axioms

MDPI

Article

Probabilistic Coarsening for Knowledge Graph Embeddings

Marcin Pietrasik [1,*] and Marek Z. Reformat [1,2]

[1] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada
[2] Information Technology Institute, University of Social Sciences, 90-113 Łódź, Poland
* Correspondence: pietrasi@ualberta.ca

Abstract: Knowledge graphs have risen in popularity in recent years, demonstrating their utility in applications across the spectrum of computer science. Finding their embedded representations is thus highly desirable as it makes them easily operated on and reasoned with by machines. With this in mind, we propose a simple meta-strategy for embedding knowledge graphs using probabilistic coarsening. In this approach, a knowledge graph is first coarsened before being embedded by an arbitrary embedding method. The resulting coarse embeddings are then extended down as those of the initial knowledge graph. Although straightforward, this allows for faster training by reducing knowledge graph complexity while revealing its higher-order structures. We demonstrate this empirically on four real-world datasets, which show that coarse embeddings are learned faster and are often of higher quality. We conclude that coarsening is a recommended prepossessing step regardless of the underlying embedding method used.

Keywords: knowledge graph; embedding; coarsening

MSC: 68T30

check for updates

Citation: Pietrasik, M.; Reformat, M.Z. Probabilistic Coarsening for Knowledge Graph Embeddings. *Axioms* **2023**, *12*, 275. https://doi.org/10.3390/axioms12030275

Academic Editors: Cheng-Shian Lin, Chien-Chang Chen and Yi-Hsien Wang

Received: 21 January 2023
Revised: 26 February 2023
Accepted: 1 March 2023
Published: 6 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Knowledge bases have received considerable research attention in recent years, demonstrating their utility in areas ranging from question answering [1,2] to knowledge generation [3–5] to recommender systems [6]. These knowledge bases are underpinned by graph structures called knowledge graphs, which describe facts as a collection of triples that relate two entities via a predicate. Advances in artificial intelligence have spurred the need to find representations of knowledge graphs that can be easily and accurately reasoned with by machines. Perhaps the most common approach to this is knowledge graph embedding, which transforms a knowledge graph into a low-dimensional embedding space, thereby providing a representation that is both intuitive and highly operable. Many [2–5,7,8] knowledge graph embedding methods have been proposed in recent years, each excelling at capturing different aspects of the knowledge graph's structure and semantics. Relatively less work has been performed on methods for preprocessing a knowledge graph prior to embedding. It has been shown on simple graphs, however, that hierarchically coarsening prior to embedding yields higher-quality embeddings [9,10]. This provides the motivation for our work, namely to investigate whether or not such an approach can improve embeddings for knowledge graphs. Specifically, we propose a simple embedding meta-strategy that can be applied to any arbitrary embedding method. Our strategy first reduces an input knowledge graph—henceforth referred to as a base graph—to a coarsened graph, along with a mapping between the entities in each knowledge graph. Coarse embeddings are then learned on the coarse graph and mapped back down as base embeddings. They may then be fine-tuned on the base graph to reintroduce information that was lost in the coarsening procedure. Figure 1 outlines the flow of this approach. The hypothesized rationale for coarsening is multifactorial, as pointed out in [9,10]. Although largely untested empirically in these works, it may be summarized as follows:

- Coarsening reduces knowledge graph size whilst preserving the global structure, potentially revealing higher-order features.
- Training schemes that rely on stochastic gradient descent may learn embeddings that fall in local minima. Initializations learned on the coarse graph may be more resistant to this problem.
- Structurally equivalent entities are embedded jointly in coarse graphs, reducing training complexity.

We evaluate our proposed strategy on the entity classification task using four real-world datasets and perform a pairwise comparison against common embedding methods. The results indicate that embedding on a coarse graph produces faster and, in many cases, higher-quality embeddings.

In summary, the contributions of our paper are as follows. To the best of our knowledge, we are the first to use coarsening as an explicit preprocessing strategy for generating knowledge graph embeddings. To this end, we propose a novel probabilistic coarsening procedure that reduces knowledge graph size while preserving its global structure. The results of our empirical evaluation allow us to conclude that coarsening is a recommended strategy regardless of the underlying embedding method being used.

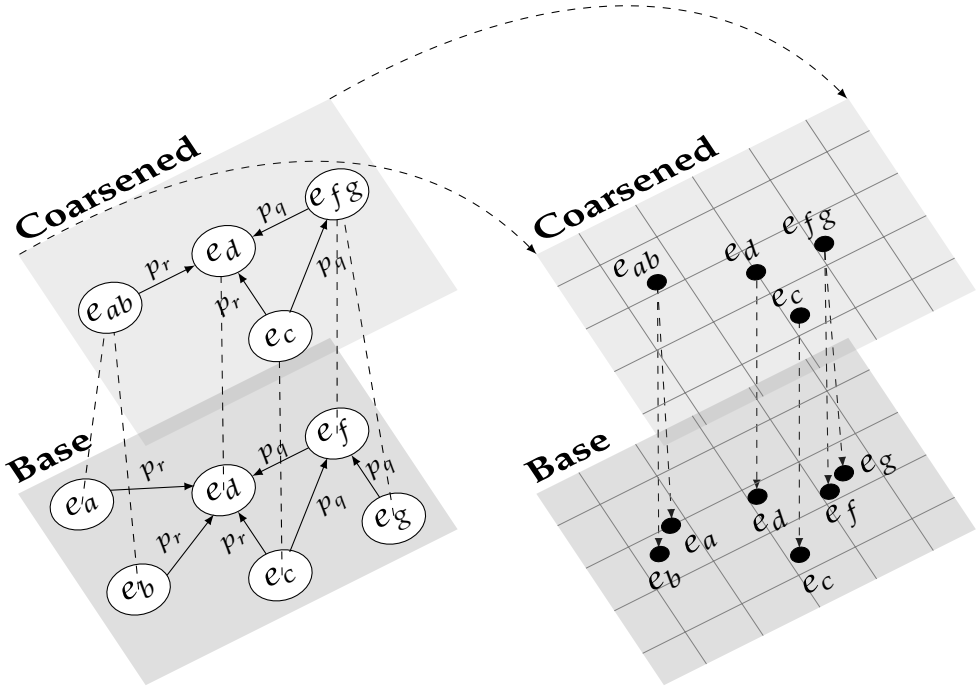

Figure 1. Toy example demonstrating our proposed embedding strategy. The logical flow is guided by dashed line arrows, starting in the bottom left corner and proceeding clockwise.

2. Related Work

Research in graph representation, including graph embeddings, has a long history rooted in mathematics with early methods discussed in [11]. More recently, deep learning has been leveraged for graph embeddings to achieve state-of-the-art results. For instance, DeepWalk [12], Large-scale Information Network Embedding [13], and node2vec [14] sample random walks on a graph and treat them as input words to the skip-gram language model [15]. The intuition behind this approach is that nodes that are sampled in the same random walks are more similar semantically and should have similar embeddings. Another class of deep approaches, Graph Convolution Networks (GCN) [16,17], utilize the convolution operator to learn neighbourhood information for graph entities. Extensions and derivatives of GCNs are ample; for a comprehensive discussion of these methods, we refer readers to [18]. Autoencoders use neural networks to reduce input to a latent

embedding before reconstructing the embedding back to the original input. This approach has shown success in generating graph embeddings in [6,19].

Meta-strategies that preprocess graphs into hierarchies before applying embedding methods have been shown to produce higher-quality embeddings at reduced training complexity. The first of these proposed, HARP [9], sequentially reduces the input graph into a hierarchy of progressively coarser graphs. These coarse graphs are then embedded, starting with the coarsest graph such that their embeddings serve as the initialization of the graph directly below it in the hierarchy. This idea was extended in Multi-Level Embedding [10] and GOSH [20], both of which modify the coarsening procedure. In the case of MILE, a hybrid of Structural Equivalence Matching and Normalized Heavy Edge Matching [21] is utilized for coarsening. Furthermore, GCNs are used to refine learned embeddings when moving down the coarse hierarchy to initialize embeddings.

Knowledge graphs present structural traits that render the aforementioned methods ill-suited for their embedding. Specifically, knowledge graphs are directed and labelled in their relations between entities. In light of this, much research has been devoted to developing methods that account for these complexities. For instance, RDF2Vec [7] uses breadth-first graph walks on the skip-gram model to generate embeddings. This model generates embeddings in which entities that are in lower-order neighbourhoods of one another are more proximal in the embedding space. Being one of the first and most popular embedding approaches, it has received significant research attention and extensions, such as KG2Vec [22], Triple2Vec [23], and RDF2Vec_oa [24]. GCNs have been extended to knowledge graphs with the Relational Graph Convolution Network (R-GCN) [3], which aggregates predicate-specific convolutions of the original model. As with RDF2Vec, its success attracted much attention and many extensions, including the Relational Graph Attention Network [25], which leverages graph attention mechanisms and QA-GNN [26], which incorporates contextual information for question and answering tasks. ConvE [5] also leverages the convolution operator in a neural framework by stacking embeddings as a matrix and convolving them in two dimensions. This approach is capable of learning highly expressive embeddings while achieving computational efficiency [27]. Translation-based methods apply the intuition that subject embeddings should be near object embeddings when translated by valid corresponding predicates. These models were pioneered by TransE [4], which was subsequently improved upon to address its challenges in modelling certain types of relations, such as one-to-many and many-to-one [28–34]. Two of these extensions, TransR [29] and TransH [28], are described later on in the paper. Factorization models, such as RESCAL [8] and DistMult [35], learn embeddings by factorizing the knowledge graph adjacency tensor into the product of entity embeddings and relation-specific translation matrices. In bilinear models, such as the two mentioned, the embedded representation of the relations forms a two-dimensional matrix. Non-bilinear factorization models, such as HolE [36] and TuckER [37], do not share this property. Deep reinforcement learning has also shown promise in this domain with MINERVA [2], which learns knowledge graph paths to find the correct entity in incomplete triples. A recent and comprehensive discussion of knowledge graph embedding methods can be found in [38].

3. Proposed Strategy

We define a knowledge graph, $\mathcal{K}$, as a collection of triples such that each triple relates a subject entity, e_s, to an object entity, e_o, via a predicate, p_r. In this view, $\mathcal{K} := \{(e_s, p_r, e_o)\} \in \mathcal{E} \times \mathcal{P} \times \mathcal{E}$ where $e \in \mathcal{E}$ is the set of entities and $p \in \mathcal{P}$ is the set of predicates. The task of knowledge graph embedding is to find a function, f, which maps each entity to the embedding space $f : \mathcal{E} \mapsto \mathbb{R}^{|\mathcal{E}| \times d}$ where d is the dimensionality of the embedding space such that $d << |\mathcal{E}||\mathcal{P}|$. Most embedding methods also embed predicates, although their formulation is highly variable.

We employ the notation used in [39] wherein a knowledge graph is interpreted by a set of entities and the tags that annotate them. In this view, a tag t, is defined as a predicate-entity pair that describes another entity, $t := (p_r, e_o) \mid (e_s, p_r)$. The order between

a tag's predicate and entity correspond to whether the tag is incoming or outgoing. Thus, each triple in $\mathcal{K}$ corresponds to two entity-tag mappings. These mappings are expressed as sets such that each entity e_a has a corresponding set of tags that annotate it, denoted as $\mathcal{A}_a$.

Our proposed strategy embeds a base graph via an intermediary coarse graph. In this process, the coarse graph is first generated from the base graph before being embedded. Coarse embeddings are then mapped back down to the base graph and fine-tuned. Our strategy may be divided into the following three steps:

1. **Probabilistic graph coarsening** reduces the base graph to a smaller, coarsened graph and returns an entity mapping between the two graphs.
2. **Coarse graph embedding** applies a predetermined embedding method on the coarse graph to obtain coarse embeddings.
3. **Reverse mapping and fine-tuning** maps coarse embeddings back down to the base graph to obtain base embeddings. Base embeddings may be fine-tuned on the base graph.

The remainder of this section describes each of these steps in detail. The intuition for this process is described visually in Figure 1, while its sequence is outlined in Algorithm 1.

3.1. Probabilistic Graph Coarsening

In this step, the base graph is reduced to create a coarse graph, denoted as $\mathcal{K}'$, such that $|\mathcal{K}'| < |\mathcal{K}|$. This procedure involves collapsing structurally similar entities in $\mathcal{K}$ to one entity cluster in $\mathcal{K}'$. Relations in $\mathcal{K}$ are extended to $\mathcal{K}'$ such that a cluster's relations are the union of its constituent entities' relations. Collapsing entities are divided into two stages, designed to preserve the first-order and second-order proximities of the base graph [13]. This allows the base graph to be reduced of structural redundancies, making it more computationally manageable and potentially revealing its global and most salient features. The mapping between base entities and coarse entities is represented by $\Psi : \mathcal{E} \mapsto \mathcal{E}'$. Coarsening is demonstrated visually on the left half of Figure 1.

3.1.1. Collapsing First-Order Neighbours

Preserving first-order proximity refers to the notion that entities should be embedded proximally to their first-order (i.e., one-hop) neighbours. By collapsing entities with their first-order neighbours, proximity is ensured as collapsed entities share identical embeddings. In undirected, single predicate graphs, edge collapsing [9,40,41] finds the largest subset of edges such that no two edges are incident to the same vertex. Vertices incident to each edge in this set is then collapsed, yielding a graph coarsened to preserve first-order proximity. Edge collapsing may be applied to knowledge graphs by assuming undirected graph relations. This approach proves too liberal in its coarsening, however, since the cost of coarsening is increased in knowledge graphs due to the loss of predicate information. In response, we restrict edge collapsing to entities whose collapsing incurs no loss of predicate information other than predicates that are incident to both entities. Formally, entity e_a is collapsed with e_b if:

$$\{t \in \mathcal{A}_a : e_b \notin t\} \subseteq \{t \in \mathcal{A}_b : e_a \notin t\} \tag{1}$$

First-order entity collapsing is demonstrated in the lower left quadrant of Figure 1, where entity e_g is collapsed with its neighbour e_f to form entity cluster e_{fg} in the coarse graph. We note that entities e_a and e_b are also valid candidates for first-order collapsing with e_d. This demonstrates the necessity of initially performing second-order neighbour collapsing since e_a and e_b are structurally equivalent and thus more similar to one another than to e_d. As such, first-order collapsing is performed after second-order collapsing, as seen in Algorithm 1, where it is captured in lines 11 to 18.

3.1.2. Collapsing Second-Order Neighbours

Second-order (i.e., two-hop) neighbours are two entities that share a first-order neighbour. The rationale for preserving second-order proximity is discussed in [13] and predicated on the intuition that entities that have many common first-order neighbours tend to exhibit similar structural and semantic properties. As such, they should be proximal to one another in the embedding space. In Figure 1, we see that second-order neighbours e_a and e_b have identical tag sets (i.e., $\mathcal{A}_a = \mathcal{A}_b$) and are thus structurally equivalent. Collapsing these entities and embedding them jointly ensures the preservation of second-order proximity in the embedding space while incurring no loss of information. We apply this reasoning to our coarsening procedure. Namely, if two second-order neighbours exhibit a high degree of structural similarity, we collapse them in the coarse graph. We measure the similarity between a pair of second-order neighbours as the Jaccard coefficient between their tag sets:

$$\text{Sim}(e_a, e_c) = \frac{\mathcal{A}_a \cap \mathcal{A}_c}{\mathcal{A}_a \cup \mathcal{A}_c} \tag{2}$$

where $\text{Sim}(e_a, e_c)$ is the similarity between e_a and e_c such that $0 \leq \text{Sim}(e_a, e_v) \leq 1$. Second-order neighbours are collapsed if their similarity is greater than or equal to a threshold, α, which is chosen such that $0 < \alpha \leq 1$. The value of α dictates the coarseness of the graph, with lower α values resulting in smaller, coarser graphs. This process may be seen as a relaxation of Structural Equivalence Matching (SEM) proposed in [10], which collapses second-order neighbours only if they are structurally equivalent. In other words, SEM is analogous to our method of $\alpha = 1$; collapsing entities e_a and e_c when $\mathcal{A}_a = \mathcal{A}_c$. Second-order collapsing is summarized in lines 2 to 10 of Algorithm 1.

Algorithm 1 Coarse knowledge graph embeddings.

Input: base graph $\mathcal{K}$; collapsing threshold α; random walk count η
Output: base embeddings $\mathbf{E}$

1: Initialize $\mathcal{K}'$ and Ψ
2: **for all** $e_a \in \mathcal{E}$ **do**
3: **for** *iteration* in 1, 2, ..., η **do**
4: Obtain second order neighbour e_c using (5) and (6)
5: Calculate $\text{Sim}(e_a, e_c)$ using (2)
6: **if** $\text{Sim}(e_a, e_c) \geq \alpha$ and $e_c \notin \Psi$ **then**
7: Collapse e_a with e_c; Update $\mathcal{K}'$ and Ψ
8: **end if**
9: **end for**
10: **end for**
11: **for all** $e_a \in \mathcal{E}$ **do**
12: **for** *iteration* in 1, 2, ..., η **do**
13: Obtain first order neighbour e_b using (3) and (4)
14: **if** (1) holds for e_a and $e_a \notin \Psi$ **then**
15: Collapse e_a into e_b; Update $\mathcal{K}'$ and Ψ
16: **end if**
17: **end for**
18: **end for**
19: Obtain coarse embeddings $\mathbf{E}'$ using (7)
20: **for all** $e_a \in \mathcal{E}$ **do**
21: Reverse map base embedding $\mathbf{E}[e_a]$ using (8)
22: **end for**
23: Fine tune base embeddings $\mathbf{E}$ using (9)

3.1.3. Neighbour Sampling

The pairwise comparison between entities and their first- and second-order neighbourhoods has a worst-case time complexity of $O(|\mathcal{E}|^2)$ and is thus computationally infeasible

for large-scale knowledge graphs. To overcome this, we propose a scheme for sampling neighbours using constrained random walks. To obtain a first-order neighbour for entity e_a, we first sample a tag from its tag set:

$$t_1 \sim \text{Uniform}(\mathcal{A}_a) \tag{3}$$

where t_1 represents one hop in a random walk on the base graph. The first-order neighbour e_b is then extracted from t_1:

$$e_b = t_1 \cap \mathcal{E} \tag{4}$$

Note that since $t_1 \cap \mathcal{E}$ is a singleton set, we can abuse the $=$ symbol such that $e_b = \{e_b\}$. The predicate $p_r = t_1 \cap \mathcal{P}$ is used as a constraint in sampling a second-order neighbour for e_a. Specifically, given e_a and its first-order neighbour e_b on predicate p_r, we only sample second-order neighbours that are incident to e_b on p_r:

$$t_2 \sim \text{Uniform}(\{t \in \mathcal{A}_b : p_r \in t \wedge e_a \notin t\}) \tag{5}$$

The second-order neighbour e_c is extracted from t_2 analogously to e_b:

$$e_c = t_2 \cap \mathcal{E} \tag{6}$$

Sampled neighbours are collapsed if they meet the aforementioned requirements, resulting in a stochastically derived coarse graph.

We sample $\eta \geq 1$ neighbours for each entity, resulting in a $O(|\mathcal{E}|\eta)$ time complexity for our sampling scheme. As a hyperparameter of coarsening, η is chosen a priori allowing for flexibility to account for knowledge graph size. In practice, we see that even small values of η yield encouraging results. The intuition behind this may be summarized as follows:

- Entities that meet the criteria for collapsing are likely to have smaller neighbourhoods.
- Entities that belong to smaller neighbourhoods have a higher chance of getting sampled as candidates for collapsing.

This allows our strategy to be performed with little added computational overhead. We note that reading a dataset has a time complexity of $O(|\mathcal{K}|)$, which may itself be more computationally taxing than coarsening on dense knowledge graphs.

3.2. Coarse Graph Embedding

Having coarsened the base graph, coarse embeddings are obtained by applying an arbitrary embedding method on the coarse graph. Since the coarse graph has all the properties of its base counterpart, no additional changes to the embedding method are necessary, merely a different input. Due to there being fewer entities in the coarse graph than its base counterpart, coarse embeddings may require fewer training steps, resulting in faster training times. We use the notation $f(\mathcal{K}')$ to denote the embedding of coarse graph $\mathcal{K}'$ to yield coarse embeddings $\mathbf{E}'$:

$$\mathbf{E}' = f(\mathcal{K}') \tag{7}$$

Line 19 in Algorithm 1 places this step in the context of our whole strategy.

3.3. Reverse Mapping and Fine Tuning

Coarse embeddings are extended down as base embeddings $\mathbf{E}$ by reversing the mapping obtained in the coarsening step:

$$\mathbf{E}[e_a] = \mathbf{E}'[\Psi(e_a)] \tag{8}$$

where $\mathbf{E}[e_a]$ indexes the base embedding for e_a. A consequence of reverse mapping is that entities that were coarsened together share identical embeddings. In applications that

rely on the distinction between these entities, this property is not desired. As such, base embedding may be fine-tuned by embedding $\mathbf{E}$ with respect to the base graph using $\mathbf{E}'$ as initialisation. This ensures that structural information that was lost in the coarsening process is reintroduced to base embeddings, and collapsed entities become delineated. Furthermore, the training process may be less likely to get stuck in local minima due to its global initialisations. We use the following notation to capture fine-tuning $\mathbf{E}$ using $\mathbf{E}'$ as initialisation:

$$\mathbf{E} = f(\mathcal{K}|\mathbf{E}') \tag{9}$$

Reverse mapping and fine-tuning are described in lines 20 to 23 of Algorithm 1 as the final steps in our strategy.

4. Evaluation

We evaluate our strategy on the entity classification task performed in [3,7], which involves embedding a knowledge graph and using the embeddings to infer entity labels. Our strategy is compared pairwise against the baseline methods used in the embedding step. This allows us to measure whether our coarsening strategy is justified in comparison to using the baseline methods conventionally. We use three baseline methods to evaluate our strategy: RDF2Vec, R-GCN, and TransE. In performing our evaluation, we extend the code provided in [3,42] and publish it online for replication (https://sites.ualberta.ca/~pietrasi/coarse_embeddings.zip (accessed on 3 March 2023).

4.1. Datasets

We use four canonical datasets from [3,7] in our evaluation: MUTAG, AIFB, BGS, and AM. Each dataset consists of a knowledge graph in Resource Description Framework format and labels for a subset of its entities. We mirror [3] in removing knowledge graph relations, which are on predicates that correlate strongly with their labels. Statistics for each dataset are provided in Table 1. What follows is a brief description of each dataset.

- **MUTAG** depicts the properties and interactions of molecules that may or may not be carcinogenic. We remove the labelling predicate `isMutagenic` from the dataset.
- **AIFB** reports the work performed at the AIFB research group and labels its members by affiliation. We remove predicates, `employs`, and `affiliation`.
- **BGS** captures geological data from the island of Great Britain and is used to predict the lithogenicity of rocks. As such, we remove the `hasLithogenesis` predicate.
- **AM** describes and categorises artefacts in the Amsterdam Museum. We remove the `materials` predicate as it correlates with artefact labels.

Table 1. Summary of datasets used in this paper.

Dataset	MUTAG	AIFB	BGS	AM
Triples	74,227	29,043	916,199	5,988,321
Entities	23,644	8285	333,845	1,666,764
Predicates	23	45	103	133
Labelled	340	176	146	1000
Classes	2	4	2	11

4.2. Procedure

Embeddings were learned using each of the baseline embedding methods on the base graph and on the coarse graph as per our strategy. To assess the quality of these embeddings, entity classification was performed by training a support vector machine on 80% labelled entities and testing on the remaining 20% using splits provided in [7]. We use the accuracy of classification on the testing entities as the metric of our strategy's performance. To account for stochasticity in this process, embeddings were learned and evaluated ten times for each dataset.

To obtain optimal results, we set aside 20% of the training entities as validation for hyperparameter selection and to prevent overfitting. In magnanimity, we used embedding hyperparameters, which were selected on validation results obtained on the base graphs. For coarsening hyperparameters, we performed exploration on $\alpha \in \{0.25, 0.5, 0.75, 1\}$ and used $\eta = 10$ in all of our experiments. The optimal hyperparameters for each dataset and baseline method may be found with our published code.

4.3. Results

The results of our evaluation are summarised in Table 2. We use the notation C(x) to refer to our strategy applied to baseline embedding method x. Our strategy improved on the baseline in 10 of the 12 experiments, 7 of which were statistically significant. Furthermore, we were able to achieve state-of-the-art performance on three of the four datasets, albeit using different baseline methods.

Table 2. Results of pairwise comparison between our strategy and baseline embedding methods as measured by accuracy (mean ± standard deviation) obtained on testing entities for each dataset. Asterisk (*) indicates superior performance as per Student's t-test at a 0.05 level of significance. Underline indicates top performance on the dataset, regardless of the baseline method.

Method	MUTAG	AIFB	BGS	AM
RDF2Vec	0.7500 ± 0.0392	0.9111 ± 0.0117	0.7828 ± 0.0327	0.8758 ± 0.0143
C(RDF2Vec)	$\underline{\mathbf{0.7956 \pm 0.0340}}$	$\mathbf{0.9167 \pm 0.0000}$	$\underline{\mathbf{0.8828 \pm 0.0178}}$	$\mathbf{0.8778 \pm 0.0211}$
Change	**6.1% ***	0.6%	**12.8% ***	0.2%
R-GCN	$\mathbf{0.7397 \pm 0.0286}$	0.9528 ± 0.0264	0.8345 ± 0.0424	$\underline{\mathbf{0.8833 \pm 0.0197}}$
C(R-GCN)	0.7294 ± 0.0242	$\underline{\mathbf{0.9694 \pm 0.0088}}$	$\mathbf{0.8690 \pm 0.0317}$	0.8828 ± 0.0138
Change	−1.4%	**1.7% ***	**4.1% ***	−0.1%
TransE	0.7397 ± 0.0422	0.8722 ± 0.0397	0.6793 ± 0.0371	0.4207 ± 0.0143
C(TransE)	$\mathbf{0.7412 \pm 0.0368}$	$\mathbf{0.9056 \pm 0.0299}$	$\mathbf{0.7759 \pm 0.0335}$	$\mathbf{0.4955 \pm 0.0179}$
Change	0.3%	**3.8% ***	**14.2% ***	**17.7% ***

Although extensive comparisons between the baseline methods with and without the application of our strategy are outside the scope of this paper, we suggest possible reasons for the improved results and how employing coarsening may overcome certain limitations present in the baselines. The main difference in using our strategy in conjunction with RDF2VEC is in how it changes the sequences obtained from the random walks on the knowledge graph. Because densely connected entities are less likely to be collapsed in the coarsening process, they are more likely to be sampled in walks on the coarsened graph. This bears some similarity to RDF2VEC Light [43], which only performs walks on entities of interest if we assume that densely connected entities are more likely to be of interest. Furthermore, on all four datasets, we see a larger reduction in the percentage of entities as opposed to triples. This too changes the nature of sampled walks, namely in that predicates are more likely to be sampled. The difference between sampling entities versus predicates was discussed in [44] and termed e-walks and p-walks, respectively. In short, it was shown that e-walks are better at capturing the relatedness of entities, and p-walks are better at capturing their semantic similarity. Because coarsening changes the sampled walks in the direction of p-walks, our strategy is theorised to be better at capturing semantic similarity between entities. In general, the idea of biasing random walks in order to improve the performance of RDF2VEC is well studied and extensively evaluated in [45]. The advantage of using our strategy in conjunction with a R-GCN baseline is that of parameter reduction. Recall that R-GCN relies on computations made on the adjacency matrix of the knowledge graph. Such a formulation results in an increasing number of parameters as the size of the knowledge graph increases, making its scalability poor for very large datasets. By coarsening the knowledge graph, we reduce the number of trainable parameters in the model, improving its efficiency. We note, however, that

our strategy appears to perform worse on R-GCN relative to the other baselines. The reason for this may be that coarsening produces graphs with a larger proportion of highly connected hub entities, which is a structural weakness of R-GCN as pointed out by its authors. The limitations of the translational assumption inherent to the TransE model is not addressed by our strategy directly, as the underlying embedding procedure is never changed. These include the inability to properly handle relational patterns such as one-to-many, many-to-one, inverse, and symmetric relations and were largely addressed by subsequent models in the translational family, such as the aforementioned TransR and TransH models. Specifically, TransR finesses the issue of entities and predicates being embedded in the same space by introducing a separate predicate embedding space. Entities are then projected to the predicate space by a predicate-specific projection matrix, and the original translation assumptions of the TransE model are applied. This solves the problem of embedding entities, which may or may not be similar to one another depending on their predicate context. TransH was proposed to handle the problem of embedding one-to-many and many-to-one relationships. To this end, it too introduced the notion of predicate-specific projections, although unlike TransR, entities and predicates were mapped in the same space. In contrast to these approaches, our strategy may be seen as improving on TransE indirectly, namely by augmenting the input data. One-to-many and many-to-one relationships are likely candidates for second-order collapsing, ensuring tight embeddings of these entities by TransE. Such a feature does not rectify TransE's challenges in handling these types of relationships but merely accepts its inadequacy and ensures that less computation is spent on embedding these entities. This is also the case for symmetrical relationships whose entities are candidates for first-order collapsing. Thus, in order to overcome the obstacles of TransE, it would be advised to apply our strategy in conjunction with one of its successors, which explicitly deals with the limitations mentioned. We note that despite our strategy's positive impact in regard to the aforementioned limitations, it is formulated to be embedding-method agnostic. As such it does not inherently seek to overcome the limitations of any particular embedding method. Finally, we see that datasets with knowledge graphs that have a higher degree of reduction at $\alpha = 0.5$ and $\eta = 10$ perform better. This is because not all knowledge graphs are equally suitable candidates for coarsening. Namely, knowledge graphs that exhibit a high degree of structural equivalency between entities lose less information in the coarsening process. We see this in the AIFB and BGS datasets, where more than half of their entities get collapsed in the coarsening step, as shown in Table 3.

Table 3. Percent reduction in coarse graphs relative to base graphs at $\alpha = 0.5$ and $\eta = 10$.

Dataset	MUTAG	AIFB	BGS	AM
Triples	52,179	20,134	501,722	4,080,981
Change	−29.7%	−30.7%	−45.2%	−31.8%
Entities	16,115	2801	78,335	944,759
Change	−31.8%	−66.2%	−76.5%	−43.3%
Predicates	23	43	97	129
Change	0%	−4.4%	−5.8%	−3.0%

Figure 2 plots the performance of our strategy compared to baselines when increasing the number of training steps performed. Due to computational constraints, we trained the embeddings up to fifty epochs. It is possible that given enough training, baseline methods could catch up in performance to our coarsening strategy, as can be seen on the AM dataset using TransE. This, however, still demonstrates that coarse graphs produce quality embeddings faster than embedding on base graphs. This is further confirmed by RDF2Vec on AIFB and AM and R-GCN on MUTAG, which show similar trendlines to the coarsened counterpart requiring fewer training steps. This suggests that our strategy is faster to train than its baseline counterparts.

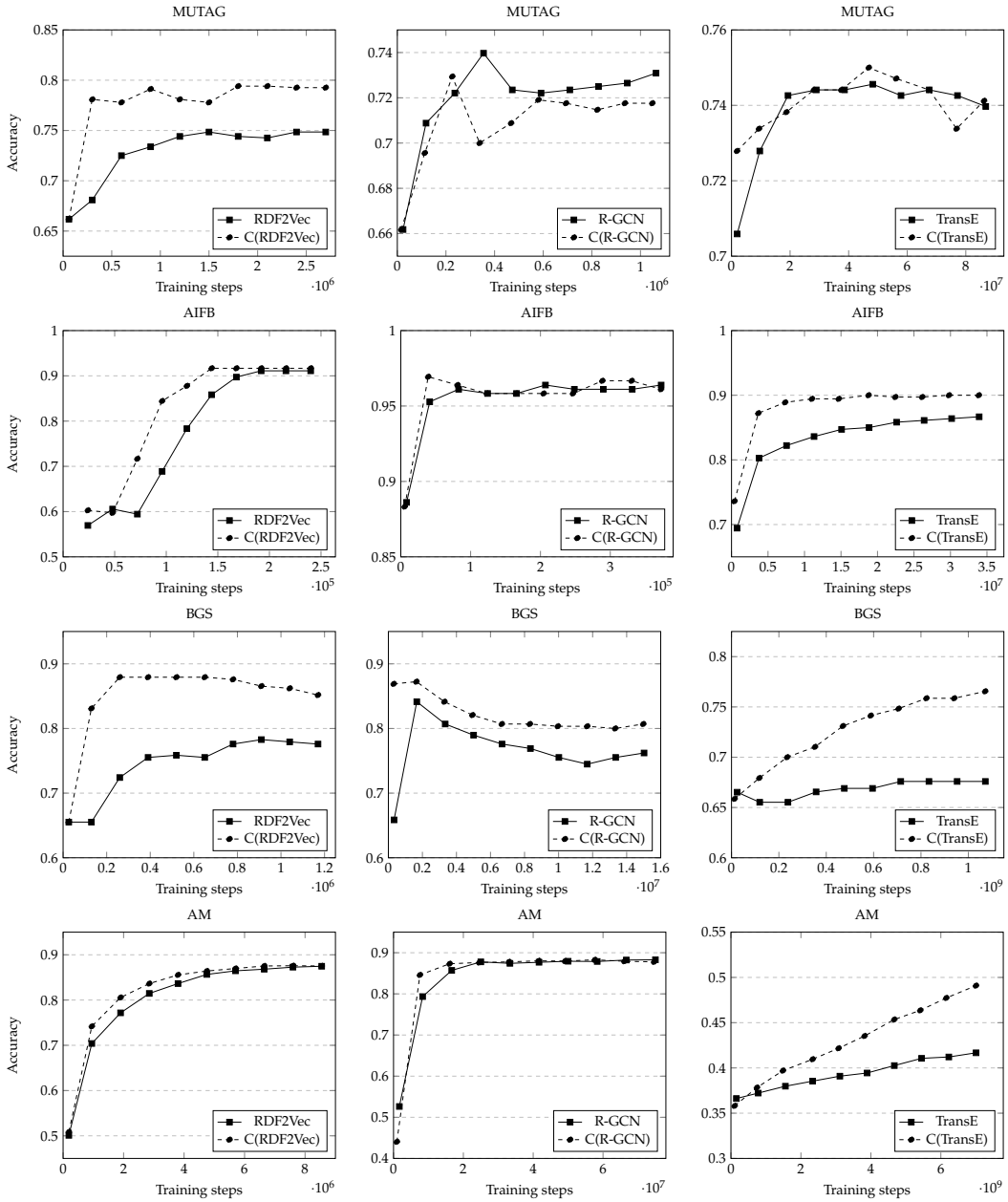

Figure 2. Pairwise comparison between the baseline method and our strategy demonstrating performance (accuracy) as a function of the number of training steps performed for each dataset.

5. Conclusions

We introduced a simple meta-strategy for embedding knowledge graphs that relies on coarsening as a preprocessing step to obtain a reduced knowledge graph prior to embedding. To this end, we adapted existing graph coarsening concepts to knowledge graphs and introduced a novel entity collapsing and neighbour sampling scheme. Our evaluation demonstrates that such an approach results in faster and, oftentimes, more accurate knowledge graph embeddings. Coupled with the fact that our strategy incurs little overhead costs, we conclude that graph coarsening is a recommended preprocessing step before applying any existing knowledge graph embedding method.

Despite the encouraging results, there are still limitations to our work and avenues for future work. The evaluation of the reverse mapping procedure could be performed by a simple pairwise comparison. Specifically, comparing the embedding quality with and without reverse mapping and fine-tuning. We anticipate, however, that on the entity classification task, such a comparison would not be indicative of the extent to which

information is regained in this step. This is because entity classification is performed on a subset of graph entities that are highly connected and thus unlikely to be collapsed in the coarsening process. This is due to the requirements for collapsing and their satisfaction being more likely by peripheral entities. As such, the embeddings of classified entities will not change during the reverse mapping and fine-tuning process. This brings attention to another limitation, namely that of evaluating the strategy on different downstream tasks, such as the commonly used link prediction task. In this task, triples not present in the knowledge graph are inferred by assigning a score to reflect the likelihood of observing such a triple. This task was not originally handled by RDF2Vec, whose principal function was data mining. It has been shown more recently [46] that RDF2Vec can be adapted to the link prediction task. As such, testing our strategy on this task is a worthwhile endeavour. Finally, future improvements to our work may be made by proposing alternate coarsening schemes, such as those which leverage graph clustering or community detection. These models present the opportunity for more intelligent collapsing of nodes, which may improve the efficiency of our overall strategy. To this end, models such as KDComm [47], Bayesian Symmetric NMF [48], the Multilayer Stochastic Blockmodel [49], and the Multilayer Mixed Membership Stochastic Blockmodel [50] can be utilized. One of the challenges to overcome if these models were to be integrated with our strategy would be that of computational efficiency. In general, the process by which these models induce communities from the knowledge graph is more expensive than our approach. This would be further exacerbated if multi-step coarsening were to be employed.

Author Contributions: Conceptualisation, M.P. and M.R.; methodology, M.P.; software, M.P.; validation, M.P.; formal analysis, M.P.; investigation, M.P.; resources, M.P and M.R.; data curation, M.P.; writing—original draft preparation, M.P.; writing—review and editing, M.R.; visualisation, M.P.; supervision, M.R.; project administration, M.R.; funding acquisition, M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data may be obtained from: https://sites.ualberta.ca/~pietrasi/coarse_embeddings.zip (accessed on 3 March 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bordes, A.; Usunier, N.; Chopra, S.; Weston, J. Large-scale simple question answering with memory networks. *arXiv* **2015**, arXiv:1506.02075.
2. Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; McCallum, A. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv* **2017**, arXiv:1711.05851.
3. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; Van Den Berg, R.; Titov, I.; Welling, M. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*; Springer: Cham, Switzerland, 2018; pp. 593–607.
4. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 2787–2795.
5. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2d knowledge graph embeddings. *arXiv* **2017**, arXiv:1707.01476.
6. Bellini, V.; Schiavone, A.; Di Noia, T.; Ragone, A.; Di Sciascio, E. Knowledge-aware autoencoders for explainable recommender systems. In Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, Vancouver, BC, Canada, 6 October 2018.
7. Ristoski, P.; Paulheim, H. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*; Springer: Cham, Switzerland, 2016; pp. 498–514.
8. Nickel, M.; Tresp, V.; Kriegel, H.P. A three-way model for collective learning on multi-relational data. In Proceedings of the 28th International Conference on Machine Learning, Bellevue,WA, USA, 28 June–2 July 2011.
9. Chen, H.; Perozzi, B.; Hu, Y.; Skiena, S. Harp: Hierarchical representation learning for networks. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA 2–7 February 2018; pp. 2127—2134.
10. Liang, J.; Gurukar, S.; Parthasarathy, S. Mile: A multi-level framework for scalable graph embedding. *arXiv* **2018**, arXiv:1802.09612.
11. Archdeacon, D. Topological graph theory. *Surv. Congr. Numer.* **1996**, *115*, 18.
12. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.

13. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015.
14. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
15. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
16. Duvenaud, D.K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R.P. Convolutional networks on graphs for learning molecular fingerprints. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2224–2232.
17. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
18. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
19. Simonovsky, M.; Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*; Springer: Cham, Switzerland, 2018; pp. 412–422.
20. Akyildiz, T.A.; Aljundi, A.A.; Kaya, K. Gosh: Embedding big graphs on small hardware. In Proceedings of the 49th International Conference on Parallel Processing (ICPP), Edmonton, AB, Canada, 17–20 August 2020; pp. 1–11.
21. Karypis, G.; Kumar, V. Multilevelk-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.* **1998**, *48*, 96–129. [CrossRef]
22. Wang, Y.; Dong, L.; Jiang, X.; Ma, X.; Li, Y.; Zhang, H. KG2Vec: A node2vec-based vectorization model for knowledge graph. *PLoS ONE* **2021**, *16*, e0248552. [CrossRef]
23. Fionda, V.; Pirró, G. Triple2Vec: Learning Triple Embeddings from Knowledge Graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020.
24. Portisch, J.; Paulheim, H. Putting rdf2vec in order. In Proceedings of the International Semantic Web Conference (ISWC 2021): Posters and Demo, Virtual Conference, 24–28 October 2021.
25. Busbridge, D.; Sherburn, D.; Cavallo, P.; Hammerla, N.Y. Relational graph attention networks. *arXiv* **2019**, arXiv:1904.05811.
26. Yasunaga, M.; Ren, H.; Bosselut, A.; Liang, P.; Leskovec, J. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In *2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 535–546.
27. Alshahrani, M.; Thafar, M.A.; Essack, M. Application and evaluation of knowledge graph embeddings in biomedical data. *PeerJ Comput. Sci.* **2021**, *7*, e341. [CrossRef]
28. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge graph embedding by translating on hyperplanes. In Proceedings of the AAAI Conference on Artificial Intelligence, Portsmouth, NH, USA, 21–26 June 2014, Volume 28.
29. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 29.
30. Ji, G.; He, S.; Xu, L.; Liu, K.; Zhao, J. Knowledge graph embedding via dynamic mapping matrix. In *53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2015; pp. 687–696.
31. Xiao, H.; Huang, M.; Hao, Y.; Zhu, X. TransA: An adaptive approach for knowledge graph embedding. *arXiv* **2015**, arXiv:1509.05490.
32. Nguyen, D.Q.; Sirts, K.; Qu, L.; Johnson, M. STransE: a novel embedding model of entities and relationships in knowledge bases. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2016; pp. 460–466.
33. Ebisu, T.; Ichise, R. Toruse: Knowledge graph embedding on a lie group. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
34. Sun, Z.; Deng, Z.H.; Nie, J.Y.; Tang, J. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
35. Yang, B.; Yih, W.t.; He, X.; Gao, J.; Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *arXiv* **2014**, arXiv:1412.6575.
36. Nickel, M.; Rosasco, L.; Poggio, T. Holographic embeddings of knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
37. Balazevic, I.; Allen, C.; Hospedales, T. TuckER: Tensor Factorization for Knowledge Graph Completion. In *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 5185–5194.
38. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Yu, P.S. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv* **2020**, arXiv:2002.00388.
39. Pietrasik, M.; Reformat, M. A Simple Method for Inducing Class Taxonomies in Knowledge Graphs. In *European Semantic Web Conference*; Springer: Cham, Switzerland, 2020; pp. 53–68.
40. Hendrickson, B.; Leland, R.W. A Multi-Level Algorithm For Partitioning Graphs. *SC* **1995**, *95*, 1–14.
41. Karypis, G.; Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* **1998**, *20*, 359–392. [CrossRef]

42. Han, X.; Cao, S.; Xin, L.; Lin, Y.; Liu, Z.; Sun, M.; Li, J. OpenKE: An Open Toolkit for Knowledge Embedding. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Brussels, Belgium, 31 October–4 November 2018.
43. Portisch, J.; Hladik, M.; Paulheim, H. RDF2Vec Light—A Lightweight Approach for Knowledge Graph Embeddings. In Proceedings of the International Semantic Web Conference, Posters and Demos, Virtual Conference, 1–6 November 2020.
44. Portisch, J.; Paulheim, H. Walk this way! entity walks and property walks for rdf2vec. In *The Semantic Web: ESWC 2022 Satellite Events: Hersonissos, Crete, Greece, 29 May–2 June 2022, Proceedings*; Springer: Cham, Switerland, 2022; pp. 133–137.
45. Cochez, M.; Ristoski, P.; Ponzetto, S.P.; Paulheim, H. Biased graph walks for RDF graph embeddings. In Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, Amantea, Italy, 19–22 June 2017; pp. 1–12.
46. Portisch, J.; Heist, N.; Paulheim, H. Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction–two sides of the same coin? *Semant. Web* **2022**, *13*, 399–422. [CrossRef]
47. Bhatt, S.; Padhee, S.; Sheth, A.; Chen, K.; Shalin, V.; Doran, D.; Minnery, B. Knowledge graph enhanced community detection and characterization. In Proceedings of the twelfth ACM International Conference on Web Search and Data Mining, Melbourne, VIC, Australia, 11–15 February 2019; pp. 51–59.
48. Shi, X.; Qian, Y.; Lu, H. Community Detection in Knowledge Graph Network with Matrix Factorization Learning. In *Web and Big Data: APWeb-WAIM 2019 International Workshops, KGMA and DSEA, Chengdu, China, August 1–3. 2019, Revised Selected Papers 3*; Springer: Cham, Switzerland, 2019; pp. 37–51.
49. Paul, S.; Chen, Y. Consistent community detection in multi-relational data through restricted multi-layer stochastic blockmodel. *Electron. J. Stat.* **2016**, *10*, 3807–3870. [CrossRef]
50. De Bacco, C.; Power, E.A.; Larremore, D.B.; Moore, C. Community detection, link prediction, and layer interdependence in multilayer networks. *Phys. Rev. E* **2017**, *95*, 042317. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.