An Efficient Holistic Schema Matching Approach

Aola Yousfi(✉), Moulay Hafid El Yazidi, and Ahmed Zellou

Software Project Management Research Team, ENSIAS,
Mohammed V University in Rabat, Rabat, Morocco
aola.yousfi@gmail.com, {my-hafid.elyazidi,ahmed.zellou}@um5.ac.ma

Abstract. Schema matching allows a certain way of communication between heterogeneous, autonomous and distributed data sources. We choose the matching approach depending on the number of sources we wish to integrate: pairwise matching approaches for a small to a medium total number of data sources, and holistic matching approaches for a big to a huge number of data sources. Nevertheless, current matching approaches were proven to achieve a very moderate matching accuracy. Moreover, holistic matching approaches operate in a series of two-way matching steps. In this paper, we present hMatcher, an efficient holistic schema matching approach. To execute holistic schema matching, hMatcher captures frequent schema elements in the given domain prior to any matching operation. To achieve high matching accuracy, hMatcher uses a context-based semantic similarity measure. Experimental results on real-world domain show that hMatcher performs holistic schema matching properly, and outperforms current matching approaches in terms of matching accuracy.

Keywords: Holistic schema matching · Semantic similarity measure · Matching accuracy

1 Introduction

Schema matching has got a lot of attention from the research community over the past three decades (see [10,18] for surveys). It is very critical for applications that manipulate data across schemas of distinct data sources, examples of areas where this kind of applications are used include mainly data integration on the World Wide Web, data warehousing, e-commerce, scientific collaboration and bioinformatics. Another reason that makes schema matching very important is that it hides the semantic and syntactic heterogeneity between different schemas and makes the latter looks like one single integrated schema, which facilitates significantly access to data.

Schema matching and schema mapping are often used interchangeably, however they in fact refer to two different things. The former refers to the process of searching for semantically corresponding elements (also called matches or semantically similar elements [4,21,22]) in multiple, heterogeneous, autonomous and scattered schemas of data sources. Figure 1 illustrates an example of the schema matching problem: given two schemata S_1 and S_2 (for the sake of clarification, we depict them as trees) describing film information, we aim to determine the matches (indicated by dotted lines), which identify schema elements representing the same concepts in the two. The latter however refers to the process of defining the relationship between elements of different schemas. Also, schema matching takes place right before schema mapping, as the output of the former is used by the latter; also schema mapping cannot work without schema matching which again emphasizes the big importance and criticality of schema matching.

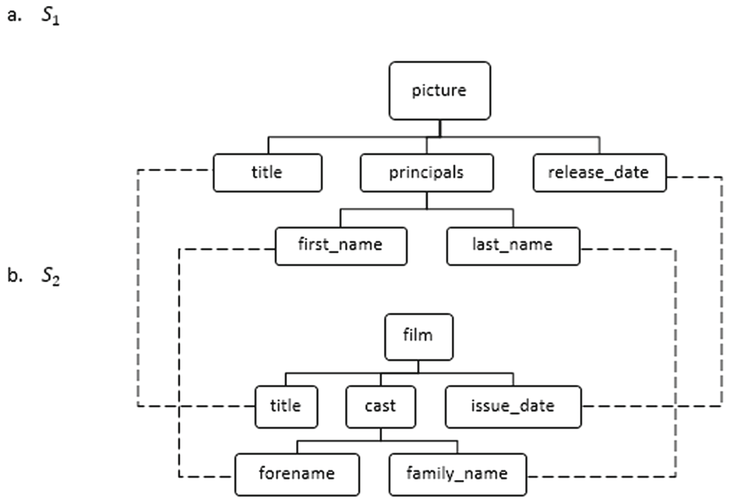

Fig. 1. Example of the schema matching problem

Researchers grouped schema matching approaches into two major categories: pairwise matching and holistic matching. The former aims at identifying the matches between two schemas at a time, which appears to be insufficient when it comes to matching a huge number of schemas. Thus, the latter came to place as it aims to match multiple schemas simultaneously, which intends to overcome the aforementioned limitations faced by pairwise schema matching approaches.

Current holistic schema matching approaches, also called collective schema matching approaches [15], (see Sect. 2) often proceed in a series of two-way matching steps which contradict with the initial goal of holistic schema matching as they do not necessarily match schemas simultaneously, but instead they match schemas incrementally: they first combine two schemas into one integrated schema using the matches between them, and then matches one schema at a time to the combined schema (e.g. PORSCHE).

Current matching approaches often reach moderate matching accuracy. This implies a continuous human assistance to correct the final results: remove false matches and add missed matches; which impacts the overall performance of the matching system in question.

The main concrete challenges, we addressed when working on this research project, are listed below:

a. Define an appropriate approach to capture frequent schema elements.
c. Define a method to reduce the total number of rare schema elements.
d. Come up with an efficient approach to holistic schema matching.

In this paper, we introduce hMatcher, an efficient holistic schema matching approach. The main idea of hMatcher is to (1) execute holistic schema matching; and (2) reach a high matching accuracy. To do so, hMatcher employs a semantic similarity measure, uses the transitivity principle, and exploits a hierarchical lexical dictionary along with an abbreviations & acronyms database.

In summary, we make the following concrete contributions:

a. We propose an algorithm to identify frequent schema elements in a particular domain.
b. We propose an approach to match multiple schemas holistically.
c. We evaluate hMatcher on a real-world domain and show that it is able to match multiple schemas at once and produce high matching accuracy.

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 defines the problem of holistic schema matching. Section 4 describes the architecture of hMatcher. Section 5 presents experimental results. Section 6 concludes this paper and discusses future research work.

2 Related Work

In this section, we review the state of the art schema matching methods that are most relevant to our work.

AgreementMakerLight (AML) [6] is an ontology matching approach. It derives from AgreementMaker [2]. AML consists of two key modules: the ontology loading module and the ontology matching module. The ontology loading module loads the ontologies and the external resources, and then generates the ontology objects. The ontology matching module main objective is to align the ontology objects generated by the previous module.

ALIN [17] is a human-interactive ontology matching approach. It takes as input two ontologies and outputs a set of alignments. It goes through two key steps. First, it defines the initial mappings. Second, it waits for human expert feedbacks and changes the mappings accordingly so as to improve the quality of the final results. The second step is repeated till experts run out of suggestions.

ALOD2Vec [14] uses the WebIsALOD database of hypernym relations extracted from the Web. It also employs both element-based information and label-based information. In order to determine the similarity score between nodes of the knowledge graph (WebIsALOD is viewed as a knowledge graph), ALOD2Vec applies RDF2Vec which transforms RDFs into vectors.

Deep Ontology MatchEr (DOME) [7] uses doc2vec and exploits large texts that describe the concepts of the ontologies. To deal with the main issue of matching similar large texts, DOME exploits topic modeling for instance Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA).

FCAMapX [1] is an automated ontology alignment system. It is based on Formal Concept Analysis, which is a mathematical model for analyzing individuals and structuring concepts.

KEPLER [9] is an ontology matching system which takes advantage of the expressiveness of the Web Ontology Language (OWL) statements by means of six complementary steps: parsing, partitioning/translation, indexing, candidate mappings identification, filtering and recovery, and alignment generation.

Lily [19] is an ontology alignment approach. Its main advantage is that it is able to process normal ontologies, weak informative ontologies, ontology mapping debugging and ontology matching tuning [20].

LogMap [8] is a scalable and logic-based ontology matching approach. It uses lexical indexation, logic-based module extraction, propositional horn reasoning, axiom tracking, local repair and semantic indexation to match two given ontologies. LogMapLt is a lightweight variant of LogMap, which essentially applies string-based matching techniques.

Simulated ANnealing-based Ontology Matching (SANOM) [12] uses the notorious Simulated Annealing (SA) [13] to discover semantically similar elements between two input ontologies, which results on a potential intermediate alignment. The evolution of that alignment needs the use of both lexical similarity metrics and structural similarity metrics.

Holontology [16] is a holistic ontology matching approach based on the Linear Program for Holistic Ontology Matching (LPHOM) approach [11]. It exploits a combination of several similarity measures and dissimilarity distances: exact match, Levenstein, Jaccard and Lin to match two ontologies or multiple ontologies at once after it converts them into an internal predefined format. Then, Holontology converts the results into alignments exported by RDF.

Current holistic schema matching methods face several challenges. First, they are unable to perform collective schema matching properly: they match schemas incrementally in a series of two-way matching steps. Second, they reach a moderate matching accuracy (see Sect. 5 for more details). Finally, they are human-dependent.

In the next section, we will state the problem we are working on in this paper and present related definitions.

3 Problem Statement

In this section, we first provide definitions related to the problem we are working on in this paper; we then state the research question; and finally we display the notations used throughout this paper.

Definition 3.1. (Set of words). A set of words $\theta = \{w_1, w_2, \ldots, w_{|\theta|}\}$ includes words extracted from a schema element e. In Sect. 4, we will describe thoroughly the process of generating a set of words.

Remark. All the words' sets generated from a schema S are denoted by Θ.

Definition 3.2. (Semantically corresponding elements). Let S_1, S_2, S_3 be three given schemata. Semantically corresponding elements φ is a tuple (E_1, E_2, E_3), where E_1, E_2 and E_3 are three subsets of schema elements from S_1, S_2 and S_3 respectively.

Definition 3.3. (Learning schemas). The learning schemas are the schemas we use to identify the initial set of frequent schema elements in the given domain. We denote the set of learning schemas by $\mathbb{S}_{Learning}$.

Definition 3.4. (Testing schemas). The testing schemas are the schemas we want to match using the frequent schema elements. We denote the set of testing schemas by $\mathbb{S}_{Testing}$.

Definition 3.5. (Frequent schema element.) Given a schema S, let f be an element from S. f is a frequent schema element if and only if it has duplicates in a certain number of schemas describing the same domain. Note that the duplicates of f can either be f as well or semantic corresponding elements of f. We denote the set of frequent schema elements by $\mathbb{F}$.

Definition 3.6. (Rare schema element). Given n schemas $S_1, S_2, \ldots, S_n$ we wish to match. Let E be the set of elements contained in $S_1, S_2, \ldots,$ and S_n, and let $\mathbb{F}$ be the set of frequent schema elements. A rare schema element $r \in E$ is a schema element that does not belong to $\mathbb{F}$. We denote the set of rare schema elements by $\mathbb{R}$.

Definition 3.7. (Problem statement). Given n input schemas $S_1, S_2, \ldots, S_n$ we wish to match. Our goal in this paper is to identify the semantic correspondences $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_{|\Phi|}\}$ between $S_1, S_2, \ldots, S_n$ as a result of matching $S_1, S_2, \ldots, S_n$ simultaneously (rather than two at a time). The matching approach also has to ensure superior matching accuracy.

The list of symbols used throughout this paper is given in Table 1.

Table 1. Symbols used throughout this paper

Symbol	Description
S, e	Schema, schema element
Θ, $\lvert\Theta\rvert$, θ, $\lvert\theta\rvert$	Sets of words, cardinality of Θ, a set of words in Θ, cardinality of θ
$\mathbb{S}_{Learning}$, $\lvert\mathbb{S}_{Learning}\rvert$	Learning schemas, cardinality of $\mathbb{S}_{Learning}$
$\mathbb{S}_{Testing}$, $\lvert\mathbb{S}_{Testing}\rvert$	Testing schemas, cardinality of $\mathbb{S}_{Testing}$
$\mathbb{S}$	$\mathbb{S} = \mathbb{S}_{Learning} \cup \mathbb{S}_{Testing}$
Φ, φ	Matches, a tuple of matches in Φ
$\mathbb{F}$, f, $\lvert\mathbb{F}\rvert$	Frequent elements list, a frequent schema element, cardinality of $\mathbb{F}$
$\mathbb{R}$, r, $\lvert\mathbb{R}\rvert$	Rare elements list, a rare schema element, cardinality of $\mathbb{R}$

In the next section, we will describe the hMatcher solution to the problem described in Definition 3.7.

4 The hMatcher Approach

The hMatcher architecture (see Fig. 2) is composed of three different components: the frequent elements generator, the schema matcher and the rare elements matcher. Let $\mathbb{S}_{Learning} \in \mathbb{S}$ be the learning schemas. First, the frequent elements generator takes as input $\mathbb{S}_{Learning}$, utilizes both a hierarchical lexical dictionary and an abbreviations & acronyms database, and provides as output the frequent schema elements $\mathbb{F}$. Let $\mathbb{S}_{Testing} \in \mathbb{S}$ be the testing schemas. The schema matcher then takes in $\mathbb{S}_{Testing}$, employs the frequent schema elements, and provides as output the matches Φ. Finally, the rare elements matcher reuses the results to find out new potential matches in the rare schema elements $\mathbb{R}$.

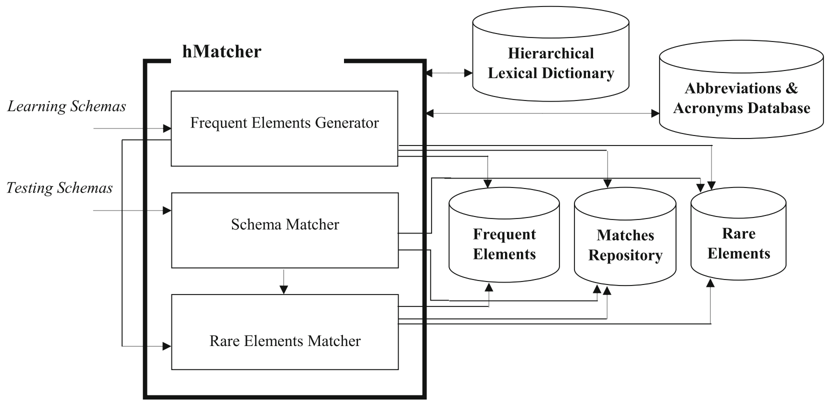

Fig. 2. The hMatcher Architecture

In the rest of this section, we first present the frequent elements generator (see Sect. 4.1), we then describe the schema matcher (see Sect. 4.2), and finally, we introduce the rare elements matcher (see Sect. 4.3).

4.1 The Frequent Elements Generator

Let $\mathbb{S}_{Learning} = \{S_1, S_2, \ldots, S_p\} \in \mathbb{S}$ be the learning schemas, the frequent elements generator employs the pre-processing method presented in [23–25] in order to extract from each schema element e a words set θ that represents its meaning. That pre-processing method proceeds as follows. It first extracts words from the schema elements using the lexical dictionary. It then substitutes abbreviations and acronyms with their corresponding full forms using the abbreviations & acronyms database. It next generates words sets. Finally, it identifies the meaning of words based on their context (a word often changes its meaning in different contexts). Note that the words sets generated from S_1 are denoted by Θ_1, the words sets generated from S_2 are denoted by Θ_2, etc. Later, the frequent elements generator operates in three different steps:

- **Step 1: Capture the matches.** Let $e_1 \in S_1$ and $e_2 \in S_2$ be two elements, and let $\theta_1 = \{w_{1,1}, w_{1,2}, \ldots, w_{1,|\theta_1|}\}$ and $\theta_2 = \{w_{2,1}, w_{2,2}, \ldots, w_{2,|\theta_2|}\}$ be their respective words sets. The frequent elements generator uses the measure (CSSM) in [24] to see whether θ_1 and θ_2 are semantically similar or not.
- **Step 2: Determine frequent schema elements.** Let $S \in \mathbb{S}_{Learning}$ be a schema and let e be an element from S. We use *EF–SF* (Element Frequency-Schema Frequency) (1) to find the degree of frequency of e.

$$EF - SF_{e \in S, \mathbb{S}_{Learning}} = EF_{e,S} \times SF_{e,\mathbb{S}_{Learning}} = e^{ef_{e,S}} \times e^{\left(\frac{sf_e}{|\mathbb{S}_{Learning}|}\right)} \quad (1)$$

Where:
  - $ef_{e,S}$ is the frequency of e in S, such that $ef_{e,S} = \frac{count\ of\ e\ in\ S}{number\ of\ elements\ in\ S}$.
  - sf_e is the number of schemas containing e.
  - $|\mathbb{S}_{Learning}|$ is the cardinality of $\mathbb{S}_{Learning}$ ($|\mathbb{S}_{Learning}| = p$).

  We say that a schema element e is frequent if and only if its degree of frequency satisfies the following criteria:

$$\lfloor EF - SF_{e \in S, \mathbb{S}_{Learning}} \rfloor >= log\left(\frac{\sqrt[2]{(m^4+1)}}{\sqrt[2]{|\mathbb{S}_{Learning}|^2 - 1}}\right) \quad (2)$$

Where:
  - m is the total number of elements in $\mathbb{S}_{Learning}$.
- **Step 3: Examine if there are more frequent elements.** If we add extra schemas to the learning schemas $\mathbb{S}_{Learning}$ but we end up having the same exact frequent schema elements ($|\mathbb{F}| = constant$), then the frequent elements generator stops. Otherwise, it repeats both steps 1 and 2.

Algorithm 1 summarizes this.

4.2 The Schema Matcher

Let $\mathbb{S}_{Testing} = \{S_{p+1}, S_{p+2}, \ldots, S_n\}$ be the testing schemas, and let $\Theta_{p+1}, \Theta_{p+2}, \ldots, \Theta_n$ be the words sets generated from $S_{p+1}, S_{p+2}, \ldots, S_n$, respectively. The schema matcher proceeds as follows (summarized in Algorithm 2).

- **Step 1: Compute the semantic similarity.** It compares the words sets $\Theta_{p+1}, \Theta_{p+2}, \ldots, \Theta_n$ to the frequent schema elements $\mathbb{F}$ according to the semantic similarity measure in [24].
- **Step 2: Find out new matches.** Every set $\theta_i \in \{\Theta_{p+1}, \Theta_{p+2}, \ldots, \Theta_n\}$ that has a semantically corresponding element f_i in $\mathbb{F}$, its schema element e_i will be added to the matches list Φ such that $\varphi \leftarrow \varphi \cup e_i$, where $f_i \in \varphi$ and $\varphi \in \Phi$.

Algorithm 1. FrequentElementsGenerator($\Theta_1, \Theta_2, \ldots, \Theta_p$)

```
Input:
  $\Theta_1, \Theta_2, \ldots, \Theta_p$: The sets of words generated from $S_1, S_2, \ldots, S_p$
Output:
  $\mathbb{F}$ : The frequent schema elements

  $\mathbb{F} \leftarrow \emptyset$
  Generate the matches $\Phi$ between $\Theta_1, \Theta_2, \ldots, \Theta_p$ according to the measure in [24]
  FOR each $\varphi$ in $\Phi$
  IF ($e \in \varphi$ AND $\lfloor EF - SF_{e \in S, \mathbb{S}_{Learning}} \rfloor >= log\left(\frac{\sqrt[2]{(m^4+1)}}{\sqrt[2]{|\mathbb{S}_{Learning}|^2-1}}\right)$) THEN
   $\mathbb{F} \leftarrow \mathbb{F} \cup e$ /* $\mathbb{F}$ stores one single element $e$ in $\varphi$ */
  END IF
  END FOR
return $\mathbb{F}$
```

Algorithm 2. SchemaMatcher($\Theta_{p+1}, \Theta_{p+2}, \ldots, \Theta_n$)

```
Input:
  $\Theta_{p+1}, \Theta_{p+2}, \ldots, \Theta_n$ : The words sets generated from $S_{p+1}, S_{p+2}, \ldots, S_n$
Output:
  $\Phi$ : The matches

  FOR each $\Theta$ in $\{\Theta_{p+1}, \Theta_{p+2}, \ldots, \Theta_n\}$
   Generate the matches $\Phi$ between $\Theta$ and $\mathbb{F}$ according to the semantic similarity measure in [24]
  END FOR
return $\Phi$
```

4.3 The Rare Elements Matcher

Theorem 1. (Transitive relation). A binary relation $\Re$ is transitive over a set A if and only if it satisfies the following criteria:

$\forall\ x, y, z \in A$, $(x \Re y \wedge y \Re z) \implies x \Re z$, i.e. for all elements x, y, z in A, whenever $\Re$ relates x to y and y to z, then $\Re$ also relates x to z.

Let S_1 and S_2 be two schemas, let $r_1 \in \mathbb{R}$ and $r_2 \in \mathbb{R}$ be two rare schema elements from S_1 and S_2, respectively; and let $\mathbb{F} = \{f_1, f_2, \ldots, f_q\}$ (where $q \in \mathbb{N}^*$) be the set of frequent schema elements. Using the transitive relation, we have the following:

$$\forall\ i \in \{1, 2, \ldots, q\}, CSSM(r_1, f_i) = CSSM(r_2, f_i) \pm 0.05 \implies r_1 \textit{ and } r_2 \textit{ are matched.}$$

If r_1 (or r_2) satisfies (2), then the frequent schema elements list is updated as follows:

$$\mathbb{F} \leftarrow \mathbb{F} \cup r_1 \text{ OR } \mathbb{F} \leftarrow \mathbb{F} \cup r_2 \text{ (not both).}$$

And the list of rare schema elements is updated as follows:

$$\mathbb{R} \leftarrow \mathbb{R} \setminus r_1 \text{ AND } \mathbb{R} \leftarrow \mathbb{R} \setminus r_2.$$

Algorithm 3 summarizes this.

Algorithm 3. RareElementsMatcher($\mathbb{R}$, $\mathbb{F}$)

```
Input:
   $\mathbb{R}$ : *Rare schema elements*
   $\mathbb{F}$ : *Frequent schema elements*
Output:
   $\Phi$, $\mathbb{F}$, $\mathbb{R}$ : *The matches; the frequent schema elements; the rare elements*

   FOR each $r_1$ and $r_2$ in $\mathbb{R}$
   IF ($CSSM(r_1, f) = CSSM(r_2, f) \pm 0.05$, $\forall\ f \in \mathbb{F}$) THEN
    $\varphi \leftarrow \varphi \cup r_1$ /* $\varphi \in \Phi$ *is the tuple that includes the matches of* $r_1$ */
    $\mathbb{F} \leftarrow \mathbb{F} \cup r_1$; $\mathbb{R} \leftarrow \mathbb{R} \setminus r_1$; $\mathbb{R} \leftarrow \mathbb{R} \setminus r_2$
   END IF
   END FOR
return ($\Phi, \mathbb{F}, \mathbb{R}$)
```

5 Experiments and Evaluations

In this section, we evaluate hMatcher in terms of matching accuracy and compare the results to other matching approaches.

5.1 Datasets

We evaluated hMatcher on the *Conference* dataset which contains 16 ontologies describing the domain of organizing academic conferences. The ontologies were used in OAEI 2019 and are publicly available on the Web[1]. The *Conference* dataset has been used by the research community for over thirteen years. It has 21 reference alignments composed from 7 out of 16 real domain ontologies.

5.2 Measures

We first exploit *Precision* (3), *Recall* (4), *Overall* (5) and $F-Measure$ (6) to evaluate the matches generated by hMatcher on the *Conference* dataset.

$$Precision = \frac{Correct\ Matches}{Correct\ Matches + Incorrect\ Matches} \tag{3}$$

(3) is the probability of correct matches among all matches returned by the matching system.

$$Recall = \frac{Correct\ Matches}{Missed\ Matches + Correct\ Matches} \tag{4}$$

(4) is the probability of correct matches returned by a matching system among the reference matches.

$$Overall = Recall \times (2 - \frac{1}{Precision}) \tag{5}$$

[1] http://oaei.ontologymatching.org/2019/.

(5) measures the amount of manual post-effort required to remove false matches and add missed matches. Unlike *Precision* and *Recall*, *Overall* can have negative values if $Precision < 0.5$. Note that if $Overall < 0$ then most matches must be produced manually, concluding that the matching system is not interesting.

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

(6) is the harmonic mean of *Precision* and *Recall*.

We then compare the results obtained on the *Conference* dataset against previously published results of twelve well-known and high-accuracy ontology matching systems (SANOM [12], AML [5], LogMap [8], XMap [3], KEPLER [9], ALIN [17], DOME [7], Holontology [16], FCAMapX [1], LogMapLt [8], ALOD2Vec [14] and Lily [19]). The evaluations are based on nine combinations of variants with crisp reference alignments: *ra1-M1*, *ra1-M2*, *ra1-M3*, *ra2-M1*, *ra2-M2*, *ra2-M3*, *rar2-M1*, *rar2-M2* and *rar2-M3* (*ra1* is the original reference alignment; *ra2* is an extension of *ra1*; and *rar2* is an updated version of *ra2* that deals with the violations of conservativity). *ra1-M1*, *ra2-M1* and *rar2-M1* are used to evaluate alignments between classes; *ra1-M2*, *ra2-M2* and *rar2-M2* are used to evaluate alignments between properties; and *ra1-M3*, *ra2-M3* and *rar2-M3* are used to evaluate both alignments between classes and properties.

5.3 Results and Discussions

Figs. 3, 4, 5, 6, 7, 8, 9, 10 and 11 depict the new and previously published results on the *Conference* dataset.

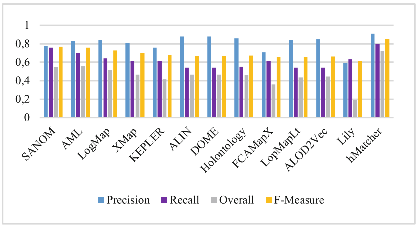

Fig. 3. ra1-M1: matching accuracy

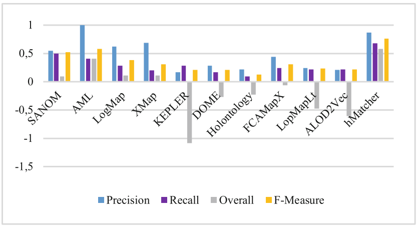

Fig. 4. ra1-M2: matching accuracy

On the one hand, the previously published findings indicate visible changes for *Precision*, *Recall*, *Overall* and *F-Measure*. They achieved high matching accuracy when evaluated based on *ra1-M1*, *ra1-M3*, *ra2-M1*, *ra2-M3*, *rar2-M1* and *rar2-M3*; and low matching accuracy even null in some situations (e.g. Lily and ALIN) with *ra1-M2*, *ra2-M2* and *rar2-M2*. On the other hand, hMatcher reached superior matching accuracy, which implies that it surpassed current matching systems almost every time with the exception of *ra1-M2* and *ra2-M2* where AML surpassed it slightly ($Precision = 1$).

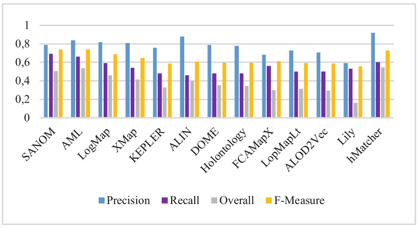

Fig. 5. ra1-M3: matching accuracy

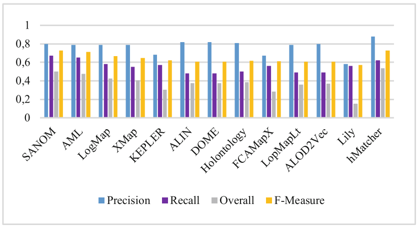

Fig. 6. ra1-M2: matching accuracy

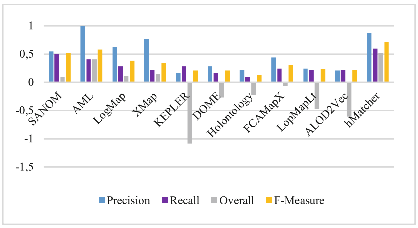

Fig. 7. ra2-M2: matching accuracy

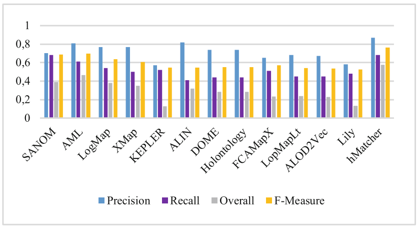

Fig. 8. ra1-M3: matching accuracy

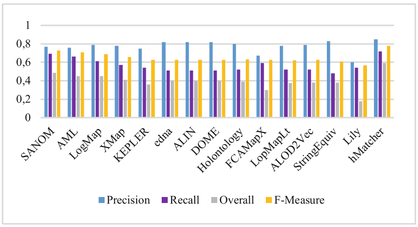

Fig. 9. rar2-M1: matching accuracy

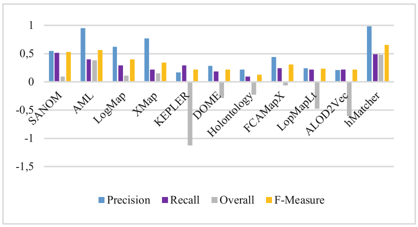

Fig. 10. rar2-M2: matching accuracy

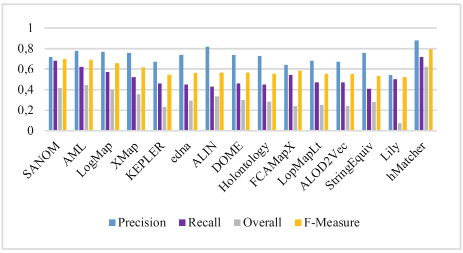

Fig. 11. rar2-M3: matching accuracy

Lily and ALIN match only classes, as a result, they failed to reach a high-matching accuracy with *ra1-M2*, *ra2-M2* and *rar2-M2*; SANOM, AML, LogMap and XMap match some but not all properties which explains their negative *Overall* with *ra1-M2*, *ra2-M2* and *rar2-M2*; KEPLER, DOME, Holontology, FCAMapX, LogMapLt and ALOD2Vec match very few properties which justifies their negative *Overall* and inferior *Precision*, *Recall* and *F-Measure* with *ra1-M2*, *ra2-M2* and *rar2-M2*; and hMatcher matches both classes and properties which justifies its positive *Overall* and high *Precision*, *Recall* and *F-Measure* with *ra1-M1*, *ra1-M2*, *ra1-M3*, *ra2-M1*, *ra2-M2*, *ra2-M3*, *rar2-M1*, *rar2-M2* and *rar2-M3*. Thus, we sum up that: (1) SANOM, AML, LogMap, XMap, KEPLER, ALIN, DOME, Holontology, FCAMapX, LogMapLt, ALOD2Vec and Lily work well with the reference alignments that consider either classes or both classes and properties. Nonetheless, they fail to match correctly with the reference alignments that consider only properties; and (2) hMatcher accomplishes better results as it achieves a superior matching accuracy regardless of the reference alignments it is compared to.

6 Conclusion and Future Work

We showed that carefully defining a holistic schema matching approach is mandatory to deliver accurate results and match a huge number of schemas. Current matching approaches often reach low matching accuracy and thus require human assistance to correct the matches; moreover, existing matching approaches are more likely to match two schemas at a time rather than many at once.

Let $\mathbb{S}_{Learning}$ be the learning schemas and $\mathbb{S}_{Testing}$ be the testing schemas, hMatcher generates frequent schema elements $\mathbb{F}$ from $\mathbb{S}_{Learning}$. It then uses $\mathbb{F}$ to capture new matches Φ in $\mathbb{S}_{Testing}$. At the end, hMatcher reuses previous results to identify new matches among the few rare schema elements we are left with.

We evaluated hMatcher on the conference dataset, the results show a high matching accuracy reached by hMatcher on the one hand; and on the other hand, an inferior matching accuracy obtained by current matching approaches.

Future interesting research directions include mainly the following:

- **Consider situations where schemas are represented using different lexical languages.** In this dissertation, we focused mainly on schemas expressed in the same lexical language. An interesting future research direction would be to match schemas regardless of the lexical language they are expressed in. A way to do that would be to implement a translator.
- **How does hMatcher impact data source selection and ordering?** Prior to returning the answer to the user, the system selects a subset of data sources that are more likely to include the answer or a piece of the answer to the user query, this process is what we call source selection; then, the system orders data sources in a descending order of their coverage (source coverage is the amount of answers to a particular query contained in the data source), this process is called source ordering. So, a future research direction would be to study the relationship between hMatcher and data source selection and ordering.

References

1. Chen, G., Zhang, S.: Fcamapx results for OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-Located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 160–166 (2018) http://ceur-ws.org/Vol-2288/oaei18_paper7.pdf
2. Cruz, I.F., Antonelli, F.P., Stroe, C.: Agreementmaker: efficient matching for large real-world schemas and ontologies. PVLDB **2**(2), 1586–1589 (2009). https://doi.org/10.14778/1687553.1687598, http://www.vldb.org/pvldb/vol2/vldb09-1003.pdf
3. Djeddi, W.E., Yahia, S.B., Khadir, M.T.: Xmap: results for OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, 8 October 2018, pp. 210–215 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper15.pdf
4. El Yazidi, M.H., Zellou, A., Idri, A.: Towards a fuzzy mapping for mediation systems. In: 2012 IEEE International Conference on Complex Systems (ICCS), pp. 1–4 (2012)
5. Faria, D., et al.: Results of AML participation in OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 125–131 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper2.pdf
6. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The AgreementMakerLight ontology matching system. In: Meersman, R. (ed.) OTM 2013. LNCS, vol. 8185, pp. 527–541. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41030-7_38
7. Hertling, S., Paulheim, H.: DOME results for OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 144–151 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper5.pdf
8. Jiménez-Ruiz, E., Grau, B.C., Cross, V.: Logmap family participation in the OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 187–191 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper11.pdf
9. Kachroudi, M., Diallo, G., Yahia, S.B.: KEPLER at OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 173–178 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper9.pdf
10. Köpke, J.: Annotation paths for matching xml-schemas. Data Knowl. Eng. **122**, 25–54 (2019). https://doi.org/10.1016/j.datak.2017.12.002
11. Megdiche, I., Teste, O., Trojahn, C.: An extensible linear approach for holistic ontology matching. In: Groth, P. (ed.) ISWC 2016. LNCS, vol. 9981, pp. 393–410. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46523-4_24
12. Mohammadi, M., Hofman, W., Tan, Y.: SANOM results for OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 205–209 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper14.pdf

13. Mohammadi, M., Hofman, W., Tan, Y.: Simulated annealing-based ontology matching. ACM Trans. Manage. Inf. Syst. **10**(1), 3:1–3:24 (2019). https://doi.org/10.1145/3314948
14. Portisch, J., Paulheim, H.: Alod2vec matcher. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 132–137 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper3.pdf
15. Rahm, E., Peukert, E.: Large-scale schema matching. Encyclopedia of Big Data Technologies (2019). https://doi.org/10.1007/978-3-319-63962-8_330-1
16. Roussille, P., Megdiche, I., Teste, O., Trojahn, C.: Holontology: results of the 2018 OAEI evaluation campaign. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 167–172 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper8.pdf
17. da Silva, J., Revoredo, K., Baião, F.A.: ALIN results for OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 117–124 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper1.pdf
18. Sutanta, E., Wardoyo, R., Mustofa, K., Winarko, E.: Survey: models and prototypes of schema matching. Int. J. Electr. Comput. Eng. (2088–8708) **6**(3), 1011–1022 (2016)
19. Tang, Y., Wang, P., Pan, Z., Liu, H.: Lily results for OAEI 2018. In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, 8 October 2018, pp. 179–186 (2018). http://ceur-ws.org/Vol-2288/oaei18_paper10.pdf
20. Yang, P., Wang, P., Ji, L., Chen, X., Huang, K., Yu, B.: Ontology matching tuning based on particle swarm optimization: preliminary results. In: Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., Pan, J.Z. (eds.) CSWS 2014. CCIS, vol. 480, pp. 146–155. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45495-4_13
21. Yazidi, M.H.E., Zellou, A., Idri, A.: FMAMS: fuzzy mapping approach for mediation systems. IJAEC **4**(3), 34–46 (2013). https://doi.org/10.4018/jaec.2013070104
22. Yazidi, M.H.E., Zellou, A., Idri, A.: Fgav (fuzzy global as views). AIP Conf. Proc. **1644**(1), 236–243 (2015)
23. Yousfi, A., El Yazidi, M.H., Zellou, A.: hmatcher: Matching schemas holistically. Int. J. Intell. Eng. Syst. **13**(5), 490–501 (2020)
24. Yousfi, A., Elyazidi, M.H., Zellou, A.: Assessing the performance of a new semantic similarity measure designed for schema matching for mediation systems. In: Nguyen, N.T., Pimenidis, E., Khan, Z., Trawiński, B. (eds.) ICCCI 2018. LNCS (LNAI), vol. 11055, pp. 64–74. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98443-8_7
25. Yousfi, A., Yazidi, M.H.E., Zellou, A.: xmatcher: Matching extensible markup language schemas using semantic-based techniques. Int. J. Adv. Comput. Sci. Appl. **11**(8), 655–665 (2020). https://doi.org/10.14569/IJACSA.2020.0110880