Journal of Information Systems Engineering and Management
2025, 10(30s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Federated Learning approach for Question and Answering on Knowledge Graphs

Abhinav Gunti[1][0009-0002-7849-8163], Abhishek Patil[1][0009-0000-5638-5999], Adithya R Narayan[1][0009-0005-3082-6350], Anant Gulati[1][0009-0001-9419-4373], and Bhaskarjyoti Das[2][0000-0003-1225-9354]

[1,3,4] *Department of Computer Science and Engineering, PES University, Bengaluru 560085, India,*
[2] *Department of Computer Science and Engineering in AI & ML, PES University, Bengaluru 560085, India,*
gunti.abhinav@gmail.com, abhijpatil2003@gmail.com, adithyarnarayan07@gmail.com, anantgulati.3@gmailcom ,bhaskarjyoti01@gmail.com

ARTICLE INFO

Received: 30 Dec 2024

Revised: 19 Feb 2025

Accepted: 27 Feb 2025

ABSTRACT

Many organizations maintain Knowledge Graphs (KG) to store their data. To query this data, multiple Questions Answering (QA) systems have been proposed. In the scenario of multi-source Knowledge Graphs or a scenario where multiple organizations come together to develop a better and secure QA system to query their own data, a Federated Learning approach seems suitable to bridge this gap which enables the learning models to learn collaboratively across decentralized data sources, ensuring total data privacy and security. Hence, we propose our solution Fed-KGQA to address this problem. This research is limited to the domain of simple questions. Our research is subdivided into two main areas. First, we use a slightly improvised Federated Learning approach based on FedR which lets the clients train embeddings (of both entities and relations) locally using algorithms like TransE and then aggregate relation embeddings from all clients in a secure manner. Second, unlike traditional approaches, we employ KG Embedding (KGE) - based QA instead of SPARQL-based QA. Here we use the embeddings generated in the federated manner to represent entities and relationships. Given a question the system tries to infer the head, its embedding and the relational embedding from the query and then uses scoring function to deduce the tail entity, providing the answer to the user's query. We test our approach and study its effectiveness on three subsets of FreeBase2M KG acting as clients with corresponding subsets of FreeBase SimpleQuestions as QA datasets. Our method shows an improvement in the accuracy as compared to plain QA on all clients while maintaining the security of clients and efficiency of the process.

Keywords: *Knowledge Graphs, Knowledge Graph Question Answering, Federated Learning*

1 INTRODUCTION

A Knowledge Graph (KG) is a structural representation of information in the form of entities (similar to nodes in normal graphs) and relations or predicates (similar to edges in normal graphs) [1]. Relations show the relationships between these entities, which are real-world objects or concepts [2]. A KG is mathematically represented as G = (E, R, T) where E, R and T are the set of entities, relations, and triples respectively. The KG stores the information or observed facts in the form of these triples $T \subset E \times R \times E$. Each triple has two entities - head and tail entity connected via a predicate indicated as (h,r,t), where $h,t \in E$ and $r \in R$. In the rapidly advancing world of data and knowledge management, KGs such as FreeBase[3] and Wikidata[4] play a vital role across various domains and applications and there is an increase in demand for highly competent Question Answering systems to query these KGs effectively.

A Knowledge Graph Question Answering (KGQA) model provides answers to questions that are asked in natural language by learning the questions and looking up the information stored in the KG. A major gap in the existing research is the task of performing question answering in the scenario where the graph data is distributed in different sources or clients with each maintaining their own knowledge graph referred to as Multi-Source KG [5]. This has applications in various domains including banking and healthcare sector. These clients can benefit from information or data that is available from other similar clients which have common entities and relations but they cannot aggregate their data into a central server or share it with each other as it raises serious data security concerns and also violates some major regulations like "European Union General Data Protection Regulation"[6].

We try to address this gap using Federated Learning. Federated Learning revolves around learning a global statistical model from the data stored on multiple devices under the constraint that the data will be processed or trained locally on the devices with the periodic communication of only the intermediate updates to the central server[7]. We aim to modify the existing QA systems through a federated approach to take advantage of common relations between these sources in a completely secure manner and achieve better results.

A number of algorithms like TransE[8], ComplEx[9] and RotatE[10] were proposed to represent the KGs mathematically by embedding the entities and relations in low-dimensional vector space embeddings to capture semantic relationships between them and enable downstream tasks like link prediction, KG completion and triple completion. These vectors referred to as Knowledge Graph Embeddings(KGE) encode the head, tail and the predicate connecting them into a geometric n-dimensional space, where the proximity or distance between embeddings reflects semantic similarities or dissimilarities between them and therefore try to deduce the connectivity patterns (for example, symmetry or anti-symmetry, composition and inversion) according to the observed knowledge facts[11].

The majority of the existing systems designed to provide answer to questions posed in natural language work by converting questions to SPARQL queries and then running these queries on the KG to generate responses and hence are constrained by certain predefined rules. These traditional QA tools based on SPARQL often struggle to effectively manage the extensive range of available data and information. Another type of QA system was proposed which first detects the head entity and the predicate from the question and uses KG embeddings to find the tail entity using scoring function or parsing through the database much like triple prediction[12]. The tail entity forms the answer to the question. We use this approach in our study as the federated framework can be leveraged while generating the embeddings.

Our study is subdivided into two sub parts, the first being the process of collecting the union relations from multiple clients and training the embeddings in a federated setting which ensures that the data from each client cannot be deduced by any of the other clients and hence privacy is preserved. The second step involves using the embeddings acquired from the previous federated step to train a KGQA system.

The subsequent sections of the paper are as follows: Section 2 offers an overview of the pertinent research related to the problem statement. Section 3 explains the approach which includes the methodology and the architecture of the models used. Section 4 covers the experimentation which includes the dataset description and also presents the comparison of final results with baseline obtained from testing our approach along with a discussion of the results. Finally, Section 5 concludes the research paper by outlining the potential avenues for future research.

2 RELATED WORK

2.1 Federated Learning frameworks for Knowledge Graphs

Existing research in the domain of Federated Learning w.r.t to KGs is limited to Knowledge Graph Completion using downstream tasks - link prediction and triple completion. Two prominent frameworks or models have been put forth - FedE[4] and FedR[13]. FedE was the first work in this domain. It proposed the idea of training embeddings locally and then sending intermediate results periodically (after every 'n' rounds, where 'n' is a preset parameter) to the server where the embeddings are aggregated using FedAvg[14] algorithm and are then updated back to clients for the forthcoming set of training rounds.

However, this model had severe privacy concerns as the mapping of entities to clients was kept in the server with the risk of exposure if any one of the clients attained backdoor access to the server. A reconstruction attack was also performed by simulating the scenario of a single corrupt client on the server by the authors of FedR which showed that the original KG could be reconstructed to an extent of 99.52%. To address this issue, FedR framework was proposed that used Private Set Union algorithm to get the union of all relations and a Secure Aggregation algorithm which let the clients add masks to the embeddings updated locally before updating them to the server and when the server aggregates the updated masked embeddings, the masks cancel out each other and the server can only access the aggregated end result.

Moreover this model proposed aggregating the relation embeddings in contrast to the FedE's approach of aggregating the entity embeddings as the number of relations are substantially less in real-world scenarios and the standard datasets (e.g., FB15k-237[14], 14,541 entities, 237 relations). This brought an insignificant decrease in the performance but helped reduce the communication costs between server and clients significantly. Hence, FedR gained substantial improvements w.r.t privacy preserving effect and communication efficiency.

2.2 Knowledge Graph Embeddings

Four prominent KGE models have been previously proposed - TransE, DistMult, ComplEx and RotatE. TransE[3] models relationships as translations which operate on entity embeddings in a low-dimension space. It assumes that the relationship embedding can be seen as the translation from the head entity's embedding to the tail entity's embedding. DistMult simplifies the scoring function by modeling interactions between entities and relations as diagonal 3way interactions in the embedding space. It involves the element-wise product of embeddings, followed by reduction using a diagonal matrix. In the ComplEx[15] algorithm we represent the entities and relations as complex-valued embeddings in a complex vector space. It extends DistMult by modeling interactions through the Hermitian dot product in the complex space, capturing rich interactions and asymmetry in relationships. RotatE[16] operates by embedding the head, tail and predicate into a complex vector space. It uses rotation operations to model relationships. It can capture various properties such as symmetry, anti-symmetry, composition, and inversion through rotations in the complex space.

All these embedding models discussed rely on their respective scoring functions which try to differentiate between the positive and negative triples (obtained by replacing either of the entities in a triple) by placing the positive triples in higher score region and the negative triples in the lower score region. For example, the scoring function for TransE algorithm is $-||h+r-t||$ (the sum of head embedding and predicate should equal that of the tail).

2.3 Knowledge Graph Question Answering

A question answering system based on embedding approach was proposed by [17]. This approach divided the problem of question answering into four steps: entity detection, relation prediction, entity linking and finally evidence integration. In entity detection, the question undergoes tokenization, with each token labeled as either entity or not entity. A neural network model was employed and the output is a token sequence representing a candidate entity. Next step is to link this candidate entity to the actual node in the graph. This process is treated as fuzzy string matching without utilizing any neural networks. In the pipeline's next step, the aim of relation prediction is to pinpoint the relationship in the question. This task is viewed as a classification task and uses neural networks.

The final task involves integrating evidence from the previous components to make a single prediction. Tuples are generated and they are ranked according to their scores.[18] proposed a similar embedding-based framework which tackles issues like entity name ambiguity and varied predicate expressions. The paper introduces recurrent neural network-based models: head entity learning model, predicate learning model, and head entity detection model. The head entity and predicate learning models predict the questions head entity representation and predicate representation respectively. The HED model reduces the head candidate space which is based on the concept - named entity recognition (NER). A relation function computes the tail representation by using a joint distance metric. learned a scoring function between the question and the corresponding triple and ranks these triples according to the score. In our work, we have used the architecture . Methods like [19] approach the QA problem as a classification task, wherein they classify the query into one of the relationship types present in KG by using neural networks models and non-neural network models.

3 APPROACH

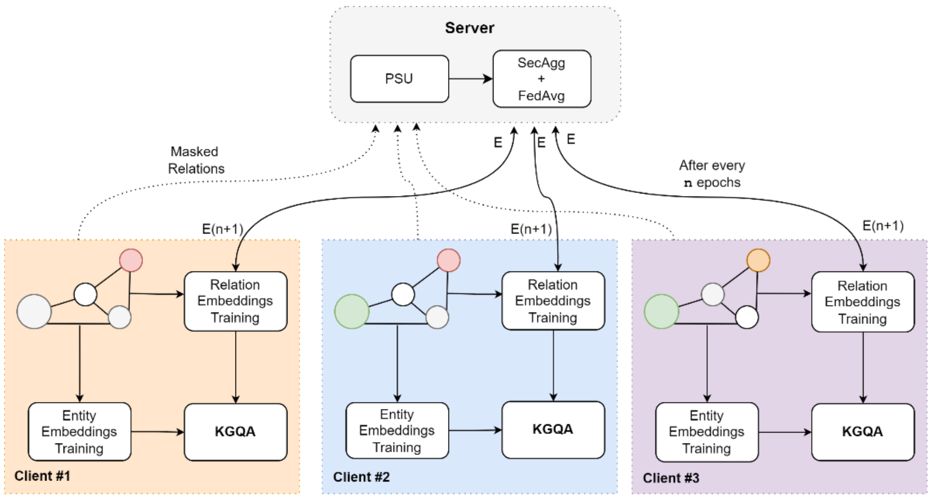

Fig.1. Fed-KGQA Architecture

Fig.1 shows the architecture for our proposed model Fed-KGQA. This architecture involves two major processes and are explained in the following subsections 3.1 and 3.2.

3.1 Training the Embeddings in a Federated Setting

We use the FedR framework that was proposed for the downstream task of KG Completion through link prediction. This framework primarily aggregates the relations of each client. The first step is to collect the relation IDs from each client and store them all in a relation table in a secure manner. This is performed using Private Set Union(PSU)[7] algorithm. PSU guarantees that the union of the relations will be calculated without revealing information about where the relations came from. Hence the server cannot know from which client each relation came from. The relation table obtained from PSU will be distributed to each client which will be used by the clients to communicate the changes to the server.

Each client trains the entity and relational embeddings on its own data locally for a preset 'n' number of rounds and after each set of local training, the updated values of embeddings will be replicated in the relation table during the communication round and sent to the server where the server aggregates them using FedAvg[10] algorithm. The process of communication of updated relational embeddings between the clients and server happens as per the Secure Aggregation[12] algorithm. In Secure Aggregation, each relational embedding is provided with a mask before being communicated to the server.

To explain the process of Secure Aggregation in simple terms, let's assume the scenario of 3 clients and the server. Assume the relation embedding distribution of each client as $R_1 = \{r_1\}$, $R_2 = \{r_2\}$ and $R_3 = \{r_1\}$ respectively. After the completion of PSU[7], the union of relations from all clients will be obtained by the server denoted as $R = \{r_1,r_2\}$. The existence vectors for relations of the clients will be

$$E_1 = (1,0), E_2 = (0,1), E_3 = (1,0)$$

During a communication round, each client u produces $s_{u,v}$ with every other client v ($s_{u,v} = s_{v,u}$ for a certain condition, for example when $s_{1,2} = s_{2,1}$). Each client computes the masked value t_u for its secret vector $s_u := \{R_u,E_u\}$, as shown below[19]:

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

Therefore, clients have their masked embedding vectors as:

$$t_1 = s_1 + s_{1,2} + s_{1,3}$$

$$t_2 = s_2 + s_{2,3} - s_{2,1}$$

$$t_3 = s_3 - s_{3,1} - s_{3,2}$$

After masking, these matrices will be uploaded to the server. The server will not be capable of obtaining original data from clients but as the masks are designed in a manner to cancel out each other, the server will be able to obtain only the end aggregated value via:

$$z = \sum_{u=1}^{3} t_u = s_1 + s_2 + s_3$$

These masks prevent the server from recognizing the actual embeddings of the relations from each client, but when they are aggregated, all the masks cancel each other out and we get the actual aggregated value at the server. This way even if one of the clients was to gain access to the server, they wouldn't be capable of deducing the presence of a certain relation from a particular client looking at the changes(if any) in the values of embeddings received from that particular client for that particular relation.

3.2 Question answering using KGQA

We begin this step of our pipeline with the output produced by the federated model. This output is the KG reduced to a low dimensional space. In FedR, an already existing embedding algorithm has been used to learn P and E such as TransE [3]. We use these KG embeddings to answer simple questions. The architecture proposed by [5] makes use of each entity E's and each predicate P's embedding representation. For each and every triple or fact (h,r,t) in the knowledge graph G, the embedding representation is denoted as (eh,pr,et). TransE function defines the relation as:

$$e_t \approx e_h + p_r$$

For the question answering process we will be using the architecture designed by [11] and [5]. This process is carried out though a course of three steps.

The first step involves using the Head Entity Detection (HED) model used by [5]. The primary aim of this model is to decrease the number of potential entities found in the search space. This is done by identifying multiple words in the question as head entities and using words which match or look very similar to these in the entity space to reduce the candidate head entities. This step implements a named entity recognition (NER) model for entity detection in text. To train this model the questions and their tagged entity and non-entity names is used for training the HED model. A question of length L is taken and all L tokens are mapped to the corresponding sequence of word embeddings using a pre-trained model like GloVe (Global Vectors). We use a bidirectional LSTM model, meaning it considers both the past and also the future contexts while processing each token in the sequence. For each element or token, the forward LSTM processes the sequence in the normal order (from left to right), while the backward LSTM processes the sequence in reverse order (from right to left). Both these passes of LSTM generate the corresponding hidden states as they process the sequence from both the directions. After both the LSTMs have processed the entire input sequence, the hidden states from both these LSTMs are concatenated. The concatenated output is supplied to the fully connected layer and finally a softmax function is applied which produces probabilities over two different label categories: head entity and non-head entity. The model learns to assign entity labels to words belonging to question which is used to recognize the head entities.

In the second step, we use the question in the training dataset and their respective predicate embeddings to train a model which will take as input the simple question and then obtain the predicate vector which exists in the embedded space. Neural Networks are used for learning predicate and head representations. Similar to the head entity detection model, a question of length L is taken and all L tokens are mapped to their corresponding word embeddings. We employ a bidirectional LSTM which considers both the past and future contexts for each token. The hidden states from both these LSTMs are concatenated. This output is then used to compute the attention weights, which are used to weight the word embeddings. The attention weights are applied to the concatenated hidden states and then the output is concatenated with the word embeddings and then supplied to a completely connected layer. The resultant is a target vector for each token. Finally, to compute the predicted predicate embeddings, the model aggregates the target vectors of all the tokens in the sequence by taking the mean of all token's target representations. Similarly, the head entity learning model will make use of the same architecture as that of the predicate learning model. In this manner, for a given question, this model will predict the head entity embedding representation directly.

The third step involves searching in the embedding space to rank all the triples according to their scores. We now have predicted the head and predicate representation for each question. Now we have to find the triple having the highest score that matches with the predicted head and predicate. Using the predicted head entity and predicate object, we iterate over the triples data to find the matched triple and we add the tail belonging to this triple as a candidate answer. We then calculate the joint distance as proposed by [5] to the predicated representations. We retrieve the results by using the scoring function and rank then according to their scores. The fact that has the smallest separation is chosen. Ultimately, the tail is returned as answer to the input simple question Q.

4 EXPERIMENTATION AND RESULTS

4.1 Dataset

FreeBase is a standard KG dataset used for all tasks in the study of Knowledge Graphs and is very large with billions of triples containing information about millions of entities and training any task on entire database requires vast amount of resources both in terms of computation and time. Hence two subsets of FreeBase - FB2M and FB5M with 2 million and 5 million entities respectively were released and have become the most popular benchmarks for various tasks. FB2M has a Question Answering (QA) dataset associated with it - SimpleQuestions[2] with roughly 80k questions in train, 20k in test and 10k in valid datasets.

But even these datasets suffer from data imperfections and are still very large when computationally intensive operations are to be performed to train on these datasets like generating embeddings for the link prediction task. Hence for downstream tasks like link prediction, triple completion, a standard benchmark was established - FB15k-237[14]. This has 15k entities and 237 relations and has served as a good benchmark. The problem associated with this dataset is that when the triples from FB15k-237[14] are mapped to SimpleQuestions[2] QA dataset, we end up with less than 1000 questions. So, we formed a dataset using the KG formed from SimpleQuestions and taking top 20k entities and mapping them to the FB2M dataset to find more triples. As a result, we formed the FBQA20k KG with 19.7k entities, 1836 relations and 278k triples. We split the triples into 3 clients to simulate the training under federated setting. Table 1 displays the statistical data for the three clients.

Table 1. Statistics of 3 clients from FBQA20k

Dataset	Entities	Relations	Triples	Train QA	Test QA	Valid QA
Client 1	18479	1358	92633	3833	730	377
Client 2	18433	1372	92633	3903	744	387
Client 3	18435	1360	92662	3768	726	324

4.2 Results

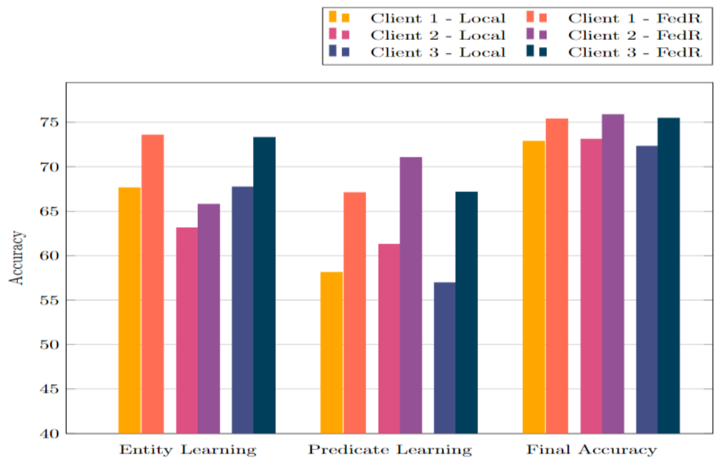

Fig.2. Results of 3 clients

To test the effectiveness of federated framework on the KGQA system, we simulate three clients as discussed earlier and use the TransE embedding model to train and generate embeddings. We take the KEQA model[5] (which uses vanilla TransE embeddings) as the baseline model. Compared to this baseline, our model achieves an improvement of **2.5%-3.2%** in final accuracy over the 3 clients as presented in Table 2.

Table 2. Results of 3 clients from FBQA20k

Client	Head Detection	Entity Learning		Predicate Learning		Final Accuracy	
		Local	FedR	Local	FedR	Local	FedR
1	85.11	67.64	73.58	58.13	67.10	72.88	75.39
2	89.11	63.15	65.79	61.30	71.06	73.11	75.88
3	89.66	67.74	73.30	56.97	67.17	72.31	75.48

The head detection accuracy, entity learning accuracy, predicate learning accuracy and final accuracy are presented in Table 2. The Local section presents the results obtained from the baseline model (training the embeddings locally and using them directly) and the FedR section presents the results from our proposed model Fed-KGQA. We credit these improvements in results to the better relation embeddings as result of training under the federated framework. This credit is also conveyed by the increase in the predicate learning accuracy as well as the entity learning accuracy, as seen in the Fig 2 which also shows that improving embeddings of relations can also indirectly result in improvement of embeddings of entities.

5 CONCLUSION AND FUTURE DIRECTIONS

We believe the improvement in accuracy can be enhanced if we were to aggregate the entity embeddings as well as opposed to the current framework which only aggregates the relational embeddings, but doing it in a secure manner using PSU and SecAgg brings in enormous communications costs specially when the number of clients scales to real-world scenarios since each client would have to coordinate with every other client in the federated network during SecAgg process to generate masks for each entity and the union of entities across all clients would be huge. So, we believe an algorithm that can aggregate both entity and relational embeddings across clients in a secure and efficient manner would help the QA system achieve better accuracy as embeddings play the major role in the domain of embedding based KGQA systems. Apart from the point raised here, we believe this study can be extended in two more directions.

The current study delves into QA on KG clients where all the clients use the same KGE model to train their entity and relation embeddings. In a real-world scenario, all the participating institutions might not use the same KGE model. In such case, developing a robust Federated QA system that can effectively aggregate embeddings across different geometric spaces is one potential future research direction.

In our study we have restricted to static or non-evolving graphs, but in a real-world scenario KGs are always evolving (newer entity and relation types are added to the existing KG). To make the QA system answer these facts, re-generating embeddings and hence re-training the QA system is necessary. In such case developing a QA system that can tackle incoming entities and relations without having to retrain the old data is another potential future direction.

REFERENCES

[1] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. p. 1247–1250. SIGMOD '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1376616.1376746

[2] Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks (2015)

[3] Bordes, A., Usunier, N., Garcıa-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (eds.) NIPS. pp. 2787–2795 (2013)

[4] Chen, M., Zhang, W., Yuan, Z., Jia, Y., Chen, H.: Fede: Embedding knowledge graphs in federated setting (2020)

[5] Huang, X., Zhang, J., Li, D., Li, P.: Knowledge graph embedding based question answering. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. p. 105–113. WSDM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3289600.3290956

[6] Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: Representation, acquisition and applications. CoRR (2020), https://arxiv.org/abs/2002.00388

[7] Jia, Y., Sun, S.F., Zhou, H.S., Du, J., Gu, D.: Shuffle-based private set union: Faster and more secure. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 2947–2964. USENIX Association, Boston, MA (Aug 2022)

[8] Kolesnikov, V., Rosulek, M., Trieu, N., Wang, X.: Scalable private set union from symmetric-key techniques. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 636–666. Springer International Publishing, Cham (2019)

[9] Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine 37(3), 50–60 (2020). https://doi.org/10.1109/MSP.2020.2975749

[10] Sekaran, R., Al-Turjman, F., Patan, R., & Ramasamy, V. (2023). Tripartite transmitting methodology for intermittently connected mobile network (ICMN). *ACM Transactions on Internet Technology*, *22*(4), 1-18.

[11] Mohammed, S., Shi, P., Lin, J.: Strong baselines for simple question answering over knowledge graphs with and without neural networks (2018)

[12] Segal, A., Marcedone, A., Kreuter, B., Ramage, D., McMahan, H.B., Seth, K., Bonawitz, K.A., Patel, S., Ivanov, V.: Practical secure aggregation for privacy-preserving machine learning. In: CCS (2017), https://eprint.iacr.org/2017/281.pdf

[13] Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space (2019)

[14] Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Marquez, L., Callison-Burch, C., Su, J. (eds.) Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1499– 1509. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015). https://doi.org/10.18653/v1/D15-1174, https://aclanthology.org/D15-1174

[15] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 2071–2080. PMLR, New York, New York, USA (20–22 Jun 2016), https://proceedings.mlr.press/v48/trouillon16.html

[16] Ture, F., Jojic, O.: No need to pay attention: Simple recurrent neural networks work! (for answering "simple" questions) (2017)

[17] Vrandecic, D., Krotzsch, M.: Wikidata: A free collaborative knowledgebase. Commun. ACM 57(10), 78–85 (sep 2014). https://doi.org/10.1145/2629489

[18] Yang, B., tau Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases (2014)

[19] Zhang, K., Wang, Y., Wang, H., Huang, L., Yang, C., Chen, X., Sun, L.: Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation (2022)