KGMatcher+ Results for OAEI 2022

Omaima Fallatah[1,2], Ziqi Zhang[1] and Frank Hopfgartner[3]

[1]Information School, The University of Sheffield, Regent Court, 211 Portobello, Sheffield S1 4DP, UK
[2]Information Systems, Umm Al Qura University, Mecca 24382, Saudi Arabia
[3]Universität Koblenz-Landau, Mainz 55118, Germany

Abstract

KGMatcher+ is a scalable and domain-independent matching system that matches the schema (classes) of large Knowledge Graphs by following a hybrid matching approach. *KGMatcher+* is composed of an instance-based matcher which only uses annotated instances of knowledge graph classes to generate candidate class alignments and a string-based matcher. This year is the second OAEI participation of *KGMatcher+*, formerly known as KGMatcher. More improvements have been added to the matcher, particularly in terms of handling imbalanced class distribution, and it is the best-performing system in the *common knowledge graphs* track this year.

Keywords

Knowledge Graphs, Instance-based Ontology Matching, Machine Learning, Schema Matching.

1. Presentation of the system

1.1. State, purpose, general statement

Combining different matching techniques is a common practice in ontology matching tools. Matching techniques are divided into three main categories [1]: (1) *Element level* techniques which discover similar entities by processing textual annotation of ontology entities, (2) *Structural level* techniques which study the relation between ontology entities to generate candidate pairs, and finally (3) *Extensional* or *Instance_based* techniques which utilize populated instances, i.e., (ABox) data to generate alignments at schema level (TBox).

The matcher proposed here is a hybrid approach that combines an element-level matcher with an instance-based one. The first matcher uses entity labels to generate candidate pairs, and the latter produces candidate class alignments by only using annotated instance names. While conventional ontologies mainly focus on modeling classes and properties, Knowledge Graphs (KGs), particularly those available on the web, are large-scale and describe numerous instances. Following a self-supervised and two-way classification approach, the presented matcher trains a classifier using instances annotated in each KG as training data. A KG classifier can then be used to classify any instance name into one of its own classes. The system is domain independent and is capable of coping with KGs with unbalanced populations (for details, see [2, 3]). This

OM2022: the International Workshop on Ontology Matching, October 23, 2022, Hangzhou, China

oafallatah1@sheffield.ac.uk (O. Fallatah); ziqi.zhang@sheffield.ac.uk (Z. Zhang); hopfgartner@uni-koblenz.de (F. Hopfgartner)
0000-0002-5466-9119 (O. Fallatah); 0000-0002-8587-8618 (Z. Zhang); 0000-0003-0380-6088 (F. Hopfgartner)

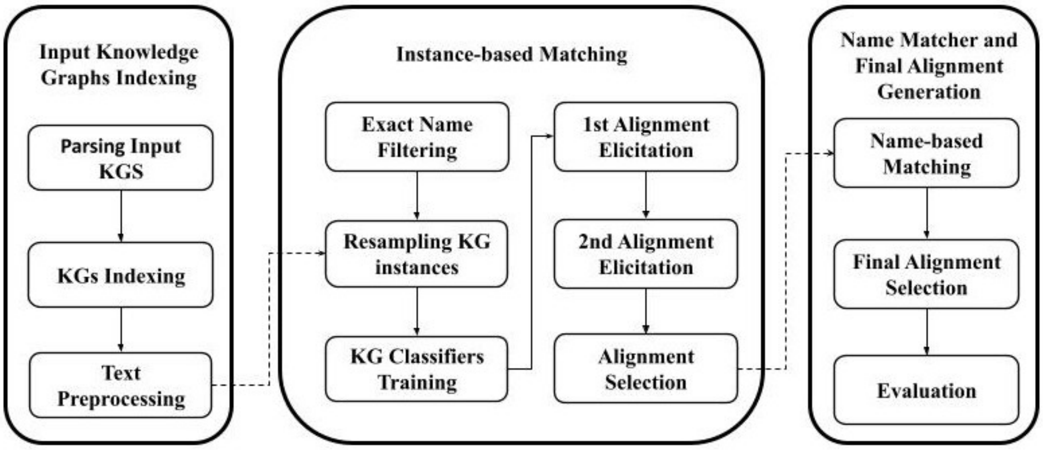


Figure 1: An overview of *KGMatcher+* process

makes the matcher particularly useful for matching large KGs with numerous populated and overlapping instances such as DBpedia [4] and YAGO [5]. This is the second OAEI participation of this matching system, KGMatcher participated in the OAEI in 2021 [6].

1.2. Specific techniques used

Given two input knowledge graphs, $\mathcal{O}$ and $\mathcal{O}'$, where $\mathcal{O}$ has a set of classes $\mathcal{O} = \{C^0_{\mathcal{O}}, C^1_{\mathcal{O}}, .., C^i_{\mathcal{O}}\}$, and each class contains a set of instances $C^i_{\mathcal{O}} = \{e^i_0, e^i_1, ..., e^i_n\}$. Similarly, $\mathcal{O}'$ contains a set of classes such that $\mathcal{O}' = \{C^0_{\mathcal{O}'}, C^1_{\mathcal{O}'}, .., C^j_{\mathcal{O}'}\}$ where $C^j_{\mathcal{O}'} = \{e^j_0, e^j_1, ..., e^j_m\}$. *KGMatcher+* has two main components: an **instance-based matcher** and a **name matcher**. The workflow of *KGMatcher+* is illustrated in figure 1.

1.2.1. Preprocessing

Given the two input KGs, the matcher starts by parsing and indexing the lexical data of the two KGs separately. Following the standard free text search/index approach, an index is created for each KG where each class is treated as a document and the content of each ‘document‘ is the concatenation of the class’s instance labels. In addition to the standard text cleaning processes, a word segmentation method is applied in order to separate multi-word entities, e.g., `academicfield`. Using a general dictionary, this method is able to infer the spaces between words and replace them with a space.

1.2.2. Instance-based Matcher

The first matching component of *KGMatcher+* belongs to the extensional matcher category. It uses a self-supervised machine learning approach to map KG classes based on their instances overlap. The matching is done in a two-way classification fashion where a *KG classifier* is trained using one KG’s instances data. Later on, that classifier is used to classify any instance name into one of its classes. Here, we summarize the matching process of *KGMatcher+*, however, readers may refer to [3] for further details.

- *Exact Name Filtering.* The matcher starts by applying an exact name filter to exclude class labels that exist in both input KGs. Given the large number of classes in typical KGs, this step works as a blocking strategy that reduces the search space of the instance-based matcher.

- *Resampling KGs Instances.* The class distribution in typical public KG tends to be highly imbalanced. Thus, the goal of this step is to balance the number of populated instances in the two input KGs to avoid biased classification results. The previous version of the system was only targeting the majority, i.e., large classes, by undersampling their instances using TF-IDF approach [7]. Different from the previous version, here, we introduced a new sampling strategy that combines undersampling the majority classes with oversampling classes with fewer instances, i.e., the minority classes. In [2], we studied 6 different strategies for handling class distribution including state-of-the-art methods such as SMOTE [8] and using cost-sensitive learning [9]. The sampling method that outperformed all other methods was when we used the TF-IDF undersampling method in addition to using random oversampling. The standard TF-IDF method which is often deployed to measure word relevance in a collection of documents is used to undersample KGs classes. Here, the TF-IDF of a word in each class represents the relevance of that word in a particular class in comparison to other classes in the KG. Therefore, for each `majority` class, the most frequent words in terms of TF-IDF score are used to undersample its instance names. Therefore, instance names that do not compose any of the words with high TF-IDF scores are discarded. Then, random oversampling is used to generate repeated random samples of instances in the classes that fall in the `minority` class category. As a result, a more balanced and indicative set of KG instances is obtained to be used as training data. Readers interested in further details of the sampling strategies incorporated in *KGMatcher+* may refer to [2].

- *Training KG classifiers.* Here, a KG classifier will be separately trained for each input KG using the previously re-sampled instances data. Pre-trained word embedding is used here as a feature to capture and present the semantics of KG instance names. Compared to traditional feature representation methods, word embedding and language models are recognized as effective ways to capture the semantic similarity of words. *KGMatcher+* is able to train two types of classifiers, a Deep Neural Network (DNN) model[1] [2] similar to other successful NLP tasks such as [10] and [11], and a pre-trained BERT model [12]. *KGMatcher+* will automatically opt to use the BERT model if a GPU is available during the runtime. The output of this phase is two classifiers, $CLS_{\mathcal{O}}$ and $CLS_{\mathcal{O}'}$ trained using the instances from the two input KGs $\mathcal{O}$, and $\mathcal{O}'$ respectively.

[1] The parameters selected for the DNN model: an input layer of pre-trained word embedding model followed by four fully connected hidden layers with 128, 128, 64, 32 rectified linear units. A dropout layer of 0.2 is added between each pair of dense layers for regulation. Finally, a *softmax* layer for multi-class classification, taking the total number of classes in the KG we are training a classifier for.

[2] The input layer is the Google News token-based model https://tfhub.dev/google/tf2-preview/nnlm-en-dim128/1

- *KG1 alignment elicitation* is the process of generating candidate pairs using the classifier trained on the first input KG. Candidate pairs are generated by iteratively applying the classifier $CLS_{\mathcal{O}}$ to instances in the other KG's classes. As a result, each instance name in $C^{j}_{\mathcal{O}'}$ is now classified into a class in $\mathcal{O}$. The candidate pair ($C^{i}_{\mathcal{O}}$,$C^{j}_{\mathcal{O}'}$) is added to the first candidate alignments set $A_{\mathcal{O}\rightarrow\mathcal{O}'}$ if the majority of $C^{j}_{\mathcal{O}'}$ were classified as instances of $C^{i}_{\mathcal{O}}$. A similarity score between [0,1] is obtained using the percentage of instances that voted for a particular class. Therefore, if 600 out of 1000 instance names in $C^{j}_{\mathcal{O}'}$ were voting for, $C^{i}_{\mathcal{O}}$ the similarity score of that pair will be 0.6.

- *KG2 alignment elicitation* is similar to the above-illustrated elicitation process. However, the roles of the two KGs are reversed where $CLS_{\mathcal{O}'}$, i.e., the classifier trained on the second KG ($\mathcal{O}'$), is applied to $\mathcal{O}$ instances in order to obtain the second candidate alignment set $A_{\mathcal{O}'\rightarrow\mathcal{O}}$.

- *Similarity computing* is where *KGMatcher+* combines the two candidate alignment sets resulting from the two-way classification method. First, the matcher separately stores each directional alignment in an alignment matrix of a $|\mathcal{O}|.|\mathcal{O}'|$ dimension. The two matrices are then aggregated into one matrix by taking the average similarity score of each pair. For example, if ($C^{6}_{\mathcal{O}}$,$C^{3}_{\mathcal{O}'}$,0.88) in $A_{\mathcal{O}\rightarrow\mathcal{O}'}$ and ($C^{5}_{\mathcal{O}'}$,$C^{3}_{\mathcal{O}}$, 0.64) in $A_{\mathcal{O}'\rightarrow\mathcal{O}}$) their aggregated similarity value will be 0.76. Consequently, the final alignments for this matcher are chosen by following the state-of-the-art automatic final alignment selection approach introduced in [13]. Given an alignment matrix, this method iteratively selects the pair with the highest similarity score for each class in both KGs.

1.2.3. Name Matcher

The second component of *KGMatcher+* is an element-level matcher, which measures the similarity of KG class labels. First, the edit distance of each class pair is measured, and then their semantic similarity is measured by a word embedding method. First, the Levenshtein distance is calculated for each class pair. Then, in terms of the word embedding similarity, a pre-trained `word2vec` model is used to represent class labels before measuring their cosine similarities. The semantic similarity is measured in a Vector Space Model, where words with high semantic relations are often represented closer to each other. In the case of multi-word labels, the vector representation of each word composing the label is aggregated with an element-wise average of the composing word vectors. Finally, the maximum of the two similarity measures is chosen as the name similarity of that pair. The threshold value of the name matcher is set to 0.8. To illustrate, if the word embedding similarity of (`RailwayStation`,`TrainStation`) is 0.83 while their Levenshtein distance is 0.56, the maximum similarity value, i.e., the word embedding similarity which is also higher than 0.8. Nonetheless, in case the two similarity scores are lower than the threshold, that pair will be excluded from the candidate alignment.

1.2.4. Post Processing

KGMatcher+ combines the results generated from the two matching components by following the same method described earlier to combine the two instance classification alignments.

1.2.5. Instance Matching

For the OAEI participation, we have adapted *KGMatcher+* to also match the instances of KGs. The instance matching component is very simple. First, standard text preprocessing techniques such as lower casing, and removing stopwords and non-alphanumeric characters are applied. Then, *KGMatcher+* generates candidate instance pairs based on the existence of the label in the opposite knowledge graph.

1.3. Adaptations made for the evaluation

KGMatcher+ is mainly developed with Python. To facilitate reusing and evaluating *KGMatcher+* and for the OAEI submission, it was packaged using a SEALS client. The wrapping service from the Matching EvaLuation Toolkit (MELT) [14] was used to warp the system's Python-process, and to generate the SEALS package.

2. Results

In this section, we present and discuss the results for each of the OAEI tracks where *KGMatcher+* was able to produce a non-empty alignment file. The results include the following OAEI tracks: Conference, Knowledge Graph, and Common Knowledge Graphs track.

2.1. Conference

In the Conference track, when following the rar2-M3 evaluation, *KGMatcher+* F1 score (0.52) is slightly lower than both baselines, i.e., *StringEquiv* (0.53) and *edna* (0.56). This particular evaluation, i.e., M3 takes into consideration both class and property matches. The fact that *KGMatcher+* does not match property justifies the negative impact of the undiscovered property alignments on the matcher's performance on this task. Further, given that the Conference track datasets do not include enough instances to apply the instance-based matcher, the name matcher is the only matcher applied to map classes. In terms of the cross-domain test case of mapping DBpedia and OntoFram, *KGMatcher+* is the second to the best-performing system.

2.2. Knowledge Graph

In the Knowledge Graph track, *KGMatcher+* was able to generate results for all 5 test cases at both classes and instances level. In terms of class matching, the matcher yields satisfactory results, with 0.79 for F1 score. The added instance matcher has positively impacted the overall matcher result on this task, with a precision of 0.94, a recall of 0.66, and F1 of 0.82. *KGMctcher+* is the second to the best-performing system in this track.

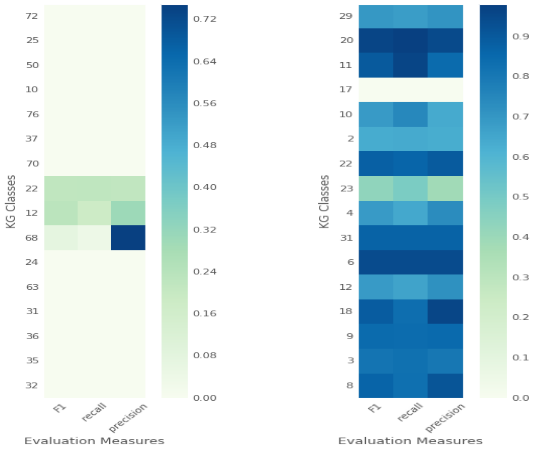
2.3. Common Knowledge Graphs

In this track, *KGMatcher+* was able to complete the task of matching the classes from 4 cross-domain and common KGs. On the task of matching NELL and DBpedia, the matcher obtained the highest F1 score of 0.95. In terms of the second task, which maps classes from Yago and

Wikidata, *KGMatcher+* is also the best-performing matching system with a recall of 0.83 and an F1 score of 0.91. *KGMctcher+* yields the best performance results on this track.

3. General comments

The results of *KGMatcher+* have been very encouraging. In the common knowledge graph track, it achieves outstanding results. This indicates that our hybrid approach, utilizing instances data to map KG classes, is able to outperform systems that use other matchers' combinations. It is important to note that the performance of *KGMatcher+* instance-based component depends on the dataset nature. Since *KGMatcher+* is learning KG classifiers by using general pre-trained word embedding models, the more representative the KG instances of real-world entities, the better the instance classification results. Figure 2 shows the difference between the performance when classifying instances from common KGs, e.g., NELL, compared to a single domain KG from the knowledge graph track. Note that the latter mainly annotates classes in the entertainment domain [15].



(a) (b)

Figure 2: The instance classification report of a 20 randomly sampled classes from the OAEI KG MemoryAlpha in (a) and NELL in (b). Note that y-axis numbers indicate class IDs.

4. Conclusion

As part of OAEI 2022, this paper presents *KGMatcher+*, a system for matching the schema of large-scale and domain-independent KGs by utilizing instances data. The process is done by learning KG classifiers, which are able to classify instances into a particular KG class. The results suggest that a hybrid approach that incorporates an instance-based technique can be highly effective particularly for matching large cross-domain KGs with imbalanced class distribution, such as Yago and Wikidata.

References

[1] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, A. Gómez-Rodríguez, Ontology matching: A literature review, Expert Systems with Applications (2015) 949–971.

[2] O. Fallatah, Z. Zhang, F. Hopfgartner, The impact of imbalanced class distribution on knowledge graphs matching, in: Proceedings of the 17th International Workshop on Ontology Matching (OM 2022), CEUR-WS, 2022.

[3] O. Fallatah, Z. Zhang, F. Hopfgartner, A hybrid approach for large knowledge graphs matching, in: Proceedings of the 16th International Workshop on Ontology Matching (OM 2021), CEUR-WS, 2021.

[4] C. Bizer, J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mende, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, C. Bizer, DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia, Semantic Web (2012) 1–5.

[5] T. P. Tanon, G. Weikum, F. Suchanek, Yago 4: A reason-able knowledge base, in: European Semantic Web Conference, Springer, 2020, pp. 583–596.

[6] O. Fallatah, Z. Zhang, F. Hopfgartner, Kgmatcher results for oaei 2021, in: CEUR Workshop Proceedings, volume 3063, 2021, pp. 160–166.

[7] H. Schütze, C. D. Manning, P. Raghavan, Introduction to information retrieval, volume 39, Cambridge University Press Cambridge, 2008.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2002) 321–357.

[9] C. Elkan, The foundations of cost-sensitive learning, in: International joint conference on artificial intelligence, volume 17, Lawrence Erlbaum Associates Ltd, 2001, pp. 973–978.

[10] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning based text classification: A comprehensive review, arXiv preprint arXiv:2004.03705 (2020).

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of machine learning research 12 (2011) 2493–2537.

[12] A. S. Maiya, ktrain: A low-code library for augmented machine learning, arXiv preprint arXiv:2004.10703 (2020).

[13] M. Gulić, B. Vrdoljak, M. Banek, Cromatcher: An ontology matching system based on automated weighted aggregation and iterative final alignment, Journal of Web Semantics 41 (2016) 50–71.

[14] S. Hertling, J. Portisch, H. Paulheim, MELT - matching evaluation toolkit, in: Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, 2019, pp. 231–245.

[15] S. Hertling, H. Paulheim, The knowledge graph track at oaei, in: European Semantic Web Conference, Springer, 2020, pp. 343–359.