SAiDL Spring 2025
Induction Assignment:
Multi-Modality

Knowledge Graph Analysis and Node Classification

Submitted by:

Saumya Kathuria
ID: 2022A7PS1106G

Submitted to:

Society for Artificial Intelligence
and Deep Learning

April 7, 2025

Contents

1 Setting up the Environment

About This Assignment

This assignment explores multi-modal knowledge graph analysis and node classification. The dataset used is dmg777k, which contains information about monuments, images, and geographical data. The goal is to analyze the knowledge graph structure and build a GNN for node classification.

In this section, we'll go through the setup process for the multimodality task.

1.1 Importing the Necessary Packages

The environment used for the project has already been provided in form of **environment.yml** in the github repository itself.

1.2 Downloading the Data

The dmg777k dataset was loaded using the kgbench library. This dataset contains information on monuments, geographical features, and images, structured as a knowledge graph.

2 Exploratory Data Analysis

For this section, kindly refer to the notebook directly. Extensive data analyis along with well written decriptions and graphs are present there.

3 Preparing the Data

3.1 Sampling a Subset

- Randomly sample a small fraction of labeled entities from the training set.
- Use these entities as seeds to build a subgraph by expanding their neighborhood.
- Iteratively include entities reachable within a fixed number of hops (e.g., 2).
- Retain all triples where the subject or object is part of the expanded neighborhood.
- This preserves local graph structure while significantly reducing data size.

3.2 Creating the Embeddings for Labeled Nodes

To effectively represent the multi-modal nature of the knowledge graph, embeddings were generated for labeled entities using two separate pre-trained models — one for textual literals and another for image data.

- **Textual literals:** BERT (Bidirectional Encoder Representations from Transformers) was used to generate embeddings for entity URIs and associated text-based literals. The tokenizer and model were loaded from the `bert-base-uncased` checkpoint, and only the mean of the last hidden state was retained as the embedding.
- **Image literals:** Vision Transformer (ViT) was employed for entities containing image literals encoded in base64. Images were decoded, preprocessed, and passed through a pre-trained ViT model (`google/vit-base-patch16-224-in21k`) to extract feature embeddings.

Algorithm 1 Sampling a Subset of the Knowledge Graph

Require: Training data $\mathcal{T}$, triples $\mathcal{G}$, sample ratio r, number of hops H
Ensure: Subgraph triples $\mathcal{G}'$, sampled entities $\mathcal{E}'$
0: Extract unique labeled entities $\mathcal{E}_{\text{labeled}}$ from $\mathcal{T}$
0: $\mathcal{E}' \leftarrow \text{RandomSample}(\mathcal{E}_{\text{labeled}}, \max(1, \lfloor r \cdot |\mathcal{E}_{\text{labeled}}| \rfloor))$
0: $\mathcal{N} \leftarrow \mathcal{E}'$ {Initialize neighborhood with sampled entities}
0: **for** $h = 1$ to H **do**
0: $\quad \mathcal{N}_{\text{new}} \leftarrow \emptyset$
0: $\quad$ **for** each (s, p, o) in $\mathcal{G}$ **do**
0: $\quad\quad$ **if** $s \in \mathcal{N}$ or $o \in \mathcal{N}$ **then**
0: $\quad\quad\quad \mathcal{N}_{\text{new}} \leftarrow \mathcal{N}_{\text{new}} \cup \{s, o\}$
0: $\quad\quad$ **end if**
0: $\quad$ **end for**
0: $\quad \mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}_{\text{new}}$
0: **end for**
0: $\mathcal{G}' \leftarrow \{(s, p, o) \in \mathcal{G} \mid s \in \mathcal{N} \vee o \in \mathcal{N}\}$
0: **return** $\mathcal{G}'$, $\mathcal{N}$ =0

To optimize computational efficiency, only up to three literals per entity were processed, and embeddings from both the entity's URI and its available literals were averaged to form a unified representation.

3.3 Generating the Embeddings

The embedding generation process was applied to all sampled labeled entities within the filtered subset of the knowledge graph. Each entity's final representation was formed by:

- Extracting its URI and associated literals from the knowledge graph.
- Encoding the URI using BERT to capture semantic information.
- Parsing and embedding literal values (text or image), with appropriate handling of data types.
- Averaging the URI and literal embeddings to obtain a final vector representation.

This resulted in a set of dense, fixed-size embeddings for labeled entities, capturing both semantic and visual modalities. These embeddings were later used as input features for downstream tasks such as node classification.

4 The Node Classification Process

4.1 Defining the Model

We implement a two-layer Relational Graph Convolutional Network (RGCN) for node classification over a knowledge graph. The architecture is designed to incorporate relation-aware message passing and allows for efficient parameterization using basis decomposition.

4.1.1 RGCN Layer:

Each RGCN layer performs relation-specific message aggregation using relation-specific transformation matrices. When the number of relations is high, we employ basis decomposition to reduce the parameter count. Specifically, each relation's weight matrix is computed as a linear combination of a smaller number of basis matrices. The forward pass propagates transformed node features along relation-specific edges and aggregates them at the destination nodes using addition.

4.1.2 RGCN Model:

The overall model contains:

- A learnable embedding layer for all entities.
- Two RGCN layers with ReLU activation and dropout for regularization.
- A linear classifier head for predicting class probabilities for each node.

4.2 Data Preparation:

We preprocess the knowledge graph to obtain:

- A mapping from entity/relation IDs to indices.
- Edge indices and corresponding relation types.
- A dictionary of node-level labels and optional pretrained embeddings.

4.3 Training Procedure:

- We train the model using negative log-likelihood loss on labeled entities.
- Early stopping is employed based on validation F1 score, with learning rate scheduling to adapt to training dynamics.
- During training, entity embeddings are optionally initialized using pretrained representations.

4.4 Evaluation:

We evaluate the model on the labeled subset using accuracy and weighted F1 score. After evaluation, predictions are mapped back to their original entity IDs.

5 Results and Discussion

Classification Results

The RGCN model was trained on the multi-modal knowledge graph with the following parameters:

- Hidden layer size: 128
- Number of bases: 30
- Learning rate: 0.01
- Weight decay: 5e-4
- Training epochs: 40

The model achieved:

- Test Accuracy: 0.5547
- F1 Score: 0.4636

6 Conclusion

This assignment focused on the analysis and modeling of a multi-modal knowledge graph that integrates both geographical and image-based data. An initial exploratory data analysis was conducted to examine the graph's structure, including its entity types, relationships, and associated literals.

Subsequently, a RGCN was implemented to perform node classification within the graph which effectively utilized the graph's relational structure in conjunction with multi-modal features—comprising textual and visual information—to deliver strong classification performance.