Semantic Web 0 (0) 1
IOS Press

CANARD: An Approach for Generating Expressive Correspondences based on Alignment Need and ABox-based Relation Discovery

Elodie Thiéblin, Ollivier Haemmerlé, Cássia Trojahn *
Institut de Recherche en Informatique de Toulouse, Toulouse, France
E-mails: elodie@thieblin.fr, ollivier.haemmerle@irit.fr, cassia.trojahn@irit.fr

Abstract. Ontology matching aims at making ontologies interoperable. While the field has fully developed in the last years, most approaches are still limited to the generation of simple correspondences. More expressiveness is however required to better address the different kinds of ontology heterogeneities. This paper presents CANARD (Complex Alignment Need and A-box based Relation Discovery), an approach for generating expressive correspondences that relies on the notion of competency questions for alignment (CQA). A CQA expresses the user knowledge needs in terms of alignment and aims at reducing the alignment space. The approach takes as input a set of CQAs as SPARQL queries over the source ontology. The generation of correspondences is performed by matching the subgraph from the source CQA to the similar surroundings of the instances from the target ontology. Evaluation has been carried out on synthetic and real-word datasets.

Keywords: ontology matching, complex alignment, competency question for alignment, user needs

1. Introduction

Ontology matching is the task of generating a set of correspondences between the entities of different ontologies. This is the basis for a range of other tasks and applications, such as data integration, ontology evolution, and query rewriting. While the field has fully developed in the last decades, most works are still dedicated to the generation of simple correspondences (i.e., those linking one single entity of a source ontology to one single entity of a target ontology). However, simple correspondences are insufficient for covering the different kinds of ontology heterogeneities. More expressiveness is achieved by complex correspondences, which can better express the relationships between entities of different ontologies. For example, the piece of knowledge that a conference paper has been accepted can be represented as a class IRI *ekaw:Accepted_Paper* in a source ontology, or as a class expression representing the papers (the range of *cmt:hasDecision* is *cmt:Paper*) having a decision of type *cmt:Acceptance* in a target ontology. The correspondence $\langle$*ekaw:Accepted_Paper,* $\exists$*cmt:hasDecision.cmt:Acceptance,* $\equiv$, $\rangle$ expresses an equivalence between the two representations of "accepted paper".

Earlier works in the field have introduced the need for expressive alignments [1, 2], and different approaches for generating them have been proposed in the literature afterwards. These approaches rely on diverse methods,

*Corresponding author. E-mail: cassia.trojahn@irit.fr.

such as correspondence patterns [3–5], knowledge-rules [6] and association rules [7], statistical methods [8, 9], genetic programming [10] or still path-finding algorithms [11]. The reader can refer to [12] for a survey on complex matching approaches. All these proposals, however, intend to cover the full common scope of the ontologies and often need large number of common instances.

While the matching space set all possible correspondences between two ontologies) is not $\mathcal{O}(mn)$ as for simple alignment generation (m and n being respectively the number of entities of the source and target ontologies), but higher than $\mathcal{O}(2^{mn})$, a space reduction strategy can be based upon on two hypothesis. First, it may be the case that the user does not need the alignment to cover the full scope of the ontologies. Focusing on the user's need can reduce the search space. Reducing the search space definitely impacts the matching performance, in particular when dealing with large knowledge bases. The second hypothesis is that for each knowledge need, the ontologies share at least one instance.

This paper presents CANARD (Complex Alignment Need and A-box based Relation Discovery), a tool that discovers expressive correspondences[1] between populated ontologies based on the notion of Competency Questions for Alignment (CQAs). CQAs represent the user knowledge needs and define the scope of the alignment. They are competency questions that need to be satisfied over two or more ontologies. It takes as input a set of CQAs translated into SPARQL queries over the source ontology. The answer to each query is a set of instances retrieved from a knowledge base described by the source ontology. These instances are matched with those of a knowledge base described by the target ontology. The generation of the correspondence is performed by matching the subgraph from the source CQA to the lexically similar surroundings of the target instances. This paper extends the work in [13] in several directions: i) analysing the impact of each choice in the whole process (impact of the number of support answers, impact of using CQAs vs. queries, reassessment with counter-examples, etc.); (ii) extending the comparison of the approach with existing systems; and (iii) relying on such analysis for answering the hypothesis. The tool source code is available[2] under the GNU Lesser General Public License v2.1.

The rest of this paper is organised as follows. Next section introduces ontology matching and CQA (Section 2), followed by an overview of the proposed approach (Section 3). The details of the approach are then presented (Section 4). Next, the experiments are then presented (Section 5), followed by a discussion on the main related work (Section 7). Finally, the conclusions and future work end the paper (Section 8).

2. Foundations

2.1. Complex ontology alignment

Ontology matching (as in [14]) is defined as the process of generating an alignment A between two ontologies: a source ontology o and a target ontology o'. A is a set of correspondences $\langle e_1, e_2, r, n\rangle$. Each correspondence expresses a relation r (e.g., equivalence ($\equiv$), subsumption ($\sqsupseteq$, $\sqsubseteq$)) between two members e_1 and e_2, and n expresses its level of confidence $[0..1]$. A member can be a single ontology entity (class, object property, data property, individual) of respectively o and o' or a complex construction which is composed of entities using constructors. Two kinds of correspondences are considered depending on the type of their members:

- a correspondence is **simple** if both e_1 and e_2 are single entities (IRIs): $\langle o{:}Paper,\ o'{:}Paper,\ \equiv,\ 1\rangle$;
- a correspondence is **complex** if at least one of e_1 or e_2 involves a constructor: $\langle o{:}Accepted\_Paper,\ \exists o'{:}hasDecision.o'{:}Acceptance,\ \equiv,\ 1\rangle$.

A simple correspondence is noted (s:s), and a complex correspondence (s:c) if its source member is a single entity, (c:s) if its target member is a single entity or (c:c) if both members are complex entities. An approach which generates complex correspondences is referred as "complex approach" or "complex matcher" below.

[1] Correspondences involving logical constructions. Correspondences involving transformations functions are out of the scope of this paper.
[2] https://framagit.org/IRIT_UT2J/ComplexAlignmentGenerator

2.2. Competency Questions for Alignment (CQAs)

In ontology authoring, in order to formalise the knowledge needs of an ontology, competency questions (CQs) have been introduced as *ontology's requirements in the form of questions the ontology must be able to answer* [15]. A competency questions for alignment (CQA) is a competency question which should (in the best case) be covered by two or more ontologies, *i.e.*, it expresses the knowledge that an alignment should cover if both ontologies' scopes can answer the CQA. The first difference between a CQA and a CQ is that the scope of the CQA is limited by the intersection of its source and target ontologies' scopes. The second difference is that this maximal and ideal alignment's scope is not known *a priori*. As CQs [16], a CQA can be expressed in natural language or as SPARQL queries. Inspired from the predicate arity in [16], the notion of **question arity**, which represents the arity of the expected answers to a CQA is adapted, as introduced in [17]:

- a *unary* question expects a set of instances or values, *e.g.*, *Which are the accepted paper? (paper1), (paper2)*;
- a *binary* question expects a set of instances or value pairs, *e.g.*, *What is the decision of which paper? (paper1, accept), (paper2, reject)*; and
- an *n-ary* question expects a tuple of size 3 or more, *e.g.*, *What is the rate associated with which review of which paper? (paper1, review1, weak accept), (paper1, review2, reject)*.

In CANARD, CQAs are limited to *unary* and *binary*, of *select* type, and no modifier. This is a limitation in the sense that it does not deal with specific kinds of SPARQL queries, as the ones involving CONSTRUCT and ASK. The approach does not deal with transformation functions or filters inside the SPARQL queries and only accepts queries with one or two variables. However, as classes and properties are unary and binary predicates, these limitations still allow the approach to cover ontology expressiveness. Questions with a binary or counting type have a corresponding selection question. For example, the question *Is this paper accepted?* has a binary type: its answers can only be *True* or *False*. The question *How many papers are accepted?* is a counting question. These two questions have the same selection question: *What are the accepted papers?*. We also restrain the question polarity to *positive* because a negative question implies that a positive information is being negated. For example, the question *Which people are not reviewers?* is a negation of the question *Who are the reviewers?*. The CQA we consider have no modifier. The question arity of the CQA is limited to *unary* and *binary* because ontologies are mostly constructed using unary predicates (classes or class expressions) and binary predicates (object or data properties).

3. Overview of CANARD

CANARD takes as input a set of CQA in the form of SPARQL SELECT queries over the source ontology. It requires that the source and target ontologies have an $\mathcal{A}box$ with at least one common instance for each CQA. The answer to each input query is a set of instances, which are matched with those of a knowledge base described by the target ontology. The matching is performed by finding the surroundings of the target instances which are lexically similar to the CQA. The idea behind the approach is to rely on a few examples (answers) to find a generic rule which describes more instances. The assumption that the user knows the source ontology and is able to write each CQA into a SPARQL query on the source ontology is made.

The overall approach is articulated in 11 steps, as depicted in Figure 1. The approach is based on **subgraphs** which are a **set of triples** for a unary CQA and a **property path** for a binary CQA. A lexical layer comparison is used to measure the similarity of the subgraphs with the CQA.

In the remainder of the paper, the examples consider the knowledge bases in Figure 2. They share common instances: *o:person1* and *o′:person1*, *o:paper1* and *o′:paper1*. Ontology *o* represents the concept of *accepted paper* as a class while *o′* models the same knowledge with a *has decision* property. The property *paper written by* is represented by a single property in *o* while in *o′*, the property *writes* links a *person* to a *document*. A criticism to this example could be that two knowledge bases may not represent the same conference, therefore they may not share common paper instances. However, these bases may have a different but overlapping scope. For example *o* could focus on the event organisation part of a conference and *o′* on reviewer management. Before detailing the main steps of the approach, we instantiate the overall approach to deal with unary and binary queries.

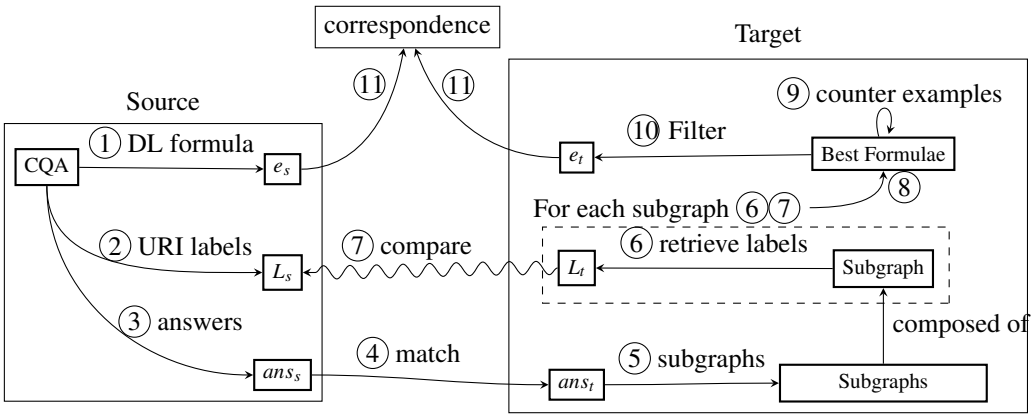
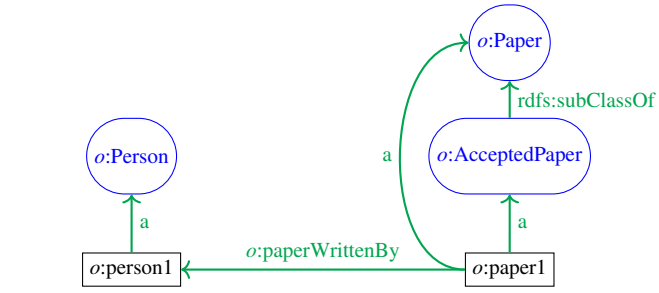
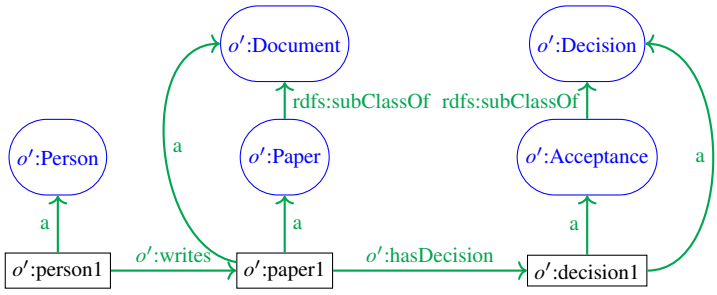


Fig. 1. Schema of the general approach.



(a) Source knowledge base



(b) Target knowledge base

Fig. 2. Source and target knowledge bases.

3.1. *Approach over a unary CQA*

In the following, Figure 1 is instantiated for a unary CQA. The SPARQL CQA is that of Figure 3(a):

1. Represent the SPARQL CQA as a DL formula e_s (*e.g.*, *o:AcceptedPaper*) (Section 4.1).
2. Extract lexical information from the CQA, L_s set labels of entities from the CQA (*e.g.*, "accepted paper").
3. Retrieve source answers ans_s of the CQA (*e.g.*, *o:paper1*).
4. Find equivalent or similar target answers ans_t to the source instances ans_s (*e.g. o:paper1* $\sim$ *o′:paper1*) (Section 4.2).

```
SELECT ?x WHERE {
  ?x a o1:AcceptedPaper.
}
```

(a) Source unary CQA.

```
SELECT ?x ?y WHERE {
  ?x o1:paperWrittenBy ?y.
}
```

(b) Source binary CQA.

Fig. 3. Source CQAs.

⑤ Extract the subgraphs of target answers (Section 4.3): for a unary query, this is the set of triples in which the target instances appear as well as the types (classes) of the subject or object of the triple (*e.g.* in DL, the description of o'*:paper1* would contain $\langle$ o'*:paper1 , o':hasDecision , o':decision1* $\rangle$, $\langle$ o'*:decision1 , rdf:type , o':Decision* $\rangle$ *and* $\langle$ o'*:decision1 , rdf:type , o':Acceptance* $\rangle$.)
⑥ For each subgraph, retrieve L_t the labels of its entities (*e.g.*, o'*:hasDecision* $\rightarrow$ "decision", o'*:decision1* $\rightarrow$ "decision for paper1", o'*:Decision* $\rightarrow$ "decision").
⑦ Compare L_s and L_t (Section 4.4).
⑧ Select the subgraphs parts with the best similarity score, transform them into DL formulae (Section 4.3) and aggregate them (Section 4.5). In this example, the part of the subgraph which is the most similar to the CQA (in terms of label similarity) is o'*:Acceptance*. The DL formula is therefore $\exists o'$*:hasDecision.*o'*:Acceptance*.
⑨ Reassess the similarity of each DL formula based on their counter-examples (Section 4.6 and Section 4.7). The counter-examples are common instances of the two knowledge bases which are described by the target DL formula but not by the original CQA.
⑩ Filter the DL formulae based on their similarity score (if their similarity score is higher than a threshold) (Section 4.8).
⑪ Put the DL formulae e_s and e_t together to form a correspondence (*e.g.*, $\langle$ *o:AcceptedPaper ,* $\exists$ o'*:hasDecision.*o'*:Acceptance ,* $\equiv$ $\rangle$) and express this correspondence in a reusable format (*e.g.*, EDOAL). The confidence assigned to a correspondence is the similarity score of the DL formula computed.

3.2. Approach over a binary CQA

① Extract source DL formula e_s (*e.g.*, *o:paperWrittenBy*) from SPARQL CQA (Section 4.1): SELECT ?x ?y WHERE ?x o1:paperWrittenBy ?y.
② Extract lexical information from the CQA, L_s set labels from the DL formula (*e.g.*, "paper written by").
③ Extract source answers ans_s of the CQA (*e.g.*, a pair of instances *(o:paper1, o:person1)*).
④ Find equivalent or similar target answers ans_t to the source instances ans_s (*e.g. o:paper1* $\sim$ o'*:paper1* and *o:person1* $\sim$ o'*:person1*) (Section 4.2).
⑤ Retrieve the subgraphs of target answers (Section 4.3): for a binary query, it is the set of paths between two answer instances as well as the types of the instances appearing in the path (*e.g.*, a path of length 1 is found between o'*:paper1* and o'*:person1*). The path is composed of only one property and there are no other instances than o'*:paper1* and o'*:person1* in this path. Their respective types are retrieved: (o'*:Paper*,o'*:Document*) for o'*:paper1* and (o'*:Person*) for o'*:person1*.
⑥ For each subgraph, retrieve L_t the labels of its entities (*e.g.*, o'*:writes* $\rightarrow$ "writes", o'*:Person* $\rightarrow$ "person", o'*:Paper* $\rightarrow$ "paper", *etc.*).
⑦ Compare L_s and L_t (Section 4.4).
⑧ Select the subgraph parts with the best score, transform them into DL formulae (Section 4.3). Keep the best path variable types if their similarity is higher than a threshold. (*e.g.*, the best type for the instance o'*:paper1* is o'*:Paper* because its similarity with the CQA labels is higher than the similarity of o'*:Document*).
⑨ Reassess the similarity of each DL formula based on their counter-examples (Section 4.6 and Section 4.7).
⑩ Filter the DL formulae based on their similarity score (Section 4.8) .
⑪ Put the DL formulae e_s and e_t together to form a correspondence (*e.g.*, $\langle$ *o:paperWrittenBy , dom(*o'*:Paper)* $\sqcap$ o'*:writes*$^-$, $\equiv$ $\rangle$ and express this correspondence in a reusable format (*e.g.*, EDOAL). The confidence assigned to a correspondence is the similarity score of the DL formula computed.

The main difference with the case of unary CQAs is in Step ④ because the two instances of the pair answer are matched instead of one, Step ⑤ and Step ⑧ which deal with the subgraph extraction and pruning.

4. Main steps of the approach

This section details the steps ①, ④, ⑤, ⑦, ⑧, ⑨ and ⑩ of Figure 1 and illustrate them with examples.

4.1. Translating SPARQL CQAs into DL formulae

In Step ①, in order to translate a SPARQL query into a DL formula, the first step is to translate it into a FOL formula and then transform it into a DL formula. A SPARQL SELECT query (in the scope of our approach) is composed of a SELECT clause containing variable names and a basic graph pattern, *i.e.*, a set of triples with variables sometimes with constructors (such as UNION or MINUS). First, the variables in the SELECT clause become the quantified variables of the formula. In unary CQA, the SELECT clause contains one variable. In binary CQA, the SELECT clause contains two variables. The SPARQL query of Figure 4, `?x` becomes the quantified variable of our formula: $\forall\ x$. Then, the basic graph pattern is parsed to find what predicates apply to the quantified variables and add them to the formula. Each triple of the basic graph pattern is either a unary or a binary predicate. If new variables are added, an existential quantifier is used for them. In the example, we find the triple ⟨ `?x` , o':hasDecision , `?y` ⟩. The FOL formula becomes $\forall\ x, \exists\ y,\ o'$*:hasDecision(x,y)*. We then recursively keep on exploring the basic graph pattern for each new variable introduced. After exploring the basic graph pattern for the variable `?y`, the FOL formula becomes $\forall\ x, \exists\ y,\ o'$*:hasDecision(x,y)* $\wedge\ o'$*:Acceptance(y)*. At the end of the process, we transform the basic graph pattern into a DL formula, which can also be translated into an EDOAL formula as shown below. $\forall\ x, \exists\ y,\ o'$*:hasDecision(x,y)* $\wedge\ o'$*:Acceptance(y)* becomes in DL: $\exists\ o'$*:hasDecision.*o'*:Acceptance*. The FOL to DL equivalence is done as in [18].

```
SELECT ?x WHERE {
  ?x o2:hasDecision ?y.
  ?y a o2:Acceptance.
}
```

Fig. 4. SPARQL SELECT query with one variable in SELECT clause.

4.2. Instance matching

In Step ④, the answers of the CQA over the source knowledge base which have been retrieved are matched with the instances of the target knowledge base. This instance matching phase relies on existing links (*owl:sameAs, skos:exactMatch, skos:closeMatch, etc.*) if they exist. If no such link exists, an exact label match is performed. When dealing with binary CQA whose results are an instance-literal value pair, the instance is matched as before (existing links or exact labels), the literal value will be matched with an exactly identical value (the datatype is not considered) in the path finding step, detailed in Section 4.3.

4.3. Retrieving and pruning subgraphs

The approach relies on subgraphs, which are sets of triples from a knowledge base. These subgraphs are found (Step ⑤), pruned and transformed into DL formulae (Step ⑧). The type of subgraphs for unary or binary CQAs is inspired from [19], which proposes an approach to find equivalent subgraphs within the same knowledge base.

*A **unary** CQA* expects a set of single instances as answer. The subgraph of a single instance is composed of a triple in which the instance is either the subject or the object, and the types (classes) of the object or subject of this triple. For example, *o′:paper1* is the subject of the triple *o′:paper1 o′:hasDecision o′:decision1* and *o′:decision1* has types (classes) *o′:Acceptance* and *o′:Decision*. A subgraph of *o′:paper1* is therefore composed of the three following triples: (i) ⟨ *o′:paper1 , o′:hasDecision , o′:decision1* ⟩, (ii) ⟨ *o′:decision1 , rdf:type , o′:Acceptance* ⟩, (iii) ⟨ *o′:decision1 , rdf:type , o′:Decision* ⟩.

When comparing the subgraph with the CQA labels, if the most similar object (resp. subject) type is more similar than the object (resp. subject) itself, the type is kept. Let us consider the *accepted paper* CQA. The most similar type of the triple of the object is *o′:Acceptance*. Therefore, triple 3 is pruned. The object of triple 1 is *o′:decision1* and the most similar object type to the CQA is *o′:Acceptance*. *o′:Acceptance* is more similar to the CQA than *o′:decision1*. Therefore, *o′:decision1* becomes a variable and triple 2 stays in the subgraph. In order to translate a subgraph into a DL formula, we first translate this subgraph into a SPARQL query:

- The answer is transformed into a variable and put in the SELECT clause. In this example, *o′:paper1* becomes a variable `?x` in the SELECT clause: `SELECT ?x WHERE`.
- The instances of the subgraphs which are not kept are transformed into variables. In this example, *o′:decision1* becomes a variable `?y`.
- These transformations are applied to the selected triples of the subgraph which become the basic graph pattern of the SPARQL query. In this example, the SPARQL query is the one in Figure 4.

Finally, the SPARQL query is transformed into a DL formula by using the same process as that described in Section 4.1: $\exists o'$*:hasDecision.*o'*:Acceptance*.

*A **binary** CQA* expects a set of pairs of instances (or pairs of instance-literal value) as answer. Finding a subgraph for a pair of instances consists in finding a path between the two instances. The shortest paths are considered more accurate. Because finding the shortest path between two entities is a complex problem, paths of length below a threshold are sought. First, paths of length 1 are sought, then if no path of length 1 is found, paths of length 2 are sought, *etc.* If more than one path of the same length are found, all of them go through the following process. When a path is found, the types of the instances forming the path are retrieved. If the similarity of the most similar type to the CQA is above a threshold, this type is kept in the final subgraph. For example, for a "*paper written by*" CQA with the answer (*o′:paper1*,*o′:person1*) in the target knowledge, a subgraph containing the following triples is found: ⟨ *o′:person1 , o′:writes , o′:paper1* ⟩, ⟨ *o′:paper1 , rdf:type , o′:Paper* ⟩, ⟨ *o′:paper1 , rdf:type , o′:Document* ⟩, ⟨ *o′:person1 , rdf:type , o′:Person* ⟩. The most similar type of *o′:person1* is *o′:Person*, which is below the similarity threshold. Triple 4 is then removed from the subgraph. The most similar type of *o′:paper1* is *o′:Paper*. Triple 3 is therefore removed from the subgraph. *o′:Paper*'s similarity is above the similarity threshold: triple 2 stays in the subgraph. The translation of a subgraph into a SPARQL query is the same for binary and unary CQAs. Therefore, the subgraph will be transformed into a SPARQL query and saved as a DL formula: *dom(o′:Paper)* $\sqcap$ *o′:writes*$^-$.

4.4. Label similarity

In step ⑦, a label similarity metric is needed to compare two sets of labels L_s and L_t. A Levenshtein [20] distance-based similarity metric was chosen. It measures the minimum number of single-character edits (insertions, deletions or substitutions) between two strings. The similarity between two sets of labels is the cartesian product of the string similarities between the labels of L_s and L_t (1). *strSim* is the string similarity of two labels l_s and l_t (2).

$$sim(L_s, L_t) = \sum_{l_s \in L_s} \sum_{l_t \in L_t} strSim(l_s, l_t) \tag{1}$$

$$strSim(l_s, l_t) = \begin{cases} \sigma \text{ if } \sigma > \tau, \text{ where } \sigma = 1 - \dfrac{levenshteinDist(l_s, l_t)}{\max(|l_s|, |l_t|)} \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

4.5. DL formula aggregation

In Step (8) of the approach, when dealing with unary CQA, the DL formulae can be aggregated. It consists in transforming one or more formulae with a common predicate into a more generic formula. This aggregation only applies to formulae which contain an instance or a literal value and which were kept in the subgraph selection step. For example, this step would apply for a formula such as $\exists\ o'$*:hasDecision.*$\{o'$*:accept*$\}$. There are three steps to the aggregation. First, we create a first aggregated formula which we call the **extension** formula. It consists in merging the instances or literal values of the formulae with the same predicate into one set of values. Let us consider that through various answers to a CQA (*e.g.*, *o′:paper1*, *o′:paper2*, *etc.*), we encountered the following formulae: $\exists\ o'$*:hasDecision.*$\{o'$*:accept*$\}$, $\exists\ o'$*:hasDecision.*$\{o'$*:strongAccept*$\}$, $\exists\ o'$*:hasDecision.*$\{o'$*:weakAccept*$\}$. The extension formula of these formulae is: $\exists\ o'$*:hasDecision.*$\{o'$*:accept, o′:strongAccept, o′:weakAccept*$\}$. The extension formula of a formula which does not share its predicate with any other is the formula itself. Then, an **intension** formula can be computed by replacing the set of values by the top class $\top$. The intension formula of the example formulae is: $\exists\ o'$*:hasDecision.*$\top$. Finally, a choice is made between the extension or intension formulae based on the predicate similarity to the CQA. If the predicate is more similar than the values, the intension formula is kept. Otherwise, the extension formula is kept. In our example, the extension formula $\exists o'$*:hasDecision.*$\{o'$*:accept, o′:strongAccept, o′:weakAccept*$\}$ is kept.

Table 1
Initial, extension, intension and final (in bold) formulae. The CQA considered is *"accepted papers"*.

Initial formulae	Extension	Intension
$\exists o'$*:hasDecision.*$\{o'$*:accept*$\}$ $\exists o'$*:hasDecision.*$\{o'$*:strongAccept*$\}$ $\exists o'$*:hasDecision.*$\{o'$*:weakAccept*$\}$	$\exists o'$***:hasDecision.***$\{o'$***:accept,*** o'***:strongAccept,*** o'***:weakAccept***$\}$	$\exists o'$*:hasDecision.*$\top$
$\exists o'$*:acceptedBy.*$\{o'$*:person1*$\}$	$\exists o'$*:acceptedBy.*$\{o'$*:person1*$\}$	$\exists o'$***:acceptedBy.***$\top$

We present two examples of initial formulae, with their respective intension and extension formulae in Table 1. These were obtained with the competency question *"accepted paper"*. In Table 1, the final formulae are in bold. Applied to the examples of Table 1:

- o'*:accept*, o'*:strongAccept* and o'*:weakAccept* are more similar to the CQA than o'*:hasDecision*. The extension form is chosen.
- o'*:acceptedBy* is more similar (based on labels) to the CQA than o'*:person1*. The intension form is chosen.

4.6. Calculating the percentage of counter-examples

In Step (9), the approach refines the DL formula similarity score by looking for counter-examples (details about the similarity score are given in Section 4.7). A **counter-example** is a common instance of the source and target ontologies which is described by the DL formula found by the approach in the target ontology but which is not described by the CQA in the source ontology. For example, let us assume that the target formula e_t is o'*:Paper* for the "accepted paper" CQA. From the target ontology, the answers o'*:paper1, o′:paper2, o′:paper3* and o'*:paper4* are retrieved from e_t and matched to the source instances respectively *o:paper1, o:paper2, o:paper3* and *o:paper4*. However, only *o:paper1* and *o:paper2* are accepted papers (and are described by the CQA) in the source ontology. Therefore *o:paper3* and *o:paper4* are counter-examples. The percentage of counter-examples is computed as follows. The answers $ans_t^{e_t}$ described by the target subgraph (e_t) are retrieved from the target knowledge. These answers are matched to source instances: $ans_s^{e_t}$. The percentage of counter-examples is the proportion of common instances $ans_s^{e_t}$ which are not answers to the CQA ($\neg(ans_s^{cqa})$). The equation for the percentage of counter examples (*percCounterExamples*) is therefore:

$$percCounterExamples = \frac{|ans_s^{e_t} \sqcap \neg(ans_s^{cqa})|}{|ans_s^{e_t}|} \tag{3}$$

In the example, the percentage of counter-example is $\frac{2}{4} = 50\%$.

4.7. DL formula similarity

In Step ⑩, the formulae are filtered based on their similarity score with the CQA. The similarity score is a combination of:

Label similarity *labelSim* is the sum of the label similarity of each entity of the formula with the CQA.

Structural similarity *structSim*. This similarity was introduced to enhance some structural aspects in a formula. In the implementation of the approach, this value is set to 0.5 for a path between the two instances of the answer, and 0 for a unary CQA subgraph. Indeed, if the label similarity of the path is 0, the structural similarity hints that the fact that a path was found is a clue in favour of the resulting DL formula.

Percentage of counter examples *percCounterExamples* which is computed in Step ⑨ and detailed Section 4.6.

The similarity score is calculated with the following equation:

$$similarity = (labelSim + structuralSim) \times (1 - percCounterExamples) \quad (4)$$

For instance, consider the similarity of $\exists o'$*:hasDecision.*o'*:Acceptance* with the unary CQA *"accepted paper"*.

- *labelSim* $= 0.8+0.0$ as *sim(labels(CQA), labels(*o'*:hasDecision))* = 0.0 and *sim(labels(CQA), labels(*o'*:Acceptance))* = 0.8
- *strucSim* $= 0.0$ because it is a unary CQA
- *percCounterExamples* $= 0.0$

The similarity of this DL formula is *similarity* $= (0.8 + 0.0) \times (1 - 0) = 0.8$

4.8. DL formula filtering

In Step ⑩, the formulae are filtered. Only the DL formulae with a similarity higher than a threshold are put in a correspondence with the CQA DL formula. If for a given CQA, there is no DL formula with a similarity higher than the threshold, only the best DL formulae with a non-zero similarity are put in the correspondence. The best DL formulae are the formulae with the highest similarity score. When putting the DL formula in a correspondence, if its similarity score is greater than 1, the correspondence confidence value is set to 1.

5. Evaluation

An automatic evaluation was performed on a synthetic dataset (Populated Conference) in order to measure the impact of various parameters on the approach. The approach was also evaluated on LOD repositories about plant taxonomy. This scenario shows how the approach performs when faced with LOD challenges such as large ontologies and millions of triples. Some of the knowledge bases chosen for this experiment are irregularly populated. This means that the same piece of knowledge can be represented in various ways in the same ontology and that all instances are not described identically. After detailing the parameters in Section 5.1 and the evaluation settings in Section 5.2, the results over the two datasets are presented (Sections 5.3 and 5.5). The discussion is then presented in Section 6.

5.1. Matching approach set-up

Label similarity A threshold is applied to the similarity measure obtained: if the similarity between two labels is below a threshold τ, this similarity is considered noisy and is set to zero.

Path length threshold The maximum path length sought is 3. Paths longer than that may bring noise in the correspondences, as the path-finding algorithm searches for all combinations of properties.

Structural similarity The structural similarity is 0 for a triple (in the case of a unary CQA) and 0.5 for a path found between two matched entities (in the case of a binary CQA). Finding a path between two instances (the matched answers of a binary CQA) is a hint that this subgraph can be correct. In opposition, the structure subgraphs for unary CQA are not that informative.

DL formula threshold The DL formulae with a similarity higher than 0.6 are kept. If a CQA has no DL formula with a similarity higher than 0.6, the best formulae are put in a correspondence (the formulae having the best similarity, if this similarity is above 0.01). This threshold was chosen to be above the structural similarity threshold (0.5) for a path subgraph. Indeed, if two paths are found but only one has a label similarity above 0, then its associated DL formula will be the only one output. These thresholds were empirically chosen.

Approach variants The other parameters (number of support answers, the Levenshtein threshold, *etc.* are varied to create *variants* of the approach (Table 2).

5.2. Evaluation settings

Evaluation datasets An automatic evaluation was performed on the populated version of the OAEI Conference benchmark [21]. This dataset is composed of 5 ontologies, with 100 manually generated CQAs. This evaluation measured the impact of various parameters on the approach. Second, a manual evaluation was carried out on the Taxon dataset about plant taxonomy, composed of 4 large populated ontologies: AgronomicTaxon [22], AgroVoc [23], DBpedia [24] and TaxRef-LD [25]. 6 CQAs from AgronomicTaxon have been manually generated. The CQA used in this evaluation are the one presented in [26] which were manually written from AgronomicTaxon CQs [22].

Evaluation metrics The evaluation metrics are based on the comparison of instance sets, as described in [27]. The generated alignment is used to rewrite a set of reference source CQAs whose results (set of instances) are compared to the ones returned by the corresponding target reference CQA. This metric shows the overall *coverage* of the alignment with respect to the knowledge needs and the best rewritten query[3]. A balancing strategy consists in calculating the intrinsic alignment *precision* based on common instances. Given an alignment A_{eval} to be evaluated, a set of CQA reference pairs cqa_{pairs} (composed of source cqa_s and target cqa_t), kb_s the source knowledge base, kb_t a target knowledge base, and f an instance set (I) comparison function:

$$coverage(A_{eval}, cqa_{pairs}, kb_s, kb_t, f) = \underset{\langle cqa_s, cqa_t \rangle \in cqa_{pairs}}{\text{average}} f(I^{kb_s}_{cqa_t}, I^{kb_t}_{bestq_t}) \quad (5)$$

Different functions f can be used for comparing instance sets (overlap, precision-oriented, recall-oriented etc.). Here, *coverage* is based on the *queryFmeasure* (also used for selecting the best rewritten query). This is motivated by the fact that it better balances precision and recall. Given a reference instance set I_{ref} and an evaluated instance set I_{eval}:

$$QP = \frac{|I_{eval} \cap I_{ref}|}{|I_{eval}|} \qquad QR = \frac{|I_{eval} \cap I_{ref}|}{|I_{ref}|} \quad (6)$$

$$queryFmeasure(I_{ref}, I_{eval}) = 2 \times \frac{QR \times QP}{QR + QP} \quad (7)$$

$$bestq_t = \underset{q_t \in rewrite(cqa_s, A_{eval}, kb_s)}{\text{argmax}}$$

[3]The description of rewriting systems is out of the scope of this paper.

$$queryFmeasure(I^{kb_t}_{cqa_t}, I^{kb_t}_{q_t}) \quad (8)$$

Balancing *coverage*, *precision* is based on classical (i.e., scoring 1 for same instance sets or 0 otherwise) or non-disjoint functions f:

$$precision(A_{eval}, kb_s, kb_t, f) = \underset{\langle e_s, e_t\rangle \in A_{eval}}{\text{average}}\ f(I^{kb_s}_{e_1}, I^{kb_t}_{e_2}) \quad (9)$$

Such metrics have been used in the automatic evaluation on the controlled populated version of the Conference dataset. Given the uneven population of Taxon (*i.e.,* a same piece of knowledge can be represented in various ways within the same ontology and that all instances are not described identically), a manual evaluation has been carried out instead in order to avoid entailing noise in the instance-based comparison. The scoring metrics used in the scoring step of the workflow are classical, recall-oriented, precision-oriented, query f-measure, overlap. A best-match (query f-measure) aggregation over the reference CQA is performed. An average of the best-match scores gives the **CQA Coverage**. The **Precision** (intrinsic precision) is calculated by comparing the source and target members of the correspondences. The scoring metrics used for the Precision are those used in the CQA Coverage. The Precision and CQA Coverage with the same scoring metric are finally aggregated in a **Harmonic Mean**.

Environment The approach and evaluation system have been executed on a Ubuntu 16.04 machine configured with 16GB of RAM running under an i7-4790K CPU 4.00GHz × 8 processors. The runtimes are given for a single run. The local SPARQL endpoints were run on the same machine with Fuseki 2[4].

5.3. Results on Conference

The approach has been run and evaluated on the populated conference 100% dataset on a local Fuseki 2 server. This choice is motivated by the fact that the search for a common instance is faster when the proportion of common instances in the source answers is higher. The implementation in Java of the evaluation system as well as the Populated Conference dataset is available[5].

The *variants* of the approach have been compared to its baseline(Table 2). The parameters which are not described in this table such as path length threshold (3), DL formula filtering threshold (0.6) and structural similarity constants (0.5 for a path, 0 for a class expression) as presented in Section 5.1. This evaluation strategy allows to isolate the parameters and measure their impact, as discussed in the following.

Table 2

Parameters of the evaluated variants of the approach: number of support answers (Nb. ans.), Levenshtein threshold in the similarity metric (Lev. thr.), type of instance matching strategy (Inst. match), computation of counter-examples (Co.-ex.), CQAs input

Evaluated variant	Nb ans.	Lev. thr.	Inst. match	Co.-ex.	CQAs
baseline	10	0.4	links		√
Levenshtein	10	0.0–1.0	links		√
Support answers	1-100	0.4	links		√
exact label match	10	0.4	labels		√
query	10	0.4	links		
query+reassess	10	0.4	links	√	
cqa+reassess	10	0.4	links	√	√

[4]https://jena.apache.org/documentation/fuseki2/

[5]https://framagit.org/IRIT_UT2J/conference-dataset-population

5.3.1. Impact of the threshold in the string similarity metric

An evaluation was performed for each version of the baseline approach with a threshold set between 0.0 and 1.0. Figure 5 shows the number of correspondence per type (*i.e.*, *(s:s)*,*(s:c)*,*(c:s)* and *(c:c)*) which were found by the variants of the approach. The evaluation results are shown in Figure 6. The number of correspondences decreases when the Levenshtein threshold increases. Numerous correspondences obtained with a low Levenshtein threshold cover a lot of CQAs (high CQA Coverage) but contain a lot of errors (low Precision). The lower the threshold, the best the CQA Coverage and the lowest the Precision. The Harmonic Mean is the highest for a threshold of 0.4 in the similarity metric. The Classical Harmonic Mean scores are rather low overall but the query f-measure Harmonic Mean scores show that even though the correspondences output are not exactly equivalences, they are quite relevant overall. The baseline approach Levenshtein threshold (0.4) was chosen based on this experiment.

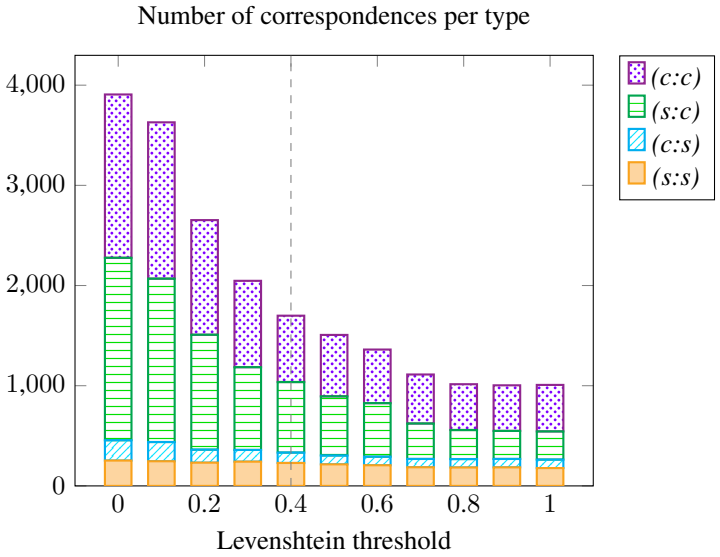


Fig. 5. Number of correspondences per type for each variant with a different Levenshtein threshold

5.3.2. Impact of the number of support answers

Different versions of the baseline approach were evaluated with a number of support answers between 1 and 100. The runtime of the approach over the 20 oriented pairs of ontologies is displayed in Figure 7 and Figure 8 shows the number of correspondences per type output by the variants. The evaluation results are shown in Figure 9. It could be observed that even with 1 answer as support, the CQA Coverage and Precision scores are high, which shows that the approach can make a generalisation from few examples. As expected, the bigger the number of support answers, the longest the process is to run. Some CQA have only 5 answers (only 5 conference instances in the population of the ontologies), that explains why the time rises linearly between 1 support answer and 5 support answer and has a lower linear coefficient for support instance over 5. The Precision scores gets lower with more support answers. The main reason is that particular answer cases which are lexically similar to the CQA labels can be discovered when a lot of instances are considered. The increase of the number of correspondences with the number of support answers shows that incorrect correspondences have been introduced.

The same problem occurs in the CQA Coverage: with 100 support answers, special cases having a higher similarity to the CQA than the expected formula can be found. As in the approach, the formulae are filtered, and when the similarity of the best formula is below a threshold (0.6), only the best one is kept. However, the overlap CQA Coverage gets slightly higher for a high number of support answers because accidental correspondences have been introduced.

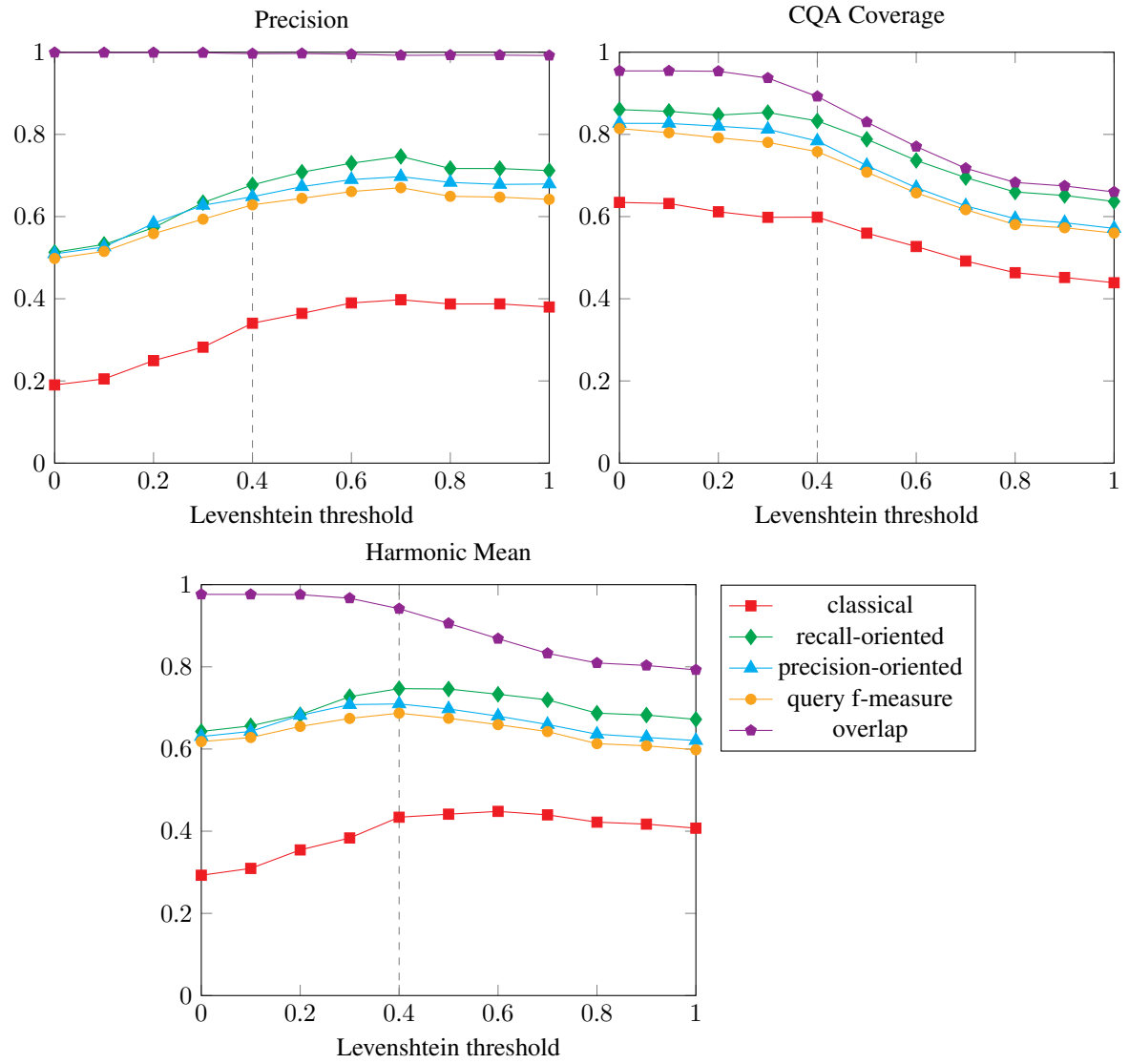


Fig. 6. Results of the evaluation with 10 support answers and variable levenshtein threshold in the string similarity measure. The baseline results are highlighted by a vertical dashed line.

5.3.3. Similar instances based on exact label match or existing links

A variant of the approach does not use existing links between instances, instead it performs an exact label match between instances. Figure 10 shows the number of correspondences per type of the baseline and its variant. Figure 11 shows the results of the baseline and its *exact label match* variant. The use of exact label match for the instance matching phase brings noise to the correspondences and lowers the Precision. The overlap Precision also decreases because the correspondence are not ensured to share a common instance. In the baseline approach, which uses *owl:sameAs* links, the support answers were by definition common instance and outputting correspondences with no overlap was not possible (except when dealing CQA with literal values).

The baseline approach (existing *owl:sameAs* links) takes **2.0 hours** to run over the 20 pairs of ontologies whereas the exact label match approach takes **59.2 hours**. The long runtime for the exact label match approach can be explained by the necessary steps to find the exact label match answers. When using direct links, these steps are replaced by directly retrieving *owl:sameAs* links, which takes about 20ms per source instance. If the number of

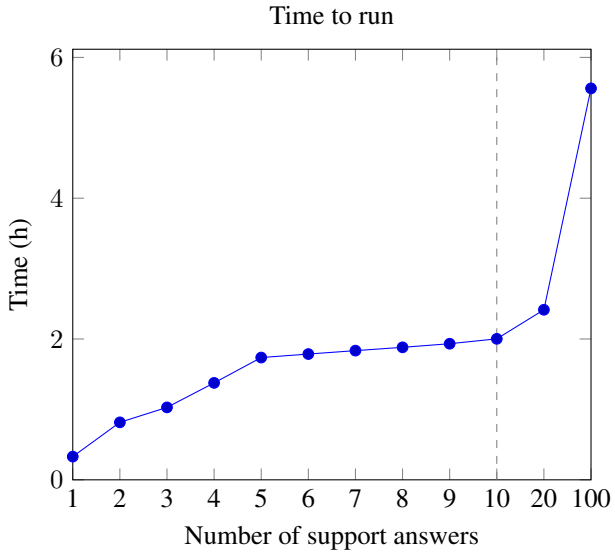


Fig. 7. Time taken by the approach to run for the 20 oriented pairs of ontologies with a different number of support answers

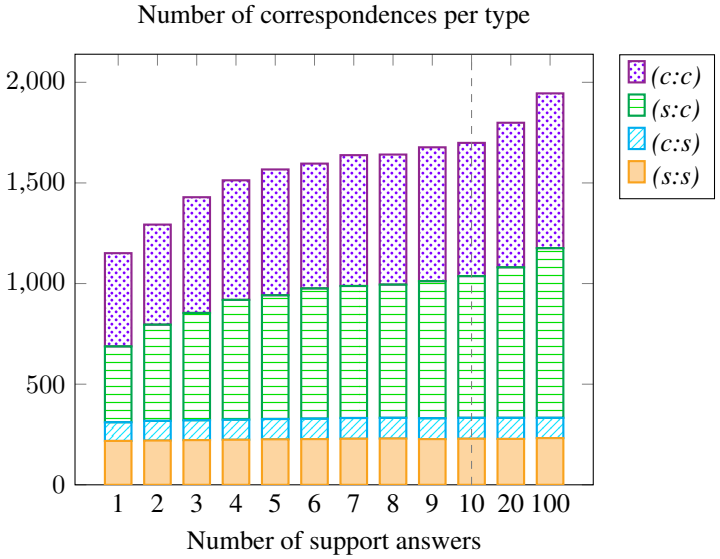


Fig. 8. Number of correspondence per type for each variant with a different number of support answers

common support answers between the source and target ontology is reached (in the baseline, when 10 support answers are found), the approach stops looking for new matches. However, when no common instance can be found, the approach looks for a match for every answer of the CQA. This fact coupled with the slow label queries results in such a long time. When common instances exist but do not share the same exact labels, the approach also looks for matches for every source answer, without success.

5.3.4. CQAs or generated queries

In order to measure how the CQA impact the results of the approach, the baseline approach is compared to a variant which does not rely on input CQA but automatically generates queries. Three types of SPARQL queries are generated for a given source ontology: *Classes*, *Properties* and *Property-Value pairs*.

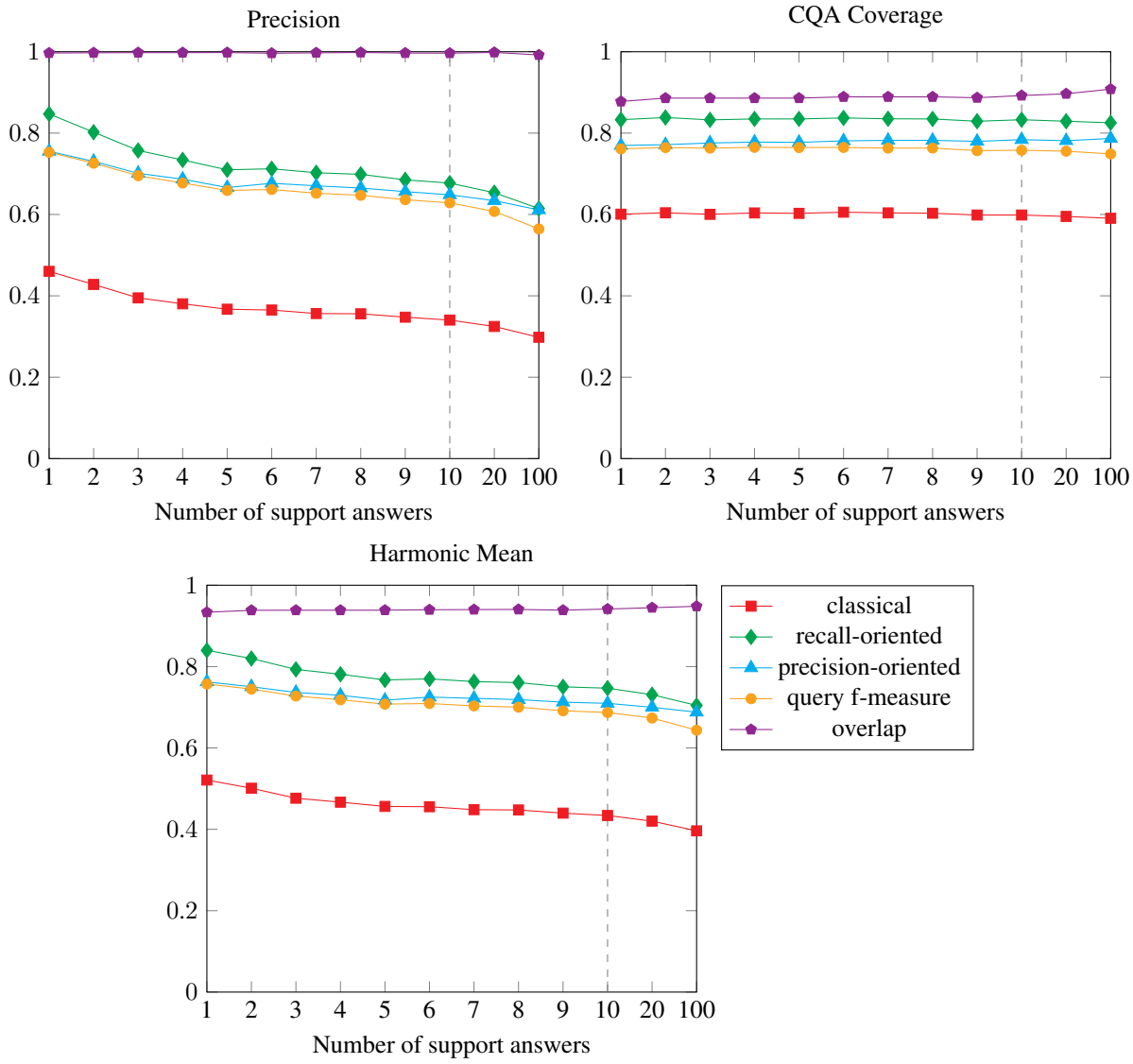


Fig. 9. Results of the evaluation with a 0.4 Levenshtein similarity threshold and variable number of support answers. The baseline results are highlighted by a vertical dashed line.

Classes For each *owl:Class* populated with at least one instance, a SPARQL query is created to retrieve all the instances of this class.

Properties For each *owl:ObjetProperty* or *owl:DatatypeProperty* with at least one instantiation, a SPARQL query is created to retrieve all the pairs of instances of this class.

Property-Value pairs Inspired by the approaches of [8, 9, 28], SPARQL queries of the following form are created:

- `SELECT DISTINCT ?x WHERE {?x o1:property1 o1:Entity1.}`
- `SELECT DISTINCT ?x WHERE {o1:Entity1 o1:property1 ?x.}`
- `SELECT DISTINCT ?x WHERE {?x o1:property1 "Value".}`

Table 3 shows the number of generated queries per source ontology of the evaluation set.

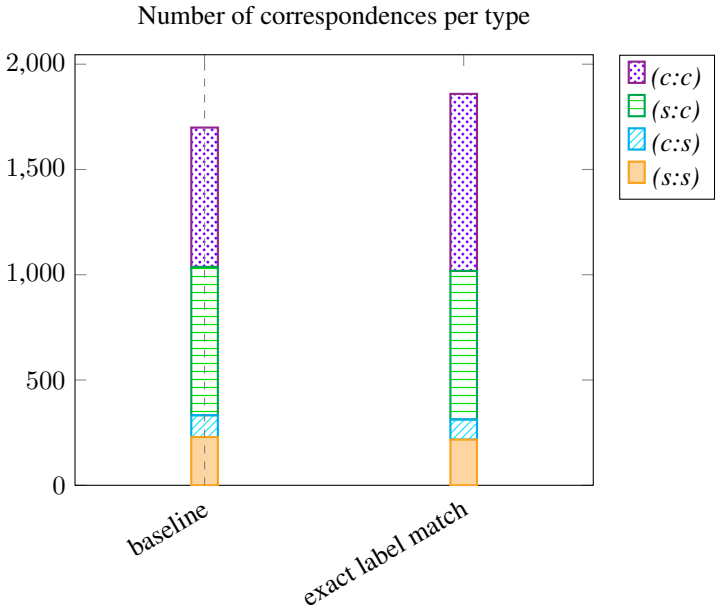


Fig. 10. Number of correspondence per type for the baseline and the variant based on exact label match

Table 3
Number of generated queries and CQAs per source ontology

Nb of queries	cmt	conference	confOf	edas	ekaw
classes	26	51	29	43	57
properties	50	50	20	28	26
properties-value	30	20	0	5	15
TOTAL	106	121	49	76	98
CQAs	34	73	54	52	65

The approach based on generated queries will not output a correspondence for each CQA in the evaluation. Therefore, the rewriting systems in the evaluation process will bring in noise. The CQA Coverage scores are comparable as only the best result is kept. The Precision of the alignment output is computed by comparing the instances of the source and target members in their respective ontologies. These Precision scores give an indicator of the actual precision of these approaches.

The results of the evaluation of the baseline (based on CQAs) and the query variant are presented in Figure 12. Figure 13 shows the number of correspondence per type. The CQA Coverage scores when the approach is based on generated queries are between 10% and 20% lower than those obtained with CQA. Indeed, the *(c:c)* correspondences it retrieves are limited to the Class-by-Attribute-Value pattern on their source member. The Precision scores are not really comparable because the ontologies were populated based on CQA and not on entities: a *Document* class may be populated with more or less instances given its subclasses. As the approach relies on common instances, the overlap Precision (percentage of correspondences whose member's instances overlap) is around 1.0. The classical Precision (percentage of correspondences whose members are strictly equivalent) is however rather low overall.

The baseline and the *query* variant both take **2.0 hours** to run on the 20 pairs of ontologies. Even if there are more queries to cover than CQA, the runtime of the *query* variant is compensated by the "difficulty" of the CQA: some CQAs contain unions or property paths and therefore take more time to be answered by the Fuseki server than the generated queries.

The number of *(s:s)* and *(s:c)* correspondences is much higher for the *query* variant. This approach generates 380 queries which express simple expressions (lines classes and properties of Table 3) and therefore, will give *(s:s)*

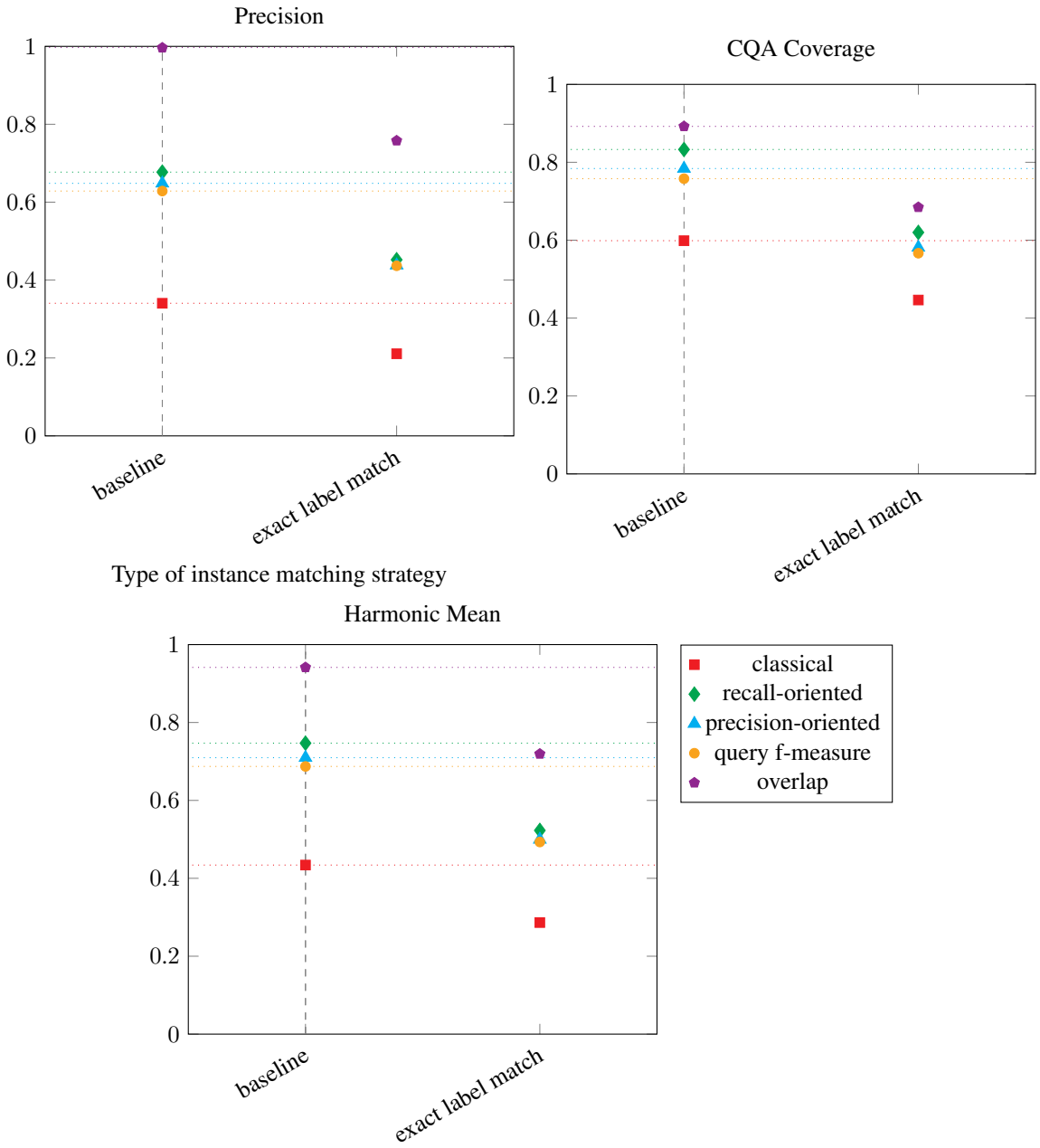


Fig. 11. Comparison of the approach results when relying on existing *owl:sameAs* links or on an exact label-based instance matching. The baseline results are highlighted by a vertical dashed line.

or *(s:c)* correspondences if a match is found. In comparison, the baseline approach relies on 133 SPARQL CQAs which represent a simple expression and 145 which represent a complex expression.

5.3.5. Similarity reassessment with counter-examples

The baseline, the *query* variant and their equivalent were run with a similarity reassessment phase. The runtime of the variants is presented in Figure 14. Figure 15 shows the number of correspondences per type output by the baseline and its variants. The results of this evaluation are presented in Figure 16.

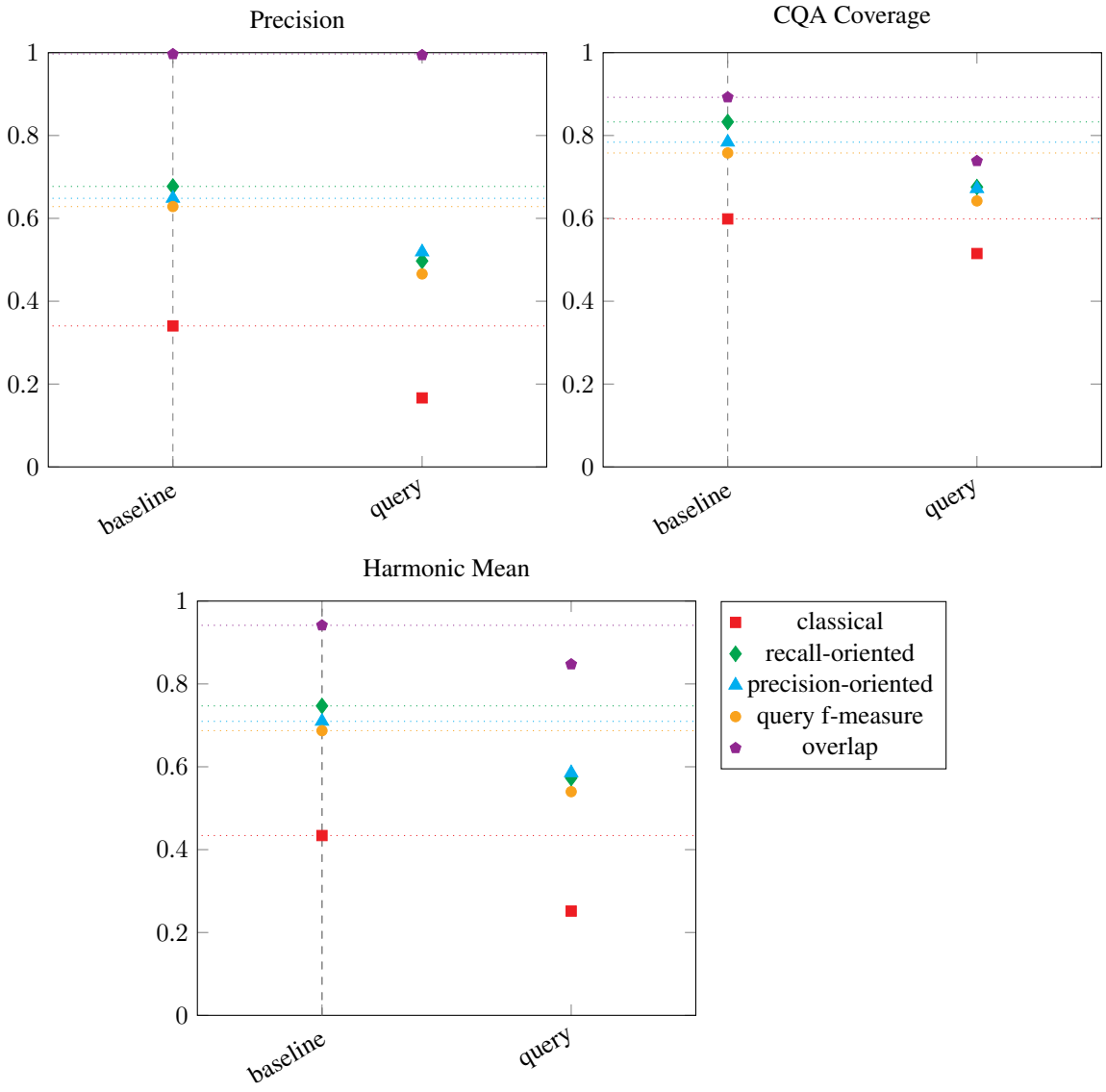


Fig. 12. Results for the baseline and the variant which generates queries (query)

The reassessment phase (finding counter examples) increases the runtime by far, especially when running queries. It took **46.4 hours** to run the *cqa+reassess* approach and **99.9 hours** to run the *query+reassess* over the 20 pairs of ontologies, when it only took **2.0 hours** for the baseline or *query* versions. The baseline approach and the generated queries variants have approximately the same runtime over the 20 pairs of ontologies. However, for a similar runtime, the results of the approach with the CQA are better than those with the generated queries.

As expected, the reassessment phase decreases the number of correspondences as they are filtered. It entails an increase of the Precision. The Precision of *cqa+reassess* is between 8% and 15% higher than that of the baseline. The Precision of *query+reassess* is between 6% and 17% higher than that of the *query* variant.

The CQA Coverage remains the same for the baseline and *cqa+reassess*. The CQA Coverage score of *query+reassess* is about 3% lower than that of *query*. As more specific correspondences are preferred over more general ones during the similarity reassessment phase, it leaves less possibilities during the rewriting phase.

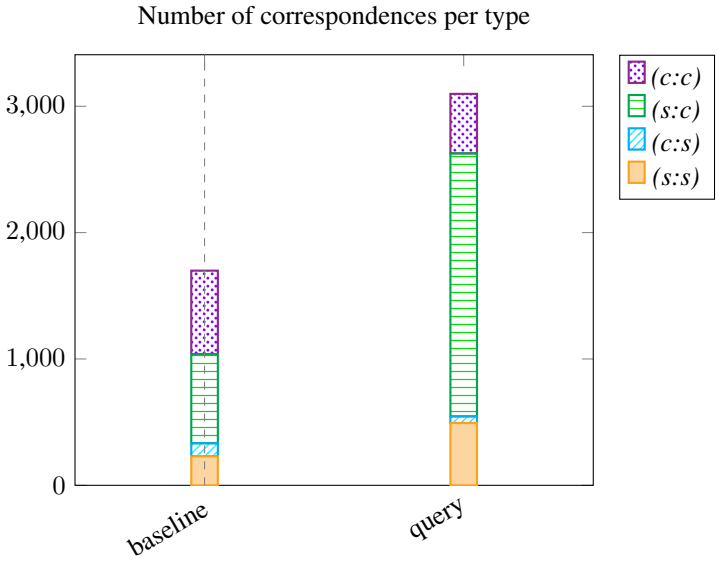


Fig. 13. Number of correspondence per type for the baseline and the variant which generates queries

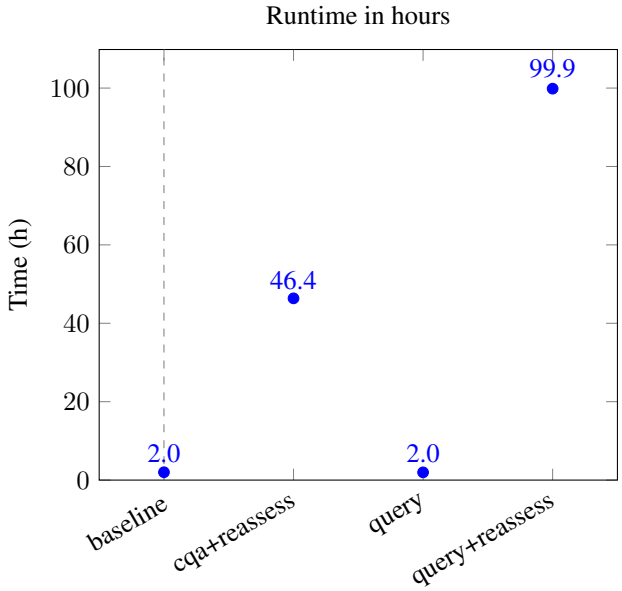


Fig. 14. Runtime of the baseline and its variants over the 20 oriented pairs of ontologies.

5.4. Comparison with existing approaches

The generated alignments were compared with three reference alignments and two complex alignment generation approaches:

Query rewriting the query rewriting oriented alignment set[6] from [29] - 10 pairs of ontologies
Ontology merging the ontology merging oriented alignment set[6] from [29] - 10 pairs of ontologies

[6]https://doi.org/10.6084/m9.figshare.4986368.v7

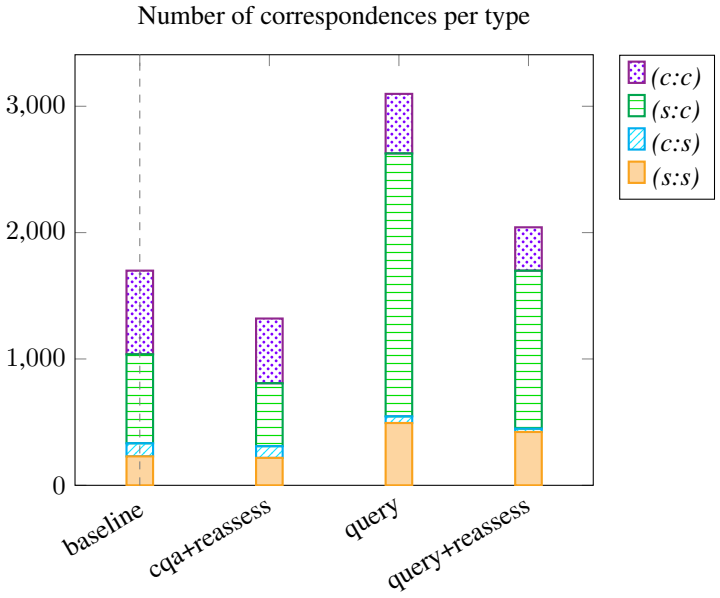


Fig. 15. Number of correspondence per type for the baseline, the variant which generates queries (*query*) and their equivalent variants with similarity reassessment based on counter-examples

ra1 the reference simple alignment[7] from the OAEI conference dataset [30] - 10 pairs of ontologies
Ritze 2010 the output alignment[8] from [4] - complex correspondences found on 4 pairs of ontologies
AMLC the output alignment[9] from [5] - output alignments between 10 pairs of ontologies

These approaches have been chosen because their implementations are available online and they output alignments in EDOAL. Ritze 2010 [4] and AMLC [5] both require simple alignments as input. They were run with ra1 as input. ra1 has then been added to Ritze 2010 and AMLC for the CQA Coverage evaluation. The Precision evaluation was made only on their output (ra1 correspondences excluded). Ritze 2010 took **58 minutes** while AMLC took about **3 minutes** to run over the 20 pairs of ontologies. Even though these two approaches are similar, this difference of runtime can be explained by the fact that Ritze 2010 loads the ontologies and parses their labels for each pattern while AMLC only loads the ontologies once. Moreover, Ritze 2010 covers 5 patterns while Faria only covers 2. Some refactoring was necessary so that the alignments could be automatically processed by the evaluation system. The ra1 dataset had to be transformed into EDOAL, instead of the basic alignment format. The Alignment API could not be used to perform this transformation as the type of entity (class, object property, data property) must be specified in EDOAL. The Ritze 2010 alignments used the wrong EDOAL syntax to describe some constructions (*AttributeTypeRestriction* was used instead of *AttributeDomainRestriction*). The AMLC alignments were not parsable because of RDF/XML syntax errors. The entities in the correspondences were referred to by their URI suffix instead of their full URI (*e.g.*, *Accepted_Paper* instead of *http://ekaw#Accepted_Paper*). Some correspondences were written in the wrong way: the source member was made out of entities from the target ontology and the target member made out of entities from the source ontology. As the evaluation of these alignments was manual so far in the OAEI complex track, these errors had not been detected. The alignments' syntax have been manually fixed so that they could be automatically evaluated.

Figure 17 shows the number of correspondences per type over the 20 pairs of ontologies. These alignments were not directional so their number of *(s:c)* and *(c:s)* correspondences are identical.

[7] http://oaei.ontologymatching.org/2018/conference/
[8] https://code.google.com/archive/p/generatingcomplexalignments/downloads/
[9] http://oaei.ontologymatching.org/2018/results/complex/conference/

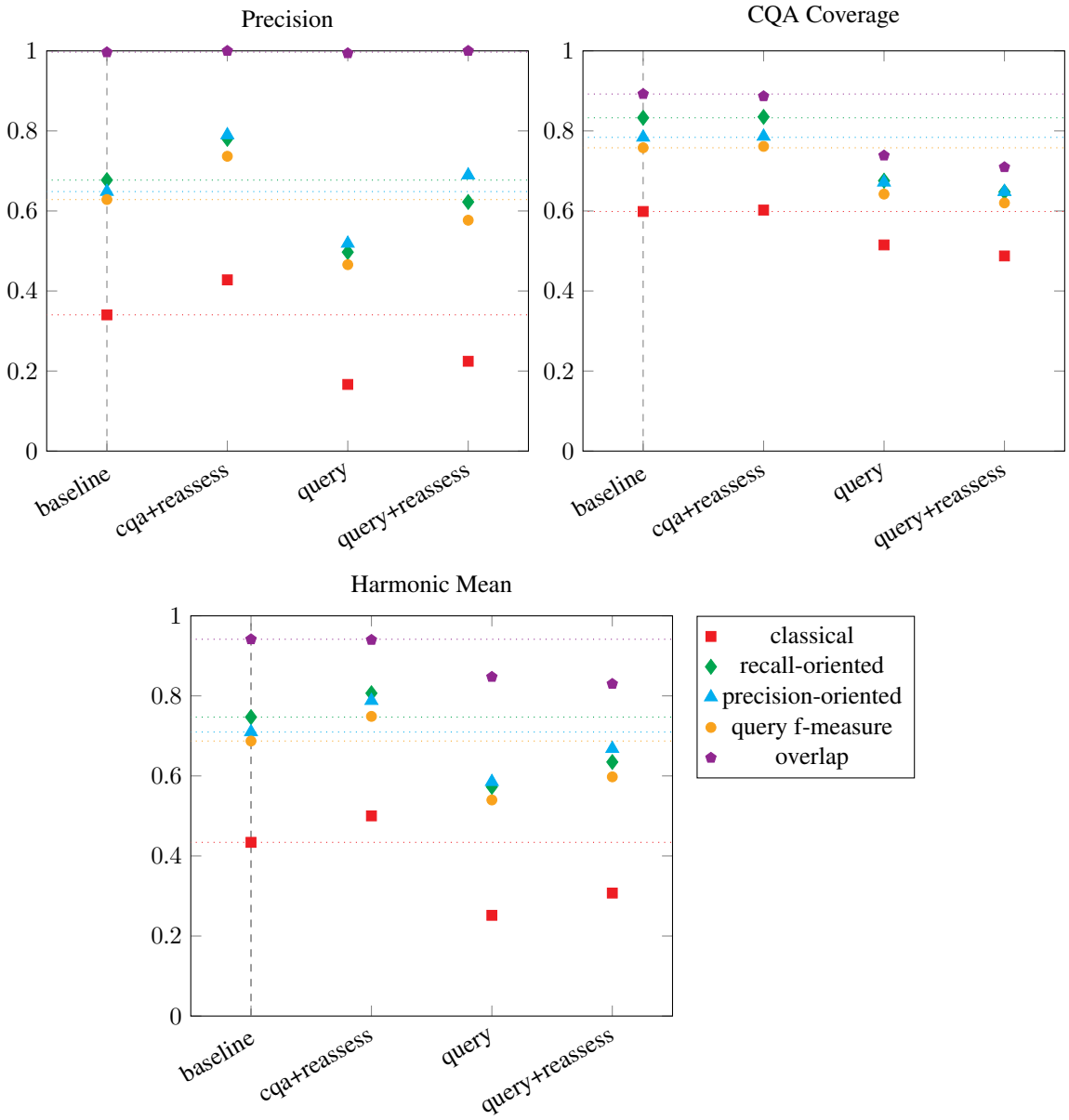


Fig. 16. Results for the baseline, the variant which generates queries (query) and their equivalent with a counter-example-based similarity reassessment

Figure 18 shows the results of the baseline approach (*baseline*), the baseline approach with counter-example-based similarity reassessment (*cqa + reassess*) and the compared alignments. The Precision results should be considered carefully. First of all, the relation of the correspondence is not considered in this score: all correspondences are compared as if they were equivalences. The Ontology merging and Query rewriting alignments contain a lot of correspondences with a subsumption relations so their classical Precision score is lower than the percentage of correct correspondences it contains. Second, the precision of the alignments is considered to be between the classical Precision and the percentage of correspondences whose members are either overlapping or both empty (*not disjoint*) due to the way the ontologies were populated.

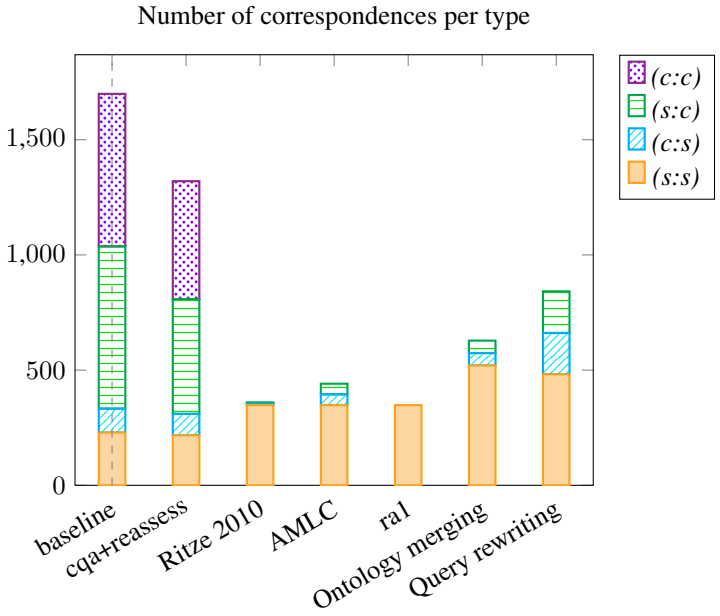


Fig. 17. Number of correspondence per type for the proposed approach, reference alignments and complex alignment generation approaches. The alignments of Ritze 2010 and AMLC include ra1.

Another limitation of the Precision score is related to the correspondences whose members are not populated in the dataset. For instance, $\langle$ *cmt:Preference , conference:Review_preference ,* $\equiv \rangle$ is a correct correspondence which was not detected as such in the Precision evaluation. The review preference of a reviewer for a paper was not part of the CQA for the population process. There is therefore no instance for either member of the correspondence.

To compensate these errors, we use the *not disjoint* scoring metric in the Precision evaluation. The score for a correspondence is 1 when the members are overlapping or both empty, and 0 otherwise. This metric gives the upper bound of the precision of an alignment. When calculating the Harmonic Mean of CQA Coverage and Precision, the *overlap* CQA Coverage was used with the *not disjoint* Precision score to give an upper bound. Indeed, in the CQA Coverage, the source query will never return empty results.

The CQA Coverage of Ritze 2010 and AMLC is higher than thats of ra1 which they include. Overall, the CQA Coverage of the other alignments (Ontology Merging, Query Rewriting, ra1, Ritze 2010 and AMLC) is lower than the score of our approach. Indeed, ra1 only contains simple equivalence correspondences, Ritze 2010 and AMLC are mostly restrained to finding *(s:c)* class expressions correspondences (and therefore do not cover binary CQA). The Ontology merging and query rewriting alignments are limited to *(s:c)*, *(c:s)* correspondences.

Globally, the Query rewriting alignment outperforms the Ontology merging in terms of CQA Coverage except for the *edas-confOf* pair. In the Ontology merging alignments, unions of properties were separated into individual subsumptions which were usable by the rewriting system. In the Query rewriting alignment, the subsumptions are unions.

CANARD obtains the best CQA Coverage scores except for the classical CQA Coverage where the Query rewriting alignment is slightly better (0.62 vs. 0.60). It can generate *(c:c)* correspondences which cover more CQA than the other alignments limited to *(s:s)*, *(s:c)* and *(c:s)*.

The Precision of our approach is overall lower than the Precision of reference alignments (considering that their Precision score is between the classical and not disjoint score). Ritze 2010 only outputs equivalent or disjoint correspondences. Its Precision score is therefore the same (0.75) for all metrics. AMLC achieves a better classical Precision than our baseline approach but contains a high number of disjoint correspondences (37% of all the output correspondences had members whose instance sets were disjoint).

Overall, as expected, the Precision scores of the reference alignments are higher than those output by the matchers. Our approach relies on CQAs and for this reason, it gets higher CQA Coverage scores than Ritze 2010 and AMLC. More over, these two matchers both rely on correspondence patterns which limit the types of correspondences they can generate.

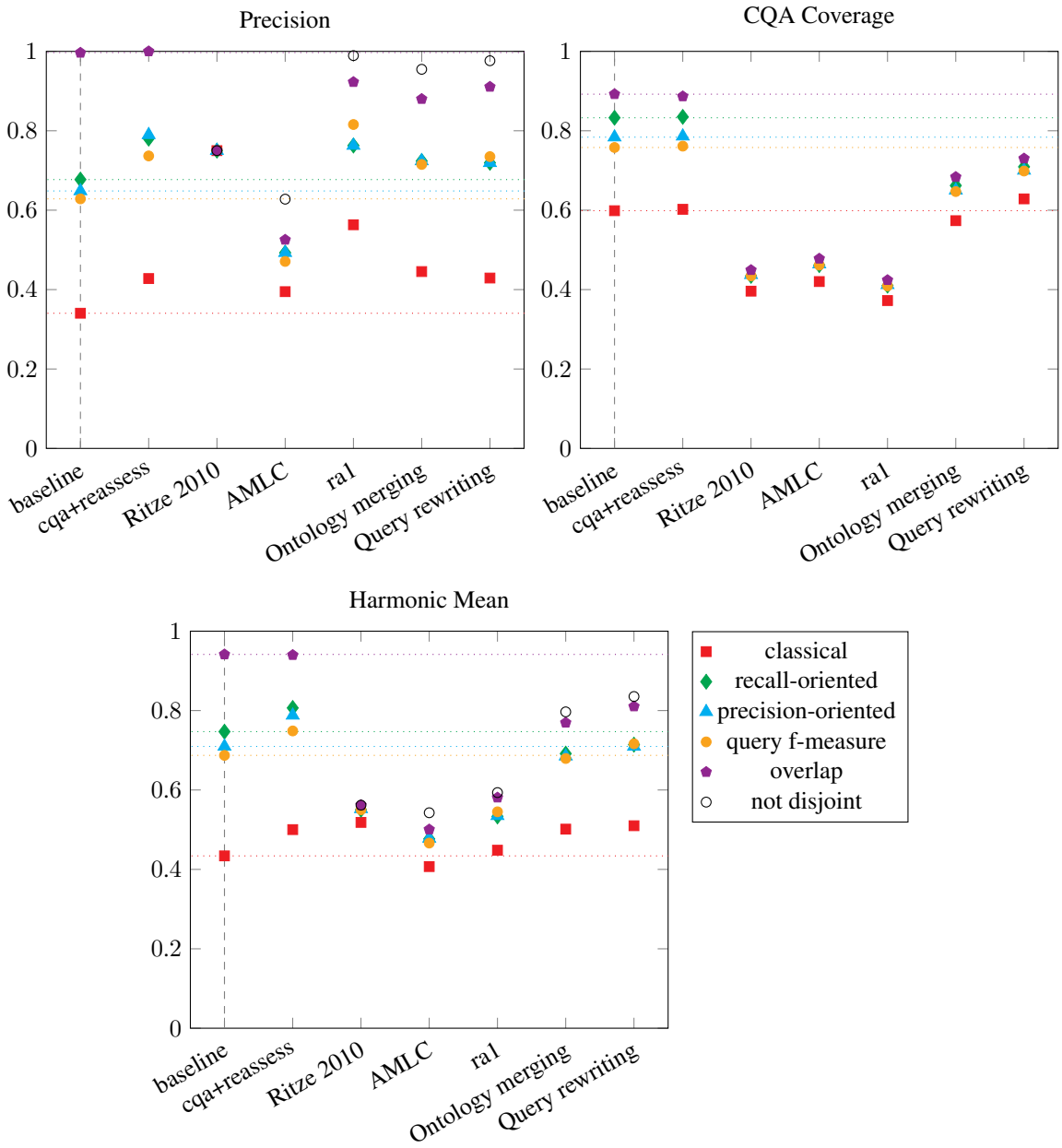


Fig. 18. Results of the proposed approach, reference alignments and complex alignment generation approaches.

5.5. *Evaluation on Taxon*

The Taxon dataset is composed of 4 ontologies which describe the classification of species: AgronomicTaxon [22], AgroVoc [23], DBpedia [24] and TaxRef-LD [25]. The CQA used in this evaluation are the one presented

in [26] which were manually written from AgronomicTaxon CQs [22]. The ontologies are are populated and their common scope is plant taxonomy. Their particularity, however, is that within a same dataset, the same information can be represented in various ways but irregularly across instances. For this reason, creating a set of reference and exhaustive CQA is not easily feasible.

The knowledge bases described by these ontologies are large. The English version of DBpedia describes more than 6.6 million entities alone and over 18 million entities[10]. The TaxRef-LD endpoint contains 2,117,434 instances[11] and the AgroVoc endpoint 754,874[11]. AgronomicTaxon has only been populated with the wheat taxonomy and only describes 32 instances. The approach has been run on the distant SPARQL endpoints but server exceptions have been encountered, probably due to an unstable network connection or an overload of the servers. A reduced version of the datasets was then stored on a local machine to avoid these problems. The reduced datasets contain all the plant taxa and their information (surrounding triples, annotations, *etc.*) from the SPARQL endpoint of the knowledge bases. Table 4 shows the number of plant taxa in each knowledge base. Even though the number of instances was reduced, the knowledge bases are still large-scale.

Table 4
Number of taxa and plant taxa in each knowledge base of the track, in its original and reduced version.

Version	AgronomicTaxon	AgroVoc	DBpedia	TaxRef-LD
Taxa (original)	32	8,077	306,833	570,531
Plant taxa (reduced)	32	4,563	58,257	47,058

The approach is run with the following settings: Levenshtein threshold: 0.4; Number of support answers: 1 and 10 (two runs); Instance matching: look for existing links (*owl:sameAs*, *skos:closeMatch*, *skos:exactMatch*) and if no target answer is found like that, perform an exact label match; No counter-example reassessment (computing the percentage of counter-examples would last too long on this dataset). The generated correspondences have been manually classified as equivalent, more general, more specific or overlapping. The classical, recall-oriented, precision-oriented and overlap scores have been calculated based on this classification.

5.5.1. Evaluation results

The number of correspondences per type is shown in Figure 19. The correspondences have been manually classified as equivalent, more general, more specific or overlapping. The classical, recall-oriented, precision-oriented and overlap scores have been calculated based on this classification. The results are shown in Figure 20.

Overall, the classical Precision and CQA Coverage scores are rather low. The Precision of the approach with 1 or 10 support answers is approximately the same. However, the CQA Coverage is higher with 10 instances. In comparison with the Conference dataset, this can be explained by the differences of population between the knowledge bases and the uneven population of a knowledge base in itself. We guess that the more support answers the approach takes, the best its CQA Coverage will be when dealing with unevenly populated ontologies.

The uneven population of some knowledge bases leads to missing correspondences. For example, *agronto:hasTaxonomicRank* is not represented for every instance of Agrovoc. *agrovoc:c_35661* which is the *Asplenium* genus taxon has no *agronto:hasTaxonomicRank* property. When this instance was used as support instance, it could not lead to the detection of a correspondence involving its rank. When running our matching approach with only 1 support instance, using this instance would result in an empty set of correspondences for some CQAs. Consequently, the CQA Coverage is globally higher for the approach with 10 support answers.

The particularity of a dataset about species taxonomy is that two taxa are likely to share the same scientific name. Our exact label match strategy is therefore rather suited for such a dataset. In some cases however, it introduced noise. For example, a confusion was made between *wheat* the plant taxon and *wheat* the consumable good, or between a *division*, part of an administrative structure and the taxonomic rank *division*.

The Levenshtein-based string similarity brings noise. For example, the correspondence $\langle$ *agrotaxon:GenusRank* , $\exists$*agronto:produces.*$\{$*agrovoc:c_8373*$\}$, $\equiv \rangle$ whose target member represents all the agronomic taxa which produce

[10] Statistics from the 2016-10 release https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10.
[11] Tested on 2019/04/12.

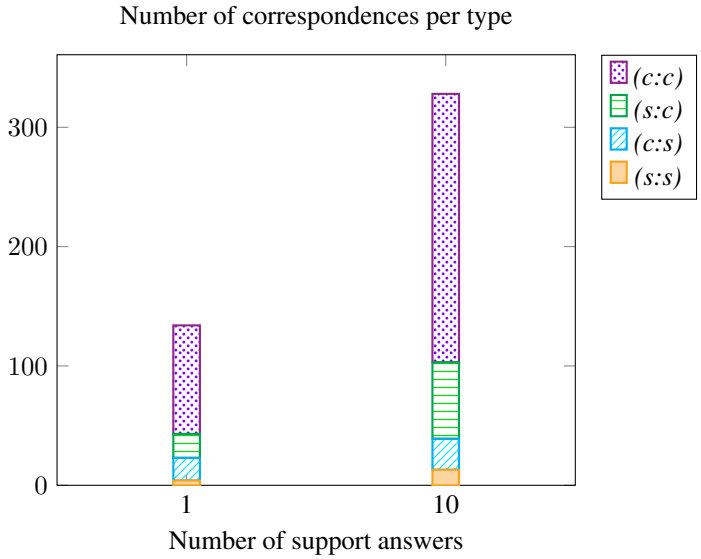


Fig. 19. Number of correspondence per type for the approach with 1 and 10 support answers on the Taxon dataset.

Table 5
Runtime (s) of our approach on each pair of ontologies. These measures are based on a single run.

	source \ target	AgronomicTaxon	AgroVoc	DBpedia	TaxRef-LD
1 sup. inst.	AgronomicTaxon	–	67	**4**	421
	AgroVoc	747	–	238	27,776
	DBpedia	**50,542**	2,733	–	2,477
	TaxRef-LD	4,517	5,758	4,017	–
10 sup. inst	source \ target	AgronomicTaxon	AgroVoc	DBpedia	TaxRef-LD
	AgronomicTaxon	–	1,084	1,019	753
	AgroVoc	1,173	–	**220**	29,351
	DBpedia	**52,214**	4,813	–	5,062
	TaxRef-LD	4,718	8,005	5,062	–

wheat has been output. This is due to the string similarity between the Malay label of wheat *"Gandum"* and the English label *"Genus"* of the *agrotaxon:GenusRank* class. We could have chosen to compare labels in the same language together but sometimes, the language of a label was missing, sometimes the scientific name was either tagged as English or Latin.

The total runtime over the 12 pairs of ontologies was **99,297s (27.6h)** for the approach with 1 support instance and **113,474s (31.5h)** for the approach with 10 support answers. The runtime per pair of ontologies is detailed in Table 5. Three factors explain the different runtime over the pairs of ontologies in Table 5.

Query difficulty Some CQA were really long to run on large knowledge bases, in particular those involving union of properties.

Percentage of source common instances The number of taxa instances can be different between knowledge-bases. AgronomicTaxon and DBpedia share 22 instances. When AgronomicTaxon, which has only 32 instances is matched to DBpedia, finding a common instance between the two is rather easy because about 68% of its instances have an equivalent in DBpedia. The other way around, is way harder because only 0.04% of DBpedia taxa instances have an equivalent in AgronomicTaxon.

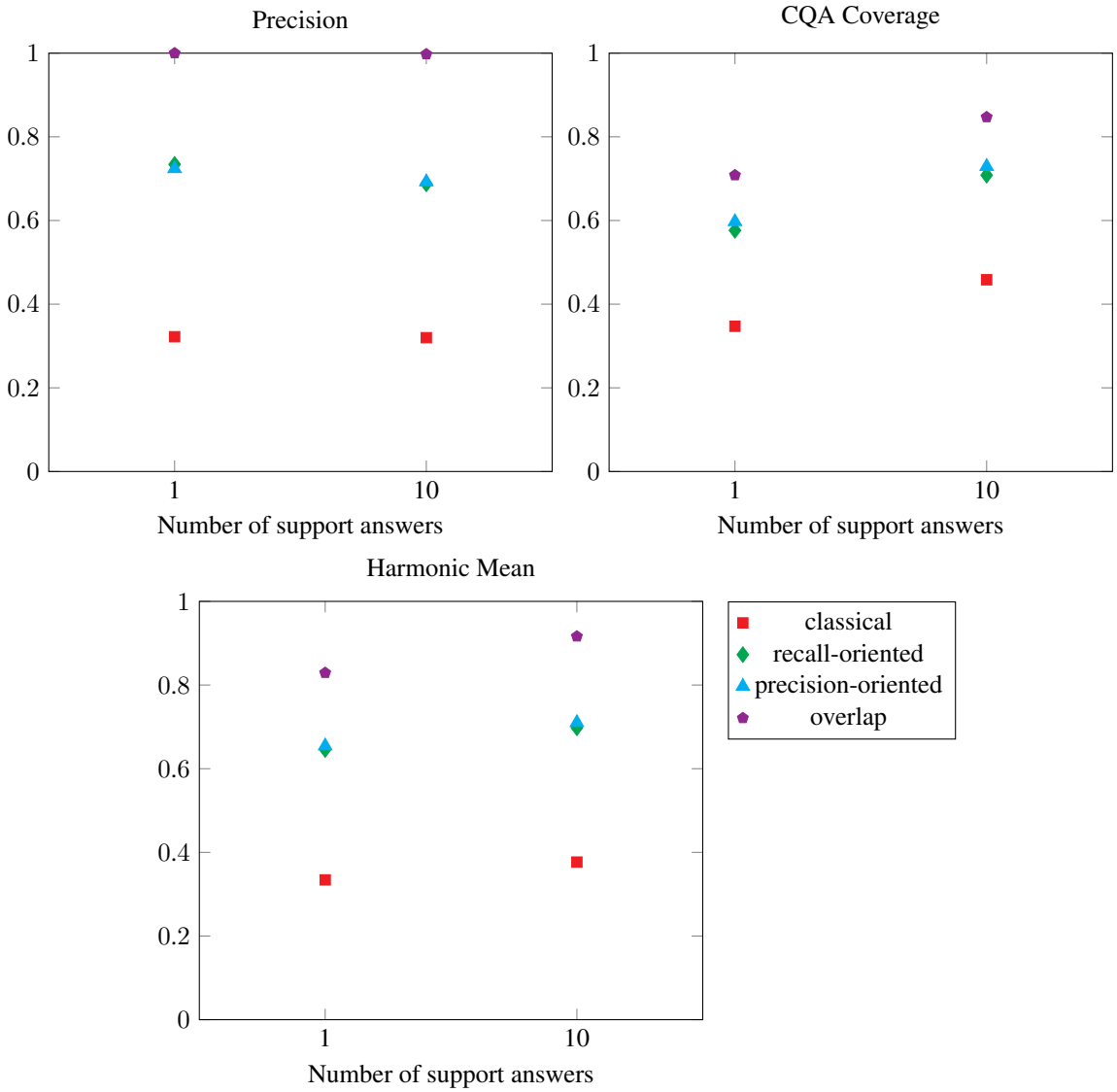


Fig. 20. Results of the approach with 1 and 10 support answers on the Taxon dataset.

Existence of instance links When no explicit instance links exist between two knowledge bases, all the source instances are explored and the exact label match is performed. This can take a lot of time according to the size of the target knowledge base.

6. Discussion

Even though the similarity metric in this implementation of the approach is naive, the results of the approach are quite high (the query f-measure Harmonic Mean score of the baseline approach is 0.70). The approach is rather CQA Coverage-oriented, as it will try to output a correspondence for each source CQA. The values of the CQA Coverage are overall higher than the Precision values. The baseline achieves a classical CQA Coverage of 0.60 which means that 60% of the CQA have been covered with a strictly equivalent match by our approach while its classical Precision score is only 0.34. Using existing identity links gives better results than exact label matches.

The use of CQA improves the Precision and CQA Coverage performance of the approach. The counter-example exploration (similarity reassessment phase) increases significantly the Precision but extends the runtime drastically. In comparison with the other matching approaches evaluated, our approach has high CQA Coverage scores. It would be interesting to compare our approach with extensional approaches such as [8, 9, 28, 31] (whose implementation was not available) even though all of them are limited to *(s:c)* and *(c:s)* correspondences. The experiment on the Taxon dataset showed that our approach can perform on large knowledge bases but its runtime can be very long. It also highlighted the need for regularly populated knowledge bases and quality instance links.

The evaluation described in Section 5 helped answer the research questions:

Is one common instance per Competency Question for Alignment enough evidence to generate complex correspondences? In the experiments on the Populated Conference benchmark and on the Taxon dataset, the approach based on only one common instance could generate complex correspondences. While in the Populated Conference dataset, the results with one support answer are slightly higher than with more support answers, in the Taxon dataset, they are lower. This can be explained by the irregular population of some Taxon dataset ontologies as well as the existence of inaccurate instance links. These aspects are also discussed in the next research question.

What is the impact of the number of support answers on the alignment quality? The impact of the number of support answers depends on the ontology population. In the experiment on the Taxon dataset, using 10 support answers instead of 1 improved the quality of the alignment. The reason is because the ontologies are not all regularly populated. The Precision score was about the same for 1 or 10 support answers while the CQA Coverage scores are about 12% higher with 10 support answers than with 1. In the Conference dataset which is regularly populated, using more support answers reduced the Precision score because noise was introduced. When dealing with many support answers, the noisy correspondences could be filtered out based on their frequency. For example, the formula $\exists$ *conference:has_the_last_name.*{*"Benson"*} only appears for one support instance of Person whereas *conference:Person* appears for all support answers. However, it was a choice in the approach design to not disregard "accidental" formulae (those which only appear for 1 answer and not in the other answers) because unevenly populated datasets may be faced with this problem. For example, in DBpedia, the taxonomic rank of a taxon can be represented in different ways: the label of a property (*e.g.*, a taxon is the *dbo:genus* of another taxon or has a *dbp:genus* literal), a link to the rank instance (*e.g.*, link to *dbr:Genus*), or the presence of a rank authority (*e.g.*, *dbp:genusAuthority*). The problem is that all the genus instances do not share the same representation. It is possible that among the genus rank instances, only one is represented as a genus rank thanks to the *dbp:genusAuthority*. This may seem statistically accidental but it is relevant to our problem.

What is the impact of the quality of the instance links on the generated alignments quality? If the links are expressed and not erroneous, the generated alignment will have a better Precision and CQA Coverage. If wrong links are used, as in the experiment with exact label matches, a lot of noise is introduced and the Precision of the alignment decreases. The CQA Coverage score also decreases because the noise can prevent correct support answers to be found and all the output correspondences for a given correspondence can be erroneous. The quality of the instance links impacts the runtime, Precision and CQA Coverage scores of our approach. This highlights the need for effective instance matching systems and the disambiguation of existing links.

Can Competency Questions for Alignment improve the Precision of generated alignments? Both the Precision and CQA Coverage scores are higher when the approach relies on CQA. The baseline and the *cqa+reassess* variants obtain a Precision score in average 15% above that of their generated query variants (*query* and *query+reassess*). The CQA Coverage also increases in average of 14% because the CQA help generate *(c:c)* correspondences which are relevant to the user (and to the evaluation). However, as part of the input CQA is used for the calculation of the CQA Coverage score, the evaluation is somewhat biased. In a user's need-oriented scenario, nonetheless, this evaluation makes sense: if users input their needs into a matcher, they may expect an output alignment which covers them well.

Does similarity reassessment based on counter-examples improve the quality of the generated alignments? When comparing the results of the baseline approach with the *cqa+reassess* variant which reassesses the similarity based on counter-examples, the CQA Coverage remains the same while the Precision is improved. The Preci-

sion of the *cqa+reassess* variant is between 8% and 15% higher than that of the baseline. The Precision of the *query+reassess* variant is between 6% and 17% higher than that of the *query* variant while its CQA Coverage is 3% lower.

Can CQAs improve the runtime performance of complex ontology matching? Comparing the *cqa+reassess* and *query+reassess* variants, the runtime is improved thanks to the use of CQA. However, when comparing the baseline *cqa* to the *query* variants, the runtime is the same. This depends on the complexity of the CQA. Our baseline approach took about 2 hours to align the 20 pairs of ontologies of the Populated Conference dataset. AMLC is much faster than our baseline approach as it took only 3 minutes. Ritze 2010 is also faster than our baseline approach. However, when running with only 1 support answer, our approach takes 33 minutes which make it faster than Ritze 2010.

What is the impact of the CQA on the type of output correspondence? Overly complex correspondences can be introduced in the alignment because of the way the approach uses the input CQA. We counted that about 14% of the *(c:c)* correspondences output by the baseline approach are overly complex, which means that they could be decomposed into simple correspondences. This comes from the translation of the input CQA into a DL formula without any analysis or decomposition of its elements. Moreover, the approach outputs more *(s:c)* and *(c:c)* correspondences than *(s:s)* and *(c:s)* which shows a tendency to output more complex than simple correspondences.

7. Related work

Classification of the approach CANARD is positionned using the characteristics in [12]. CANARD can generate *(s:s), (s:c)* and *(c:c)* correspondences depending on the shape of the input CQA. It focuses on correspondences with logical constructors. The approach relies on a path to find the correspondences for binary CQAs. For the unary CQAs, we classify CANARD as *no structure* because it does not explicitly rely on atomic or composite patterns. The source member form is fixed before the matching process by the CQA but the target member form is unfixed, therefore we classify it as *fixed to unfixed*. CANARD relies on ontology and instance-level evidence. CANARD fits in the formal resource-based because it relies on CQA and existing instance links, its implementation is string-based because of the label similarity metric chosen (see Section 5.1), it is also graph-based and instance-based.

Comparison to other matching approaches The matching approaches generating expressive correspondences involve different techniques such as relying on templates (called patterns) and/or instance evidence. The approaches in [3, 4] apply a set of matching conditions (label similarity, datatype compatibility, etc.) to detect correspondences that fit certain patterns. The approach of [32] uses the linguistic frames defined in FrameBase to find correspondences between object properties and the frames. KAOM [6] relies on *knowledge rules* which can be interpreted as probable axioms. The approaches in [8, 9] use statistical information based on the linked instances to find correspondences fitting a given pattern. The one in [11] uses a path-finding algorithm to find correspondences between two knowledge bases with common instances. The one in [31] iteratively constructs correspondences based on the information gain from matched instances between the two knowledge-bases. [5] relies on lexical similarity and structural conditions to detect correspondence patterns, close to [3]. None of these approaches involve, however, the user before or during the matching process. As in [8, 9, 11, 28, 31], CANARD relies common instances. Differently from them, it does not rely on correspondence patterns. Finally, CQA have not been adapted nor used for matching.

SPARQL CQA In our approach, CQA are used as basic pieces of information which will be transformed as source members of correspondences. Their formulation in a SPARQL query over the source ontology is a limitation of the approach as a user would need to be familiar with SPARQL and the source ontology. However, in the scenario where someone wants to publish and link a knowledge base he or she created on the LOD cloud, this person is already familiar with the source ontology and can reuse the CQ of their own ontology. In other cases, one could rely on question answering systems which generate a SPARQL query from a question in natural language. This kind of system is evaluated in the QALD open challenge [33].

Generalisation process Ontology matching approaches relying instances infer general statements, *i.e.*, they perform a generalisation[12]. This is the principle of *machine learning* in general and methods such as *Formal Concept Analysis* [34] or *association rule mining* [35]. These generalisation processes however require a considerable amount of data (or instances). Approaches such as the ones from [8, 9, 28, 31] rely on large amounts of common ontology instances for finding complex correspondences. Few exceptions in ontology matching rely on few examples. For instance, the matcher of [36] relies on example instances given by a user. With this information, the generalisation can be performed on few examples. The idea behind our approach is to rely on a few examples to find general rules which would apply to more instances. In particular, the generalisation phase of our approach is guided by the CQA labels. Thanks to that, only one instance is sufficient for finding a correspondence. This would apply to knowledge bases which represent different contexts or points of view but whose ontologies are overlapping.

8. Conclusions

This paper has presented a complex alignment generation approach based on CQAs. The CQA define the knowledge needs of a user over two or more ontologies. The use of CQAs is both a strength of the approach as it allows for a generalisation over few instances and a limitation as it requires that the user is able to express her or his needs as SPARQL queries. It depends as well on the quality of the instance matches. The approach can be extended in several directions: one could consider exploring more sophisticated instance-based matching approaches and, alternatively, conditional or link keys (systems generating keys could also benefit from complex correspondences to improve their results); designing a purely T-Box strategy based on both linguistic and semantic properties of the ontologies and CQAs; or still dividing the problem in sub-tasks through ontology partitioning (given the inherent high search space in this task). Last but not least, incoherence resolution systems for complex alignments are scarce.

References

[1] P.R. Visser, D.M. Jones, T.J. Bench-Capon and M. Shave, An analysis of ontology mismatches: heterogeneity versus interoperability, in: *AAAI 1997 Spring Symposium on Ontological Engineering, Stanford CA., USA*, 1997, pp. 164–72.

[2] A. Maedche, B. Motik, N. Silva and R. Volz, MAFRA - A MApping FRAmework for Distributed Ontologies, in: *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, A. Gómez-Pérez and V.R. Benjamins, eds, Lecture Notes in Computer Science, Vol. 2473, Springer, 2002, pp. 235–250. doi:10.1007/3-540-45810-7_23.

[3] D. Ritze, C. Meilicke, O. Šváb-Zamazal and H. Stuckenschmidt, A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences, in: *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) Chantilly, USA, October 25, 2009*, P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, N.F. Noy and A. Rosenthal, eds, CEUR Workshop Proceedings, Vol. 551, CEUR-WS.org, 2009.

[4] D. Ritze, J. Völker, C. Meilicke and O. Šváb-Zamazal, Linguistic analysis for complex ontology matching, in: *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010), Shanghai, China, November 7, 2010*, P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, M. Mao and I.F. Cruz, eds, CEUR Workshop Proceedings, Vol. 689, CEUR-WS.org, 2010.

[5] D. Faria, C. Pesquita, B.S. Balasubramani, T. Tervo, D. Carriço, R. Garrilha, F.M. Couto and I.F. Cruz, 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, October 8, 2018*, P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, M. Cheatham and O. Hassanzadeh, eds, CEUR Workshop Proceedings, Vol. 2288, CEUR-WS.org, 2018, pp. 125–131.

[6] S. Jiang, D. Lowd, S. Kafle and D. Dou, Ontology matching with knowledge rules, in: *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXVIII*, Vol. 28, Springer, 2016, pp. 75–95. doi:10.1007/978-3-662-53455-7_4.

[7] L. Zhou, M. Cheatham and P. Hitzler, AROA Results for 2019 OAEI, in: *Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26, 2019*, 2019, pp. 107–113. http://ceur-ws.org/Vol-2536/oaei19_paper4.pdf.

[8] R. Parundekar, C.A. Knoblock and J.L. Ambite, Discovering Concept Coverings in Ontologies of Linked Data Sources, in: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J.X. Parreira, J. Hendler, G. Schreiber, A. Bernstein and E. Blomqvist, eds, Lecture Notes in Computer Science, Vol. 7649, Springer, 2012, pp. 427–443. doi:10.1007/978-3-642-35176-1_27.

[12] 'They infer general statements or concepts from specific cases' (Oxford Dictionary, "Generalisation" Retrieved June 3 2019 from https://en.oxforddictionaries.com/definition/generalization

[9] B. Walshe, R. Brennan and D. O'Sullivan, Bayes-ReCCE: A Bayesian Model for Detecting Restriction Class Correspondences in Linked Open Data Knowledge Bases, *Int. J. Semant. Web Inf. Syst.* **12**(2) (2016), 25–52, ISSN 1552-6283. doi:10.4018/IJSWIS.2016040102.

[10] B.P. Nunes, A.A.M. Caraballo, M.A. Casanova, K.K. Breitman and L.A.P.P. Leme, Complex matching of RDF datatype properties, in: *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*, P. Shvaiko, J. Euzenat, T. Heath, C. Quix, M. Mao and I.F. Cruz, eds, CEUR Workshop Proceedings, Vol. 814, CEUR-WS.org, 2011.

[11] H. Qin, D. Dou and P. LePendu, Discovering Executable Semantic Mappings Between Ontologies, in: *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I*, R. Meersman and Z. Tari, eds, Lecture Notes in Computer Science, Vol. 4803, Springer, 2007, pp. 832–849. doi:10.1007/978-3-540-76848-7_56.

[12] É. Thiéblin, O. Haemmerlé, N. Hernandez and C. Trojahn, Survey on complex ontology matching, *Semantic Web* **11**(4) (2020), 689–727. doi:10.3233/SW-190366. https://doi.org/10.3233/SW-190366.

[13] É. Thiéblin, O. Haemmerlé and C. Trojahn, Generating Expressive Correspondences: An Approach Based on User Knowledge Needs and A-Box Relation Discovery, in: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I*, J.Z. Pan, V.A.M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne and L. Kagal, eds, Lecture Notes in Computer Science, Vol. 12506, Springer, 2020, pp. 565–583. doi:10.1007/978-3-030-62419-4_32. https://doi.org/10.1007/978-3-030-62419-4_32.

[14] J. Euzenat and P. Shvaiko, *Ontology Matching, Second edition*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-38720-3 978-3-642-38721-0.

[15] M. Grüninger and M.S. Fox, Methodology for the Design and Evaluation of Ontologies. International Joint Conference on Artificial Inteligence, in: *Workshop on Basic Ontological Issues in Knowledge Sharing*, Vol. 15, 1995.

[16] Y. Ren, A. Parvizi, C. Mellish, J.Z. Pan, K. van Deemter and R. Stevens, Towards Competency Question-Driven Ontology Authoring, in: *The Semantic Web: Trends and Challenges*, Vol. 8465, Springer, 2014, pp. 752–767. ISBN 978-3-319-07442-9 978-3-319-07443-6. doi:10.1007/978-3-319-07443-6_50.

[17] E. Thiéblin, O. Haemmerlé and C. Trojahn, Complex matching based on competency questions for alignment: a first sketch, 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, October 8, 2018*, CEUR Workshop Proceedings, Vol. 2288, CEUR-WS.org, 2018, pp. 66–70.

[18] A. Borgida, On the relative expressiveness of description logics and predicate logics, *Artificial intelligence* **82**(1–2) (1996), 353–367.

[19] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song and D. Zhao, Semantic SPARQL Similarity Search over RDF Knowledge Graphs, *Proceedings of the VLDB Endowment* **9**(11) (2016), 840–851, ISSN 2150-8097. doi:10.14778/2983200.2983201. http://dx.doi.org/10.14778/2983200.2983201.

[20] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, *Soviet physics doklady* **10**(8) (1966), 707–710.

[21] É. Thiéblin and C. Trojahn, Conference v3.0 : A populated version of the Conference dataset, in: *ISWC Poster Track*, 2019.

[22] C. Roussey, J. Chanet, V. Cellier and F. Amarger, Agronomic taxon, in: *Proceedings of the 2nd International Workshop on Open Data, WOD 2013, Paris, France, June 3, 2013*, V. Christophides and D. Vodislav, eds, ACM, 2013, pp. 5–154. doi:10.1145/2500410.2500415. https://doi.org/10.1145/2500410.2500415.

[23] C. Caracciolo, A. Stellato, S. Rajbahndari, A. Morshed, G. Johannsen, J. Keizer and Y. Jaques, Thesaurus maintenance, alignment and publication as linked data: the AGROVOC use case, *International Journal of Metadata, Semantics and Ontologies* **7**(1) (2012), 65, ISSN 1744-2621, 1744-263X. doi:10.1504/IJMSO.2012.048511. http://www.inderscience.com/link.php?id=48511.

[24] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, DBpedia: A Nucleus for a Web of Open Data, in: *The Semantic Web: The 6th International Semantic Web Conference ISWC and the 2nd Asian Semantic Web Conference ASWC*, K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber and P. Cudré-Mauroux, eds, LNCS, Vol. 4825, Springer Berlin Heidelberg, Busan, Korea, 2007, pp. 722–735. ISBN 978-3-540-76297-3 978-3-540-76298-0.

[25] F. Michel, O. Gargominy, S. Tercerie and C. Faron-Zucker, A Model to Represent Nomenclatural and Taxonomic Information as Linked Data.Application to the French Taxonomic Register, TAXREF, in: *Proceedings of the 2nd International Workshop on Semantics for Biodiversity (S4BioDiv 2017) co-located with 16th International Semantic Web Conference (ISWC 2017)*, Vol. 1933, A. Algergawy, N. Karam, F. Klan and C. Jonquet, eds, CEUR-WS.org, Vienna, Austria, 2017. http://ceur-ws.org/Vol-1933/paper-3.pdf.

[26] E. Thiéblin, N. Hernandez, C. Roussey and C. Trojahn, Cross-querying LOD data sets using complex alignments: an experiment using AgronomicTaxon, Agrovoc, DBpedia and TAXREF-LD, *IJMSO* **13**(2) (2018), 104–119. doi:10.1504/IJMSO.2018.098387. https://doi.org/10.1504/IJMSO.2018.098387.

[27] É. Thiéblin, O. Haemmerlé and C. Trojahn, Automatic evaluation of complex alignments: An instance-based approach, *Semantic Web* **12**(5) (2021), 767–787. doi:10.3233/SW-210437. https://doi.org/10.3233/SW-210437.

[28] R. Parundekar, C.A. Knoblock and J.L. Ambite, Linking and Building Ontologies of Linked Data, in: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, P.F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J.Z. Pan, I. Horrocks and B. Glimm, eds, Lecture Notes in Computer Science, Vol. 6496, Springer, 2010, pp. 598–614. doi:10.1007/978-3-642-17746-0_38.

[29] E. Thiéblin, O. Haemmerlé, N. Hernandez and C. Trojahn, Task-Oriented Complex Ontology Alignment: Two Alignment Evaluation Sets, in: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, A. Gangemi, R. Navigli, M. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam, eds, Lecture Notes in Computer Science, Vol. 10843, Springer, 2018, pp. 655–670. doi:10.1007/978-3-319-93417-4_42.

[30] O. Šváb-Zamazal and V. Svátek, The Ten-Year OntoFarm and its Fertilization within the Onto-Sphere, *Web Semantics: Science, Services and Agents on the World Wide Web* **43** (2017), 46–53, ISSN 15708268. doi:10.1016/j.websem.2017.01.001.

[31] W. Hu, J. Chen, H. Zhang and Y. Qu, Learning complex mappings between ontologies, in: *The Semantic Web - Joint International Semantic Technology Conference, JIST 2011, Hangzhou, China, December 4-7, 2011. Proceedings*, J.Z. Pan, H. Chen, H. Kim, J. Li, Z. Wu, I. Horrocks, R. Mizoguchi and Z. Wu, eds, Lecture Notes in Computer Science, Vol. 7185, Springer, 2011, pp. 350–357. doi:10.1007/978-3-642-29923-0_24.

[32] J. Rouces, G. de Melo and K. Hose, Complex Schema Mapping and Linking Data: Beyond Binary Predicates, in: *Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016)*, S. Auer, T. Berners-Lee, C. Bizer and T. Heath, eds, CEUR Workshop Proceedings, Vol. 1593, CEUR-WS.org, 2016.

[33] C. Unger, C. Forascu, V. López, A.N. Ngomo, E. Cabrio, P. Cimiano and S. Walter, Question Answering over Linked Data (QALD-4), in: *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*, L. Cappellato, N. Ferro, M. Halvey and W. Kraaij, eds, CEUR Workshop Proceedings, Vol. 1180, CEUR-WS.org, 2014, pp. 1172–1180. http://ceur-ws.org/Vol-1180/CLEF2014wn-QA-UngerEt2014.pdf.

[34] B. Ganter, G. Stumme and R. Wille (eds), Formal Concept Analysis, Foundations and Applications, in *Lecture Notes in Computer Science*, Vol. 3626, Springer, 2005. ISBN 3-540-27891-5. doi:10.1007/978-3-540-31881-1. https://doi.org/10.1007/978-3-540-31881-1.

[35] R. Agrawal, T. Imieliński and A. Swami, Mining association rules between sets of items in large databases, in: *Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, USA, May 26-28 1993*, Vol. 22, S. Jajodia and P. Buneman, eds, ACM, 1993, pp. 207–216.

[36] B. Wu and C.A. Knoblock, An Iterative Approach to Synthesize Data Transformation Programs, in: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Q. Yang and M.J. Wooldridge, eds, AAAI Press, 2015, pp. 1726–1732.