Temporal Link Prediction Using Graph Neural Networks

Leman Zakaryayeva
Department of Computer Engineering
Akdeniz University
Antalya, Turkey

***Abstract*—This project focuses on the problem of Temporal Link Prediction, which involves predicting the existence of an edge between two nodes in a temporal graph within a given time window. We employ the Graph Neural Network (GNN) model, specifically the GConvGRU model, which combines Graph Convolutional Networks (GCNs) and Gated Recurrent Units (GRUs) to handle the temporal and structural aspects of the graph. The model is trained and evaluated on two datasets, with the goal of predicting edge probabilities across temporal graphs. The results show the model's potential, although further improvements are needed.**

***Index Terms*—Temporal Link Prediction, Graph Neural Networks, GConvGRU, Temporal Graphs, AUC, Machine Learning**

I. Introduction

In real-world systems, relationships between entities often evolve over time, creating **temporal graphs**. Temporal Link Prediction is the task of predicting the formation of future edges based on the historical interactions between nodes. In this project, we aim to address this problem by applying **Graph Neural Networks (GNNs)**, which are particularly suitable for handling graph-structured data, to predict edge formation in temporal graphs. We use the **GConvGRU** model, which incorporates both the spatial (graph) and temporal (time-series) aspects of the data.

II. Methodology

The methodology for this project can be divided into the following steps:

A. Data Preprocessing

The preprocessing steps included handling missing node features and addressing edge feature sparsity:

- **Missing Node Features:** We replaced any negative values (-1) in the node features with zeros to handle missing data.
- **Edge Feature Sparsity:** The dataset contains edge types, and their features were handled by averaging the edge-type features based on the edge type.
- **Filtering Valid Nodes:** Only valid nodes with corresponding features were kept for edge prediction.

B. Model Architecture

The **GConvGRU** model was used, which combines **Graph Convolutional Networks (GCN)** for capturing graph structure and **Gated Recurrent Units (GRU)** for handling temporal dynamics. This model learns both the graph's spatial structure and the temporal evolution of the edges. The **GConvGRU** was implemented using the **PyTorch Geometric Temporal** library.

C. Training Process

We trained the model on both datasets using a shared architecture with adaptable hyperparameters. The training procedure included:

- **Hyperparameter tuning:** The learning rate, number of epochs, and other parameters were fine-tuned to optimize performance.
- **Optimization:** We used the **Adam optimizer** with a learning rate of 0.0003 and implemented a learning rate scheduler to dynamically adjust the learning rate during training.
- **Loss function:** The **Binary Cross-Entropy (BCE) loss** with logits was used, with a positive class weight to account for the imbalance in the dataset.

D. Evaluation

The model was evaluated using **AUC (Area Under the Curve)** score, which measures the ability of the model to distinguish between positive and negative edge formations. The evaluation was performed on test data, with predictions made for edge formation probabilities.

E. Optimization

Further optimization steps include fine-tuning the model's hyperparameters (e.g., increasing dropout rates, adjusting the learning rate) and exploring other model architectures such as **Graph Attention Networks (GAT)** and **GCN**.

III. Results

The model was evaluated on **Dataset A** and achieved an **AUC score of 0.5004**. This score suggests that the model's performance was similar to random predictions, indicating room for improvement. Future work will focus on tuning the model and experimenting with other GNN architectures to enhance its predictive power.

IV. Discussion and Future Work

Although the initial results are not satisfactory, the model demonstrates potential in handling temporal graph data. Future steps include:

- Experimenting with different GNN architectures (e.g., **GAT**, **GCN**).
- Further optimizing hyperparameters.
- Implementing more advanced temporal feature engineering and data augmentation techniques.

V. Conclusion

This project demonstrates the application of GNNs for **Temporal Link Prediction**. While the results are not optimal, the model provides a solid foundation for further improvement. Further research into different architectures and optimization strategies will be conducted to improve the model's performance.

VI. Acknowledgments

We would like to thank our professors and peers for their valuable feedback during the development of this project.

VII. Images and Visualizations

In the following sections, we present the images generated during the model's evaluation and training process.

A. Node Feature Shape

The shape of the node features is shown in Fig. 1. The dataset contains **19,442 nodes**, each represented by **9 features**. This large node set provides a complex structure for the model to learn from.

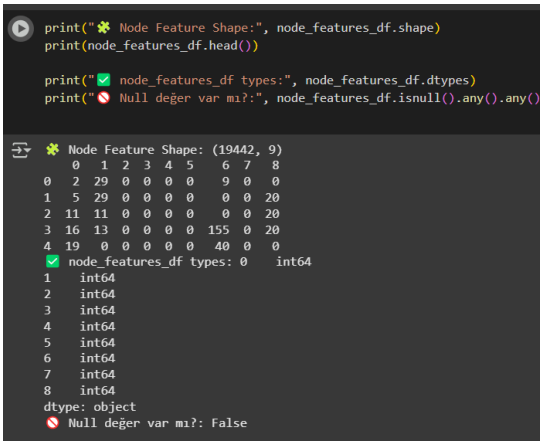

Fig. 1. Node Feature Shape

B. Epoch Loss Over Time

Fig. 2 shows the **loss values** during training over 15 epochs. The loss decreases consistently over time, which indicates that the model is learning from the data. However, the loss values are still relatively high, suggesting that the model needs further optimization to reduce errors and improve prediction accuracy.

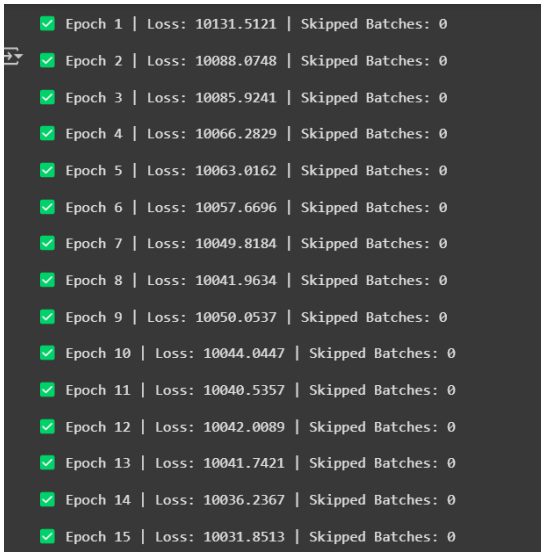

Fig. 2. Epoch Loss Over Time

C. Predictor Output Distribution

The histogram in Fig. 3 shows the distribution of the **predicted probabilities** for edge formation. As observed, most predictions are clustered around the 0.78-0.79 range, with a few outliers. This indicates that the model is producing relatively narrow probability estimates. To improve this, future work will focus on diversifying the model's predictions and improving its generalization.

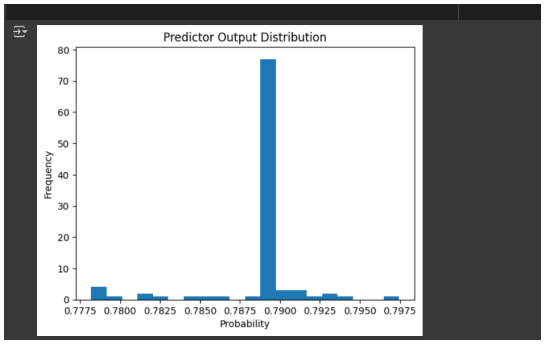

Fig. 3. Predictor Output Distribution

References

[1] Benedek Rozemberczki, et al., "PyTorch Geometric Temporal: A Library for Temporal Graph Learning," GitHub, 2020. [Online]. Available: https://github.com/benedekrozemberczki/pytorch$_geometric_temporal$.