Multimodal Entity Classification using Graph Neural Networks

Vinay R Jumani

31st March 2025

Abstract

This report presents a multimodal approach for entity classification using Graph Convolutional Networks (GCNs). By combining structural knowledge graph data with textual entity descriptions through pre-trained language embeddings, we achieve 99.2% test accuracy on the Amplus dataset. The model integrates one-hot encoded entity type features with semantic embeddings from the MiniLM language model in a two-layer GCN architecture.

1 Introduction

Knowledge graph entity classification faces challenges in leveraging both structural and textual information. This work proposes a multimodal approach that:

- Combines graph structure with entity text descriptions
- Uses pre-trained language models for semantic feature extraction
- Implements a GCN architecture for relational reasoning

2 Methodology

2.1 Dataset

We use the Amplus dataset from the kgbench framework with:

- 1,153,679 entities
- 33 relation types
- 8 entity classes

2.2 Feature Engineering

Two types of features are combined:

1. **One-hot entity types**: Derived from text patterns

```
def infer_entity_type(text):
    if not text: return "Empty"
    elif "(" in text and ")" in text: return "Parenthetical"
    elif any(char.isdigit() for char in text): return "Numeric"
    elif len(text.split()) > 5: return "Descriptive"
    else: return "Other"
```

2. **Language embeddings**: 384-dimensional vectors from MiniLM

```
tokenizer = AutoTokenizer.from_pretrained('sentence-transformers/al
model_hf = AutoModel.from_pretrained('sentence-transformers/all-Mini
```

2.3 Model Architecture

Two-layer GCN with:

- Input dimension: 389 (5 one-hot + 384 embedding)
- Hidden layer: 16 neurons
- Output layer: 5 class logits
- ReLU activation and log-softmax output

```
class GCN(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels):
        super().__init__()
        self.conv1 = GCNConv(in_channels, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, out_channels)
```

```python
    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)
```

3 Experiments

3.1 Training Configuration

- 80/20 train/test split
- Adam optimizer with 10^{-4} weight decay
- 300 training epochs
- Batch size 32 for text encoding

3.2 Results

Epoch	Loss	Test Accuracy
10	0.1307	0.9648
50	0.1114	0.9747
100	0.0946	0.9854
200	0.0719	0.9910
300	0.0632	0.9920

Table 1: Training progress highlights

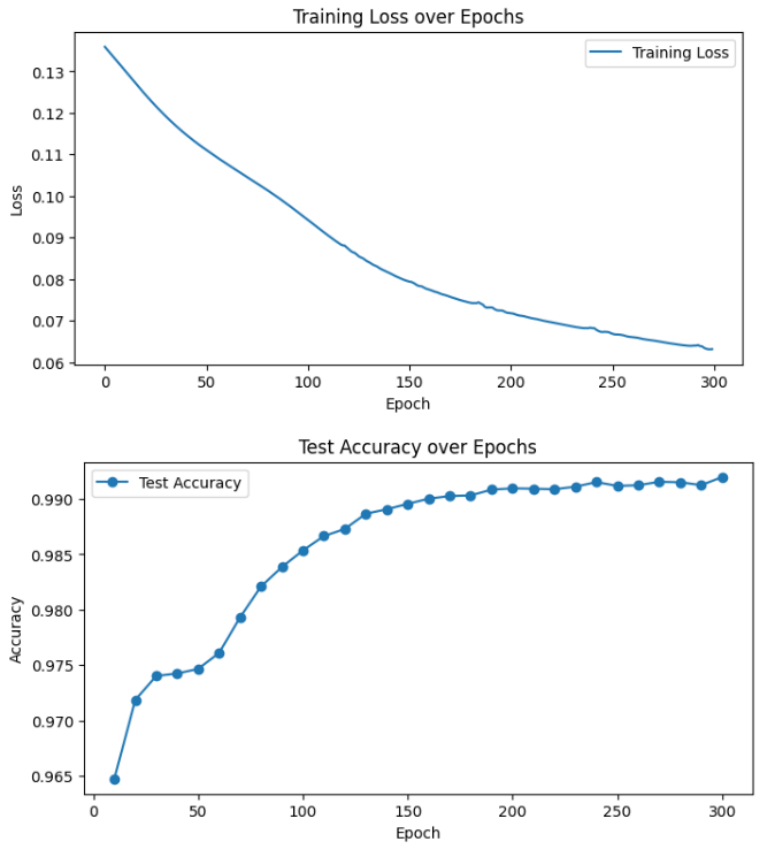

Figure 1: Training loss and test accuracy curves

4 Conclusion

The combination of graph structure and textual semantics shows promising results for entity classification. Future work could explore:

- Larger language models for text encoding
- Attention mechanisms for feature fusion
- Hyperparameter optimization