Context-Aware Instance Matching Through Graph Embedding in Lexical Semantic Space

Ali Assi[1](✉), Hamid Mcheick[2], and Wajdi Dhifli[3]

[1] University of Quebec At Montreal, Montreal H2X3Y7, Canada
assi.ali@courrier.uqam.ca
[2] University of Quebec At Chicoutimi,
555, University Boulevard, Chicoutimi, Canada
hamid_mcheick@uqac.ca
[3] Univ. Lille, EA2694, 3 rue du Professeur Laguesse,
BP83, 59006 Lille Cedex, France
wajdi.dhifli@univ-lille.fr

Abstract. Instance matching is one of the processes that facilitate the integration of independently designed knowledge bases. It aims to link co-referent instances with an `owl:sameAs` connection to allow knowledge bases to complement each other. In this work, we present VDLS, an approach for automatic alignment of instances in RDF knowledge base graphs. VDLS generates for each instance a virtual document from its local description (*i.e.*, data-type properties) and instances related to it through object-type properties (*i.e.*, neighbors). We transform the instance matching problem into a document matching problem and solve it by a vector space embedding technique. We consider the pretrained word embeddings to assess words similarities at both the lexical and semantic levels. We evaluate our approach on multiple knowledge bases from the instance track of OAEI. The experiments show that VDLS gets prominent results compared to several state-of-the-art existing approaches.

Keywords: Data linking · Instance matching · RDF graph · Semantic web

1 Introduction

Linked Open Data (LOD) includes several Knowledge Bases (KBs) expressed by ontologies in the form of RDF graphs from various domains of applications such as geography, biology, *etc.* These KBs are often created independently from each other. They may contain resources (with distinct descriptions) that are co-referring, but not explicitly defined. *Instance Matching* (IM) is the process of matching instances across these different knowledge bases, that refer to the

https://doi.org/10.1007/978-3-030-22999-3_37

same entity in the real-world (*e.g.*, the same person in two different knowledge bases).

The existing IM approaches can be categorized as domain-dependent [19,20] and domain-independent approaches [7,12,16,21]. An IM approach is called domain-dependent when it deals with KBs related to a specific domain (*e.g.*, music domain). Otherwise, it is called domain-independent. For more details about IM approaches, we refer the reader to [6,15]. In fact, most of the existing approaches mainly depend on the result of the instances' property alignments. The property alignment process aims to match properties from different KBs, that have similar semantics. This process is not trivial since KBs are usually expressed by their specific ontologies in order to describe the RDF graph instances and relations. For example, the information included in the "address" property in a source KB can be represented by several properties (*e.g.*, street, zipcode, *etc.*) in another KB. Thus, the property alignment will not find its corresponding property in the target KB. As a result, such information will be ignored even if it may be worthy to consider it for IM. Indeed, in some cases the description of instances does not carry properties with similar semantics. However, they can contain information with some expressive relationships. For example, given a KB that describes the instance *"Arthur Purdy Stout"* with two properties: "`f:profession`" and "`f:death_Place`", the former property says *"Researcher in Surgical pathology and pathologists"* as his profession. The latter one indicates *"New York City"* where he died. Now given another KB that describes his "`d:place_of_birth`" as *"New York City"* and the information about his career can be deduced from the text given by the property "`d:description`" which states *"Arthur Purdy Stout, M.D., (1885–1967) was a noted American surgeon and diagnostician."*. As you notice, the two descriptions have the same meaning. However, they cannot be inferred by any of the existing property alignment-based approaches.

In this paper, we propose an approach that tackles these drawbacks. Our approach represents instances as virtual documents where each of the latter is represented by a collection of words, and is generated for each instance declared in the KB. It consists of a "bag of words" extracted from the identifier of the instance, predicates, as well as the ones from all of its neighbors. To capture the semantic string similarity between words as "pathologist" and "diagnostician" in the above example, we use the distributed representation of words (*e.g.*, FastText [3], GloVe [18]) also known as word embedding. This latter represents words by a low dimensional dense vector, such that the vectors for similar words (*i.e.*, "pathologist", "diagnostician") are close to each others in their semantic embedding space.

Our major contributions are summarized as follows: (1) We propose a new idea for building virtual documents for instances. (2) We include words embeddings for capturing their semantics (*i.e.*, synonym and terminological variants). (3) We transform the instance matching problem into a document similarity problem and we solve it using lexical semantic similarity technique [13].

We experimentally validate our approach, Virtual Document Lexical Similarity (termed VDLS), on four KBs from the benchmark instance matching track of OAEI 2009 and 2010. The obtained results show that our approach gets highly competitive results compared to state-of-the-art approaches.

2 Preliminaries

In the semantic web, the meaning of a "concept" is similar to the notion of a `Class` in the Object-Oriented Programming (OOP) view. Thus, resources created according to the structure of a class are known as instances of that class [1].

The Resource Description Framework (RDF) data model [9] represents the descriptions of the entities (*i.e.*, concepts) and the instances by RDF expressions, called triples, in the form `<subject, predicate, object>`. A `subject` can be a URI or a blank node. The latter represents an anonymous entity. An `object` can be a URI, a blank node, or a basic value (*e.g.*, a string, a date, an integer, *etc*). A `predicate` allows to model a relationship between the `subject` and the `object`.

Definition 1 ***(RDF knowledge base graph).*** *An RDF Knowledge Base (KB) graph is a set of facts in the form* `<subject, predicate, object>` $\in (\mathcal{E} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{B})$*, where* $\mathcal{E}$ *is the set of instances,* $\mathcal{B}$ *is the set of blank nodes,* $\mathcal{P}$ *is the set of predicates and* $\mathcal{L}$ *is the set of literals (basic values).*

An RDF KB graph can adhere or not to an ontology. In the rest of the paper, we write KB shortly to refer to RDF KB graph.

Instance Matching (IM) is the problem of identifying instances that co-refer to the same object of the real world. It can be seen as a process of building the predicate `owl:sameAs` between the co-referent instances belonging to different KBs. Formally:

Definition 2 ***(Instance matching).*** *Given two input sets of instances* $\mathcal{S}$ *and* $\mathcal{T}$ *belonging to two different KBs, the aim of IM is to compute the set* $\mathcal{M} = \{(i_1, i_2) \mid (i_1, i_2) \in \mathcal{S} \times \mathcal{T}, <i_1, owl{:}sameAs, i_2>\}$.

Instance matching is a difficult task [5] mainly due to textual variation of the property values, incompleteness, presence of erroneous information, multilingualism, *etc.*

3 Instance Matching Through Graph Embedding and Lexical Semantic Similarity

3.1 Virtual Document

An instance e is described by a set of triples: $T(e) = \{t_1, \ldots, t_n\}$. All these triples share the same subject e denoted by a URI. For every instance in the KB, a "Virtual Document" (VD) is created. It is represented as a collection of words extracted from different parts of its triples (i.e., from the URI's subject and its properties' values). We consider to treat the URIs as literals as in [17]. This approach is based on the assumption that many URIs encompass valuable information. It detects a pattern from the characters of the URIs. The pattern has the form (prefix-infix-(suffix)). The prefix (P) is the URI domain. The (optional) suffix (S) contains details about the format of the data or named anchor. The infix (I) represents the local identifier of the target instance.

Example 1. In the URI: http://people.csail.mit.edu/lalana_kagal/foaf#me, the prefix is "http://people.csail.mit.edu", the infix is "/lalana_kagal", and the suffix is "/foaf#me".

Definition 3 ***(Statement).*** *A statement t is the smallest irreducible representation for linking one object s to another object o or a literal l via a predicate p. Formally: $t = <s, p, o>$ where $s \in \mathcal{E} \cup \mathcal{B}$, $p \in \mathcal{P}$ and $o \in \mathcal{E} \cup \mathcal{B} \cup \mathcal{L}$.*

In the following, we refer to the infix of a given URI e by $I(e)$. Indeed, we refer to the different parts of a statement t by $subject(t)$ to designate its subject (*i.e.*, s), by $object(t)$ to designate its object (*i.e.*, o) when this object is a URI or by $value(t)$ when the object is a basic value. We have only included in $T(e)$ the forward statements (triples) from the instance e. We suppose that such triples allow to describe the main features of the instance e. Thus, we define the neighbors of e as follows:

Definition 4 ***(Forward neighbors).*** *Let e be an instance, $\mathcal{B}$ be the set of blank nodes and t be any statement with e as its subject. The instance e has a set of forward neighbors denoted by FN and defined as:*

$$FN(e) = \bigcup_{e=subject(t)} \{object(t)\} \tag{1}$$

Definition 5 ***(Local name of an instance).*** *Let e be an instance in $\mathcal{E}$. The local name of e, denoted by $LN(e)$, is equal to the infix of e, i.e., $LN(e) = I(e)$.*

Definition 6 ***(Local name of a blank node).*** *Let $\_:b$ be a blank node ($\_:b \in \mathcal{B}$) and t be any statement with $\_:b$ as its subject. The local name of $\_:b$, denoted by $LN(\_:b)$, is defined as equal to the local names of its direct 1-hop forward neighbors in the graph:*

$$LN(\_:b) = \sum_{o \in FN(\_:b)} N(o) \tag{2}$$

where N is a function that returns the name of the object $o \in \mathcal{E} \cup B$. The function N is equal to I when the object $o \in \mathcal{E}$. Whereas, it leads to a recursive extended definition of the local name of a blank node when $o \in \mathcal{B}$.

Definition 7 ***(Recursive local name of a blank node).*** *Let $\_:b$ be a blank node ($\_:b \in \mathcal{B}$) and t be any statement with $\_:b$ as its subject. The recursive local* *name of _:b, denoted by LN_k(_:b), is defined as the local names of leaf nodes of at most k-hops paths starting form _:b and ending in object nodes $o \in \mathcal{E}$:*

$$\forall k \geq 1, LN_k(\_:b) = LN(\_:b) + \sum_{\substack{\_:b=subject(t)\\ o=object(t)\\ o\in\mathcal{B}}} LN_{k-1}(o) \tag{3}$$

Note that the recursive definition could terminate in less then k-hops in the case where all the neighbor nodes are leaf nodes. Otherwise, the function terminates in k-hops (in the worst case), if there exists at least a path of length greater or equal to k composed of only blank nodes and that starts from the root blank node of the recursive function. The benefits of this limitation are twofold. Indeed, this allows to alleviate the computation in cases of big KB graph datasets or in cases where the datasets contain long paths of consecutive blank nodes. In addition, this avoids the trap of infinite loops in cases where a cycle of blank nodes exists.

Definition 8 ***(Local description of an instance).*** *Let e be an instance denoted by a Uniform Resource Identifier (URI) and t be any statement with e as its subject. The local description of e, denoted by Des(e), is a collection of words defined by:*

$$Des(e) = \alpha_1 \times LN(e) + \alpha_2 \times Data(e) + \alpha_3 \times \sum_{\substack{e=subject(t)\\ o=object(t)\\ o\in FN(e)}} LN(o) \tag{4}$$

where $Data$ is the set of *values* extracted from $T(e)$ (*i.e.*, basic values) and the coefficients α_1, α_2 and α_3 are three fixed constants in $\{0, 1\}$. LN depends on the strategy used when creating the URIs (*i.e.*, $\alpha_1 = 1$ when the URIs information is meaningful).

Definition 9 ***(Local description of a blank node).*** *Let _:b be a blank node (_:b $\in \mathcal{B}$) and t be any statement with _:b as its subject. The local description of _:b, denoted by LD(_:b), is defined as equal to its local data values (Data(_:b)) and the local descriptions of its direct 1-hop forward neighbors in the graph:*

$$LD(\_:b) = Data(\_:b) + \sum_{\substack{\_:b=subject(t)\\ o\in FN(\_:b)}} D(o) \tag{5}$$

where D is a function that returns the description of the object $o \in \mathcal{E} \cup \mathcal{B}$. If $o \in \mathcal{E}$, then the function D will be equal to Des. Whereas, in the case where $o \in \mathcal{B}$, then the local description of a blank node will be defined in a recursive way as follows.

Definition 10 ***(Recursive local description of a blank node).*** *Let _:b be a blank node (_:b $\in \mathcal{B}$) and t be any statement with _:b as its subject. The recursive* *local description of _:b, denoted by $LD_k(\_{:}b)$, is defined as the local descriptions of leaf nodes of the k-hops paths starting form _:b and ending in object nodes $o \in \mathcal{E}$:*

$$\forall k \geq 1, LD_k(\_{:}b) = LD(\_{:}b) + \sum_{\substack{o=object(t)\\ \_{:}b=subject(t)\\ o\in\mathcal{B}}} LD_{k-1}(o) \tag{6}$$

Note also that here the iterations of the recursive definition will terminate in at most k-hops for the same arguments discussed in Definition 7.

Definition 11 ***(Virtual document of an instance).*** *The virtual document of an instance e (denoted by $VD(e)$) is defined as:*

$$VD(e) = Des(e) + \alpha \times \sum_{e' \in FN(e)} Des(e') \tag{7}$$

where α is a parameter defined in $\{0, 1\}$. If the node has a rich description, then we can limit VD(e) to the local description of e provided in the set of 1-hop neighbors by setting the parameter α to 0. However, in some applications, the local description of e in the KB graph could be poor and thus it would be judicious to incorporate additional information on e from farther neighbors in the KB by performing walks in the graph. This could be performed by setting α to 1.

3.2 Lexical Semantic Vector

The lexical semantic vector method was introduced in [13] then modified in [10]. Given two VDs (VD_1, VD_2), we create a combined list of vocabulary, denoted by L, that consists of all the unique words in VD_1 and VD_2, *i.e.*, $L = VD_1 \cup VD_2$. Then, we compute the pairwise similarity of each word v_L in L with every word v_1 in VD_1. This leads to create a lexical semantic vector $\boldsymbol{V_1}$ that contains the maximum similarities between each word in L and all words in VD_1 (respectively for VD_2). Formally, each element in the lexical semantic vectors $\boldsymbol{V_1}$ and $\boldsymbol{V_2}$ is defined as:

$$\boldsymbol{V_{1j}} = \max_{1 \leq i \leq |VD_1|} Sim(v_{1i}, v_{Lj}) : \forall v_{Lj} \in L \tag{8}$$

$$\boldsymbol{V_{2j}} = \max_{1 \leq i \leq |VD_2|} Sim(v_{2i}, v_{Lj}) : \forall v_{Lj} \in L \tag{9}$$

where Sim is a similarity measure (cosine in this paper) that computes the similarity between pairs of words based on their embedding vectors.

Definition 12 ***(Lexical semantic word embedding).*** *A word embedding (also called dense distributed representation [2]) is a learned representation where each word is mapped to a real-valued dense vector in a semantic vector space. This is based on the assumption that words with similar contexts will have similar meanings and also will have similar representations (i.e., close vectors in the semantic vector space).*

3.3 Indexing

After determining the VDs that correspond to the source KB, we build an inverted-index, *i.e.*, a binary vector representation (1 the presence or 0 the absence) from the words of these VDs.

Infrequent-Words: One of the common problems encountered while building the inverted-index representation is that multiple used keywords are highly frequent and thus they do not sufficiently provide specificity for the instances. In order to alleviate this drawback, we define *Infrequency* as a measure that quantifies the infrequency of a given word in a dataset:

$$Infrequency(token) = \frac{1}{\log_2\big(WF(token) + 1\big)} \tag{10}$$

where WF is the number of VDs containing the token. *Infrequency* is theoretically defined in $[0, 1]$. A word appearing only once in a KB has an infrequency of 1. In contrast, the more a word appears in the KB, the more its infrequency is close to 0. In fact, 0 represents a theoretical lower bound for *infrequency* where *WF* leans toward $+\infty$. In practical cases, the maximum word frequency *WF* is bounded by the size of the dataset. Thus, we propose to normalize (MinMax normalization) the infrequency values to make them fall within the range of [0, 1]. Note that a token is considered as an infrequent-word when its infrequency (after normalization) is higher or equal to a predefined threshold γ.

Common Infrequent-Words: Let I_1 and I_2 be two instances for a source and a target KB, respectively. I_2 is considered as a potential matching candidate for I_1 (*i.e.*, *Candidate*(I_1, I_2) = *True*), if both instances share a number of "common infrequent-words" that is higher or equal to a predefined threshold β. Formally:

$$Candidate(I_1, I_2) = \mid I_1 \cap I_2 \mid \geq \beta \tag{11}$$

3.4 Approach Overview

Our approach VDLS (see Fig. 1) starts by parsing the given source and target KBs then it builds the VDs for each instance in them according to the Definition 11. Once the VDs corresponding to the source KB are determined, an inverted-index is set up from their words. Note that infrequency takes effect only over the words of the source KB. By using this index, each source instance gets its candidates from the target KB. A target instance is a candidate for a source instance if both share at least β common infrequent-words in their descriptions as described in Eq. 11. The similarity between the instances is computed between their corresponding Lexical Semantic embedding vectors as detailed in Sect. 3.2. Once this computation step is done, we select for each source instance its best candidate (*i.e.*, top similarity score) as a co-referent.

4 Experimental Evaluation

Datasets. We evaluate our proposed method on two benchmark datasets (Table 1): PR (synthetic) and A-R-S (real) used in OAEI 2010 and 2009, respectively. PR is a small benchmark including two persons data and one restaurants RDF data. A-R-S includes three RDF files named *eprints*, *rexa* and *dblp* related to scientific publications RDF data. For each set, the OAEI provides a mapping file ("gold standard") including the co-referent pairs between the source and target RDF files.

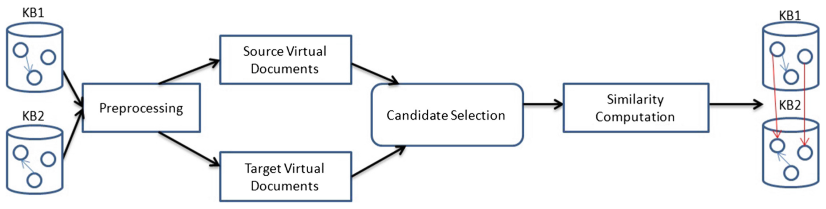

Fig. 1. An overview of our instance matching approach.

Table 1. Benchmarks statistics

Benchmarks	Datasets	Source	Target	Gold standard
PR	Person1	2000 URIs	1000 URIs	500
	Person2	2400 URIs	800 URIs	400
	Restaurant	399 URIS	2256 URIs	112
A-R-S	eprints-rexa	1130 URIS	18,492 URIs	777
	eprints-dblp	1130 URIs	2, 650, 832 URIs	554

Experimental Setup. For word embedding vectors, we use a dataset of pre-trained word vectors embedding developed by Facebook research team using FastText [3]. This dataset is trained on Wikipedia corpus and the generated vectors are of dimensionality equal to 300. If a word does not exist in the pre-trained embeddings, a unique vector will be created at test time. We run the experiments with different infrequency thresholds γ between [0,1] with a step size of 0.1. In Table 2, we report the optimal obtained results. We set $\beta = 2$ in Eq. 11 to represent the minimum required number of words between two instances to be considered as a candidate pair. In addition, we allow the paths to be composed at most of 2 blank nodes (*i.e.*, $k = 2$ in Eq. 3). Our approach is implemented in Java and the KBs are parsed using the RDF4J library. We execute the experiments on a Linux server with 128 GB RAM and two Intel E5-2683 v4 "Broadwell" CPUs of 2.1 Ghz.

4.1 Results and Discussion

Analysis of the Effect of the Context on the IM. We first analyze the effect of including (or not) the context information on the accuracy of the IM process. Table 2 reports the obtained results for VDLS with the context information ($\alpha = 1$) and without it ($\alpha = 0$). We notice that when the instances contain a sufficient local description information, the neighboring information either does not bring any benefit to the matching process (the case of Person1 and Person2) or it leads to an information overload and thus may hinder the IM results (the case of Restaurant). In the case of largely and noisy datasets as in eprints-rexa, the neighboring information permits to relax the effect of the noisy data and thus to enhance the accuracy of the matching process.

Table 2. F-measure results of VDLS with ($\alpha = 1$) and without ($\alpha = 0$) the context information.

Datasets	Person1	Person2	Restaurant	eprints-rexa	eprints-dblp
VDLS ($\alpha = 0$)	**1**	**1**	**1**	0.85	0.89
VDLS ($\alpha = 1$)	**1**	**1**	0.98	**0.87**	**0.9**

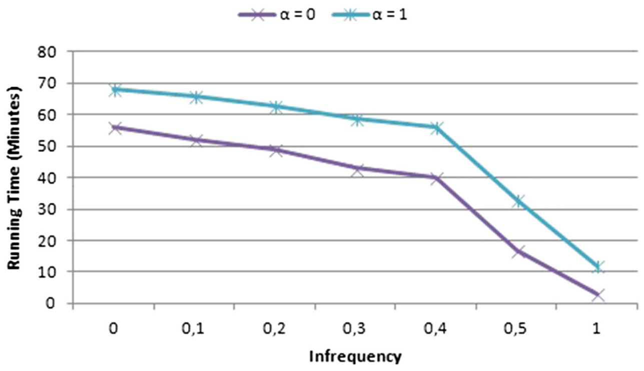

Fig. 2. The effect of Infrequency on the running time of VDLS (using the eprints-rexa datasets).

Analysis of the Effect of Word Infrequency on the Running Time. A higher F-measure requires a higher number of words and thus lower values of infrequency. This requires more running time. Figure 2 shows a clear example of the evolution of running time of VDLS with respect to different infrequency thresholds. Indeed, the required computation time increases when we add more words (lower infrequency) in the VDs. However, such a consideration makes VDLS subject to the "no free lunch" principle [22], where the gain in accuracy comes with an offset of computational cost. Hence, a trade-off between running time and accuracy is essential especially with large scale datasets.

Comparative Analysis. In Table 3, we report the results of multiple state-of-the-art IM approaches on the PR and A-R-S benchmarks and we compare our approach against them. On the PR benchmark, VDLS performed overall better than the other approaches. It was able to correctly retrieve all the co-referent instance, except for Restaurant where VDLS ($\alpha = 1$) where the F-measure was 0.98. This was due to the effect of information overload, where the context (neighbor) information did slightly hinder the accuracy of the IM process. As for the A-R-S benchmark, the IM was more difficult than with the PR benchmark. Indeed, VDLS outperformed all the other approaches in terms of F-measure yet the best results did not exceed 0.87 and 0.9 respectively for eprints-rexa and eprints-dblp. By analyzing the false matchings, we noticed that several of these instances were isolated nodes in the RDF graph and thus their VDs did lack context information.

Table 3. Comparative analyses of F-measure results on PR and A-R-S benchmarks.

Datasets	Person1	Person2	Restaurant	eprints-rexa	eprints-dblp
VDLS ($\alpha = 0$)	**1**	**1**	**1**	0.85	0.89
VDLS ($\alpha = 1$)	**1**	**1**	0.98	**0.87**	**0.90**
PARIS [21]	**1**	**1**	0.91	-	-
ObjectCoref [7]	**1**	0.95	0.90	-	-
ASMOV-D [8]	0.87	0.24	0.70	-	-
CODI [16]	0.91	0.36	0.72	-	-
RIMOM [11]	**1**	0.97	0.81	0.80	0.73
DSSIM [14]	-	-	-	0.38	0.13
HMATCH [4]	-	-	-	0.62	0.65
VMI [12]	-	-	-	0.85	0.66

5 Conclusion and Future Works

In this paper, we proposed a property-independent approach for IM and a new method for building VDs corresponding to the instances. We also transformed the IM problem into a document matching problem and we created lexical semantic vectors to measure the similarity between two VDs. We have compared our approach to state-of-the-art methods on benchmark datasets, and we achieved very promising results.

As future work, we will include more pruning heuristics to reduce the number of candidates for each query instance. It will also be interesting to propose an extension for VDLS that leverages parallel and distributed computation to efficiently handle big data scenarios with large-scale KBs.

References

1. Antoniou, G., Van Harmelen, F.: A Semantic Web Primer. MIT Press, Cambridge (2008)
2. Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems, vol. 19. MIT Press, Cambridge, MA (2007)
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
4. Castano, S., Ferrara, A., Montanelli, S., Lorusso, D.: Instance matching for ontology population. In: Italian Symposium on Advanced Database Systems (SEBD), pp. 121–132. ICAR-CNR (2008)
5. Ferrara, A., Lorusso, D., Montanelli, S., Varese, G.: Towards a benchmark for instance matching. In: Proceedings of the 3rd International Conference on Ontology Matching, vol. 431, pp. 37–48. CEUR-WS. org (2008)
6. Ferraram, A., Nikolov, A., Scharffe, F.: Data linking for the semantic web. Semant. Web Ontol. Knowl. Base Enabled Tools Serv. Appl. **169**, 326 (2013)
7. Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic web. In: Proceedings of the 20th International Conference on World Wide Web, pp. 87–96. ACM (2011)
8. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. Web Semant. Sci. Serv. Agents World Wide Web **7**(3), 235–251 (2009)
9. Klyne, G., Carroll, J.J.: Resource description framework (RDF): concepts and abstract syntax (2004). http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
10. Konopik, M., Prazák, O., Steinberger, D., Brychcín, T.: UWB at SemEval-2016 task 2: interpretable semantic textual similarity with distributional semantics for chunks. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 803–808 (2016)
11. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: a dynamic multistrategy ontology alignment framework. IEEE Trans. Knowl. Data Eng. **21**(8), 1218–1232 (2009)
12. Li, J., Wang, Z., Zhang, X., Tang, J.: Large scale instance matching via multiple indexes and candidate selection. Knowl. Based Syst. **50**, 112–120 (2013)
13. Li, Y., McLean, D., Bandar, Z., O'Shea, J., Crockett, K.A.: Sentence similarity based on semantic nets and corpus statistics. IEEE Trans. Knowl. Data Eng. **18**(8), 1138–1150 (2006)
14. Nagy, M., Vargas-Vera, M., Motta, E.: DSSim - managing uncertainty on the semantic web. In: CEUR Workshop Proceedings, OM, vol. 304. CEUR-WS.org (2007)
15. Nentwig, M., Hartung, M., Ngonga Ngomo, A.C., Rahm, E.: A survey of current link discovery frameworks. Semant. Web **8**(3), 419–436 (2017)
16. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging terminological structure for object reconciliation. In: Aroyo, L., et al. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 334–348. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13489-0_23
17. Papadakis, G., Demartini, G., Fankhauser, P., Kärger, P.: The missing links: discovering hidden same-as links among a billion of triples. In: Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services, pp. 453–460. ACM (2010)

18. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: In EMNLP (2014)
19. Raimond, Y., Sutton, C., Sandler, M.B.: Automatic interlinking of music datasets on the semantic web. In: LDOW, vol. 369 (2008)
20. Rowe, M., Group, O.: Interlinking distributed social graphs. In: In Linked Data on the Web Workshop, WWW 2009, April 2009 (2009)
21. Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS: probabilistic alignment of relations, instances, and schema. Proc. VLDB Endow. **5**(3), 157–168 (2011)
22. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)