ATBox Results for OAEI 2020

Sven Hertling[0000−0003−0333−5888] and Heiko Paulheim[0000−0003−4386−8195]

Data and Web Science Group, University of Mannheim, Germany
{sven,heiko}@informatik.uni-mannheim.de

Abstract. ATBox matcher is a scalable system for instance (A-Box) and schema (T-Box) matching. It uses two pipelines for generating candidates and use the schema matches to further improve the instance correspondences. Using a string blocking method, ATBox is able to align huge ontologies and can run on OAEI tracks like largebio and knowledge graph. The results look promising but further features for better finding correct instance matches can be developed.

Keywords: Ontology Matching · Knowledge Graph

1 Presentation of the system

Nearly all systems submitted to the Ontology alignment Evaluation Initiative (OAEI) are able to align ontologies which are also called TBox (terminological component) in description logics (DL). But due to the fact that there are more and more instance tracks like spimbench, link discovery, geolink cruise and knowledge graph the presented matcher is also able to match the ABox (assertion component). Thus the matcher is called ATBox which focuses on both the ABox and TBox.

Especially the knowledge graph track needs scalable systems which can deal with hundred of thousands of instances. Thus the basis of this matcher is a good blocking approach. The recall oriented result is afterwards fine tuned to increase the precision. Given this scalability, ATBox is also able to match large knowledge graphs like DBpedia [1] or YAGO [4].

1.1 State, purpose, general statement

The overall matching strategy of ATBox is shown in figure 1. The T-Box and A-Box have different processing pipelines but the correspondences are combined in the end to get the final alignment.

T-Box matching is applied for all classes and properties (`owl:ObjectProperty`, `owl:DatatypeProperty`, and `rdf:Property`). They are retrieved by the jena[1] methods *OntModel.listClasses()* and *OntModel.listAllOntProperties()*.

[1] `https://jena.apache.org`

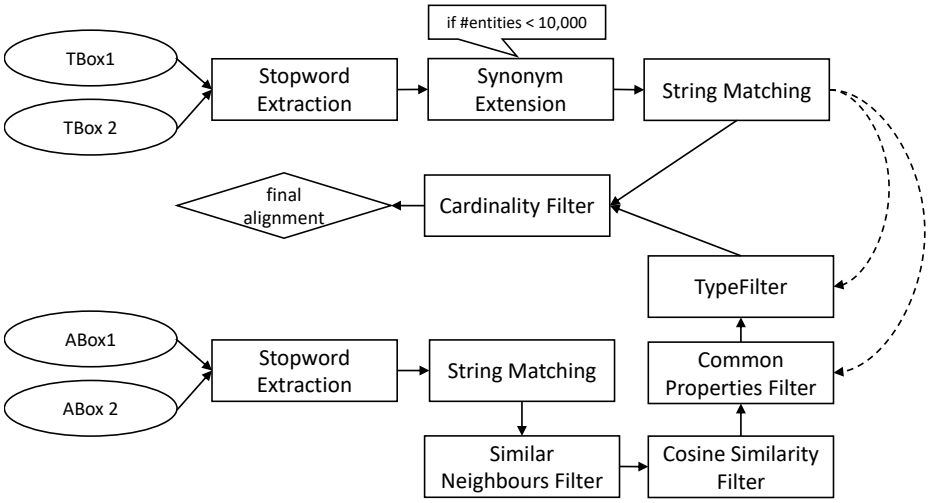

Fig. 1. Overview of the ATBox matcher strategy.

The T-Box matching (classes and properties) starts with the stopword extraction. In some cases the labels and/or fragments (which we define as the part after the last hashtag symbol # or slash /) contains tokens which appears very often like `class`, `infobox` etc as in `class Person`. If such tokens appears in more than 20% of all classes/properties (considered separately), then it is extracted as a corpus specific stop word. In case there are many such stop words, they are restricted to the five most occurring stop words.

The synonyms (which are used during string matching) are extracted from the English Wiktionary to cover many different domains. The extraction is made easy with DBnary[5], a dataset containing Wikitonary as RDF. First all resources of type `http://kaiko.getalp.org/dbnary#Page` within the English domain are chosen. Then we follow the `describes` relation and extract all resources connected with property `synonym`. Furthermore we follow the relation `sense` to find also synonyms of all the given senses. The lemmas are extracted directly from the URI.

The string matching contains multiple different steps which are shown in table 1. All processing applies to `rdfs:label` and in case it is missing to the URI fragment. If such text is exactly the same, the generated correspondence has a confidence of 1.0. During the normalization step, a word written in camel case[2] is separated with whitespace (e.g. `hasAge` to `has Age`) and afterwards lowercased. In case some UTF-8 characters are not normalized, we apply a normalization step for them (e.g. an accented character can be encoded in multiple different ways in UTF-8). All possible punctuations are furthermore removed and multiple whitespaces are combined into one. In case the normalized text matches, a confidence of 0.9 is assigned. In the `normalizeParentheses` step, all text in

[2] `https://en.wikipedia.org/wiki/Camel_case`

parentheses is removed. If the remaining normalized text (same as in `normalize` step) is equal, it assigns a confidence of 0.8. `DefaultStopwords` removes a given set of stopwords while keeping all other processing steps as before (confidence is 0.7). In the last processing, we also remove the corpus depended stopwords and allow a levenstein distance of 1 (but only in case the text is longer than 6 characters). In case it matches we generate a correspondence with confidence of 0.6. If the amount of concepts are less than 10,000 for source and target, then a synonym step is added with a confidence of 0.5. In this step, the extracted synonyms are used to replace (possibly multiple) tokens with all available synonyms.

All string processing steps are executed in order starting with the highest confidence. If a match is found the remaining steps are also executed to find possible other candidates. As an example, a correspondence like <Harry_Potter,harry_potter, =, 0.9> is already found, then the processing continues and can find another correspondence like <Harry_Potter,Harry_Potter(Book), =, 0.8>.

Table 1. Matching steps for T-Box.

Processsing	Confidence	Levenstein
equality	1.0	no
normalize	0.9	no
normalizeParentheses	0.8	no
defaultStopwords	0.7	no
corpusStopwords	0.6	yes(1,6)
synonyms	0.5	no

1.2 Specific techniques used

We used the following matching components of MELT[3]:

- ScalableStringProcessingMatcher
- StopwordExtraction
- SimilarNeighboursFilter
- CommonPropertiesFilter
- CosineSimilarityConfidenceMatcher
- SimilarTypeFilter
- NaiveDescendingExtractor

1.3 Adaptations made for the evaluation

ATBox matcher is also available as a SEALS package. Due to clashes of dependecies of SEALS and ATBox, we decided to use the external SEALS packaging mechanism of the MELT framework[3]. It generates an intermediate matcher which executes an external process which runs in its own java virtual machine (JVM). Thus different versions of dependencies is not a problem.

1.4 Link to the system and parameters file

ATBox matcher can be downloaded from
`https://www.dropbox.com/s/q57rzoec9zeumi2/ATBox.zip?dl=0`.

2 Results

This section discusses the results of ATBox for each track of OAEI 2020 where the matcher is able to produce results. The following tracks are included: anatomy, conference, largebio, phenotype, and knowledge graph track.

Specific matching strategies and interfaces for the interactive and complex track are currently not implemented.

All track results are discussed in the final version of this paper.

2.1 Anatomy

2.2 Conference

2.3 Largebio

2.4 Phenotype

2.5 Knowledge Graph

3 General comments

3.1 Comments on the results

3.2 Discussions on the way to improve the proposed system

We would like to increase the number of feature generators. One could for example use images associated with the concepts to further distinguish true positive from false positive correspondences.

Furthermore we could also improve the schema matches given all the instance correspondences as already shown in DOME matcher[2].

4 Conclusions

In this paper, we have analyzed the results of ATBox matcher in OAEI 2020. It shows that the system is very scalable and can generate class, property and instance alignments. Most of the used matching components are furthermore included in MELT to allow other system developers to reuse them.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Hertling, S., Paulheim, H.: Dome results for oaei 2019. OM@ ISWC **2536**, 123–130 (2019)
3. Hertling, S., Portisch, J., Paulheim, H.: Melt - matching evaluation toolkit. In: SEMANTICS. Karlsruhe. (2019)
4. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: A spatially and temporally enhanced knowledge base from wikipedia. Artificial Intelligence **194**, 28–61 (2013)
5. Sérasset, G.: Dbnary: Wiktionary as a lemon-based multilingual lexical resource in rdf. Semantic Web **6**(4), 355–361 (2015)