BLINK: Blank Node Matching Using Embeddings

Alexander Becker[1 [0009−0003−8212−4647]], Mohamed Ahmed Sherif[1 [0000−0002−9927−2203]], and Axel-Cyrille Ngonga Ngomo[1 [0000−0001−7112−3516]]

DICE group, Department of Computer Science, Paderborn University
https://dice-research.org/
beckeral@campus.upb.de,mohamed.sherif@upb.de,axel.ngonga@upb.de

Abstract. Knowledge graphs (KGs) differ significantly over multiple different versions of the same data source. They also often contain blank nodes that do not have a constant identifier over all versions. Linking such blank nodes from different versions is a challenging task. Previous works propose different approaches to create signatures for all blank nodes based on named nodes in their neighborhood to match blank nodes with similar signatures. However, these works struggle to find a good mapping when the difference between the KGs' versions grows too large. In this work, we propose BLINK, an embedding-based approach for blank node linking. BLINK merges two KGs' versions and embeds the merged graph into a latent vector space based on translational embeddings and subsequently matches the closest pairs of blank nodes from different graphs. We evaluate our approach using real-world datasets against state-of-the-art approaches by computing the blank node matching for isomorphic graphs and graphs that contain triple changes (i.e., added or removed triples). The results indicate that BLINK achieves perfect accuracy for isomorphic graphs. For graph versions that contain changes, such as having up to 20% of triples removed in one version, BLINK still produces a mapping with an *Optimal Mapping Deviation Ratio* of under 1%. These results show that BLINK leads to a better linking of KGs over different versions and similar graphs adhering to the linked data guidelines.

Keywords: Knowledge graphs · Data Integration · Linked Data · Blank Nodes · Blank Node Matching

1 Introduction

Knowledge graphs (KGs) evolve over time where the updated KG versions normally include added, corrected, and deleted triples. While relational databases often create transaction logs describing changes to previous versions [11], RDF KGs are often distributed as a set of triples without any transaction history. Approaches such as [1] create versioning strategies composing single triple changes to hierarchies. Handling blank nodes is challenging because the identifiers of blank nodes from different graphs do not share global identifiers. Moreover, RDF tools and libraries normally do not keep the same blank node identifiers after processing [9]. This causes the need to rename them or completely delete and re-add parts of the graph that are only connected by one or multiple blank nodes [1].

According to the linked data guidelines [3], KGs should be linked to each other to add context to data and create opportunities to gain insights by combining different data

sources. For KGs with multiple versions, links must be created between the different versions to not lose connectivity to other graphs or add unnecessary redundancy [33] when publishing new versions. Creating links is not a big challenge for entities that are named by a URI but is a nontrivial problem for blank nodes that can have different local names over different versions.

Addressing this problem is important as blank nodes are widely popular in semantic web standards [21]. For instance, blank nodes are used as a way to model lists as defined in the RDF 1.1 standard [27] and recommended by SKOS [16]. Furthermore, blank nodes are used to model simple existential statements. For example, if it is certain that a professor offers a course in the next semester but it is not yet determined how the course is named, a blank node may be used to signal that there is going to be a course, but no other information is available yet. Blank nodes are also used in ontology languages. For example, in the OWL2 standard [22], restrictions are modeled using blank nodes:

```
_:x a owl:Restriction.
_:x owl:onProperty author.
_:x owl:allValuesFrom Researcher.
```

Blank nodes are also used in a variety of other contexts that require matching them, as discussed in [19], and are widely used in different KGs with more than 66% of all domains of the BTC-2012 dataset [12] using at least some blank nodes [15].

In this paper, we present BLINK, an approach for blank node linking that computes a mapping between the blank nodes of two graphs by first merging the two graphs into one merged graph and computing entity embedding vectors for the blank nodes in the merged graph. We then split the blank nodes of the merged graphs into two sets based on what graph they originated from and create a mapping between the two sets based on the distances between the latent vector embeddings of the blank nodes.

This paper is structured as follows: Section 2 contains the necessary definitions to formally define the problem of graph matching and error measures used to evaluate the quality of mappings. In Section 3, we give an overview of related work tackling the same problems we solve in this work. We then present our approach in Section 4. Section 5 contains experiments where we compare our approach to other algorithms on real-world KGs and synthetic ones. Finally, we conclude our findings and explain how this work can be extended in the future in Section 6.

2 Preliminaries

Definition 1 (Knowledge Graph). *A Knowledge Graph (KG) $G(\mathcal{R}, \mathcal{B}, \mathcal{P}, \mathcal{L})$ is a set of triples $(s, p, o) \in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{B})$, where $\mathcal{R}$ is the set of all resources, $\mathcal{B}$ is the set of all blank nodes, $\mathcal{P}$ is the set of all predicates, and $\mathcal{L}$ the set of all literals.*

Definition 2 (Blank Radius). *The blank radius of a blank node is the minimum number of triples we need to traverse to find a non-blank node. For example, if a blank node is directly connected to a non-blank node, the blank radius is 1. If the blank node is only connected to other blank nodes, but at least one of them is connected to a non-blank*

node, the blank radius is 2. The blank radius of a graph is the maximum blank radius of all blank nodes in the graph.

Definition 3 (Blank node component (BComponent)). *Each blank node is part of exactly one BComponent which is defined as the maximal subgraph that contains the blank node and consists only of triples containing blank nodes as subject and object (as defined in [18]).*

Definition 4 (Graph Mapping). *A mapping between two graphs $G(\mathcal{R}, \mathcal{B}, \mathcal{P}, \mathcal{L})$, and $G'(\mathcal{R}', \mathcal{B}', \mathcal{P}', \mathcal{L}')$ is a function M that maps all resources, literals, predicates, and blank nodes from G to G'. Formally $M : (\mathcal{R} \times \mathcal{B} \times \mathcal{P} \times \mathcal{L}) \rightarrow (\mathcal{R}' \times \mathcal{B}' \times \mathcal{P}' \times \mathcal{L}')$.*

Mapping named resources, predicates, and literals is a straightforward task as only the URI or literal is to be matched. On the other hand, matching blank nodes is a non-trivial task as blank nodes do not always have a unique injective property and two blank nodes may have the same connections to other nodes making it impossible to differentiate them.

Definition 5 (Isomorphism). *Two graphs G, and G' are isomorphic, iff there exists a bijective mapping function M between them, so that $(s, p, o) \in G \iff (M(s), M(p), M(o)) \in G'$.*

Definition 6 (Graph Delta). *The difference between two graphs G, G', denoted $\Delta(G, G')$, is defined as the number of triples that need to be added or removed to make the graphs isomorphic. $\Delta(G, G')$ is computed as the number of triples that need to be added to G and G' so that G and G' are isomorphic. Formally, $\Delta(G, G') = |\{(s, p, o) \in G \wedge (s, p, o) \notin G'\}| + |\{(s, p, o) \in G' \wedge (s, p, o) \notin G\}|$.*

While the previous definition defines the number of different triples between two graphs, which cannot be 0 for different graphs, as blank nodes are only named locally and differ between different documents [31], we define a similar function Δ_M to measure the quality of a mapping function M. Δ_M measures the graph delta between two graphs after a mapping M is applied to one of them that renames blank nodes.

Definition 7 (Mapping Delta). $\Delta_M(G, G') = |\{(s, p, o) \in G \wedge (M(s), M(p), (o)) \notin G'\}| + |\{(M(s)^{-1}, M(p)^{-1}, M(o)^{-1}) \in G' \wedge (s, p, o) \notin G\}|$.

To evaluate the mapping functions and measure their quality relatively to the optimal mapping function achieving the lowest possible value of Δ_M in cases where the graphs are not isomorphic, we define *the Optimal Mapping Deviation Ratio (OMDR)* as follows:

Definition 8 (Optimal Mapping Deviation Ratio (OMDR)). *Let G and G' be two non-isomorphic graphs: $OMDR_M(G, G') = \dfrac{\Delta_M(G, G')}{\Delta_{optimal}(G, G')}$.*

As we require that the graphs are non-isomorphic, the function is well-defined. If the mapping given to the function is optimal, i.e., needs the same number of triple additions and deletions to make the graphs isomorphic as the optimal mapping, the

OMDR is 1. In the case where the mapping M leads to a value of Δ_M twice as high as that of the optimal mapping, the OMDR will be 2. Mapping algorithms therefore try to minimize their Δ_M to have an OMDR as close to 1 as possible. Two different non-optimal mappings can be compared by using the ratio of their respective OMDRs.

The two main problems we solve with our approach are the *isomorphism task* and the *differential task*, as defined in [18]. For solving the *isomorphism task*, two isomorphic graphs are given as input, such that there exists a mapping leading to $\Delta_M = 0$. The RDF triples are not given in the same order to prevent blank node matching algorithms from using the order of the two graphs as a criterion for matching the graphs. The isomorphism task does not aim to find out whether two graphs are isomorphic efficiently, as studied in [10], but we aim here to find a mapping for cases where the isomorphism of two graphs is already known.

The main goal of the *differential task* is to map two different graphs or two different versions of the same graph to each other while minimizing the Δ between the two graphs, where $\Delta \neq 0$. We want to minimize the Δ_M, i.e., the amount of required triple additions and deletions, to make the graphs as isomorphic as possible.

3 Related Work

The blank node matching problem can be interpreted as an assignment problem that can be optimized using the *Hungarian* algorithm [23]. The approach proposed in [30] is able to find the optimal solution in a case when there is no triple containing blank nodes as both subject and object, but cannot correctly handle graphs with multiple connected blank nodes. Other algorithms to match blank nodes focus on computing a signature value for each blank node based on the triples they are part of and then map the blank nodes with closest signatures to each other [4]. *Tzitzikas et al.* [30] propose the SIGN algorithm that computes the signature based on the `rdf:type` value for the blank node, the predicate and subject of incoming triples, and the predicate and object of outgoing triples. *Lantzaki et al.* [18] extend the SIGN approach with the R-SIGN algorithm that includes the order of neighboring blank nodes into the signature. As the R-SIGN approach works with a variable radius it is able to match blank nodes far away from any non-blank nodes. *Oraskari and Törmä* [26] also build upon the SIGN algorithm by iteratively recomputing the signatures to include nodes further away in the signature computation if the current signature is not unique. Hogan [13, 14] creates canonical labels for blank nodes based on a coloring of blank nodes assigned by hashing the neighborhood of them to decide if two graphs are isomorphic. *Lee et al.* [20] use the *MapReduce* process to match blank nodes if they have a high similarity score based on a high amount of similar triples with a configurable threshold.

While our approach focuses on matching blank nodes between different graphs, there are also methods for linking named entities across different graphs. *Trivedi et al.* [28] propose LinkNBed where they learn representations of entities and relations across multiple graphs and also take duplicates into account by utilizing the embedding, the neighborhood and attributes of an entity. *Buneman* and *Staworko* try to solve the graph alignment problem using bisimulation [6] using different alignment methods like similarity alignment and overlap alignment.

Another related research area to our work is that of *Knowledge graph embedding (KGE)* algorithms that can be used to create latent space vector representations of nodes. In recent years, dozens of such KGE techniques have been developed, we list here only a few of them. RESCAL [25] computes a three-way factorization of an adjacency tensor representing the input KG. The adjacency tensor is then decomposed into a product of a core tensor and embedding matrices. HolE [24] uses circular correlation as its compositional operator. ComplEx [29] is a *KGE* model using latent factorization, in which complex valued embeddings are utilized for handling a large variety of binary relations including symmetric and anti-symmetric relations. On the other hand, TransE [5] is an energy-based KGE model, in which a relation r between two entities h and t corresponds to a translation of their embeddings. More details on knowledge graph embedding approaches and applications can be found in [32] and [7]. These and other embedding models can be used for computing the embedding vectors for our approach.

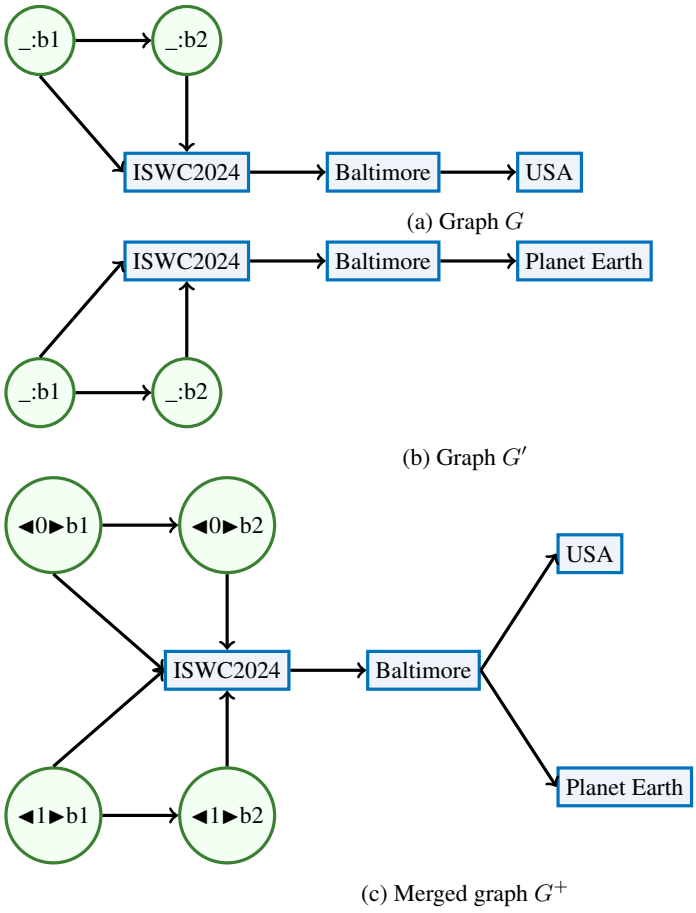
4 Approach

BLINK takes two graphs G and G' as input and creates a mapping M from G to G' minimizing $\Delta_M(G, G')$. The key idea of BLINK is to compute an embedding for the union of both graphs G^+ and match the blank nodes that are close in the latent embedding vector space. We divide our approach into three parts: First, we propose a method to match G and G' into a single graph G^+ in Section 4.1. In Section 4.2, we then give an overview of the embedding model we use. Finally in Section 4.3, we present our algorithm BLINK to compute a mapping from the embedding vectors of G^+.

4.1 Preprocessing / Graph Merging

We start our approach by merging the two input KGs into one with the following procedure. We insert every triples that does not involve blank nodes from both G and G' into G^+. To be able to distinguish from which graph a blank node originated, we extend the local name of each blank node in its originating graph to create a globally unique name. The global name consists of a symbol ◀ that is not contained in any URI in the graphs, then a 0 if the blank node originates from G or a 1 if the blank node originates from G', another ▶ to terminate the global identifier part and concatenate the locally unique name the blank node had in the graph it originates from. We then add the triples with the renamed blank nodes to G^+. Note that BLINK does not rely on keeping the local name the same but is only kept for easier evaluation. Instead, the local names could also be mapped to random strings without impacting the performance of BLINK. We only need to distinguish from which graph the node came from and to distinguish the blank node from other blank nodes coming from the same graph.

The merged model G^+ now contains all the necessary information for the embedding algorithm. An example of this algorithm is shown in Figure 1. Figures 1a and 1b contain the two graphs G and G' respectively while Figure 1c contains the merged graph. Note that because of the nature of the construction of G^+, there are no direct connections from any blank node of G to any blank node of G'. By merging all triples containing only named resources, we share knowledge between the two graphs enabling



(a) Graph G

(b) Graph G'

(c) Merged graph G^+

Fig. 1: An example of how our preprocessing would merge two graphs into one.

the embedding process to take more training data into account for computing the latent space vectors of named resources and thereby making them more stable against being pushed away by connections to blank nodes from different graphs.

4.2 Embedding

Our approach utilizes a translation-based embedding model based on TransE [5]. TransE aims to make the sum of the head entity embedding and relation embedding equal to the tail entity embedding as shown in Equation 1, with γ being a margin parameter, σ being the sigmoid function, and $\text{emb}(x)$ returning the embedding vector of x. We use the default $\gamma = 4$ for our approach as implemented within the DICE embeddings framework [8]. Our matching is based on the fact that similar nodes have similar latent space embedding vectors. To not lose any information, we also include literals in the

embedding process.

$$\hat{y}_i(\text{head, rel, tail}) = \sigma(\gamma - \|\text{emb(head)} + \text{emb(rel)} - \text{emb(tail)}\|_2) \tag{1}$$

The main challenge to overcome is the fact that we do not have any connections between any blank node from G to any blank node from G'. Going with our example in Figure 1c, there is no connection from ◀0▶b1 to ◀1▶b2 and no connection from ◀1▶b1 to ◀0▶b2 even though _:b1 from G and _:b1 from G' as well as _:b2 from G and _:b2 from G' are semantically equivalent. This would lead the embedding process to push the latent space vectors of ◀0▶b2 and ◀1▶b2 away from each other as there is a connection from ◀0▶b1 to ◀0▶b2 but not from ◀0▶b1 to ◀1▶b2.

Therefore we have to adapt the training process and change the way the target value is decided.

Normally, the neural network should handle each input triple x_i=(head, rel, tail) in such a way that $\hat{y}_i$=1 if the triple exists in the graph and $\hat{y}_i$=0 if the triple does not exist in the graph. But this would not be the case with our previously defined challenge. Therefore, we have changed the target function as shown in Equation 2.

$$y_i(\text{head, rel, tail}) = \begin{cases} 1 & :(\text{head, rel, tail}) \in G^+ \\ 0.5 & :\text{head and tail are both blank nodes} \wedge \text{head} \in G \wedge \text{tail} \in G' \\ 0.5 & :\text{head and tail are both blank nodes} \wedge \text{head} \in G' \wedge \text{tail} \in G \\ 0 & :\text{otherwise} \end{cases} \tag{2}$$

We then adapt the binary cross entropy (BCE) loss function to ignore the triples with y_i=0.5 in the loss function computation by multiplying the result of the BCE loss function with a term to ignore said triples as shown in Equation 3. These two changes result in the model ignoring any triple that contains blank nodes originating from two different graphs preventing the embedding vectors of similar blank nodes from being pushed away from each other.

$$L(y_i, \hat{y}_i) = -\sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \cdot ((y_i - 0.5)^2 \cdot 4) \tag{3}$$

4.3 Matching

Let B be the set of all blank nodes in G and B' be the set of all blank nodes in G'. We then compute the pairwise *Euclidean distance* between the embedding vectors of all blank nodes within B and all embedding vectors of all blank nodes within B' into the distance matrix D with $D_{i,j}$ being the Euclidean distance between the embedding vectors of B_i and B'_j. Let $M = \emptyset$ be the resulting mapping of the algorithm. Then our matching algorithm proceeds to match the closest blank nodes until there are no unmatched blank nodes left in B or B' as shown in Algorithm 1. By using this algorithm we get a one-to-one mapping for all blank nodes mapping the most similar blank nodes to each other if $|B| = |B'|$. If the number of blank nodes is not the same in both graphs then this algorithm computes the most probable $\min(|B|, |B'|)$ matches and leaves the

Algorithm 1 MatchBlankNodes(B, B', D)

Input: The set of blank nodes of both graphs B and B' and a matrix D containing the pairwise euclidean distances of all embedding vectors of the nodes of B to the nodes of B'
Output: A mapping M between the blank nodes
while $B \neq \emptyset \wedge B' \neq \emptyset$ **do**
  $(i, j) \leftarrow \arg\min_{(i,j)} D_{i,j}$
  $M \leftarrow M \cup (B_i, B'_j)$
  $B \leftarrow B - \{B_i\}$
  $B' \leftarrow B' - \{B'_j\}$
  $D'_{k,l} \leftarrow D_{k,l} \quad \forall \quad (k,l) \quad 1 \leq k < i \wedge 1 \leq l < j$
  $D'_{k,l} \leftarrow D_{k+1,l} \quad \forall \quad (k,l) \quad i < k \leq |B| \wedge 1 \leq l < j$
  $D'_{k,l} \leftarrow D_{k,l+1} \quad \forall \quad (k,l) \quad 1 \leq k < i \wedge j < l \leq |B'|$
  $D'_{k,l} \leftarrow D_{k+1,l+1} \quad \forall \quad (k,l) \quad i < k \leq |B| \wedge j < l \leq |B'|$
  $D \leftarrow D'$
end while
return M

remaining nodes without a matched blank nodes from the other graph. By removing already matched blank nodes from the yet-to-be-matched nodes and the distance matrix, we ensure that we do not run into problems if multiple blank nodes have the same connections to other nodes and therefore very similar embedding vectors. Otherwise, it would be possible for multiple blank nodes of one graph to match with a single blank node from the other graph which would leave some blank nodes of the other graph without a matching blank node.

5 Evaluation & Results

To evaluate our approach, we conducted experiments using real-world KGs. We compare the results of our approach with multiple different numbers of training epochs to current state-of-the-art algorithms described in Section 3.

Research questions. We aim to answer the following research questions:

Q_1. Does the accuracy of BLINK increase with a larger number of training epochs?
Q_2. How does a *higher number of BComponents* affect BLINK for the isomorphism task?
Q_3. Is BLINK able to match blank nodes correctly for *larger BComponents* coming from isomorphic graphs?
Q_4. How is the accuracy impacted by adding triples that contain existing blank nodes as subjects and random strings as predicates and objects?
Q_5. Are the approaches robust concerning added triples with blank nodes as subjects and random known relations and entities from the graph as predicates and objects?
Q_6. What effect does *splitting up BComponents* have on the accuracy of the approaches?

Dataset	|Triples|	|BNodes|	|BBNodes|	|BTriples|	|BBTriples|	BRadius	|BC|
Military	30000	2827	1452	30000	2749	2	78
CraftNode	29980	4739	2815	29980	3846	2	893
Syn$_l$	$\sum_{i=1}^{l} 4 \cdot 4^i$	$\sum_{i=1}^{l} 4^{i-1}$	$\sum_{i=1}^{l} 4^{i-2}$	$\sum_{i=1}^{l} 4 \cdot 4^i$	$\sum_{i=1}^{l} 4{\cdot}4^{i-1}$	l	1

Table 1: Dataset statistics for the used datasets showing in order: number of triples, number of blank nodes, number of blank nodes only connected to other blank nodes, number of triples containing at least one blank node, number of triples containing two blank nodes, the blank radius, and the number of BComponents of the graph. The *CraftNode* dataset consists of only 29980 triples because there are 20 duplicate triples in the last 30000 lines of the dataset.

Setup. We present the results for BLINK using $\{16, 32, 64, 128, 256\}$ training epochs dubbed BLINK $_{[\#epochs]}$. The embedding and matching of our approach use the DICE embeddings library[1] [8] .We use an embedding dimension of 32 and a batch size of 1024 for our experiments. To make the results reproducible, we use 1 as a seed for random number computation wherever applicable. As a baseline or matching quality, we use the signature algorithm [30] extended by [18], dubbed SIGN, and the radius-based signature algorithms [18] with $\{2, 3, 1000\}$ as parameters for the radius dubbed R-SIGN $_{[radius]}$. We also report the result for not mapping any blank nodes as NO MAPPING and the best achieveable Δ_M dubbed OPTIMAL MAPPING.

Datasets. The first and second datasets we use for our experiment are subsets of LINKEDGEODATA [2]. In particular, we use the last 30000 lines of the *MilitaryThingNode* and *CraftNode* subset of LINKEDGEODATA because of the high number of triples containing blank nodes. Statistics regarding the datasets are listed in Table 1. As a third dataset, we use a synthetic dataset dubbed Syn_l with l being the number of layers generated by us representing one big BComponent created by starting with a single blank node and subsequently adding a layer of $4^{\text{layer index}}$ blank nodes around it and therefore increasing the blank radius of the graph by one. All blank nodes, except for the first one, have 4 random connections to blank nodes of the previous layer. The outermost layer consists of 4^{l+1} named resources again randomly connected with 4 links to blank nodes of the previous layer.

5.1 Isomorphism Mapping

As input for the *isomorphism task*, we created two copies of the same dataset, where we changed the order of one of them to prevent the algorithms from making decisions based on the order of triples. As both dataset instances contain the exact same triples, the optimal solution for matching the blank nodes results in $\Delta_M{=}0$.

Q1: Accuracy over different numbers of training epochs. The results for the *MilitaryThingNode* dataset are shown in Figure 2. As expected, the Δ_M is 60000 if no blank

[1] version 0.0.5 `https://pypi.org/project/dicee/0.0.5/`

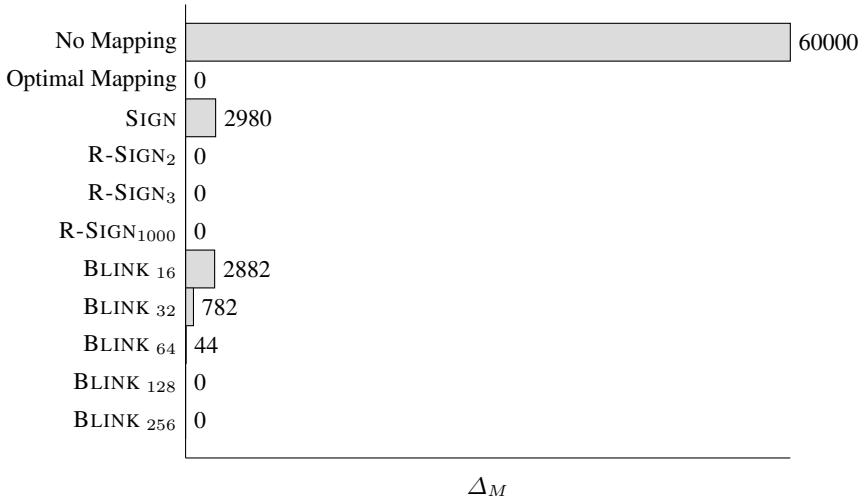


Fig. 2: Δ_M for the isomorphism task using the *MilitaryThingNode* dataset for the blank node matching algorithms from Section 5.

nodes are matched because every triple in the dataset contains at least one blank node. Therefore, no triple from one graph is equal to any triple in the other graph and all triples from one graph have to be removed and all triples from the other graph have to be added to make both graphs equal. The SIGN algorithm can reduce the Δ_M to 2980 while the R-SIGN algorithm obtains an optimal result for all radii. This result is expected, as the R-SIGN algorithm is proven to produce the optimal result for the *isomorphism task* in case the radius parameter is at least as big as the blank radius of the graph[18]. The mapping of BLINK achieves a Δ_M of 2882, 782, and 44 for 16, 32, and 64 training epochs, respectively. Starting at 128 training epochs, BLINK produces a perfect mapping. This answers our first research question, Q_1, indicating that our embedding process indeed puts similar blank nodes closer to each other in the embedding space the more epochs we train. Accordingly, the quality of the mapping produced by BLINK increases with an increasing number of training epochs.

Q2: Higher number of BComponents. The mapping Δ for the *CraftNode* dataset is shown in Figure 3. The SIGN algorithm can match the graphs up to a Δ_M of 7174 while the R-SIGN algorithms can find a perfect mapping as the blank radius of this radius is only 2. BLINK achieves a Δ_M of 9496 and 4658 for 16 and 32 epochs respectively. Starting at 64 epochs, BLINK produces an optimal mapping, which again confirms our answer for Q_1. The number of blank nodes that are only connected to other blank nodes and especially the number of BComponents in this dataset are larger than in the *MilitaryThingNode* dataset which shows that BLINK is also able to handle datasets with less known URIs and more possibilities for incorrect matches. This answers our research question Q_2.

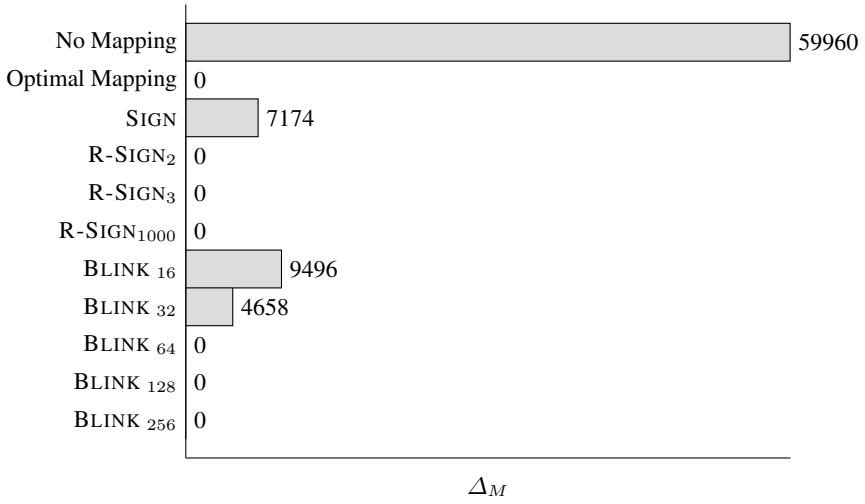


Fig. 3: Δ_M for the isomorphism task using the *CraftNode* dataset for the blank node matching algorithms from Section 5.

Q3: Large BComponents. To answer our research question Q_3, we conducted an experiment with the Syn$_l$ dataset with l=$\{1, 2, 3, 4, 5, 6, 7\}$ to find out how well BLINK can handle different BComponent sizes. The results are presented as a heatmap in Figure 4. After only 16 training epochs, BLINK can find a perfect matching for the synthetic datasets up to a radius of 5. For a radius of 6, BLINK requires more training epochs as there is a small Δ_M for 16 epochs but it can produce a perfect result after 32 epochs. The task gets harder when the blank radius is 7, but even then BLINK can compute an optimal mapping after 64 epochs. This answers our research question Q_3. Interestingly, even the signature and radius-based signature algorithms with a radius smaller than the blank radius achieve perfect accuracy. This happens because the extended version of the signature algorithm [18] creates deterministic counters for adjacent blank nodes enhancing the performance massively for isomorphic graphs in comparison to the default SIGN algorithm [30].

5.2 Differential Mapping

For the *differential task*, we focus on how well the approaches perform when changing the dataset. We simulate different versions of a dataset by adding and removing triples from one graph instance. In this case, the optimal mapping solution would have a Δ_M equal to the number of added and removed triples. We evaluate the results of this set of experiments using the *Optimal Mapping Deviation Ratio* (OMDR) as defined in Definition 8. As the OMDR depends on the Δ_M of the optimal mapping as its denominator, the OMDR can increase for a higher number of changed triples, because the optimal mapping also has a higher Δ even though the absolute number of triple changes increases. The optimal Δ for this task is determined by the number of triples added or

	Syn_1	Syn_2	Syn_3	Syn_4	Syn_5	Syn_6	Syn_7
Optimal Mapping	0	0	0	0	0	0	0
No Mapping	32	160	672	2720	10912	43680	174752
SIGN	0	0	0	0	0	0	0
R-$SIGN_2$	0	0	0	0	0	0	0
R-$SIGN_3$	0	0	0	0	0	0	0
R-$SIGN_{1000}$	0	0	0	0	0	0	0
BLINK $_{16}$	0	0	0	0	0	84	28954
BLINK $_{32}$	0	0	0	0	0	0	758
BLINK $_{64}$	0	0	0	0	0	0	0
BLINK $_{128}$	0	0	0	0	0	0	0
BLINK $_{256}$	0	0	0	0	0	0	0

Fig. 4: Δ_M for the isomorphism task using synthetic datasets.

removed from the dataset instance we change. Interestingly, 1% of added triples would result in an optimal Δ of 300, while 10% would result in Δ=3000. Therefore, the reported numbers in Figures 5, 6, and 7 are not to be compared over different columns because the OMDR base is different in each column. A higher OMDR in column c_1 is not worse than the OMDR in another column c_2 if the optimal Δ is smaller in column c_1.

Q4: Adding triples with new objects. For the first experiment of this task, we use the *MilitaryThingNode* dataset and add new triples with existing subject URIs, random predicate URIs, and random object URIs to one instance of the dataset. We want to evaluate how the algorithms can deal with new additions to the dataset and at what percentage of new triples the performance starts to drop. We therefore conduct this experiment with $\{2, 4, 6, 8, 10, 20, 50\}\%$ of added triples. Figure 5 shows the results of the experiment. Even for 2% of added triples, no signature-based algorithm produces a mapping with less than 10 times more needed changes to make the graphs isomorphic in comparison to the optimal mapping. BLINK, however, is able to compute a perfect result after 128 training epochs, mapping no blank node incorrectly to its equivalent blank node in the other graph. Even for only 16 training epochs, BLINK only has an optimal mapping deviation of almost half of the signature algorithms. These results show that BLINK can quickly produce results outperforming the state of the art and slowly converge to the smallest Δ_M achievable for the given dataset.

Q5: Adding triples with existing objects. For the second experiment of the *differential task*, we aim to answer research question Q_5. We start with the *MilitaryThingNode* dataset, but instead of adding triples with random predicates and objects, we add triples with already existing predicates and objects. We want to find out if BLINK is also able to compute good mappings between the two graphs if not only new entities are added and linked to existing entities but existing entities are linked to other already existing

	2%	4%	6%	8%	10%	20%	50%
Optimal Mapping	1.000	1.000	1.000	1.000	1.000	1.000	1.000
No Mapping	101.000	51.000	34.333	26.000	21.000	11.000	5.000
SIGN	14.333	11.848	10.720	9.514	9.457	7.128	4.129
R-SIGN$_2$	10.743	10.515	10.048	9.178	9.241	7.084	4.120
R-SIGN$_3$	10.743	10.515	10.048	9.178	9.241	7.084	4.120
R-SIGN$_{1000}$	10.743	10.515	10.048	9.178	9.241	7.084	4.120
BLINK $_{16}$	5.780	3.413	2.620	2.228	1.993	1.494	1.321
BLINK $_{32}$	2.870	2.253	1.941	1.811	1.709	1.441	1.195
BLINK $_{64}$	1.067	1.065	1.068	1.068	1.041	1.055	1.143
BLINK $_{128}$	1.000	1.000	1.000	1.082	1.001	1.002	1.019
BLINK $_{256}$	1.000	1.000	1.000	1.020	1.000	1.000	1.002

Fig. 5: Optimal Mapping Deviation Ratios for the differential task with random triples added to the *MilitaryThingNode* dataset. The different columns show the Δ for different percentages of added triples.

entities. This is important because in real-world applications new links between existing entities can be added to complete a knowledge graph as in link prediction [34] or through manual completion of the knowledge graph. We conduct the experiment with $\{1, 2, 4, 6, 8, 10, 20\}$ % of added triples. The results of this experiment are shown in Figure 6. For a small amount of 1% added triples, all algorithms are able to generate mapping for some blank node. However, the SIGN algorithm still only achieves an OMDR of 19.26. The radius-based sign algorithms all achieve the same result again as the radius of the dataset remains 2. They achieve an OMDR of 11.193. BLINK produces a more accurate mapping at just 16 epochs of training with an Optimal Mapping Deviation Ratio of 11.907, i.e., almost twice as accurate as the R-SIGN algorithms. With a growing number of epochs, BLINK can produce a result that is even more accurate with an OMDR of 3.72. For a higher number of added triples (10%), all algorithms are closer to the number of changes required with the optimal mapping. Not mapping any blank nodes results in an OMDR of 21 which is surpassed by the SIGN algorithm with a deviation ratio of 9.858. The R-SIGN algorithms only marginally beat the SIGN algorithm in accuracy with a deviation ratio of 9.749. BLINK, however, is still able to produce a nearly accurate mapping with a deviation ratio of 1.977 for 16 epochs and 1.286 for 256 training epochs. This result indicates that BLINK can correctly match blank nodes even if the dataset gets altered by adding more links between the existing resources in the dataset. BLINK is more robust against changes than the SIGN and R-SIGN algorithms for different amounts of added triples and outperforms their mapping for all numbers of training epochs we tested. Furthermore, training of BLINK would lead to an even smaller Δ_M computed by BLINK allowing the user to choose how much time should be invested into finding a high-quality mapping by increasing or lowering the number of training epochs. This answers our research question Q_5.

	1%	2%	4%	6%	8%	10%	20%
Optimal Mapping	1.000	1.000	1.000	1.000	1.000	1.000	1.000
No Mapping	201.000	101.000	51.000	34.352	26.000	21.000	11.002
SIGN	19.260	13.827	14.432	11.805	9.707	9.858	7.025
R-SIGN$_2$	11.193	10.543	13.392	11.367	9.463	9.749	6.999
R-SIGN$_3$	11.193	10.543	13.392	11.367	9.463	9.749	6.999
R-SIGN$_{1000}$	11.193	10.543	13.392	11.367	9.463	9.749	6.999
BLINK $_{16}$	11.907	5.730	3.380	2.596	2.213	1.977	1.487
BLINK $_{32}$	6.787	2.817	2.210	1.953	1.778	1.637	1.403
BLINK $_{64}$	3.613	1.390	1.442	1.447	1.422	1.406	1.318
BLINK $_{128}$	3.807	1.250	1.295	1.331	1.346	1.329	1.292
BLINK $_{256}$	3.720	1.163	1.220	1.251	1.310	1.286	1.278

Fig. 6: Optimal Mapping Deviation Ratios for the differential task with triples containing known predicates and objects added to the *MilitaryThingNode* dataset. The different columns show the deviation for different percentages of added triples.

Q6: Removing triples. For the third experiment of the *differential task* task, we removed a different number of triples from the dataset. To prevent completely removing blank nodes, we made sure that each blank node is still contained in at least one triple. We conducted the experiment with $\{1, 2, 4, 6, 8, 10, 20\}$ % of removed triples to determine how many triples are necessary for BLINK and the other algorithms to work and at what percentage of triple removal the performance starts to drop. Because of the triple removal, some of the BComponents get split up which poses an extra challenge for the signature-based algorithms. The amount of BComponents is shown in Table 2.

The results of this experiment are shown in Figure 7. Like the last experiment, our results show that BLINK outperforms all signature-based algorithms. The OMDR of BLINK is less than half of the OMDR of all other algorithms for all tested percentages of removed triples. Even when removing 20% of triples from the graph, BLINK computes a mapping with only a 0.3% higher Δ_M than the optimal mapping. Meanwhile, the signature algorithms only achieve a 627.6% higher Δ_M than the optimal possible mapping. The OMDR for BLINK stays almost constant for a different number of removed triples (and therefore split up BComponents) with OMDRs between 1.000 and 1.008 for 256 training epochs. This answers our last research question Q_6. Our results

Removed Triples	0%	1%	2%	4%	6%	8%	10%	20%
BComponents	78	99	107	133	156	197	215	354

Table 2: The number of BComponents for the dataset versions with different percentages of triples removed.

	1%	2%	4%	6%	8%	10%	20%
Optimal Mapping	1.000	1.000	1.000	1.000	1.000	1.000	1.000
No Mapping	199.000	99.000	49.000	32.333	24.000	19.000	9.000
SIGN	31.913	22.043	16.250	13.323	11.868	10.077	6.285
R-SIGN$_2$	23.920	18.833	15.232	12.917	11.675	9.970	6.276
R-SIGN$_3$	23.920	18.833	15.232	12.917	11.675	9.970	6.276
R-SIGN$_{1000}$	23.920	18.833	15.232	12.917	11.675	9.970	6.276
BLINK $_{16}$	10.493	5.760	3.412	2.597	2.192	1.956	1.494
BLINK $_{32}$	3.653	2.367	1.670	1.481	1.337	1.261	1.175
BLINK $_{64}$	1.107	1.107	1.032	1.024	1.019	1.008	1.012
BLINK $_{128}$	1.000	1.000	1.000	1.006	1.004	1.000	1.003
BLINK $_{256}$	1.000	1.000	1.008	1.000	1.000	1.003	1.003

Fig. 7: Optimal Mapping Deviation Ratios for the differential task with random triples removed from the *MilitaryThingNode* dataset. The different columns show the deviation for different percentages of removed triples.

indicate that BLINK is robust against splitting up BComponents as well as overall missing triples in the graph.

6 Conclusion & Future Work

In this paper, we propose BLINK, an approach for matching blank nodes that works by merging two graphs, embedding the merged graph into a latent vector space, and subsequently matching the blank nodes that are closest to each other in the embedding space. The experiments regarding the *isomorphism task* show that BLINK's accuracy is up to par with current state-of-the-art algorithms, as it also finds the optimal mapping solution after a sufficient number of epochs. For the *differential task*, our finding is that BLINK significantly outperforms other algorithms and in most cases computes a solution that is very close to the optimal mapping.

Further studies need to be carried out to assess the impact of using different embedding models and loss functions on the accuracy of the mapping. It is also worth investigating whether removing a part of the triples based on a low gain of information, similar to the process described in [17], results in a better runtime/accuracy tradeoff than using all triples for the embedding vector computation. It might also be beneficial to combine BLINK with signature-based approaches to reduce the time it takes to compute a matching. For example, blank nodes that have BComponent matches with the other graph can be computed quickly by using R-SIGN [18]. BLINK could then be used to match blank nodes that do not have an equivalent BComponent in the other graph.

Supplemental Material Statement: Source code for merging graphs and signature-based algorithms as well as the delta computation and the source code for the embedding

process, training files, datasets, blank node mappings, created models, and a CSV of computed embedding vectors for each entity are available from our Github[2].

Acknowledgements: This work has been supported by the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within the project SAIL under the grant no NW21-059D. This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070305.

Bibliography

[1] Auer, S., Herre, H.: A versioning and evolution framework for rdf knowledge bases. In: Virbitskaite, I., Voronkov, A. (eds.) Perspectives of Systems Informatics. pp. 55–69. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[2] Auer, S., Lehmann, J., Hellmann, S.: Linkedgeodata: Adding a spatial dimension to the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009. pp. 731–746. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[3] Berners-Lee, T.: Linked-data design issues. W3C design issue document (June 2009), https://www.w3.org/DesignIssues/LinkedData.html

[4] Berners-Lee, T., Connolly, D.: Delta: an ontology for the distribution of differences between rdf graphs. World Wide Web, http://www. w3. org/DesignIssues/Diff **4**(3), 4–3 (2004)

[5] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 26. Curran Associates, Inc. (2013), https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf

[6] Buneman, P., Staworko, S.: Rdf graph alignment with bisimulation. Proc. VLDB Endow. **9**(12), 1149–1160 (aug 2016). https://doi.org/10.14778/2994509.2994531, https://doi.org/10.14778/2994509.2994531

[7] Cao, J., Fang, J., Meng, Z., Liang, S.: Knowledge graph embedding: A survey from the perspective of representation spaces. ACM Comput. Surv. **56**(6) (mar 2024). https://doi.org/10.1145/3643806

[8] Demir, C., Ngomo, A.C.N.: Hardware-agnostic computation for large-scale knowledge graph embeddings. Software Impacts (2022)

[9] Fiorelli, M., Pazienza, M.T., Stellato, A., Turbati, A.: Version control and change validation for rdf datasets. In: Garoufallou, E., Virkus, S., Siatri, R., Koutsomiha, D. (eds.) Metadata and Semantic Research. pp. 3–14. Springer International Publishing, Cham (2017)

[10] Grohe, M., Schweitzer, P.: The graph isomorphism problem. Communications of the ACM **63**(11), 128–134 (2020)

[2] https://github.com/becker-al/BLINK

[11] Haerder, T., Reuter, A.: Principles of transaction-oriented database recovery. ACM computing surveys (CSUR) **15**(4), 287–317 (1983)
[12] Harth, A.: Billion Triples Challenge data set. Downloaded from http://km.aifb.kit.edu/projects/btc-2012/ (2012)
[13] Hogan, A.: Skolemising blank nodes while preserving isomorphism. In: Proceedings of the 24th International Conference on World Wide Web. pp. 430–440 (2015)
[14] Hogan, A.: Canonical forms for isomorphic and equivalent rdf graphs: Algorithms for leaning and labelling blank nodes. ACM Trans. Web **11**(4) (jul 2017). https://doi.org/10.1145/3068333
[15] Hogan, A., Arenas, M., Mallea, A., Polleres, A.: Everything you always wanted to know about blank nodes. Journal of Web Semantics **27-28**, 42–69 (2014). https://doi.org/https://doi.org/10.1016/j.websem.2014.06.004, https://www.sciencedirect.com/science/article/pii/S1570826814000481, semantic Web Challenge 2013
[16] Isaac, A., Summers, E.: SKOS simple knowledge organization system primer. W3C note, W3C (Aug 2009), https://www.w3.org/TR/2009/NOTE-skos-primer-20090818/
[17] Karegowda, A.G., Manjunath, A., Jayaram, M.: Comparative study of attribute selection using gain ratio and correlation based feature selection. International Journal of Information Technology and Knowledge Management **2**(2), 271–277 (2010)
[18] Lantzaki, C., Papadakos, P., Analyti, A., Tzitzikas, Y.: Radius-aware approximate blank node matching using signatures. Knowledge and Information Systems **50** (02 2017). https://doi.org/10.1007/s10115-016-0945-9
[19] Lantzaki, C., Tzitzikas, Y.: Tasks that require, or can benefit from, matching blank nodes. CoRR **abs/1410.8536** (2014), http://arxiv.org/abs/1410.8536
[20] Lee, T., Im, D.H., Won, J.: Similarity-based change detection for rdf in mapreduce. Procedia Computer Science **91**, 789–797 (2016)
[21] Mallea, A., Arenas, M., Hogan, A., Polleres, A.: On blank nodes. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) The Semantic Web – ISWC 2011. pp. 421–437. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
[22] Motik, B., Patel-Schneider, P.: OWL 2 web ontology language mapping to RDF graphs (second edition). W3C recommendation, W3C (Dec 2012), https://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/
[23] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics **5**(1), 32–38 (1957). https://doi.org/10.1137/0105003, https://doi.org/10.1137/0105003
[24] Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 1955–1961. AAAI'16 (2016)
[25] Nickel, M., Tresp, V., Kriegel, H.: A three-way model for collective learning on multi-relational data. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011. pp. 809–816. Omnipress (2011), https://icml.cc/2011/papers/438_icmlpaper.pdf

[26] Oraskari, J., Törmä, S.: Rdf-based signature algorithms for computing differences of ifc models. Automation in Construction **57** (06 2015). https://doi.org/10.1016/j.autcon.2015.05.008

[27] Patel-Schneider, P., Hayes, P.: RDF 1.1 semantics. W3C recommendation, W3C (Feb 2014), https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/

[28] Trivedi, R., Sisman, B., Dong, X.L., Faloutsos, C., Ma, J., Zha, H.: LinkNBed: Multi-graph representation learning with entity linkage. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 252–262. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-1024, https://aclanthology.org/P18-1024

[29] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)

[30] Tzitzikas, Y., Lantzaki, C., Zeginis, D.: Blank node matching and rdf/s comparison functions. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) The Semantic Web – ISWC 2012. pp. 591–607. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[31] Völkel, M., Enguix, C.F., Kruk, S.R., Zhdanova, A.V., Stevens, R., Sure-Vetter, Y.: Semversion - versioning rdf and ontologies. Tech. rep., Universität Karlsruhe (TH) (2005)

[32] Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)

[33] Yang, L., Huang, L., Lu, H., Xu, F.: Removal mechanism of redundant blank nodes in linked data. In: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA). pp. 1118–1121 (2018). https://doi.org/10.1109/ICIEA.2018.8397878

[34] Zamini, M., Reza, H., Rabiei, M.: A review of knowledge graph completion. Information **13**(8), 396 (2022)