Towards the smart use of embedding and instance features for property matching

Daniel Ayala, Inma Hernández, David Ruiz
Universidad de Sevilla
ETSII, Avda. Reina Mercedes, s/n. Sevilla, Spain
e-mail: {dayala1, inmahernandez, druiz}@us.es

Erhard Rahm
Leipzig University
Institut für Informatik. Leipzig 04109, Germany
e-mail: rahm@informatik.uni-leipzig.de

2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00209

***Abstract*—Data integration tasks such as the creation and extension of knowledge graphs involve the fusion of heterogeneous entities from many sources. Matching and fusion of such entities require to also match and combine their properties (attributes). However, previous schema matching approaches mostly focus on two sources only and often rely on simple similarity measurements. They thus face problems in challenging use cases such as the integration of heterogeneous product entities from many sources.**

We therefore present a new machine learning-based property matching approach called LEAPME (LEArning-based Property Matching with Embeddings) that utilizes numerous features of both property names and instance values. The approach heavily makes use of word embeddings to better utilize the domain-specific semantics of both property names and instance values. The use of supervised machine learning helps exploit the predictive power of word embeddings.

Our comparative evaluation against five baselines for several multi-source datasets with real-world data shows the high effectiveness of LEAPME.

***Index Terms*—data integration; machine learning; knowledge graphs;**

I. Introduction

Data integration tasks such as the creation and refinement of knowledge graphs have to increasingly deal with the matching and fusion of data from many sources, e.g., different web sites, already created knowledge bases and repositories. Such knowledge graphs (KG) physically integrate numerous entities with their properties (attributes) and relationships as well as associated metadata about entity types and relationship types in a graph-like structure [28]. Many companies (including Google, Facebook, and Amazon) are increasingly relying on the integrated and curated information in knowledge graphs and there is also an increasing amount of research on KG creation [2], [6], [10], [23], [31], [33], [35], [37] and KG exploitation, e.g. for question answering [16], [38].

Integrating new data sources and their entities into a KG requires matching the properties of entities, e.g., to focus entity matching on comparable properties or to fuse the values of equivalent properties.

Matching properties is far from trivial, especially with many sources. As an example, Fig. 1 shows camera entities (from a real dataset used in our evaluation) from three sources that may be integrated into a product KG together with some property matches indicated by symbols of the same shape. The example shows that there are numerous similar but differently named properties with diverse instance values. Matching properties often have completely different names, e.g., for properties "camera resolution", "effective pixels" and "megapixel".

To help solve the property matching problem in the case of such scenarios, we present a new approach called LEAPME (LEArning-based Property Matching with Embeddings). It uses supervised machine learning and makes use of the typically good availability of instance in a KG. LEAPME applies a dense neural network and a large set of features to classify a pair of properties from different sources as related or not.

The proposed features make heavy use of word embeddings computed from both the property names and their instance values. The use of embeddings gives the classifier information about the semantic proximity between two properties even when their string similarity is low. For example, we expect different words related to camera resolution such as "MP", "resolution" or "megapixels" to have similar embedding vectors. The use of property values provides additional information that is not tied to the name of a property, and makes the proposal applicable to scenarios in which the properties do not have meaningful names, e.g., identifiers that are automatically generated by information extraction approaches [30].The use of machine learning helps use these features in a smart way, learning what features are more important and how they must be combined.

The next section describes related work on schema matching and the previous use of machine learning for this task. Section III formally describes the problem of property matching. Section IV describes LEAPME in detail. Section V contains the evaluation with several baselines. Section VI summarizes our contributions and discusses potential future work. Additional details about our technique and an expanded evaluation can be found in [3].

II. Related work

In the last decades, a huge amount of research has been devoted to schema and ontology matching to automatically determine corresponding schema attributes (properties) and ontology concepts. As described in several survey articles and books [13], [24], [29], most of the proposed approaches focus on pairwise matching between two schemas or ontologies and utilize a combination of several similarity values to determine

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00209

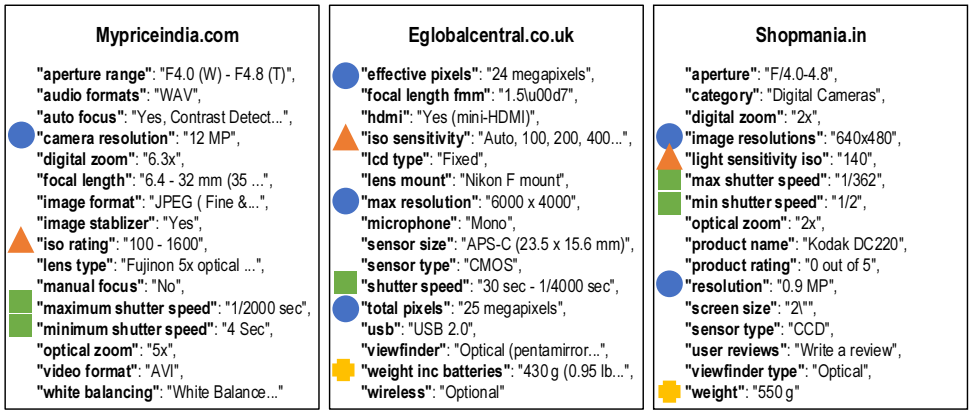

Fig. 1. Camera properties from different sources. Two properties from different sources being annotated with the same shape denotes a match.

likely matches. The most common approach is to determine the linguistic similarity of properties either based on string similarity metrics, synonym information from background knowledge resources such as dictionaries (e.g., WordNet [19]), or, more recently, pre-trained word embeddings [15], [18]. Background knowledge resources can even include a corpus of formerly matched schemas as support for a new match [20], [27]. Some approaches additionally utilize the structural similarity of elements (e.g., based on the similarity of neighbors in an ontology) and the similarity of associated instance data [11], [26].

The use of supervised machine learning is being increasingly applied for a simplified configuration of schema and ontology matching, since it can be considered a way to aggregate several similarity metrics or matchers, removing the need to set manual thresholds or use vector distance metrics such as the cosine similarity, which give the same weight to all features [8], [9], [12], [17], [21], [22], [32], [32], [34]. The training data consists of the similarity of matching and non-matching pairs of schema/ontology elements together with multiple similarity values, e.g., according to different linguistic and structural similarities. Surprisingly, instance similarities or word embeddings have not been utilized so far in these approaches.

Most previous work focuses on pairwise schema and ontology matching for two sources [27] while we have to deal with an arbitrary number of sources with different sets of properties per entity type. While multi-source property matching also builds on pairwise property matching, the degree of heterogeneity and thus the difficulty to achieve good match quality increases with more sources. In our approach, we will determine pairwise similarities between properties that can be maintained in a similarity graph of properties from several sources. Such a graph can be used as input for clustering so that all matching properties are in the same cluster that can be used as a basis to fuse these properties.

III. Problem Definition

We first provide some preliminary definitions that are needed to understand the description of the problem and our proposal.

Source: A source S is a location from where information comes, e.g., a website, a relational database, or a SPARQL endpoint, among other examples.

Entity and class: An entity e is a representation of something that can be uniquely identified, usually corresponding to some real world object. Entities belong to a certain source and a source-specific type or class C, and we denote the source and class of entity e with $S(e)$ and $C(e)$, respectively.

Property and instances: A property is an attribute to describe information about entities. The values of a property are literals known as instances. Our algorithm processes a collection of property instances represented as tuples (p, e, v) where p is the property name, e is the entity (identifier), and v is the property value. The components of a property instance i=(p, e, v) are denoted by $p(i)$, $e(i)$, and $v(i)$.

Class schema: For the sake of flexibility in applications such as E-commerce, we do not assume the existence of a predefined schema with a fixed set of properties per class. Rather, we view the schema of class C as the collection of all differently named properties for entities of class C in the respective sources. Individual entities may use any subset of these class properties.

Property matching: Task of determining correspondences between the properties of different class schemas from different sources.

The problem we address is property matching for properties of the same class (e.g., camera properties). We consider the case of multi-source matching so that properties may relate to entities from an arbitrary number of sources. Correspondences are not limited to equivalence relationships but also to more complex relationships between semantically related properties. A property in one source may thus have 0, 1 or several

matching properties in another source, e.g., as for property "shutter speed" in Figure 1.

IV. Our approach

Having defined the problem of property matching, we give an overview of our proposal LEAPME (Section IV-A), and describe in detail how features are computed (Section IV-B). We discuss the use of embeddings in Section IV-C. Finally, we describe aspects related to the implementation of LEAPME in Section IV-D.

A. Overview

LEAPME is a supervised ML-based property matching approach that focuses on the use of novel features. It computes features from property instances, property names, and property pairs to obtain large feature vectors that can be properly handled by a classifier.

Algorithm 1 describes the main steps of LEAPME.

1) First, there is the initialization of the instance feature vector IF, the property feature vector PF, the property pair feature vector PPF, and the output similarity graph (collection of matches) Sim (line 1 of Algorithm 1).
2) Next, the instance features are determined by every instance with the help of function $iFeatures$, and added to the respective property in the instance feature vector IF (lines 2-3 of Algorithm 1). The features we determine will be described below - they include meta-features about the instance values as well as an embedding vector for the specific property value.
3) In lines 4-6 of Algorithm 1 we compute property features with the help of function $pFeatures$. They can be derived for the property name or based on the aggregation of instance features, e.g., average values of numeric instance features.
4) For each property pair, we compute the property pairs features using function $ppFeatures$ (lines 7-9 of Algorithm 1), which may be partially based on the aggregation of property features.
5) We use the input training data with their labeled property pairs and associated feature vectors to train a classification model using function $trainClassifier$ in line 10 ($labeled(PPF)$ denotes the already labeled property pairs). Then, we apply the trained classifier to the unlabeled property pairs to obtain a match decision and similarity score for each pair (lines 11-12 of Algorithm 1).

Algorithm 1: LEAPME

Input:
- I: set of property instances from m sources
- labeled property pairs (training)

Output:
- Sim: set of property pairs with similarities (similarity graph)

Variables:
- IF: Map<$Property, FeaturesVectorSet$> with instance features vectors, grouped by property
- PF: Map<$Property, FeaturesVector$> with property features vectors.
- PPF: Map<$PropertyPair, FeaturesVector$> with property pair features vectors.
- m: classification model

```
1  initialize(IF, PF, PPF, Sim)
   // Steps 1-4: compute features
2  for i in I do
3      IF[p(i)] ← IF[p(i)] ∪ iFeatures(i))
4  for (p, V) in IF do
5      PF[p] ← pFeatures(p)
6  for p1 in keyset of PF do
7      for p2 from different source in keyset of PF do
8          PPF[(p1, p2)] ← ppFeatures(p1, p2)
   // Step 5: training and classification
9  m ← trainClassifier(labeled(PPF))
10 for (p1, p2) : v in unlabeled(PPF) do
11     Sim.add((p1, p2, m.classify(v)))
```

B. Features

Since we classify pairs of properties, the features that are ultimately fed to the classifier must be associated to a pair of properties. However, as we have mentioned, LEAPME considers features at several levels that can be later transformed into property pairs features. Next, we describe in detail each of these levels:

Instance features: These features are computed from each individual instance of a property (that is, a features vector is obtained for each property value) independently of the property names.

Property features: These features are computed for each individual property. They include all features computed from the property name, such as the average embeddings vector of its words. Furthermore, by grouping the instance features on a per-property basis, we can aggregate them and turn them into property features.

Property pair features: These features are computed for each pair of properties to be classified. These are the final features actually fed to the classifier. Aggregated property features can also be used to determine property pair features, e.g. by computing the numeric difference between two vectors of such features.

C. Embeddings and classification

When matching properties, a high value of the string similarity of the property names is usually a clear indicator of a match. Low similarity, however, can be caused by the issues we mentioned in Section I.

To overcome these limitations, we propose the use of word embeddings for both property names and property values. They can provide rich information about the semantics of a

property that can help solve some issues such as the potentially low string similarity between synonymous properties.

Embeddings vectors usually have hundreds of components with unknown meanings that may require nonlinear combinations to properly exploit their predictive power. For that reason, LEAPME uses a neural network for classification, which is also a popular choice in the related work and is able to properly weight features even when there is a large amount of them.

D. Implementation

Type	Id	Description	# of features
Instance	1	The fraction and number of occurrences of several character types (letters (uppercase, lowercase, and both), mark characters, numbers, punctuation, symbols, separators, other)	18
	2	The fraction and number of occurrences of several token types (words, words starting with a lowercase letter, words starting with an uppercase letter followed by a non-separator character, uppercase words, numeric strings)	10
	3	The numeric value of the instance (-1 if it is not a number)	1
	4	The average embeddings vector of the words in the instance	300
Property	5	The average of every instance feature	329
	6	The average embeddings vector of the words in the property name	300
Properties pair	7	The difference between the features vectors of the two properties	629
	8	The optimal string alignment distance between the property names	1
	9	The Levenshtein distance between the property names	1
	10	The Full Damerau-Levenshtein distance between the property names	1
	11	The longest common substring distance between the property names	1
	12	The 3-gram distance between the property names	1
	13	The cosine distance between the 3-gram profiles of the property names	1
	14	The Jaccard distance between the 3-gram profiles of the property names	1
	15	The Jaro-Winker distance between the property names	1

TABLE I
FEATURES USED IN OUR IMPLEMENTATION.

Table I provides an overview about the features we have implemented. Instance features are computed with TAPON [4], [5], which includes several format-related features to which we added the embedding ones.

To compute embeddings, we use the pre-trained GloVe approach [25][1], specifically for the uncased Common Crawl corpus that includes 300-dimensional vectors for 1.9 million words. Unknown words are mapped to a vector filled with zeroes. For each property value and name we determine the average embeddings of the individual words.

Regarding the architecture of the neural network behind LEAPME, it consists of two fully connected hidden layers of sizes 128 and 64. We use a batch size of 32 and perform 10 epochs with learning rate 10^{-3}, 5 with 10^{-4}, and 5 with 10^{-5}. We fine-tuned these hyper-parameters manually in preliminary tests, though most alterations (such as changing the size of the layers) do not significantly impact on the results. The final layer has two neurons from which the final score is obtained for the two possible outcomes (positive/negative). This allows the use of the positive output as a similarity score, which is useful for post-processing steps such as property clustering.

V. EVALUATION

We experimentally evaluate our property matching approach LEAPME on four real-word datasets with up to 24 sources. We analyze the impact of different amounts of training data and the effectiveness of the different kinds of features; in particular, the use of embeddings for both property values and property names. We further compare LEAPME with five baselines and study the use of transfer learning. The focus is on match quality with the standard metrics precision, recall and F-measure (F1 score).

We first give some details about the studied feature configurations for LEAPME and the baseline approaches. Next, we describe the four datasets. The results are discussed in subsection V-C. The evaluated implementations along with the detailed results and additional material are available online[2].

A. Feature configurations and baselines

The rich set of features exploited by supervised learning is a main advantage of LEAPME and we therefore analyze the effectiveness of the different kinds of features in detail. Along one dimension, we compare the use of instance-related features only, name-related features only and the combined use of both kinds of features. Another dimension is the consideration of embedding-based features only, non-embedding features only or the combined use of both kinds of features. In total, this sums up to 9 possible feature configurations to analyze.

The LEAPME results are compared to the results obtained by the following baselines: the latest Github implementation of Agreement Maker Light [14] (AML); the latest Github implementation of FCA-Map [7]; an implementation of the machine learning proposal by Nezhadi et al. [22]; an implementation of SemProp [15] using the following thresholds: 0.2 for SynM, 0.2 for SeMa(-), and 0.4 for SeMa(+); and an implementation of the proposal by Duan et al. [11] based on local-sensitive hashing (LSH), using minhash with a band size of 1.

B. Datasets

For our evaluation, we use four real-word datasets with different kinds of e-commerce products (cameras, headphones, phones, and TV sets) from multiple sources.

All datasets align the properties in each source to a reference ontology. We consider that two properties are related (matching) when they are both aligned to the same reference property.

The camera dataset comes from the DI2KG19 challenge [1]. It is the largest dataset with 24 sources, more than 3200 properties and about 9200 matching property pairs. We limited the number of entities to 100 per source in order to balance

[1]https://nlp.stanford.edu/projects/glove/

[2]https://github.eii.us.es/dayala1/LEAPME

Info	Dataset	Train. %	LEAPME P	LEAPME R	LEAPME F1	LEAPME(emb) P	LEAPME(emb) R	LEAPME(emb) F1	LEAPME(-emb) P	LEAPME(-emb) R	LEAPME(-emb) F1	Nezhadi P	Nezhadi R	Nezhadi F1	AML P	AML R	AML F1	FCA-Map P	FCA-Map R	FCA-Map F1	SemProp P	SemProp R	SemProp F1	LSH P	LSH R	LSH F1
Instances	cameras	20%	0.66	0.55	**0.59**	0.72	0.52	0.58	0.55	0.43	0.44	-	-	-	-	-	-	-	-	-	-	-	-	0.54	0.73	0.62
		80%	0.93	0.75	0.83	0.91	0.77	**0.83**	0.64	0.59	0.61	-	-	-	-	-	-	-	-	-	-	-	-			
	headphones	20%	0.54	0.61	0.56	0.61	0.64	**0.60**	0.54	0.57	0.54	-	-	-	-	-	-	-	-	-	-	-	-	0.75	0.43	0.55
		80%	0.76	0.70	**0.69**	0.64	0.70	0.64	0.60	0.51	0.53	-	-	-	-	-	-	-	-	-	-	-	-			
	phones	20%	0.60	0.59	0.58	0.58	0.63	**0.59**	0.47	0.41	0.42	-	-	-	-	-	-	-	-	-	-	-	-	0.74	0.21	0.33
		80%	0.84	0.75	0.79	0.85	0.74	**0.79**	0.59	0.44	0.50	-	-	-	-	-	-	-	-	-	-	-	-			
	tvs	20%	0.61	0.62	**0.60**	0.61	0.62	0.60	0.49	0.57	0.52	-	-	-	-	-	-	-	-	-	-	-	-	0.78	0.28	0.41
		80%	0.83	0.74	**0.78**	0.84	0.73	0.78	0.65	0.60	0.61	-	-	-	-	-	-	-	-	-	-	-	-			
Names	cameras	20%	0.89	0.88	**0.88**	0.87	0.86	0.86	0.91	0.75	0.82	0.86	0.82	0.83	0.99	0.61	0.75	0.99	0.38	0.55	0.82	0.75	0.78	-	-	-
		80%	0.99	0.98	**0.98**	0.98	0.98	0.98	0.95	0.76	0.84	0.96	0.93	0.94										-	-	-
	headphones	20%	0.68	0.79	**0.73**	0.67	0.81	0.72	0.82	0.62	0.70	0.73	0.69	0.70	0.95	0.36	0.52	0.99	0.37	0.54	0.67	0.48	0.56	-	-	-
		80%	0.84	0.82	**0.82**	0.83	0.81	0.81	0.91	0.58	0.70	0.80	0.72	0.75										-	-	-
	phones	20%	0.70	0.71	**0.70**	0.65	0.74	0.67	0.80	0.51	0.61	0.64	0.56	0.59	0.98	0.34	0.50	0.99	0.34	0.50	0.62	0.68	0.65	-	-	-
		80%	0.93	0.84	**0.88**	0.91	0.85	0.88	0.92	0.51	0.66	0.74	0.68	0.71										-	-	-
	tvs	20%	0.62	0.77	0.67	0.70	0.78	0.72	0.85	0.68	**0.75**	0.67	0.70	0.68	0.97	0.40	0.57	0.99	0.34	0.50	0.66	0.65	0.66	-	-	-
		80%	0.95	0.86	**0.90**	0.93	0.84	0.88	0.93	0.70	0.80	0.83	0.79	0.81										-	-	-
Both	cameras	20%	0.91	0.83	**0.87**	0.83	0.77	0.79	0.88	0.74	0.80	0.86	0.82	0.83	0.99	0.61	0.75	0.99	0.38	0.55	0.82	0.75	0.78	0.54	0.73	0.62
		80%	0.99	0.97	**0.98**	0.98	0.97	0.98	0.93	0.82	0.87	0.96	0.93	0.94												
	headphones	20%	0.74	0.81	**0.76**	0.65	0.80	0.70	0.79	0.68	0.73	0.73	0.69	0.70	0.95	0.36	0.52	0.99	0.37	0.54	0.67	0.48	0.56	0.75	0.43	0.55
		80%	0.89	0.87	0.88	0.88	0.90	**0.89**	0.80	0.68	0.72	0.80	0.72	0.75												
	phones	20%	0.71	0.72	**0.70**	0.59	0.70	0.63	0.66	0.52	0.56	0.64	0.56	0.59	0.98	0.34	0.50	0.99	0.34	0.50	0.62	0.68	0.65	0.74	0.21	0.33
		80%	0.93	0.85	**0.89**	0.92	0.86	0.89	0.83	0.57	0.68	0.74	0.68	0.71												
	tvs	20%	0.64	0.80	**0.70**	0.60	0.81	0.67	0.71	0.67	0.67	0.67	0.70	0.68	0.97	0.40	0.57	0.99	0.34	0.50	0.66	0.65	0.66	0.78	0.28	0.41
		80%	0.95	0.89	**0.92**	0.94	0.86	0.90	0.88	0.77	0.82	0.83	0.79	0.81												

TABLE II
RESULTS SUMMARY WITH F1 SCORES. LEAPME(EMB) = LEAPME WITH EMBEDDING FEATURES ONLY. LEAPME(-EMB) = LEAPME WITHOUT EMBEDDING FEATURES.

their size and impact. The other datasets contain headphones, phones and TV product entities and correspond to the WDC Gold Standard for Product Matching and Product Feature Extraction [36]. These are much smaller than the camera dataset and there are different numbers of entities per source leading to a less balanced setting than that of the camera dataset. In our analysis of the results, we will refer to the three smaller and imbalanced datasets as low-quality datasets as opposed to the high-quality camera dataset.

We take a fraction of the sources of a dataset (at random) for training. We use the examples that involve two sources of data in the training set to train the classifier, and test it with the rest. We performed experiments using different training fractions: 0.2 and 0.8. For each of these fractions and for each dataset, we ran LEAPME 25 times, using different random combinations of training sources.

For all datasets, the training data consists of two negative (non-matching) pairs of properties for every positive (matching) pair, and the negative pairs are randomly selected.

C. Results

Next, we compare the results obtained by different configurations of LEAPME, as well as those obtained by the five baselines.

We summarize the average results, including F1 scores, in Table II for both 20% and 80% training data. The table also provides results for the sole use of instance features and the sole use of name features, again differentiated by the use of embedding features only, non-embedding features only or both. The best F1 results of each row have been marked in bold. We make the following observations:

- Unsupervised techniques can achieve a high precision but struggle to reach a similar recall.
- For all datasets, LEAPME achieves a better F1 score than all baseline approaches even when using only 20% training data. For 80% training data, it achieves excellent F1 scores from 88% (for headphones) to 98% (for cameras). In this case, the baselines are outperformed especially for the low-quality and more challenging datasets (headphones, phones, and TVs). The unsupervised baselines were outperformed by up to 42 F1 percentage points (50 vs 92% for the TV dataset) and the supervised baseline of Nezhadi by up to 18 percentage points (71 vs 89% for the phones dataset).
- When only using property names LEAPME without embedding features already outperforms the baselines. The embedding features for property names are the most effective features in LEAPME. Their use alone is more more effective than the use of non-embedding features relying on string similarities.
- Only using instance features achieves weaker results for LEAPME than using name features especially with little training. Again, using embedding features is more effective than using the non-embedding ones that focus on format-oriented meta-features. Still, the combination of both instance and name features helps to achieve a slight improvement over the sole use of name features in most cases.

VI. Conclusions

We have presented LEAPME, a new powerful approach for matching properties from many sources. It is a machine learning approach that utilizes a large spectrum of features, in particular embedding features, on both property names and instance values. Our evaluation with four real-world multi-source datasets shows that LEAPME clearly outperforms several baseline approaches representing the current state-of-the art. The improvements are even achieved for relatively little training data.

In future work, we will investigate the use of LEAPME within a more comprehensive data integration approach for knowledge graphs that also includes entity matching and clustering as well as data fusion. In particular, we plan to evaluate different methods for deriving clusters of equivalent properties from the match results determined with LEAPME.

Acknowledgement

Our work was supported the Spanish R&D&I programme by grants TIN2016-75394-R, PID2019-105471RB-I00, and P18-RT-1060.

References

[1] 1st Int. Workshop on challenges and experiences from Data Integration to Knowledge Graphs. Di2kg challenge. http://di2kg.inf.uniroma3.it/. Accessed: 2020-02-13.

[2] D. Ayala, A. Borrego, I. Hernández, C. R. Rivero, and D. Ruiz. Aynec: All you need for evaluating completion techniques in knowledge graphs. In *Proc. European Semantic Web Conference (ESCW)*, pages 397–411. Springer, 2019.

[3] D. Ayala, I. Hernández, D. Ruiz, and E. Rahm. Leapme: Learning-based property matching with embeddings. *arXiv preprint arXiv:2010.01951*, 2020.

[4] D. Ayala, I. Hernández, D. Ruiz, and M. Toro. Tapon: A two-phase machine learning approach for semantic labelling. *Knowledge-Based Systems*, 163:931–943, 2019.

[5] D. Ayala, I. Hernández, D. Ruiz, and M. Toro. Tapon-mt: A versatile framework for semantic labelling. *Information Systems*, 83:57–68, 2019.

[6] A. Borrego, D. Ayala, I. Hernández, C. R. Rivero, and D. Ruiz. Generating rules to filter candidate triples for their correctness checking by knowledge graph completion techniques. In *Proc. 10th Int. Conf. on Knowledge Capture*, pages 115–122, 2019.

[7] F. Chang, G. Chen, and S. Zhang. FCAMap-KG results for OAEI 2019. In *CEUR Workshop Proceedings*, volume 2536, pages 138–145. RWTH, 2019.

[8] C. Curino, G. Orsi, and L. Tanca. X-som: A flexible ontology mapper. In *18th Int. Workshop on Database and Expert Systems Applications (DEXA)*, pages 424–428. IEEE, 2007.

[9] W. E. Djeddi and M. T. Khadir. Ontology alignment using artificial neural network for large-scale ontologies. *Int. Journal of Metadata, Semantics and Ontologies*, 8(1):75–92, 2013.

[10] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proc. 20th Int. conference on Knowledge discovery and data mining (KDD)*, pages 601–610, 2014.

[11] S. Duan, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Instance-based matching of large ontologies using locality-sensitive hashing. In *International Semantic Web Conference*, pages 49–64. Springer, 2012.

[12] K. Eckert, C. Meilicke, and H. Stuckenschmidt. Improving ontology matching using meta-level learning. In *European Semantic Web Conference (ESWC)*, pages 158–172. Springer, 2009.

[13] J. Euzenat, P. Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[14] D. Faria, C. Pesquita, T. Tervo, F. M. Couto, and I. F. Cruz. AML and AMLC Results for OAEI 2019. 2536:123–130, 2019.

[15] R. C. Fernandez, E. Mansour, A. A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 989–1000. IEEE, 2018.

[16] X. Huang, J. Zhang, D. Li, and P. Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 105–113, 2019.

[17] R. Ichise. Machine learning approach for ontology mapping using multiple concept similarity measures. In *Int. Conf. on Computer and Information Science (ICIS)*, pages 340–346. IEEE, 2008.

[18] P. Kolyvakis, A. Kalousis, and D. Kiritsis. Deepalignment: Unsupervised ontology matching with refined word vectors. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 787–798, 2018.

[19] F. Lin and K. Sandkuhl. A survey of exploiting Wordnet in ontology matching. In *IFIP Int. Conf. on Artificial Intelligence in Theory and Practice*, pages 341–350. Springer, 2008.

[20] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *21st International Conference on Data Engineering (ICDE'05)*, pages 57–68. IEEE, 2005.

[21] A. Marie and A. Gal. Boosting schema matchers. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 283–300. Springer, 2008.

[22] A. H. Nezhadi, B. Shadgar, and A. Osareh. Ontology alignment using machine learning techniques. *Int. Journal of Computer Science & Information Technology*, 3(2):139, 2011.

[23] D. Obraczka, A. Saeedi, and E. Rahm. Knowledge graph completion with FAMER. In *Proc. DI2KG*, 2019.

[24] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, and A. Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971, 2015.

[25] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[26] H. Qin, D. Dou, and P. LePendu. Discovering executable semantic mappings between ontologies. In *OTM Confederated Int. Conferences "On the Move to Meaningful Internet Systems"*, pages 832–849. Springer, 2007.

[27] E. Rahm. Towards large-scale schema and ontology matching. In *Schema matching and mapping*, pages 3–27. Springer, 2011.

[28] E. Rahm. The case for holistic data integration. In *Proc. ADBIS*, pages 11–27. Springer, 2016.

[29] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[30] J. C. Roldán, P. Jiménez, and R. Corchuelo. On extracting data from tables that are encoded using html. *Knowledge-Based Systems*, page 105157, 2019.

[31] A. Saeedi, E. Peukert, and E. Rahm. Incremental multi-source entity resolution for knowledge graph completion. In *Proc. ESWC (to appear)*, 2020.

[32] K. M. Shenoy, K. Shet, and U. D. Acharya. Nn-based ontology mapping. In *Int. Conf. on Advances in Information Technology and Mobile Communication*, pages 122–127. Springer, 2012.

[33] L. Shi, S. Li, X. Yang, J. Qi, G. Pan, and B. Zhou. Semantic health knowledge graph: Semantic integration of heterogeneous medical knowledge and services. *BioMed research international*, 2017, 2017.

[34] D. Spohr, L. Hollink, and P. Cimiano. A machine learning approach to multilingual and cross-lingual ontology matching. In *Int. Semantic Web Conference (ISWC)*, pages 665–680. Springer, 2011.

[35] P. Szekely, C. A. Knoblock, J. Slepicka, A. Philpot, A. Singh, C. Yin, D. Kapoor, P. Natarajan, D. Marcu, K. Knight, et al. Building and using a knowledge graph to combat human trafficking. In *Int. Semantic Web Conference (ISWC)*, pages 205–221. Springer, 2015.

[36] Web Data Commons. Gold standard for product matching and product feature extraction. http://webdatacommons.org/productcorpus/. Accessed: 2020-02-13.

[37] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu. Cross-lingual knowledge graph alignment via graph matching neural network. *arXiv preprint arXiv:1905.11605*, 2019.

[38] W. Zheng, J. X. Yu, L. Zou, and H. Cheng. Question answering over knowledge graphs: question understanding via template decomposition. *Proc. of the VLDB Endowment*, 11(11):1373–1386, 2018.