Hybrid Attention-Based Prototypical Networks for Noisy Few-Shot Relation Classification

AAAI 2019
Motivation

• Relation Classification:
  – Finding the semantic relation between entity mentions in text

• Supervised Learning:
  – Manual labeling is time-consuming

• Distant Supervision:
  – Use the relation of two entities in a Knowledge Base as the semantic relation of two entity mentions in text
  – Introduce noise to labeling
  – Few instances for rare relations
Motivation

• Model RC as Few-shot Learning (FSL)
  – Few examples per relation

• Approaches for FSL:
  – Transfer learning: Use information of abundant labels for rare labels
  – Metric Learning: Learn distance distributions among labels
  – Meta Learning: Learn to learn
    • Prototypical networks
Motivation

• Prototypical networks are mainly used for CV

• Challenges for NLP:
  – More diversity
  – More noise

• This paper address:
  – RC with rare instances per class and noisy labels
  – Use prototypical network as a technique to model RC as FSL addressing diversity and noise in prototypical networks
Contributions

• Introducing Two levels of attention:
  – Feature level: Select most useful features for computing prototypes
  – Instance level: Selects most useful instances in support set based on the given query

• Analyzing robustness to noise:
  – Compared to vanilla prototypical network their approach is more robust to noise in labels
Model

\[ d_1 = \| z_1 - c_1 \|^2 \]

\[ d_2 = \| z_2 - c_2 \|^2 \]

\[ d_N = \| z_N - c_N \|^2 \]
Model

• Inputs to model:

\[ S = \{(x_1^1, h_1^1, t_1^1, r_1), \ldots, (x_1^{n_1}, h_1^{n_1}, t_1^{n_1}, r_1), \ldots, (x_m^1, h_m^1, t_m^1, r_m), \ldots, (x_m^{n_m}, h_m^{n_m}, t_m^{n_m}, r_m)\}, \]

\[ r_1, r_2, \ldots, r_m \in \mathcal{R}, \]

• Use CNN to encode the sentences:

• Inputs to CNN:
  – GloVe embedding
  – Position Embedding
Prototypical Network

- Find prototypes:

\[ c_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_i^j, \]

- Classify query:

\[ p_\phi(y = r_i | x) = \frac{\exp \left( -d(f_\phi(x), c_i) \right)}{\sum_{j=1}^{|\mathcal{R}|} \exp \left( -d(f_\phi(x), c_j) \right)}, \]
Instance Level Attention

• Compute attention weight for each instance in support set based on the query instance:

\[
\alpha_j = \frac{\exp(e_j)}{\sum_{k=1}^{n_i} \exp(e_k)},
\]

\[
e_j = \text{sum}\left\{\sigma(g(x_i^j) \odot g(x))\right\},
\]

• Find Prototypes:

\[
c_i = \sum_{j=1}^{n_i} \alpha_j x_i^j.
\]
Feature Level Attention

• Find attention weight per feature of prototypes:
  – 2D CNN

• Find distance of query to prototypes:

\[ d(s_1, s_2) = z_i \cdot (s_1 - s_2)^2 \]
Experiments

• FewRel:
  – Training: 64 relations
  – Dev: 16 relations
  – Test: 20 relations
  – 700 instances per relation

• Noise Level:
  – Randomly change relation labels to wrong labels
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional Window Size $m$</td>
<td>3</td>
</tr>
<tr>
<td>Word Embedding Dimension $d_w$</td>
<td>50</td>
</tr>
<tr>
<td>Position Embedding Dimension $d_p$</td>
<td>5</td>
</tr>
<tr>
<td>Hidden Layer Dimension $d_h$</td>
<td>230</td>
</tr>
<tr>
<td>Batch Size</td>
<td>4</td>
</tr>
<tr>
<td>Training Classes for One Batch</td>
<td>20</td>
</tr>
<tr>
<td>Initial Learning Rate</td>
<td>0.1</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>$10^{-5}$</td>
</tr>
<tr>
<td>Learning Rate Decay $\gamma$</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Robustness to noise

<table>
<thead>
<tr>
<th>Noise Rate</th>
<th>Model</th>
<th>5 Way 5 Shot</th>
<th>5 Way 10 Shot</th>
<th>10 Way 5 Shot</th>
<th>10 Way 10 Shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>Proto</td>
<td>89.05 ± 0.09</td>
<td>90.79 ± 0.08</td>
<td>81.46 ± 0.13</td>
<td>84.01 ± 0.13</td>
</tr>
<tr>
<td></td>
<td>Proto-HATT</td>
<td>90.12 ± 0.04</td>
<td>92.06 ± 0.06</td>
<td>83.05 ± 0.05</td>
<td>85.97 ± 0.08</td>
</tr>
<tr>
<td>10%</td>
<td>Proto</td>
<td>87.63 ± 0.10</td>
<td>90.15 ± 0.08</td>
<td>79.39 ± 0.14</td>
<td>83.05 ± 0.12</td>
</tr>
<tr>
<td></td>
<td>Proto-HATT</td>
<td>88.74 ± 0.06</td>
<td>91.45 ± 0.05</td>
<td>81.09 ± 0.08</td>
<td>85.08 ± 0.07</td>
</tr>
<tr>
<td>30%</td>
<td>Proto</td>
<td>82.45 ± 0.09</td>
<td>87.64 ± 0.07</td>
<td>72.43 ± 0.12</td>
<td>79.31 ± 0.11</td>
</tr>
<tr>
<td></td>
<td>Proto-HATT</td>
<td>84.71 ± 0.07</td>
<td>89.59 ± 0.05</td>
<td>75.68 ± 0.11</td>
<td>82.43 ± 0.07</td>
</tr>
<tr>
<td>50%</td>
<td>Proto</td>
<td>72.91 ± 0.15</td>
<td>81.71 ± 0.10</td>
<td>61.11 ± 0.17</td>
<td>71.29 ± 0.14</td>
</tr>
<tr>
<td></td>
<td>Proto-HATT</td>
<td>76.57 ± 0.07</td>
<td>85.17 ± 0.09</td>
<td>65.97 ± 0.11</td>
<td>76.42 ± 0.13</td>
</tr>
</tbody>
</table>
Comparing to Baselines

<table>
<thead>
<tr>
<th>Model</th>
<th>5 Way 5 Shot</th>
<th>10 Way 5 Shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune*</td>
<td>68.66 ± 0.41</td>
<td>55.04 ± 0.31</td>
</tr>
<tr>
<td>kNN*</td>
<td>68.77 ± 0.41</td>
<td>55.87 ± 0.31</td>
</tr>
<tr>
<td>Meta Network*</td>
<td>80.57 ± 0.48</td>
<td>69.23 ± 0.52</td>
</tr>
<tr>
<td>GNN*</td>
<td>81.28 ± 0.62</td>
<td>64.02 ± 0.77</td>
</tr>
<tr>
<td>SNAIL*</td>
<td>79.40 ± 0.22</td>
<td>68.33 ± 0.25</td>
</tr>
<tr>
<td>Proto*</td>
<td>84.79 ± 0.16</td>
<td>75.55 ± 0.19</td>
</tr>
<tr>
<td>Proto</td>
<td>89.05 ± 0.09</td>
<td>81.46 ± 0.13</td>
</tr>
<tr>
<td>Proto-IATT</td>
<td>89.63 ± 0.08</td>
<td>82.16 ± 0.13</td>
</tr>
<tr>
<td>Proto-FATT</td>
<td>89.70 ± 0.03</td>
<td>82.45 ± 0.05</td>
</tr>
<tr>
<td>Proto-HATT</td>
<td>90.12 ± 0.04</td>
<td>83.05 ± 0.05</td>
</tr>
</tbody>
</table>
Convergence Speed
Representation Visualization

- **Feature Attention:**
  - (a) Features with lower scores.
  - (b) Features with higher scores.

- **Sentence encoding with attention:**
  - (c) Emb trained without HATT.
  - (d) Emb trained with HATT.
Question?