IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 February 2024, accepted 22 May 2024, date of publication 30 May 2024, date of current version 10 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3407669

RESEARCH ARTICLE

Federated Learning FedLTailor: A Dynamic Weight Adjustment and Personalized Fusion Approach

HONG ZHENG AND SHANQIN LI
School of Computer Science and Engineering, Changchun University of Technology, Changchun, Jilin 130022, China
Corresponding author: Hong Zheng (1440813190@qq.com)

ABSTRACT In this study, we primarily address the issue of uneven quality of client embeddings in existing federated learning frameworks for knowledge graph completion. Although existing frameworks provide preliminary solutions to the heterogeneity of data in knowledge graphs, there are still deficiencies in their aggregation strategies. To address this issue, we introduce a new federated learning framework called FedLTailor that focuses on optimizing the aggregation process. FedLTailor employs a dynamic weight adjustment strategy to enhance the weight proportion of clients with higher embedding quality during the aggregation process, thereby optimizing the performance of the global model. Moreover, FedLTailor adopted a unique personalized fusion strategy to mitigate potential discrepancies between the global model and local clients. Experimental results on four federated knowledge graph datasets demonstrate FedLTailor's significant advantages in addressing aggregation issues in federated knowledge graph completion tasks, as well as its broad adaptability across various knowledge graph embedding techniques. Additionally, the design and experimental validation of the FedLTailor framework offers new insights into the field of federated learning, particularly in handling distributed knowledge graph data, showcasing its potential applicability and effectiveness.

INDEX TERMS Federated learning, knowledge graph completion, knowledge graph.

I. INTRODUCTION

Knowledge graphs, which are structured representations of knowledge, have been widely used in numerous applications, such as search engines, recommendation systems, and intelligent assistants. In this vast network of relationships, nodes and edges map entities and their interconnections, respectively, thereby vividly and efficiently capturing the complexity of the real world. This provides a solid and rich knowledge base for intelligent systems. However, given continuous development and change in knowledge, knowledge graphs often face the problem of incomplete information. Knowledge graph completion is crucial for addressing this challenge. Thus link prediction [1] techniques have become a research hotspot, aiming to complete knowledge graphs by predicting missing entities or relationships. This not only enhances the coherence of knowledge graphs, but also supports the advanced understanding and reasoning capabilities of intelligent systems.

With the widespread application of knowledge graphs, new challenges have emerged, among which the ''data island'' problem is particularly critical. This issue primarily stems from different organizations, institutions, or companies being unwilling or unable to share their data with the outside world owing to privacy, security, and commercial considerations, leading to the limitation and dispersion of information in knowledge graphs. Faced with this limitation, federated learning offers a potential solution. Under the federated learning framework, each client uses its own local data for model training and uploads only the model's weights or updates information to a central server rather than the original data. The central server is responsible for integrating model updates from various clients, thereby optimizing and refining the global model. This allows data owners to

The associate editor coordinating the review of this manuscript and approving it for publication was Rajesh Kumar.

collaboratively build and optimize a more comprehensive and efficient knowledge graph while protecting personal privacy and commercial interests [2]. FedEC [2] is a federated learning framework designed specifically for knowledge graphs. It aggregates the same entity embeddings from different clients, instead of simply collecting triplet data. In this process, the model integrates knowledge across clients, significantly improving the quality of the entity embeddings. The framework proposes solutions to the data heterogeneity problem encountered when predicting the missing links in knowledge graphs in a federated environment. On the client side, a contrastive learning strategy was used to optimize the training process by comparing global and local embeddings, aiming to control the distance between global and local embeddings and effectively address the data heterogeneity issue. Through this method, the framework demonstrates better adaptability and efficiency in dealing with data differences between different clients, thereby improving the accuracy of knowledge graph completion. In the server-side aggregation process, FedEC employs the FedAvg [3] strategy for average aggregation. However, even for the same entity within different clients, the trained embedding effects may differ owing to the different relationships involved. This information can be used to manage diverse teams. Not every member of a team possesses the same skills or efficiency. If resources or rewards are distributed equally, high-performance members may feel underutilized or unmotivated, and the team's potential may not be fully realized. Conversely, if high-performance members are given more resources and responsibilities, a team's overall efficiency and outcomes can be significantly enhanced.

To address these issues, we propose a new federated knowledge graph completion framework, called FedLTailor. By analyzing the loss values of different clients, it was found that even under the same training rounds, the quality of embeddings produced by each client varied, with higher loss values indicating poorer training outcomes. Therefore, we adopted a strategy in which the loss values calculated by clients during training are uploaded to the server side. Through the application of Temperature-scaled Softmax, these were converted into new weight values, making the embedding aggregation process more aligned with the actual conditions. Embeddings of higher quality are assigned greater weights, whereas those of lower quality are assigned lower weights, thereby enhancing the quality and effectiveness of global model aggregation. Moreover, a personalized aggregation strategy is introduced. Specifically, the aggregated global entity embeddings are combined with the local entity embeddings of each client before aggregation, which helps to control the differences between global and local embeddings more precisely.

We conducted extensive experiments on four federated knowledge graph datasets and adopted various representative knowledge graph embedding methods as embedding learners within FedLTailor. The completion results of the federated knowledge graphs indicate that FedLTailor achieved significant performance compared with various baselines, demonstrating the effectiveness of our framework.

The contributions of this work are summarized as follows:

- We propose a framework called FedLTailor that aims to efficiently learn knowledge graph embeddings in a federated learning environment. This framework possesses broad adaptability and can be applied to a variety of current knowledge graph embedding technologies.
- In response to the uneven quality of the embeddings, a dynamic weight adjustment strategy was proposed to optimize the performance of the global model.
- To address the potential discrepancies between the global model and local clients, a personalized fusion strategy was introduced at the global aggregation stage to enhance the adaptability of the global model.

II. RELATED WORK

A. KNOWLEDGE GRAPH EMBEDDING

Entities and relationships in knowledge graphs are presented in the form of triplets, whereas deep learning models are more adept at handling data in continuous vector formats. Therefore, converting entities and relationships in knowledge graphs into vector forms enables more effective utilization of deep learning models for training. This conversion is known as 'knowledge graph embedding,' which also facilitates the execution of various downstream tasks, such as link prediction [4], [5], [6], [7], question-answering systems [5], and recommendation systems [6].

Translation Distance Models: These models are based on the spatial relationships between entities, such as translation or rotation, to represent relationships. TransE [7]: Maps entities and relationships in the knowledge graph into the same vector space, assuming that a relationship is a ''translation'' between two entity vectors. RotatE [8]: This method views relationships as a ''rotation'' between two entity vectors in the complex space. Similar to TransE, it operates in the same vector space. However, the relationships are represented by rotation instead of translation.

Semantic Matching Models: These models focus on the degree of matching or semantic similarity between entities and relationship vectors. DistMult [9] evaluated the match between entities and relationships by using element-wise multiplication. For a given relationship, it multiplies the vectors of the corresponding entities element-wise and then sums them to produce a score indicating the degree of match between these entities and the relationship. ComplEx [10]: Similar to DistMult, introduced complex-valued embeddings, allowing for the capture of richer and more complex relationship patterns, particularly asymmetric relationships.

B. FEDERATED LEARNING

Federated learning, a method that allows models to be trained across multiple local devices without directly sharing raw data, is based on the principle that each device holding data sources retains its original data and sends only the local model parameters to the central server. This approach not

only protects data privacy but also reduces the need to transfer large volumes of data.

Some studies addressed the issue of data heterogeneity. FedProx [11] introduces a proximal term to constrain the local update stage. MOON [12] applied model-level contrastive learning to deep-learning models in a federated setting. FedALA [13] incorporates personalized adaptive local aggregation for client models to capture necessary information from the global model, which primarily focuses on collaborative learning in the field of computer vision rather than knowledge graphs.

In research on knowledge graph embedding (KGE) within federated settings, FedE [14] adapted FedAvg for KGE by aggregating local embedding updates, allowing knowledge graph embeddings of individual clients to learn from the embeddings of others. FedEC identifies the data heterogeneity issue when learning KGE in federated environments, and addresses it by incorporating contrastive learning on the client side, which compares global embeddings with local embeddings to control the gap between them. These studies showcase methods for knowledge graph embedding in federated environments. They uniformly used the FedAvg method for average aggregation without considering the critical issue of varying embedding quality across different clients. The simple average aggregation strategy treats high and low-quality embeddings equally, giving them the same weight, which often leads to outcomes that are far from ideal. We propose a new method for optimizing the aggregation process. Specifically, a Temperature-scaled Softmax mechanism was employed to normalize the loss values of different clients, thereby calculating a more reasonable weight distribution. Based on this new weight, a personalized strategy was introduced for more effective entity aggregation. This approach not only optimizes the aggregation results but also significantly improves the quality of model embeddings while maintaining the privacy-preserving characteristics of federated learning.

III. METHODOLOGY

A. INTRODUCTION TO FEDERATED KNOWLEDGE GRAPH PREDICTION TASK

A knowledge graph is defined as $\mathcal{G} = \{E, R, T$, where $\mathcal{E}$ represents a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{T}$ is a set of triplets. Within T, each triplet is represented as (h, r, t), where $h, t \in \mathcal{E}$, and $r \in \mathcal{R}$. The link prediction task in knowledge graphs involves predicting the missing entity given an entity and a relation, either as $(h, r, ?)$ or $(?, r, t)$, essentially predicting $e \in \mathcal{E}$ so that (h, r, e) or (e, r, t) forms a new triplet to complete $\mathcal{G}$.

In the process of performing knowledge graph link prediction in a federated environment, each local client has a unique knowledge graph. These graphs differ across the clients. However, entities may overlap across multiple clients. This overlap serves as a prerequisite for knowledge graphs in a federated environment to learn embedding knowledge from other knowledge graphs. We assume that information regarding aligned entities is provided via a private set intersection method [15], and that this information is securely stored on the main server, meaning that the unique set of entities across all clients is preserved on the server. It is important to emphasize that, similar to other federated learning efforts [11], privacy protection was not the primary focus of this study. However, standard privacy protection measures can still be integrated into our framework [15]. This approach not only enhances the efficiency and accuracy of knowledge graph link prediction but also ensures the protection of data security and user privacy.

B. FRAMEWORK OPERATION FLOW

In our framework, data initialization is the first step. Initially, clients generate a relation embedding matrix R_0 from the relation set $\mathcal{R}^c$ in their knowledge graph $\mathcal{G}^c = \{\mathcal{E}^c, \mathcal{R}^c, \mathcal{T}^c\}$, based on predetermined upper and lower bounds. Subsequently, the server generates an entity embedding E_0 from the saved entity set according to predetermined upper and lower bounds. Here, the number of rows in the matrix represents the index and the number of columns represents the preset embedding dimensions. The subscript "0" indicates the embeddings generated during initialization. Next, the server distributes E_0 to all clients, ensuring that they begin training at the same entity-embedding starting point.

With these preparations, the clients can begin the embedding training process. Using the received entity embedding matrix E_0 and its own initialized relation embedding matrix R_0, the triplet set $\mathcal{T}^c$ was trained in batches of predetermined sizes. In this training, the triplet set $\mathcal{T}^c$ is matched with the entity and relation embeddings through indexing. The operational flow of the FedLTailor framework is illustrated in Figure 1 and the specific training process of the framework is described below.

A self-adversarial negative-sampling loss function [8] was used during the training process on the client side. In this method, the scores of positive and negative samples are required. The triplet $\mathcal{T}^c$ held by each client acts as a positive sample, whereas negative samples are constructed from known positive samples that do not appear in $\mathcal{T}^c$. That is, based on the known positive sample (head entity, relation, tail entity), denoted as (h, r, t), we change the tail entity t to obtain a new triplet $\left(h, r, t'\right)$. This newly constructed triplet did not exist among the positive samples, indicating that it is a valid negative sample. The designated embedding learner is then used to score the positive and negative samples. The loss function was calculated based on the scores obtained for the positive and negative samples. The formula used was as follows:

$$L_{kge}(h, r, t) = -\log\sigma\left(s(h, r, t) + \gamma\right) - \sum_{i=1}^{k} p\left(h, r, t_i'\right) \log\sigma\left(-s\left(h, r, t_i'\right) - \gamma\right), \tag{1}$$

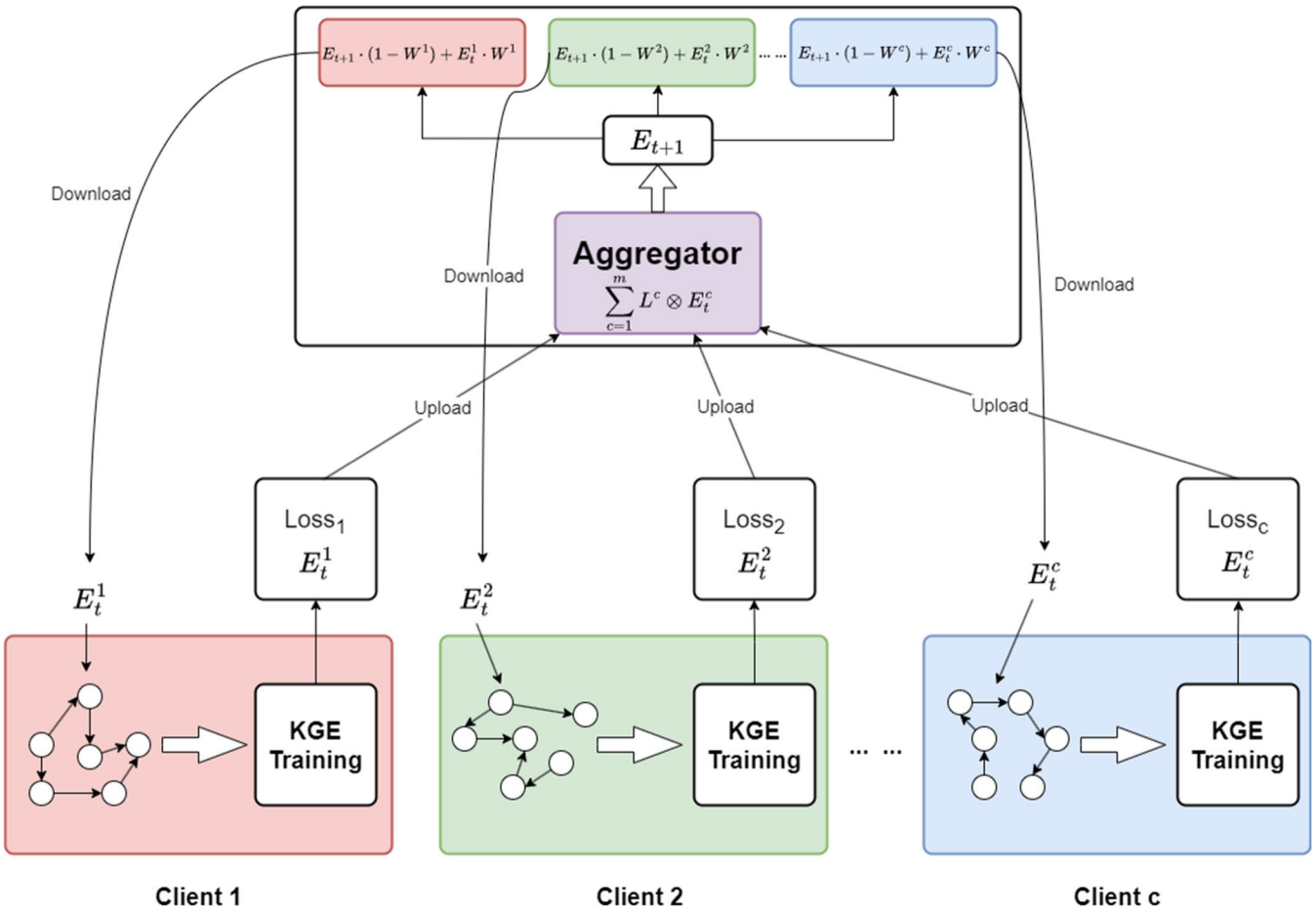

FIGURE 1. Overall procedure of one round in FedLTailor.

Here, γ represents a margin: the number of negative samples is determined by parameter k, and $p\left(h, r, t_j'\right)$ are the specified weights for each negative sample, representing the relative importance of the negative sample within the entire set of negative samples. The formula used was as follows:

$$p\left(h, r, t_j'\right) = \frac{\exp\left(\beta s\left(h, r, t_j'\right)\right)}{\sum_i \exp\left(\beta s\left(h, r, t_i'\right)\right)}, \tag{2}$$

where β is the temperature coefficient. A contrastive learning strategy [12] was integrated into the loss function. This means that the loss introduces a comparison between global and local embeddings using the following formula:

$$L_{con} = -\log \frac{\exp\left(\frac{\mathrm{sim}(\mathrm{E_t^c}, \mathrm{E_t})}{\tau}\right)}{\exp\left(\frac{\mathrm{sim}(\mathrm{E_t^c}, \mathrm{E_t})}{\tau}\right) + \exp\left(\frac{\mathrm{sim}(\mathrm{E_t^c}, \mathrm{E_{t-1}^c})}{\tau}\right)}, \tag{3}$$

Here, τ represents the temperature coefficient, and sim is the cosine similarity. The final loss function is a combination of the self-adversarial negative sampling loss function and contrastive loss, using the following formula:

$$\mathcal{L}(T) = \sum_{(h,r,t)\in\mathcal{T}} \mathcal{L}_{kge}(h, r, t) + \mu \mathcal{L}_{con}, \tag{4}$$

Here, μ controls the weight of contrastive loss. During the client training process, our goal was to increase the score for positive samples and decrease the score for negative samples. After completing client training, the new entity embeddings, $\mathrm{E_t^c}$ were uploaded to the central server. However, considering the heterogeneity in the embedding quality from different clients, directly assigning the same weight for aggregation to entity embeddings of varying quality may lead to suboptimal results. To address this issue, we propose a dynamic weight adjustment strategy as an alternative to the traditional average weight aggregation method. By applying the losses from different clients to a Temperature-scaled Softmax function, a new set of aggregation weights was obtained, where clients with higher losses received smaller weights and those with lower losses received larger weights. The calculation formula is as follows:

$$\mathrm{E_{t+1}} \longleftarrow \sum_{c=1}^{m} L^c \otimes \mathrm{E_t^c}, \tag{5}$$

Here, m is the number of selected clients, $\otimes$ represents the Hadamard product, and the weight vector L^c is determined based on the losses uploaded by each client by calculating the corresponding weight for every entity in E_t^c. This is defined as follows:

$$L_i^c = \begin{cases} 0, & \text{if} v_i^c = 0, \\ \dfrac{\exp\left(\frac{\alpha - loss_c}{\delta}\right)}{\sum_{c=1}^{x} \exp\left(\frac{\alpha - loss_c}{\delta}\right)}, & \text{if} v_i^c = 1, \end{cases}$$
$$\text{where } v_i^c = \begin{cases} 1, & \text{if} \varepsilon_i^{fed} \in \varepsilon^c \\ 0, & \text{if} \varepsilon_i^{fed} \notin \varepsilon^c, \end{cases} \quad (6)$$

where δ is a temperature parameter, and α is a predetermined constant value. By subtracting the corresponding loss from α, the loss can be intuitively converted into a quality score, which makes it easier to interpret and understand. $x = \sum_{c=1}^{m} v_i^c$, where v^c is the entity vector of client c, indicating the entities owned by that client. If $v_i^c = 0$, then the client does not contain the entity. If $v_i^c = 1$, then the client possesses that entity.

In studying FedEC, it was observed that by incorporating contrastive learning to control the distance between global and local embeddings, it effectively adjusted the differences between global and local embeddings in handling heterogeneity issues. Inspired by this, we chose to introduce a personalized strategy in the aggregation function by adding local embeddings to the global embeddings, thus ensuring that the distance between the global and local embeddings is effectively controlled. The formula used is as follows:

$$E_{t+1}^c \longleftarrow E_{t+1} \cdot \left(1 - W^c\right) + E_t^c \cdot W^c, \quad (7)$$

$$W^c = \frac{T^c + \frac{N^c}{N}}{2}, T^c = \frac{\exp\left(\frac{\alpha - loss_c}{\delta}\right)}{\sum_{c=1}^{m} \exp\left(\frac{\alpha - loss_c}{\delta}\right)}, \quad (8)$$

Here, T^c differs from the previously mentioned L^c; L^c represents a weight vector, whereas T^c is merely a constant derived from the loss uploaded by clients. N^c/N is the ratio of the number of triplets in a certain client to the total number of triplets across all the clients. By blending global embeddings with a certain proportion of local embeddings and distributing them to clients, we aimed to narrow the gap between the global and local embeddings. This method can enhance the adaptability of global embeddings and address data-heterogeneity issues to a certain extent.

C. INTEGRATION PART

In the federated and standalone environments, the obtained embeddings exhibited different characteristics. Integrating the embeddings learned in a federated environment with those learned in a standalone environment yielded superior results, a finding that was confirmed by FedEC. Inspired by the integration strategy of FedEC, we conducted a series of experiments and found that integrating different embedding learners could produce better embedding effects, making the integrated embeddings superior to those obtained using a Single method. We discovered that embeddings learned in a federated environment by various embedding learners (such as TransE, DistMult, ComplEx, and RotatE) could be integrated with embeddings learned in a standalone environment using RotatE to achieve better results. The methods include adaptive ensemble and direct ensemble. It is important to emphasize that this integration process is conducted after the training of embeddings is complete, and not as part of the training. The uniqueness of this approach lies in optimizing the overall performance through post-training integration, without affecting the independent training of each model. Our research not only confirms the effectiveness of the integration strategy in enhancing the performance of knowledge graph completion tasks but also showcases the complementary advantages of different embedding methods in a federated environment.

Direct ensemble Method: This method involves directly merging the triplet prediction scores $S^{(fed)}(h, r, t)$ obtained in the federated learning environment with the triplet prediction scores $S^{(sig)}(h, r, t)$ obtained in the non-federated standalone environment.

$$S^{(ens)}(h, r, t) = \frac{S^{(fed)}(h, r, t) + S^{(sig)}(h, r, t)}{2}. \quad (9)$$

Adaptive ensemble Method: This approach achieves targeted fusion based on their respective importance by weight training the triplet prediction scores $S^{(fed)}(h, r, t)$ obtained in the federated learning environment and the triplet prediction scores $S^{(sig)}(h, r, t)$ obtained in the non-federated standalone environment.

$$S^{(ens)}(h, r, t) = w_1 S^{(fed)}(h, r, t) + w_2 S^{(sig)}(h, r, t). \quad (10)$$

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

Subgraphs were extracted from the traditional FB15k-237 [16] and NELL-995 [17] benchmarks to construct benchmark datasets for federated knowledge graphs. Triplets were randomly divided among three different clients based on various relations in the graph and were named FB15k-237-Fed3 and NELL-995-Fed3. To further investigate the effectiveness of FedLTailor in environments with different numbers of clients, additional subdivisions were made, generating two subsets: FB15k-237-Fed5 and FB15k-237-Fed10. The data for each client were divided into training, validation, and test sets with ratios of 0.8,0.1, and 0.1, respectively. The relevant dataset statistics are listed in Table 1. s

TABLE 1. Statistics of datasets.

Dataset	#C	#Rel	#Ent	#Tri
FB15k-237-Fed3	3	79	12595.3	103373
FB15k-237-Fed5	5	47.4	11260.4	62023.2
FB15k-237-Fed10	10	23.7	8340.1	31011.6
NELL-995-Fed3	3	66.7	33453.7	51404.3

#C represents the number of clients, #Rel represents the average number of relations in the clients, #Ent represents the average number of entities in the clients, #Tri represents the average number of triples in the clients.

To evaluate the performance of each client, the Mean Reciprocal Rank (MRR) and Hits@N metric for link ranking were selected as evaluation criteria. Our focus on link prediction is primarily aimed at predicting tail entities (h, r, ?). To ensure the accuracy of the predictions, we filtered out combinations that had already appeared in the triplet set, thus concentrating on predicting triplets that had not yet appeared.

To demonstrate the effectiveness of our FedLTailor framework, we conduct comparisons in four settings: federated, standalone, FedEC, and FedE. In these settings, we compared classic embedding learners such as TransE, RotatE, DistMult, and ComplEx. The standalone version refers to each client independently conducting embedding training based solely on its own knowledge graph, without involving any global aggregation updates. Conversely, the federated version involves aggregating triplets from all clients to conduct embedding training collectively. However, this approach violates the privacy protection principles. FedE represents the first attempt to apply the FedAvg strategy to federated knowledge graph embeddings. FedEC is an improvement observed after when FedE encounters issues with data heterogeneity. This approach overcomes the challenge of data heterogeneity by introducing contrastive learning.

For the experimental parameter settings, we used the Adam [18] optimizer with a learning rate of 0.001. The number of training epochs E for the local clients was set to three, and the batch size B was set to 512. The local training loss parameter μ was 0.1, the contrastive learning temperature τ was 0.2, the global aggregation temperature δ was 0.55, and the constant α was 1. For the FB15k-237-Fed3, NELL-995-Fed3, and FB15k-237-Fed5 datasets, we set the client selection ratio F to 1 in each round. For the FB15k-237-Fed10 dataset, the ratio is set to 0.5. In the experimental setup, we set the embedding dimension for the TransE and DistMult embedding learners to 128, and for RotatE and ComplEx, the embedding dimension was set to 256. In addition, we fixed margin γ to 10, set negative sampling temperature β to 1, and set the number of negative samples to 256. During the training process, we used a validation set to assess the prediction accuracy. Specifically, for Single and Collective environments, we evaluated the validation set every 10 epochs, and in the training of FedLTailor, the validation set evaluations were conducted every 5 rounds. If the evaluation result on the validation set reached a maximum in any round, and this maximum remained unchanged in the subsequent 15 evaluations, we terminated the training.

B. EXPERIMENTAL RESULTS

Table 2 summarizes the prediction results for FB15k-237-Fed3 and NELL-995-Fed3 datasets. Each client had its own test set, and we presented a comprehensive summary of the prediction results from all clients. Notably, FedLTailor and FedEC provided two different sets of results: direct ensemble and adaptive ensemble.

According to the prediction results in Table 2, FedLTailor significantly outperformed the standalone knowledge graph embedding methods in the link- prediction task.

First, it was observed that for the FB15k-237-Fed3 dataset, compared with the standalone versions, FedLTailor achieved relative increases in the MRR scores for TransE, DistMult, ComplEx, and RotatE of 9.50%, 11.75%, 13.59%, and 5.99%, respectively. For the NELL-995-Fed3 dataset, the growth rates of the four methods are 20.48%, 34.74%, 35.07%, and 16.67%, respectively.

Second, compared with the collective version, FedLTailor achieved MRR increases on the FB15k-237-Fed3 dataset of 3.00%, 8.11%, 6.64%, and 2.79% respectively, whereas for the NELL-995-Fed3 dataset, these increases were 10.12%, 15.39%, 14.07%, and 2.64%, respectively.

Furthermore, compared with the FedE method, FedLTailor achieved MRR growths on the FB15k-237-Fed3 dataset of 2.50%, 3.01%, 4.19%, and 0.48%, respectively. For the NELL-995-Fed3 dataset, the increases were 8.58%, 13.69%, 7.66%, and 1.73%.

Finally, compared to the FedEC method, FedLTailor's MRR increases for the direct ensemble (DE) method on the FB15k-237-Fed3 dataset were 1.74%, 0.49%, 2.18%, and 0.20%, respectively, and for the adaptive ensemble (AE) method, the increases were 2.67%, 3.87%, 3.64%, and 0.64%, respectively. On the NELL-995-Fed3 dataset, DE increases were 7.67%, 11.81%, 5.42%, and 0.79%, and AE increases were 3.77%, 13.27%, 7.63%, and 2.19%.

In summary, these data highlight the exceptional performance of FedLTailor in the link prediction task, surpassing not only the standalone and collective versions, but also other federated learning methods such as FedE and FedEC.

Analyzing the data from Table 2, FedLTailor+ shows a clear advantage in the link prediction performance. Specifically, for the direct ensemble (DE) method, FedLTailor+ showed an improvement in the predictive performance over FedEC+. The MRR increase for FedLTailor+ compared with FedEC+ was 0.31%. Furthermore, for the NELL-995-Fed3 dataset, the increase is 0.79%.

Similarly, FedLTailor+ demonstrated a superior performance when considering the adaptive ensemble (AE) method. Across the entire dataset, the increase relative to that of FedEC+ was 1.07%. For the NELL-995-Fed3 dataset, the increase in FedLTailor+ over FedEC+ was more significant 1.47%.

Overall, whether hrough DE or AE integration methods, FedLTailor+ is superior to FedEC+, especially on the NELL-995-Fed3 dataset, where its advantages are more pronounced.

It can be clearly observed from Table 3 that when employing DistMult as the embedding learner and in environments with a greater number of clients, FedLTailor demonstrated outstanding performance in the link prediction task.

TABLE 2. Results on FB15k-237-Fed3 and NELL-995-Fed3. Numbers highlighted in red denote the best results among different KGE methods.

KGE	Setting	FB15k237-Fed3				NELL-995-Fed3			
		MRR	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10
TransE	Single	0.4074	0.2843	0.5546	0.6430	0.3311	0.1959	0.4886	0.5862
	Collective	0.4331	0.3059	0.5866	0.6785	0.3622	0.2247	0.5222	0.6231
	FedE	0.4352	0.3065	0.5930	0.6827	0.3674	0.2209	0.5365	0.6428
	FedEC(DE)	0.4364	0.3051	0.5972	0.6865	0.3705	0.2231	0.5424	0.6451
	FedEC(AE)	0.4345	0.3057	0.5902	0.6819	0.3708	0.2253	0.5391	0.6435
	FedLTailor(DE)	0.4440	0.3100	0.6067	0.6932	0.3989	0.2541	0.5669	0.6655
	FedLTailor(AE)	0.4461	0.3114	0.6112	0.6960	0.3848	0.2390	0.5554	0.6567
DistMult	Single	0.3985	0.2992	0.5145	0.5908	0.3078	0.2231	0.4036	0.4763
	Collective	0.4119	0.3050	0.5380	0.6209	0.3595	0.2622	0.4699	0.5490
	FedE	0.4323	0.3243	0.5599	0.6406	0.3648	0.2653	0.4769	0.5585
	FedEC(DE)	0.4327	0.3235	0.5634	0.6447	0.3564	0.2568	0.4708	0.5513
	FedEC(AE)	0.4287	0.3187	0.5612	0.6415	0.3662	0.2682	0.4778	0.5578
	FedLTailor(DE)	0.4348	0.3232	0.5691	0.6517	0.3985	0.2944	0.5150	0.6032
	FedLTailor(AE)	0.4453	0.3309	0.5819	0.6665	0.4148	0.3071	0.5360	0.6247
ComplEx	Single	0.3916	0.2900	0.5088	0.5934	0.3187	0.2254	0.4209	0.5114
	Collective	0.4171	0.3111	0.5404	0.6262	0.3773	0.2737	0.4933	0.5791
	FedE	0.4269	0.3200	0.5519	0.6372	0.3998	0.2960	0.5187	0.6104
	FedEC(DE)	0.4306	0.3226	0.5573	0.6413	0.3982	0.2921	0.5199	0.6108
	FedEC(AE)	0.4292	0.3216	0.5542	0.6409	0.3999	0.2948	0.5222	0.6113
	FedLTailor(DE)	0.4400	0.3301	0.5685	0.6518	0.4198	0.3094	0.5457	0.6352
	FedLTailor(AE)	0.4448	0.3321	0.5779	0.6620	0.4304	0.3213	0.5547	0.6444
RotatE	Single	0.4311	0.3182	0.5649	0.6480	0.3875	0.2810	0.5061	0.5968
	Collective	0.4445	0.3198	0.5960	0.6855	0.4404	0.3262	0.5702	0.6637
	FedE	0.4547	0.3267	0.6097	0.6976	0.4443	0.3240	0.5852	0.6786
	FedEC(DE)	0.4540	0.3274	0.6089	0.6948	0.4424	0.3230	0.5812	0.6731
	FedEC(AE)	0.4540	0.3256	0.6105	0.6974	0.4423	0.3193	0.5851	0.6784
	FedLTailor(DE)	0.4549	0.3284	0.6091	0.6951	0.4459	0.3287	0.5812	0.6765
	FedLTailor(AE)	0.4569	0.3287	0.6150	0.6994	0.4520	0.3342	0.5875	0.6837
	FedEC+(DE)	0.4568	0.3334	0.6075	0.6950	0.4666	0.3526	0.5976	0.6892
	FedEC+(AE)	0.4490	0.3181	0.6094	0.6950	0.4558	0.3284	0.6056	0.6989
	FedLTailor+(DE)	0.4582	0.3372	0.6070	0.6912	0.4703	0.3560	0.6010	0.6902
	FedLTailor+(AE)	0.4538	0.3253	0.6115	0.6962	0.4625	0.3385	0.6094	0.7008

TABLE 3. Results on FB15k-237-Fed5 and FB15k-237-Fed10 with DistMult as the KGE method. Numbers highlighted in red denote the best results.

Setting	FB15k-237-Fed5				FB15k-237-Fed10			
	MRR	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10
Single	0.3867	0.2869	0.5010	0.5793	0.3777	0.2783	0.4916	0.5709
Collective	0.4073	0.2990	0.5338	0.6198	0.4004	0.2880	0.5331	0.6225
FedE	0.4055	0.2917	0.5420	0.6299	0.4018	0.2851	0.5419	0.6282
FedEC(DE)	0.4210	0.3112	0.5504	0.6347	0.4089	0.2932	0.5446	0.6338
FedEC(AE)	0.4058	0.2894	0.5447	0.6320	0.4039	0.2853	0.5448	0.6336
FedLTailor(DE)	0.4234	0.3056	0.5650	0.6509	0.4171	0.2955	0.5626	0.6540
FedLTailor(AE)	0.4287	0.3113	0.5673	0.6552	0.4225	0.3008	0.5699	0.6584

First, on the FB15k-237-Fed5 dataset, the FedLTailor's MRR score increased by 10.87% compared to the standalone version. Compared to the collective version, the MRR increased by 5.25%. Furthermore, compared with FedE, FedLTailor's MRR increase for this dataset is 5.73%.

Specifically, compared with FedEC's direct ensemble (DE) method, FedLTailor showed an MRR increase of 0.57% on the FB15k-237-Fed5 dataset. When considering the adaptive ensemble (AE) method, the comparison with FedEC was even more significant, with FedLTailor's MRR increase reaching 5.64%.

In summary, FedLTailor demonstrated clear advantages in all comparisons, particularly when compared with FedEC's direct ensemble and adaptive ensemble methods, where its performance improvements are especially prominent.

C. MODEL ANALYSIS

In this section, we focus on exploring the impact of different client selection ratios F (i.e., only a subset of clients is selected for aggregation in each round) on the performance. For this purpose, we selected TransE as the embedding learner, and FB15k-237-Fed10 as the experimental dataset. The experiment considered multiple client selection ratios, including 0.3, 0.5, 0.8, and 1, and recorded the performance of Hits@10 on the validation set every 5 rounds.

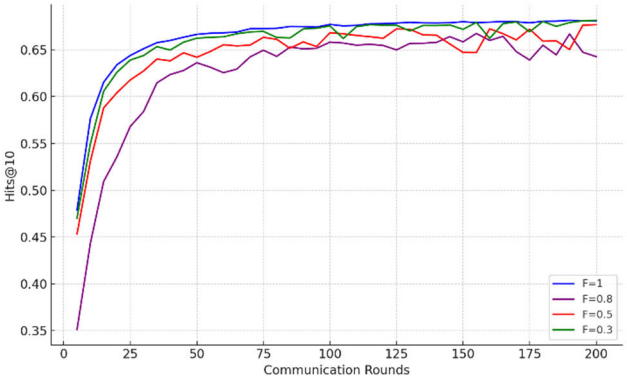

FIGURE 2. The Hits@10 results with different client selection ratios F.

Through the visualization of the data in Figure 2, we observed some interesting phenomena. First, as the client selection ratio F increases, the convergence speed of the model accelerates and a higher Hits@10 performance can be achieved. This suggests that a higher degree of parallelism may speed up the learning process and improve final performance. However, on the other hand, increasing the ratio F also means greater computational resources and time consumption. Therefore, determining the appropriate client selection ratio F is a balancing act that requires finding a balance between the performance improvement and resource consumption.

In summary, the research in this section reminds us that selecting the appropriate client selection ratio F is crucial when practically deploying FedLTailor, ensuring high performance while also considering the actual resource and time constraints.

Analyzing the impact of the aggregation weight vector L^c generated by different temperatures δ on the experimental results, the experiment employs DistMult as the embedding model and is conducted on four datasets, using Hits@10 as the evaluation metric. The role of the temperature coefficient δ in weight allocation and aggregation strategy and its impact on the overall performance of the model was thoroughly understood. Increasing the temperature δ leads to more equalized aggregation weights, while decreasing the temperature δ results in a sharper distribution of aggregation weights,

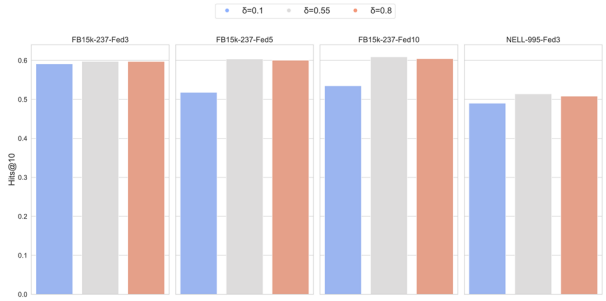

FIGURE 3. The results with different temperature δ.

meaning that more weight is given to the better-performing clients. Therefore, the temperature value δ is intentionally lowered to achieve a sharper and more discriminative weight distribution. According to the experimental results shown in Figure 3, when the temperature parameter δ is set at approximately 0.55, the model demonstrates better performance. This indicates that by finely tuning the temperature parameter, the overall performance of the model can be effectively optimized.

V. CONCLUSION

This study introduces an innovative federated learning framework, FedLTailor, that effectively addresses the challenges posed by data heterogeneity and the uneven quality of client embeddings through a dynamic weight adjustment strategy and a personalized fusion strategy. Our approach not only optimizes the performance of the global model but also adapts better to the differences among various clients, enhancing the model's robustness and adaptability in diverse data environments.

The experimental results demonstrate that FedLTailor exhibits excellent performance across multiple federated knowledge graph datasets. By employing different knowledge graph embedding techniques, we confirmed the broad adaptability and effectiveness of FedLTailor, which offers a powerful and flexible solution for knowledge graph completion in federated environments.

Overall, the introduction of FedLTailor represents not only technical innovation, but also shows strong potential for practical applications. Its successful implementation provides new perspectives and tools to address key issues in federated learning. Future work will focus on further optimizing the FedLTailor framework to address a wider range of challenges and to explore its potential applications in other federated learning scenarios.

Although FedLTailor has achieved promising experimental results in federated knowledge graph embedding tasks, new challenges are emerging with the resurgence of knowledge engineering and the rapid expansion of knowledge graphs. The knowledge graphs used in the experiments are represented purely symbolically, which has shown limitations in enabling machines to understand complex real-world scenarios. Combining federated learning with multimodal knowledge graphs (MMKGs) is a crucial step towards achieving human-level machine intelligence. However, the complexity of multimodal data, such as text, images, and audio, may introduce more severe data heterogeneity issues. This heterogeneity could result in the aggregated global model being unable to effectively address the specific needs of each client, posing a significant challenge.

REFERENCES

[1] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. AAAI*, Jun. 2014, vol. 28, no. 1.
[2] C. Mingyang, Z. Wen, and Y. Zonggang, "Federated knowledge graph completion via embedding-contrastive learning," *Knowl.-Based Syst.*, vol. 252, Sep. 2022, Art. no. 109459.

[3] B. McMahan, E. Moore, and D. Ramage, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[4] N. Zhang, S. Deng, and Z. Sun, "Relation adversarial network for low resource knowledge graph completion," in *Proc. Web Conf.*, 2020, pp. 1–12.
[5] Y. Hao, Y. Zhang, and K. Liu, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, 2017, pp. 221–231.
[6] F. Zhang, N. J. Yuan, and D. Lian, "Collaborative knowledge base embedding for recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 353–362.
[7] A. Bordes, N. Usunier, and A. Garcia-Duran, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, 26.
[8] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "RotatE: Knowledge graph embedding by relational rotation in complex space," 2019, *arXiv:1902.10197*.
[9] B. Yang, W.-T. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," 2014, *arXiv:1412.6575*.
[10] T. Trouillon, J. Welbl, and S. Riedel, "Complex embeddings for simple link prediction," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2071–2080.
[11] T. Li, A. K. Sahu, and M. Zaheer, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.
[12] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10708–10717.
[13] J. Zhang, Y. Hua, and H. Wang, "FedALA: Adaptive local aggregation for personalized federated learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 9, 2023, pp. 11237–11244.
[14] M. Chen, W. Zhang, and Z. Yuan, "Fede: Embedding knowledge graphs in federated setting," in *Proc. 10th Int. Joint Conf. Knowl. Graphs*, 2021, pp. 80–88.
[15] C. Chen, J. Cui, G. Liu, J. Wu, and L. Wang, "Survey and open problems in privacy preserving knowledge graph: Merging, query, representation, completion and applications," 2020, *arXiv:2011.10180*.
[16] K. Toutanova, D. Chen, and P. Pantel, "Representing text for joint embedding of text and knowledge bases," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1499–1509.
[17] W. Xiong, T. Hoang, and W. Yang Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," 2017, *arXiv:1707.06690*.
[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

HONG ZHENG received the Ph.D. degree in engineering from Jilin University. She is currently a Lecturer and an Associate Professor with the School of Computer Science and Engineering, Changchun University of Technology. Her research interests include machine learning and deep learning algorithms, with a current focus on natural language processing and federated learning, as well as their applications in intelligent finance and assisted medical fields.

SHANQIN LI is currently pursuing the Master of Engineering degree with the School of Computer Science and Engineering, Changchun University of Technology. His research interests include the application of federated learning and knowledge graphs in various domains.