Efficient Federated Learning on Knowledge Graphs via Privacy-preserving Relation Embedding Aggregation

Kai Zhang[1], Yu Wang[2], Hongyi Wang[3], Lifu Huang[4], Carl Yang[5], Xun Chen[6], Lichao Sun[1]
[1]Lehigh University, [2]University of Illinois Chicago, [3]Carnegie Mellon University,
[4]Virginia Tech, [5]Emory University, [6]Samsung Research America
kaz321@lehigh.edu, ywang617@uic.edu, hongyiwa@andrew.cmu.edu,
lifuh@vt.edu, j.carlyang@emory.edu, xun.chen@samsung.com, lis221@lehigh.edu

Abstract

Federated learning (FL) can be essential in knowledge representation, reasoning, and data mining applications over multi-source knowledge graphs (KGs). A recent study FedE first proposes an FL framework that shares entity embeddings of KGs across all clients. However, entity embedding sharing from FedE would incur a severe privacy leakage. Specifically, the known entity embedding can be used to infer whether a specific relation between two entities exists in a private client. In this paper, we introduce a novel attack method that aims to recover the original data based on the embedding information, which is further used to evaluate the vulnerabilities of FedE. Furthermore, we propose a **Fed**erated learning paradigm with privacy-preserving **R**elation embedding aggregation (FEDR) to tackle the privacy issue in FedE. Besides, relation embedding sharing can significantly reduce the communication cost due to its smaller size of queries. We conduct extensive experiments to evaluate FEDR with five different KG embedding models and three datasets. Compared to FedE, FEDR achieves similar utility and significant improvements regarding privacy-preserving effect and communication efficiency on the link prediction task.

1 Introduction

Knowledge graphs (KGs) are critical data structures to represent human knowledge and serve as resources for various real-world applications, such as recommendation and question answering (Gong et al., 2021; Liu et al., 2018). However, most KGs are usually incomplete and naturally distributed to different clients. Despite each client can explore the missing links with their own KGs by knowledge graph embedding (KGE) models (Lin et al., 2015), exchanging knowledge with others can further enhance completion performance because the overlapping elements are usually involved in different KGs (Chen et al., 2021; Peng et al., 2021).

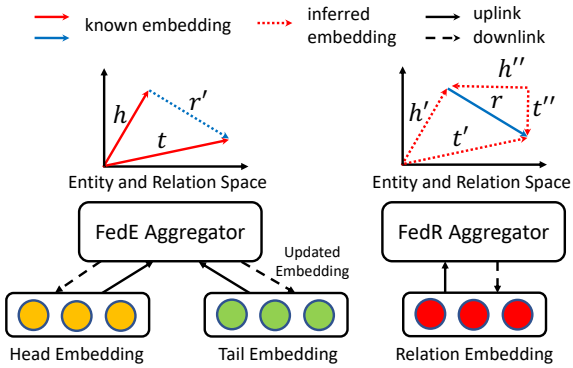


Figure 1: FedE aggregates entity embeddings from clients while FEDR aggregates relation embeddings. Since in FEDR, there would be infinite embedding pairs of head and tail given a relation embedding, the inference attack would fail.

To exchange knowledge, the first federated learning (FL) framework for KG – FedE is recently proposed, where each client trains local embeddings on its KG while the server receives and aggregates only locally-computed updates of entity embeddings instead of collecting triplets directly (Chen et al., 2021). However, at the very beginning in FedE, the server should collect the entity sets of every client for entity alignment, which will lead to unintentional privacy leakage: 1) entity's information, such as the customer's name, is usually sensitive but it is fully exposed to the server; 2) the relation embedding will be inferred and be exploited for knowledge graph reconstruction attack if there exists the malicious server (see Section 3.1). Therefore, we propose FEDR that adopts relation embedding aggregation to tackle the privacy issue in FedE. The major difference is shown in Figure 1. Besides, the number of entities is usually greater than the number of relations in real-world graph databases, so sharing relation embedding is more communication-efficient.

We summarize the following contributions of our work. 1) We present a KG reconstruction attack method and reveal that FedE suffers a potential

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 613–621
December 7-11, 2022

privacy leakage due to a malicious server and its colluded clients. 2) We propose FEDR, an efficient and privacy-preserving FL framework on KGs. Experimental results demonstrate that FEDR has the competitive performance compared with FedE, but gains substantial improvements in terms of privacy-preserving effect and communication efficiency.

2 Background

Knowledge graph and its embedding. KG is a directed multi-relational graph whose nodes correspond to entities and edges of the form (head, relation, tail), which is denoted as a triplet (h, r, t). KGE model aims to learn low-dimensional representations of elements in a KG via maximizing scoring function $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$ of all embedding of triplets. In other words, as depicted in Figure 1, we can infer relation embedding in terms of $\mathbf{r}' = \arg\max_{\mathbf{r}} f(\mathbf{h}, \mathbf{r}, \mathbf{t})$ given entity embeddings, but we cannot obtain $\mathbf{t}' = \arg\max_{\mathbf{t}} f(\mathbf{h}, \mathbf{r}, \mathbf{t})$ merely based on known relation embedding $\mathbf{r}$.

Federated learning and FedE. FL allows different clients to collaboratively learn a global model without sharing their local data (McMahan et al., 2017). In particular, the aim is to minimize: $\min_w f(w) = \mathbb{E}_k [F_k(w)]$, where $F_k(w)$ is the local objective that measures the local empirical risk of k-th client. Compared to model sharing in vanilla FL , FedE introduces a new mechanism that aggregates only entity embedding. More concretely, the server maintains a complete table including entity embeddings and the corresponding entity IDs, and the server can identify if an entity exists in a client for entity alignment.

3 Methodology

3.1 Knowledge Graph Reconstruction

The purpose of knowledge graph reconstruction attack is to recover original entities and relations in a KG given traitor's information including parital or all triplets and the corresponding embeddings, namely element-embedding pairs. The attack procedure for FedE is summarized as follows (suppose there is a malicious server and one traitor):

1) The server colludes with one client C1 to obtain its element-embedding pairs $\langle (E, \mathbf{e}), (R, \mathbf{r}) \rangle$.

2) Infer the target client's relation embedding by calculating $\mathbf{r}' = \arg\max_{\mathbf{r}} f(\mathbf{h}, \mathbf{r}, \mathbf{t})$.

3) Measure the discrepancy between the inferred element embedding such as relation embedding $\mathbf{r}'$ and all known $\mathbf{r}$ with cosine similarity.

4) Infer the relation R' as R, E' as E with corresponding largest similarity scores. Then the target client's KG/triplet can be reconstructed. More details are included in Appendix A.

LR	30%		50%		100%	
	ERR	TRR	ERR	TRR	ERR	TRR
C2	0.2904	0.0607	0.4835	0.1951	0.9690	0.7378
C3	0.2906	0.0616	0.4846	0.1956	0.9685	0.7390

Table 1: Privacy leakage on FB15k-237 with TransE.

Privacy leakage quantization in FedE. We define two metrics: *Triplet Reconstruction Rate* (TRR) and *Entity Reconstruction Rate* (ERR) to measure the ratio of corretly reconstructed triplets and entities to the relevant whole number of elements, respectively. We let the server owns 30%, 50%, 100% trained element-embedding pairs from C1, the traitor, to reconstruct entities and triplets of others. The results of privacy leakage on FB15k-237 (Toutanova et al., 2015) over three clients are summarized in Table 1. LR in the table denotes information (element-embedding pairs) leakage ratio from C1. It is clear that the server only needs to collude with one client to obtain most of the information of KGs on other clients. In a word, FedE is not privacy-preserving.

Algorithm 1: FEDR Framework.

Input : local datasets T^c, number of clients C, number of local epochs E, learning rate η

Server excutes:

collect relations from clients via PSU

initialize relation table with relation embedding $\mathbf{E}_0^r$

for round $t = 0, 1, ...$ **do**

  Send the relation table to all clients

  Sample a set of clients C_t

  for $c \in C_t$ **do in parallel**

    $\mathbf{E}_{t+1}^{r,c}, \mathbf{v}^c \leftarrow \texttt{Update}(c, \mathbf{E}_t)$

  $\mathbf{E}_{t+1}^r \leftarrow (\mathbb{1} \oslash \sum_{c=1}^{C_t} \mathbf{v}^c) \otimes \sum_{c=1}^{C_t} \mathbf{E}_{t+1}^{r,c}$ via SecAgg

Client excutes $\texttt{Update}(c, \mathbf{E})$**:**

for each local epoch $e = 1, 2, ..., E$ **do**

  for each batch $\mathbf{b} = (\mathbf{h}, \mathbf{r}, \mathbf{t})$ of T^c **do**

    $\mathbf{E} \leftarrow \mathbf{E} - \eta \nabla \mathcal{L}$, where $\mathbf{E} := \{\mathbf{E}^{e,c}, \mathbf{E}^{r,c}\}$

  Mask relation embedding: $\mathbf{E}^{r,c} \leftarrow \mathbf{M}^{r,c} \otimes \mathbf{E}^{r,c}$

return $\mathbf{E}^{r,c} \in \mathbf{E}, \mathbf{v}^c := \mathbf{M}^{r,c}$

3.2 FEDR

The overall procedure of FEDR framework is described in Algorithm 1. Before aggregation works, the server acquires all IDs of the unique relations from local clients and maintains a relation table

Dataset		DDB14				WN18RR				FB15k-237			
Model	Setting	C = 5	C = 10	C = 15	C = 20	C = 5	C = 10	C = 15	C = 20	C = 5	C = 10	C = 15	C = 20
TransE	Local	0.4206	0.2998	0.2464	0.2043	0.0655	0.0319	0.0378	0.0285	0.2174	0.1255	0.1087	0.0874
	FedE	0.4572	0.3493	0.3076	0.2962	0.1359	0.1263	0.1204	0.1419	0.2588	0.2230	0.2065	0.1892
	FEDR	**0.4461**	0.3289	0.2842	0.2761	0.0859	0.0779	0.0722	0.0668	**0.2520**	0.2052	0.1867	0.1701
RotatE	Local	0.4187	0.2842	0.2411	0.2020	0.1201	0.0649	0.0513	0.0155	0.2424	0.1991	0.1526	0.0860
	FedE	0.4667	0.3635	0.3244	0.3031	0.2741	0.1936	0.1287	0.0902	0.2682	0.2278	0.2199	0.1827
	FEDR	0.4477	0.3184	0.2765	0.2681	0.1372	0.1271	0.1074	**0.0912**	0.2510	0.2080	0.1854	0.1586
DistMult	Local	0.2248	0.1145	0.0764	0.0652	0.0654	0.0517	0.0548	0.0374	0.1133	0.0773	0.0765	0.0689
	FedE	0.3037	0.2485	0.2315	0.1877	0.1137	0.0946	0.0766	0.0670	0.1718	0.1129	0.0901	0.0753
	FEDR	**0.4219**	**0.3146**	**0.2685**	**0.2577**	**0.1350**	**0.1202**	**0.1198**	**0.0898**	**0.1670**	0.0999	**0.0884**	**0.0814**
ComplEx	Local	0.3406	0.2025	0.1506	0.1247	0.0035	0.0033	0.0033	0.0022	0.1241	0.0694	0.0571	0.0541
	FedE	0.3595	0.2838	0.2411	0.1946	0.0153	0.0115	0.0108	0.0122	0.1603	0.1161	0.0944	0.0751
	FEDR	**0.4287**	**0.3235**	**0.2747**	**0.2611**	**0.0203**	**0.0152**	**0.0152**	**0.0166**	**0.1716**	**0.1174**	**0.1075**	**0.0993**
NoGE	Local	0.3178	0.2298	0.1822	0.1580	0.0534	0.0474	0.0371	0.0372	0.2315	0.1642	0.1246	0.1042
	FedE	0.3193	0.3171	0.2678	0.2659	0.0789	0.0697	0.0632	0.0533	0.2412	0.1954	0.1730	0.1637
	FEDR	**0.4312**	**0.3127**	**0.2604**	0.2452	0.0669	0.0543	0.0530	0.0499	**0.2432**	0.1822	0.1448	0.1282

Table 2: Link prediction results (MRR). **Bold** number denotes FEDR performs better than or close to (within 3% performance decrease) FedE. Underline number denotes the better result between FEDR and Local.

via Private Set Union (PSU), which computes the union of relations, without revealing anything else, for relation alignment (Kolesnikov et al., 2019). Hence, the server does not know the relations each client holds. The constructed relation table is then distributed to each client, and in each communication round, partial clients are selected to perform local training (see Appendix B.2) to update element embeddings $\mathbf{E}^c$ that will be masked by the masking indicator $\mathbf{M}^{r,c}$ and uploaded to the server later. Here $\mathbf{M}_i^{r,c} = 1$ indicates the i-th entry in the relation table exists in client c. Considering that the server can retrive relations from each client by detecting if the embedidng is a vector of $\mathbf{0}$, we exploit Secure Aggregation technique (SecAgg, see Appendix C) in the aggregation phase as described in *line 8* in Algorithm 1, where $\oslash$ is element-wide division, $\otimes$ is element-wide multiplication, and $\mathbb{1}$ is an all-one vector. The fundamental idea behind SecAgg is to mask the uploaded embeddings such that the server cannot obtain the actual ones from each client. However, the sum of masks can be canceled out, so we still have the correct aggregation results (Bonawitz et al., 2017). Specifically, in FEDR, the server cannot access correct masking vectors $\mathbf{v}^c$ and embeddings $\mathbf{E}_{t+1}^{r,c}$ but only access the correct sum of them, namely, $\sum_{c=1}^{C_t} \mathbf{v}^c$ and $\sum_{c=1}^{C_t} \mathbf{E}_{t+1}^{r,c}$, respectively. At the end of round t, the aggregated $\mathbf{E}_{t+1}^{c}$ will be sent back to each client $c \in C_t$ for next-round update.

4 Experiments

We carry out several experiments to explore FEDR's performance in link prediction, in which the tail t is predicted given head h and relation r.

Datasets. We evaluate our framework through experiments on three public datasets, FB15k-237, WN18RR (Dettmers et al., 2018) and a disease database – DDB14 (Wang et al., 2021). To build federated datasets, we randomly split triplets to each client without replacement. Note that, random split makes data heterogeneous among all the clients, and ensures fair comparison between FedE and FedR.

KGE Algorithms. Four commonly-used KGE algorithms – TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), DisMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) are utilized in the paper. We also implement federated NoGE (Nguyen et al., 2022), a GNN-based algorithm.

4.1 Effectiveness Analysis

The commonly-used metric for link prediction, mean reciprocal rank (MRR), is exploited to evaluate FEDR's performance. We take FedE and Local, where embeddings are trained only on each client's local KG, as the baselines. Table 2 shows the link prediction results under settings of different number of clients C. We observe that FEDR comprehensively surpasses Local under all settings of the number of clients, which indicates that relation aggregation makes sense for learning better embeddings in FL. Take NoGE as an example, FEDR gains $29.64 \pm 0.037\%$, $22.13 \pm 0.065\%$, and $11.84 \pm 0.051\%$ average improvement in MRR on three dataset. Compared with FedE, FEDR usually presents the better or similar results with the KGE models of DistMult and its extensive version ComplEx on all datasets. We also observe that both entity and relation aggregations succeed in beating

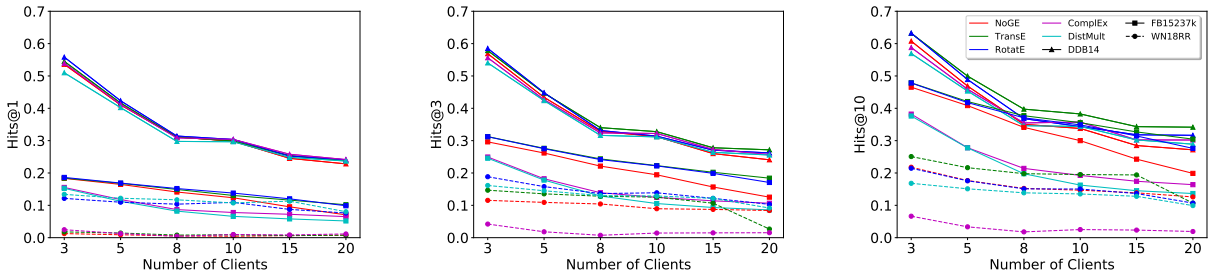


Figure 2: Experimental results of hit rates on three datasets.

`Local` setting but gain marginal improvement with DistMul and ComplEx on DDB14 and WN18RR datasets. Specially, KGE models fails to obtain reasonable results in federated with ComplEx. A potential reason could be that the averaging aggregation is not suitable for complex domains especially on the extremely unbalanced data (*w.r.t* number of unique entities and relations in a KG). Although FedE performs better than FEDR with TranE and RotatE, the absolute performance reductions between FedE and FEDR are mostly (13/16 = 81%) within 0.03 in MRR on both DDB14 and FB15k-237, which illustrates that FEDR is still effective. The theoretical explanations behind these results *w.r.t* data heterogeneity, and characteristics of FL and KGE models need further studies.

To further assess relation aggregation strategy, we compare performance of different KGE models regarding Hit Rates, which is shown in Figure 2. Similar to MRR, Hit Rates drop with the increasing number of clients because of the more sparse knowledge distribution. All KGE models behave well and consistently on DDB14 dataset while there are large deviations of performance between each model on WN18RR and FB15k-237. This phenomenon is attributed to the biased local knowledge distribution, which is implicitly shown by the number of local entities.

4.2 Privacy Leakage Analysis

Compared with entity aggregation, additional knowledge is required to perform reconstruction attack in FEDR because it is almost impossible to infer any entity or triplet from relation embeddings only. Therefore, we assume the server can access all entity embeddings without entity's IDs from clients. For simplicity, we let the server holds all information from C1, which is the same as the attack in Section 3.1 (LR=100%). The difference of adversary knowledge in FedE and FEDR is outlined in Table 3. Besides, for fair comparison of FedE and FEDR, PSU and SecAgg are not considered.

	GEE	LEE	GRE	LRE
FedE	✔	✔	✗	✗
FedR	✗	✔	✔	✔

Table 3: Summary of adversary knowledge. "G" represents "Global", "L" represents "Local". "EE" and "RE" represent entity and relation embeddings, respectively.

Table 4 presents the privacy leakage quantization in FEDR over three clients. The results shows that relation aggregation can protect both entity-level and graph-level privacy well even if providing additional local entity embeddings without considering encryption techniques. In addition, we observe that despite the relation embedding can be exploited directly in FEDR instead of inference, the privacy leakage rates in FEDR are still substantially lower than the ones in FedE. For example, according to Table 1, for C2, FEDR obtains relative reduction of 98.50% and 99.52% in ERR and TRR, respectively. Note that once PSU and SecAgg are applied, FEDR can successfully defense against KG reconstruction attack and gain **NO** privacy leakage.

Dataset	FB15k-237		WN18RR		DDB14	
	ERR	TRR	ERR	TRR	ERR	TRR
C2 **w/o**	145.43	35.04	22.00	9.89	19.39	10.10
C3 **w/o**	129.77	22.01	18.44	9.23	8.87	5.05
C2 **w**	**0**	**0**	**0**	**0**	**0**	**0**
C3 **w**	**0**	**0**	**0**	**0**	**0**	**0**

Table 4: Privacy leakage in FEDR with TransE ($\times 10^{-4}$). **w** and **w/o** represent encryptions are applied or not.

4.3 Communication Efficiency Analysis

In this section, the product of data sizes and communication rounds is calculated to measure the communication cost. Considering the performance difference between FEDR and FedE, for fair comparison of communication efficiency, we count the

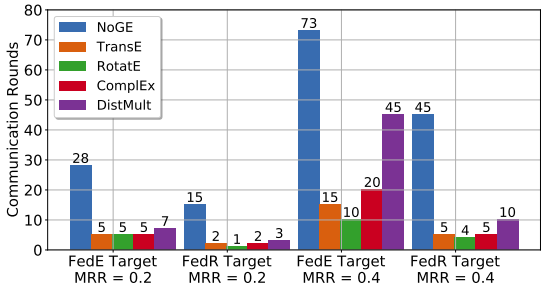


Figure 3: Number of communication rounds to reach a target MRR for FedE and FEDR with a fixed $C = 5$.

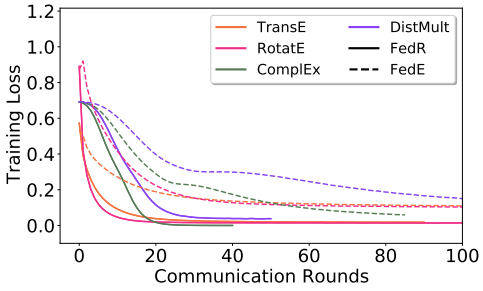
rounds when the model reaches a pre-defined MRR target on the validation dataset. Specifically, we set two different MRR targets: 0.2 and 0.4. Since all models perform well on DDB14, we take the setting with $C = 5$ on DDB14 as an example in this section. The required rounds for each model are depicted in Figure 3. We observe that FEDR reaches the target with much less rounds compared with FedE. For instance, FEDR-DistMult reaches the target MRR = 0.4 within 10 rounds while FedE uses 45 rounds. Also, according to statistics of federated datasets in Table 5, the average of the number of unique entities in FedE and unique relations in FEDR are 4462.2 and 12.8, respectively. We use the number of entities/relations to reflect data size, and by using relation aggregation, $99.89 \pm 0.029\%$ of cost is reduced in average for all clients when the target MRR is 0.2, while $99.90 \pm 0.042\%$ of cost is reduced in average when the target MRR is 0.4. These results demonstrate that our proposed framework is more communication-efficient.

4.4 Convergence Analysis

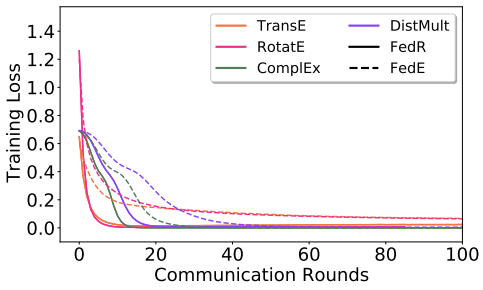
The convergence curves considering four KGE models and three dataset are shown in Figure 4. The solid and dashed lines represent curves *w.r.t* FEDR and FedE, respectively. We do not show the curves of NoGE because the aggregated embeddings does not influence local training. We observe that FEDR usually converge faster than FedE. Some lines are incomplete over communication rounds because early-stop technique in terms of validation MRR is used in the experiments.

5 Conclusion and Future Work

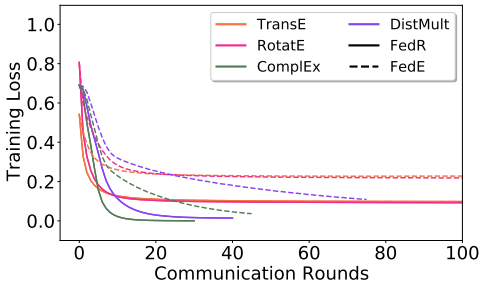
In this paper, we conduct the first empirical quantization of privacy leakage to federated learning on knowledge graphs, which reveals that recent work, FedE, is susceptible to reconstruction attack based on shared element-embedding pairs when



(a) DDB14



(b) WN18RR



(c) FB15k-237

Figure 4: Training loss versus communication ($C = 5$).

there are dishonest server and clients. Then we propose FEDR, a privacy-preserving FL framework on KGs with relation embedding aggregation that defenses against reconstruction attack effectively. Experimental results show that FEDR outperforms FedE w.r.t data privacy and communication efficiency and also maintains similar utility.

In real-world applications, different organizations may use different KGE models, which may influence overall performance by embedding aggregation, how to design an effective FL framework in this case and how to perform KG reconstruction attack/defense are our future research directions.

6 Limitations

Both FEDR and FedE are sensitive to data distribution. For example, if we build subgraphs in terms of relations, FEDR may not effective because of less overlapping relations among clients. It is still an open question that how to develop an FL architecture over arbitrarily non-iid KGs.

References

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. Fede: Embedding knowledge graphs in federated setting. In *The 10th International Joint Conference on Knowledge Graphs*, pages 80–88.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-second AAAI conference on artificial intelligence*.

Fan Gong, Meng Wang, Haofen Wang, Sen Wang, and Mengyue Liu. 2021. Smr: Medical knowledge graph embedding for safe medicine recommendation. *Big Data Research*, 23:100174.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. 2019. Scalable private set union from symmetric-key techniques. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 636–666. Springer.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Ziqing Liu, Enwei Peng, Shixing Yan, Guozheng Li, and Tianyong Hao. 2018. T-know: a knowledge graph-based question answering and infor-mation retrieval system for traditional chinese medicine. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 15–19.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723.

Dai Quoc Nguyen, Vinh Tong, Dinh Phung, and Dat Quoc Nguyen. 2022. Node co-occurrence based graph neural networks for knowledge graph link prediction. In *Proceedings of WSDM 2022 (Demonstrations)*.

Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333.

Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Differentially private federated knowledge graphs embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1416–1425.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2021. Relational message passing for knowledge graph completion. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1697–1707.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. *Advances in Neural Information Processing Systems*, 32:2735–2745.

A Knowledge Graph Reconstruction

We summarize the knowledge graph reconstruction attack in Algorithm 2. Note that in the algorithm, i) and ii) refer to different operations, and only one will be performed in FedE or FEDR.

Algorithm 2: Knowledge graph reconstruction including attack in FEDE/FEDR.

Adversary knowledge: Local entity embeddings – **LEE**, local relation embeddings – **LRE**, element-embedding paris from a client – **EEP**, type of the used KGE model.

Entity reconstruction:

for entity embedding $\hat{e} \in$ **LEE do**
  for entity-embedding $(E, e) \in$ **EEP do**
    Calculate similarity between e and $\hat{e}$
    Update the inferred entity $\hat{E} = E$ with the greatest similarity score
return the reconstructed entity set $\{\hat{E}\}$

Triple reconstruction:
only one of i) and ii) will be implemented

i) **for** entity embeddings $(\hat{h}, \hat{t}) \in$ **LEE do**
  Calculate relation embedding $\hat{r}$ based on the scoring function of used KGE model, e.g. $\hat{r} = \hat{t} - \hat{h}$ with TransE
  for relation-embedding$(R, r) \in$ **EEP do**
    Calculate similarity between r and $\hat{r}$
    Update the inferred relation $\hat{R} = R$ with the greatest similarity score
return the reconstructed relation set $\{\hat{R}\}$

ii) **for** relation embedding $\hat{r} \in$ **LRE do**
  for relation-embedding$(R, r) \in$ **EEP do**
    Calculate similarity between r and $\hat{r}$
    Update the inferred relation $\hat{R} = R$ with the greatest similarity score
return the reconstructed relation set $\{\hat{R}\}$

Utilize $\{\hat{E}\}$ and $\{\hat{R}\}$ to reconstruct triples.

B Implementation Details

For TransE, RotatE, DistMult, and ComplEx, we follow the same setting as FedE (Chen et al., 2021). Specifically, the number of negative sampling, margin γ and the negative sampling temperature α are set as 256, 10 and 1, respectively. Note that, we adopt a more conservative strategy for embedding aggregation where local non-existent entities will not be taken as negative samples compared to FedE. For NoGE, we use GCN (Kipf and Welling, 2017) as encoder and QuatE (Zhang et al., 2019) as decoder. Once local training is done in a communciation round, the embeddings are aggregated and the triplet is scored by the decoder. The hidden size of 1 hidden layer in NoGE is 128.

Dataset	#C	#Entity	#Relation
DDB14	5	$4462.20_{\pm 1049.60}$	$12.80_{\pm 0.84}$
	10	$3182.60_{\pm 668.89}$	$12.60_{\pm 0.70}$
	15	$2533.86_{\pm 493.47}$	$12.50_{\pm 0.74}$
	20	$2115.59_{\pm 385.56}$	$12.35_{\pm 0.75}$
WN18RR	5	$21293.20_{\pm 63.11}$	$11.00_{\pm 0.00}$
	10	$13112.20_{\pm 46.70}$	$11.00_{\pm 0.00}$
	15	$9537.33_{\pm 45.45}$	$11.00_{\pm 0.00}$
	20	$7501.65_{\pm 31.72}$	$11.00_{\pm 0.00}$
FB15k-237	5	$13359.20_{\pm 27.36}$	$237.00_{\pm 0.00}$
	10	$11913.00_{\pm 31.56}$	$237.00_{\pm 0.00}$
	15	$10705.87_{\pm 36.93}$	$236.87_{\pm 0.35}$
	20	$9705.95_{\pm 44.10}$	$236.80_{\pm 0.41}$

Table 5: Statistics of federated datasets. The subscripts denote standard deviation. # denotes "number of".

If not specified, the local update epoch is 3, the embedding dimension of entities and relation is 128. Early stopping is utilized in experiments. The patience, namely the number of epochs with no improvement in MRR on validation data after which training will be stopped, is set as 5. We use Adam with learning rate 0.001 for local model update. All models are trained using one Nvidia 2080 GPU with 300 communication rounds at maximum.

B.1 Statistics of Datasets

To build federated datasets, we randomly split triples to each client without replacement, then divide the local triples into the train, valid, and test sets with a ratio of 80/10/10. The statistics of datasets after split is described in Table 5.

B.2 Client Update

The client update, or loca knowledge graph embedding update, corresponds to $\texttt{Update}(c, \mathbf{E})$ in Algorithm 1 starting from *line 9*, which learns both embeddings of entities and relations.

For a triplet (h, r, t) in client c, we adopt the self-adversarial nagative sampling (Sun et al., 2019) for effectively optimizing non-GNN KGE models:

$$\mathcal{L}(h, r, t) = -\log \sigma(\gamma - f_r(\mathbf{h}, \mathbf{t})) - \sum_{i=1}^{n} p(h, r, t'_i) \log \sigma(f_r(\mathbf{h}, \mathbf{t}'_i) - \gamma),$$

where γ is a predefined margin, σ is the sigmoid function, f is the scoring function that varies as shown in Table 6, and $(\mathbf{h}, \mathbf{r}, \mathbf{t}'_i)$ is the i-th negative triplet, which can be sampled from the following distribution:

$$p(h, r, t'_j | \{(h_i, r_i, t_i)\}) = \frac{\exp \alpha f_r(\mathbf{h}, \mathbf{t}'_i)}{\sum_i \exp \alpha f_r(\mathbf{h}, \mathbf{t}'_i)}$$

where α is the temperature of sampling. There would be E epoches of traning on the client at a round to update local-view embeddings $\mathbf{E}$ including entity and relation embeddings, but only local relation embeddings $\{\mathbf{E}^{r,c}\}$ will be sent to server.

For NoGE, we follow its plain design by minimizing the binary cross-entryopy loss function:

$$\mathcal{L} = -\sum_{(h,r,t)} \left(l_{(h,r,t)} \log\left(\texttt{sigmoid}(f(\mathbf{h},\mathbf{r},\mathbf{t})\right)\right) + \left(1 - l_{(h,r,t)}\right) \log\left(1 - \texttt{sigmoid}(f(\mathbf{h},\mathbf{r},\mathbf{t})\right))$$

$$\text{in which, } l_{(h,r,t)} = \begin{cases} 1 & \text{for } (h,r,t) \in G \\ 0 & \text{for } (h,r,t) \in G' \end{cases}$$

where G and G' are collections of valid and invalid triplets, respectively.

B.3 Scoring Function

Model	Scoring Function
TransE	$-\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$
RotatE	$-\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$
DistMult	$\mathbf{h}^\top \operatorname{diag}(\mathbf{r})\mathbf{t}$
ComplEx	$\operatorname{Re}\left(\mathbf{h}^\top \operatorname{diag}(\mathbf{r})\bar{\mathbf{t}}\right)$
NoGE	$\langle a'_h, a_t\rangle + \langle b'_h, b_t\rangle + \langle c'_h, c_t\rangle + \langle d'_h, d_t\rangle$
KB-GAT	$\left(\|_{m=1}^{\Omega} \operatorname{ReLU}\left(\left[\vec{h}_i, \vec{g}_k, \vec{h}_j\right] * \omega^m\right)\right) \cdot \mathbf{W}$

Table 6: A list of scoring functions for KGE models implemented in this paper. The scoring function used in NoGE comes from QuatE (Zhang et al., 2019).

C Secure Aggregation in FEDR

In this section, we illustrate how SecAgg works in FEDR through a simple exmaple including three clients with two relations. Mathematically, we assume the distribution of relation embeddings as $\mathbf{R}_1 = \{r_1\}, \mathbf{R}_2 = \{r_2\}$ and $\mathbf{R}_3 = \{r_1\}$, respectively. After PSU, the server will obtain a set of relations $\mathbf{R} = \{r_1, r_2\}$. Besides, we denote the corresponding masking vectors as $\mathbf{M}_1 = (1,0), \mathbf{M}_2 = (0,1)$ and $\mathbf{M}_3 = (1,0)$.

In one communication round, once all clients complete local training and prepare for the aggregation phase, via Diffie-Hellman secret sharing (Bonawitz et al., 2017), each client u generates $s_{u,v}$ randomly for every other client, and they agree on the large prime number l. Then each party u compute the masked value t_u for its secret vector s_u, where $s_u := \{\mathbf{R}_u, \mathbf{M}_u\}$, shown as below:

$$t_u = s_u + \sum_{u<v} s_{u,v} - \sum_{u>v} s_{v,u} \pmod l,$$

where $s_{u,v} = s_{v,u}$ for a specific condition, e.g. $s_{1,2} = s_{2,1}$. Therefore, each client holds its masked matrix as follows:

$$\begin{aligned} t_1 &= s_1 + s_{1,2} + s_{1,3} \pmod l, \\ t_2 &= s_2 + s_{2,3} - s_{2,1} \pmod l, \\ t_3 &= s_3 - s_{3,1} - s_{3,2} \pmod l, \end{aligned}$$

Next, these masked matrices are uploaded to the server. Now the server cannot obtain the actual information from clietns but could extract the correct aggregated value via:

$$\begin{aligned} \mathbf{z} &= \sum_{u=1}^{3} t_u \\ &= \sum_{u,v=1}^{3} \left(s_u + \sum_{u<v} s_{u,v} - \sum_{u>v} s_{v,u}\right) \\ &= \sum_{u=1}^{3} s_u \pmod l \end{aligned}$$

D Additional Results

In this section, we introduce additional experimental results of KB-GAT in a federated manner for link prediction.

D.1 Experiment result with KB-GAT

Since the aggregated information is not exploited in the local training in NoGE, we also implement KB-GAT (Nathani et al., 2019), the other GNN model but it can take advantages of both graph structure learning and global-view information aggregation. However, Fed-KB-GAT is memory-consuming. For KB-GAT, we use GAT (Veličković et al., 2018) as encoder and ConvKB (Nguyen et al., 2018) as decoder. Although the input to KB-GAT is the triple embedding, this model update neural network weights to obtain the final entity and relation embeddings. In each communication, we let the aggregated embeddings be the new input to KB-GAT, we find using small local epoches lead to bad performance because the model is not fully trained to produce high-quality embeddings. Therefore, we set local epoch of GAT layers as 500, while local epoch of convlutional layers as 150. Embedding size is 50 instead of 128 like others since we suffers memory problem using this model.

We conduct KB-GAT with both entity aggregation and relation aggregation on DDB14 with $C = 3$ as shown in Table 7. Due to the good performance of RotatE, we also compare KB-GAT with

Model	Setting	MRR	Hit@1	Hit@3	Hit@10
RotatE	Local	0.5347	0.5311	0.5459	0.5912
	FedE	0.6087	0.5070	0.6774	0.7916
	FEDR	0.5834	0.5583	0.5852	0.6326
KB-GAT	Local	0.4467	0.4369	0.4620	0.4755
	FedE	**0.5622**	**0.5471**	**0.5634**	**0.5887**
	FEDR	0.5034	0.4861	0.5301	0.5644

Table 7: Extensive experimental resutls on DDB14 with $C = 3$. **Bold** number denotes the best result in FedE and underline number denotes the best result in FEDR.

RotatE. Hit@N is also utilized in the evaluation. From the table, KB-GAT beats RotatE in regard of all evaluation metrics in both FedE and FedR setting. However, how to implement federated KB-GAT in a memory-efficient way is still an open problem.