Generating Expressive Correspondences: An Approach Based on User Knowledge Needs and A-Box Relation Discovery

Elodie Thiéblin, Ollivier Haemmerlé, and Cássia Trojahn(✉)

IRIT & Université de Toulouse 2 Jean Jaurès, Toulouse, France
elodie@thieblin.fr, {ollivier.haemmerle,cassia.trojahn}@irit.fr

Abstract. Ontology matching aims at making different ontologies interoperable. While most approaches have addressed the generation of simple correspondences, more expressiveness is required to better address the different kinds of ontology heterogeneities. This paper presents an approach for generating complex 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 scope. 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 lexically similar surroundings of the instances from the target ontology. Evaluation of the approach has been carried out on both synthetically generated and real-word datasets.

1 Introduction

Ontology matching aims at enabling interoperability between knowledge expressed in different ontologies. It is an active area with different solutions been proposed from various disciplines, e.g., databases, statistical, natural language processing, and machine learning. The matching process can be seen as the task of generating a set of correspondences (i.e., an alignment) between the entities of different ontologies, usually one source and one target ontologies [5]. Despite the variety of proposals, most of the matching approaches 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). This kind of correspondence is however not expressive enough for fully covering the different kinds of ontology heterogeneities (lexical, semantic, conceptual). Complex correspondences (i.e., those involving logical constructors or transformation functions) are rather required [27]. For example, the piece of knowledge expressing an accepted paper can be represented as a class IRI (e.g., *Accepted_Paper*) in a source ontology, or as a class expression representing the papers having acceptance as decision of type in a target ontology (e.g., $Paper \sqcap \exists\, hasDecision.Acceptance$). Expressive correspondences are required for expressing these different representations. For citing some applications and domains requiring

J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12506, pp. 565–583, 2020.
https://doi.org/10.1007/978-3-030-62419-4_32

such kind of correspondences, in the cultural heritage domain, the need for complex correspondences has been identified for data integration or data translation applications [15]. In the agronomic domain, complex alignments help cross-query linked open data repositories [24]. In the biomedical domain, complex alignments have also been used to build a consensual model from heterogeneous terminologies [11].

While the matching space for generating complex correspondences is higher than $\mathcal{O}(2^{mn})$ (m and n being respectively the number of entities of the source and target ontologies), a space reduction strategy can be based upon on an assumption that, in practical, it may be the case that the user does not need the alignment to cover the full scope of the ontologies. This assumption goes in opposite to the existing complex alignment generation approaches which intend to cover the full common scope of the aligned ontologies. This has the side effect of neglecting the user needs. This also impacts the matching performance, in particular when dealing with large knowledge bases.

This paper presents an approach for generating complex correspondences that relies on competency questions for alignment (CQAs) as a way of expressing user knowledge needs in terms of alignment. This approach is based on the following hypothesis: (i) users are able to express their needs in terms of SPARQL queries; and (ii) for each knowledge need, the knowledge bases share at least one common instance.

Based on these hypothesis, the approach takes as input a set of CQAs translated into SPARQL queries over the source ontology. Each answer is a set of instances retrieved from a knowledge base described by the source ontology. At least one instance is common with respect to the target knowledge base. The generation of the correspondence is then performed by matching the subgraph from the source CQA to the lexically similar surroundings of the target instances.

The main contributions of this paper can be summarised as follows: (i) a novel notion of competency question for alignment is introduced as a way for reducing the matching scope (this notion can hence be applied on the generation of simple correspondences); (ii) a CQA based matching approach able to generate complex correspondences involving logical constructions[1]; (iii) a comparison of the proposed approach to state-of-the-art ones; and (iv) a discussion of their strengths and weaknesses.

The rest of the paper is organised as follows. Next section introduces ontology matching and CQA (Sect. 2), followed by the presentation of the approach (Sect. 3). The evaluation is presented (Sect. 4), followed by a discussion on the related work (Sect. 5). Finally, the conclusions and future work end the paper (Sect. 6).

[1] Complex correspondences with transformations functions are out of the scope of this paper.

2 Foundations

2.1 Complex Ontology Alignment

Ontology matching (as in [5]) is defined as the process of generating an alignment A between two ontologies: a source ontology o_1 and a target ontology o_2. 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_1 and o_2 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): $\langle o_1$*:Paper*, o_2*:Paper*, $\equiv$, *1*$\rangle$
- a correspondence is **complex** if at least one of e_1 or e_2 involves a constructor: $\langle o_1$*:Accepted_Paper*, $\exists o_2$*:hasDecision.*o_2*:Acceptance*, $\equiv$, *1*$\rangle$

A simple correspondence is usually noted (s:s), and a complex correspondence can be (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" in the remainder of this paper.

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* [8]. A competency question 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 in the best case (if both ontologies' scopes can answer the CQA). The first difference between CQA and 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 it is the purpose of the alignment). As the ontology authoring competency questions (CQs) [19], a CQA can be expressed in natural language or as SPARQL queries. Inspired [19], the notion of **question arity**, which represents the arity of the expected answers to a CQA is introduced:

- A *unary* question expects a set of instances, e.g., "Which are the accepted papers?" *(paper1), (paper2).*
- A *binary* question expects a set of instances or value pairs, e.g., "What is the decision on a paper?" *(paper1, accept), (paper2, reject).*
- A *n-ary* question expects a tuple of size $\geq$ 3, e.g., "What is the decision associated with the review of a given paper?" *(paper1, review1, weak accept), (paper1, review2, reject).*

CQAs for the approach are limited to *unary* and *binary* questions, of *select* type, and no modifier. This is a limitation in the sense that we do 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.

3 Matching Approach

The proposed approach takes as input a set of CQAs in the form of SPARQL SELECT queries over the source ontology. It requires that the source and target ontologies have an *Abox* 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 overall approach is articulated in 11 steps (Fig. 1). It is based on **subgraphs** which are a **set of triples** for a unary CQA and a **property path** for a binary CQA. The implementation of the approach is publicly available[2].

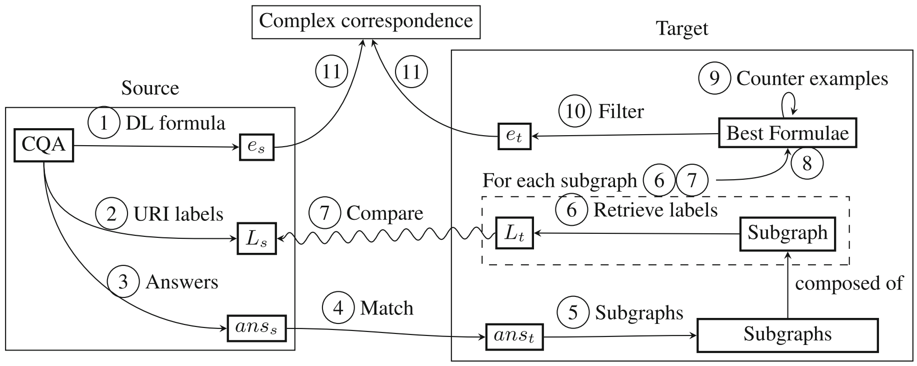
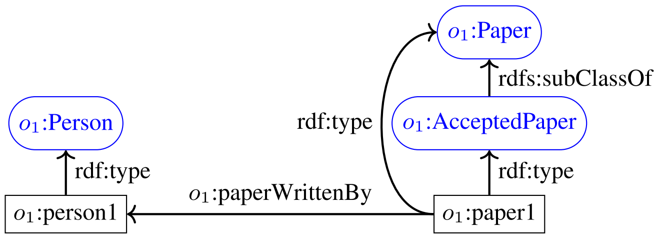


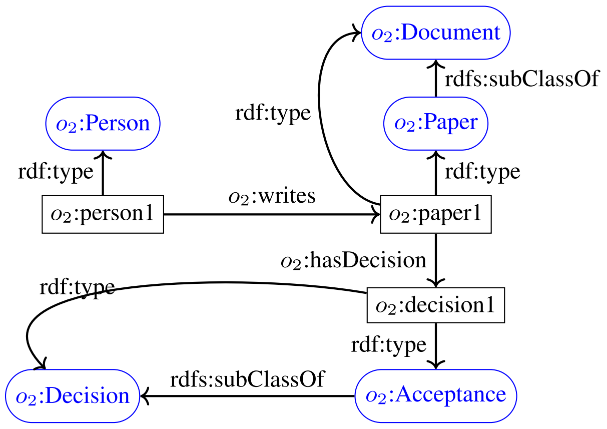
Fig. 1. Schema of the general approach.

In the remainder of the paper, the examples consider the knowledge bases in Fig. 2. They share common instances: o_1*:person1* and o_2*:person1*, o_1*:paper1* and o_2*:paper1*. Ontology o_1 represents the concept of *accepted paper* as a class while o_2 models the same knowledge with a *has decision* property. The property *paper written by* is represented by a single property in o_1 while in o_2, the property *writes* links a *person* to a *document*. A criticism to this example could be that

[2] https://framagit.org/IRIT_UT2J/ComplexAlignmentGenerator.



(a) Source knowledge base



(b) Target knowledge base

Fig. 2. Source and target knowledge bases.

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_1 could focus on the event organisation part of a conference and o_2 on reviewer management. Before detailing the main steps of the approach, we instantiate the overall approach to deal with unary and binary queries.

3.1 Approach over a Unary CQA

We instanciate Fig. 1 for a unary CQA. The SPARQL CQA is that of Fig. 3.

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

Fig. 3. SPARQL SELECT query with one variable.

① Represent the CQA as a DL formula e_s (*e.g.*, o_1*:AcceptedPaper*) (Sect. 3.3).

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_1*:paper1*).

④ Find equivalent or similar target answers ans_t to the source instances ans_s (*e.g.* o_1*:paper1* $\sim$ o_2*:paper1*) (Sect. 3.4).

⑤ Extract the subgraphs of target answers (Sect. 3.5): 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_2*:paper1* would contain $\langle$ o_2*:paper1* , o_2*:hasDecision* , o_2*:decision1* $\rangle$, $\langle$ o_2*:decision1* , *rdf:type* , o_2*:Decision* $\rangle$ *and* $\langle$ o_2*:decision1* , *rdf:type* , o_2*:Acceptance* $\rangle$.

⑥ For each subgraph, retrieve L_t the labels of its entities (*e.g.*, o_2*:hasDecision* $\rightarrow$ "decision", o_2*:decision1* $\rightarrow$ "decision for paper1", o_2*:Decision* $\rightarrow$ "decision").

⑦ Compare L_s and L_t (using an edit distance metric).

⑧ Select the subgraphs parts with the best similarity score, transform them into DL formulae (Sect. 3.5) and aggregate them (Sect. 3.6). In this example, the part of the subgraph which is the most similar to the CQA (in terms of label similarity) is o_2*:Acceptance*. The DL formula is therefore $\exists o_2$*:hasDecision*.o_2*:Acceptance*.

⑨ Reassess the similarity of each DL formula based on their counter-examples (Sect. 3.7 and Sect. 3.8). 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) (Sect. 3.9).

⑪ Put the DL formulae e_s and e_t together to form a correspondence (*e.g.*, $\langle$ o_1*:AcceptedPaper* , $\exists$ o_2*:hasDecision*.o_2*: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

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.

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

Fig. 4. SPARQL SELECT query with two variables.

① Extract source DL formula e_s (*e.g.*, o_1*:paperWrittenBy*) from SPARQL CQA (Fig. 4).

② Extract lexical information from the CQA, L_s set labels of atoms 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_1*:paper1,* o_1*:person1)*.

④ Find equivalent or similar target answers ans_t to the source instances ans_s (*e.g.* o_1*:paper1* $\sim$ o_2*:paper1* and o_1*:person1* $\sim$ o_2*:person1*).

⑤ Retrieve the subgraphs of target answers: 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_2*:paper1* and o_2*:person1*). The path is composed of only one property and there are no other instances than o_2*:paper1* and o_2*:person1* in this path. Their respective types are retrieved: (o_2*:Paper*,o_2*:Document*) for o_2*:paper1* and (o_2*:Person*) for o_2*:person1*.

⑥ For each subgraph, retrieve L_t the labels of its entities (*e.g.*, o_2*:writes* $\rightarrow$ "writes", o_2*:Person* $\rightarrow$ "person", o_2*:Paper* $\rightarrow$ "paper", *etc.*).

⑦ Compare L_s and L_t.

⑧ Select the subgraph parts with the best score, transform them into DL formulae. Keep the best path variable types if their similarity is higher than a threshold. (*e.g.*, the best type for the instance o_2*:paper1* is o_2*:Paper* because its similarity with the CQA labels is higher than the similarity of o_2*:Document*).

⑨ Reassess the similarity of each DL formula based on their counter-examples.

⑩ Filter the DL formulae based on their similarity score (if their similarity score is higher than a threshold).

⑪ Put the DL formulae e_s and e_t together to form a correspondence (*e.g.*, $\langle$ o_1*:paperWrittenBy* , *dom(*o_2*:Paper)* $\sqcap$ o_2*: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.

3.3 Translating SPARQL CQAs into DL Formulae

In Step ①, in order to translate a SPARQL query into a DL formula, the query is translated into a FOL formula and then transformed it into a DL formula. Here, a SPARQL SELECT query 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 CQAs, the SELECT clause contains one variable. In binary CQAs, the SELECT clause contains two variables. For instance, the SPARQL query of Fig. 3, *?x* becomes the quantified variable of our formula: $\forall x$. Then, the basic graph pattern is parsed in order 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, the triple is find ⟨ *?x* , o_2:hasDecision , *?y* ⟩. The FOL formula becomes $\forall x, \exists y, o_2$*:hasDecision(x,y)*. It is then recursively keeping 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_2*:hasDecision(x,y)* $\wedge$ o_2*:Acceptance(y)*. At the end of the process, the basic graph pattern is transformed into a DL formula (also translated into an EDOAL[3]): $\forall x, \exists\ y$, o_2*:hasDecision(x,y)* $\wedge$ o_2*:Acceptance(y)* becomes in DL: $\exists$ o_2*:hasDecision.*o_2*:Acceptance*. The FOL to DL equivalence is done as in [3].

3.4 Instance Matching

In Step $\textcircled{4}$, 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. With binary CQAs, 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 identical value in the path finding step, detailed next.

3.5 Retrieving and Pruning Subgraphs

The whole approach relies on subgraphs, which are sets of triples from a knowledge base. In Step $\textcircled{5}$, these subgraphs are found and then pruned and transformed into DL formulae in Step $\textcircled{8}$. The type of subgraphs is inspired from [33], 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_2*:paper1* is the subject of the triple o_2*:paper1* o_2*:hasDecision* o_2*:decision1* and o_2*:decision1* has types (classes) o_2*:Acceptance* and o_2*:Decision*. A subgraph of o_2*:paper1* is therefore composed of the following triples:

1. ⟨ o_2*:paper1* , o_2*:hasDecision* , o_2*:decision1* ⟩
2. ⟨ o_2*:decision1* , *rdf:type* , o_2*:Acceptance* ⟩
3. ⟨ o_2*:decision1* , *rdf:type* , o_2*: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. Considering the *accepted paper* CQA, the most similar type of the triple of the object is o_2*:Acceptance*. Therefore, triple 3 is pruned. The object of triple 1 is o_2*:decision1* and the most similar object type to the CQA

[3] http://alignapi.gforge.inria.fr/edoal.html.

is o_2*:Acceptance*. o_2*:Acceptance* is more similar to the CQA than o_2*:decision1*. o_2*:decision1* becomes a variable and triple 2 stays in the subgraph.

In order to translate, a subgraph into a DL formula, firstly this subgraph is translated into a SPARQL query:

– The answer is transformed into a variable and put in the SELECT clause. In this example, o_2*: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 the example, o_2*: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 Fig. 3.

Finally, the SPARQL query is transformed into a DL formula by using the same process as that described in Sect. 3.3: $\exists o_2$*:hasDecision.*o_2*: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_2*:paper1*,o_2*:person1*) in the target knowledge (Fig. 2), a subgraph containing the following triples is found: ⟨ o_2*:person1* , o_2*:writes* , o_2*:paper1* ⟩, ⟨ o_2*:paper1* , *rdf:type* , o_2*:Paper* ⟩, ⟨ o_2*:paper1* , *rdf:type* , o_2*:Document* ⟩, ⟨ o_2*:person1* , *rdf:type* , o_2*:Person* ⟩.

The most similar type of o_2*:person1* is o_2*:Person*, which is below the similarity threshold. The triple 4 is then removed from the subgraph. The most similar type of o_2*:paper1* is o_2*:Paper*. The triple 3 is therefore removed from the subgraph. o_2*: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 the following DL formula: *dom(*o_2*:Paper)*[4] $\sqcap$ o_2*:writes*$^{-}$.

3.6 DL Formula Aggregation

In Step (8), 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

[4] *dom* is introduced here for denoting the domain of a property.

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_2:hasDecision.*{*o_2:accept*}.

In a first step, a first aggregated formula is created, called the **extension** formula. It consists in merging the instances or literal values of the formulae with the same predicate into one set of values. Considering that through various answers to a CQA (*e.g.*, *o_2:paper1*, *o_2:paper2*, *etc.*), the following formulae is extracted:

$\exists$*o_2:hasDecision.*{*o_2:accept*},$\exists$*o_2:hasDecision.*{*o_2:strongAccept*}, $\exists$*o_2:hasDecision.*{*o_2:weakAccept*}.

The extension formula of these formulae is:

$\exists$*o_2:hasDecision.*{*o_2:accept,o_2:strongAccept,o_2: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 $\top$. It allows for fully abstracting the formula. The intension formula of the example formulae is: $\exists$ *o_2: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. Applied to the examples of Table 1, *o_2:accept*, *o_2:strongAccept* and *o_2:weakAccept* are more similar to the CQA than *o_2:hasDecision*. The extension form is chosen.

3.7 Calculating Counter-Examples

In Step (9), the DL formula similarity score is refined by looking for counterexamples (details on the similarity score are given in Sect. 3.8). A **counterexample** 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, assuming that the target formula e_t is *o_2:Paper* for the "accepted paper" CQA. From the target ontology, the answers *o_2:paper1, o_2:paper2, o_2:paper3* and *o_2:paper4* are retrieved from e_t and matched to the source instances respectively *o_1:paper1, o_1:paper2, o_1:paper3* and *o_1:paper4*. However, only *o_1:paper1* and *o_1:paper2* are accepted papers (and are described by the CQA) in the source ontology, then *o_1:paper3* and *o_1:paper4* are counterexamples.

The percentage of counterexamples 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 counterexamples 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 counterexamples is therefore:

$$percCounterExamples = \frac{|ans_s^{e_t} \sqcap \neg(ans_s^{cqa})|}{|ans_s^{e_t}|} \quad (1)$$

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

3.8 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 counterexamples *percCounterExamples* which is computed in Step ⑨ and detailed Sect. 3.7.

The similarity score is as following:

$$sim = (labelSim + strucSim) \times (1 - percCounterExamples) \tag{2}$$

In the example above, the similarity of $\exists o_2$*:hasDecision.*o_2*:Acceptance* with the unary CQA *"accepted paper"* is calculated as follows:

- $labelSim = 0.8 + 0.0$ because
  - *sim(labels(CQA), labels(*o_2*:hasDecision))* $= 0.0$
  - *sim(labels(CQA), labels(*o_2*:Acceptance))* $= 0.8$
- $strucSim = 0.0$ because it is a unary CQA
- $percCounterExamples = 0.0$

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

3.9 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.

4 Experiments

4.1 Dataset and Metrics

Two evaluation settings have been considered here. First, an automatic evaluation was performed on the populated version of the OAEI Conference benchmark [28]. 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 [23], AgroVoc [4], DBpedia [2] and TaxRef-LD [13]. 6 CQAs from AgronomicTaxon have been manually generated. These two datasets are the populated ones used in the first OAEI complex track [25].

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

Initial formulae	Extension	Intension
$\exists o_2\text{:}hasDecision.\{o_2\text{:}accept\}$ $\exists o_2\text{:}hasDecision.\{o_2\text{:}strongAccept\}$ $\exists o_2\text{:}hasDecision.\{o_2\text{:}weakAccept\}$	$\exists o_2\text{:}\boldsymbol{hasDecision}.\{o_2\text{:}\boldsymbol{accept}, o_2\text{:}\boldsymbol{strongAccept}, o_2\text{:}\boldsymbol{weakAccept}\}$	$\exists o_2\text{:}hasDecision.\top$

The evaluation metrics used here are the ones adopted in the OAEI 2019 campaign[5]. These metrics are based on the comparison of instance sets. 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[6]. 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:

$$coverage(A_{eval}, cqa_{pairs}, kb_s, kb_t, f) = \underset{\langle cqa_s, cqa_t\rangle \in cqa_{pairs}}{\text{average}} f(I^{kb_t}_{cqa_t}, I^{kb_t}_{bestq_t}) \quad (3)$$

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}:

$$QP = \frac{|I_{eval} \cap I_{ref}|}{|I_{eval}|} \qquad QR = \frac{|I_{eval} \cap I_{ref}|}{|I_{ref}|} \quad (4)$$

$$queryFmeasure(I_{ref}, I_{eval}) = 2 \times \frac{QR \times QP}{QR + QP} \quad (5)$$

$$bestq_t = \underset{q_t \in rewrite(cqa_s, A_{eval}, kb_s)}{\text{argmax}} queryFmeasure(I^{kb_t}_{cqa_t}, I^{kb_t}_{q_t}) \quad (6)$$

[5] http://oaei.ontologymatching.org/2019/complex/index.html.
[6] The description of rewriting systems is out of the scope of this paper.

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

$$precision(A_{eval}, kb_s, kb_t, f) = \underset{\langle e_1, e_2\rangle \in A_{eval}}{\text{average}} f(I_{e_1}^{kb_s}, I_{e_2}^{kb_t}) \quad (7)$$

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 tool and evaluation have been executed on an Ubuntu 16.04 machine configured with 16 GB of RAM running under an i7-4790K118 CPU 4.00GHzx8 processors. The runtimes are given for a single run.

4.2 Results and Discussion

Impact of Parameters. The impact of the different matching parameters (Table 2) has been measured on the Conference dataset (for sake of space, only the results varying the CQAs and counterexamples are presented in Table 3). The Levenshtein edit distance has been used as similarity metric, with a path max length of 3 properties (empirically chosen) and formula filtering threshold confidence value of 0.6 or best formulae. The higher the Levenshtein threshold, the more formulae have been filtered out. When Levenshtein threshold increases, we observe: stagnation of runtime, decrease of number of correspondences, increase of precision, and decrease of coverage. The higher the number of support answers, more *accidental correspondences* appear, with satisfying results with 1 support answer. When the number of support answers increases: increase of runtime, increase of number of correspondences, decrease of precision, stagnation of coverage. With respect to impact of CQAs with respect to automatically generated queries, overall better results with CQAs both in terms of coverage and precision, as a higher number of correspondences are generated by using the queries (introducing noise). Finally, computing counterexamples increases precision (reducing the number of generated correspondences), keeping the coverage. However, the runtime increases considerably (from 2 h up to 46 h).

Comparison with Existing Approaches. The alignments generated by the proposed approach have been compared to the following ones (Table 4)[7]: (i) query rewriting (Rew), the query rewriting oriented alignment set from [26] - 10 pairs of ontologies; (ii) ontology merging (Mer), the ontology merging oriented alignment set from [26] - 10 pairs of ontologies; (iii) ra1, the reference simple alignment from the OAEI conference dataset - 10 pairs of ontologies; (iv) Ritze, the output alignment[8] from [21] - complex correspondences found on 4 pairs of ontologies; and (v) AMLC, the output alignment[9] from [6] - output alignments between 10

[7] The choice of these alignments was based on the fact that they were the publicly available complex matchers at the time of running the experiments.

[8] https://code.google.com/archive/p/generatingcomplexalignments/downloads/.

[9] http://oaei.ontologymatching.org/2018/results/complex/conference/.

Table 2. Variation of parameters.

Variant	Nb ans	Levesthein threshold	Counterexamples	CQAs
Baseline	10	0.4		√
Levenshtein	10	0.0–1.0		√
Support answers	1–100	0.4		√
Query	10	0.4		
Counterexamples	10	0.4	√	√

Table 3. Results with the variation of parameters.

	Impact CQA		Impact counterexamples	
	CQAs	queries	no counterexamples	counterexamples
Runtime	2 h	2 h	2 h	46 h
Number of correspondences	1,699	3,098	1,699	1,320
Precision	0.63	0.47	0.63	0.74
Coverage	0.76	0.64	0.76	0.76

pairs of ontologies. Our approach is the only one able to generate more expressive (c:c) correspondences. Overall our approach obtains the best coverage scores comparing to the all other approaches. With respect to precision, with classical instance comparison function, we obtain the worst results (0.4 with counterexamples). However, precision results should be considered carefully. First, the relation of the correspondence is not considered in this score. Merg. and Rew. alignments contain many correspondences with a subsumption relation, 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. In order to compensate these errors, we use the non-disjoint scoring metrics in the precision evaluation. The score for a correspondence is 1 when the members are overlapping or both empty, and 0 otherwise. With the non-disjoint precision, we outperform Ritze and AMLC, with equivalent values for ra1, Mer. and Rew. Comparing the systems between them, the Rew. alignment outperforms the Merg. in terms of CQA Coverage. In fact, in the Merg. alignments, unions of properties were separated into individual subsumptions which were usable by the rewriting system. Ritze only outputs equivalent or disjoint correspondences. Its precision score is therefore the same for all metrics. AMLC achieves a better classical precision than our baseline approach but contains a high number of disjoint correspondences. Overall, as expected, the precision scores of the reference alignments are higher than those output by the matchers. Moreover, Ritze and AMLC both rely on correspondence patterns which limit the types of correspondences they can generate.

Table 4. Comparative results. CQA is calculated with query Fmeasure and precision with classical and not disjoint instance comparison set functions (noted classical-non disjoint in the results).

	baseline	counterexamples	Ritze	AMLC	ra1	Mer	Rew
Correspondence type	(c:c)	(c:c)	(s:c)	(s:c)	(s:s)	(s:c)	(s:c)
Runtime	2 h	46 h	1 h	0 h03			
Number of correspondences	1,699	1,320	360	441	348	628	842
Precision	0.3–1	0.4–1	0.8	0.4–0.6	0.6–1	0.4–1	0.4–1
Coverage	0.8	0.8	0.4	0.5	0.4	0.6	0.7

Table 5. Evaluation on the Taxon dataset.

	v1	v10
Runtime	28 h	32 h
Number of correspondences	134	328
Precision	0.3–1	0.3–1
Coverage	0.3–0.7	0.5-0.8

Manual Evaluation. A manual evaluation has been carried out on the Taxon dataset (Table 5). The baseline has been declined in two versions, using counterexamples: $v1$ with 1 support and $v10$ with 10 supports. In the lack of *owl:sameAs* links, then the links have been generated by a exact match on the instance labels. In this dataset, given the uneven population, more support instances entail a better coverage.

5 Related Work

Comparison to Other Matching Approaches. Earlier works in the ontology matching field have introduced the need for complex ontology alignments [12,30], and different approaches for generating them have been proposed in the literature afterwards (the reader can refer to [27] for a survey on them). These approaches involve different techniques such as relying on templates of correspondences (called patterns) and/or instance evidence. The approaches in [20,21] apply a set of matching conditions (label similarity, datatype compatibility, etc.) to detect correspondences that fit certain patterns. The approach of [22] uses the linguistic frames defined in FrameBase to find correspondences between object properties and the frames. KAOM [10] relies on *knowledge rules* which can be interpreted as probable axioms. The approaches in [16,17,31] use statistical information based on the linked instances to find correspondences fitting a given pattern. The approach in [14] uses genetic programming on instances to find correspondences with value transformation functions between two knowledge bases. The one in [18] uses a path-finding algorithm to find correspondences between two knowledge bases with common instances. The one in [9] iteratively constructs correspondences based on the information gain from matched instances between the two knowledge-bases. More recently, [6] relies on lexical similarity and structural conditions to detect correspondence patterns, close to [20]. Comparing our proposal to those described above, none of the complex approaches involve the user before or during the matching process. Like the ones in [9,16–18,31], we rely on the hypothesis that the knowledge bases contain common instances. Furthermore, as for the matching processing in general, in particular for the complex matching approaches in [20,21], we rely on the hypothesis that the ontologies in the knowledge base have a relevant lexical layer. Differently from most of them, the proposed does not rely on correspondence patterns. Finally, competency questions have not been adapted nor used for ontology 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 [29].

Generalisation Process. Ontology matching approaches relying on the $\mathcal{A}box$ of ontologies infer general statements from the instances, *i.e.*, they perform a generalisation[10]. This is the principle of *machine learning* in general and methods such as *Formal Concept Analysis* [7] or *association rule mining* [1]. These generalisation processes however require a considerable amount of data (or instances). Approaches such as the ones from [9,16,17,31] 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 [32] 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.

6 Conclusions and Future Work

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

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

generalisation over few instances and a limitation as it requires users to be able to express her or his needs as SPARQL queries in terms of the source vocabulary (this is however the only manual effort required for the user). The approach 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; taking into account specialization/generalization relations; 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. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD Conference, pp. 207–216 (1993)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Borgida, A.: On the relative expressiveness of description logics and predicate logics. Artif. Intell. **82**(1–2), 353–367 (1996)
4. Caracciolo, C., et al.: Thesaurus maintenance, alignment and publication as linked data: the AGROVOC use case. Int. J. Metadata Semant. Ontol. **7**(1), 65 (2012)
5. Euzenat, J., Shvaiko, P.: Conclusions. Ontology Matching, pp. 399–405. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38721-0_13
6. Faria, D.: Results of AML in OAEI. In: OM Workshop, pp. 125–131 (2018)
7. Ganter, B., Stumme, G., Wille, R. (eds.): Formal Concept Analysis. LNCS (LNAI), vol. 3626. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31881-1
8. Grüninger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing (1995)
9. Hu, W., Chen, J., Zhang, H., Qu, Y.: Learning complex mappings between ontologies. In: Pan, J.Z., et al. (eds.) JIST 2011. LNCS, vol. 7185, pp. 350–357. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29923-0_24
10. Jiang, S., Lowd, D., Kafle, S., Dou, D.: Ontology matching with knowledge rules. In: Hameurlain, A., Küng, J., Wagner, R., Chen, Q. (eds.) Transactions on Large-Scale Data- and Knowledge-Centered Systems XXVIII. LNCS, vol. 9940, pp. 75–95. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53455-7_4
11. Jouhet, V., Mougin, F., Bréchat, B., Thiessard, F.: Building a model for disease classification integration in oncology, an approach based on the national cancer institute thesaurus. J. Biomed. Semant. **8**(1), 6:1–6:12 (2017)
12. Maedche, A., Motik, B., Silva, N., Volz, R.: MAFRA — A MApping FRAmework for distributed ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 235–250. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45810-7_23

13. Michel, F., Gargominy, O., Tercerie, S., Faron-Zucker, C.: A model to represent nomenclatural and taxonomic information as linked data. Application to the French Taxonomic Register, TAXREF. In: 2nd International Workshop on Semantics for Biodiversity (2017)
14. Pereira Nunes, B., Mera, A., Casanova, M.A., Fetahu, B., P. Paes Leme, L.A., Dietze, S.: Complex matching of RDF datatype properties. In: Decker, H., Lhotská, L., Link, S., Basl, J., Tjoa, A.M. (eds.) DEXA 2013. LNCS, vol. 8055, pp. 195–208. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40285-2_18
15. Nurmikko-Fuller, T., et al.: Building complex research collections in digital libraries: a survey of ontology implications. In: Proceedings of the 15th ACM/IEEE-CE Joint Conference on Digital Libraries, pp. 169–172. ACM (2015)
16. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and building ontologies of linked data. In: Patel-Schneider, P.F., et al. (eds.) ISWC 2010. LNCS, vol. 6496, pp. 598–614. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17746-0_38
17. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Discovering concept coverings in ontologies of linked data sources. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 427–443. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35176-1_27
18. Meersman, R., Tari, Z. (eds.): OTM 2007. LNCS, vol. 4803. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76848-7
19. Ren, Y., Parvizi, A., Mellish, C., Pan, J.Z., van Deemter, K., Stevens, R.: Towards competency question-driven ontology authoring. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 752–767. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07443-6_50
20. Ritze, D., Meilicke, C., Šváb Zamazal, O., Stuckenschmidt, H.: A pattern-based ontology matching approach for detecting complex correspondences. In: OM Workshop (2009)
21. Ritze, D., Völker, J., Meilicke, C., Šváb Zamazal, O.: Linguistic analysis for complex ontology matching. In: OM Workshop (2010)
22. Rouces, J., de Melo, G., Hose, K.: Complex schema mapping and linking data: beyond binary predicates. In: Workshop on Linked Data on the Web (2016)
23. Roussey, C., Chanet, J., Cellier, V., Amarger, F.: Agronomic taxon. In: Workshop on Open Data, pp. 5:1–5:4 (2013)
24. Thiéblin, E., Amarger, F., Hernandez, N., Roussey, C., Trojahn Dos Santos, C.: Cross-querying LOD datasets using complex alignments: an application to agronomic taxa. In: Garoufallou, E., Virkus, S., Siatri, R., Koutsomiha, D. (eds.) MTSR 2017. CCIS, vol. 755, pp. 25–37. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70863-8_3
25. Thiéblin, E., Cheatham, M., dos Santos, C.T., Zamazal, O., Zhou, L.: The first version of the OAEI complex alignment benchmark. In: ISWC Poster Track (2018)
26. Thiéblin, É., Haemmerlé, O., Hernandez, N., Trojahn, C.: Task-oriented complex ontology alignment: two alignment evaluation sets. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 655–670. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_42
27. Thiéblin, E., Haemmerlé, O., Hernandez, N., Trojahn, C.: Survey on complex ontology matching. Semant. Web J. (2019)
28. Thiéblin, É., Trojahn, C.: Conference v3.0 : a populated version of the conference dataset. In: ISWC Poster Track (2019)

29. Unger, C., et al.: Question answering over linked data (QALD-4). In: Working Notes for CLEF, pp. 1172–1180 (2014)
30. Visser, P.R., Jones, D.M., Bench-Capon, T.J., Shave, M.: An analysis of ontology mismatches: heterogeneity versus interoperability. In: AAAI, pp. 164–172 (1997)
31. Walshe, B., Brennan, R., O'Sullivan, D.: Bayes-ReCCE: a Bayesian model for detecting restriction class correspondences in linked open data knowledge bases. Int. J. Semant. Web Inf. Syst. **12**(2), 25–52 (2016)
32. Wu, B., Knoblock, C.A.: An iterative approach to synthesize data transformation programs. In: IJCAI, pp. 1726–1732 (2015)
33. Zheng, W., Zou, L., Peng, W., Yan, X., Song, S., Zhao, D.: Semantic SPARQL similarity search over RDF knowledge graphs. VLDB Endowment **9**(11), 840–851 (2016)