I Incorporating Syntactic and Semantic Information in
Word Embeddings using Graph Convolutional Networks

Shikhar Vashishth1  Manik Bhandari1∗ Prateek Yadav2∗
Piyush Rai3  Chiranjib Bhattacharyya1  Partha Talukdar1

1Indian Institute of Science
2Microsoft Research, 3IIT Kanpur
{shikhar,manikb,chiru,ppt}@iisc.ac.in
tr-ptyad@microsoft.com, piyush@cse.iitk.ac.in

1 Hyperparameters

Our vocabulary consists of 150k most frequent words in Wikipedia corpus. Following (Mikolov et al., 2013a; Pennington et al., 2014), we have reported results for 300-dimensional embeddings on intrinsic tasks. However, for extrinsic tasks results are reported for 256-dimensional embeddings since pre-trained ELMo model is available for only \{128, 256, 512, 1024\} sizes. For training baselines, we use the code provided by the authors with the default hyperparameters. For training SynGCN and SemGCN, we use Adam optimizer (Kingma and Ba, 2014) with learning rate of 0.001. Following (Mikolov et al., 2013b), subsampling is used with threshold parameter $t = 10^{-4}$. The target and neighborhood embeddings are initialized randomly using Xavier initialization (Glorot and Bengio, 2010). In GCN, number of layers $(k)$ is taken as 1 and ReLU is used as the activation function.

2 Evaluating performance with same semantic information

In this section, we present the complete set of results for comparison of SemGCN against other methods when provided with the same semantic information (synonyms from PPDB). Similar to Section 9.3, in Table 1 and 2 (Please look at table on the next page), we present the comparison on intrinsic tasks and extrinsic tasks. Please refer Section 9.4 for more details.

<table>
<thead>
<tr>
<th>Method</th>
<th>POS</th>
<th>SQuAD</th>
<th>NER</th>
<th>Coref</th>
</tr>
</thead>
<tbody>
<tr>
<td>SynGCN</td>
<td>95.4±0.1</td>
<td>79.6±0.2</td>
<td>89.5±0.1</td>
<td>65.8±0.1</td>
</tr>
<tr>
<td>Retro-fit (X,1)</td>
<td>94.8±0.1</td>
<td>79.6±0.1</td>
<td>88.8±0.1</td>
<td>66.0±0.2</td>
</tr>
<tr>
<td>Counter-fit (X,1)</td>
<td>94.7±0.2</td>
<td>79.9±0.1</td>
<td>88.2±0.3</td>
<td>65.5±0.1</td>
</tr>
<tr>
<td>JointReps (X,1)</td>
<td>95.4±0.2</td>
<td>79.4±0.3</td>
<td>89.1±0.3</td>
<td>65.6±0.0</td>
</tr>
<tr>
<td>SemGCN (X,1)</td>
<td>95.5±0.1</td>
<td>80.4±0.2</td>
<td>89.7±0.2</td>
<td>66.1±0.2</td>
</tr>
</tbody>
</table>

Table 1: Comparison of different methods when provided with same semantic information (synonym) for fine tuning SynGCN embeddings. Please refer Section 9.4 of paper for details.

References


∗Contributed equally to the work.
Table 2: Evaluation of different methods for incorporating same semantic information (synonym) initialized using various pre-trained embeddings (X). M(X, R) denotes the fine-tuned embeddings using method M taking X as initialization embeddings. R denotes the number of semantic relations used as defined in Section 9.3. SemGCN outperforms other methods in 11 of the 15 settings. SemGCN with SynGCN gives the best performance across all tasks (highlighted using ·). Please refer Section 9.4 for details.