CANARD complex matching system: results of the 2018 OAEI evaluation campaign

Elodie Thiéblin, Ollivier Haemmerlé, Cassia Trojahn

IRIT & Université de Toulouse 2 Jean Jaurès, Toulouse, France
{firstname.lastname}@irit.fr

Abstract. This paper presents the results obtained by the CANARD system in the OAEI 2018 campaign. CANARD can produce complex alignments. This is the first participation of CANARD in the campaign. Even though the system has been able to generate alignments for one only complex dataset (Taxon), the results are promising.

1 Presentation of the system

1.1 State, purpose, general statement

The CANARD (Complex Alignment Need and A-box based Relation Discovery) system discovers complex correspondences between populated ontologies based on Competency Questions for Alignment (CQAs). Competency Questions for Alignment (CQAs) represent the knowledge needs of a user and define the scope of the alignment [3]. They are competency questions that need to be satisfied over two or more ontologies. Our approach 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 graph-pattern from the source query to the lexically similar surroundings of the target instances.

1.2 Specific techniques used

The CQAs that are taken as input by CANARD are limited to class expressions (interpreted as a set of instances). The approach is developed in 11 steps, as depicted in Figure 1:

① Extract source DL formula e_s from SPARQL CQA.
② Extract lexical information from the CQA, L_s set labels of atoms from the DL formula.
③ Extract source instances $inst_s$.
④ Find equivalent or similar (same label) target instances $inst_t$ to the source instances $inst_s$.

(5) Retrieve description of target instances: set of triples and object/subject type.

(6) For each triple, retrieve L_t labels of entities.

(7) Compare L_s and L_t using a string comparison metric (e.g., Levenshtein distance with a threshold).

(8) Keep the triples with the summed similarity of their labels above a threshold τ. Keep the object(/subject) type if its similarity is better than the one of the object(/subject).

(9) Express the triple into a DL formula e_t.

(10) Aggregate the formulae e_t into an explicit or implicit form: if two DL formulae have a common atom in their right member (target member), the atoms which differed are put together.

(11) Put e_s and e_t together in a correspondence ($e_s \equiv e_t$) and express this correspondence in EDOAL. The average string similarity between the aggregated formula and the CQA labels gives the confidence value of the correspondence.

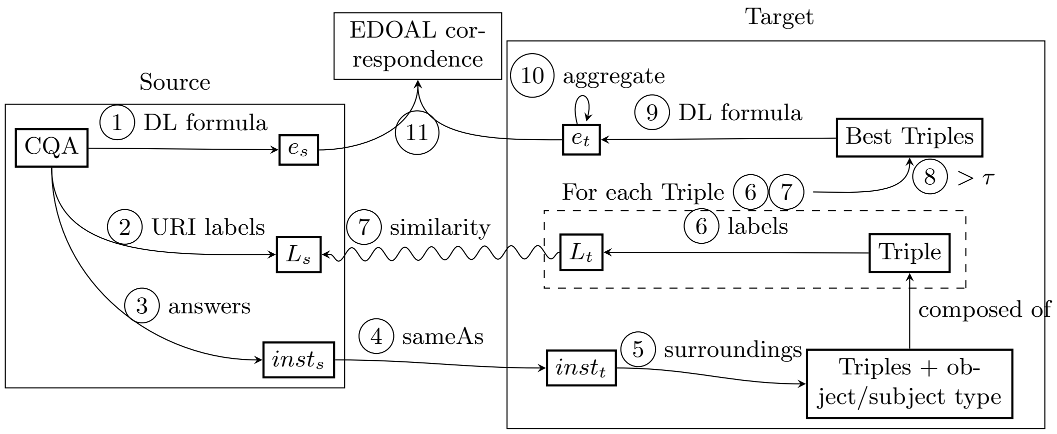

Fig. 1: Schema of the general approach.

The instance matching phase (step (4)) is based on existing *owl:sameAs*, *skos:closeMatch*, *skos:exactMatch* and exact label matching. The similarity between the sets of labels L_s and L_t of step (7) is the cartesian product of the string similarities between the labels of L_s and L_t (equation 1).

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

strSim is the string similarity between two labels l_s and l_t (equation 2). τ is the threshold for the similarity measure. In our experiments, we have empirically set up $\tau = 0.5$.

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

The confidence value given to the final correspondence (step $\textcircled{11}$) is the similarity of the triple it comes from or average similarity if it comes from more than one triple. The confidence value is reduced to 1 if it is initially calculated over 1.

1.3 Adaptations made for the evaluation

Automatic generation of CQAs The CQAs can not be given as input in the evaluation as none are available in the OAEI datasets. We developed a CQA generator that was integrated to the version of the system used in the evaluation. This generator produces two types of SPARQL queries: *Classes* and *Property-Value pairs*.

Classes For each *owl:Class* populated with at least one instance, a SPARQL query is created to retrieve all the instances of this class. If `<o1#class1>` is a populated class of the source ontology, the following query is created:
`SELECT DISTINCT ?x WHERE {?x a <o1#class1>.}`

Property-Value pairs Inspired by the approaches of [1,2,4], we create SPARQL queries of the form

- `SELECT DISTINCT ?x WHERE {?x <o1#property1> <o1#Value1>.}`
- `SELECT DISTINCT ?x WHERE {<o1#Value1> <o1#property1> ?x.}`
- `SELECT DISTINCT ?x WHERE {?x <o1#property1> "Value".}`

These property-value pairs are computed as follow: for each property (object or data property), the number of distinct object and subject values are retrieved. If the ratio of these two numbers is over a threshold (arbitrarily set to 30) and the smallest number is smaller than a threshold (arbitrarily set to 20), a query is created for each of the less than 20 values. For example, if the property `<o1#property1>` has 300 different subject values and 3 different object values (`"Value1"`, `"Value2"`, `"Value3"`), the ratio $|subject|/|object| = 300/3 > 30$ and $|object| = 3 < 20$. The 3 following queries are created as CQAs:

- `SELECT DISTINCT ?x WHERE {?x <o1#property1> "Value1".}`
- `SELECT DISTINCT ?x WHERE {?x <o1#property1> "Value2".}`
- `SELECT DISTINCT ?x WHERE {?x <o1#property1> "Value3".}`

The threshold on the smallest number ensures that the property-value pairs represent a category. The threshold on the ratio ensures that properties represent categories and not properties with few instanciations.

Implementation adaptations In the initial version of the system, Fuseki server endpoints are given as input. For the SEALS evaluation, we embedded a Fuseki server inside the matcher. The ontologies are downloaded from the SEALS repository, then uploaded in the embedded Fuseki server before the matching process can start. This downloading-uploading phase may take time, in particular when dealing with large files.

The CANARD system in the SEALS package is available at `http://doi.org/10.6084/m9.figshare.7159760.v1`. The generated alignments in EDOAL format are available at `http://oaei.ontologymatching.org/2018/results/complex/taxon/CANARD.html` (link to each pair of task). Note that, as described below, CANARD was able to generate results for the Taxon track.

2 Results

The CANARD system could only output correspondences for the Taxon dataset of the Complex track. Indeed, the other datasets of this track do not contain instances and least of all common instances.

Table 1 shows the run-time of CANARD on all pairs of ontologies in the Taxon track, as well as the characteristics of the output alignments. As the alignment process is directional, we do not obtain symmetrical results for a pair of ontologies. CANARD is able to generate different kinds of correspondences: (1:1), (1:n) and (m:n). The best precision was obtained for the pair agronomicTaxon-agrovoc with a precision of 0.57. CANARD did not output any correspondence for 4 oriented pairs (in grey in Table 1). These empty results can be due to the fail of the instance matching phase of our approach. We could observe that with TaxRef as the source knowledge base, no correspondence could be generated. The exception is the pair *taxref-agrovoc* where 8 correspondences were found but only involving *skos:exactMatch* or *skos:closeMatch* properties in the constructions. The incorrect correspondences of this pair have a low confidence (between 0.05 and 0.30).

Looking for the query rewriting task in Taxon, CANARD's alignment was used to rewrite the most queries (best *qwr*). As CANARD does not deal with binary CQAs, none of the 3 binary queries × 12 pairs of ontologies = 36 binary query cases could be dealt with. Out of the 2 unary queries × 12 pairs = 24 unary query cases, CANARD could deal with 6 unary cases needing a complex correspondence and 2 needing simple correspondences for a total of (8/24) 33% of unary query cases.

Overall, for the query cases needing complex correspondences, (0+6/28+16) 14% were covered by CANARD. For all the query cases, the CANARD system could provide an answer to (8/36+24) 13% of all cases.

3 General comments

The CANARD approach relies on common instances between the ontologies to be aligned. Hence, when such instances are not available, as for the Conference,

Test Case ID	Run Time (s)	output corres.	correct corres.	prec.	(1:1)	(1:n)	(m:n)
agronomicTaxon-agrovoc	37	7	4	0.57	0	7	0
agronomicTaxon-dbpedia	75	17	3	0.18	3	14	0
agronomicTaxon-taxref	87	9	3	0.33	1	8	0
agrovoc-agronomicTaxon	20	0		NaN	0	0	0
agrovoc-dbpedia	128	13	3	0.23	0	0	13
agrovoc-taxref	87	8	0	0	0	0	8
dbpedia-agronomicTaxon	556	0		NaN	0	0	0
dbpedia-agrovoc	236	37	0	0	0	20	17
dbpedia-taxref	333	43	14	0.33	0	17	26
taxref-agronomicTaxon	269			NaN	0	0	0
taxref-agrovoc	283	8	0	0	0	0	8
taxref-dbpedia	351	0		NaN	0	0	0
Global	2468	142	27	0.20	4	66	72

Table 1: Results of CANARD on the Taxon track

GeoLink and Hydrography datasets, the approach is not able to generated complex correspondences. Furthermore, CANARD is need-oriented and requires a set competency questions to guide the matching process. Here, these "questions" have been automatically generated based on a set of patterns.

The current version of the system is limited to finding complex correspondences involving classes and properties are not yet taken into account. We plan to extend the systems to take binary relations in the next version. Another point that we would like to improve is the semantics of the confidence of the correspondences.

With respect to the technical environment, as mentioned before, the initial version of the system receives as input the endpoints of the populated ontologies. Using SEALS, the large ontologies are stored into repositories. Our systems hence downloads them and stores them into an embedded Fuseki server. This configuration is not ideal as we have to deal with large knowledge bases. Furthermore, we struggled with the SEALS dependencies in order to correctly package our system into the SEALS format.

As we focus on user needs in order to avoid dealing with the whole alignment space, it could be interesting to having more need-oriented tasks with respect to the alignments coverage.

4 Conclusions

This paper presented the adapted version of the CANARD system and its preliminary results in the OAEI 2018 campaign. This year, we have been participated only in the Taxon track, in which ontologies are populated with common instances. CANARD was the only system to output complex correspondences on the Taxon track.

Acknowledgements

Cassia Trojahn has been partially supported by the French CIMI Labex projet IBLiD (Integration of Big and Linked Data for On-Line Analytics).

References

1. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and building ontologies of linked data. In: ISWC. pp. 598–614. Springer (2010)
2. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Discovering concept coverings in ontologies of linked data sources. In: ISWC. pp. 427–443. Springer (2012)
3. Thiéblin, E., Haemmerlé, O., Trojahn, C.: Complex matching based on competency questions for alignment: a first sketch. In: Ontology Matching Workshop. p. 5 (2018)
4. Walshe, B., Brennan, R., O'Sullivan, D.: Bayes-recce: A bayesian model for detecting restriction class correspondences in linked open data knowledge bases. International Journal on Semantic Web and Information Systems 12(2), 25–52 (2016)