A Gold Standard Dataset for Large Knowledge Graphs Matching

Omaima Fallatah[1,2][0000−0002−5466−9119], Ziqi Zhang[1][0000−0002−8587−8618], and Frank Hopfgartner[1][0000−0003−0380−6088]

[1] Information School, The University of Sheffield, Sheffield, UK
{oafallatah1,ziqi.zhang,f.hopfgartner}@sheffield.ac.uk
[2] Department of Information Systems, Umm Al Qura University, Saudi Arabia
oafallatah@uqu.edu.sa

Abstract. In the last decade, a remarkable number of Knowledge Graphs (KGs) were developed, such as DBpedia, NELL and Google knowledge graph. These KGs are the core of many web-based applications such as query answering and semantic web navigation. The majority of these KGs are semi-automatically constructed, which has resulted in a significant degree of heterogeneity. KGs are highly complementary; thus, mapping them can benefit intelligent applications that require integrating different KGs such as recommendation systems and search engines. Although the problem of ontology matching has been investigated and a significant number of systems have been developed, the challenges of mapping large-scale KGs remain significant. In 2018, OAEI has introduced a specific track for KG matching systems. Nonetheless, a major limitation of the current benchmark is their lack of representation of real-world KGs. In this work we introduce a gold standard dataset for matching the schema of large, automatically constructed, less-well structured KGs based on DBpedia and NELL. We evaluate OAEI's various participating systems on this dataset, and show that matching large-scale and domain independent KGs is a more challenging task. We believe that the dataset which we make public in this work makes the largest domain-independent gold standard dataset for matching KG classes.

Keywords: Knowledge Graphs · Schema Matching · Evaluation Dataset.

1 Introduction

In the last decade, different KGs have been created as a result of years of information extraction practices and crowdsourcing. DBpedia [3], YAGO [20], and NELL [4] are examples of large domain-independent KGs. Such KGs cover multiple domains of knowledge such as medical, music, and publications. KGs play a significant role in many applications such as reasoning, search engines and e-commerce, while also being part of the linked open data domain [17].

Moreover, due to their automatically-constructed and independently-designed nature, such KGs contain overlapping and complementary facts. For instance, `Bone` and `Artery` are classified under `BodyPart` in NELL while being classified as `AnatomicalStructure` in DBpedia.

This problem of semantic heterogeneity has been thoroughly studied in the Semantic Web community, with many ontology matching systems being developed and surveyed [2]. Matching systems are annually evaluated through the Ontology Alignment Evaluation Initiative (OAEI)[3]. While a new track for KG matching has been introduced to OAEI's annual campaign since 2018, the challenges of aligning large-scale KGs remain significant [12]. Currently, existing gold standards are not well representative of real-world KGs. Such KGs are known for sharing complementary facts about real-world entities such as people and places, while current datasets are predominantly domain-dependent [1]. Further, the size of the existing gold standard does not accurately represent the complexity of matching large-scale KGs that imply a significantly larger search space due to orders of magnitude larger number of classes.

This work proposes a gold standard dataset for matching the classes of large, automatically constructed, inadequately structured, and domain-independent KGs. The introduced benchmark is based on DBpedia and NELL. Although both KGs are widely used in semantic web researches and can be considered highly influential, they are yet to be consolidated, even though the majority of LOD cross-domain datasets, including KGs, are interlinked to DBpedia[4] which serves as a central link to many LOD datasets. According to [19], NELL is considered as the most complementary KG to other larger KGs such as DBpedia with an average of 10% gain of instances, while merging other large KGs can only lead to a 5% gain. Therefore, we believe they are the best candidates for a gold standard dataset for aligning large cross-domain KGs. We conduct an experiment to evaluate the performance of OAEI's different participating systems on this dataset, and show that mapping the classes of open KG is a much more challenging task than the existing OAEI KG matching benchmark.

The rest of this paper is structured as follows. We start by reviewing the problem, and the current gold standard datasets for matching KGs in Section 2. Then, we describe the process of building the proposed dataset in Section 3. In Section 4, we present the results of evaluating current matching systems on the proposed gold standard . We close with a discussion and a conclusion in Sections 5 and 6 respectively.

2 Related Work

Ontology matching has been a well-studied problem which centers on discovering corresponding entities across two distinct ontologies [8]. In the last decade, many matching systems were developed and evaluated annually at the OAEI event.

[3] `http://oaei.ontologymatching.org/`
[4] `https://lod-cloud.net`

The initiative provides over ten benchmark datasets in different tracks for various matching systems to be evaluated. Examples of main tracks are Anatomy, Conference, Complex Matching, Large Biomedical, and Interactive Matching.

KGs are often compared to ontologies since both are used for data representation purposes. Different from former ontology, open KGs are large-scale, multi-domain and less well-formatted compared to ontologies [22]. Similar to ontologies, KGs entities also suffer from semantic heterogeneity where the same real-world entities are described using different terminologies.

While there have been many well established matching systems for OAEI's different matching tracks, the need for KG matchers remains an open area of research [12]. Research in this domain has only been established since 2018, when OAEI introduced a new track dedicated to KG matching[5]. Since then, ontology matching tools have been evaluated on the provided benchmark, and multiple KG matchers have participated in the latest version in 2019 [1]. Although matching KGs has been a growing area of research recently, there is still a lack of gold standard datasets that represent diverse KGs.

The benchmark dataset currently used to evaluate systems in OAEI's KG track is constructed from DBkWik [11], which is a KG created from wikis shared on a wiki hosting platform. The individual KGs from the DBkWik project were used to create the ground truth datasets for this track. The track consists of five test cases where each test case is aimed at matching both the schema, including classes and properties, and the instance level of two KGs. The schema level correspondences were built by ontology experts while the instance level correspondences were automatically extracted [1]. To the best of our knowledge this gold standard is the only benchmark available to evaluate KG matching systems. However, the number of mapped classes is considerably small, i.e., less than 50 [12]. Therefore, this dataset does not represent the complexity of matching real-world KGs where hundreds of classes can be matched.

In terms of large domain-independent KGs, there are many published according to the Semantic Web standards. Some of them are based on Wikipedia, such as YAGO and DBpedia. Originally, DBpedia is a knowledge base constructed from structured data embedded on Wikipedia [3]. DBpedia also involves crowdsourcing communities to maintain the quality of the mapping between Wikipedia's articles and the structured knowledge in their KG. In contrast, NELL is a fully automated learned KG under the Never-Ending Language Learner project, which uses machine learning to read and extract knowledge from free text on the web. It started with a seed KG that continuously evolves by learning patterns from text to extract facts that are used to constantly grow and update the seed KG [4]. Since its launch in 2010, NELL has grown to a KG containing 50 million facts[6]. While the schema of the majority of Wikipedia based KGs cover multiple types of properties, NELL graph schema is very basic. It does not contain as many relations between instances [19]. Another KG

[5] http://oaei.ontologymatching.org/2019/knowledgegraph/index.html
[6] http://rtw.ml.cmu.edu/rtw/

of a taxonomy structure is WebIsALOD [10]. However, the latter only covers hypernymy relations and does not distinguish classes from instances.

3 Approach

3.1 Overview

As mentioned earlier in Section 1, we use NELL and DBpedia as both KGs share a significant amount of complementary facts. In this work, we deploy DBpedia 2016-10 version [7] using SPARQL query endpoint to return schema information. As for NELL, since a query end point is not available we obtained schema information by parsing a NELL dump file[8] which contains every fact learned by the project so far. As a result, DBpedia has over 750 classes while NELL has around 290 classes. Let $\boldsymbol{P}$ be the set of pair-wise classes across the two KGs, then the number of all possible pairs is 218,660. Since our goal is to use human annotators to identify all mappable pairs of classes, a greedy approach will lead to a dataset that is expensive to annotate and likely to be overwhelmed with negative pairs. Instead, we first apply a **Blocking Strategy** to manually generate a set of candidate pairs $\boldsymbol{C}$ which is a subset of $\boldsymbol{P}$ with significantly reduced number of negative class pairs. Next, we perform a **Candidate Filtering Strategy** by applying two similarity measures to each pair in $\boldsymbol{C}$ to further reduce the search space for human annotators. Another screening was done after the filtering stage to ensure that none of the discarded classes had a potential match in the corresponding graph. Finally, for **Dataset Annotation**, we asked human annotators to determine alignment of the resulting class pairs to construct the gold standard dataset.

3.2 Generating Candidate Pairs

Given the two KGs, we set one as source and one as target. Details about our source and target choices will be explained later. Moreover, given $\boldsymbol{P}$, the set of all possible class pairs from the two KGs, we apply a **Blocking Strategy** which requires manually screening the two KG class structures. The result of this process is a set $\boldsymbol{C}$ which should eliminate as many true negatives as possible while maintaining as many as (if not all) true positives. To illustrate the complexity of the task, the classes named `School` in both KGs refer to different types of schools. For instance, it is categorized as a subclass of `EducationalInstitutions` in DBpedia while being a super class of `HighSchool` and `University` classes in NELL. Given this structural inconsistency issue, a preliminary study aimed at aligning the higher level of concepts across the two KGs was necessary.

[7] `https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10`,visited on 14-2-2020

[8] `http://rtw.ml.cmu.edu/rtw/resources`, iteration number 1115, visited on 22-2-2020

We manually created two subsets $\boldsymbol{A}$ and $\boldsymbol{B}$, where the first is a set of NELL classes that have a possible corresponding class in DBpedia, and the second is a set of DBpedia classes that have a possible corresponding class in NELL. These two sets were created in two phases. First, we started by comparing the common root classes across the two KGs, e.g., `Person` or `Place`. Then, all of their non-root (descendant) classes were added to $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively. For instance, all the descendant classes of $Person_{nell}$ and $Person_{dbp}$ were added to each of $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively. Second, we examined other possible classes in which their root classes do not share an overlap of words, i.e., they were not selected in the first step. A valid example are the two classes $AcdemicSubject_{dbp}$ and its possible equivalent class $AcademicField_{nell}$. While the former is a subclass of `TopicalConcept`, the second is a subclass of `everypromotedthing`. The latter is the root class of the KG taxonomic tree, i.e., the equivalent of `OWL:Thing` in DBpedia. Therefore, our second screening phase was aimed at all descendant classes in both KGs whose name values share overlapping words while their super classes do not share overlapping words.

As a result of this blocking strategy, a total of 18,492 candidate pairs were generated in $\boldsymbol{C}$ as the product of $\boldsymbol{A}$ and $\boldsymbol{B}$. We believe this blocking strategy will not incorrectly discard any true positives because we have examined all discarded classes to identify any possible match in the opposite KG. As shown in Table 1, the number of distinct classes from DBpedia and NELL is 138 and 134 respectively. Nonetheless, this number of candidate pairs remains expensive for an annotation task. Therefore, we proceed by the Candidate Filtering Strategy to further reduce the numbers of pairs that need to be annotated by human annotators while maintaining pair completeness.

Table 1. Number of classes and instances in the created dataset

Dataset	#Classes	#Instances	Avg #instance per class
DBpedia	138	631,461	4,576
NELL	134	1,184,377	8,905

3.3 Candidate Filtering

In this section, we introduce the similarity measures applied to the candidate pairs resulting from the prior phase. We apply a string-based and an instance-based similarity measure combined with a low threshold to maximise the chance to retain all true positives. We apply a **String-based Similarity** measure to class names only since NELL does not offer other metadata descriptions of classes. However, using only a string-based matcher can not guarantee a high recall as both KGs use different names to describe the same classes. We then apply an **Instance-based Similarity** measure to capture any possible true positive pairs where string similarity could have failed to recognize them. To the best of our knowledge, a matching approach that can handle a substantial number of instances, such as in the case of KGs , is yet to be established. Therefore,

Section 3.3 discusses the implementation of our preliminary instance-based approach. We believe that combining both measures can ensure high (if not full) recall of true positive pairs. It is also worth mentioning that due to the structural irregularity in both KGs, structural-based similarity measures were excluded.

String-based Similarity Measure We apply the *Levenshtein* [16] edit distance approach. This method has shown improvements over alternative string-based measures, particularly for matching classes [5]. Here the similarity between class names in each candidate pair is measured. This value is then normalized by dividing the value by the length of the longer string, i.e., class name, to produces a value between [0.0, 1.0]. For this task we only retain a pair if the similarity score of the two class names exceeds 0.4. State-of-the-art matching systems that utilize an edit distance approach often apply a higher threshold, which can be up to 0.8, to eliminate the number of false positive alignments [2]. Nonetheless, in order to capture as many true positive pairs as possible, we use a threshold that is twice lower than the state-of-the-art methods.

Instance-based Similarity Measure This method casts the matching process based on the principle of free-text index and search, which scales to very large datasets. On a typical index/search scenario a collection of resources (i.e. web documents) is indexed in a vector space where documents are represented with weighted vectors of their text content. Weighting approaches, such as TF/IDF, are used to weight term occurrences in the documents. A query given to a search engine will also be converted into a vector representation and then matched against all the vectors stored in the index. The matching is done by similarity measures such as the cosine function where a ranked list of top K documents related to the query is retrieved. Similarly, we propose to treat both KGs as a collection of documents where each document corresponds to a class in a KG and each term corresponds to the name of an instance. To map similar classes, a query is built by sampling instance names from a source KG's class, and matching against the index of the target KG. The equivalent class is determined based on the search result, which is a ranked list of classes whose instance names overlap with those in the query. We exploit *Apache Solr*[9], a state-of-the-art free text index and search engine. The pseudocode for the entire similarity measure is illustrated in Algorithm1.

During the **Indexing** process, a separate index is created for the source and target KGs. Classes from each KG are represented in documents that contain the concatenation of the class's instance names. The documents' contents are indexed using the standard Solr indexing process, including tokenisation, stemming, lemmatization, lower casing, and term-weighing. For our particular task an index is needed for NELL and DBpedia to perform the matching task. Thus, we run the following query to obtain all instance names for each DBpedia class:

```
SELECT ?name
```

[9] `https://lucene.apache.org/solr/`

```
WHERE{ ?entity a <http://dbpedia.org/ontology/%ClassName>.
       ?entity rdfs:label ?name.
       Filter (lang(?name)="en")}
```

After each query, a new document representing a DBpedia class is added and indexed in the designated DBpedia index. Similarly, an index was created for NELL which contained indexed documents of instance names parsed from the NELL facts dump.

Algorithm 1 Instance-Based Similarity Measure

Require:
1: *source* ← a list of classes in *Source* KG
2: *target* ← a list of classes in *Target* KG
3: **for** Class a_n in *source* **do**
4: *count* = 1
5: *candidate* = [ ]
6: **while** count ≤ 30 **do**
7: *query* ← a concatenation of 20 instance names of class a_n
8: *results* ← search(*query*,*target*) in the *target* index
9: **for** b_n in *results* **do**
10: candidate.append(b_n)
11: **end for**
12: count++
13: **end while**
14: *candidate_pairs* ← Top three frequent classes in *candidate* paired with a_n
15: **end for**

To perform the matching process, NELL and DBpedia were treated as source and target respectively. Consequently, queries are generated by sampling instances names from NELL's classes. This process can be performed in the opposite direction; however, some of DBpedia's classes have missing instances. This implies that a query cannot be created from such empty classes. For example, classes such as `State`, `Zoo`, `Profession` are all leaf classes and supposed to be populated with individuals but the links between class's name and its instances are missing in the KG. A case in point is `California`[10] and `Florida`[11]: both are defined in the data with classes (i.e., rdf:type) other than `State`. This problem was encountered in 20 classes from the 138 classes selected from DBpedia. With DBpedia being the center of the LOD datasets in mind, many options can be explored in order to fulfill this gap. This includes using instances from *SKOS* concepts or another KG that already has an established mapping with DBpedia, such as WikiData or WebIsALOD. Nonetheless, we believe that performing a one-way search is sufficient for capturing all positive pairs for the annotation task.

In terms of the **Search** process, we aim to discover class pairs that share a significant number of overlapping instance names across two KGs. Our em-

[10] `http://dbpedia.org/page/California`
[11] `http://dbpedia.org/page/Florida`

pirical test on a smaller sample of the dataset showed that two key factors can directly impact the search (matching) result. The first one is the number of instance names to be used in the query string. Due to these KGs' instances being automatically extracted, and the large number of instances per class, using either a too-large or too-small number of instance names to create queries will result in no similar documents (classes) being retrieved or false positive pairs. The second factor impacting the search result is the number of searches (iterations) performed on each class to determine its equivalent class. Because of the restriction of the query length, concatenating the names of all class instances is not feasible. Moreover, by using a sample of instance names, different results can be retrieved depending on the sample. Our experiment has shown that we can obtain the maximum number of true positive pairs when concatenating 20 instances per query and performing 30 iterations per class.

To demonstrate, for a class a_n in NELL, a random 20 instances of that class are obtained and concatenated to form a query string. That query is then matched against all documents (classes) in the target index, i.e., DBpedia. Consequently, a list of classes whose instances overlap with those in the query are retrieved. For example, if the following results were retrieved when sampling instances from class $Airport_{nell}$ in the source KG:

Iteration 1 -> {$Airport_{dbp}$, $City_{dbp}$, $Port_{dbp}$}
Iteration 2 -> {$City_{dbp}$, $Port_{dbp}$, $Airport_{dbp}$}
Iteration 3 -> {$Airport_{dbp}$, $Port_{dbp}$}
Iteration n-1 -> {$Airport_{dbp}$, $City_{dbp}$}
Iteration n -> {$Airport_{dbp}$, $City_{dbp}$, $Street_{dbp}$}

By the end of the 30th iteration, we add three pairs of candidate alignments for class $Airport_{nell}$. Only the three most frequently retrieved classes among all iterations are added as positive pairs with a non-zero as similarity score. For the above example, the following pairs will be added: ($Airport_{nell}$,$Airport_{dbp}$), ($Airport_{nell}$,$City_{dbp}$), and ($Airport_{nell}$, $Port_{dbp}$). Notice that $Airport_{nell}$ is not matched to $Street_{dbp}$ as the latter only appeared once during the search process.

Combining Similarity Measures As our goal for this particular task is to discover potentially matching pairs to be annotated by human annotators, our aim is to ensure a high (if not full) recall, which was achieved by combining the two similarity measures. We applied the above mentioned similarity measures to the 18,492 class pairs obtained in the prior phase. Only pairs that obtained a similarity score higher than 0.4 by the String-based method or a non-zero value by the Instance-based method were considered for the annotation task. Following the above automated approach, we performed another manual screening to discover remaining equivalent classes from NELL and DBpedia that were not included in the potential pairs. By inspecting all pairs discarded by the filtering process we were able to identify and recover 8 pairs. A total of 596 pairs were created for the human annotation task.

3.4 Dataset Annotation

In order to create a gold-standard dataset of matching classes, we asked human annotators to determine the alignment for the previously discovered pairs, and then aggregate their interpretations by the majority votes, as human annotators can have different interpretation of correspondence. We have also performed a study of the inter-annotator agreement (IAA). The dataset was annotated by twenty research students and validated by two computer scientists. The participants were provided with guiding instructions to complete the task. Several labels were allowed to annotate pairs which are `a match`, `not a match`, `more general`, and `more specific`. The latter two options are often used in the ontology domain to label subsumption relation in ontologies. The reason we gave the annotators this option is that it can be possible in a few cases. For example, while DBpedia has two separate class for `State` and `Province`, NELL has one class named `StateOrProvince` which combines both.

Each participant annotated around 50 pairs on average. In order to observe (IAA), 400 random pairs are duplicated among 12 annotators such that each pair is annotated by 3 different annotators. The average IAA for this task was measured using Cohen's kappa based on a sample of the dataset and it was 0.83. The dataset was then validated by two experts. This was mainly to ensure that the subsumption relations were used properly. Therefore, a subsumption relation was only added to the dataset if there was an agreement by the experts. The gold standard mapping resulting from this annotation task is publicly available as two test cases[12]. The small test case includes a few instances per class, while the full test case contains the full A-box information for the included classes. The latter can be used to benchmark instance-based matching systems. The size of the gold standard is 129 equivalent class pairs with 24 non-trivial matches, i.e., not an exact matching string of class labels. Currently, the larger dataset in OAEI's KG track carries only 15 class matches, while the maximum number of non-trivial matches is 10. This makes the proposed dataset the largest domain-independent gold standard for matching KG classes. This gold standard is considered as a *partial gold standard* since some classes in both KGs have no equivalent class in the corresponding KG.

4 Evaluation

We evaluated the performance of the matching systems that participated in the KG track in OAEI 2019 event on the proposed gold standard. The Matching Evaluation Toolkit MELT [13] was used to perform this evaluation along with the SEALS client. The following systems were evaluated: POMAP++ [15], AML [9], FCAMap-KG [6], LogMap [14], LogMapLt, LogMapKG, LogMapBio, DOME [11], Wiktionary [18], and the string matcher used as a baseline for the KG track.

[12] `https://github.com/OmaimaFallatah/KG_GoldeStandard`

We evaluated the class alignments resulting from each matcher based on precision, recall, and f-measure. Results of the evaluation are shown in Table 4. Since the proposed gold standard is only a partial gold standard, and to avoid over-penalising systems that may discover reasonable matches that are not coded in our gold standard, we ignore any predicted matches if neither of the classes in that pair is present as a true positive pair with another class in our gold standard. As an example, for a class a_n we only consider the alignment (a_n, b_n) as a false positive, if the gold standard has a true positive pair containing either a_n or b_n but not both in the same pair.

Table 2. Performance of the KG track participants in OAEI on the proposed dataset compared to their performance on the OAEI KG track **starwars-swtor** benchmark

	Proposed Dataset			**OAEI KG benchmark**		
Matcher	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
POMAPP++	0.0	0.0	0.00	0.0	0.0	0.00
AML	1.00	0.61	0.75	1.00	0.87	0.93
FCAMap-KG	0.96	0.62	0.75	1.00	0.80	0.89
LogMap	0.98	0.79	0.88	1.00	0.80	0.89
LogMapKG	0.98	0.79	0.88	1.00	0.80	0.89
LogMapBio	0.98	0.79	0.88	1.00	0.80	0.89
LogMapLt	1.00	0.60	0.75	1.0	0.73	0.85
DOME	0.99	0.63	0.77	0.93	0.87	0.90
Wiktionary	0.99	0.79	0.88	1.00	0.87	0.93
KGbaselineLabel	1.0	0.61	0.76	1.00	0.80	0.89

As Table 4 shows, we have also evaluated the matchers on the **starwars-swtor** test case, which is the largest dataset in the track in terms of the size of class correspondences (which is 15). The best performing systems on the OAEI dataset in terms of recall are DOME, Wiktionary and AML; however, DOME and AML have obtained a lower recall (0.6) in our dataset, while Wiktionary is one of the best performing systems on our dataset. In contrast, the second to best performing matchers on the OAEI dataset, i.e., the LogMap family, obtained a recall of 0.79, which is the best recall on our gold standard. Nonetheless, 27 out of the 129 true positive pairs were not discovered by any matcher in the LogMap family. Among the evaluated matchers, LogMApKG and FCAMap-KG are the only systems that are particularly designed to match KGs. While the latter is the second best performing system in OAEI's 2019 KG track, particularly in matching classes, it has obtained a recall of 0.62 on our dataset. In terms of the precision on our dataset, the scores are fairly high since most systems were only able to discover trivial matches. However, the recall ranges between 0.6 and 0.79, which shows that the dataset contains class correspondences that are difficult to find. Hence, all systems need further improvements in order to map the classes of large and domain-independent KGs.

5 Discussion

From the results presented above, the following three patterns were observed. **First**, while current tools are able to produce high-quality results for well-formed ontologies, such techniques are not as well-performing when applied on KGs that lack textual descriptions. For instance, DOME is a matcher that trains a doc2vec model using all available metadata descriptions for ontologies. This can explain the matcher's low performance on our dataset as it requires a large amount of text. **Second**, many ontology matching systems utilizes structural knowledge available in well-structured ontologies such as disjoint axioms to refine their alignments [6]. Examples of systems that follow such an approach are AML and LogMap. However, as a result of the lack of schematic information in NELL, structural-based techniques can be difficult to apply in this case. **Third**, matching strategies used when two resources are from a specific domain setting are not applicable for domain-independent settings where classes contain information about real-world entities described with different terminologies. Therefore, in order to tackle the problem of KG matching, the need for specialized matching tools remains significant. Recently, many matching tools based on entity embedding are being proposed but only tested with domain-dependent datasets or in task-oriented settings, (e.g., [7,21]). Tailoring such methods for multi-domain KG matching and testing them on our gold standard can lead to deeper understanding and discovery in this domain.

6 Conclusion

In this paper, we developed the largest gold standard dataset for matching the classes of large KGs. Our gold standard is based on two highly influential KGs, and one of them is yet to be linked to the LOD. We evaluated several state-of-the-art matching tools on this dataset and showed that the task of matching large, domain-independent KGs remains very challenging. We argue that matching large, domain-independent and automatically constructed KGs has significant utility and therefore, future work should be devoted further into this area. We believe that our dataset and findings will foster research in this direction.

References

1. Algergawy, A., Faria, D., Ferrara, A., Fundulaki, I., Harrow, I., Hertling, S., Jiménez-Ruiz, E., Karam, N., Khiat, A., Lambrix, P., et al.: Results of the ontology alignment evaluation initiative 2019. In: CEUR Workshop Proceedings. pp. 46–85 (2019)
2. Anam, S., Kim, Y.S., Kang, B.H., Liu, Q.: Review of ontology matching approaches and challenges. International Journal of Computer Science and Network Solutions pp. 1–27 (2015)
3. Bizer, C., Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mende, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web pp. 1–5 (2012)

4. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Twenty-Fourth AAAI Conference on AI (2010)
5. Cheatham, M., Hitzler, P.: String similarity metrics for ontology alignment. In: International semantic web conference. pp. 294–309 (2013)
6. Chen, G., Zhang, S.: Identifying mappings among knowledge graphs by formal concept analysis. In: OM@ ISWC. pp. 25–35 (2019)
7. Dhouib, M.T., Zucker, C.F., Tettamanzi, A.G.: An ontology alignment approach combining word embedding and the radius measure. In: International Conference on Semantic Systems. pp. 191–197 (2019)
8. Euzenat, J., Shvaiko, P.: Ontology matching: state of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering (2013)
9. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The agreementmakerlight ontology matching system. In: OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". pp. 527–541. Springer (2013)
10. Hertling, S., Paulheim, H.: Webisalod: providing hypernymy relations extracted from the web as linked open data. In: International Semantic Web Conference. pp. 111–119 (2017)
11. Hertling, S., Paulheim, H.: DOME results for OAEI 2018. CEUR Workshop Proceedings pp. 144–151 (2018)
12. Hertling, S., Paulheim, H.: The knowledge graph track at oaei. In: European Semantic Web Conference. pp. 343–359 (2020)
13. Hertling, S., Portisch, J., Paulheim, H.: Melt-matching evaluation toolkit. In: International Conference on Semantic Systems. pp. 231–245 (2019)
14. Jiménez-Ruiz, E.: Logmap family participation in the OAEI 2019. In: CEUR Workshop Proceedings. pp. 160–163 (2019)
15. Laadhar, A., Ghozzi, F., Megdiche Bousarsar, I., Ravat, F., Teste, O., Gargouri, F.: Pomap: An effective pairwise ontology matching system. In: 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. pp. 161–168 (2017)
16. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady. pp. 707–710 (1966)
17. Paulheim, H.: Machine learning with and for semantic web knowledge graphs. In: Reasoning Web International Summer School. pp. 110–141. Springer (2018)
18. Portisch, J., Hladik, M., Paulheim, H.: Wiktionary matcher. In: CEUR Workshop Proceedings. pp. 181–188 (2019)
19. Ringler, D., Paulheim, H.: One knowledge graph to rule them all? In: Joint German/Austrian Conference on Artificial Intelligence. pp. 366–372 (2017)
20. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web. pp. 697–706 (2007)
21. Trisedya, B.D., Qi, J., Zhang, R.: Entity alignment between knowledge graphs using attribute embeddings. In: Proceedings of the AAAI Conference on AI. pp. 297–304 (2019)
22. Zhang, Z., Gentile, A.L., Blomqvist, E., Augenstein, I., Ciravegna, F.: An unsupervised data-driven method to discover equivalent relations in large Linked Datasets. Semantic Web pp. 197–223 (2017)