A framework for differentially-private knowledge graph embeddings

Han, Xiaolin; Dell'Aglio, Daniele; Grubenmann, Tobias; Cheng, Reynold; Bernstein, Abraham

Published in:
Journal of Web Semantics

DOI (link to publication from Publisher):
10.1016/j.websem.2021.100696

Publication date:
2022

Document Version
Accepted author manuscript, peer reviewed version

A Framework for Differentially-Private Knowledge Graph Embeddings

Xiaolin Han[a], Daniele Dell'Aglio[b,c], Tobias Grubenmann[d], Reynold Cheng[a], Abraham Bernstein[c]

[a]*Department of Computer Science, The University of Hong Kong, Hong Kong, China*
[b]*Department of Computer Science, Aalborg University, Aalborg, Denmark*
[c]*Department of Informatics, University of Zurich, Zurich, Switzerland*
[d]*Department of Computer Science, University of Bonn, Bonn, Germany*

Abstract

Knowledge graph (KG) embedding methods are at the basis of many KG-based data mining tasks, such as link prediction and node clustering. However, graphs may contain confidential information about people or organizations, which may be leaked via embeddings. Research recently studied how to apply differential privacy to a number of graphs (and KG) analyses, but embedding methods have not been considered so far. This study moves a step towards filling such a gap, by proposing the Differential Private Knowledge Graph Embedding (DPKGE) framework.

DPKGE extends existing KG embedding methods (e.g., TransE, TransM, RESCAL, and DistMult) and processes KGs containing both confidential and unrestricted statements. The resulting embeddings protect the presence of any of the former statements in the embedding space using differential privacy. Our experiments identify the cases where DPKGE produces useful embeddings, by analyzing the training process and tasks executed on top of the resulting embeddings.

Keywords: Differential privacy, Knowledge graph embeddings

1. Introduction

The open data movement contributed to the evolution of the web by making an unprecedented amount of free and accessible data available. Part of the success is due to the semantic web, which provided a set of solutions to publish data on the web. Central to the semantic web is the role of knowledge graphs (KGs), graph-based data structures with nodes representing entities and edges specifying relations among such entities. Examples of popular open knowledge graphs are Wikdata [43] and DBPedia [3], which store general domain knowledge and make them accessible on the web.

Large amounts of data, however, are still stored by different organizations [33]. Opening these datasets is challenging for several reasons, including privacy, as they often contain confidential information about individuals (e.g., gender) or companies (e.g., assets). For example, HINCare[1] is a non-profit organization building a platform that integrates data from different NGOs about elderly care using KGs. HINCare has no interest in keeping a monopoly on its data, but it is prevented from sharing their data because of privacy concerns. It follows that both companies and Non-Governmental Organizations (NGOs) may shy away from sharing their data due to the lack of privacy guarantees, which might even legally prohibit them from sharing their data, and the implied risk of incurring significant fines (e.g., via the European GDPR[2]). As the example of HINCare shows, there is, however, interest in sharing such data due to their potential value in describing the characteristics and properties of communities and populations. While KGs are an ideal solution to share data following open web standards, we observe a need for techniques to guarantee the privacy of individuals.

So far, various techniques to protect confidential information have been proposed. Most of these techniques [24] build on top of anonymity, hiding the identity of the individuals and their confidential information. For example, k-anonymity [40] ensures that an individual may not be distinguished from at least $k-1$ other individuals. Data swapping [11] switches attribute values between individuals to hide them, while preserving the overall characteristics of the population. Whereas those techniques have been used in the context of data publication, they have been shown to be vulnerable to privacy attacks. Two well-known privacy leakage examples are the Netflix challenge [30] and the Massachusetts hospital dataset [39]. Differential Privacy (DP) [14] emerged as a solution to overcome the limitations of anonymization techniques. The goal of DP is to introduce plausible deniability by adding noise to data to protect the presence (or absence) of any confidential statement in the dataset.

Email addresses: `xlhan@cs.hku.hk` (Xiaolin Han), `dade@cs.aau.dk` (Daniele Dell'Aglio), `grubenmann@cs.uni-bonn.de` (Tobias Grubenmann), `ckcheng@cs.hku.hk` (Reynold Cheng), `bernstein@ifi.uzh.ch` (Abraham Bernstein)

[1] `https://www.hincare.hku.hk`

[2] `https://eur-lex.europa.eu/eli/reg/2016/679/oj`

We believe that there is an opportunity to combine differential privacy with the recent trend of knowledge graph embeddings [44, 6]. KG embeddings introduce techniques to represent knowledge graphs in low-dimensional vector spaces and, hence, in a numeric space. Such representations facilitate many important machine learning and data mining tasks [5, 19, 28, 31, 45], e.g., link prediction, entity classification, entity resolution, relation extraction, question answering, recommender systems, graph completion, and clustering. Sharing embeddings instead of the original knowledge graph, therefore, still enables a large number of applications.

The goal of this research is to investigate the idea of privacy-preserving knowledge graph embeddings. Among the various privacy techniques, we believe differential privacy is particularly suitable to obfuscate the KG embeddings due to their numerical representation of the KG content. Current studies on DP and knowledge graphs focused on the query answering process, where answers to analytical queries were perturbed [12, 35, 38]. To the best of our knowledge, this is the first study about applying differential privacy preserving techniques on knowledge graph embeddings.

The main result of our study is the *Differentially Private Knowledge Graph Embedding* (DPKGE) framework. This framework extends existing embedding methods (e.g., TransE and RESCAL) into differentially private embedding methods. It is worth noting that embedding techniques are not enough to overcome the privacy issue: the vectors may preserve confidential information. As recent research in deep learning has shown [16], it is possible to reconstruct images from a trained face recognition model. To overcome this problem, Abadi et al. [1] introduced differential privacy into the training phase of deep learning models by adding noise to the Stochastic Gradient Descent (SGD). This modification leads to the Differentially Private Stochastic Gradient Descent (DPSGD) [1]. The SGD is responsible for optimizing the model and as such, might encode sensitive information into the model that can be retrieved at a later stage by an attacker. By replacing the SGD by a DPSGD, sensitive information contained in the training set can be better protected against such attacks. Inspired by this, DPKGE introduces DP in the KG embeddings learning phase by exploiting DPSGD to protect the learning of sensitive statements in the KG. The DPSGD introduces noise in the computation of the gradient while minimizing the target loss function.

Our experimental results show that it is feasible to introduce DP during the learning phase of knowledge graph embeddings. They also suggest that enforcing differential privacy only on confidential statements results in a higher utility of the embeddings for tasks such as clustering and link prediction. To establish the performance of our framework on real datasets, we complement the datasets usually used in embedding experiments (FB15k[3], FB15k-237[4], and YAGO3-10[5]) with two new ones (MIMIC-III [22] and eICU [34]), which contain both confidential and unrestricted information from the health sector. Since these two datasets are not available in RDF, we provide appropriate mappings to create the corresponding knowledge graphs for the evaluation of privacy-sensitive data.

The contributions of our paper can be summarized as follows:

- We formalize the problem of differential private knowledge graph embeddings.
- We introduce our DPKGE framework to transform existing embedding methods into differential private embedding methods and provide theoretical guarantees on the differential privacy of the DPKGE methods.
- We provide mappings to create knowledge graphs from the MIMIC-III and eICU datasets, which can be used to test privacy algorithms for knowledge graphs. Moreover, the DPKGE methods have been extensively evaluated on five datasets regarding utility, privacy, clustering, and link prediction. They show that DPKGE can improve the utility of the embeddings while preserving the differential privacy of confidential information.

The remainer of this article is structured as follows. Section 2 discusses background and related research. Section 3 proposes our differential privacy framework for knowledge graph embeddings. Section 4 presents the experiments and their results. Section 5 discusses the limitations of our framework. Section 6 concludes our paper and discusses future research.

2. Background and Related Research

Knowledge Graphs (KGs) capture information in graph-based data structures, where nodes denote entities and directed labeled edges denote relationships among them. Examples of KGs include DBpedia [3] and Wikidata [43]. Inspired by [5], we formally define a knowledge graph as follows.

Definition 1 (Knowledge graph)**.** *Let $\mathcal{E}$ and $\mathcal{L}$ denote the set of entities and relationships. A knowledge graph $\mathcal{K} \subset \mathcal{E} \times \mathcal{L} \times \mathcal{E}$ is a set of statements (h, l, t), where $h, t \in \mathcal{E}$ and $l \in \mathcal{L}$.*

In the following, we introduce KG embeddings and differential privacy.

[3] `https://everest.hds.utc.fr/lib/exe/fetch.php?media=en:fb15k.tgz`

[4] `https://github.com/louisccc/KGppler/raw/master/datasets/fb15k-237.tgz`

[5] `https://github.com/louisccc/KGppler/raw/master/datasets/YAGO3-10.tar.gz`

2.1. KG Embedding Methods

KG embeddings [6] recently emerged as a solution to represent the content of a KG in a dense vector space, which can be used in different tasks such as link prediction [5] or recommender systems [37]. Studies on KG embeddings can broadly be grouped into two categories: translational and bilinear models. Translational models project entity embeddings into a relation-specific space. Many variants of translation strategies have been developed toward this research line. In contrast to translational models, bilinear models use bilinear functions to model the entities and relations embeddings.

TransE [5] is one of the most popular translational models. The idea behind TransE is that, given a statement (h, l, t), the embedding of the tail entity t should be as close as possible to the sum of the embeddings of the head entity h and the relation l. Many variants of TransE were proposed to overcome the limitations of this method, such as coping with one-to-many, many-to-one, and many-to-many relations. TransH [46] models the relation of two entities as a translation operation on a hyperplane. TransM [15] precalculates the weight for the scoring function of each statement in TransE, which is the distance between the head entity embedding plus relation embedding and the tail entity embedding. TransM multiplies the precalculated weights with the scoring function to optimize the model. TransR [28] represents the embeddings of entities and relations in separate spaces instead of one common space to capture the idea that there may exist many different relations that focus on multiple aspects of entities. TransD [20] includes the diversity of entities in the model, and it is capable of handling large-scale graphs due to a limited number of parameters.

Bilinear models describe relationships with special quadratic functions, which are bilinear. RESCAL [31] applies tensor factorization on multi-relational data to learn the embeddings of entities and relations. DistMult [47] restricts matrix operators for relations to be a diagonal matrix in order to reduce the number of relation parameters. ComplEx [42] further extends the scoring function into a complex-valued function to handle various binary relations, such as symmetric and antisymmetric relations. It is simpler and more efficient compared with the standard model. TuckER [2] uses Tucker decomposition on the binary multi-relational data to learn the embeddings of entities and relations.

2.2. Differential Privacy

Differential Privacy (DP) is a framework proposed by Dwork et al. [14] to protect the presence of records in a dataset. DP introduces mechanisms M as processes that transform an input dataset D in a computational result $M(D)$. The idea of DP [14] is that the result of executing M over a dataset D is similar to the result of executing M over a neighbor dataset D' (i.e., D and D' differ in one record, denoted as $||D - D'||_1 = 1$), protecting the presence (or absence) of any user in the dataset.

Definition 2 ((ε, δ)-differential privacy [14])**.** *An algorithm M is (ε, δ)-differentially private if for every $E \subseteq \text{Range}(M)$ and for all D, D' such that $||D - D'||_1 = 1$:*

$$P[M(D) \in E] \leq e^{\epsilon} \cdot P[M(D') \in E] + \delta, \qquad (1)$$

where the probability space is over the coin flips of M. If $\delta = 0$, M is ε-differentially private.

The parameters ϵ and δ regulate differential privacy. ϵ is the *privacy budget*, which controls the trade-off between privacy and utility. The lower the ϵ values, the higher amount of privacy is enforced in the mechanism M. The value δ allows a mechanism M to violate the ϵ-differential privacy definition, as M can output results E such that $\frac{P[M(D)\in E]}{P[M(D')\in E]} \geq e^{\epsilon}$. While it is ideal to have (ϵ, δ)-differential privacy mechanisms, they often find application in scenarios where ϵ-differential privacy is considered too strict. Dwork and Roth [14] suggests to use values of δ not bigger than $\frac{1}{|D|}$, where $|D|$ is the size of the dataset.

DP has been largely studied in the database research area. The initial focus has been on query answering, with a set of solutions that has led to systems able to cope with queries including a large set of operators [29, 26, 8]. While those techniques may be viable to expose knowledge graph information through query interfaces, they do not suit our target scenario. Data mining and training machine learning models require a high number of queries, which lead to a large amount of ε. We need solutions that can perform such tasks with a limited consumption of privacy budget. Another set of studies from database research focus on data publication. Kotsogiannis et al. [25] introduce one-sided differential privacy to share location data. They distinguish between sensitive and non-sensitive locations, ensuring that the former are protected under DP. Cunningham et al. [10] propose a novel technique for publishing trajectory data under differential privacy. Their method exploits geographical information and metadata about the trajectory points to guide the obfuscation process, increasing the overall utility of the result. These studies are related to our as they aim at releasing data sets that preserve the privacy of the individuals described in the original data. In our study, specifically, we study how to release a knowledge graph as KG embeddings, which can be used for data mining or machine learning tasks. As in [25], we account for the fact that only part of the data is sensitive and needs protection.

Differential privacy for graphs. While DP initially focused on datasets defined as a set of records with the same structure, recent studies have focused on different data models. When moving to graphs, the main difference relies on the notion of neighboring datasets. Hay et al. [17] propose two definitions for undirected unlabeled graphs. Two graphs are *edge-neighbor* if they differ in one edge, and *node-neighbor* if they differ in one node and the edges involving such a node. The two neighbor definitions lead to the edge- and node-differential privacy, respectively.

Initial research in this area focused on edge-differential privacy, proposing mechanisms for typical graph operations, such as node degree distribution [17], minimum spanning tree cost [32] and cuts [4]. Mechanisms for node-differential privacy were proposed in later years, such as [9, 23], considering particular classes of graphs and known constraints on their characteristics.

Differently from the graph considered in the aforementioned studies, knowledge graphs are multi-modal directed graphs. Silva et al. [38] propose a system to compute differentially private statistics over social relationship RDF graphs, which are defined as directed graphs with only one property. SihlQL [12] is a differentially-private query language for computing histograms from streams of knowledge graphs. Reuben [35] extended the definitions of Hay et al. for multi-modal directed graphs.

Since knowledge graph embeddings represent every node in a graph as a vector, the presence or absence of a node can be immediately detected by the presence or absence of the corresponding vector. Hence, node-differential privacy is not directly applicable to knowledge graph embeddings. In this study, we build on the edge-differential privacy notion. We adapt the definition in [35] for edge-neighboring knowledge graphs as follows.

Definition 3 (Edge-neighboring knowledge graphs)**.** *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be two knowledge graphs. $\mathcal{K}_1$ and $\mathcal{K}_2$ are edge-neighbor if they differ in one statement, that is, $\exists(h,l,t) \in \mathcal{K}_1 \cup \mathcal{K}_2$ s.t. $(\mathcal{K}_1 \setminus \mathcal{K}_2) \cup (\mathcal{K}_2 \setminus \mathcal{K}_1) = \{(h,l,t)\}$.*

In this research, we focus on designing an edge-differential privacy mechanism to compute knowledge graph embeddings.

Differential privacy for stochastic gradient descent. Stochastic Gradient Descent (SGD) is a common state-of-the-art solution to solve optimization problems typical of machine learning scenarios, including knowledge graph embeddings. Abadi et al. [1] propose the Differentially Private SGD (DPSGD) algorithm. The idea of DPSGD is to inject Gaussian noise to the gradients according to the following formula:

$$\tilde{\mathrm{g}} \leftarrow \frac{1}{b}\Big(\sum_i \bar{\mathrm{g}}(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\Big), \tag{2}$$

where b is the batch size, $\mathbf{I}$ the identity matrix, and $\sigma^2 C^2$ the variance of the Gaussian noise mechanism. The parameter C is a threshold that controls the clipping of each gradient. DPSGD has been proposed in the context of deep learning. The gradient of a sample x_i is defined as:

$$\mathrm{g}(x_i) \leftarrow \nabla_\theta \mathcal{L}(\theta, x_i), \tag{3}$$

where $\mathcal{L}$ is the loss function, and θ are the parameters in the deep learning model which needs to be optimized. The clipped gradient of a sample x_i is defined as:

$$\bar{\mathrm{g}}(x_i) \leftarrow \mathrm{g}(x_i) / \max\left(1, \frac{\|\mathrm{g}(x_i)\|_2}{C}\right), \tag{4}$$

where $\|\mathrm{g}(x_i)\|_2$ is the L_2 norm of $\mathrm{g}(x_i)$.

The algorithm for the differentially private stochastic gradient descent, Algorithm 2, can be found in the Appendix Appendix A. The following Theorem by [1] states an important relationship, which we will make use of in this paper.

Theorem 1. [1] *There exist constants c_1 and c_2 so that given the sampling probability $q = L/N$ and the number of steps T, for any $\epsilon < c_1 q^2 T$, Algorithm 2 is (ε, δ)-differentially private for any $\delta > 0$ if we choose*

$$\sigma \geq c_2 \frac{q\sqrt{T \log(1/\delta)}}{\varepsilon} . \tag{5}$$

The adoption of the Gaussian noise to achieve DP leads to the problem of quantifying the effective privacy budget ε. Abadi et al. [1] propose an *accountant mechanism* to estimate an upper bound for ϵ, which is specifically designed for the DPSGD to provide a tighter bound than similar estimation methods. For the accountant to work, the standard deviation of the added Gaussian noise needs to be proportional to the L_2 norm of the gradient, or larger. By clipping the gradients, the L_2 norm of the gradients is at most C. Consequently, adding random noise from a Gaussian distribution $\mathcal{N}(0, \sigma^2 C^2 \mathbf{I})$ ensures that the noise requirements above are met.

It is worth noting that the choice of the clipping parameter does not affect the estimated upper bound for ϵ, because the accountant assumes that each gradient has an L_2 norm of C and a proportional amount of noise of added to it. Nevertheless, the actual ϵ might change with varying C. The following example illustrates this.

Example 1. Let $\sigma_1 = 1$, $\sigma_2 = 10$, and $C = 1$. It follows that the added noise is a sample from $\mathcal{N}(0, 1\mathbf{I})$ and $\mathcal{N}(0, 100\mathbf{I})$ for σ_1 and σ_2, respectively. Let us assume that every single gradient appearing in the mechanism M has an L_2 norm smaller than 0.1. This means that no gradient is affected by the clipping. Now, let $C' = 0.1$. Again, no gradient is affected by the clipping. With this new parameter C', the added noise is $\mathcal{N}(0, 0.01\mathbf{I})$ distributed for σ_1 and $\mathcal{N}(0, 1\mathbf{I})$ for σ_2. Since in both cases the gradients are not clipped, the case (σ_1, C) is equivalent to (σ_2, C'), as the exact same amount of noise is added. Consequently, they have the same expected value for ϵ. However, the accountant estimates the ϵ value for (σ_1, C) much higher than for (σ_2, C'). This is because the accountant is not aware that the gradients have not been clipped and only knows that the added noise is at least proportional to the gradients.

A key point in the above example is that the gradients have not been clipped in both cases, meaning that the chosen C values are inappropriately high. When the clipping affects the gradient, the two settings (σ_1, C) and (σ_2, C') are not equivalent anymore, and the above reasoning does not apply.

One might conclude that the best strategy to ensure an accurate estimation of ϵ with the accountant would be to choose a value for C that ensures that all gradients are affected by clipping. Unfortunately, it is not that simple, as a too strict clipping might negatively affect the utility of the outcome of the algorithm when some gradients are drastically shortened. Abadi et al. [1] recommend to choose a value for C that equals the median of the L_2 norms of all gradients. However, depending on the setting and the chosen mechanism M, different values for C might yield better results.

3. Differential Privacy for KG Embeddings

In this section, we discuss how to construct a differentially private knowledge graph embedding algorithm. To better understand how knowledge graph embeddings are affected by differential and non-differential private embeddings, we begin with introducing the following example.

Figure 1(a) shows a graph with four statements. One would obtain the embedding space in Figure 1(b) by a state-of-the-art embedding method, like TransE [5], without considering differential privacy (NDP). As the graph contains a statement about Rose having a hearing disease, the embedding space will preserve that information, and the sum of the vectors associated to *Rose* and the *has disease* property will be similar to the *Hearing Disease* vector.

The idea of plausible deniability boils down to the fact that one may deny the presence of some personal information in the dataset. This leads to the knowledge graph in Figure 1(d), which is the same as the one in Figure 1(a), except for the missing statement *(Rose, has disease, Hearing Disease)*. Figure 1(e) shows the embeddings generated in this case: as the statement about Rose is missing, the three vectors are not related anymore.

Differential privacy introduces the idea that the graphs in Figure 1(a) and Figure 1(d) should lead to similar embeddings, as the two graphs differ in only one statement, i.e., the two graphs are neighbors. This is depicted in the two embedding spaces in Figures 1(c) and 1(f).

Without DPKGE, one would need to remove the two confidential statements from the knowledge graph before the embedding process. However, this might lower the quality of the embedding. Looking at the example in Figure 1, by removing the red statements, Rose and Sue would be less similar, as they would share only the *citizen of* relation.

To better control the privacy injection, we also distinguish between *unrestricted* and *confidential* statements, e.g., the dark and red arrows in Figure 1(a) and Figure 1(d). DPKGE focuses on protecting confidential statements, using differential privacy to hide their presence (or absence) from the KG. For example, by applying DPKGE on the knowledge graph in 1(a), the similarity between Rose and Sue in the embedding is based on both relations, *citizen of* and *has disease*, and thus, the embedding has a higher quality.

The idea behind our solution to the problem illustrated above is that we can extend existing KG embedding algorithms by introducing the DPSGD method to inject noise in the embedding learning phase, as verified in Section 4. In Figure 2, DPKGE samples confidential statements and unrestricted statements per batch without replacement in a stochastic way by balancing the ratio of the number of sampled unrestricted statements and the number of sampled confidential statements. DPKGE ensures all unrestricted and confidential statements are covered. In the batch of confidential statements, it adds Gaussian noises to the gradients and updates entity embeddings and relation embeddings by optimization. In the batch of unrestricted statements, it follows standard optimization procedures to update the entity embeddings and relation embeddings. We first introduce assumptions on the data and the algorithm. Then, we present DPKGE as a generic framework to create differentially private KG embeddings.

3.1. $\mathcal{C}$-edge-neighboring Knowledge Graphs

Given a KG, we distinguish its content between *confidential* statements, i.e., statements which the data curator wants to keep private, and *unrestricted* statements, i.e., statements which are accessible to everyone.

Definition 4 (Unrestricted and confidential KGs)**.** $\mathcal{K} = \langle\mathcal{U}, \mathcal{C}\rangle$ *is a knowledge graph composed by two disjoint sets of statements* $\mathcal{U}$ *and* $\mathcal{C}$*, i.e.* $\mathcal{U} \cap \mathcal{C} = \emptyset$*, denoting the unrestricted and confidential statements, respectively, i.e.* $\mathcal{K} = \{(h,l,t)|(h,l,t) \in \mathcal{U} \vee (h,l,t) \in \mathcal{C}\}$.

While annotating the statements as confidential or unrestricted can be done manually in small knowledge graphs, this may become an expensive operation when the size increase. One possibility can be to annotate predicates as confidential such that every statement containing this predicate is considered confidential. This could be enabled by introducing privacy-related meta-level properties, similarly to [49].

Based on Definition 4, we focus the privacy-preserving mechanism on the confidential statements. We capture this idea by introducing the notion of $\mathcal{C}$-edge-neighboring knowledge graphs as follows.

Definition 5 ($\mathcal{C}$-edge-neighboring knowledge graphs)**.** *Two knowledge graphs* $\mathcal{K}_1 = \langle\mathcal{U}, \mathcal{C}_1\rangle$ *and* $\mathcal{K}_2 = \langle\mathcal{U}, \mathcal{C}_2\rangle$ *are* $\mathcal{C}$*-edge-neighbor if:*

1. *they differ in one confidential statement* (h,l,t)*, i.e.,* $||\mathcal{C}_1| - |\mathcal{C}_2|| = 1$ *and* $\mathcal{C}_1 \setminus \mathcal{C}_2 \cup \mathcal{C}_2 \setminus \mathcal{C}_1 = \{(h,l,t)\}$*,*
2. *they have the same labels, i.e.,* $\{l \mid \exists(h,l,t) \in \mathcal{K}_1\} = \{l \mid \exists(h,l,t) \in \mathcal{K}_2\}$*, and*
3. *they have the same entities, i.e.,* $\{e \mid \exists(e,l,t) \in \mathcal{K}_1 \vee \exists(h,l,e) \in \mathcal{K}_1\} = \{e \mid \exists(e,l,t) \in \mathcal{K}_2 \vee \exists(h,l,e) \in \mathcal{K}_2\}$.

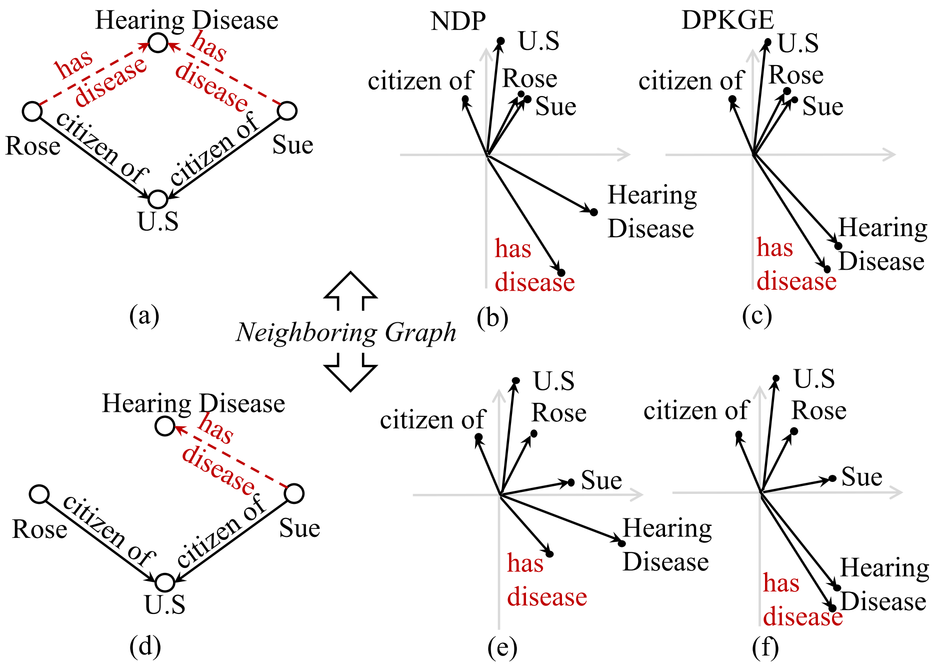


Figure 1: A graph with unrestricted statements (solid black line) and confidential statements (dashed red line).

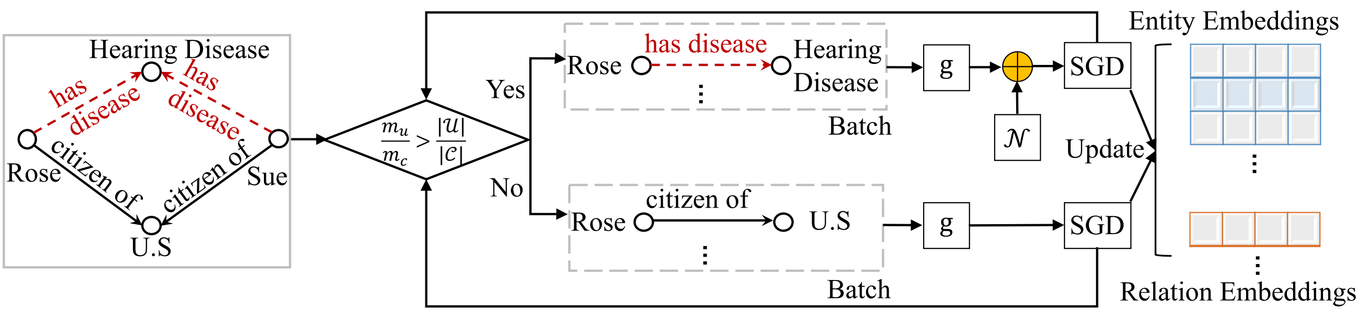


Figure 2: The overall framework of the DPKGE. Edges in $\mathcal{C}$ are indicated using a dashed red arrow, edges in $\mathcal{U}$ as solid black ones. The variables $m_{\mathcal{U}}$, $m_{\mathcal{C}}$, g and $\mathcal{N}$ denote the number of sampled unrestricted statements, the number of sampled confidential statements, gradient and Gaussian noise.

$\mathcal{C}$-edge-neighboring knowledge graphs can be used in hypothetical scenarios to study how a mechanism behaves if a statement is added or removed from a knowledge graph. In differential privacy, we are particularly interested in how the privacy of the statements in the knowledge graph is affected when some statement is added or removed. In practice, such neighboring knowledge graphs occur when the graph evolves over time and statements are added, removed, or changed.

The first condition of Definition 5 is a restriction of edge-neighboring knowledge graphs for confidential statements. When $\mathcal{U} = \emptyset$, i.e., $\mathcal{U}$ is empty, this condition is the same as described in Definition 3. The second and third conditions ensure that the embeddings of neighboring KGs have the same entities and labels. This is necessary because translational embedding methods like TransE and TransM produce embedding vectors for each entity and label in the knowledge graph. Therefore, if two neighboring KGs contain different entities (or labels), one could immediately distinguish which entities (or labels) are involved in the statement that is different between the two neighbors. Fortunately, in large KGs, entities, and relations usually occur multiple times to describe the complex relationships. Therefore, the situation that two neighboring graphs contain different entities (labels) is rare in such KGs.

3.2. Gradient-separable Embedding Algorithm

This study focuses on KG embedding methods that use a variant of the gradient descent method or a gradient-based optimization method (e.g., SGD, Adam, Adagrad). We assume that the gradient descent or gradient-based optimization is the only part of the algorithm which accesses the data to update the embeddings. We call an algorithm that adheres to the aforementioned conditions a *gradient-separable* embedding method because the algorithm can be separated into two parts: the gradient, which affects the embeddings based on the input, and the rest of the algorithm.

Definition 6 (Gradient-separable embedding algorithm). *An embedding algorithm $\mathcal{A}$ is* gradient-separable *if $\mathcal{A}$ contains a gradient descent or gradient-based optimization method ∇ and produces an output O such that:*

1. *O is initialized randomly, and*
2. *O is only updated through ∇.*

We illustrate how this definition applies to RESCAL, TransE, and TransM. RESCAL updates the embedding matrices of entities and relations with either the gradient descent or the alternating least squares method. The version of RESCAL relying on the latter is not gradient-separable. Moreover, RESCAL initializes the matrices either randomly or through the eigen-decomposition of the KG tensor: the latter is not compatible with the gradient-separable definition. Therefore, RESCAL using random initialization and gradient descent is gradient-separable.

TransE is gradient-separable because the embeddings are only updated through the gradient descent method, and the initialization of the embeddings is randomized. The same argument can be made for TransM: The difference between TransE and TransM is that the latter assigns a weight to each statement before updating the embeddings via SGD. Hence, TransM is also gradient-separable.

3.3. The DPKGE Methods

The idea behind the DPKGE methods is that a knowledge graph $\mathcal{K}$ may contain confidential statements, which should be embedded in a privacy-preserving way, and unrestricted statements, which should be embedded by a standard approach. Different KGs may contain different ratios of unrestricted statements $\mathcal{U}$ and confidential statements $\mathcal{C}$. Since each batch can contain, either, only unrestricted statements from $\mathcal{U}$, or, only confidential statements from $\mathcal{C}$, we need a way to make sure that the number of sampled unrestricted statements $m_{\mathcal{U}}$ and the number of sampled confidential statements $m_{\mathcal{C}}$ is as close as possible to the actual ratio of $\mathcal{U}$ and $\mathcal{C}$ in each epoch. This leads to the question at each iteration whether we should sample from $\mathcal{U}$ or $\mathcal{C}$. Therefore, we introduce an adaptive framework which is a stochastic optimization algorithm in which the batch is randomly chosen at each step, and at the same time, the iterated ratio of $m_{\mathcal{U}}$ and $m_{\mathcal{C}}$ is maintained to achieve the actual ratio of $\mathcal{U}$ and $\mathcal{C}$ as close as possible. The experimental results in Section 4 show that the treatment of confidential and unrestricted statements in DPKGE can preserve privacy for the confidential statements while maintaining the utility in many data mining tasks, e.g., link prediction.

Algorithm 1 shows the pseudo-code of how to turn a gradient-separable algorithm into a DPKGE method that is $\mathcal{C}$-edge-differential-private.[6] The two variables $m_{\mathcal{U}}$ and $m_{\mathcal{C}}$ keep track of how many times the algorithm already processed a batch of unrestricted and confidential statements, respectively. These two variables ensure that batches of unrestricted and confidential statements are processed according to the ratio $|\mathcal{U}|/|\mathcal{C}|$. In Lines 4–9, the algorithm checks if it should run a batch of unrestricted or confidential statements, to ensure that $m_{\mathcal{U}}/m_{\mathcal{C}}$ is close to $|\mathcal{U}|/|\mathcal{C}|$. If neither the unrestricted nor the confidential statements are favored, the algorithm picks a batch at random (Line 9). In Lines 10–17, the algorithm calculates the differential private gradient descent. Otherwise, it calculates the ordinary gradient descent without privacy guarantees (Lines 18–22). Note that the `getPositiveAndNegativeSamples` function in lines 11 and 19 samples corrupted statements (h', l, t') for their corresponding positive statements (h, l, t). It means that randomly sampling a head entity h' or a tail entity t' for each relation l as the negative statement for each positive one (h, l, t). The loss function $\mathcal{L}$ is used to optimize the embeddings of the entities and the relations, i.e., parameter θ in Algorithm 1. The loss function differs from different knowledge graph embedding algorithms. The core idea is to use the embeddings of the entities and the relations to model the statement (h, l, t). For example, the loss function of TransE is defined as the sum of the embeddings of h and l minus t.

In this way, different knowledge graph embedding algorithms can be easily plugged in the DPKGE methods. At the same time, different knowledge graphs (KGs) with different ratios of unrestricted statements $\mathcal{U}$ and confidential statements $\mathcal{C}$ can be iterated by a uniform framework in a stochastic optimization way. This is in contrast to [1], where all information is considered confidential, and hence, there is no need to balance the number of sampled unrestricted and confidential information.

3.4. The DPKGE Methods are $\mathcal{C}$-edge-differentially Private

Before discussing the differential privacy properties of our approach, we discuss how applying DPSGD to knowledge graph embeddings differs from the deep learning case as presented in [1]. The main difference between the two settings is that in deep learning, all the neurons in the neural network are updated in each iteration of the algorithm. In contrast, knowledge graph embedding methods update only a subset of all the embeddings which are involved in each iteration. An iteration means a single gradient update of the embeddings of the entities and the relations. The number of iterations equals the number of batches required to pass through all the statements in one epoch. Therefore, DPSGD in KG embeddings should only add

[6] To avoid divisions by zero, all boolean expressions must use short-circuit evaluation.

Algorithm 1: Differentially private knowledge graph embedding

Input : Knowledge graph $\mathcal{K} = \langle \mathcal{U}, \mathcal{C} \rangle$, loss function $\mathcal{L}(\theta)$, learning rate λ, noise multiplier σ, batch size B, norm clipping C

`// Initialize embedding method`
Initialize();
`// Set counters`
$m_{\mathcal{U}}, m_{\mathcal{C}} \leftarrow 0$;
`// Iterate until stopping conditions are met`
Loop
    `// Determine which batch to run`
    if $|\mathcal{U}| = 0 \vee (m_{\mathcal{C}} = 0 \wedge m_{\mathcal{U}} > 0) \vee m_{\mathcal{U}}/m_{\mathcal{C}} > |\mathcal{U}|/|\mathcal{C}|$ **then**
        $batch \leftarrow confidential$;
    else if $|\mathcal{C}| = 0 \vee (m_{\mathcal{U}} = 0 \wedge m_{\mathcal{C}} > 0) \vee m_{\mathcal{U}}/m_{\mathcal{C}} < |\mathcal{U}|/|\mathcal{C}|$ **then**
        $batch \leftarrow unrestricted$;
    else
        $batch \leftarrow \text{Random}(\{confidential, unrestricted\})$;
    if $batch = confidential$ **then**
        `// Optimize confidential statements`
        $T \leftarrow$ getPositiveAndNegativeSamples($\mathcal{C}$, B);
        foreach $i \in T$ **do**
            $\mathrm{g}_i \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, i)$;
            $\bar{\mathrm{g}}_i \leftarrow \mathrm{g}_i / \max\left(1, \frac{\|\mathrm{g}_i\|_2}{C}\right)$;
        $\tilde{\mathrm{g}}_T \leftarrow \frac{1}{b}(\sum_{i \in T} \bar{\mathrm{g}}_i + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$;
        $\theta_{t+1} \leftarrow \theta_t - \lambda \cdot \tilde{\mathrm{g}}_T$;
        $m_{\mathcal{C}} \leftarrow m_{\mathcal{C}} + 1$;
    else
        `// Optimize unrestricted statements`
        $T \leftarrow$ getPositiveAndNegativeSamples($\mathcal{U}$, B);
        $\mathrm{g}_T \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, T)$;
        $\theta_{t+1} \leftarrow \theta_t - \lambda \cdot \mathrm{g}_T$;
        $m_{\mathcal{U}} \leftarrow m_{\mathcal{U}} + 1$;
    `// Update embeddings according to gradient`
    updateEmbeddings(θ_{t+1});

Output: Embeddings

noise to those embeddings which are affected in the current iteration. Another difference is that we distinguish between unrestricted and confidential statements, and only add noise to the latter. This also affects the sampling ratio q in Algorithm 1, which differs from the sampling ratio for the Algorithm in [1]. The sampling ratio q in Algorithm 1, which we denote as q_{DPKGE} to distinguish it from the sampling ratio in [1], is $q_{\text{DPKGE}} = B/|\mathcal{C}|$, where B is the batch size and $|\mathcal{C}|$ is the size of confidential statements. In [1], the sampling ratio, which we denote as q_{DL}, is $q_{\text{DL}} = B/(|\mathcal{U}| + |\mathcal{C}|)$. Due to this difference, we need to further analyze the differential privacy guarantee of the DPSGD when replacing q_{DL} with q_{DPKGE}.

Theorem 2. *DPSGD ∇ in Algorithm 1 is $\mathcal{C}$-edge-differentially private.*

PROOF. According to Theorem 1, there exist constants c_1 and c_2 such that given the sampling probability q_{DL} of the confidential statements and the number of steps T, for any

$$\epsilon < c_1 q_{\text{DL}}^2 T \, , \tag{6}$$

the Algorithm is (ϵ, δ)-differentially private for any $\delta > 0$ if we choose

$$\sigma \geq c_2 \frac{q_{\text{DL}} \sqrt{T \log(1/\delta)}}{\epsilon} \, . \tag{7}$$

The sampling probability q_{DPKGE} of the confidential statements in Algorithm 1 is $q_{\text{DPKGE}} = \frac{B}{|\mathcal{C}|}$. By substituting q_{DL} with q_{DPKGE} we can immediately conclude that Algorithm 1 is (ϵ, δ)-differentially private for any $\delta > 0$ and any $\epsilon < c_1 T \frac{B^2}{|\mathcal{C}|^2}$ if we choose

$$\sigma \geq c_2 \frac{\frac{B}{|\mathcal{C}|} \sqrt{T \log(1/\delta)}}{\epsilon} \, .$$

□

Note that for $|\mathcal{U}| = 0$, we have $q_{\text{DPKGE}} = q_{\text{DL}}$. In this case, any constants c_1 and c_2 satisfying Equations 6 and 7 in the proof of Theorem 2 are also satisfied for the Algorithm in [1] by substituting q_{DPKGE} with q_{DL}. Consequently, they also share the same epsilon bounds which can be obtained by combining Equations 6 and 7 and setting $q_{\text{DPKGE}} = \frac{B}{|\mathcal{C}|}$:

$$c_2 \frac{\frac{B}{|\mathcal{C}|} \sqrt{T \log(1/\delta)}}{\sigma} \leq \epsilon < c_1 T B^2 / |\mathcal{C}|^2 \, . \tag{8}$$

Next, we prove our main theorem, that the DPKGE methods are (ϵ, δ)-$\mathcal{C}$-edge-differentially private.

Theorem 3. *Let $\mathcal{A}$ be a gradient-separable embedding algorithm with a DPSGD ∇ that is (ϵ, δ)-$\mathcal{C}$-edge-differentially private after $n \in \mathbb{N}$ iterations when initialized randomly. Then, $\mathcal{A}$ is (ϵ, δ)-$\mathcal{C}$-edge-differentially private after n iterations.*

PROOF. Let f be the function which maps the outcome from the DPSGD ∇ to the outcome O_i in each iteration $i \in \{1, \ldots, n\}$ of $\mathcal{A}$. Since $\mathcal{A}$ is gradient-separable and hence, O_i is only updated through ∇, it follows that we can write $O_i = f(O_{i-1}, \nabla(\mathcal{K}))$, where $\mathcal{K}$ is a knowledge graph. Fix an arbitrary $E_i \subseteq \text{Range}(f(O_{i-1}, \cdot))$ and let $T_i = \{x \in \text{Range}(\nabla) : f(O_{i-1}, x) \in E_i\}$, then

$$P[f(O_{i-1}, \nabla(\mathcal{K})) \in E_i] = P[\nabla(\mathcal{K}) \in T_i] \, . \tag{9}$$

Since the initial O_0 is initialized randomly, the question of whether $\mathcal{A}$ is $\mathcal{C}$-edge-differentially private can be reduced to whether ∇ is $\mathcal{C}$-edge-differentially private. □

Finally, we observe that the DPKGE methods are not node-differentially private. To see this, we observe that removing a node from the set of confidential statements affects the number of embedding vectors produced by the embedding method. Therefore, it is simple to distinguish between embeddings containing a specific node, and those not containing the node. Hence, the node-differential privacy property is violated.

4. Experiments

This section evaluates the DPKGE methods through an extensive set of experiments.

Interactions between differential privacy and learning. The first two experiments are designed to gain insights into how differential privacy and the learning process affect each other.

The idea of DPKGE is to inject noise to achieve privacy. Such noise affects the learning process, which, in the worst case, may not converge. In the first analysis, described in Section 4.2, we study the loss function for different embedding methods and datasets. We study how the function evolves over time and how it compares to non-differentially private methods.

The second analysis, in Section 4.3, complements the first. In this experiment, we study the impact of the learning process on privacy. In deep learning without differential privacy, where no noise injection is involved, a longer learning process will improve the utility of the learned model. The reason is that model parameters will be adjusted with more iterations to minimize the loss function. However, the introduction of a privacy dimension brings a new metric in addition to utility. Since DPKGE is a learning algorithm that involves noise injection, a longer learning process will yield more injected noise during the training. Intuitively, the longer the learning process, the more information is revealed by the learned model, and consequently, the learned model is more vulnerable to privacy leaks. In this context, learned model refers to the learned embeddings of the entities and the relations as they are optimized in DPKGE. In this experiment, we study how the utility-privacy trade-off evolves over time.

Using differentially private embeddings. The second set of experiments studies the behavior of the DPKGE methods in the context of four applications.

The first application is clustering. Clustering is an unsupervised method to group similar items. As such, it is ideal for inspecting how differential privacy affects the embeddings in the vector space. Training multiple embedding spaces with the same method and parameters should ideally lead to the same clusters. Moreover, the clusters obtained by applying the methods with and without differential privacy on the same dataset should be the same. We describe this analysis in Section 4.4, where we exploit clustering to infer insights on the utility of the embeddings.

In Section 4.5 we discuss the second application, link prediction. The idea of link prediction is to discover new links in a knowledge graph by studying how likely such a link would fit into the embedding of the knowledge graph. However, if the knowledge graph itself is hidden—because of privacy concerns—link prediction can also be exploited trying to reconstruct existing links in a knowledge graph. Hence, if the embedding is DP, link prediction should not perform significantly differently depending on whether a certain link is present in the knowledge graph or not.

In Section 4.6, we introduce an attacker based evaluation. We trace confidential statements by using DPKGE and NDP methods. The evaluation result illustrates the difference between DPKGE and NDP methods when tracing confidential statements by an attacker.

The last analysis, presented in Section 4.7, showcases DPKGE in the context of a case study. We build similar knowledge graphs and study the result of link prediction over them. This anecdotal experiment is useful to illustrate how the DPKGE methods work and how differential privacy affects the resulting models.

In the next sub-section, we discuss the setup of our experiments. This includes the data sets used, the baselines, metrics, parameter settings, and implementation details about our own methods.

4.1. Experimental Setup

In the following, we introduce the datasets, the evaluation metrics, implementation details, the baselines, and parameter settings.

Data sets. We consider five datasets, summarized in Table 1. Three of them, FB15k, FB15k-237 and YAGO 3-10 are de-facto standard datasets to test KG embeddings. *FB15k* is based on Freebase and was initially proposed in [5]. *FB15k-237* is another subset of Freebase built to overcome FB15k limitations. It was initially proposed in [41]. *YAGO3-10* is another KG used to benchmark embedding methods. As the name suggests, YAGO3-10 is a subset of YAGO. FB15k, FB15k-237, and YAGO3-10 do not define confidential statements. To use the DPKGE methods, we randomly set r percent of the statements of the three datasets as confidential, where $r \in \{0, 25, 50, 75, 85, 95, 100\}$. We only set r for FB15k, FB15k-237, and YAGO 3-10 in this way since they do not define any confidential information. To overcome the limitations of randomly defining certain statements as confidential, we also included in our evaluation two real-life datasets in the health domain, eICU and MIMIC-III, that let us define confidential statements in a more natural way as opposed to random selection.

The other two data sets, MIMIC-III and eICU, already include confidential statements. The MIMIC-III dataset is a database about patients admitted to critical care units at a tertiary care hospital [22]. We use Ontop [7] to map the

Table 1: Statistics of the data sets used in the experiments. "Ent." denotes the number of entities, and "Rel." denotes the number of relations.

Data Set	Ent.	Rel.	#Train	#Validate	#Test
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,541	237	272,115	17,535	20,466
YAGO3-10	123,182	37	1,079,040	5,000	5,000
eICU	122,186	16	289,719	29,824	33,724
MIMIC-III	308,878	97	1,482,059	152,565	181,626

dataset to RDF.[7] The resulting graph includes billion of triples, and current implementations of embedding techniques can hardly cope with such a scale - we estimate that TransE would require more than 290 days to learn a model. Therefore, we sample the graph as follows. We select all admissions from January 2150[8], which is around 0.1% of the whole dataset. The resulting knowledge graph contains around 1.8 million statements, so it is in the same order of magnitude as YAGO3-10. Among the 1,482,059 training statements, there are 652,605 confidential statements and 829,454 unrestricted statements.

The eICU dataset is a de-identified database about patients admitted to ICUs across the United States between 2014 to 2015 [34]. As for the MIMIC-III dataset, we use Ontop to map the data into RDF.[7] We randomly select 0.6% of patients' ICU data to obtain a dataset where the number of statements is in the same order of magnitude as FB15k and FB15k-237. The resulting knowledge graph contains around 350 thousand statements. Among the 289,719 training statements, 165,917 are confidential and 123,802 are unrestricted.

Evaluation Metrics. For each test statement, we calculate two ranks with respect to corrupted statements where the head or the tail is replaced with another entity. The two ranks are with respect to the two cohorts of corrupted statements with replaced head and tail, respectively. Finally, the ranks of the correct statements are counted. As in [5], we remove the statements generated in the corruption process that appear in the training, validation, or test, as they are actually correct statements. By keeping them, it is possible they get ranked above the test statements, introducing an error in the evaluation procedure. The average rank of the test statements within its cohort gives the filtered mean rank *MR*. The probability that a test statement is ranked among the ten highest within its cohort gives as the Hits@10 metric (shortly *Hits*). For MR and Hits, we report averages and standard deviations over five runs.

[7] See `https://github.com/xiaolinhan/DPKGE_public.git` for the mapping files.

[8] For privacy protection, all dates are randomly shifted into a date between the year 2100 and 2200

Implementation Details. We consider DPKGE applied to TransE, TransM, RESCAL, and DistMult, which we denote as TransE$_{\text{DPKGE}}$, TransM$_{\text{DPKGE}}$, RESCAL$_{\text{DPKGE}}$, and DistMult$_{\text{DPKGE}}$. We built them using the Pykg2vec [48] and the TensorFlow Privacy[9] libraries. As explained in Sections 3.3 and 3.4, our methods add noises to the entity and relation embeddings of confidential statements contained in each batch.

We ran the link prediction experiments five times and report the average and standard deviation of *MR* and *Hits*.

Baselines. We use three groups of baselines. The first group, referred as the *NDP methods*, includes the state-of-the-art versions of four embedding methods: TransE, TransM, RESCAL, and DistMult.

The second group runs the *NDP methods* on the datasets without confidential statements. We denote such baselines with TransE$^{\mathcal{U}}$, TransM$^{\mathcal{U}}$, RESCAL$^{\mathcal{U}}$ and DistMult$^{\mathcal{U}}$. In the case of FB15k, FB15k237, and YAGO3-10 we set r to 50%, as it is comparable to the ratio of confidential statements in MIMIC-III and eICU – 44.03% and 57.27%, respectively.

The third group, *FullDP methods*, considers naïve differentially private versions of Algorithm 1, which do not distinguish between confidential and unrestricted statements and add noise to everything. We denote the methods in this group as TransE$_{\text{FullDP}}$, TransM$_{\text{FullDP}}$, RESCAL$_{\text{FullDP}}$, and DistMult$_{\text{FullDP}}$.

Parameter Settings. For FB15k, we set the embedding size $k = 50$ for entities and relations representations, the learning rate $\lambda = 0.01$, and margin $\gamma = 1.0$ by following the optimal configurations suggested in [5] for TransE. Following the recommendations in [1], we set the batch size b as $\sqrt{N}$, where N is the number of training statements. We set the number of epochs $l = 100$. The noise multiplier σ can assume values in $\{0.7, 1.0, 1.3, 10.0\}$. In this way, we can observe the effect of different noise degrees on effectiveness, differential privacy, and convergence. We set the σ values as in [1] and include additionally a value of 10.0 to illustrate the impact on extreme choices for σ. The value δ is set as the inverse of the training data size, as suggested in [1, 14]. We conduct hyper-parameter tuning by using a bayesian optimizer for all the methods on all the datasets. The only exception is the hyper-parameter tuning for TransE on FB15k: in this case, we use the hyper-parameters proposed by [5]. The search spaces of the learning rate, hidden size, margin, optimizer and L1 flag are [0.001, 0.1], [50, 512], [0.0, 10.0], {"adam", "adagrad"} and {True, False}, respectively. In the following, we discuss how we set the clipping parameter C.

Setting C. As discussed in Section 2, the parameter C is introduced as part of the differential privacy algorithm,

[9] `https://github.com/tensorflow/privacy`

and it is not present in standard embedding methods. Different choices of C provide different trade-offs between the utility of the embedding algorithm and the surplus of noise added in each iteration. When C is set too small, it limits the utility of the embedding; when it is set too big, it limits the privacy of the embedding.

To find the value of C to be used in the experiment, we study it experimentally. For this, we run different combinations of C and σ to see which C value performs the best. The results of our analysis for different C values with the FB15k-237 dataset are shown in Tables 2 and 3. The best average values are bolded when varying the value of C. We do not report the data about other datasets, as they follow a similar trend for varying C from 20 to 100 percentiles.

We observe that a value of C at the lower end of the distribution of the norms yields the best performance. This indicates that the embedding methods suffer less from gradient clipping than, for example, deep learning in [1]. Therefore, we set the clipping value C at the 20 percentile of the normal distribution of the observed gradients during training as the default setting for C.

4.2. Utility of the DPKGE methods

To study the utility and the convergence of the learning process, we trained embedding models using the DPKGE methods for different parameters on the datasets we introduced in the previous section. We train embedding models with NDP and FullDP methods as two terms of comparison. Fig. 3 shows the trend of the loss functions over time, i.e., epochs. In the case of YAGO3-10 and MIMIC-III, we only show results for $\sigma = 1.0$ because these two large datasets require long training time, e.g., Fig. 3(m) required more than 100 hours of computation.

First, we note that the DPKGE methods (dotted lines in the figure) converge, even if the loss is higher than the NDP baselines (dashed lines). Such a difference can be explained by the fact that the confidential statements introduce noise that disturbs the optimization process. While in this section we consider the utility of the embeddings overall, in Sections 4.5 and 4.7 we break down the analysis to get additional insights on how the methods affect the different statements.

The learning process for FB15k, Fb15k-237, and YAGO3-10 quickly converges for the NDP and DPKGE methods. In the case of MIMIC-III and eICU, the decrease of the loss function is slighter.

When comparing it with FullDP methods (solid lines), we observe that the loss functions of TransE_{FullDP}, TransM_{FullDP}, RESCAL_{FullDP}, and DistMult_{FullDP} decrease slower than the other methods. It is worth noting that in most of the cases, the loss values of the DPKGE methods are closer to the one of the NDP methods than the FullDP ones. This suggests that it is beneficial to focus the injection of noise on those embedding vectors related to the confidential statements.

Table 2: Analysis of C over FB15k-237 on DistMult and TransE. MR and Hits are shown by averages and standard deviation over five runs.

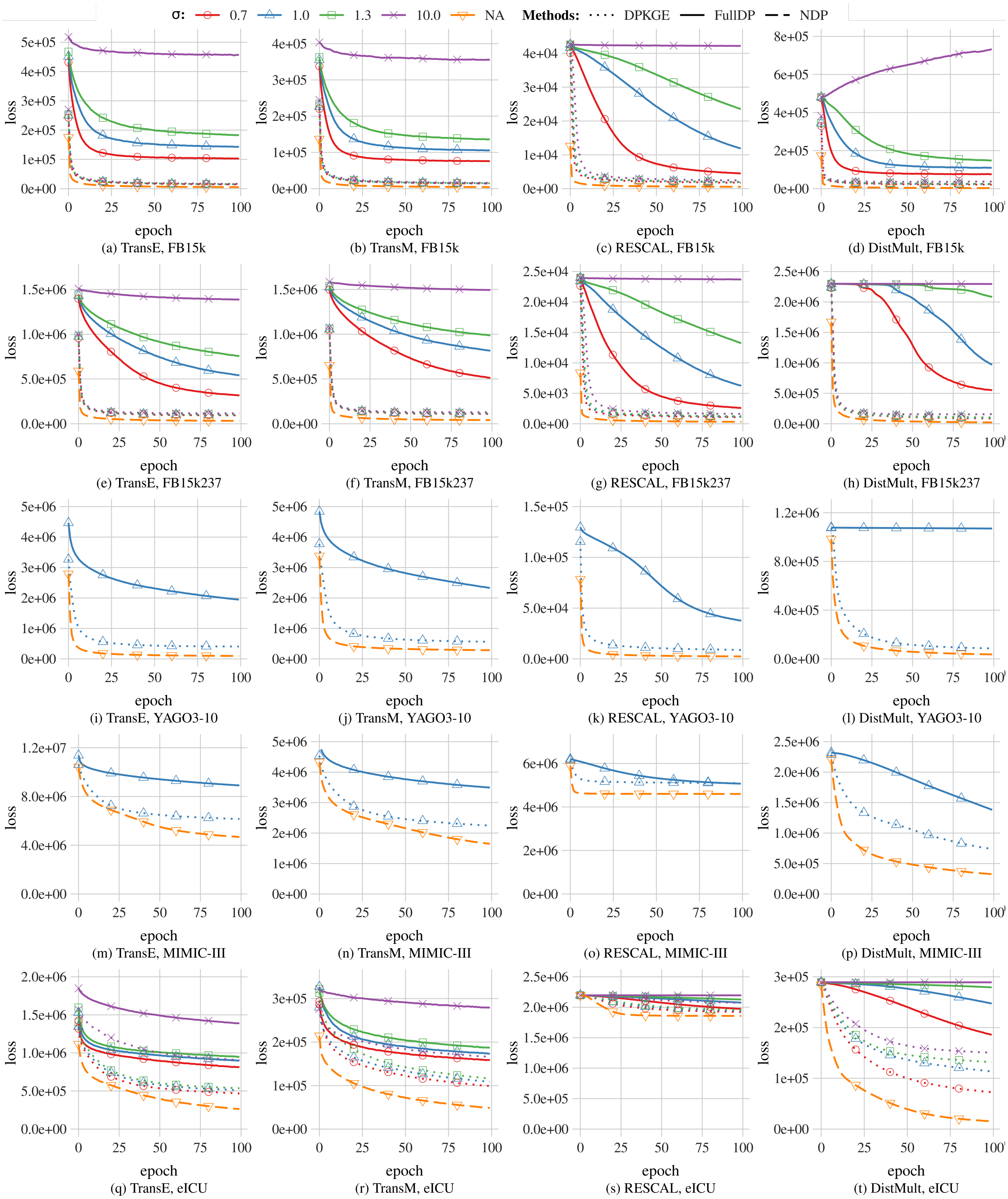
Method	C	σ	MR	Hits
DistMult	20	0.7	**465.81±34.56**	32.60±0.55
		1.0	**507.36±32.08**	31.98±0.85
		1.3	**529.74±26.99**	**32.49±0.45**
		10.0	715.41±25.57	30.97±0.65
	50	0.7	551.06±25.17	32.91±0.68
		1.0	537.43±17.78	**32.39±1.05**
		1.3	562.81±31.42	31.90±0.54
		10.0	721.71±45.21	**31.02±0.72**
	80	0.7	568.86±44.38	**32.95±0.60**
		1.0	596.53±26.95	32.11±0.55
		1.3	600.98±24.32	31.28±0.76
		10.0	**713.97±23.50**	30.54±0.46
	100	0.7	674.90±29.98	31.43±0.65
		1.0	700.29±30.03	31.19±0.72
		1.3	723.76±39.92	30.75±0.69
		10.0	749.89±38.48	30.82±0.80
TransE	20	0.7	**250.45±6.82**	**40.28±0.50**
		1.0	**259.23±8.80**	39.87±0.79
		1.3	**263.46±9.33**	39.47±0.51
		10.0	287.03±4.69	**39.47±0.32**
	50	0.7	262.85±6.61	39.77±0.30
		1.0	261.32±6.43	**40.03±0.40**
		1.3	269.26±9.22	**39.83±0.96**
		10.0	**280.44±16.80**	38.98±0.25
	80	0.7	273.11±12.73	39.47±0.99
		1.0	271.29±6.60	39.21±0.39
		1.3	268.90±4.79	39.34±0.64
		10.0	290.32±8.88	38.86±0.47
	100	0.7	293.15±5.99	38.75±0.84
		1.0	278.00±13.02	39.00±0.41
		1.3	295.95±3.91	38.66±0.35
		10.0	290.55±12.21	38.65±0.53

The DPKGE methods are less affected by σ than FullDP ones. This can be explained by the fact that the unrestricted statements help stabilize the convergence of the embedding – an effect we hoped to achieve by distinguishing between confidential and unrestricted statements.

The FullDP methods also offer interesting insights on the learning process when all the statements in the knowledge graph are confidential. In general, we observe that the learning process converges for values σ lower than 10. The only exception is DistMult_{FullDP} in Figure 3(l), where the loss value does not visibly decrease. When σ is 10, the loss function generally decreases very slowly or is almost constant. In the case of Figure 3(d), however, the loss increases over time. These results suggest that high values of σ are not useful when there the number of sensitive statements is very high.

4.3. Privacy of the DPKGE methods

In the previous experiment, we studied how differential privacy affects the learning process. We now analyze



(a) TransE, FB15k (b) TransM, FB15k (c) RESCAL, FB15k (d) DistMult, FB15k
(e) TransE, FB15k237 (f) TransM, FB15k237 (g) RESCAL, FB15k237 (h) DistMult, FB15k237
(i) TransE, YAGO3-10 (j) TransM, YAGO3-10 (k) RESCAL, YAGO3-10 (l) DistMult, YAGO3-10
(m) TransE, MIMIC-III (n) TransM, MIMIC-III (o) RESCAL, MIMIC-III (p) DistMult, MIMIC-III
(q) TransE, eICU (r) TransM, eICU (s) RESCAL, eICU (t) DistMult, eICU

Figure 3: Loss plots for TransE, TransM, RESCAL and DistMult using the five datasets. For the DPKGE methods, in FB15k, FB15k237, and YAGO3-10, 50% of the statements are set as confidential; MIMIC-III and eICU already include 44.03% and 57.27% confidential statements, respectively.

Table 3: Analysis of C over FB15k-237 on TransM and RESCAL. MR and Hits are shown by averages and standard deviation over five runs.

Method	C	σ	MR	Hits
TransM	20	0.7	**238.18±6.05**	**41.47±0.46**
		1.0	**249.01±4.29**	**40.88±0.69**
		1.3	**254.98±6.13**	**41.30±0.31**
		10.0	**260.65±9.04**	40.66±0.15
	50	0.7	245.33±9.02	41.13±0.51
		1.0	257.76±5.42	40.71±1.12
		1.3	258.78±3.28	40.72±0.94
		10.0	261.72±12.78	40.26±0.39
	80	0.7	255.50±5.38	40.77±0.49
		1.0	262.92±8.76	40.65±0.71
		1.3	268.96±4.60	40.48±0.50
		10.0	260.67±8.22	**41.00±0.53**
	100	0.7	271.05±8.63	40.28±0.61
		1.0	269.53±6.14	40.35±0.51
		1.3	276.16±11.45	40.25±0.35
		10.0	268.63±3.97	40.25±0.37
RESCAL	20	0.7	**389.18±37.30**	**35.87±0.46**
		1.0	**421.87±20.58**	**35.41±0.57**
		1.3	**426.68±16.43**	**34.75±0.66**
		10.0	**547.41±31.44**	**33.61±0.60**
	50	0.7	415.09±13.12	35.44±0.23
		1.0	467.01±10.64	34.20±0.48
		1.3	470.47±24.42	34.65±0.37
		10.0	557.22±40.19	33.41±0.38
	80	0.7	436.41±54.19	34.60±0.47
		1.0	478.58±18.05	34.39±0.53
		1.3	479.17±27.56	34.06±0.36
		10.0	576.07±32.24	32.66±0.77
	100	0.7	541.70±33.69	33.37±0.56
		1.0	556.85±34.71	33.24±0.50
		1.3	602.73±29.88	33.44±0.21
		10.0	569.41± 14.26	32.92±0.39

the vice versa: how the learning process affects differential privacy. The privacy budget ε is a useful value to quantify the risk of privacy leaks: the lower its value, the less probable a privacy leak may happen. As explained in Section 3.3, in the context of DPKGE, the differentially private SGD is controlled through a parameter σ, whereas, ε is estimated through the accountant. It follows that it is not straightforward to determine how the learning process affects ε.

We tracked the value of ε for the DPKGE and FullDP methods on the five datasets. The results are reported in Figure 4: as the trends are similar for the different combinations of methods and datasets, the figure shows the behaviour of TransE$_{\text{DPKGE}}$ and TransE$_{\text{FullDP}}$ on FB15k and YAGO3-10. The plots show that ε increases during the training. This is because each epoch in the training phase potentially leaks additional information into the embeddings, which can be exploited to reconstruct the original dataset.

Figure 4(a) reports the performance of DPKGE and

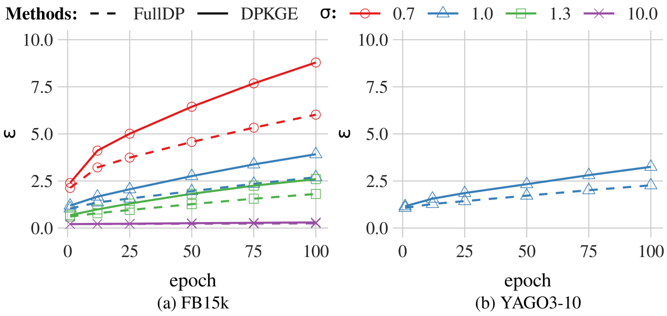


Figure 4: ϵ over epochs of TransE on FB15k and YAGO3-10

FullDP TransE for different values of σ. We observe that in all the methods, after a ca. 10 epochs, ϵ grows linearly. As σ decreases, the growth of ε becomes steeper. Moreover, the difference between DPKGE and FullDP increases as well.

When σ is 10, ε has an almost constant value close to 0 for both DPKGE and FullDP. This suggests that the learning process has strong privacy protection. Even if the two values are similar, we observe two different behavior of the loss function in Figure 3(a). In the case of DPKGE, the loss decreases similarly to the ones associated with other values of σ. In the case of FullDP, the loss decreases very slowly, suggesting that the resulting model has very low utility. This confirms that unrestricted statements play a key role in the overall quality of the learned embedding model.

Comparing the plots in Figures 4(a) and 4(b), we observe that the behaviour of DPKGE and FullDP for σ set to 1 is almost identical. It means that ε is not affected by the size of the dataset.

Finally, we note that whereas more epochs can yield a lower value for the loss function (c.f. Figures 3(a) and 3(i)), each epoch increases the ε value for the differential privacy (c.f. Figure 4). It is worth noting that while ε increases linearly, the *loss* decreases exponentially. This suggests that the length of the training can be tuned to maximize the privacy-utility trade-off.

4.4. Clustering

We conduct a clustering task to show that DPKGE can preserve the similarity among embeddings of entities such as the ones illustrated in Figure 1. We use 762 patients in the testing data of eICU, and follow the clustering method in [31], i.e., k-means, to evaluate the similarity among embeddings of patients. We compute five different embedding models by applying TransE$_{\text{NDP}}$ on the KG $\mathcal{K}_{\text{eICU}}$ with both unrestricted and confidential statements. We also compute five models by applying TransE$_{\text{DPKGE}}$ with $\sigma = 0.7$ on the same KG $\mathcal{K}_{\text{eICU}}$. Finally, we compute five different embedding models by applying TransE$_{\text{NDP}}$ on the KG $\mathcal{U}_{\text{eICU}}$ with only the unrestricted statements.

Next, we apply the k-means algorithm with $k \in [2, 4]$ to each embedding model. We clustered when k equals 2,

Table 4: NMI among (C_{NDP_i}, C_{DPKGE_i}) pairs when k=2,3,4

		k	ave	stddev
$\mathcal{C}_{\mathrm{NDP}}$	$\mathcal{C}_{\mathrm{DPKGE}}$	2	**0.96**	0.02
$\mathcal{C}_{\mathrm{NDP}}$	$\mathcal{C}_{\mathrm{U\text{-}NDP}}$	2	0.02	0.01
$\mathcal{C}_{\mathrm{NDP}}$	$\mathcal{C}_{\mathrm{DPKGE}}$	3	**0.34**	0.02
$\mathcal{C}_{\mathrm{NDP}}$	$\mathcal{C}_{\mathrm{U\text{-}NDP}}$	3	0.04	0.03
$\mathcal{C}_{\mathrm{NDP}}$	$\mathcal{C}_{\mathrm{DPKGE}}$	4	**0.28**	0.02
$\mathcal{C}_{\mathrm{NDP}}$	$\mathcal{C}_{\mathrm{U\text{-}NDP}}$	4	0.05	0.02

3, and 4 in Table 4. As the largest average Normalised Mutual Information (NMI) score when k equals 4 is already low, i.e., 0.28, we stopped searching for $k > 4$. We obtain 45 clustering results of the patients. The first 15 clustering results are from TransE$_{\mathrm{NDP}}$ on $\mathcal{K}_{\mathrm{eICU}}$ and denoted as the set $\mathcal{C}_{\mathrm{NDP}} = \{C^{k,i}_{\mathrm{NDP}} | k \in [2,4] \wedge i \in [1,5]\}$, where $C^{k,i}_{\mathrm{NDP}}$ is the i-th clustering of TransE$_{\mathrm{NDP}}$ on $\mathcal{K}_{\mathrm{eICU}}$ with k clusters. Similar, the next 15 results are from TransE$_{\mathrm{DPKGE}}$ on $\mathcal{K}_{\mathrm{eICU}}$ and are denoted as $\mathcal{C}_{\mathrm{DPKGE}} = \{C^{k,i}_{\mathrm{DPKGE}} | k \in [2,4] \wedge i \in [1,5]\}$. Finally, the last 15 results are from TransE$_{\mathrm{NDP}}$ on $\mathcal{U}_{\mathrm{eICU}}$ and are denoted as $\mathcal{C}_{\mathrm{U\text{-}NDP}} = \{C^{k,i}_{\mathrm{U\text{-}NDP}} | k \in [2,4] \wedge i \in [1,5]\}$, i.e., they only consider the unrestricted statements.

The clustering results $\mathcal{C}_{\mathrm{NDP}}$ are used as the gold standard for the similarity among patients. If a clustering is similar to the gold standard, we conclude that the clustering also preserves a good similarity among patients. To calculate the similarity between clustering results from $\mathcal{C}_{\mathrm{NDP}}$ and $\mathcal{C}_{\mathrm{DPKGE}}$, and from $\mathcal{C}_{\mathrm{NDP}}$ and $\mathcal{C}_{\mathrm{U\text{-}NDP}}$, we use the NMI score.

Table 4 reports the average (*ave*) and standard deviation (*stddev*) of the NMI values between the clustering results over five runs. The best average values are bolded in Table 4. The NMI scores for each pair of clustering results are available in Appendix Appendix B. We observe that the clustering results from $\mathcal{C}_{\mathrm{NDP}}$ are more similar to the ones from $\mathcal{C}_{\mathrm{DPKGE}}$ than the ones from $\mathcal{C}_{\mathrm{U\text{-}NDP}}$. The NMI between clustering results from $\mathcal{C}_{\mathrm{NDP}}$ and $\mathcal{C}_{\mathrm{DPKGE}}$ is 0.96 when $k = 2$, and around 0.3 when k is 3 and 4. The NMI between clustering results from $\mathcal{C}_{\mathrm{NDP}}$ and $\mathcal{C}_{\mathrm{U\text{-}NDP}}$, in comparison, are 0.02, 0.04 and 0.05 for k equals 2, 3 and 4. Hence, TransE$_{\mathrm{DPKGE}}$ can preserve higher similarity to the gold-standard than standard TransE when removing restricted statements.

Moreover, the low standard deviation we observe suggests that the stochastic elements involved in the process, e.g., the computation of the embedding models or the cluster initialization for k-means, have a limited impact on the results.

4.5. Effectiveness of Link Prediction

In this experiment, we investigate if focusing the DP computation only on confidential statements yields embeddings with higher utility. We report on our results in Tables 5, 6, 7, 8, 9, and Figure 5. Similarly to the experiment in Section 4.2, σ is set to $\{0.7, 1.0, 1.3, 10.0\}$ for the methods on FB15k, FB15k-237, eICU), and σ is set to 1.0 in the case of YAGO3-10 and MIMIC-III due to the computational time to train the models on these two datasets.

As the YAGO3-10 and MIMIC-III datasets are extremely large, we only consider the case where σ is 1.0. Tables 5, 6, 7, 8 and 9 show the average and standard deviation of MR and $Hits$ for five runs on the different methods when C equals 20 percentile of the gradient normal distribution. For FB15k, FB15k-237, and YAGO3-10, we set $r = 50\%$, while for eICU and MIMIC-III, we use the actual confidential statements in the datasets as restricted statements. Note that, however, the NDP and FullDP methods do not distinguish between confidential and unrestricted statements and, hence, the effectiveness of these methods applies to any setting with arbitrary r value. This is denoted in the table with the “any” value in the r column.

Compared with NDP methods applied on KGs removing confidential statements, denoted as TransE$^{\mathcal{U}}$, TransM$^{\mathcal{U}}$, RESCAL$^{\mathcal{U}}$ and DistMult$^{\mathcal{U}}$, the DPKGE methods outperform in most of settings on five datasets. It confirms the hypothesis that removing confidential statements would degrade the performance in the view of utility.

The tables also show the ϵ values at the 100^{th} epoch estimated through the accountant [1]. The DPKGE methods have a slightly higher ϵ compared to the FullDP methods on the same knowledge graph embedding algorithms and the same datasets. This does not necessarily mean that FullDP methods are more private, however, as the accountant, which estimates ϵ, only provides an upper bound for ϵ. This means that the real value for ϵ could be lower than estimated by the method and, consequently, the real ϵ for the FullDP methods is not necessarily lower than for the DPKGE methods.

We believe that this estimation is less tight for the DPKGE methods because the accountant considers only the confidential statements. This means that from the accountant's point of view, the whole KG is smaller than it actually is, i.e., when $r = 50\%$, the accountant believes that the KG size is 50% smaller. Smaller KGs are more difficult to keep private than larger ones, as each statement has a larger impact on the embedding result. This can also be observed when comparing the ϵ values of the smaller FB15k dataset with the larger YAGO3-10 dataset.

As for the loss functions in Section 4.2, the DPKGE methods have MR and Hits values which are much closer to the NDP methods than the FullDP ones in most cases. Most notably, the MR and Hits values of RESCAL do not differ much between the DPKGE and NDP versions. Also, whereas the Hits values are in the same order of magnitude for the DPKGE and NDP methods, the Hits of FullDP significantly drop one to several orders of magnitude. For RESCAL and DistMult, the Hits even drop to zero or close to zero for the FullDP methods.

In Table 8 and 9, the DPKGE methods show better MR and Hits values than the FullDP ones, in general. Specifi-

Table 5: Performance of link prediction over FB15k. MR and Hits are shown by averages and standard deviation over five runs. Since the NDP methods do not guarantee any differential privacy, ϵ and σ are set as "–". The best average values are bolded in each cell.

Method	σ	r	MR	Hits	ϵ
TransE			**88.15±4.07**	**51.47±1.05**	
TransE$^{\mathcal{U}}$	–	–	245.37±7.44	35.55±0.88	–
TransE$_{\text{DPKGE}}$		50%	**267.48±15.23**	**35.35±1.11**	8.79
TransE$_{\text{FullDP}}$	0.7	any	1307.26±35.97	10.01±0.69	6.02
TransE$_{\text{DPKGE}}$		50%	**286.21±17.99**	**34.28±1.06**	3.92
TransE$_{\text{FullDP}}$	1.0	any	1978.67±21.82	7.29±0.62	2.7
TransE$_{\text{DPKGE}}$		50%	**293.14±20.59**	**34.68±1.32**	2.60
TransE$_{\text{FullDP}}$	1.3	any	2510.84±94.11	5.82±0.26	1.81
TransE$_{\text{DPKGE}}$		50%	**300.57±20.62**	**34.17±1.77**	0.30
TransE$_{\text{FullDP}}$	10.0	any	6390.88±52.02	0.82±0.16	0.25
TransM			**84.29±4.31**	**52.69±1.21**	
TransM$^{\mathcal{U}}$	–	–	236.68±15.02	36.01±0.65	–
TransM$_{\text{DPKGE}}$		50%	**216.41±13.73**	**39.78±0.76**	8.79
TransM$_{\text{FullDP}}$	0.7	any	1207.41±68.77	10.81±0.39	6.02
TransM$_{\text{DPKGE}}$		50%	**223.53±17.29**	**38.49±0.51**	3.92
TransM$_{\text{FullDP}}$	1.0	any	1705.77±40.86	7.99±0.41	2.7
TransM$_{\text{DPKGE}}$		50%	**234.90±8.71**	**38.62±1.04**	2.60
TransM$_{\text{FullDP}}$	1.3	any	2244.03±99.44	6.54±0.49	1.81
TransM$_{\text{DPKGE}}$		50%	**250.71±13.00**	**38.01±1.01**	0.30
TransM$_{\text{FullDP}}$	10.0	any	6303.84±196.43	0.94±0.20	0.25
RESCAL			**116.24±5.71**	**46.37±0.35**	
RESCAL$^{\mathcal{U}}$	–	–	349.28±33.94	33.70±0.53	–
RESCAL$_{\text{DPKGE}}$		50%	**265.60±11.78**	**34.97±1.09**	8.79
RESCAL$_{\text{FullDP}}$	0.7	any	617.71±26.98	27.71±0.49	6.02
RESCAL$_{\text{DPKGE}}$		50%	**290.77±8.14**	**34.83±0.65**	3.92
RESCAL$_{\text{FullDP}}$	1.0	any	1544.82±109.59	22.41±0.77	2.7
RESCAL$_{\text{DPKGE}}$		50%	**310.49±14.12**	**34.84±0.64**	2.60
RESCAL$_{\text{FullDP}}$	1.3	any	3735.61±216.61	12.55±1.33	1.81
RESCAL$_{\text{DPKGE}}$		50%	**376.20±22.79**	**33.99±0.49**	0.30
RESCAL$_{\text{FullDP}}$	10.0	any	7296.84±41.68	0.09±0.09	0.25
DistMult			**147.93±11.09**	**48.34±0.89**	
DistMult$^{\mathcal{U}}$	–	–	515.61±17.22	31.85±0.82	–
DistMult$_{\text{DPKGE}}$		50%	**393.51±36.80**	**28.75±1.59**	8.79
DistMult$_{\text{FullDP}}$	0.7	any	1080.28±34.87	7.19±0.72	6.02
DistMult$_{\text{DPKGE}}$		50%	**443.72±40.37**	**29.12±0.81**	3.92
DistMult$_{\text{FullDP}}$	1.0	any	1459.05±27.37	6.41±0.40	2.7
DistMult$_{\text{DPKGE}}$		50%	**464.93±18.07**	**28.42±1.28**	2.60
DistMult$_{\text{FullDP}}$	1.3	any	1897.72±66.78	4.51±0.65	1.81
DistMult$_{\text{DPKGE}}$		50%	**584.29±32.05**	**27.40±0.75**	0.30
DistMult$_{\text{FullDP}}$	10.0	any	7378.43±139.39	0.23±0.04	0.25

cally, the MR and Hits values remain stable with σ equals 1.3 for DPKGE, but drop significantly for FullDP in Table 8.

Impact of σ on Effectiveness. The effectiveness of the DPKGE methods for $r = 50\%$ is not much affected for values $\sigma \in \{0.7, 1.0, 1.3\}$, which are the values used in [1]. However, when $\sigma = 10$, there is a noticeable drop in MR and Hits. To explain the good effect of the DPKGE methods for $r = 50\%$ (even for large σ), we note that only half of the statements in the KG are affected by σ. Hence, to a certain degree, even a large amount of noise can be compensated by unrestricted statements. At the same time, higher σ values have a positive effect on ϵ. For the FullDP methods, the impact of σ is more pronounced. Even for low values, we can already observe a significant drop in effectiveness. Hence, this experiment suggests that the number of confidential statements in the KG should be taken into account while setting σ.

Effectiveness when Varying r. Figure 5 shows ϵ, MR, Hits, and for different percentages of r. The ϵ value decreases when r increases. The plot is the same for each DPKGE method because the accountant is affected by σ and by

Table 6: Performance of link prediction over YAGO3-10. MR and Hits are shown by averages and standard deviation over five runs. Since the NDP methods do not guarantee any differential privacy, ϵ and σ are set as "–". The best average values are bolded in each cell.

Method	σ	r	MR	Hits	ϵ
TransE			**991.84±69.20**	**22.29±0.35**	
TransE$^{\mathcal{U}}$	–	–	3531.16±242.55	17.79±0.54	–
TransE$_{\text{DPKGE}}$		50%	**2034.91±102.30**	**18.76±0.59**	3.25
TransE$_{\text{FullDP}}$	1.0	any	10300.87±295.83	8.06±0.37	2.27
TransM			**1061.27±34.28**	**18.16±0.51**	
TransM$^{\mathcal{U}}$	–	–	2877.73±109.58	15.39±0.69	–
TransM$_{\text{DPKGE}}$		50%	**1803.95±133.14**	**15.95±0.46**	3.25
TransM$_{\text{FullDP}}$	1.0	any	8396.76±205.86	10.34±0.61	2.27
RESCAL			**4255.91±521.06**	**10.74±1.15**	
RESCAL$^{\mathcal{U}}$	–	–	7812.01±314.68	8.35±0.39	–
RESCAL$_{\text{DPKGE}}$		50%	**5382.13±539.47**	**8.73±0.63**	3.25
RESCAL$_{\text{FullDP}}$	1.0	any	17849.35±1006.09	4.70±0.16	2.27
DistMult			**2622.48±186.67**	**10.54±1.15**	
DistMult$^{\mathcal{U}}$	–	–	5285.32±249.43	8.02±0.55	–
DistMult$_{\text{DPKGE}}$		50%	**3212.65±254.10**	**11.18±0.55**	3.25
DistMult$_{\text{FullDP}}$	1.0	any	58610.68±1619.00	0.29±0.19	2.27

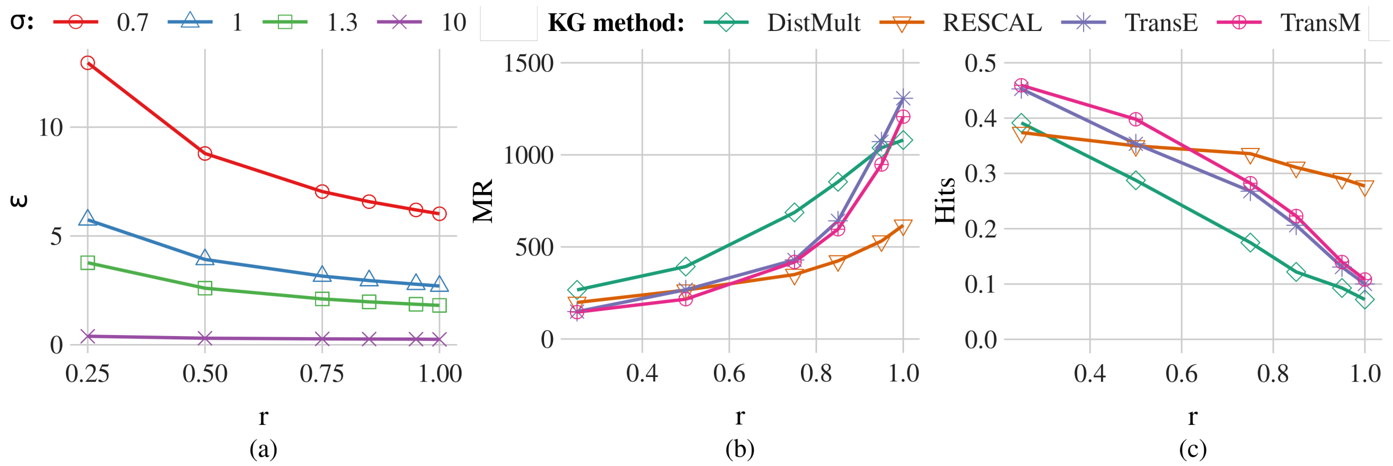


Figure 5: Privacy budget ϵ and performance in the link prediction task on FB15k when r varies. In (a), the plot is the same for each DPKGE method. In (b) and (c), σ is set to 0.7.

the number of confidential batches, but not by the embedding methods themselves. It is worth stressing that, as discussed before, the plot does not imply that with higher r, the methods have stronger privacy guarantees, as the epsilon estimation is only an upper bound which does not take into account the unrestricted statements.

When r increases, MR values increase as well, while Hits values decrease. It follows that the utility of the embeddings in the link prediction task decreases overall. Figure 5(b) and (c) show that the biggest drop in utility occurs when r is above about 80%, suggesting that it is important to have unrestricted statements in the dataset to build the embedding space. Moreover, the performance of TransE and TranM drop heavier than RESCAL and DistMult when r increases from 80% to 100%.

4.6. Attacker based Evaluation

The goal of the following attacker-based evaluation is to study how the DPKGE methods hinder an attacker in judging whether specific statements are part of the knowledge graph or not. If an attacker is tasked in revealing confidential statements, one common approach would be to calculate the rank of different tail entities with the help of a link prediction method. Given a head entity h and a relation l, tail entities t with a lower (better) rank are considered more likely to form a statement (h, l, t) that is part

[10] /people/person/profession

[11] /award/award_nominee/award_nominations./award/award_nomination/award_nominee

[12] /award/award_nominated_work/award_nominations./award/award_nomination/award

Table 7: Performance of link prediction over FB15k-237. MR and Hits are shown by averages and standard deviation over five runs. Since the NDP methods do not guarantee any differential privacy, ϵ and σ are set as "–". The best average values are bolded in each cell.

Method	σ	r	MR	Hits	ϵ
TransE			**179.01±7.30**	**44.79±0.89**	
TransE$^{\mathcal{U}}$	–	–	346.85±9.75	31.03±0.65	–
TransE$_{\text{DPKGE}}$		50%	**250.45±6.82**	**40.28±0.49**	10.08
TransE$_{\text{FullDP}}$	0.7	any	639.19±33.80	32.19±0.42	6.86
TransE$_{\text{DPKGE}}$		50%	**259.23±8.79**	**39.86±0.78**	4.49
TransE$_{\text{FullDP}}$	1.0	any	1045.20±18.84	29.92±0.61	3.08
TransE$_{\text{DPKGE}}$		50%	**263.45±9.33**	**39.46±0.51**	2.96
TransE$_{\text{FullDP}}$	1.3	any	1705.34±59.04	28.81±0.40	2.05
TransE$_{\text{DPKGE}}$		50%	**287.03±4.69**	**39.46±0.31**	0.32
TransE$_{\text{FullDP}}$	10.0	any	5311.87±98.10	8.30±0.20	0.26
TransM			**176.23±7.82**	**45.34±0.25**	
TransM$^{\mathcal{U}}$	–	–	316.83±7.54	32.41±0.31	–
TransM$_{\text{DPKGE}}$		50%	**238.17±6.04**	**41.47±0.45**	10.08
TransM$_{\text{FullDP}}$	0.7	any	781.76±19.57	32.97±0.54	6.86
TransM$_{\text{DPKGE}}$		50%	**249.01±4.29**	**40.87±0.69**	4.49
TransM$_{\text{FullDP}}$	1.0	any	1420.87±80.00	30.85±0.71	3.08
TransM$_{\text{DPKGE}}$		50%	**254.98±6.12**	**41.30±0.30**	2.96
TransM$_{\text{FullDP}}$	1.3	any	2279.18±70.83	28.79±0.49	2.05
TransM$_{\text{DPKGE}}$		50%	**260.64±9.04**	**40.65±0.14**	0.32
TransM$_{\text{FullDP}}$	10.0	any	5330.69±144.14	8.58±0.23	0.26
RESCAL			**328.73±29.59**	**39.88±0.27**	
RESCAL$^{\mathcal{U}}$	–	–	626.00±60.85	29.06±0.71	–
RESCAL$_{\text{DPKGE}}$		50%	**389.18±37.30**	**35.86±0.45**	10.08
RESCAL$_{\text{FullDP}}$	0.7	any	691.48±25.19	30.63±0.42	6.86
RESCAL$_{\text{DPKGE}}$		50%	**421.86±20.58**	**35.41±0.56**	4.49
RESCAL$_{\text{FullDP}}$	1.0	any	1613.17±108.78	24.08±0.67	3.08
RESCAL$_{\text{DPKGE}}$		50%	**426.68±16.43**	**34.74±0.66**	2.96
RESCAL$_{\text{FullDP}}$	1.3	any	3474.53±243.40	13.04±0.90	2.05
RESCAL$_{\text{DPKGE}}$		50%	**547.41±31.44**	**33.60±0.60**	0.32
RESCAL$_{\text{FullDP}}$	10.0	any	7165.76±91.82	0.03±0.02	0.26
DistMult			**422.37±16.74**	**34.40±0.24**	
DistMult$^{\mathcal{U}}$	–	–	752.85±60.14	22.35±0.83	–
DistMult$_{\text{DPKGE}}$		50%	**465.80±34.56**	**32.60±0.54**	10.08
DistMult$_{\text{FullDP}}$	0.7	any	1544.88±81.16	10.37±0.55	6.86
DistMult$_{\text{DPKGE}}$		50%	**507.36±32.07**	**31.97±0.84**	4.49
DistMult$_{\text{FullDP}}$	1.0	any	3611.03±531.27	9.43±0.59	3.08
DistMult$_{\text{DPKGE}}$		50%	**529.74±26.99**	**32.48±0.44**	2.96
DistMult$_{\text{FullDP}}$	1.3	any	6245.80±297.08	4.28±1.83	2.05
DistMult$_{\text{DPKGE}}$		50%	**715.40±25.56**	**30.96±0.64**	0.32
DistMult$_{\text{FullDP}}$	10.0	any	7066.83±100.42	0.45±0.08	0.26

of the original KG than tail entities with a higher (worse) rank. Thus, in order to understand how the DPKGE methods affect the success of such an attack, we study how the ranks of tail entities of confidential statements are affected by the DPKGE methods.

To conduct this study, we first need to change our viewpoint to the one of the attacker. We assume that the attacker has access to the embeddings of the KG but not the knowledge graph itself. This means that the attacker knows the entities and relations that are part of the KG, but not which statements (i.e., combinations of entities and relations) are part of it. In our attacker scenario, the attacker chooses a head $h \in \mathcal{E}$ and a relation $l \in \mathcal{L}$ and tries to deduce whether for a specific tail entity t the statement (h, l, t) is part of the knowledge graph or not. We will call *candidate statement* the statement (h, l, t) for which an at-

Table 8: Performance of link prediction over eICU. MR and Hits are shown by averages and standard deviation over five runs. Since the NDP methods do not guarantee any differential privacy, ϵ and σ are set as "–". The best average values are bolded in each cell.

Method	σ	r	MR	Hits	ϵ
TransE			**14662.63±149.68**	14.30±0.59	
TransE$^{\mathcal{U}}$	–	–	31012.00±730.35	**15.88±0.43**	–
TransE$_{\text{DPKGE}}$		57.27%	**19251.74±209.56**	**33.78±0.75**	9.21
TransE$_{\text{FullDP}}$	0.7	any	19917.49±208.57	25.40±0.54	6.77
TransE$_{\text{DPKGE}}$		57.27%	**19366.46±271.27**	**34.10±0.38**	4.11
TransE$_{\text{FullDP}}$	1.0	any	21393.56±646.72	25.48±0.91	3.04
TransE$_{\text{DPKGE}}$		57.27%	**20069.86±365.55**	**33.62±1.23**	2.72
TransE$_{\text{FullDP}}$	1.3	any	22298.05±476.42	25.54±0.59	2.03
TransE$_{\text{DPKGE}}$		57.27%	28902.93±2364.13	11.98±2.08	0.30
TransE$_{\text{FullDP}}$	10.0	any	**26707.12±334.47**	**23.48±0.19**	0.26
TransM			**21540.11±635.02**	**9.83±0.69**	
TransM$^{\mathcal{U}}$	–	–	30209.67±291.58	9.61±0.63	–
TransM$_{\text{DPKGE}}$		57.27%	**23926.09±574.16**	9.54±0.47	9.21
TransM$_{\text{FullDP}}$	0.7	any	25960.80±740.67	**12.19±0.99**	6.77
TransM$_{\text{DPKGE}}$		57.27%	**24169.05±469.47**	10.39±1.20	4.11
TransM$_{\text{FullDP}}$	1.0	any	26876.66±645.67	**10.54±1.36**	3.04
TransM$_{\text{DPKGE}}$		57.27%	**25517.92±964.13**	**9.43±1.63**	2.72
TransM$_{\text{FullDP}}$	1.3	any	27781.02±466.43	8.65±0.61	2.03
TransM$_{\text{DPKGE}}$		57.27%	**33833.45±2360.65**	**3.59±0.33**	0.30
TransM$_{\text{FullDP}}$	10.0	any	42986.19±2238.23	0.09±0.07	0.26
RESCAL			**18211.51±97.83**	**28.11±1.76**	
RESCAL$^{\mathcal{U}}$	–	–	42585.14±1182.17	19.52±0.96	–
RESCAL$_{\text{DPKGE}}$		57.27%	**23818.40±328.32**	**30.63±1.35**	9.21
RESCAL$_{\text{FullDP}}$	0.7	any	32489.87±510.57	27.01±1.27	6.77
RESCAL$_{\text{DPKGE}}$		57.27%	**24423.61±356.99**	**31.14±0.35**	4.11
RESCAL$_{\text{FullDP}}$	1.0	any	36043.25±610.17	18.75±1.67	3.04
RESCAL$_{\text{DPKGE}}$		57.27%	**24608.88±434.08**	**30.47±0.35**	2.72
RESCAL$_{\text{FullDP}}$	1.3	any	36974.35±490.41	12.39±1.10	2.03
RESCAL$_{\text{DPKGE}}$		57.27%	**40062.61±842.14**	**12.01±3.67**	0.30
RESCAL$_{\text{FullDP}}$	10.0	any	55549.63±931.95	0.03±0.05	0.26
DistMult			**24259.69±951.66**	5.88±.14	
DistMult$^{\mathcal{U}}$	–	–	37233.58±2669.08	**12.98±1.37**	–
DistMult$_{\text{DPKGE}}$		57.27%	**25565.57±3268.94**	**16.30±6.52**	9.21
DistMult$_{\text{FullDP}}$	0.7	any	43564.15±1749.16	8.63±1.42	6.77
DistMult$_{\text{DPKGE}}$		57.27%	**26375.88±4039.03**	**13.83±4.03**	4.11
DistMult$_{\text{FullDP}}$	1.0	any	50747.99±1697.34	4.32±1.73	3.04
DistMult$_{\text{DPKGE}}$		57.27%	**27811.36±5600.49**	**11.41±3.44**	2.72
DistMult$_{\text{FullDP}}$	1.3	any	52250.45±784.97	2.72±1.05	2.03
DistMult$_{\text{DPKGE}}$		57.27%	**35824.17±3782.50**	**6.05±0.73**	0.30
DistMult$_{\text{FullDP}}$	10.0	any	53660.22±1542.87	0.02±0.05	0.26

tacker needs to decide whether it is part of the knowledge graph. To decide which candidate statements might be part of the knowledge graph, the attacker has to observe the rank of the candidate statements. Those candidate statements with very low (good) rank have the highest likelihood of being part of the knowledge graph. We denote the process of choosing an h and an l and compare the rank for different candidate statement (h, l, t) as a single *attack*. To see how good the DPKGE methods protect the confidential statements, we are particularly interested in attacks with candidate statements that are actually (i) part of the knowledge graph, and (ii) are considered confidential. We will call attacks that meet those requirements *critical attacks*. Hence, we ask ourselves: *for critical attacks, how high is the rank of those tail entities t for which (h, l, t) is a confidential statement in the KG?*

Table 9: Performance of link prediction over MIMIC-III. MR and Hits are shown by averages and standard deviation over five runs. Since the NDP methods do not guarantee any differential privacy, ϵ and σ are set as "–". The best average values are bolded in each cell.

Method	σ	r	MR	Hits	ϵ
TransE			**23647.72±560.54**	**30.91±0.51**	
TransE$^{\mathcal{U}}$	–	–	75037.60±2330.78	16.79±0.16	–
TransE$_{\mathrm{DPKGE}}$		44.03%	36832.10±999.14	23.42±0.42	3.23
TransE$_{\mathrm{FullDP}}$	1.0	any	**31767.64±1307.00**	**34.32±0.65**	2.12
TransM			**15675.34±448.28**	**34.25±1.23**	
TransM$^{\mathcal{U}}$	–	–	67626.25±3973.61	17.18±0.47	–
TransM$_{\mathrm{DPKGE}}$		44.03%	**24891.37±1803.36**	24.55±0.38	3.23
TransM$_{\mathrm{FullDP}}$	1.0	any	37891.27±943.44	**33.05±0.55**	2.12
RESCAL			**45388.82±803.32**	**30.83±3.59**	
RESCAL$^{\mathcal{U}}$	–	–	83410.55±4031.40	18.89±1.28	–
RESCAL$_{\mathrm{DPKGE}}$		44.03%	61338.77±1786.17	23.51±2.97	3.23
RESCAL$_{\mathrm{FullDP}}$	1.0	any	**41433.16±3750.48**	**31.43±1.53**	2.12
DistMult			**22932.37±2357.55**	7.97±4.18	
DistMult$^{\mathcal{U}}$	–	–	75115.09±6539.61	**10.61±2.37**	–
DistMult$_{\mathrm{DPKGE}}$		44.03%	**27643.34±7003.10**	**17.22±2.96**	3.23
DistMult$_{\mathrm{FullDP}}$	1.0	any	70114.11±9291.34	14.77±3.14	2.12

Table 10: Example of three sampled confidential statements on the FB15k.

i	head	relation	tail
1	Ingmar Bergman	profession[10]	Actor
2	J. T. Walsh	award_nominee[11]	Paul Sorvino
3	Brideshead Revisited	award[12]	Primetime Emmy Award

Figure 6: Histogram of the ranks in TransE and TransE$_{\mathrm{DPKGE}}$. The count value of the bars of both TransE and TransE$_{\mathrm{DPKGE}}$ start at 0 (i.e., the bars partially overlap).

To study this question, we use FB15k and TransE. We generate 1000 critical attacks with randomly chosen head h and relation l. Then, we calculate the rank of a randomly chosen tail t such that (h, l, t) is a confidential statement in the KG. We do this for both the DPKGE and NDP versions of TransE. Figure 6 shows the distribution of tail ranks for both methods. We observe that the ranks for the DPKGE method are higher than for the NDP method, suggesting that a critical attack is more often successful (i.e., the attacker deduces that the confidential statement is part of the knowledge graph) when the NDP method is used rather than the DPKGE method. We run a Mann-Whitney U test on the two rank distributions to validate our observation. The p value for the test is 8.28e-44, i.e., it is very unlikely that the difference in the distributions can be explained by randomness. Consequently, the DPKGE method protects the confidential statements better than the NDP method.

4.7. Use Case: Differential Privacy of Link Prediction

The goal of the use case is to show whether we can determine the existence of one statement by running one algorithm on two neighboring datasets. For the FullDP method, its MR and Hits values are worse as shown in Tables 5, 6, 7, 8, 9, which is too large to conclude the existence of the statement on two neighboring datasets. Therefore, FullDP is not used in Section 4.7. To conclude, we present three examples to illustrate the benefits of DPKGE. Starting from FB15k, we built a knowledge graph $\mathcal{K}$ with r=0.5, i.e., 50% of the statements are confidential. Moreover, $\mathcal{K}$ contains the three confidential statements in Table 10, denoted with $i \in \{1, 2, 3\}$. We build the three neighbor knowledge graphs $\mathcal{K}_1$, $\mathcal{K}_2$ and $\mathcal{K}_3$, each of them without one of the statements in the table, e.g., $\mathcal{K} \setminus \mathcal{K}_1 = \{1\}$. We train TransE$_{NDP}$ and TransE$_{DPKGE}$ on the four knowledge graphs. For each statement i, we predict its rank using the embeddings learned with TransE$_{NDP}$ and TransE$_{DPKGE}$ over $\mathcal{K}$ and $\mathcal{K}_i$.

Table 11 shows the link prediction results. For the statement 1, the tail ranks for TransE$_{\mathrm{NDP}}$ trained on $\mathcal{K}$ and $\mathcal{K}_1$ in the link prediction task are 2 and 9, respectively. When using TransE$_{\mathrm{DPKGE}}$ on the two graphs, the ranks are both 2. When using NDP embeddings, the tail ranks differ more than in the case of DP embeddings. We can observe a similar trend also for the statements 2 and 3. As expected, the embedding spaces of neighbor KGs are more similar when using DPKGE. Hence, DPKGE embeddings are better at hiding the fact whether one of the selected statements is indeed part of the KG.

5. Limitations and Threats to Validity

Whilst DPKGE shows that many of today's methods can be adopted with differentially private versions thereof, our approach and evaluation also face a number of limitations.

First, the current version of our framework assumes that two neighboring graphs only differ in one edge. Sometimes, however, a better definition would be to consider a set of edges to be grouped in making one joint statement. For example, the address of a person consists of multiple statements like street name, street number, and ZIP code. If the address of a person is changed or removed, this usually involves changing or removing all address-related statements together. Another example relates to joint statements, such as reified statements. Hence, further research is needed to extend the neighborhood definition to edge groups.

Second, choosing which statements are confidential can be a complex task, as oftentimes, statements correlate with other ones. Differential privacy protects the statements in $\mathcal{C}$ against privacy leaks caused by comparing the embeddings of $\mathcal{C}$-edge-neighboring knowledge graphs. However, differential privacy does not protect from privacy leaks from statements in $\mathcal{U}$. This is particularly important in graph settings, where a large degree of auto-correlation has been found [13, 21, 50]. Such correlations can be exploited to reason on the unrestricted information to infer confidential statements. Hence, data curators will have to choose the confidential statements wisely. Finding good methods to support ontology engineers in analysing KGs and studying such correlation between unrestricted and confidential statements is out of the scope of this article, but it is a natural direction where to expand this research.

Third, a huge challenge in DP is choosing your privacy budget well. When epsilon is too big, then privacy leaks are too likely. When epsilon is too small, the amount of noise is too high. Indeed, choosing a good epsilon is the topic of ongoing research [27, 18]. In our method, epsilon cannot be set directly. Hence finding a good parameter setting that takes into account somewhat intuitive measures for the chance of a privacy leak still needs to be investigated.

Finally, to address any threats to external validity, more experiments with datasets containing well-defined restricted statements and possible privacy threats are needed. Whilst our choice of MIMIC-III and eICU is a good start, we need to better understand the relationship between the parameter choices, the choice of restricted edges, and the chance of a privacy leak.

Table 11: Example link prediction rank of neighboring datasets on FB15k using TransE.

	Rank			
i	$\text{TransE}_{\text{NDP}}$ @$\mathcal{K}$	$\text{TransE}_{\text{NDP}}$ @$\mathcal{K}_i$	$\text{TransE}_{\text{DPKGE}}$ @$\mathcal{K}$	$\text{TransE}_{\text{DPKGE}}$ @$\mathcal{K}_i$
1	2	9	2	2
2	1	7	7	6
3	2	16	11	10

6. Conclusions and Future Research

In this paper, we studied differentially private knowledge graph embedding. We propose a new general framework (DPKGE) to adapt knowledge graph embedding algorithms to differentially private ones. Moreover, we suggest that it is possible to apply differential privacy for the confidential statements in the knowledge graph only whilst keeping the utility of non-sensitive statements to improve overall performance. In addition, we propose an adaptive sampling algorithm to retain the same ratio of confidential and unrestricted statements in each epoch in a stochastic optimized way. We evaluate four DPKGE methods on five datasets, two of them containing real confidential statements from the health sector. Extensive experiments regarding utility, privacy, clustering, and link prediction have been conducted to evaluate the quality of the DP-KGE methods. The results show that DPKGE gives a high utility while applying differential privacy to the confidential statements. Thus, DPKGE is a feasible framework for differential private knowledge graph embedding.

In future research, we plan to adapt more knowledge graph embedding algorithms (e.g., RDF2vec [36]) with DPKGE, and evaluate DPKGE on more datasets. We also plan to explore limitations discussed in Section 5. We aim to investigate suitable methods to let ontology engineers and data curators define confidential edges. We envision methods that support human activity, which for example, automatically identify correlations between groups of statements and suggest confidential statements. Such methods may also be extended to support the tuning of the privacy parameters.

Whatever the results of future explorations, this paper introduces a central building block to share knowledge graphs containing sensitive information via privacy-preserving embeddings—a task of central importance to process KGs whilst adhering to privacy considerations.

Acknowledgements

Reynold Cheng and Xiaolin Han were supported by the Research Grants Council of Hong Kong (RGC Projects HKU 17229116,17205015), University of Hong Kong (Projects 104005858,104005994), HKU-TCL Joint Research Center for Artificial Intelligence (Project 200009430), and the Guangdong–Hong Kong-Macau Joint

Laboratory Program 2020 (Project 2020B1212030009). Daniele Dell'Aglio was partially supported by the Swiss National Science Foundation (contract #407550_167177). Tobias Grubenmann was supported by the Federal Ministry of Education and Research (BMBF), Germany, under Simple-ML (01IS18054), and the European Commission under PLATOON (872592) and Cleopatra (812997). We also thank the anonymous reviewers for their insightful feedback.

References

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L., 2016. Deep learning with differential privacy, in: CCS, pp. 308–318.

[2] Balažević, I., Allen, C., Hospedales, T.M., 2019. Tucker: Tensor factorization for knowledge graph completion. arXiv preprint arXiv:1901.09590 .

[3] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S., 2009. DBpedia - A crystallization point for the Web of Data. Journal of web semantics 7, 154–165.

[4] Blocki, J., Blum, A., Datta, A., Sheffet, O., 2012. The Johnson-Lindenstrauss Transform Itself Preserves Differential Privacy, in: FOCS, pp. 410–419.

[5] Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data, in: NIPS, pp. 2787–2795.

[6] Cai, H., Zheng, V.W., Chang, K.C.C., 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. TKDE 30, 1616–1637.

[7] Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G., 2016. Ontop: Answering SPARQL queries over relational databases. Semantic Web 8, 471–487. doi:10.3233/SW-160217.

[8] Cao, L., Xiao, D., Yan, Y., Madden, S., Li, G., 2021. Atlantic: making database differentially private and faster with accuracy guarantee. Proceedings of the VLDB Endowment 14, 2755–2758.

[9] Chen, S., Zhou, S., 2013. Recursive mechanism: towards node differential privacy and unrestricted joins, in: SIGMOD Conference, ACM. pp. 653–664.

[10] Cunningham, T., Cormode, G., Ferhatosmanoglu, H., Srivastava, D., 2021. Real-World Trajectory Sharing with Local Differential Privacy. Proc. VLDB Endow. 14, 2283–2295.

[11] Dalenius, T., Reiss, S.P., 1982. Data-swapping: A technique for disclosure control. Journal of Statistical Planning and Inference 6, 73–85. doi:https://doi.org/10.1016/0378-3758(82)90058-1.

[12] Dell'Aglio, D., Bernstein, A., 2020. Differentially private stream processing for the semantic web, in: The Web Conference 2020, ACM / IW3C2. pp. 1977–1987.

[13] Do, K., Tran, T., Venkatesh, S., 2018. Knowledge graph embedding with multiple relation projections, in: 2018 24th International Conference on Pattern Recognition (ICPR), IEEE. pp. 332–337.

[14] Dwork, C., Roth, A., 2014. The Algorithmic Foundations of Differential Privacy.

[15] Fan, M., Zhou, Q., Chang, E., Zheng, T.F., 2014. Transition-based knowledge graph embedding with relational mapping properties, in: PACLIC, pp. 328–337.

[16] Fredrikson, M., Jha, S., Ristenpart, T., 2015. Model inversion attacks that exploit confidence information and basic countermeasures, in: CCS '15: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM. pp. 1322—1333.

[17] Hay, M., Li, C., Miklau, G., Jensen, D.D., 2009. Accurate Estimation of the Degree Distribution of Private Networks, in: ICDM, pp. 169–178.

[18] Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B.C., Roth, A., 2014. Differential Privacy: An Economic Method for Choosing Epsilon, in: CSF, IEEE Computer Society. pp. 398–410.

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

[20] Ji, G., He, S., Xu, L., Liu, K., Zhao, J., 2015. Knowledge graph embedding via dynamic mapping matrix, in: ACL (1), pp. 687–696.

[21] Jia, Y., Wang, Y., Jin, X., Cheng, X., 2018. Path-specific knowledge graph embedding. Knowledge-Based Systems 151, 37–44.

[22] Johnson, A.E., Pollard, T.J., Shen, L., Lehman, L.w.H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L.A., Mark, R.G., 2016. MIMIC-III, a freely accessible critical care database. Sci Data 3 doi:10.1038/sdata.2016.35.

[23] Kasiviswanathan, S.P., Nissim, K., Raskhodnikova, S., Smith, A.D., 2013. Analyzing Graphs with Node Differential Privacy, in: TCC, Springer. pp. 457–476.

[24] Kirrane, S., Villata, S., d'Aquin, M., 2018. Privacy, security and policies: A review of problems and solutions with semantic web technologies. Semantic Web 9, 153–161.

[25] Kotsogiannis, I., Doudalis, S., Haney, S., Machanavajjhala, A., Mehrotra, S., 2020. One-sided Differential Privacy, in: ICDE, IEEE. pp. 493–504.

[26] Kotsogiannis, I., Tao, Y., He, X., Fanaeepour, M., Machanavajjhala, A., Hay, M., Miklau, G., 2019. Privatesql: a differentially private sql query engine. Proceedings of the VLDB Endowment 12, 1371–1384.

[27] Lee, J., Clifton, C., 2011. How Much Is Enough? Choosing \(\epsilon\) for Differential Privacy, in: ISC, Springer. pp. 325–340.

[28] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X., 2015. Learning entity and relation embeddings for knowledge graph completion, in: AAAI, pp. 2181–2187.

[29] McSherry, F., 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis, in: SIGMOD Conference, ACM. pp. 19–30.

[30] Narayanan, A., Shmatikov, V., 2008. Robust De-anonymization of Large Sparse Datasets, in: IEEE Symposium on Security and Privacy, pp. 111–125.

[31] Nickel, M., Tresp, V., Kriegel, H.P., 2011. A three-way model for collective learning on multi-relational data., in: ICML, pp. 809–816.

[32] Nissim, K., Raskhodnikova, S., Smith, A.D., 2007. Smooth sensitivity and sampling in private data analysis, in: STOC, ACM. pp. 75–84.

[33] Noy, N.F., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J., 2019. Industry-scale knowledge graphs: lessons and challenges. Commun. ACM 62, 36–43.

[34] Pollard, T.J., Johnson, A.E.W., Raffa, J.D., Celi, L.A., Mark, R.G., Badawi, O., 2018. The eICU collaborative research database, a freely available multi-center database for critical care research. Scientific Data doi:10.1038/sdata.2018.178.

[35] Reuben, J., 2018. Towards a Differential Privacy Theory for Edge-Labeled Directed Graphs, in: Sicherheit, pp. 273–278.

[36] Ristoski, P., Paulheim, H., 2016. Rdf2vec: Rdf graph embeddings for data mining, in: International Semantic Web Conference, Springer. pp. 498–514.

[37] Ristoski, P., Rosati, J., Noia, T.D., Leone, R.D., Paulheim, H., 2019. RDF2Vec: RDF graph embeddings and their applications. Semantic Web 10, 721–752.

[38] Silva, R., Leal, B., Brito, F., Vidal, V.M., Machado, J.C., 2017. A Differentially Private Approach for Querying RDF Data of Social Networks, in: IDEAS, pp. 74–81.

[39] Sweeney, L., 1997. Weaving technology and policy together to maintain confidentiality. J. of Law, Medicine and Ethics 25, 98–110.

[40] Sweeney, L., 2002. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-

Based Systems 10, 557–570.

[41] Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M., 2015. Representing text for joint embedding of text and knowledge bases, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal. pp. 1499–1509. URL: https://www.aclweb.org/anthology/D15-1174, doi:10.18653/v1/D15-1174.

[42] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G., 2016. Complex embeddings for simple link prediction, in: PMLR 48, pp. 2071–2080.

[43] Vrandecic, D., Krötzsch, M., 2014. Wikidata: a free collaborative knowledgebase. Commun. ACM 57, 78–85.

[44] Wang, Q., Mao, Z., Wang, B., Guo, L., 2017a. Knowledge Graph Embedding: A Survey of Approaches and Applications. TKDE 29, 2724–2743.

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

[46] Wang, Z., Zhang, J., Feng, J., Chen, Z., 2014. Knowledge graph embedding by translating on hyperplanes, in: AAAI, pp. 1112–1119.

[47] Yang, B., Yih, W.t., He, X., Gao, J., Deng, L., 2014. Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 .

[48] Yu, S.Y., Chhetri, S.R., Canedo, A., Goyal, P., Faruque, M.A.A., 2019. Pykg2vec: A Python Library for Knowledge Graph Embedding. arXiv:1906.04239 .

[49] Zhao, Y., Vetere, G., Pan, J.Z., Faraotti, A., Monti, M., Wu, H., 2015. Meta-Level Properties for Reasoning on Dynamic Data, in: JIST, Springer. pp. 271–279.

[50] Zhu, J.Z., Jia, Y.T., Xu, J., Qiao, J.Z., Cheng, X.Q., 2018. Modeling the correlations of relations for knowledge graph embedding. Journal of Computer Science and Technology 33, 323–334.

Appendix A. Differentially private SGD Algorithm

Algorithm 2: Differentially private SGD [1]

Input : Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate λ_t, noise scale σ, group size L, gradient norm bound C.

Initialize: θ_0 randomly

for $t \in [T]$ **do**

  Take a random sample L_t with sampling probability L/N;

  `// Compute gradient`

  For each $i \in L_t$ compute $\mathrm{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$;

  `// Clip gradient`

  $\bar{\mathrm{g}}_t(x_i) \leftarrow \mathrm{g}_t(x_i)/\max\left(1, \frac{||\mathrm{g}_t(x_i)||_2}{C}\right)$;

  `// Add noise`

  $\tilde{\mathrm{g}}_t \leftarrow \frac{1}{L}\sum_i \left(\bar{\mathrm{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$;

  `// Descent`

  $\theta_{t+1} \leftarrow \theta_t - \lambda_t \tilde{\mathrm{g}}_t$;

Output : θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Appendix B. NMI values for the clustering results

Tables B.12, B.13, B.14, B.15, B.16, B.17 show the NMI scores for k $\in \{2, 3, 4\}$.

Table B.12: NMI values for the pairs in $\{(C^{k,i}_{\text{NDP}}, C^{k,i}_{\text{DPKGE}}) \in \mathcal{C}_{\text{NDP}} \times \mathcal{C}_{\text{DPKGE}}\}$ when k=2. The average value is 0.96 and the standard deviation is 0.02.

NMI	$C^{2,1}_{\text{DPKGE}}$	$C^{2,2}_{\text{DPKGE}}$	$C^{2,3}_{\text{DPKGE}}$	$C^{2,4}_{\text{DPKGE}}$	$C^{2,5}_{\text{DPKGE}}$
$C^{2,1}_{\text{NDP}}$	0.94	0.92	0.94	0.97	0.97
$C^{2,2}_{\text{NDP}}$	0.97	0.94	0.97	0.95	0.95
$C^{2,3}_{\text{NDP}}$	1.0	0.97	1.0	0.97	0.97
$C^{2,4}_{\text{NDP}}$	0.95	0.97	0.95	0.93	0.93
$C^{2,5}_{\text{NDP}}$	0.97	0.95	0.97	1.0	1.0

Table B.13: NMI values for the pairs in $\{(C^{k,i}_{\text{NDP}}, C^{k,i}_{\text{U-NDP}}) \in \mathcal{C}_{\text{NDP}} \times \mathcal{C}_{\text{U-NDP}}\}$ when k=2. The average value is 0.02 and the standard deviation is 0.01.

NMI	$C^{2,1}_{\text{U-NDP}}$	$C^{2,2}_{\text{U-NDP}}$	$C^{2,3}_{\text{U-NDP}}$	$C^{2,4}_{\text{U-NDP}}$	$C^{2,5}_{\text{U-NDP}}$
$C^{2,1}_{\text{NDP}}$	0.02	0.04	0.03	0.01	0.02
$C^{2,2}_{\text{NDP}}$	0.02	0.03	0.03	0.01	0.02
$C^{2,3}_{\text{NDP}}$	0.02	0.04	0.03	0.01	0.02
$C^{2,4}_{\text{NDP}}$	0.02	0.04	0.03	0.01	0.02
$C^{2,5}_{\text{NDP}}$	0.02	0.04	0.03	0.01	0.02

Table B.14: NMI values for the pairs in $\{(C^{k,i}_{\text{NDP}}, C^{k,i}_{\text{DPKGE}}) \in \mathcal{C}_{\text{NDP}} \times \mathcal{C}_{\text{DPKGE}}\}$ when $k = 3$. The average value is 0.34 and the standard deviation is 0.02.

NMI	$C^{3,1}_{\text{DPKGE}}$	$C^{3,2}_{\text{DPKGE}}$	$C^{3,3}_{\text{DPKGE}}$	$C^{3,4}_{\text{DPKGE}}$	$C^{3,5}_{\text{DPKGE}}$
$C^{3,1}_{\text{NDP}}$	0.33	0.36	0.34	0.35	0.36
$C^{3,2}_{\text{NDP}}$	0.32	0.3	0.31	0.31	0.3
$C^{3,3}_{\text{NDP}}$	0.35	0.33	0.34	0.35	0.34
$C^{3,4}_{\text{NDP}}$	0.37	0.37	0.35	0.37	0.35
$C^{3,5}_{\text{NDP}}$	0.32	0.35	0.34	0.35	0.33

Table B.15: NMI values for the pairs in $\{(C^{k,i}_{\text{NDP}}, C^{k,i}_{\text{U-NDP}}) \in \mathcal{C}_{\text{NDP}} \times \mathcal{C}_{\text{U-NDP}}\}$ when $k = 3$. The average value is 0.04 and the standard deviation is 0.03.

NMI	$C^{3,1}_{\text{U-NDP}}$	$C^{3,2}_{\text{U-NDP}}$	$C^{3,3}_{\text{U-NDP}}$	$C^{3,4}_{\text{U-NDP}}$	$C^{3,5}_{\text{U-NDP}}$
$C^{3,1}_{\text{NDP}}$	0.06	0.1	0.01	0.06	0.03
$C^{3,2}_{\text{NDP}}$	0.01	0.02	0.01	0.02	0.01
$C^{3,3}_{\text{NDP}}$	0.04	0.04	0.04	0.05	0.02
$C^{3,4}_{\text{NDP}}$	0.06	0.08	0.06	0.08	0.06
$C^{3,5}_{\text{NDP}}$	0.04	0.08	0.01	0.06	0.02

Table B.16: NMI values for the pairs in $\{(C^{k,i}_{\text{NDP}}, C^{k,i}_{\text{DPKGE}}) \in \mathcal{C}_{\text{NDP}} \times \mathcal{C}_{\text{DPKGE}}\}$ when $k = 4$. The average value is 0.28 and the standard deviation is 0.02.

NMI	$C^{4,1}_{\text{DPKGE}}$	$C^{4,2}_{\text{DPKGE}}$	$C^{4,3}_{\text{DPKGE}}$	$C^{4,4}_{\text{DPKGE}}$	$C^{4,5}_{\text{DPKGE}}$
$C^{4,1}_{\text{NDP}}$	0.26	0.27	0.24	0.25	0.28
$C^{4,2}_{\text{NDP}}$	0.28	0.31	0.29	0.29	0.32
$C^{4,3}_{\text{NDP}}$	0.27	0.26	0.25	0.29	0.27
$C^{4,4}_{\text{NDP}}$	0.3	0.3	0.26	0.32	0.29
$C^{4,5}_{\text{NDP}}$	0.28	0.3	0.25	0.27	0.31

Table B.17: NMI values for the pairs in $\{(C^{k,i}_{\text{NDP}}, C^{k,i}_{\text{U-NDP}}) \in \mathcal{C}_{\text{NDP}} \times \mathcal{C}_{\text{U-NDP}}\}$ when $k = 4$. The average value is 0.05 and the standard deviation is 0.02.

NMI	$C^{4,1}_{\text{U-NDP}}$	$C^{4,2}_{\text{U-NDP}}$	$C^{4,3}_{\text{U-NDP}}$	$C^{4,4}_{\text{U-NDP}}$	$C^{4,5}_{\text{U-NDP}}$
$C^{4,1}_{\text{NDP}}$	0.02	0.07	0.03	0.03	0.05
$C^{4,2}_{\text{NDP}}$	0.01	0.08	0.03	0.06	0.04
$C^{4,3}_{\text{NDP}}$	0.04	0.08	0.05	0.05	0.05
$C^{4,4}_{\text{NDP}}$	0.05	0.09	0.07	0.05	0.05
$C^{4,5}_{\text{NDP}}$	0.03	0.1	0.05	0.09	0.07