic and semantic matching by employing string and language-based operations on data. The match is constructed as a pipeline of operations that involve external ontologies and thesauri used to find semantic matches using the coherent groups. Additionally, it uses syntactic operations such as Jaccard similarity. Finally, the user represents an important component involved in the process to provide values for different thresholds.

TABLE 2.2: The table indicates the problems identified in the literature, categorized based on the group of problems, and shows the number of papers addressing them.

Category	**Problem Type**	**# of papers addressing the problem**
General approach	-	6
Multiple databases	Data integration	4
	Data translation	2
	Developing federated databases	2
	Developing global dictionaries	2
	Developing a unified knowledge base	1
Big data era	Data discovery	2
	Data curation	1

2.4 Conclusion

To conclude, this chapter presents the evolution of the schema matching problems, the data types used by the algorithms and the type of match. Moreover, it shows that schema matching has been proposed as a solution to one specific problem from 1994 and gradually evolved to being a generic approach for a wide range of problems since 2001. Therefore, the problems solved by the schema matching algorithms are not specific anymore and the newest algorithms can be employed for any of the following matters: data integration, data warehousing, data discovery, data management systems, federated databases, web-based architectures, web services, message translation, and peer-data management. A summary of the problems identified and the number of occurrences can be observed in Table 2.2. Finally, as the summary [Bernstein, Madhavan, and Rahm, 2011] of schema matching evolution from 2011 indicates, the schema matching research did develop over time and the matter has been analyzed in isolation, as a mature topic, without attaching it to the problem spectrum or its application field.

Chapter 3

Taxonomy

The chapter presents a detailed classification of the schema matching techniques identified in the literature review. The taxonomy is constructed by following the type of data used by the algorithms and the match operation employed. To reiterate, the scope of the thesis consists of one or more match operations that consume only relational data. Nonetheless, relational data can have various formats, that can impact the difficulty of schema matching. The easiest case is represented when the schema structures are the same in terms of attribute names or described concept, but the real challenge occurs when the schemas have similar, but non-identical data, different structures or different notations for the same concept [Madhavan, Bernstein, and Rahm, 2001].

The starting point for constructing the taxonomy is the survey from Rahm and Bernstein, 2001, which outlines and explains the main categories in schema matching methodology. Then the taxonomy is enriched with the findings of Shvaiko and Euzenat, 2005 which extend the schema-based approaches in particular.

Regardless of the approach, the classification starts with the *match* operation, which Rahm and Bernstein, 2001 categorized into two big areas: individual approaches and combined approaches. The individual approaches use a single match criterion to compute the mappings, while the combined ones use multiple match criteria. The most broad category is the one concerning the individual approaches and the following classification results:

Cat. 1 Instance vs schema based

Cat. 2 Element vs structure matching

Cat. 3 Language vs constraint

Cat. 4 Matching cardinality

Cat. 5 Auxiliary information

The first category (**Cat. 1**), *instance vs schema based*, is the one that divides the methodologies into another two big portions, thus being able to further sub-classify the strategies into granular categories.

3.1 Schema-based approaches

Schema-based approaches only use the schema information to find similarities between candidates. It contains details about the data types, names, descriptions, relationships and constraints [Rahm and Bernstein, 2001]. Furthermore, schema-based matching is sub-classified by considering the granularity of match [Rahm and Bernstein, 2001, Shvaiko and Euzenat, 2005] into element level or structure level (**Cat. 2**).

The element level matching computes mappings between individual elements from schema structure, without analysing the relationships between them. On the other hand, the structure level matching analyses how the elements interact and how they are connected inside the schema [Shvaiko and Euzenat, 2005]. Ideally, two schema structures should have total matching, but commonly, only a part of the elements are truly matching [Rahm and Bernstein, 2001].

Before describing the next layer (**Cat. 3**), Shvaiko and Euzenat, 2005 introduce the *input interpretation layer*. This layer emphasizes the distinction between the *internal* and *external* techniques. The internal techniques are further classified into *semantic* and *syntactic* techniques. The semantic approach refers to semantics to interpret the input data, while the syntactic approach analyses the structure of the input data using different algorithms to infer meaningful statistics. The external techniques are the **Cat. 5**, *auxiliary information*, which refer to additional resources that help interpret the input data, such as user input, thesauri, dictionaries, and ontologies. Next, we will extend **Cat. 3** based on the input interpretation layer, taking into consideration the internal and external categories and element vs. structure approaches.

3.1.1 Element-level techniques

As previously mentioned, element level techniques analyze the elements from the schema, such as column names, and they are either syntactic or external. The **syntactic** layer specifies the techniques which manipulate the data as a sequence of characters. The techniques identified in the literature review are the following:

1. String-based
   The approach is used to match string literals or descriptions of schema entities as a sequence of characters. The most used string manipulation methods are: *prefix* (mostly used to identify acronyms and their correspondents), *suffix* (check whether one string ends in the another one, e.g phone and telephone), *n-grams* (computes the number of common sub-strings between two strings) and *edit distance*. The edit distance represents the number of insertions, deletions or substitutions required to transform one string into another one, normalized by the length of the longest string.

2. Language-based
   The language-based approach considers the string literals as actual words in a given language. Therefore, the approach makes use of the Natural Language Processing (NLP) techniques, such as *tokenization* (the words are split into entities by a certain tokenizer that recognizes punctuation, cases, digits, blank spaces), *lemmatization* (the words are morphological analyzed and transformed into a basic form) and *elimination* (the connection words such as prepositions, conjunctions, articles or most frequently used words are discarded).

3. Constraint-based
   The constraint-based techniques are concerned with the internal constraints of the entities, such as *data types* (comparing the entities with respect to their data type) and *multiplicity comparison* (checks whether or not the entity can be collected into structures that have cardinality constraints e.g. the set of five cars is closer to the set of five automobiles than the set of seven watercraft).

The **external** techniques use the auxiliary information to enhance the power of match by interpreting the input as a human can. Following we will shortly describe the methods, while a more concise version can be found in Shvaiko and Euzenat, 2005:

1. Linguistic resources
   From the linguistic resources, the ones that are used to match words based on their linguistic relations (synonyms and hyponyms) are: *common knowledge thesauri* (used to find the meaning of words in the schema by sense of hierarchy, e.g. WordNet) and *domain specific thesauri* (used to find the meaning of words that are specific to a certain domain which does not belong to the common knowledge).

2. Alignment reuse
   The alignment reuse refers to the possibility of reusing previously matched schema structures. The motivation resides on the fact that many schema structures are similar, especially if they are derived from the same domain. The approach suggests that the structure should be decomposed into smaller chunks that will be tested for match against the previously matched structures.

3. Upper level formal ontologies
   The ontologies are external sources, logical-based systems that provide formal specification on top of existing thesauri. Such systems are: Suggested Upper Merged Ontology (SUMO) and Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE).

3.1.2 Structure-level techniques

The structure-level techniques compute the match between entities by analyzing how the entities are interacting together [Rahm and Bernstein, 2001]. Here, the relationship between entities prevails over the individual elements. For this level, besides syntactic and external techniques, the semantics are also present [Shvaiko and Euzenat, 2005].

The **syntactic** approaches are based on or derived from graph techniques. Therefore, the schema is transformed into a graph structure where the mappings are computed using either node levels, relation levels or the entire tree [Shvaiko and Euzenat, 2005]. The approaches identified in the literature are graph matching and tree matching.

1. Graph matching
   Graph matching represents a computation expensive approach. Thus, in schema matching, graph matching is treated as an optimization problem (e.g minimizing distance) for which graph methods are employed. In a Direct Acyclic Graph (DAG), the match is computed considering certain graph elements such as *children*, *leaves* and *relations*. Therefore, the similarity between inner nodes is often computed by examining the similarity between the children nodes, leaf nodes or the relations between them.

2. Tree matching
   A few techniques transform the schemas into trees instead of graphs, to emphasize the fact that each node has one and only parent and to use unique traversal methods. The match is computed by comparing the *leaves*, which also helps to compare the sub-trees.

The structure-level presents the same sub-categories as the element-level. Thus, besides the syntactic approaches, external approaches are also accounted for. Although they are mentioned in the literature [Shvaiko and Euzenat, 2005], the papers described in our literature review do not mention the existence of any external approaches at the structure level and they will not be further detailed.

Furthermore, the structure-level presents one more sub-category: the **semantic** layer. The semantic layer is represented by the *model-based* algorithms such as described in Shvaiko and Euzenat, 2005. The survey presents two structure-level semantic approaches: propositional satisfiability and description logics, but only the former was identified in the literature review.

1. Propositional satisfiability (SAT)
   SAT decomposes a graph into node problems and each possible pair of matching nodes is transformed given the formula: $Axiom \rightarrow rel(context_1, context_2)$. "The formula is valid iff the negation is unsatisfiable".

3.2 Instance-based approaches

The instance-based approaches can enrich the schema structures by providing extra information about the described data. Such data might augment the entities, by examining the instances and computing statistics or performing the match operation on the instances [Rahm and Bernstein, 2001]. If the schema structure is missing, the instances can help reconstruct it, either manually or automatically [Rahm and Bernstein, 2001].

Most of the techniques presented in Section 3.1 are also applicable in the case of instances [Rahm and Bernstein, 2001] such as both element-level techniques: the syntactic and external approaches. The structure-level approaches are not suitable for a high volume of data, because they are used to infer structures that can provide a meaningful characterization of the data. However, data instances do not have any structure, thus the algorithms developed to extract information from the data are essential for finding mappings without any structure.

3.2.1 Element-level approaches

In the case of schema-based techniques, the column names and their constraints are considered elements. On the other hand, in instance-based approaches, every data instance is considered an element. Therefore, the same syntactic techniques can be employed, such as the *string-based and language-based* that are used to extract concepts depended on the frequency of words and *constraints* that are used to constrain the data in certain intervals, ranges or other metrics such as average, maximum or minimum [Rahm and Bernstein, 2001].

Although the volume of data is significantly bigger than the schema-level data, certain approaches need external information to further augment the data. Repeatedly, the same schema-based external approaches are suitable for instance based. Therefore, the *linguistic resources* are used to find synonyms and hypernyms, while *the upper-level formal ontologies* are used to derive the concepts. However, *the alignment reuse* is not suitable for data instances because of the high variation of data.

3.2.2 Similarity measures

Another approach that uses the instance-level data is represented by the *similarity measures*. The similarity measures usually use the data distributions, union or intersection to infer statistics about the data. They do not manipulate it as the string or language methods do and they do not represent external approaches either, because they only use the available data without any additional information. Such approaches are:

1. Earth Mover Distance (EMD) [Zhang et al., 2010] - computes the similarity between data distribution of different columns. The default approach is primarily used for numeric data. However, a few extensions have been elaborated to consume non-numeric data (such as a string) based on histograms.

2. Jaccard similarity or Jaccard coefficient [Chaudhuri, Ganti, and Kaushik, 2006] - the method is based on a formula that uses the number of common words and the total number of words. The number of common words is represented by the set intersection, while the total is computed using the set union.

3.3 Combined approaches

Both schema-based and instance-based methods represent individual matching techniques, successful by themselves but even more powerful when combined. Therefore, a few papers have included in their methodology different combinations of individual matching algorithms to improve the performance. Combining the match operations can be done in one of the two following ways:

1. hybrid matcher - combines different algorithms using multiple match criteria or external resources, such as a string-based match algorithm combined with data type compatibility tables. The advantage of using a hybrid matcher consists of improved performance because it reduces the number of transfers between schemas and improved effectiveness. After all, the poor matches can be filtered out.

2. composite matcher - combines the results generated by multiple individual or hybrid approaches. It is more flexible than the hybrid matcher because it can construct new matchers by reusing the results in any order.

3.4 Discussion

The detailed taxonomy presents a general classification of the schema matching techniques identified in the literature review (Chapter 2). The findings are summarized in Table 3.2 and it presents the methods discussed in the previous sections and the characteristics of each framework. The list of frameworks is chronologically ordered, which in combination with the color codes, allows us to identify the period where a certain type of data was predominant. As the table indicates, in 1994, SemInt experimented with both schema level and instance-level data. However, the following eight years of schema matching development were concerned only with the schema level data. From 2003, the algorithms oscillated between both types and easily adopted the combination of two.

Besides the categories highlighted in Table 3.2, each methodology presents additional features that do not belong to the taxonomy. These features are concerned with the number of user interventions and the means to apply them or the type of algorithms.

In terms of user intervention, a few papers mention that the algorithms do not require any user input or they do not indicate anything about such an aspect. However, the majority (73%) of the literature review describes semi-automatic frameworks thus, the intervention type and the total amount can be observed in Table 3.3. According to the literature, the user can add thresholds, initial mappings, match algorithms or rules, he can be a mediator and finally, he can validate and verify the output. Among the research that mentions how the user interacts with the system, the most encountered actions are: validating the final results, providing the thresholds and initial mappings.

Aside from the user contribution, the research also mentions the type of algorithms used to generate the mappings. These algorithms either use the output of the matchers described in the taxonomy or are completely independent. Such methods are classified as machine learning algorithms, neural networks, diverse statistics based algorithms or different types of clustering. The usage and distribution of the methods among the literature reviewed in Chapter 1 is presented in Table 3.1. Interestingly, the state-of-the-art algorithms do not exploit the neural networks, but they are depending on the traditional methods such as diverse machine learning classifiers or statistics. The only mention of the neural networks occurs from research developed in 1994. Moreover, the majority of the frameworks are focused on implementing only one type of algorithm, except for *SemInt* and *iMap* that combine machine learning algorithms with either neural networks or statistics.

In the next chapter, we will use the taxonomy to select the most varied in approach papers for conducting a reproducibility analysis. Firstly, we will define the terms and describe the status of reproducibility in research and the database community. Following, we will detail the criteria used to select the papers, compute the degree of reproducibility and finally, present the implementation design decision we made to reproduce the studies.

TABLE 3.1: The table presents the main type of algorithms discussed in the literature review and the correspondence with the frameworks

Paper	Machine Learning	Clustering	Neural Networks	Statistics
SemInt	✔		✔	
Dike		✔		
LSD	✔			
ARTEMIS		✔		
Clio	✔			
iMap	✔			✔
Madhavan et al., 2005				✔
Zhang et al., 2011		✔		
Data Tamer				✔
Total	4	3	1	3

TABLE 3.2: The table summarizes the findings in the taxonomy and the characteristics of each research described in the literature survey. The blue checkmarks (✔) represent the *schema level* data, the yellow checkmarks (✔) represent the *instance level* data, while the green checkmarks (✔) represent both types. The black checkmarks (✔) are independent of the data type.

	Language based	String based	Constraint based	Linguistic resources	Alignment reuse	Ontologies	Earth Movers Distance	Jaccard similarity	Graph matching	Tree matching	Propositional Satisfiability	Hybrid	Composite
	Syntactic			External			Similarity		Syntactic		Semantic	Combination	
	Element level								Structure level				
SemInt, 1994		✔	✔		✔							✔	
TranScm, 1998		✔	✔						✔			✔	
Dike, 1998				✔								✔	
SKAT, 1999	✔	✔				✔						✔	
LSD, 2000										✔			✔
ARTEMIS, 2001	✔	✔		✔								✔	
Clio, 2001				✔									
Cupid, 2001	✔	✔	✔	✔						✔		✔	
COMA, 2002	✔	✔	✔		✔				✔			✔	
Similarity Flooding, 2002		✔							✔			✔	
Data interpretation, 2003									✔				
S-Match, 2004	✔	✔		✔						✔	✔	✔	
iMap, 2004			✔	✔								✔	
Madhavan et al., 2005										✔			✔
DUMAS, 2005	✔	✔											✔
Zhang et al., 2011		✔					✔		✔				
Data Tamer, 2013	✔	✔	✔	✔		✔							✔
Barcelos, 2015				✔				✔					
SemProp, 2018	✔	✔		✔		✔		✔					

TABLE 3.3: The table describes the types of user intervention identified in the literature review and classify the frameworks according to the correspondence between them

	Intervention type					
Paper	Add thresholds	Add initial mappings	Add matchers	Validation	Verification	Mediator
SemInt	✔			✔		
TranScm			✔			✔
Dike		✔		✔	✔	
SKAT		✔		✔		
LSD		✔				
ARTEMIS	✔			✔		
Clio					✔	
Cupid	✔	✔				
COMA			✔	✔	✔	
Similarity flooding				✔	✔	
iMap		✔	✔	✔	✔	
Zhang et al., 2011	✔					
Data Tamer	✔	✔		✔		
SemProp	✔					
Total	6	6	3	8	5	1

Chapter 4

Reproducibility study

4.1 Introduction

"Reproducibility refers to the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator"[Goodman, Fanelli, and Ioannidis, 2016]. As part of the thesis, one of the contributions is represented by a reproducibility study. This study aims to assess the reproducibility level of the schema matching research and to investigate how well the state-of-the-art systems described in the literature survey perform, considering the criteria described further in this chapter. Furthermore, the reproduced versions will undergo a benchmark, where possible, to assess the accuracy of the reproduced versions against the golden standard and the reported results in the initial studies.

Another definition of reproducibility is given in the ACM Guidelines[1] and they imply that a different team with a different experimental setup should obtain the same results. Besides *reproducibility*, another term is used interchangeably or to denote *repeatability*. The term *replicability* refers to "the ability of a researcher to duplicate the results of a prior study if the same procedures are followed but new data is collected" [Goodman, Fanelli, and Ioannidis, 2016]. Due to the variety in terminology, a concise description is necessary to establish the differences between the words and formulate the issues faced by the scientific community. Goodman, Fanelli, and Ioannidis, 2016 presented the differences between reproducibility and repeatability and the characteristics of each are classified under three categories: *methods reproducibility*, *results reproducibility* and *inferential reproducibility*.

Methods reproducibility suggests that all the details and steps of the study are available to develop the same procedure or system. The system should return the same results using the information described in the initial study using the same data if is publicly available or provided by the authors upon request [Collberg, Proebsting, and Warren, 2015].

Results reproducibility is described as the *repeatability* term and it indicates that the same results should be accomplished during an independent study that follows very similar procedures to the original study.

Finally, one of the categories that is rarely mentioned is *inferential reproducibility*. It represents the scientific research that draws the same conclusions from a replicated study or from "a reanalysis of the original study" [Goodman, Fanelli, and Ioannidis, 2016]. It represents a different category from the methods and result reproducibility because a similar conclusion can be drawn from using different data, receiving different results or from an entirely different implementation methodology. However, inferential reproducibility is the most disputable category because of the differences in opinions between scientists, due to their own understandings and expertise.

[1] https://www.acm.org/publications/policies/artifact-review-badging

In Artificial Intelligence (AI) area, another classification is used to denote the level of reproducibility. The classification is based on the aforementioned one and it provides a ranked list based on the type of results or documentation [Gundersen and Kjensmo, 2018]. The categories are the following:

C1 - *experimental reproducibility* - It implies that the same results are produced by using the same implementation and data as in the original study.

C2 - *data reproducibility* - The category implies that the same results should be produced if an alternative implementation is used, but the same data as mentioned in the original study.

C3 - *methods reproducibility* - This category suggests that the same results should be obtained using an alternative implementation and a different dataset.

Regardless of the differences in terminology, the awareness among the scientific community towards reproducibility, in any flavor, is increasing, due to surveys that suggest a reproducibility crisis. According to Baker, 2016, the research is facing a reproducibility crisis in many domains such as chemistry, earth and environment, biology, medicine, physics and engineering, and others. The majority (52%) of researchers that were interviewed indicated the existence of a significant crisis. To mitigate it, the community encourages adopting the practices that lead to a successful reproducible study. One example is represented by Papers with Code[2], an online platform with the mission of making the code and evaluation tables publicly available. The motivation of reproducing one's research relies on the following: ensure that the results are not influenced by hidden factors [Manolescu et al., 2008], discover bugs that lead to non-similar results [Peng, 2011], increase the impact and visibility [Freire, Bonnet, and Shasha, 2012] and allow future research to rely on published and available research.

The matter has been tackled in the database community ever since 2008 when SIGMOD published a report regarding the assessment of results reproducibility category [Manolescu et al., 2008]. Although interested in the matter, the authors claimed that the reproducibility issue should remain optional due to the intellectual property rights some might have. The experiment was successful because 66% of the submitted papers provided the necessary means to repeat the experiments and the majority of the participants reported that the process is helpful and "it raises the standards for the community" [Manolescu et al., 2008]. Moreover, in the same year, the SIGKDD conference announced that the papers which do not provide the implementation details and parameter specifications will be downgraded. Similarly, the VLDB conference included in the review guidelines the necessity of experiment descriptions. Ever since, the conferences incentivise the authors to provide the necessary requirements for assessing the results reproducibility and the awarded papers are publicly available[3] [4] [5].

Aside from the efforts employed by the conferences to raise the attention to the reproducibility subject, research has been published to address the pitfalls and suggest recommendations on how to improve. To assess how well a paper can be reproduced, three characteristics of the experiments were identified [Freire, Bonnet, and

[2] https://paperswithcode.com/

[3] http://db-reproducibility.seas.harvard.edu/

[4] https://vldb-repro.com/

[5] https://www.kdd.org/awards/sigkdd-best-research-paper-awards

Shasha, 2012]: depth, portability and coverage. The *depth* of the experiments suggests how much they were made available and the default level is represented by the figures that are associated with the paper. The next levels are: availability of the scripts and data used to generate the figures, the data used for the experiments, the actual experiments and the system, either as a white box with details about build and deployment or as a black box. The *portability* represents the system's dependency on the running environment and it can be measured in three levels: the system can run only on the original environment, on a similar environment (different machine, same operating system) or a different one (different machine and different operating system). Finally, the *coverage* represents the proportion of the experiments that can be reproduced: partially or totally.

The three characteristics represent a good indication of how well a paper is reproducible and while analysing the research, the following pitfalls have been identified [Vitek and Kalibera, 2011]:

- *unclear experimental goals* - as a consequence of the fact that the experiments are usually poorly defined, authors report good results even if the improvements are small;
- *implicit assumptions or experimental methodology* - the experiments should contain all the necessary steps and all the assumptions should be explicitly declared;
- *proprietary benchmarks and data sets* - experiments that do not indicate the data sources diminish their value, because the claims can not be verified;
- *weak statistical analysis* - most of the measurements are effectuated without any indication of the uncertainty;
- *measuring the wrong thing / right thing meaninglessly*;
- *lack of a proper baseline* - instead of referring to the state-of-the-art, many authors use as a baseline their implementation by adding or removing certain optimisations;
- *unrepresentative workloads* - either using a data distribution that does not reflect a real life scenario or using an over-representative benchmark.

To overcome these pitfalls, the minimum recommended is that the code and data should be openly available in a centralised location where the data should have a non-proprietary format [Peng, 2011]. Moreover, the research community needs open source benchmarks to assess the quality of the algorithms on objective data [Vitek and Kalibera, 2011]. Fulfilling these requirements will further encourage publishing reproduction studies and will help repeating the published results.

4.2 Paper selection

Chapter 4 aims to assess the degree of reproducibility of the state-of-the-art described in the literature survey (Chapter 2). In this section we will present the criteria used for choosing the papers that will belong to the reproducibility analysis. The main criterion used for selecting the papers is represented by how varied in approach the algorithms are. The variety in approach can be observed in Table 3.2, which summarizes the findings in the taxonomy and the characteristics of each research described in the literature survey. Moreover, the taxonomy presents three main categories for schema matching algorithms based on the data type they use: schema level, instance level and the combination of both. We chose one representative research for each category as follows:

- **schema level** - The schema level data is the most used type of data in the literature review (Table 3.2) and it accounts for 55% of the total. We chose as the most representative paper, the one from Madhavan, Bernstein, and Rahm, 2001, named further **Cupid**[6]. Cupid as COMA [Do and Rahm, 2002] and S-Match [Giunchiglia, Shvaiko, and Yatskevich, 2004] belong to six sub-categories. However, Cupid represents the most popular schema matching algorithm and it is widely referenced in the literature.

- **instance level** - The instance level data became popular from 2005, although the first paper mentioned in our literature review is from in 2003. In total, we discovered only four papers that use instance data, which accounts for 20% of the total. We decided that the paper which qualifies for the reproducibility analysis should be the one by Zhang et al., 2011, because it explores both element and structure level and it uses the clustering algorithm which represents an additional complexity level. The paper will be further named **Instance Clustering**.

- **schema and instance level** - The algorithms that use both schema and instance level data have been used since 1994 until 2018, which represents the entire analysis period of our review. Most of the algorithms explore the same subcategories and we decided to analyse the newest, which is represented by Fernandez et al., 2018b, referred next as **SemProp**.

4.3 Reproducibility analysis

The purpose of this analysis is to determine the degree of reproducibility of the most representative papers from the literature review, which were previously selected in Section 4.2. We want to provide a quantitative evaluation before starting the development of the algorithms or performing the experiments. Previously (Section 4.1), we have identified three characteristics that specifies to what extent a paper is reproducible: depth, portability and coverage. Although *depth* can be assessed a priori, the *portability* and the *coverage* can not be assessed until the experiments are run or the algorithms are developed, which does not constitute the goal of the analysis.

Moreover, in the database community, the reproducibility assessment is not well enforced and it is based on volunteering. Thus, the authors that want to pass the reproducibility test should provide external materials with details about system, data, code and experiments[7]. This type of assessment does not match with our goal, because we want to measure the degree of reproducibility based on the details provided in the papers.

Therefore, in order to achieve our goal, we use the categories *C1*, *C2* and *C3* described in the introduction (Section 4.1) and all the variables identified under each category as presented in Gundersen and Kjensmo, 2018 and stated in Table 4.1. We use the following formulas to quantify the degree of reproducibility:

[6] We used the extended version of the paper

[7] http://db-reproducibility.seas.harvard.edu/#Guidelines

$$C1D(e) = \frac{\delta_1 Method(e) + \delta_2 Data(e) + \delta_3 Experiment(e)}{\delta_1 + \delta_2 + \delta_3},$$

$$C2D(e) = \frac{\delta_1 Method(e) + \delta_2 Data(e)}{\delta_1 + \delta_2},$$

$$C3D(e) = Method(e),$$

where e denotes the experiment and δ represents an uniform weight ($\delta_i = 1$). Each factor $Method, Data, Experiment$ represents a weighted sum of the truth values for the variables indicated in Table 4.1 using the same uniform weights.

TABLE 4.1: The table indicates the main categories assessed in the reproducibility study and the variables for computing the degree of reproducibility for each paper. Based on these variables, we present the correspondence between them and each paper and compute the reproducibility degree for each category.

Categories	Variables		Cupid	Instance Clustering	SemProp
Method	Problem		✔	✔	✔
	Objective/Goal		✔	✔	✔
	Research method		✔	✔	✔
	Research questions		✔	✔	✔
	Pseudo code		✔	✔	✔
Data	Training data	Input data			
	Test data				
	Validation data				
	Results			✔	✔
Experiment	Hypothesis		✔	✔	✔
	Prediction		—	—	—
	Method source code				✔
	Hardware specifications			✔	✔
	Software dependencies			✔	
	Experiment setup		✔	✔	✔
	Experiment source code				
Score	C1D		0.44	0.66	0.66
	C2D		0.5	0.625	0.625
	C3D		1	1	1

The variables for each factor are the result of the investigation of AI studies, though a few of these variables are not applicable for assessing the reproducibility level of schema matching research. For the *Data* factor, unlike AI, the schema matching research does not use training, validation and test data. We use the complete dataset to run the algorithms, while the validation is performed against the gold standard. Thus, we will not take into consideration the training data and test data variables, but we will refer to those as *input data*, which is represented by the data sources indicated in the experiments, while the validation data will be represented by the gold standard. In terms of results, we can validate the performance of a model by executing different evaluation metrics on the output generated by the algorithms (set of matches) or by having the results of the evaluation metrics, without any output. The indication of any of the two will be considered as a correspondence of the *results* variable. For the $Experiment$ factor, schema matching does not provide any prediction for the hypothesis and often both the hypothesis and the prediction indicate the same thing. Thus, we will not take into consideration the prediction variable.

The target of the reproducibility study is represented by having a high value for the $C1D(e)$ metric, which reports the degree to which the same results can be obtained using the same implementation and the same data. A high value for the $C2D(e)$ score is also a good indication of reproducibility and it represents the degree to which the same results can be achieved using an alternative implementation, but the same data. The $C3D(e)$ presents the most relaxed requirements in terms of documentation and experiments and a high value is expected.

The maximum value for each metric is

$$\max_{i \in 1,2,3} CiD(e) = 1,$$

and according to Table 4.1, only one paper has the $C1D(e)$ value below 0.5, which represents a low reproducibility degree. However, `Instance Clustering` and `SemProp` both have $C1D(e) = 0.66$ which indicates that is possible to reproduce the papers to some extent. Unhappily, none of the algorithms specify the input data or the validation data. Although they indicate the name of the open-source databases, they do not specify the year or the version number. Giving the fact that the majority of the open-source databases are constantly updating and changing, we can not assume that the input data exists. Such an assumption will introduce bias because we can not assess how much the differences between the versions will influence the final results. Moreover, none of the papers suggest the experiment source code and only one indicates the method source code. Finally, as predicted, the $C3D(e)$ has the biggest value among the metrics and it suggests that the studies are well defined, although the experiments can not be performed using the same implementation and dataset.

In conclusion, `Cupid`, `Instance Clustering` and `SemProp` are method reproducible, as they achieved the highest score possible for $C3D(e)$ metric. Although the papers are not experimental reproducible, nor data reproducible, in the next chapter we will discuss the implementation design decisions that we have made to achieve an alternative implementation. Finally, using a different dataset we will be able to assess the method reproducibility and compare the results.

4.4 Implementation design decision

In the previous section, we proved that `Cupid`, `Semprop` and `Instance Clustering` are only method reproducible, which represents achieving the same results, using an alternative implementation and a different dataset. Thus, in this section, we present the design decisions made to achieve a similar algorithm implementation and behavior.

4.4.1 Cupid

Description

"Generic schema matching with Cupid" [Madhavan, Bernstein, and Rahm, 2001] presents an approach that consumes only schema level information, which is transformed into a schema tree. The most simple schema tree representation is the transformation of relational databases: the schema contains tables that contain columns. Using the tree structure, the algorithm captures structural information, which will be used to adjust the similarities between elements. Prior to computing the structural similarity, the algorithm computes the linguistic similarity, which is based on the name of the elements, their synonyms, and hypernyms and their type, such as the data type or the high-level concept expressed (e.g. *Price* and *Cost* belong to the *Money* concept). Moreover, the algorithm relies heavily on the existence of a thesaurus that is used to compute the linguistic similarity. The thesaurus should provide information about the synonyms and hypernyms of the elements and also the category they belong to. Furthermore, in the pre-processing phase, where the elements are split into tokens, the thesaurus is used to provide the concept of each token and expand any existing abbreviations. Finally, the pipeline also involves the user, as he can add initial mappings between elements, provide thresholds and finally, verify and validate the result.

Based on the algorithm description, we have identified three key parts: the linguistic matching, the structural matching and the pipeline that incorporates the two similarities and outputs the matches. Furthermore, we will describe the limitations and the choices made to overcome them.

Linguistic matching

The linguistic matching contains three steps: normalization, categorization and comparison and is responsible for computing the similarity based on the column names and their semantic properties. First, we will discuss *normalization*.

The normalization step provides a set of tokens that are syntactically similar, meaning that any additional characters or special formatting will be reduced to a normalized form. The step is also divided into four other sub-steps: the tokenization, expansion, elimination, and tagging. Both expansion and tagging are dependent on the thesaurus, which the authors argue that it "plays a crucial role in linguistic matching" [Madhavan, Bernstein, and Rahm, 2001]. Although they mention that it has significant importance in the algorithm, the authors do not indicate the source of the thesaurus. This fact allows us to freely choose any thesaurus and we decided on using *WordNet*[8]. However, WordNet does not provide expansions for abbreviations for all the domains or all the words, nor can it indicate concepts. Therefore, in our implementation, we excluded the expansion and the concepts from the normalization phase.

The next step is the categorization which aims to group the elements in order to reduce the number of comparisons. The categories are determined by the concepts, data types, and containers. The containers are a special type, because they represent the combination of two or multiple elements under a single one (e.g. *Street* and *City* are part of *Address*, therefore *Address* is the container). Since we can not identify both concepts and containers, we rely only on the data types for the categorization phase.

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

In the comparison step, we compute the linguistic similarity, which is based on the name similarity of the elements. The name similarity is computed as a weighted mean of the tokens based on their categories and it uses the synonyms and hypernyms from the thesaurus to calculate the similarity score. If the information is not available, the authors suggest using the sub-strings of the words to match them. For this type of computation, we have used the n-gram distance as defined in [Kondrak, 2005].

Structural matching

The structural matching is using the tree structure in order to compute the similarity between the elements that occur in the same context. It is mentioned that for all the leaf elements, the structural similarity is initialized with the data type compatibility, which is retrieved from a compatibility table provided by the user. It is also mentioned that the maximum value for data type compatibility is 0.5, in order to allow a further increase in value during another step of the algorithm. As the thesaurus, the source of the data type compatibility table is not mentioned. We consider that manually providing such compatibility table is biased, because the values can be subjective and dependent on the user's preferences. Therefore, we rely on the name similarity formula to compute the data type similarities. For the non-leaf elements, the structural similarity it is computed using the formula introduced in the paper:

$$ssim(s,t) = \frac{\left| \begin{array}{c} \{x \mid x \in leaves(s) \land \exists y \in leaves(t), stronglink(x,y)\} \\ \cup \{x \mid x \in leaves(t) \land \exists y \in leaves(s), stronglink(y,x)\} \end{array} \right|}{|leaves(s) \cup leaves(t)|},$$

where the *stronglink* represents the weighted similarity:

$$wsim = w_{struct} \times ssim + (1 - w_{struct}) \times lsim.$$

Pipeline

The algorithm combines the structural similarity and the linguistic similarity using the formula for *wsim* presented above. Moreover, it uses tree traversal to access the elements and compute the similarities, all being incorporated under a single algorithm, named *tree match*. The algorithm is straightforward and we did not encounter difficulties implementing it.

Additionally, the paper describes methods for including other properties that might be detected in XML/XSD or object-oriented schemas, such as *isDerivedFrom* relationships. However, relational databases do not contain such types of relationships and we decided to omit that part of the algorithm.

Another addition to the algorithm is represented by mapping the referential constraints, which can be part of the relational schema elements. The authors describe the method to incorporate the constraints in the tree schema, which results in a Direct Acyclic Graph (DAG) [Thulasiraman and Swamy, 2011]. However, traversing the DAG does not output a unique path. The problem is addressed in the paper but does not offer a concrete solution: "determining which ordering would be best is still an open problem" [Madhavan, Bernstein, and Rahm, 2001]. Therefore, we decided to omit this part as well.

TABLE 4.2: The table summaries the differences in implementation between the original study and our interpretation

Original	Own interpretation
Thesaurus	WordNet
Abbreviation expansion	N/A
Concepts	N/A
Containers	N/A
Sub-string matching	N-Gram distance
Data compatibility table	Name similarity for data types
isDerivedFrom relationships	N/A
Constraints	N/A

Conclusion

In conclusion, a summary of the algorithm changes that we have employed can be observed in Table 4.2. The first three rows in the table with *not applicable* (N/A) instances are the result of the unspecified source of the thesaurus in the original paper and missing information from our chosen thesaurus. This emphasizes its importance in the algorithm and different results are expected depending on the vocabulary.

4.4.2 Instance Clustering

Description

Instance Clustering represents an algorithm entirely based on the data instances, being categorized in the taxonomy as an instance-level algorithm. It is independent of any external information and it uses strictly the distribution of the data and graph operations. The algorithm contains two phases: computing the distribution clusters and computing attributes.

The distribution clusters are calculated using the EMD metric. Based on a user-defined threshold, all the values under the threshold are considered to be part of the same cluster and they are mapped in a graph structure as neighboring nodes.

The second phase consumes the connected components of the graph structure, which are the result of either a depth-first search or breadth-first search [Thulasiraman and Swamy, 2011]. It uses the intersection EMD to differentiate between columns that do not have any values in common but reported a high value for the default EMD. Phase two follows the same principle as the first one: after the intersection EMD is computed, the values under the threshold represent the same cluster and are being connected in the graph. Next, the adjacency matrix of the graph is computed and the edges are split into positives and negatives based on their value in the matrix. Phase two outputs the correlation clustering, which represents the result of solving a linear integer problem. As in the previous section, we will detail the challenges and the solutions for each phase.

Phase one

In this phase, the default EMD is used to find the elements that correspond together, forming the clusters. The algorithm is very well defined and we managed to successfully follow every step. The only expected difference between the implementations is the EMD formula. Originally, EMD consumes numbers instead of strings.

However, the authors mention an alternative for strings and we obtained the same distributions as pictured in the paper, which we consider a sufficiently good validation of our implementation. Finally, considering performance, the EMD is computed using quantile histograms, which results in an approximate distribution, as stated in the original paper.

Phase two

Phase two follows the same logic as phase one, with the difference that it uses intersection EMD instead of the default. The intersection EMD is computed following the formula:

$$EMD_{\cap}(C, C') = \frac{1}{2)}(EMD(C, C \cap C') + EMD(C', C \cap C')),$$

where the C, C' represent the columns. The values below the threshold represent similar attributes and they are mapped in a graph structure as neighboring elements. Next, the adjacency matrix is computed and it is used to detect the positive and negative edges. Finally, the graph is solved using correlation clustering, which minimizes the number of disagreements: "the number of positive edges whose endpoints are in different clusters, plus the number of negative edges whose endpoints are in the same cluster" [Zhang et al., 2011].

One change that we employed in our algorithm is related to the linear program that is used to solve the correlation clustering problem. In the paper, *CPLEX*[9] is mentioned as the liner integer problem solver. However, we opted for the open-source version *PuLP*[10]. The reasons for choosing PuLP over CPLEX are strictly related to the portability and ease of access, as CPLEX is an integrated framework that requires more assistance during installation.

Conclusion

In conclusion, phase one is implemented entirely as stated in the paper, while phase two differs in the choice of the linear program solver. Regardless of this difference, we are expecting very similar results in the evaluation section.

4.4.3 SemProp

Description

SemProp represents an algorithm that uses both schema information and data instances to find matches between databases. The algorithm relies on word embeddings and it tackles two limitations: the inability of word embeddings to address a combination of words (e.g. "drug interaction") and handling the words that do not exist in the vocabulary. Therefore, the paper introduces *coherent groups*, which focuses on solving the two limitations. Moreover, the algorithm uses both semantic approaches, such as coherent groups, and syntactic approaches. The results of each approach represent the similarity between the words, which is then combined in a DAG structure together with match operators such as union, set-difference and the structural summarizer. We identified three main steps of the algorithm: the semantic matching step, the syntactic matching, and the final step, structural summarizer. In

[9] https://www.ibm.com/nl-en/analytics/cplex-optimizer

[10] https://pythonhosted.org/PuLP/

what follows, we will discuss in detail each step, the difficulties identified and the solutions we propose.

Semantic matching

The semantic matching introduces the coherent groups, which represent the average of all-pairs similarities between two elements, where the elements can have more than two words. Based on the result and given a user-defined threshold, the elements are split into positive signals (the values above the threshold) and negative signals (the values below the threshold). The negative signals are further used to filter the results of the syntactic matchers. Moreover, this phase manages the special case where the word embedding thesaurus does not contain all the words of the elements. The unknown words are then penalized and the similarity becomes zero. This solution is based on the fact that the data should be clean (it does not contain typos), which implies that the word does not exist in the vocabulary. Another special case is identified when computing the similarities between identical words, which results in perfect matching. The authors decided to discard this similarity, which will increase the weight for the rest of the words.

Based on the description of the coherent groups and the special cases, we did not encounter any difficulties in implementing the semantic matching step.

Syntactic matching

In the syntactic matching phase two types of matchers are employed: one instance based matcher that uses Jaccard similarity [Gower, 1985] and one matcher that computes the syntactic similarity of names. The problem identified in this step is the choice of matchers. In the original paper, it is mentioned that the state-of-the-art algorithms are applied, but they are referenced using the survey [Rahm and Bernstein, 2001] that we followed in the literature review. The survey presents six algorithms for syntactic name similarity and we do not have any indication on which is the one referred in the original paper of SemProp. One of the six algorithms is *Cupid*, for which we already have an alternative implementation of it that we can use.

Structural summarizer

The structural summarizer uses ontologies to reduce the number of links between elements and refine the matches. The algorithm starts when an element has more than N (`summaryThreshold`) links. The first step implies retrieving all the ancestors of the classes from the ontology, starting at the root and adding them to a trie structure. Next, we should find the class that summarizes most of the links. To achieve this, we should find the child node that represents the largest number of links. If the ratio between the number of links contained by the node and the total number of links is smaller than a threshold, all its children are returned. They represent now the new links to summarise. The most confusing part of the algorithm is the last part, which indicates another method in their pseudo-code (Figure 4.1), which is not detailed anywhere in the paper. At this point, the implementation of the structural summarizer is blocked.

Moreover, another pseudo-code is given to solve the problem of two elements having links to classes with relationships between them. The pseudo-code contains parts that are not self-explanatory and no explanation or interpretation is given. At this point, we can not make any assumptions about the algorithm and the implementation is revoked.

Algorithm 1: Structural Summarizer

```
def structuralSummarizer(links, cuttingRatio):
  hierarchySeqs = getSequences(links)
  trie.addSeqs(hierarchySeqs)
  (linksToSum, cutter) = findCutter(trie,cuttingRatio)
  outputLinks = []
  newLink = reduceTo(linksToSum, cutter)
  outputLinks.add(newLink)
  return outputLinks
def findCutter(trie, numSequences, totalNumSequences,
                cuttingRatio):
  chosenChild, numReprSeqs = chooseMaxReprChild(trie)
  ratioCut = numReprSeqs / totalNumSequences
  if ratioCut > cuttingRatio:
    return findCutter(chosenChild, numReprSeqs,
                        totalNumSequences)
  else:
    linksToSum = retrieveLeafsFromNode(chosenChild)
    return linksToSum, chosenChild
```

FIGURE 4.1: The structural summarizer algorithm as presented in Fernandez et al., 2018b. The highlighted part represents the new function that is not described anywhere in the paper.

Conclusion

Although the algorithm could not be implemented due to missing implementation details, the authors provided[11] the link to the open-source code of the framework that incorporates the entire SemProp algorithm. We encountered difficulties identifying all the parts of the algorithm as described in the paper, but one of the authors replied to our questions via email and we were able to construct the entire pipeline. We concluded that indicating the source code of the algorithm allowed the authors to omit many implementation details.

4.5 Conclusion

In conclusion, reproducibility represents a term often used interchangeably with repeatability and it describes an approach still young in the data community. As our reproducibility study presents, the papers do not provide the two most essential variables: the data and the source code. However, if the source of the data is indicated, important details about the versions or retrieval year are not specified. Besides the method source code, the papers should also contain the code for the experiments and explicitly indicate the golden standard or provide scripts to construct it based on the open-source data.

Finally, our study reported that the most representative state-of-the-art papers are only methods reproducible, which indicates that the same results should be obtained using an alternative implementation and different data. Furthermore, we demonstrated that an alternative implementation is possible and in the next chapter we will evaluate the models implemented using similar open-source data.

[11] The link is mentioned in the conclusion of the paper

Chapter 5

Evaluation

In the *Evaluation* chapter we continue the reproducibility study. Previously, we described the term *reproducibility* and presented the reproducibility status of schema matching. Furthermore, we selected three representative studies based on the data type they employ and in accordance with categories presented in the taxonomy: one schema-based paper, one instance-based and a paper that uses both schema and instances. Finally, we analyzed the papers and demonstrated that they are only **methods reproducible**, which implies that the same results should be obtained using an alternative implementation and a different dataset. We have provided an alternative implementation of each algorithm in the previous chapter, while in this chapter we will present the results of running the alternative versions of the algorithms on the selected data and conclude the reproducibility study. The **goal** of the evaluation section is to demonstrate that we have successfully implemented the algorithms and the results are similar with the ones reported in the original papers.

In terms of data, each algorithm consumes a different dataset as described in the original studies [Madhavan, Bernstein, and Rahm, 2001, Fernandez et al., 2018b, Zhang et al., 2011]. Although the exact version or source of each dataset is not always specified in the original papers, we have searched for the most accurate representation of each dataset, as it is mentioned in the original papers. Besides the dataset, the gold standard is also needed and rarely available. Thus, we have contacted the authors and solicited the gold standard data or we have developed it by manually searching for the correspondences or employing automatic methods. More details about each dataset are available in the sections concerning each algorithm.

The evaluation will be performed against the gold standard and we will adopt the most used metrics in schema matching evaluation: precision, recall and the F1-score (or F-measure) [Do, Melnik, and Rahm, 2002, Yatskevich, 2003, Duchateau, Bellahsene, and Hunt, 2007, Bellahsene et al., 2011]. In order to fully understand the terms, we provide the following definitions:

Gold standard *represents the ground truth or all the real matchings. It is usually created manually by the experts in the field or provided by automated means and verified by the experts* [Do, Melnik, and Rahm, 2002].

Precision *represents the amount of correct matchings returned by the algorithm among the total machings returned* [Do, Melnik, and Rahm, 2002]:

$$Precision = \frac{correct\ matchings}{total\ matchings}$$

Recall *represents the amount of correct matchings generated by the algorithm among the gold standard* [Duchateau, Bellahsene, and Hunt, 2007]:

$$Recall = \frac{correct\ matchings}{gold\ standard}$$

F1-Score *represents the accuracy of a test and it is a trade-off between the precision and recall. It is defined as the harmonic mean between the precision and recall as follows* [Bellahsene et al., 2011]:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

For a schema matching algorithm the metrics represent the following: maximum precision indicates that all the generated matchings were correct, maximum recall signifies that the correct matchings generated by the algorithm represents the entire gold standard dataset and a schema matching algorithm can achieve maximum F1-score if all the generated matchings are correct and they represent all the matchings from the gold standard.

The chapter is structured according to the papers analysed: *Cupid*, *SemProp* and *Instance Clustering*. Moreover, the chapter contains an additional analysis of data used in industry settings. The goal is to measure the performance of the algorithms in a real-life situation, instead of using only synthetic datasets.

5.1 Cupid

In the original paper, the evaluation of Cupid is conducted as a comparative study and the performance of the algorithm is compared against the performance of Dike [Palopoli, Sacca, and Ursino, 1998] and MOMIS [Bergamaschi, Castano, and Vincini, 1999], two other similar algorithms. The algorithms are first tested on canonical examples and small subsets from the data are reported in the results. The data and the tests are not mentioned and the results are reported in a table using a binary variable which indicates the ability of the algorithm to identify the elements (yes or no).

Another test is performed on real data and the schemas are provided as pictures, instead of actual data files. The test contains two scenarios: one scenario containing XML data and the other one containing relational data. Our goal implies testing the algorithms only on the relational data, thus we focused on the second scenario. Furthermore, the initial study briefly mentions the gold standard and the authors do not use any metrics to measure the performance. Instead, they report what pairs of elements were matched and not matched by each algorithm.

In our evaluation, we use the same relational data as indicated in one of the figures from the original paper and we devised the gold standard according to the specifications. Moreover, we augmented the gold standard with the primary-foreign key (PK-FK) pairs that were not originally indicated.

Considering the fact that we conduct the evaluation on an alternative implementation, we investigate the performance of the algorithm using different values of the thresholds than mentioned in the original paper. The algorithm uses seven different thresholds, but we have tested only five for different values: th_{accept} because it determines which elements are matching based on their leaf similarity, th_{low} and th_{high} accordingly, because they should be lower or higher than the *accept* threshold and they determine which similarities to increase or decrease for the second run and w_{struct} for leaves and non-leaves nodes, which are used to assign different weights to either linguistic or structural similarity based on the node type. The initial thought was that w_{struct} leads to similar results, but on a different scale. However, the experiments revealed a different behavior. The thresholds that we do not change are the multiplicative thresholds because they only impact the result on the second run of the algorithm that determines the relation between table names and th_{ns} which represents the name similarity threshold that is used to prune the number of element to element comparisons, which decreases the processing time. The thresholds are summarized in Table 5.1.

Figure 5.1 presents the precision, recall and F1-score using different values for the thresholds, as indicated in Table 5.1. The best F1-score value is achieved for $w_{struct\ leaf} = 0.5$ and $th_{accept} = 0.37$. However, the precision is approximately 0.2, while the recall is between 0.4 and 0.5. We can also observe that for a small value of th_{accept} we achieve maximum recall, while for bigger values we achieve maximum precision. This behavior is due to the output of the algorithm as follows:

- For a **small** th_{accept}, the algorithm outputs all or almost all possible combinations between the schema elements. Therefore, the recall is maximum, while the precision is significantly small.

- Increasing the value of th_{accept}, we noticed that the number of returned matchings is decreasing, until the maximum values where it outputs a small list (approximately five matchings), thus we achieved the highest precision.

Moreover, we noticed an unusual behavior while increasing the value of $w_{struct\ leaf}$. A high value favors more the structural similarity and less the linguistic similarity. For a relational dataset, this is not the behavior we prefer, because the structure of a relational dataset is very simple: the schema contains the tables and the tables contain the columns. Therefore, a higher weight for the linguistic similarity is more preferable as the results indicate.

Due to these differences, to compare the output presented in the original paper and the output achieved using our alternative version, we decided to set the thresholds according to the F1-score, which reports the trade-off between the precision and recall. Therefore, the best values from Table 5.1 are chosen according to the fifth plot in Figure 5.1.

Our alternative version and the original version have more similarities than differences in results, as presented in Table 5.2. They both find the same columns and make the same mistakes, but our version cannot identify the primary-foreign key combinations. The inability to discover the PK-FK matchings is motivated by the fact that we did not map the constraints because of the missing implementation details in the original paper.

To conclude, we proved that although a few implementation details are missing and we used slightly different algorithms or thesaurus, we achieved the same general behavior, however not the same results.

TABLE 5.1: The table summarizes the thresholds that we tuned, it shows the value ranges used, the values for the best configuration and the values reported in the original paper.

Threshold	**Value range**	**Best value**	**Best value original paper**
th_{accept}	0.05–0.5 (step 0.02)	0.37	0.5
th_{low}	th_{accept} - 0.01	0.27	0.35
th_{high}	th_{accept} + 0.01	0.47	0.6
$w_{struct\ leaf}$	0.1–1.0 (step 0.1)	0.5	0.5
$w_{struct\ non\ leaf}$	$w_{struct\ leaf} + 0.1$	0.6	0.6

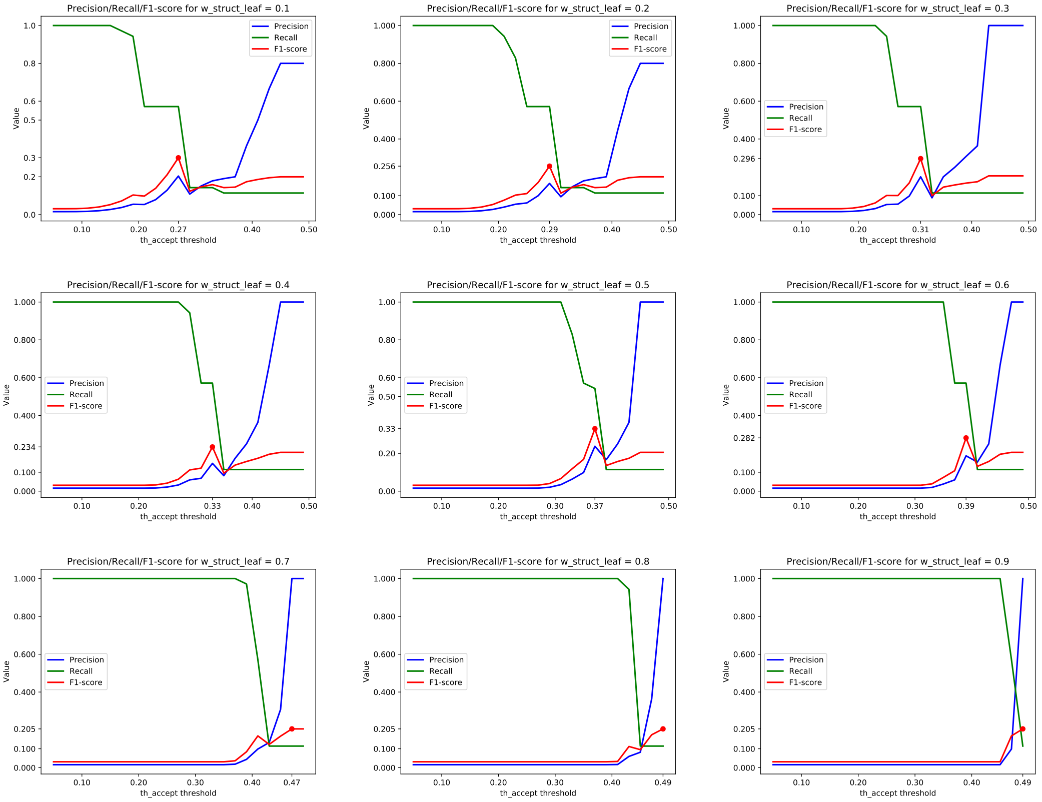


FIGURE 5.1: The precision, recall and F1-score for different configurations of the parameters.

TABLE 5.2: The table presents the similarities and differences between the output of Cupid in the original paper and the output of Cupid in our alternative version

Original version	**Our version**
Orders/OrderDetails -> Sales	Same except the KFs
Products -> Products	Same except the KF
Customers -> Customers	Only two matchings from six
Region/Territory -> Geography	All
TerritoryRegion -> Geography	None
Three PostalCode columns -> Customers.PostalCode	Same except the KF
No match CustomerName -> ContactFirstName	Same
No match CustomerName -> ContactLastName	Same

5.2 SemProp

The evaluation of *SemProp* conducted in the original paper is based on the ability of the algorithm to create quality links between different elements, which means positive matchings. Thus, the evaluation section is based on six scenarios that test different parts of the algorithm and show how each component improves the precision and recall, having as the baseline the performance of the syntactic algorithm alone.

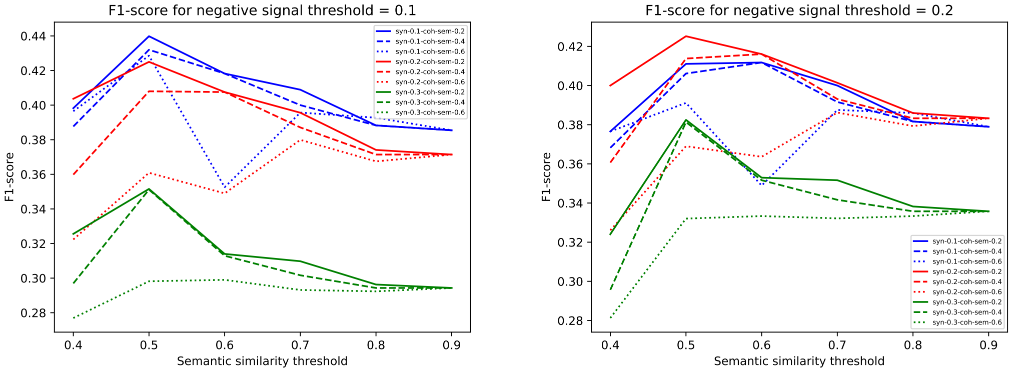


FIGURE 5.2: The precision, recall and F1-score for different configurations of the parameters.

The evaluation is conducted on three different datasets: two public and one private, each using different ontologies accordingly. Despite the fact that a few datasets are public, the authors do not report the versions used or the gold standard. To overcome this issue, we have emailed one of the authors and he provided us with the gold standard between the ChEMBL_22 database[1] [Gaulton et al., 2017] and EFO ontology[2].

In our evaluation, we use the ChEMBL_22 dataset, the EFO ontology and the gold standard provided and we measure the precision, recall, and F1-score. This dataset configuration represents only half of the data used in the original paper for evaluation, thus we can not obtain the same results. However, our goal is to prove that the algorithm expresses the same behaviors: (1) decreasing the F1-score while increasing the syntactic similarity threshold and (2) based on the best configuration, demonstrate the sensitivity of the structural summarizer in relation to two thresholds: *cuttingRatio* - "higher values indicate that the algorithm will only summarize when a large portion of the links are children of an ancestor" [Fernandez et al., 2018b] and *summaryThreshold* - "the best values [..] are achieved as soon as there is more than one link between a source element and an ontology class" [Fernandez et al., 2018b].

Firstly, we want to discover which parameters yield the best F1-score. Although in the original paper only a few of the thresholds are mentioned, we discovered more thresholds in the repository[3]. Therefore, we conducted several experiments in order to determine the best configuration. The tested parameters summarized in Table 5.3 are the semantic similarity threshold *sem* used to output the positive signals, the negative signal threshold *neg* which implies that all the pairs with the semantic similarity threshold below *neg* are negative and will be used to prune the results of the syntactic matcher, the semantic similarity threshold for the coherent groups $coh - sem$ which implies that if the semantic distance of two word embeddings is bigger than the threshold, then the words belong in the same coherent group and the syntactic similarity threshold *syn* which is the only one mentioned in the original paper. The behavior of the latter one indicates that for a bigger value, the precision, recall, and F1-score are decreasing. We were able to reproduce the result and

[1] http://chembl.blogspot.com/2016/09/chembl-22-released.html

[2] https://www.ebi.ac.uk/ols/ontologies/efo

[3] https://github.com/mitdbg/aurum-datadiscovery

Figure 5.2 illustrates the same behavior as follows: each figure presents three different configurations for each syntactic similarity syn value based on the semantic similarity threshold for coherent groups $cog - sem$. We can observe that the best F1-score is achieved in the first figure, which corresponds to $neg = 0.1$. Thus, the best thresholds are: $neg = 0.1$, $sem = 0.5$, $coh - sem = 0.2$ and $syn = 0.1$.

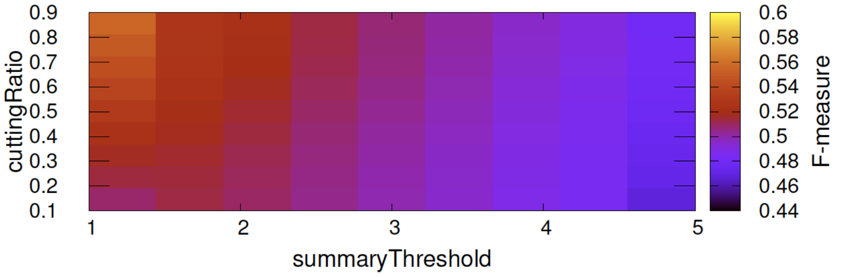


FIGURE 5.3: Sensitivity of StructS as provided in the original paper.
Source: Fernandez et al., 2018b

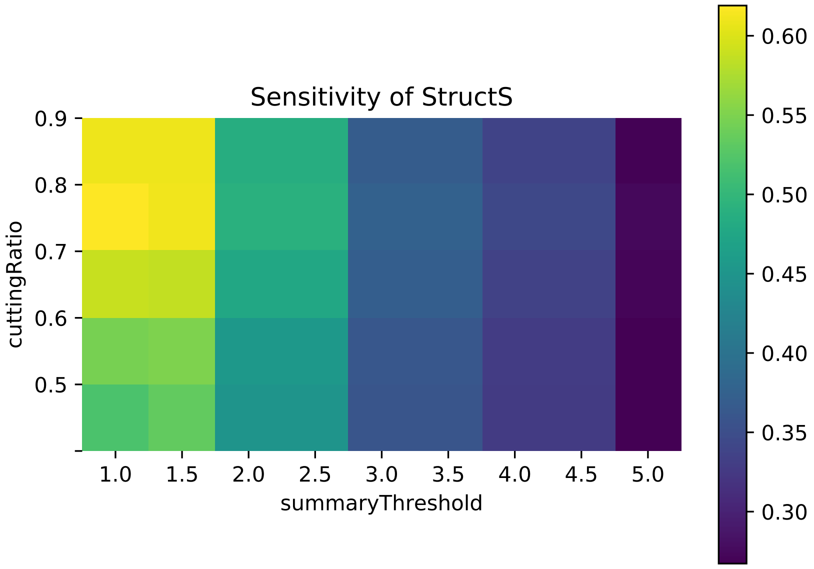


FIGURE 5.4: Sensitivity of StructS as provided by our experiments.

Furthermore, we want to prove the same sensitivity of the structural summarizer in relation to the cutting ratio and the summary threshold as provided in the original paper (Figure 5.3). As a consequence of the fact that the authors do not mention the complete configuration of the algorithm, we decided to use the best values resulted from the first experiment and noted in Table 5.3. Moreover, due to time limitations, the value range used for cutting ratio is $0.5 - 0.9$ instead of $0.1 - 0.9$ as indicated in Figure 5.3. The result of the experiment is illustrated in Figure 5.4 and it shows the same sensitivity as in the original paper.

In conclusion, we proved that SemProp is methods reproducible because we obtained the same results using a different dataset. To the best of our knowledge, the implementation is the same as in the original paper, thus the similarity between results was expected.

TABLE 5.3: The table summarizes the thresholds that we tuned, it shows the value ranges used, the values for the best configuration and the values reported in the original paper.

Threshold	Value range	Best value	Best value original paper
sem	0.4–0.9 (step 0.1)	0.5	—
$coh-sem$	0.2–0.6 (step 0.2)	0.2	—
neg	0.1, 0.2	0.1	—
syn	0.1–0.3 (step 0.1)	0.1	0.2

5.3 Instance Clustering

The experiments in the *Instance Clustering* paper [Zhang et al., 2011] are conducted using three open-source datasets: TPC-H[4], IMDB[5] and DBLP[6]. However, the datasets do not have any version associated and we noticed that IMDB is refreshed daily, DBLP has additional attributes with every release and TPC-H provides corrections or additional items in the database with every revision. Therefore, we chose to conduct the experiments using only the TPC-H example from the paper and we used version 2.18.0. It is a small subset of the TPC-H database and the authors mention the gold standard as an illustration.

The metrics used for evaluation are custom precision and recall, which use the attributes and the matchings between the columns inside an attribute, instead of simple comparison between matchings pairs. The metrics are defined as follows:

Let $A = \{A_1, A_2, ..., A_m\}$ be the set of discovered attributes and $T = \{T_1, T_2, ..., T_m\}$ be the set of attributes from the gold standard. "Let A_i correspond to T_j if and only if the majority of columns in A_i also belong in T_j" [Zhang et al., 2011]. The following formulas are used to compute the precision and recall for an attribute:

$$Precision(A_i) = \frac{|A_i \cap T_j|}{|A_i|}$$

$$Recall(A_i) = \frac{|A_i \cap T_j|}{|T_j|}$$

The final precision and recall are computed as an average over the precision and recall for each attribute:

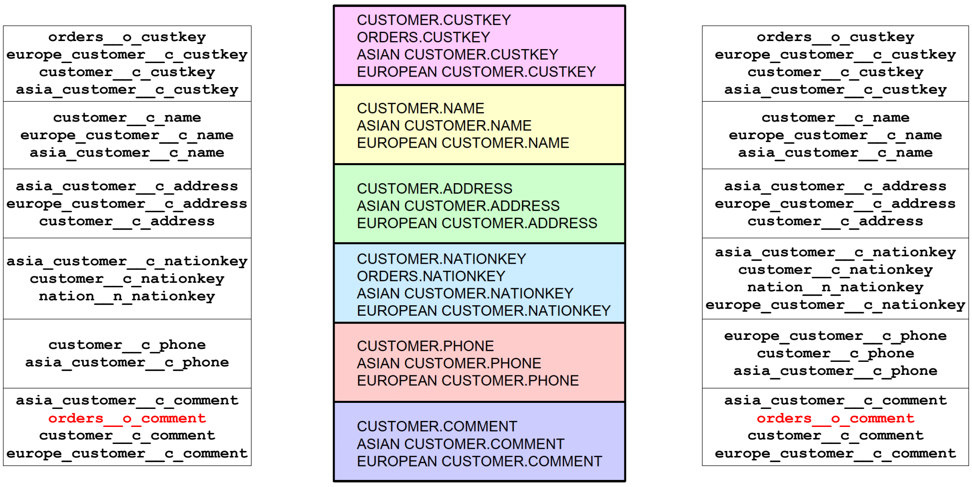
$$Precision(A) = \frac{\sum_{i=1}^{m} Precision(A_i)}{m}$$

$$Recall(A) = \frac{\sum_{i=1}^{m} Recall(A_i)}{m}$$

[4] http://www.tpc.org/tpch/

[5] https://www.imdb.com/interfaces/

[6] http://dblp.uni-trier.de/xml/



orders__o_custkey europe_customer__c_custkey customer__c_custkey asia_customer__c_custkey
customer__c_name europe_customer__c_name asia_customer__c_name
asia_customer__c_address europe_customer__c_address customer__c_address
asia_customer__c_nationkey customer__c_nationkey nation__n_nationkey
customer__c_phone asia_customer__c_phone
asia_customer__c_comment orders__o_comment customer__c_comment europe_customer__c_comment

(A) $\theta = 0.1$

CUSTOMER.CUSTKEY ORDERS.CUSTKEY ASIAN CUSTOMER.CUSTKEY EUROPEAN CUSTOMER.CUSTKEY
CUSTOMER.NAME ASIAN CUSTOMER.NAME EUROPEAN CUSTOMER.NAME
CUSTOMER.ADDRESS ASIAN CUSTOMER.ADDRESS EUROPEAN CUSTOMER.ADDRESS
CUSTOMER.NATIONKEY ORDERS.NATIONKEY ASIAN CUSTOMER.NATIONKEY EUROPEAN CUSTOMER.NATIONKEY
CUSTOMER.PHONE ASIAN CUSTOMER.PHONE EUROPEAN CUSTOMER.PHONE
CUSTOMER.COMMENT ASIAN CUSTOMER.COMMENT EUROPEAN CUSTOMER.COMMENT

(B) Original paper

orders__o_custkey europe_customer__c_custkey customer__c_custkey asia_customer__c_custkey
customer__c_name europe_customer__c_name asia_customer__c_name
asia_customer__c_address europe_customer__c_address customer__c_address
asia_customer__c_nationkey customer__c_nationkey nation__n_nationkey europe_customer__c_nationkey
europe_customer__c_phone customer__c_phone asia_customer__c_phone
asia_customer__c_comment orders__o_comment customer__c_comment europe_customer__c_comment

(C) $\theta = 0.2$

FIGURE 5.5: The figure illustrates the output of our experiments (A and C) and the output reported in Zhang et al., 2011 (B).

The performance of the algorithm using the small subset is not explicitly indicated in the original paper, thus we do not have any measurements to compare with. Instead, we compare our results with the attributes indicated in the original paper and illustrated in Figure 5.5b.

We conducted two experiments using two different cutoff thresholds $\theta = 0.1$ and $\theta = 0.2$, which represent the smallest and the largest values used in the evaluation section in the original paper. We conformed to the rest of the specifications and we used the maximum number of quantiles, which is reported to output positive results: $quantiles = 256$.

The outcome of our experiments is illustrated in Figure 5.5. It can be observed that both experiments add an extra element in the last cluster: the column *comment* from the *orders* table. Moreover, for a smaller θ, one element is missing from the *phone* cluster and one from the *nationkey* cluster.

In conclusion, we demonstrated that the same results can be obtained using an alternative implementation and a different dataset. However, we can only assume that the datasets are different because there is no indication of the version in the original paper.

5.4 Industry case study

In the previous sections, we demonstrated that we can achieve similar behaviors or results using the alternative implementation of the algorithms and datasets that resemble the ones used to evaluate the algorithms in the original papers. In this section, we evaluate the algorithms in an industry setting and discuss their performance in comparison with the synthetic data.

The industry setting is represented by ING. The company faces data integration problems such as manually creating mediation schemas. This represents an issue for the company for the following reasons: firstly, manually inspecting all the data and gaining insights is a very laborious process and secondly, it requires permanent maintenance due to the continuous transformation of data (data is produced daily). One approach to solving the issue is schema matching. Therefore, we will evaluate the algorithms on a small subset of the data and decide if they are suitable for the company environment.

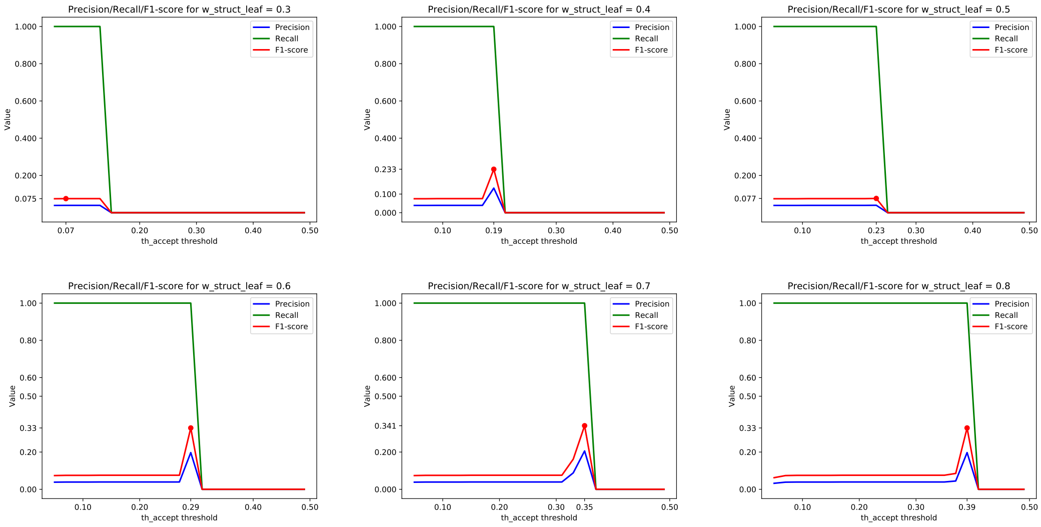


FIGURE 5.6: The precision, recall and F1-score for different configurations of the parameters of the *Cupid* algorithm in the industry setting.

The evaluation scenario includes datasets produced by systems containing configuration data. Ideally, the results of running the algorithms on the configuration data will help the company integrate different datasets such as historical data, metrics and event logs. The proposed dataset for the evaluation contains two tables with 33 and 16 columns and 1000 rows each.

As demonstrated in the previous sections, not all the algorithms are suitable for a certain scenario. Therefore, for the scenario proposed by the ING data science team together with the data extraction team, we are not able to run *SemProp*, due to the fact that it requires ontologies in order to infer the meaning of the data. In the setting proposed by ING, the meaning of the data resides outside of an open-source ontology thus, this algorithm did not undergo the evaluation.

On the other hand, *Cupid* can be evaluated on the proposed dataset and we used the same approach as in Section 5.1. We performed a search for the best values of the thresholds presented in Table 5.1 using the same value ranges in order to discover the configuration that outputs the best F1-score. We used experts from ING to inspect the results and devise the gold standard to derive statistics. As such, the result of the experiments can be observed in Figure 5.6, which illustrates a very similar behavior with the results presented in Section 5.1 of evaluating the algorithm on the synthetic data.

The last algorithm evaluated at ING is the *Instance Clustering*. Reporting the results to the experts, they could not agree on what should be the correct clustering output, therefore we could not provide any measurements, such as precision, recall, and F1-score. However, they indicated that the performance is significantly better than the performance of *Cupid*. Nevertheless, *Instance Clustering* produces false positive due to the similarities in the data. One example of such similarity is noticed between columns *Description* and *ShortDescription* that share many values. Essentially, both teams were satisfied with the results of the algorithm. As such, the next steps proposed by the company is increasing the number of data instances in order to be able to determine the accuracy of the algorithm. Such scenario could not be implemented yet, due to internal policies.

In conclusion, according to the results and the resources necessary to execute the algorithms (human experts to assess the quality, time and hardware to process each configuration), ING concluded that these algorithms are not sustainable, by cause of resource limitation. The motivation behind the conclusion relies on the fact that data is updated very often and these algorithms require careful maintenance in order to achieve satisfactory long term results. Although *Cupid* can be used to discover the relations between tables based on the structure and the linguistic similarity, it is very resource-intensive. Similarly, *Instance Clustering* requires special hardware specifications in order to output explicit results, that can be easily verified by the experts.

5.5 Conclusion

The chapter continues the reproducibility study, by performing experiments with the goal of achieving the same results. As indicated in Chapter 4, the papers do not provide the datasets, the gold standard or the experimental set-up. Under these circumstances, achieving the same results is not possible. However, we wanted to demonstrate that the algorithms express the same behavior, as such: increasing or decreasing the precision, recall or F1-score by tuning certain parameters. Moreover, we proved that the performance is also affected by the missing implementation details.

Furthermore, we evaluated *Cupid* and *Instance Clustering* in an industry setting and the performance of the algorithms is similar to the performance reported on toy data. *Cupid* is very sensitive to thresholds and the F1-score is rather low, producing a large number of false positives. On the other hand, *Instance Clustering* is less sensitive to thresholds, being able to cluster similar columns together. To achieve a more accurate result, the thresholds and the number of quantiles have to be carefully tuned.

In conclusion, we can not expect the same results by relying on the descriptions presented in the original papers. The best approach is having the source code and at least a small subset from the same dataset. In the absence of them, we can only demonstrate similar behavior.

Chapter 6

Conclusion

Over the past years, the reproducibility issue has been addressed more and more in various domains. The academic area encouraged reproducibility by offering awards or by rejecting the papers that are not reproducible. However, in the data management community, adopting the guidelines and rules for a successful reproducible paper is still a process in development. In this research, we address the issue by conducting a reproducibility study on the most representative state-of-the-art schema matching papers.

Firstly, we presented a literature survey on schema matching techniques on a period of 24 years, from 1994 until 2018. The survey helped us understand the problems solved by schema matching, which could be classified in three large categories based on the popularity of the database solutions: the multiple database era - when schema matching was employed for federated databases and unified knowledge bases, the general approach era - when the focus changed from particular problems to more general applications, independent of a specific issue and the big data era - when people are concerned with data curation and data discovery.

Based on the solutions described in the literature survey, we created a detailed taxonomy and we identified the most important three categories of schema matching techniques: schema-based - that uses only the schema information, instance-based - that uses only the data instances and the combination of both. According to this classification, we have identified the most representative paper for each category and proceed with the reproducibility study. The selected papers are: *Cupid* [Madhavan, Bernstein, and Rahm, 2001], *SemProp* [Fernandez et al., 2018b] and *Instance Clustering* [Zhang et al., 2011].

The study presents three types of reproducibility: experimental reproducibility - where the same results are achieved using the same implementation and data as in the original paper, data reproducibility - where the same results are achieved by using the same implementation, but different data and methods reproducible - where the same results are achieved by using an alternative implementation and different data. All three papers proved to be only methods reproducible, because they do not indicate the source code, the data used in the experiments, nor the experimental pipeline. As such, a high score for the methods reproducible category was expected, as the category presents the most relaxed requirements. Furthermore, we proceeded with the implementation of the algorithms and presented the implementation design decision in comparison with the specifications from the original papers.

Finally, our reproducibility study concludes with the evaluation section. Compared with the original paper, we proved similar behaviors for all the algorithms implemented. Moreover, we performed an industry case study, where we evaluated the algorithms using industry data. The results were similar to the ones achieved with the synthetic dataset and the company was satisfied with the outcome. However, considering the fact that the algorithms require careful maintenance and an abundance of resources, the company considered that they are not sustainable.

In conclusion, the data management community needs to evolve and assume the reproducibility as part of the requirements for publishing a successful paper. Furthermore, more studies should be conducted to assess the evolution of reproducible research and encourage the community to share the source code and data. As a first step in this direction, our code is publicly available on Github[1]:
https://github.com/AndraIonescu/reproducing-schema-matching.

6.1 Future work

Extend the reproducibility analysis

One limitation of the research is represented by the small number of papers that were subject to the reproducibility analysis. Therefore, the analysis can be extended by including the rest of the papers from the literature survey. Moreover, only the papers that achieve the highest score for the experiment reproducibility metric should be implemented and evaluated. The result of the extended analysis would bring new insights about the status and degree of reproducibility in the data management community and would raise awareness about its importance.

Integrate the schema matching solutions

The variety of schema matching algorithms presented in the literature survey is emphasizing the importance of having a solution to decide upon an algorithm that performs the best under a specific scenario. One solution should be a dedicated platform to host all the open-source algorithms and the reproduced ones. Therefore, multiple implementations of the same algorithms can undergo quality measurements such as accuracy, execution time and complexity. Finally, the platform can increase the visibility and importance of the reproducibility issue in the data management community.

Benchmarking framework

Another solution to the problem of having multiple schema matching algorithms without a performance indicator is the existence of a dedicated benchmarking framework. Therefore, different algorithms can be evaluated on specific datasets and report the performance metrics such as precision, recall, and F1-score. The results will help to decide which algorithm is the best for a given dataset.

[1] https://github.com/

Bibliography

Baker, Monya (2016). “1,500 scientists lift the lid on reproducibility”. In: *Nature News* 533.7604, p. 452.

Batini, Carlo, Maurizio Lenzerini, and Shamkant B. Navathe (1986). “A comparative analysis of methodologies for database schema integration”. In: *ACM computing surveys (CSUR)* 18.4, pp. 323–364.

Bellahsene, Zohra et al. (2011). “On evaluating schema matching and mapping”. In: *Schema matching and mapping*. Springer, pp. 253–291.

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

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

Bilke, Alexander and Felix Naumann (2005). “Schema matching using duplicates”. In: *21st International Conference on Data Engineering (ICDE’05)*. IEEE, pp. 69–80.

Castano, Silvana and Valeria De Antonellis (1999). “Deriving global conceptual views from multiple information sources”. In: *Conceptual Modeling*. Springer, pp. 44–55.

— (2001). “Global viewing of heterogeneous data sources”. In: *IEEE Transactions on Knowledge and Data Engineering* 13.2, pp. 277–297.

Chaudhuri, Surajit, Venkatesh Ganti, and Raghav Kaushik (2006). “A primitive operator for similarity joins in data cleaning”. In: *22nd International Conference on Data Engineering (ICDE’06)*. IEEE, pp. 5–5.

Clifton, Chris et al. (2004). “Privacy-preserving data integration and sharing”. In: *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, pp. 19–26.

Collberg, Christian, Todd Proebsting, and Alex M Warren (2015). “Repeatability and benefaction in computer systems research”. In: *University of Arizona TR* 14, p. 4.

Cortez, Eli et al. (2015). “Annotating database schemas to help enterprise search”. In: *Proceedings of the VLDB Endowment* 8.12, pp. 1936–1939.

Dhamankar, Robin et al. (2004). “iMAP: discovering complex semantic matches between database schemas”. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, pp. 383–394.

Do, Hong-Hai, Sergey Melnik, and Erhard Rahm (2002). “Comparison of schema matching evaluations”. In: *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*. Springer, pp. 221–237.

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

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

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

Dong, Xin Luna and Divesh Srivastava (2013). "Big data integration". In: *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, pp. 1245–1248.

Duchateau, Fabien, Zohra Bellahsene, and Ela Hunt (2007). "XBenchMatch: a benchmark for XML schema matching tools". In: *The VLDB Journal*. Vol. 1. Springer Verlag, pp. 1318–1321.

Fernandez, Raul Castro et al. (2018a). "Aurum: A data discovery system". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, pp. 1001–1012.

Fernandez, Raul Castro et al. (2018b). "Seeping semantics: Linking datasets using word embeddings for data discovery". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, pp. 989–1000.

Freire, Juliana, Philippe Bonnet, and Dennis Shasha (2012). "Computational reproducibility: state-of-the-art, challenges, and database research opportunities". In: *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. ACM, pp. 593–596.

Gaulton, Anna et al. (2017). "The ChEMBL database in 2017". In: *Nucleic acids research* 45.D1, pp. D945–D954.

Giunchiglia, Fausto, Pavel Shvaiko, and Mikalai Yatskevich (2004). "S-Match: an algorithm and an implementation of semantic matching". In: *European semantic web symposium*. Springer, pp. 61–75.

Goodman, Steven N, Daniele Fanelli, and John PA Ioannidis (2016). "What does research reproducibility mean?" In: *Science translational medicine* 8.341, 341ps12–341ps12.

Gower, John C (1985). "Measures of similarity, dissimilarity and distance". In: *Encyclopedia of Statistical Sciences, Johnson and CB Read* 5, pp. 397–405.

Gundersen, Odd Erik and Sigbjørn Kjensmo (2018). "State of the art: Reproducibility in artificial intelligence". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Halevy, Alon Y (2001). "Answering queries using views: A survey". In: *The VLDB Journal* 10.4, pp. 270–294.

Heimbigner, Dennis and Dennis McLeod (1985). "A federated architecture for information management". In: *ACM Transactions on Information Systems (TOIS)* 3.3, pp. 253–278.

Kang, Jaewoo and Jeffrey F Naughton (2003). "On schema matching with opaque column names and data values". In: *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, pp. 205–216.

Kondrak, Grzegorz (2005). "N-gram similarity and distance". In: *International symposium on string processing and information retrieval*. Springer, pp. 115–126.

Li, Wen-Syan and Chris Clifton (1994). "Semantic integration in heterogeneous databases using neural networks". In: *Proceedings of the 20th international conference on very large data bases*.

Madhavan, Jayant, Philip A Bernstein, and Erhard Rahm (2001). "Generic schema matching with cupid". In: *vldb*. Vol. 1, pp. 49–58.

Madhavan, Jayant et al. (2005). "Corpus-based schema matching". In: *21st International Conference on Data Engineering (ICDE'05)*. IEEE, pp. 57–68.

Manolescu, Ioana et al. (2008). "The repeatability experiment of SIGMOD 2008". In: *ACM SIGMOD Record* 37.1, pp. 39–45.

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

Miller, Renée J et al. (2001). "The Clio project: managing heterogeneity". In: *SIgMOD Record* 30.1, pp. 78–83.

Milo, Tova and Sagit Zohar (1998). "Using schema matching to simplify heterogeneous data translation". In: *vldb*. Vol. 98. Citeseer, pp. 24–27.

Mitra, Prasenjit, Gio Wiederhold, and Jan Jannink (1999). "Semi-automatic integration of knowledge sources". In: *Proceedings of Fusion'99, July 1999*.

Palopoli, Luigi, Domenico Sacca, and Domenico Ursino (1998). "Semi-automatic, semantic discovery of properties from database schemes". In: *Proceedings. IDEAS'98. International Database Engineering and Applications Symposium (Cat. No. 98EX156)*. IEEE, pp. 244–253.

Peng, Roger D (2011). "Reproducible research in computational science". In: *Science* 334.6060, pp. 1226–1227.

Popa, Lucian et al. (2002). "Translating web data". In: *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, pp. 598–609.

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

Rahm, Erhard and Andreas Thor (2005). "Citation analysis of database publications". In: *ACM Sigmod Record* 34.4, pp. 48–53.

Serafini, Luciano et al. (2003). *An algorithm for matching contextualized schemas via SAT*. Tech. rep. University of Trento.

Sheth, Amit P and James A Larson (1990). "Federated database systems for managing distributed, heterogeneous, and autonomous databases". In: *ACM Computing Surveys (CSUR)* 22.3, pp. 183–236.

Shvaiko, Pavel and Jérôme Euzenat (2005). "A survey of schema-based matching approaches". In: *Journal on data semantics IV*. Springer, pp. 146–171.

Stonebraker, Michael et al. (2013). "Data Curation at Scale: The Data Tamer System." In: *CIDR*.

Thulasiraman, Krishnaiyan and Madisetti NS Swamy (2011). *Graphs: theory and algorithms*. John Wiley & Sons.

Vitek, Jan and Tomas Kalibera (2011). "Repeatability, reproducibility and rigor in systems research". In: *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*. IEEE, pp. 33–38.

Yatskevich, Mikalai (2003). *Preliminary evaluation of schema matching systems*. Tech. rep. University of Trento.

Zhang, Meihui et al. (2010). "On multi-column foreign key discovery". In: *Proceedings of the VLDB Endowment* 3.1-2, pp. 805–814.

— (2011). "Automatic discovery of attributes in relational databases". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, pp. 109–120.