Zero-shot Node Classification
with Decomposed Graph Prototype Network

Zheng Wang, Jialong Wang, Yuchen Guo, Zhiguo Gong

KDD 2021

https://github.com/zhengwang100/dgpn
Outline

• Motivation and Problem
  • Related work (Zero-shot & Node classification)
  • The problem of Zero-shot Node Classification (ZNC)

• Our solution
  • Step I: Acquiring High-Quality CSDs
  • Step II: Designing well-generalized graph-based learning models

• Experiments
Related Works

• Zero-shot learning

Classify the samples belonging to the classes that have no labeled data. Most ZSL methods are based on the external description and human-made attributes. Limited to computer vision or natural language processing.

• Graph Node Classification

The method of processing the graph is divided into the early shallow method and the recent deep Graph neural network method. Nevertheless, existing methods generally all assume that every class in the graph has some labeled nodes.
Traditional Node Classification

Labeled nodes (for train):
Class 1  Class2  Class 3

We have input with graph and corresponding labeled nodes for every class, and our goal is to predict labels on the unlabeled nodes.
Zero-shot Node Classification (ZNC)

(a) Input: graph and labels

(b) Output: predict labels

Labeled nodes (for train):
Class 1 □ Class 2 ○ Class 3 ◦ (unseen)

Although class 3 has no labeled samples for training (i.e., the zero-shot setting), we still want to “find” those nodes belonging to this class.
Outline

• Motivation and Problem
  • Background or motivation
  • The problem of Zero-shot Node Classification (ZNC)

• Our solution
  • **Step I: Acquiring High-Quality CSDs**
  • Step II: Designing well-generalized graph-based learning models

• Experiments
Step I: Acquiring High-Quality CSDs

• Acquiring two kinds (candidates) of CSDs from Wikipedia

<table>
<thead>
<tr>
<th>Class name</th>
<th>Class description (wiki page)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>In computing, a database is an organized collection of data stored and accessed electronically from a computer system. Where…….</td>
</tr>
<tr>
<td>ML</td>
<td>Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data....</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial intelligence (AI) is intelligence demonstrated by machines, unlike the natural intelligence displayed by humans and…….</td>
</tr>
</tbody>
</table>

Word2Vec model: Bert

Label-CSDs

Text-CSDs
Step I: Acquiring High-Quality CSDs

• Evaluate the quality of the automatically generated CSDs

Empirical probability (generated from class center representations)

\[ Pr(c_j|c_i) = \frac{\exp(o_i^T \cdot o_j)}{\sum_{t, t \neq i} \exp(o_i^T \cdot o_t)} \]

Probability (generated from our CSDs' vectors)

\[ \hat{Pr}(c_j|c_i) = \frac{\exp(s_i^T \cdot s_j)}{\sum_{t, t \neq i} \exp(s_i^T \cdot s_t)} \]

Calculate the distance:

\[ \frac{1}{|C|} \sum_{c_i \in C} \text{dis}(Pr(\cdot|c_i), \hat{Pr}(\cdot|c_i)) \]
Step I: Acquiring High-Quality CSDs

- CSDs’ Evaluation Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CSDs Type</th>
<th>KL Divergence</th>
<th>Cosine Similarity</th>
<th>Euclidean Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>LABEL-CSDs</td>
<td>0.0154</td>
<td>0.9978</td>
<td>0.1787</td>
</tr>
<tr>
<td></td>
<td>TEXT-CSDs</td>
<td><strong>0.0109</strong></td>
<td><strong>0.9985</strong></td>
<td><strong>0.1552</strong></td>
</tr>
<tr>
<td>Citeseer</td>
<td>LABEL-CSDs</td>
<td>0.0120</td>
<td>0.9980</td>
<td>0.1620</td>
</tr>
<tr>
<td></td>
<td>TEXT-CSDs</td>
<td><strong>0.0077</strong></td>
<td><strong>0.9987</strong></td>
<td><strong>0.1328</strong></td>
</tr>
<tr>
<td>C-M10M</td>
<td>LABEL-CSDs</td>
<td>0.0062</td>
<td>0.9990</td>
<td>0.1175</td>
</tr>
<tr>
<td></td>
<td>TEXT-CSDs</td>
<td><strong>0.0026</strong></td>
<td><strong>0.9996</strong></td>
<td><strong>0.0735</strong></td>
</tr>
</tbody>
</table>

Here, ‘↓’ indicates the lower the better, whereas ‘↑’ indicates the higher the better. Compared with the Label-CSDs, Text-CSDs always perform better.
Outline

• Motivation and Problem
  • Background or motivation
  • The problem of Zero-shot Node Classification (ZNC)

• Our solution
  • Step I: Acquiring High-Quality CSDs
  • Step II: Designing well-generalized graph-based learning models

• Experiments
Step II: Designing well-generalized graph-based learning models

• Traditional GCNs

Traditional GCN processes the input serially.
Step II: Designing well-generalized graph-based learning models

• GCNs Decomposition

We Decompose the two layers GCN into three parts and use them for subsequent locality and compositionality.
Step II: Designing well-generalized graph-based learning models

- Locality and Compositionality

We take the feature of each order neighbor of the node as the local part and use their combination as the global compositional part.
Decomposed Graph Prototype Network

This joint learning not only enhances the locality of the node representation that is critical for zero-shot generalization, but also guarantees the discriminability of the global compositional representation for the final node classification.

Figure 2: The architecture of Decomposed Graph Prototype Network (DGPN).
Outline

• Motivation and Problem
  • Related work
  • The problem of Zero-shot Node Classification (ZNC)

• Our solution
  • Step I: Acquiring High-Quality CSDs
  • Step II: Designing well-generalized graph-based learning models

• Experiments
Experimental Settings

• Datasets with seen/unseen splits:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>Edges</th>
<th>Features</th>
<th>Classes</th>
<th>Class Split I [Train/Val/Test]</th>
<th>Class Split II [Train/Val/Test]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>5,429</td>
<td>1,433</td>
<td>7</td>
<td>[3/0/4]</td>
<td>[2/2/3]</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3,327</td>
<td>4,732</td>
<td>3,703</td>
<td>6</td>
<td>[2/0/4]</td>
<td>[2/2/2]</td>
</tr>
<tr>
<td>C-M10M</td>
<td>4,464</td>
<td>5,804</td>
<td>128</td>
<td>6</td>
<td>[3/0/3]</td>
<td>[2/2/2]</td>
</tr>
</tbody>
</table>

• Baselines
  • DAP & DAP(CNN)
  • ESZSL
  • ZS-GCN & ZS-GCN(CNN)
  • WDVS\textsc{c}
  • Hyperbolic-ZSL
  • RandomGuess
Compare with Baselines

- Our method DGPN always outperforms all baselines by a significant margin, gives 11.94% and 16.55% improvements.
- Baselines still outperform Random Guess.
- Simple classical methods (like DAP and ESZSL) generally get better results than those recently proposed complex ones (like ZS-GCN and Hyperbolic-ZSL)

### Table 3: Zero-shot node classification accuracy (%).

<table>
<thead>
<tr>
<th>Class Split</th>
<th>Cora</th>
<th>Citeseer</th>
<th>C-M10M</th>
</tr>
</thead>
<tbody>
<tr>
<td>RandomGuess</td>
<td>25.35±1.28</td>
<td>24.86±1.63</td>
<td>33.21±1.08</td>
</tr>
<tr>
<td>DAP</td>
<td>26.56±0.37</td>
<td>34.19±0.97</td>
<td>39.20±0.54</td>
</tr>
<tr>
<td>DAP(CNN)</td>
<td>27.80±0.67</td>
<td>30.45±0.93</td>
<td>32.97±0.71</td>
</tr>
<tr>
<td>ESZSL</td>
<td>27.35±0.00</td>
<td>30.32±0.00</td>
<td>37.00±0.00</td>
</tr>
<tr>
<td>ZS-GCN</td>
<td>25.73±0.46</td>
<td>26.62±0.20</td>
<td>37.89±1.15</td>
</tr>
<tr>
<td>ZS-GCN(CNN)</td>
<td>16.01±3.27</td>
<td>21.18±1.58</td>
<td>36.44±0.97</td>
</tr>
<tr>
<td>WDVSIC</td>
<td>30.62±0.38</td>
<td>23.46±0.11</td>
<td>38.12±0.35</td>
</tr>
<tr>
<td>Hyperbolic-ZSL</td>
<td>25.36±0.41</td>
<td>34.18±0.88</td>
<td>35.80±2.25</td>
</tr>
<tr>
<td><strong>DGPN (ours)</strong></td>
<td><strong>34.15±0.28</strong></td>
<td><strong>38.16±0.11</strong></td>
<td><strong>44.17±0.21</strong></td>
</tr>
<tr>
<td>Improve↑</td>
<td>+11.53%</td>
<td>+11.61%</td>
<td>+12.68%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Split II</th>
<th>Cora</th>
<th>Citeseer</th>
<th>C-M10M</th>
</tr>
</thead>
<tbody>
<tr>
<td>RandomGuess</td>
<td>32.69±1.48</td>
<td>50.48±1.70</td>
<td>49.73±1.56</td>
</tr>
<tr>
<td>DAP</td>
<td>30.22±1.21</td>
<td>53.30±0.22</td>
<td>46.79±4.16</td>
</tr>
<tr>
<td>DAP(CNN)</td>
<td>29.83±1.23</td>
<td>50.07±1.70</td>
<td>46.29±3.36</td>
</tr>
<tr>
<td>ESZSL</td>
<td>38.82±0.00</td>
<td>55.32±0.00</td>
<td>56.57±0.00</td>
</tr>
<tr>
<td>ZS-GCN</td>
<td>29.53±0.91</td>
<td>52.22±1.21</td>
<td>55.28±0.41</td>
</tr>
<tr>
<td>ZS-GCN(CNN)</td>
<td>33.20±0.32</td>
<td>49.27±0.73</td>
<td>51.37±1.27</td>
</tr>
<tr>
<td>WDVSIC</td>
<td>34.13±0.67</td>
<td>52.70±0.68</td>
<td>46.26±2.58</td>
</tr>
<tr>
<td>Hyperbolic-ZSL</td>
<td>37.02±0.28</td>
<td>46.27±0.39</td>
<td>55.07±0.77</td>
</tr>
<tr>
<td><strong>DGPN (ours)</strong></td>
<td><strong>48.40±0.31</strong></td>
<td><strong>62.40±0.32</strong></td>
<td><strong>63.46±0.42</strong></td>
</tr>
<tr>
<td>Improve↑</td>
<td>+24.68%</td>
<td>+12.80%</td>
<td>+12.18%</td>
</tr>
</tbody>
</table>

The best method is bolded, and the second-best is underlined.
Compare with Different CSDs

The performance of all methods (including ours) declines significantly, compared to those results in Table 3 where TEXT-CSDs are used.

This indicates node attributes contain richer and useful information than graph structure information.
Compare Node Attributes with Adjacency Matrix

Table 5: Zero-shot node classification accuracy (%) using the graph adjacency information as node attribute information.

<table>
<thead>
<tr>
<th>Class Split I</th>
<th>TEXT-CSDs</th>
<th></th>
<th></th>
<th>LABEL-CSDs</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cora</td>
<td>Citeseer</td>
<td>C-M10M</td>
<td>Cora</td>
<td>Citeseer</td>
<td>C-M10M</td>
</tr>
<tr>
<td>DAP</td>
<td>30.76</td>
<td>33.98</td>
<td>36.76</td>
<td>28.57</td>
<td>19.38</td>
<td>30.91</td>
</tr>
<tr>
<td>ESZSL</td>
<td>24.98</td>
<td>33.20</td>
<td>36.34</td>
<td>30.22</td>
<td>30.05</td>
<td>34.61</td>
</tr>
<tr>
<td>ZS-GCN</td>
<td>28.43</td>
<td>33.35</td>
<td>36.87</td>
<td>23.26</td>
<td>30.26</td>
<td>33.90</td>
</tr>
<tr>
<td>WDVSc</td>
<td>18.98</td>
<td>28.77</td>
<td>33.84</td>
<td>29.73</td>
<td>23.03</td>
<td>30.35</td>
</tr>
<tr>
<td>Hyperbolic-ZSL</td>
<td>19.96</td>
<td>12.16</td>
<td>35.80</td>
<td>28.53</td>
<td>12.45</td>
<td>30.82</td>
</tr>
<tr>
<td>DGPN (ours)</td>
<td>32.96</td>
<td>38.03</td>
<td>40.01</td>
<td>31.28</td>
<td>31.85</td>
<td>35.75</td>
</tr>
</tbody>
</table>

The results which are better than those of RandomGuess are typeset in blue.

This indicates node attributes contain richer and useful information than graph structure information.
Model Ablation

- ProNet: the variant that replaces the decomposed GCNs part with fully-connect layer
- ProNet+GCN: the variant that removes the local loss part in our method.
- ProNet+GCN+LL: full model

Both two parts (the decomposed GCNs part and local loss part) contribute to the final performance, which evidently demonstrates their effectiveness.

Figure 3: Model ablation under Class Split I.
Hyper-Parameters Searching

- Show the usefulness of the graph structure information and the lazy random walk strategy.

(a) Cora  (b) Citeseer  (c) C-M10M

Figure 4: Effects of $K$ and $\beta$ in our method under Class Split I. Grid numbers denote the classification accuracy (%). Color indicates the performance (the deeper the better).
Some interesting findings

1. The quality of CSDs is the key to the ZNC problem; we can rank the importance as: CSDs ≫ node attributes>graph structure.

2. Comparing to “RandomGuess”, we can rank the general performance of those two CSDs as: TEXT-CSDs ≫ LABEL-CSDs ≥ RandomGuess.

3. With high-quality CSDs, graph structure information can be very useful or even be comparable to node attributes.

4. Through subtly recasting the concepts, locality and compositionality can be well adapted to graph-structured data.
Summary

• Three main contributions:
  • Novel problem: Zero-shot Node Classification (ZNC)
  • Novel CSDs acquisition and evaluation strategy
  • Novel zero-shot method DGPN

• Code available at: https://github.com/zhengwang100/dgpn

• Other related topics
  • Node Classification: https://en.wikipedia.org/wiki/Collective_classification
  • Zero-shot Graph Embedding (ZGE):
    https://zhengwang100.github.io/project/zero_shot_graph_embedding.html
THANK YOU!

Zheng Wang       https://zhengwang100.github.io