Australian National University

Learning-based Attributed Graph Matching

Yanxi Lu

Supervisor: Dr. Qing Wang

Outline

- Background
  - Problem Definition
  - Existing methods and motivation
- Methodology
  - Overview
  - Computing node embedding
  - Matching
- Experimental Results
- Future Work

Background

- Attributed graphs

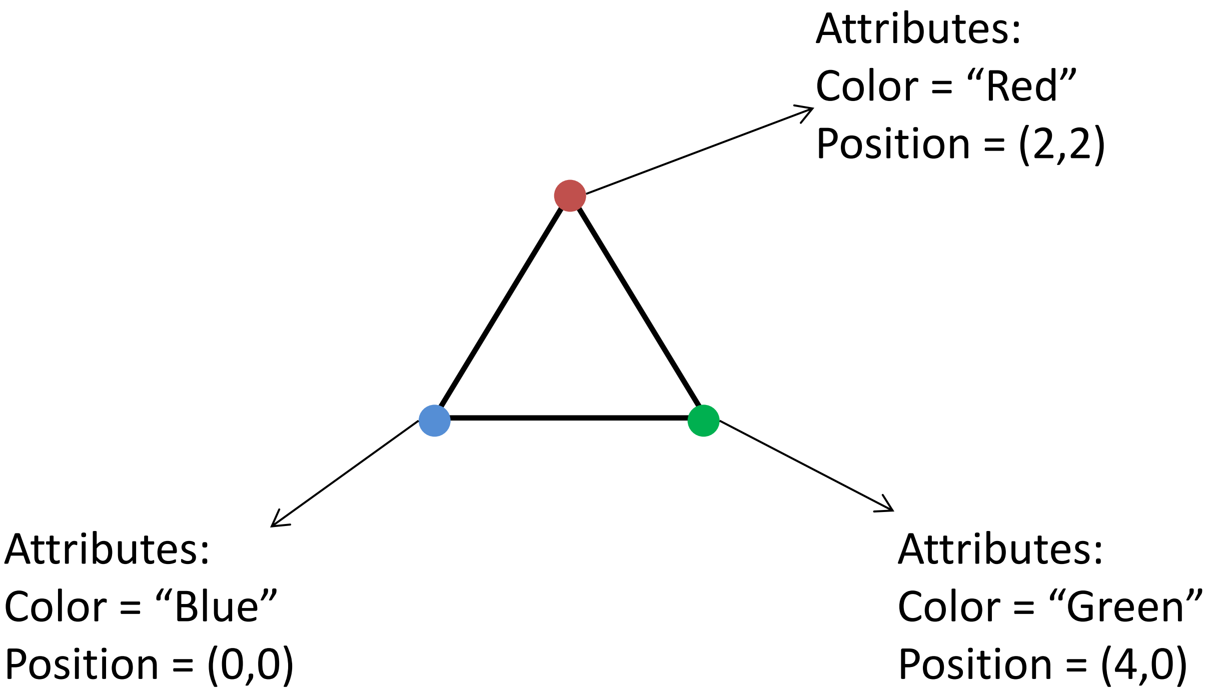

Figure 1: Example of Attributted Graph

Background

- Attributed graph Matching

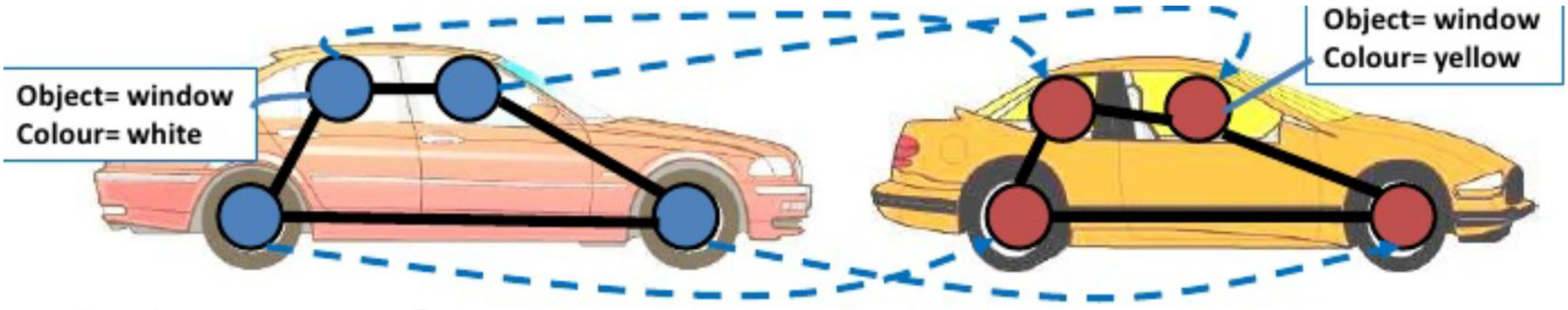

Figure 2: Example of graph matching[1]. Both the structure of the car(denoted by the structure of nodes) and the attributes are matched.

Problem Definition

- Given two attributed graphs G and G', for any connected query subgraph g of G, find top-k matching **connected** subgraphs in G'.

- Real-life Applications
  - Computer Vision
    - 2D & 3D Image analysis
    - Object Recognition
  - Graph Database Indexing and Retrieval
    - Biometric Indentification
    - Recommender System
  - NLP
    - Document matching
  - Many others

Why learning?

- NP problem - Computationally expensive, requires approximation
- Common Mathematical approach[2]

  Step 1: Compute pairwise distance(similarity) between nodes in both graph to give a distance matrix. The distance is computed using a predefined measure.

  Step 2: Match nodes using approximation algorithms
- Shortcoming of the approach above:

  - time & space complexity

  - handcrafted distance measure

Existing Learning Approaches

- Graph Neural Network(GNN)

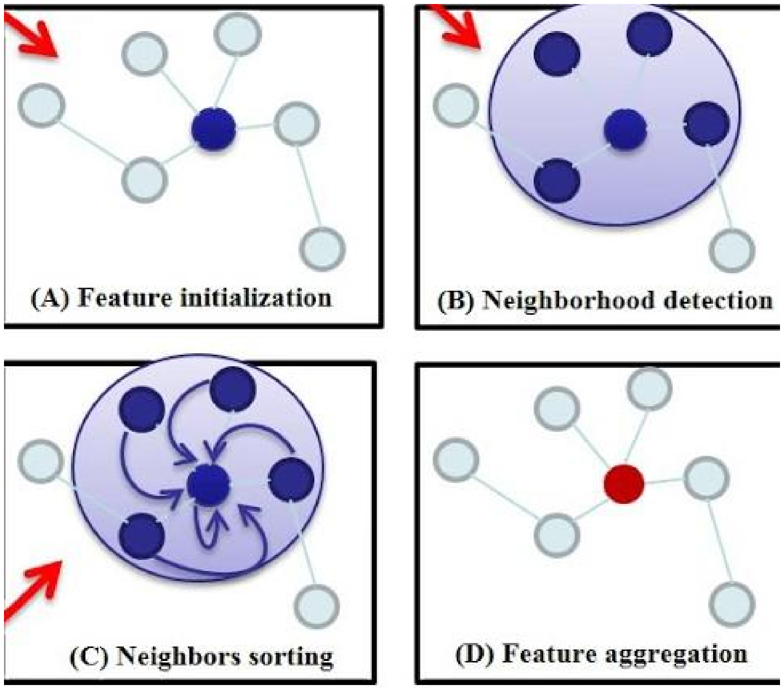

Figure 3: An illustration of 1-degree (hop) graph neural network[3]. The network works based on the principle of message passing.

Existing Learning Approaches

- GNN for graph matching

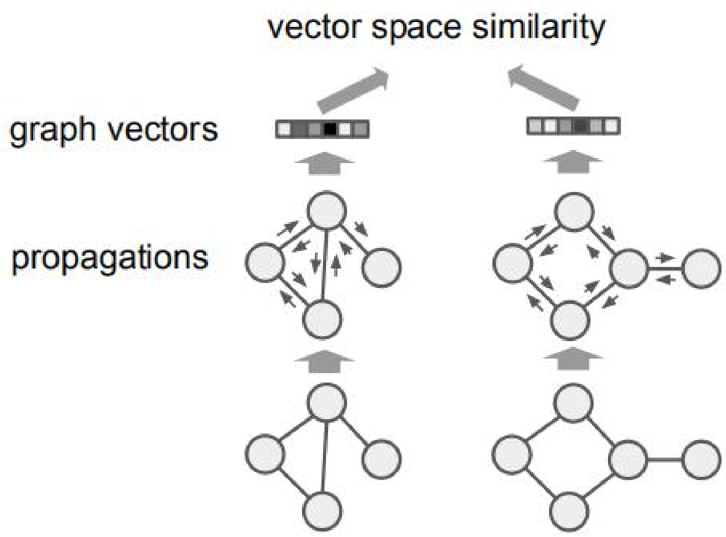

Figure 4: Graph matching using GNN[4]. GNN is used to process each graph to generate a graph level embedding, then the two embeddings are compared to compute the similarity.

Existing Learning Approaches

- Graph Matching network

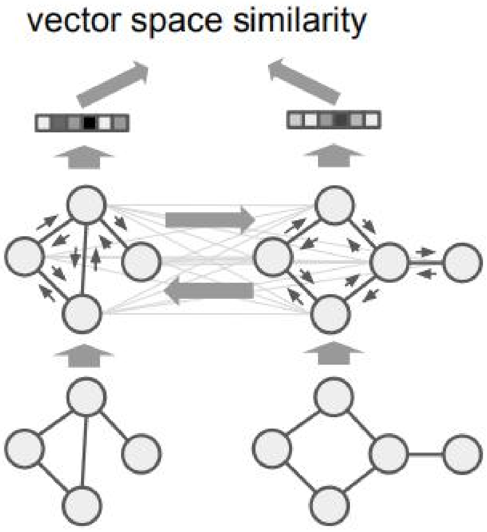

Figure 5: Graph matching network[4]. Uses attention mechanism to learn cross-graph correlation when training.

Existing Approaches - Problems

- Similarity based on graph-level representations
- Does not specify which part matches and on what basis do the two graphs match
- Generally focused on matching the attributes
- Capture more structural information with deeper GNNs, however, exponentially more computationally expensive
- Does not actively search for matching subgraphs, only compute the similarity of given graphs
- Searching for nodes based on closest embeddings often gives unconnected graphs

My Approach

- Two parts in the model
  - Embedding part
  - Matching part

- General idea: first generate node embeddings, then learn to matching the subgraphs

Synthetic Dataset

- Randomly generate 150 pairs of matching subgraphs of size 5 with attributes Color $\in$ {R, G, B}
- Matching based on attributes/structure or both
- Aggregate the first subgraph of each pair to form G
- Aggregate the second subgraph of each pair to form G'
- 2:1:1 for training data, validation data and testing data

Embedding part

- Model

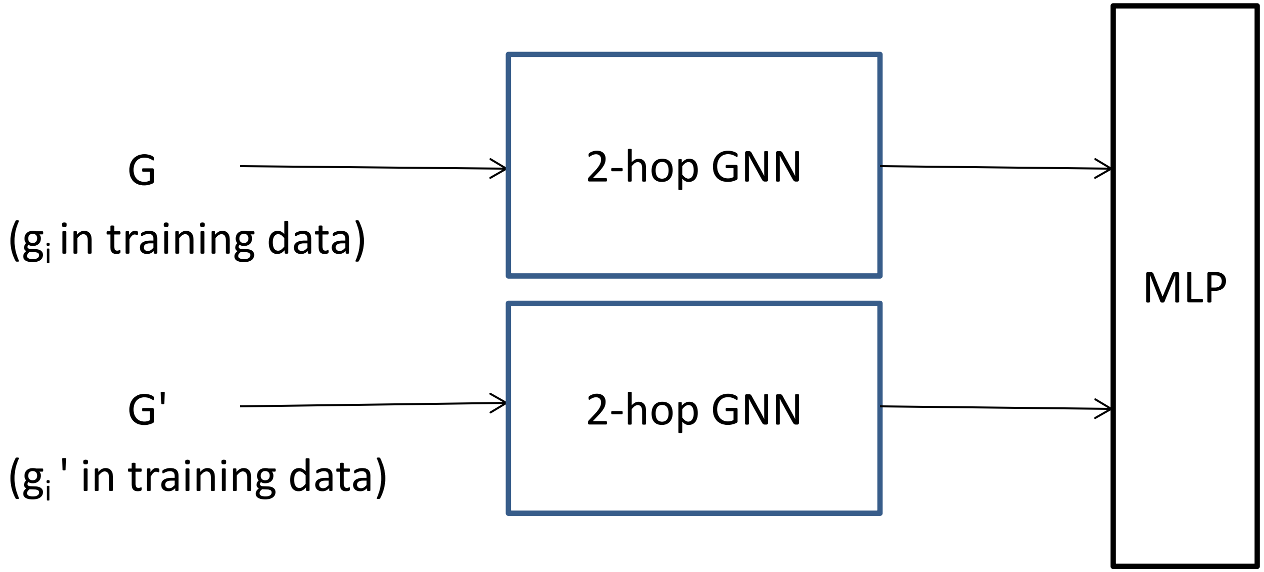

Figure 6: I use separate 2-hop GNNs for each graph. Matching subgraphs are passed into the model one pair at one time.

Embedding part

- Result: vector representation of each node in G and G'

- Loss:

  Embedding of a subgraph = concatenation of constituent node embeddings
  L_E = Euclidean distance between the embedding of each training pair

Matching part

- High-degree GNN can capture more structural information, but
    - computationally expensive
    - structural information not clearly shown in the result

- My approach:
    - inspired by NLP methods
    - Nodes in the same graph are related
    - Treat graphs as sequence of nodes
    - Use LSTM to learn about the correlation in the sequence

Matching part

- Bi-directional LSTM(BiLSTM)

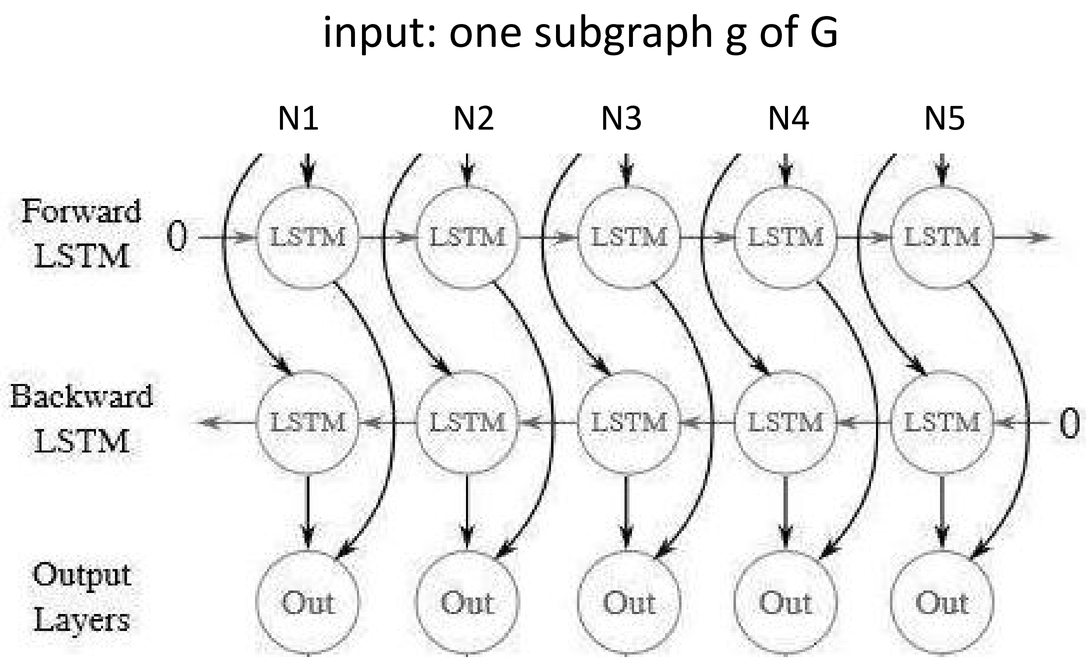

Figure 7: 1 layer Bi-directional LSTM used in the model.[5] The input is one subgraph in each pair.

Matching part

- LSTM: capable of catching long-short term dependency between nodes, may be capable to capture all levels of relations between nodes.

- How to sequence the nodes?
  - random sequence

Nodes before may also have dependencies on the nodes after. Bi-directional LSTM allows learning the connections in both directions

Matching part

- Result: vector representation of nodes, with structure information of the subgraphs considered. To get the matching subgraph g'' for g, find the closest node in G' for each node in g with the shortest euclidean distance.

- Matching Loss:
  For each training pair g in G and g' in G',
  L_M = euclidean distance between g'' and g'

- Total loss:
  $L_E + \gamma L_M$ where γ is a parameter to be optimized via validation

Experimental Result

- Based on the synthetic dataset with subgraphs of 5 nodes
- ~ 20 % accuracy rate(g'' = g')
- ~ 20 % g'' found connected
- > 40 % all nodes in g' within a 2 hop neighbourhoods of g''
- Numbers are not good apparently
- relatively unresearched field, hard to find baselines

Future Work

- Try using GMN for the embedding part
- Try further imbedding the nodes into Wasserstein space
- Fine-tune the model to improve the result
- Test on complex real-world datasets
- Find baselines to compare with

Thank you!

Australian National University

References

[1]: Image from: https://www.slideshare.net/mobile/raul_A/attributed-graph-matching-of-planar-graphs

[2]: D.H.Kim,I.D.Yun and S.U.Lee,A New Attributed Relational Graph Matching Algorithm Using the Nested Structure of Earth Mover's Distance, 17th International Conference on Pattern Recognition, p. 48-51, 2004.

[3]. Image from: Jun Wu, Jingrui He, Jiejun Xu, https://www.kdd.org/kdd2019/

[4]. Image from: Y. Li et.al. Graph Matching Networks for Learning the Similarity of Graph Structured Objects. ICLR 2019.

[5]. Image from: https://www.sohu.com/a/283978749_717210