Structure Representation Learning by Jointly Learning to Pool and Represent

Anonymous ACL submission

Abstract

Structure representation learning is a task to provide an overall representation for a given structure (e.g., sequential text, non-sequential graph). This representation characterizes the property of that structure. Previous methods decompose the task into an element representation learning phase and a pooling phase to aggregate element representations. Their pooling phase only considers the final representation of each element without considering the relationship between these elements that are used only to construct representations of elements. In this paper, we conjecture that classification performance suffers from the lack of relation exploitation while pooling and propose the Self-Attention Pooling to dynamically provide centrality scores for pooling based on the self-attention scores from the element representation learning. Simply applying Self-Attention Pooling improves model performance on 3 sentence classification tasks (↑2.9) and 5 graph classification tasks (↑2.1) on average.

1 Introduction

We use structure representation learning to denote learning a summary representation for a natural structure like a sequence or a non-sequential graph. For example, we can predict the property of a sentence that consists of a sequence of words, with its representations (Wang et al., 2019). In addition to the sequence, the structure can also be a non-sequential graph that is composed of nodes (Reimer and Hahn, 1988; Yao et al., 2018). This task usually follows a pipeline that first learns the representation of the elements and then pools the representations of these elements based on their final representations (Kim, 2014). The pooling layer first predicts the centrality of each element and then either weighted-sum element representations according to their centrality or selects element representations with significant centrality.

Most recently proposed models follow an element representation-based pooling method. For example, in sentence classification, scoring is obtained through the attention of artificial [CLS] token to natural words (Radford et al., 2018; Devlin et al., 2019). In graph classification, to get the centrality of each node, we can exploit the static graph topology (Lee et al., 2019) in addition to the representation of the nodes (Gao and Ji, 2019). A potential issue with this element representation-based pooling method is that obtaining the structure representation by separately considering the representation of the elements does not exploit the relation between the elements. This issue makes the model overly dependent on the element representation to encode the relationship between them and sequentially learns the representations of elements and the pool operation. The relation between elements that help learn element representations can also help learn structures (Voita et al., 2019; Jawahar et al., 2019).

To address this issue, we propose jointly learning to represent the elements and pool the elements by sharing the self-attention modules from element representation learning. Simply applying Self-Attention Pooling to dynamically provide centrality scores for pooling based on the self-attention scores from the element representation learning improves model performance on 3 sentence classification tasks (↑2.9) and 5 graph classification tasks (↑2.1) on average.

1 We compare with CLS Pooling from BERT for sequence pooling and SAGPooling for non-sequence pooling.
representation learning. Specifically, we utilize the accumulation of all the attention the element receives to indicate its centrality. We also design specific applications on sentence classification with the BERT model and graph classification with the Graph Attention model (refer to Fig. 1). In sentence classification, we extend BERT finetuning so that the relationship between natural words can be applied to pooling instead of just using the relationship between the artificial [CLS] token and natural tokens. We extend graph representation learning on graph classification by exploiting automatically learned node relations instead of just using static graph topology.

2 Related Work

Pooling plays an important role in both sequential (Socher et al., 2011; Chen et al., 2015; Safari et al., 2020) and non-sequential structure representation learning (Lee et al., 2019). Most methods separately learn element representations and pooling and do not exploit the relation between elements (Kim, 2014; Ying et al., 2018).

**Sequential Pooling** Sequential pooling objects to obtain a representation of a piece of text. Previous methods usually perform an average or maximum operation on every position (Kim, 2014; Ma et al., 2019; Song et al., 2020), or sum the representations of positions with their feature weights (Yang et al., 2016; Wu et al., 2020). The powerful pretrained language model BERT (Devlin et al., 2019) directly applies CLS pooling with an artificial [CLS] token (Devlin et al., 2019), which aggregates information by attending representations of other positions. However, these methods neglect the relation between all positions, and the CLS pooling is only learned in the finetuning phase of BERT. Recent studies find that attention weights can indicate keywords, but they do not study its effectiveness in pooling and downstream tasks like sentence classification (Clark et al., 2019; Ding and Luo, 2021).

**Non-sequential Pooling** Non-sequential pooling aims to extract the overall representation of a non-sequence. The graph is a well-studied non-sequence. Previous research mainly disassembled it into two parts: node representation learning and graph pooling. Traditional graph pooling takes the node representation into account (Gao and Ji, 2019), and recent methods propose to utilize graph topology to model the node relation (Lee et al., 2019; Murphy et al., 2019; Yuan and Ji, 2020), but the relationship automatically learned in node representation learning is still not considered.

3 Proposal

3.1 Self-Attention Pooling

To model dynamic relation in the structure representation, we propose Self-Attention Pooling. It links the construction of element representation and structure representation (i.e., pooling). For learning element representation, self-attention module updates the representation of each element. For pooling, weights are centrality scores that reflect the importance of each element in a structure. Inspired by PageRank (see Section 5), we define the centrality score of an element by its overall attention scores\(^2\) received from other elements. While for learning structure representation, the centrality scores are ranked for top-k selection or weighted sum of the structure representation. We define \(X\) as the input structure, \(N\) as the number of elements and \(X_j^{(m)}\) as the element at layer \(m\). Then, the element representation \(X_j^{(m)}\) and the centrality \(S_j^{(m)}\) can be formulated as follows:

\[
X_j^{(m)} = \sum_{i=1}^{N} \alpha_{i,j} X_i^{(m-1)} \quad (1)
\]

\[
S_j^{(m)} = \sum_{i=1}^{N} \alpha_{i,j} \quad (2)
\]

where \(\alpha_{i,j}\) is the self-attention score from element \(i\) to \(j\), and \(\sum_{j=1}^{N} \alpha_{i,j} = 1\). For conciseness, we omit the description of the non-aggregation neural network and focus on the element aggregation.

3.2 Self-Attention based Sequence Pooling

As illustrated in Fig. 1 (a), for sequential structure, our objective is to learn sequence representation for downstream tasks like sentence classification. Here the element representation can be seen as position representation, e.g., word-level or subword-level representation. For sequence pooling, we study the powerful BERT model and compare its pooling methods. Therefore, we pool the representations from the last hidden layer of the BERT encoder.

We compare with the CLS pooling, mean-pooling, and max-pooling. Although been default in BERT pooling, CLS pooling merely takes \(X_0^{(m)} = \sum_{j=0}^{N} \alpha_{0,j} X_j^{(m-1)}\) as the sequence representation. In contrast, BERT is pretrained with all the positions rather than only the CLS position.

\(^2\)For pooling, we use the averaged self-attention scores overheads.
As shown in Fig. 1 (b), for graph structure, nodes while calculating $\alpha_{i,j}$ from various layers. According to Eq. 1, we exploit the relation between all positions to obtain centrality scores for each position. The overall sequence representation is \[ \sum_{j=0}^{N} X_j \cdot S_j. \]

### 3.3 Self-Attention based Graph Pooling

As shown in Fig. 1 (b), for graph structure, nodes and graphs represent elements and structures respectively. Here $X_j$ (j=1,2,...,N) denote the feature of each node. We compare our method with two baselines for graph representation: gPool that considers only node features, formulated\(^3\) as $Z = X^{(l)}(i) \Theta^{(l)} / \|g^{(l)}\|$. SAGPool that considers both features of nodes and the overall graph topology, roughly\(^4\) described as $Z = \sigma(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}X\Theta)$. Different from previous work, Self-Attention Pooling exploits node relations from the graph attention mechanism (Veličković et al., 2018) (GAT) directly, which is also crucial for node representation. It is slightly different from Eq. 1 because $\alpha_{i,j}$ is only calculated among each node and its neighbors. In GAT, $e_{ij}$ is a logit calculated from concentrated element representation, $N(i)$ denotes node $i$ and its neighbours. The centrality scores are calculated as:

\[ \alpha_{i,j} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N(i)} \exp(e_{ik})} \quad (3) \]
\[ S_j = Z_j + \sum_{i \in N(j)} \alpha_{i,j} \quad (4) \]

Since the attention in graph is local, we propose iterative Self-Attention Pooling as:

\[ w_j = \sum_{i \in N(j)} \alpha_{i,j}, \alpha_i = w_i \cdot \alpha_i, S_j = Z_j + \sum_{i \in N(j)} \alpha_{i,j} \quad (5) \]

After getting the centrality score $S_j$ of each node in the current graph, we can mask out the nodes with low importance and retrain the others for further calculation.

### 4 Experiments

#### 4.1 Datasets

**Sequence Classification** In our experiments, we consider a single sentence or a sentence pair as a sequence. We use CoLA for single sentence classification, MRPC, and RTE for sentence-Pair classification. CoLA (Warstadt et al., 2018) is a dataset for grammatical acceptability, consisting of 10,657 sentences from 23 linguistics publications. MRPC (Dolan and Brockett, 2005) is a dataset for natural language inference. Given a premise and a hypothesis, models are expected to select the best answer between entailment, neutral, and contraction.

**Graph Classification** We experiment with five large graph datasets from the benchmark datasets (Kersting et al., 2016). D&D (Dobson and Doig, 2003; Shervashidze et al., 2011) and PROTEINS (Dobson and Doig, 2003; Borgwardt et al., 2005) are both protein datasets that are classified as enzymes or non-enzymes. Nodes represent the amino acids and two nodes are connected if they are less than 6 Angstroms apart. NCI1 and NCI109 are commonly used. FRANKENSTEIN (Orsini et al., 2015) is a set of molecular graphs (Costa and De Grave, 2010). Its label denotes whether a molecule is a mutagen or non-mutagen. D&D, PROTEINS, NCI, NCI109,
Table 2: Results on graph classification tasks. gPool gets pooling scores from features. SAGPool uses the graph topology. Self-Attention Pooling introduces learned node relations from node representation learning to pooling.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>D&amp;D</th>
<th>PROTEINS</th>
<th>NCI1</th>
<th>NCI09</th>
<th>FRANKENSTEIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>gPool (Gao and Ji, 2019)</td>
<td>73.74±0.45</td>
<td>72.80±0.17</td>
<td>70.04±0.44</td>
<td>70.10±1.23</td>
<td>75.97±0.53</td>
</tr>
<tr>
<td>SAGPool (Lee et al., 2019)</td>
<td>75.01±0.50</td>
<td>72.99±0.12</td>
<td>72.37±0.22</td>
<td>71.63±0.54</td>
<td>76.09±0.57</td>
</tr>
<tr>
<td>S.A. Pooling (Ours)</td>
<td>76.00±0.71</td>
<td><strong>74.12±0.40</strong></td>
<td>74.60±0.22</td>
<td>73.81±0.41</td>
<td>79.02±0.70</td>
</tr>
<tr>
<td>Iterative S.A. Pooling (Ours)</td>
<td><strong>76.23±0.13</strong></td>
<td>74.06±0.40</td>
<td><strong>74.70±0.25</strong></td>
<td><strong>74.33±0.15</strong></td>
<td><strong>79.30±0.68</strong></td>
</tr>
</tbody>
</table>

Table 3: Layer chosen for Self-Attention Pooling.

<table>
<thead>
<tr>
<th>Layer Metric</th>
<th>CoLA</th>
<th>RTE</th>
<th>MRPC</th>
<th>Accuracy</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS12</td>
<td>56.5 (−)</td>
<td>65.7 (−)</td>
<td>84.1 (−)</td>
<td>88.9 (−)</td>
<td></td>
</tr>
<tr>
<td>L12</td>
<td>59.8(↑3.3)</td>
<td>68.2(↑2.5)</td>
<td>86.8(↑2.7)</td>
<td>90.7(↑1.8)</td>
<td></td>
</tr>
<tr>
<td>L10-12</td>
<td><strong>60.1(↑3.6)</strong></td>
<td><strong>68.6(↑2.9)</strong></td>
<td><strong>87.3(↑3.2)</strong></td>
<td><strong>91.0(↑2.1)</strong></td>
<td></td>
</tr>
<tr>
<td>L9-12</td>
<td>59.8(↑3.3)</td>
<td><strong>69.7(↑4.0)</strong></td>
<td>86.6(↑2.5)</td>
<td>90.6(↑1.7)</td>
<td></td>
</tr>
<tr>
<td>L1-12</td>
<td>59.5(↑3.0)</td>
<td><strong>69.7(↑4.0)</strong></td>
<td>83.8(↑0.3)</td>
<td>88.7(↓0.2)</td>
<td></td>
</tr>
</tbody>
</table>

6 Conclusion

We propose Self-Attention Pooling to learn representation and pooling simultaneously, allowing the structure representation learning to take element relation into account. Self-Attention Pooling substantially improves the sequential structure and non-sequential structure.
References


Francesco Orsini, Paolo Frasconi, and Luc De Raedt. 2015. Graph invariant kernels. In Twenty-Fourth International Joint Conference on Artificial Intelligence.


Table 4: Results on GLUE.

A Results on GLUE

We use the BERT\textsubscript{base} model implemented by Transformers (Wolf et al., 2020), and follow the default setting of their "text-classification" directory for the training and evaluation on GLUE without tuning any hyper-parameters. Table 4 shows the full results and average performance. For STS-B, we report Pearson metric. For other new tasks, we report accuracy. On average, Self-Attention Pooling improves CLS Pooling by 1.3 points.

B Experiment Details on Graph Classification

Our experiments on graph classification (Section 4.2) follow the implementation of the "proteins_topk_pool.py" file in pytorch-geometric (Fey and Lenssen, 2019). We set three GNN layers and apply pooling for each layer, retaining 80% nodes at a time. The Self-Attention Pooling implemented on each layer only takes the self-attention of the current layer into account.