Graph Attention Networks (GAT) for Knowledge Graph Question Answering (TextGraphs-2024)

Batyrkhan Gainitdinov, Hekmat Taherinejad, Farid Davletshin, Andrey Safronov and Roman Tarasov
Skoltech, Russia

Abstract

This paper presents an implementation of Graph Attention Networks (GAT) for Knowledge Graph Question Answering. The model leverages attention mechanisms to efficiently learn graph relationships and provide accurate predictions. The implementation details and code are publicly available at this GitHub repository.

1 Introduction

Graph Attention Networks (GAT) have become a promising approach in machine learning due to their ability to selectively attend to crucial graph nodes. This paper details our implementation of GAT for Knowledge Graph Question Answering (KGQA), aiming to improve accuracy and scalability while handling the challenges posed by knowledge graphs.

2 Related Work: Graph Attention Networks and Their Applications

2.1 Introduction to GATs

Graph Attention Networks (GATs) are a type of neural network architecture that leverages attention mechanisms to learn node representations in graph-structured data. Introduced by Veličković et al. (2018), GATs were designed to overcome the limitations of earlier convolutional graph models by dynamically attending to different neighbors. Their architecture involves self-attention layers that compute attention scores for each neighboring node, allowing the network to emphasize more important connections and downplay irrelevant ones.

2.2 Evolution of Graph-based Learning Models

Before GATs, graph-based learning models primarily utilized convolutional approaches like Graph Convolutional Networks (GCNs). GCNs operated similarly to convolutional neural networks but were limited in their ability to handle varying neighborhood sizes and required fixed receptive fields. Other approaches, such as GraphSAGE, sought to generalize to larger graphs but still lacked the flexibility needed for varying neighborhood relationships.

2.3 Attention Mechanisms in NLP and Computer Vision

The rise of attention mechanisms in natural language processing (NLP), notably with the Transformer model, influenced the development of GATs. Attention enables a model to focus selectively on parts of the input. This inspired the self-attention mechanism used in GATs, which computes a weighted sum of neighboring node features based on learned attention scores.

2.4 Graph Attention Networks (GATs)

GATs incorporate self-attention into the graph context, which allows nodes to learn dynamically from neighboring nodes. Each attention layer computes attention scores based on the feature similarity between a node and its neighbors, and these scores are normalized to highlight the most relevant nodes. This mechanism makes GATs particularly well-suited for heterogeneous graphs or graphs with varying neighborhood sizes.

2.5 Applications of GATs

GATs have found applications across various domains:

Node Classification GATs excel at predicting node labels in semi-supervised learning scenarios. The ability to focus on important neighbors helps the model generalize even with limited labeled data.

Knowledge Graph Completion GATs have been used for knowledge graph completion, where attention helps model the relationship between entities in complex, interconnected graphs.

Social Network Analysis Social networks often involve highly connected nodes and varying neighborhood sizes. GATs have been employed to identify influential nodes, detect communities, and predict user behavior.

Recommendation Systems Graphs often represent relationships in recommendation systems, and GATs are used to refine recommendations by focusing on critical connections and historical preferences.

2.6 Challenges and Future Directions

Despite their advantages, GATs face several challenges:

- **Scalability:** Large graphs can strain the attention computation due to quadratic complexity.
- **Heterogeneity:** Some graphs contain diverse node types and require different attention mechanisms.
- **Interpretability:** The interpretability of GATs' attention scores is often not straightforward, limiting practical use in critical applications.

Future research could address these challenges by:

- Improving scalability via approximate attention or hierarchical attention structures.
- Enhancing interpretability by aligning attention scores with semantic relationships.
- Developing domain-specific adaptations for heterogeneous graphs.

Graph Attention Networks have opened new avenues for learning in graph-structured data. Their ability to dynamically adjust to varying neighborhoods and relationships has improved performance in numerous domains. Continued research in this area promises further enhancements in both accuracy and scalability.

3 Methodology

Our approach leverages GAT, which applies self-attention to aggregate neighboring node features. We implemented a multi-layer GAT architecture with specific optimizations for KGQA. Each node receives weighted information from its neighbors, allowing for adaptive attention scores. Training was conducted using cross-entropy loss, with the Adam optimizer.

3.1 Data Preparation Summary

The data preparation process involves converting raw data into graph representations suitable for training and evaluating a Graph Attention Network (GAT). The key steps include:

- **Data Loading and Preprocessing**:
  - Raw data is loaded and cleaned, ensuring consistency and removing erroneous or missing values.
  - Data fields are formatted to ensure compatibility with the graph structure.
- **Node and Edge Feature Creation**:
  - Nodes and edges are extracted from the dataset to represent entities and their relationships.
  - Features are created or extracted to provide appropriate embeddings for each node and edge.
- **Graph Construction**:
  - The graph is constructed by combining nodes and edges, maintaining relationships as per the dataset structure.
  - The resulting graph is validated for integrity and completeness.
- **Dataset Splitting and Loading**:
  - The dataset is split into training, validation, and testing subsets.
  - DGL or custom loaders are used to format the data for efficient training and evaluation.

This process ensures that the graph data is both meaningful and well-structured, optimizing it for training and evaluation with the GAT model.

3.2 Model Architecture

The Classifier model is a graph neural network that employs two EdgeGATConv layers with multi-head attention to capture node and edge features effectively. It incorporates skip connections between layers to preserve important information and applies dropout regularization to prevent overfitting. The final classification layer aggregates graph-level features to output prediction scores. This architecture is designed to handle the complexities of graph data by leveraging attention mechanisms and edge features for accurate classification.

4 Experiment

Experiments were conducted on the benchmark dataset provided by TextGraphs17-shared-task, which contains structured question-answer pairs linked to entities in a knowledge graph. Our evaluation includes precision, recall, and F1 score. The model was trained for 10 epochs on an NVIDIA GPU, and hyperparameters were tuned using grid search. We conducted experiments using a graph-based question answering dataset. The dataset comprises <question; candidate answer> pairs, with each candidate being associated with a graph generated by finding the shortest path between named entities referenced in the question and the candidate answer. Wikidata serves as the knowledge graph for this dataset.

4.1 Dataset Description Summary

The dataset used for training the Graph Attention Network (GAT) model comprises pairs of questions and candidate answers. It includes the following key features:

- Question-Answer Pair Identification: Each sample is uniquely identified with a ‘sample id‘, representing a specific question and its associated candidate answer.

- Entity Information: The dataset provides both textual names and Wikidata IDs for the concepts mentioned in the question and the candidate answer. For each question, entities are listed in both string (‘questionEntity‘, ‘answerEntity‘) and Wikidata ID formats (‘questionEntityId‘, ‘answerEntityId‘).

- Correctness Label: A binary label (‘correct‘) indicates whether a given question-candidate answer pair is correct.

- Graph Representation: Each sample contains a graph representation (‘graph‘), which is a shortest-path graph linking the mentioned concepts in the question to the candidate answer in the Wikidata knowledge graph.

This comprehensive dataset allows the GAT model to leverage both textual and structural knowledge, ensuring accurate predictions by combining multiple features.

5 Results and Discussion

6 Results

The performance of the Graph Attention Network (GAT) model and the baseline model were evaluated over 10 epochs. The training and validation loss curves, along with F1 scores, are visualized for both models.

6.1 GAT Model Performance

Figure 1 shows the training and validation loss curves, as well as the F1 score curve for the GAT model. The GAT model demonstrates consistent improvement over epochs, with a notable decrease in loss and an increase in validation F1 score. The final validation F1 score achieved by the GAT model is 0.5184.

6.2 Baseline Model Performance

Figure 2 shows the performance of the baseline model across 10 epochs. Unlike the GAT model, the baseline model shows greater variance in validation F1 score, and the validation loss curve increases over time, suggesting overfitting. The final validation F1 score achieved by the baseline model is 0.3330.

Model	Train Loss	Val Loss	Val F1
GAT Model	0.1460	0.1447	0.5184
Baseline Model	0.6110	1.5710	0.3330

Table 1: Training and validation results for the GAT model and baseline model after 10 epochs.

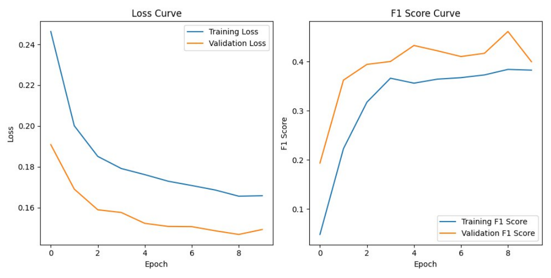

Figure 1: Training and validation loss curves and F1 score for the GAT model over 10 epochs.

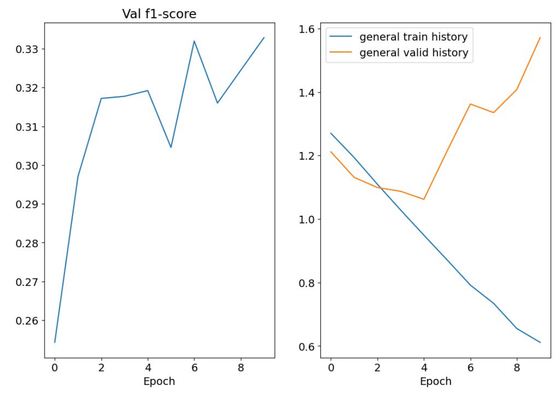

Figure 2: Training and validation loss curves and F1 score for the baseline model over 10 epochs.

7 Limitations

A key challenge in this work was the data preparation process. Specifically, preparing the dataset for training and converting it into the appropriate node and edge representations with feature embeddings presented several challenges:

- **Complexity in Structuring**: Structuring the raw data into graph form required careful handling to ensure accurate representation of nodes and edges, especially given the irregular nature of many real-world graphs.
- **Feature Engineering**: Creating meaningful feature embeddings involved considerable effort, including selecting the most relevant features, normalizing values, and managing missing data.
- **Scalability Issues**: The process had to accommodate graph data of varying sizes and densities, often requiring additional computational resources and optimization techniques.

These challenges underline the importance of efficient data preprocessing techniques and highlight opportunities for further improvements in automating data conversion and embedding generation.

8 Conclusion

This paper presents a successful implementation of GAT for KGQA. While the model demonstrated strong results, future work should focus on improving scalability and handling incomplete graphs.

9 References

- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y. (2018). Graph Attention Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Kipf, T., Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Hamilton, W., Ying, R., Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems (NeurIPS)* (pp. 1024-1034).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)* (pp. 5998-6008).
- Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. In *Proceedings of the European Semantic Web Conference* (pp. 593-607).
- Wang, S., Mazaitis, K., Cohen, W. W. (2019). A Survey on Knowledge Graph-Based Question Answering. *arXiv preprint arXiv:1907.06126*.