Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks

喻小惠

摘要: 文本分类是自然语言处理中的一类基础任务，近年图神经网络也开始应用到此领域。然而，现存的基于图的文本分类方法既不能捕获词的上下文语境也不能实现新词的归纳式学习。在本论文中为了解决以上问题提出了 TextING 方法，首先为每一个文档构建图然后利用自身的图结构学习节点表达，这种方法也能为从未出现的新词产生节点表达，最后通过设计读出层将单个图中的节点表达整合为图表达，实验结果表明该方法在文本分类任务中表现出优越性能。

关键词: 文本分类；图神经网络；

1、 引言

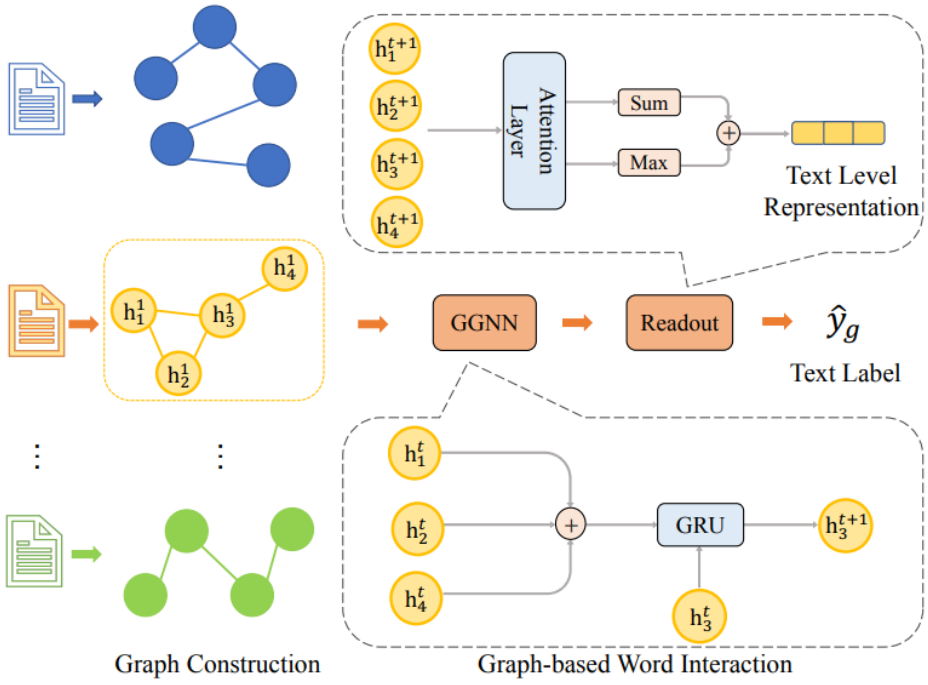
文本分类是自然语言处理的一个基础任务之一，受到众多学者的关注。早期文本分类方法根据领域知识，利用一系列预定义的规则来对文档进行分类。随着机器学习理论的不断完善，机器学习模型引发了学者们的巨大关注，通过利用标注样本，有效地学习到了文本和标签之间的关联。传统机器学习模型难以适用大规模数据，模型拓展性差，人们开始逐渐聚焦深度学习模型。基于机器学习的文本分类方法可大致分为两步：对数据提取特征，将提取的特征通过分类器来做分类预测。在基于机器学习的文本分类的分类过程中，朴素贝叶斯，逻辑回归[8]，决策树，支持向量机等方法被用做分类器使用。虽然基于机器学习的文本分类方法只需要较少的数据就能取得不错的效果，但仍存在费时费力，文本表示能力差等问题。随着深度学习的发展，自然语言处理研究者们开始将解决文本问题的方法聚焦到神经网络模型上。神经网络能自动提取特征，具有较强的表达能力，在解决文本分类问题上有着较好的效果。基于深度学习的文本分类方法离不开词向量的提出，词向量被广泛用于文本分类中。

2、 相关工作

2019 年，Yao 等人[1]提出了一种基于图卷积网络的文本分类模型 TextGCN。该模型基于词共现信息和文档词关系，构建了一个基于全局语料库文本图卷积框架。该文本图的节点为词和文档，用 one-hot 向量初始化，词和词之间边的权重与词和文档之间用分别用 PMI 计算和 TF-IDF 来计算，词和文档的节点表示通过 GCN 来更新，通过 softmax 层来预测文档标签。为了解决 GCN 存在的不能进行在线测试，消耗高内存问题，同年 Huang 等人[2]提出为每个文本建立图，参数进行全局共享，另外构图过程中采用一个小的滑动窗口，这样不仅提取到了更多的局部特征，还大大的减少了边的数量，以及拥有更低的内存消耗。为了解决短文本的稀疏性以及半监督场景问题，同年 Hu 等人[3]提出了一种新颖的异质图神经网络 HGAT 模型。首先，提出了一种灵活的异质图信息框架来建模短文本，然后提出了一种异质图注意网络用于基于双级注意力机制的短文本分类，该注意力包括节点级和类型级注意力。

3、 本文方法

3.1 本文方法概述



3.2 图构建

将首次出现的词作为图顶点，用节点特征作为初始嵌入，将顶点之间的共现关系作为边。其中珙共现关系是指两个词同时出现在适应性的滑动窗口（默认情况下滑动窗口的大小为 3）。然后用标准方法对文本进行预处理，包括停用词去除等。

3.3 基于门控图神经网络的节点交互

2015 年提出的门控图神经网络来学习节点的嵌入，节点间通过消息传递机制获取 k 阶内的邻居节点信息，然后通过聚合算法结合自身信息完成自身表达更新。具体的交互公式是：

$$
\begin{aligned}
\mathbf{a}^t &= \mathbf{A}\mathbf{h}^{t-1}\mathbf{W}_a, \\
\mathbf{z}^t &= \sigma\left(\mathbf{W}_z\mathbf{a}^t + \mathbf{U}_z\mathbf{h}^{t-1} + \mathbf{b}_z\right), \\
\mathbf{r}^t &= \sigma\left(\mathbf{W}_r\mathbf{a}^t + \mathbf{U}_r\mathbf{h}^{t-1} + \mathbf{b}_r\right), \\
\tilde{\mathbf{h}}^t &= \tanh\left(\mathbf{W}_h\mathbf{a}^t + \mathbf{U}_h(\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{b}_h\right) \\
\mathbf{h}^t &= \tilde{\mathbf{h}}^t \odot \mathbf{z}^t + \mathbf{h}^{t-1} \odot \left(1 - \mathbf{z}^t\right),
\end{aligned}
$$

3.4 读出层

当节点的表达更新完成，他们需要被聚合成图级别的表达从而作为文本的表达，最终基于此完成本文分类。分别用两个多层感知机完成对特征信息的提取和非线性转换，另外使用平均池化和最大化池化方法完成图级别的文档表达，表达公式如下所示：

$$
\begin{aligned}
\mathbf{h}_v &= \sigma\left(f_1(\mathbf{h}_v^t)\right) \odot \tanh\left(f_2(\mathbf{h}_v^t)\right), \\
\mathbf{h}_{\mathcal{G}} &= \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \mathbf{h}_v + \text{Maxpooling}\left(\mathbf{h}_1 ... \mathbf{h}_{\mathcal{V}}\right),
\end{aligned}
$$

4、 复现细节

4.1 与已有开源代码对比

本复现工作主要以论文原作者发布在 GITHUB 上的开源代码为基

准，探索了几种经典的消息传递方法来更新节点表达以及不同的池化方法对最终文本分类正确率的影响。下图中展示了以最大池化和平均池化共同整合节点表达的方法。

```
def _call(self, inputs):
    x = inputs

    # soft attention
    att = tf.sigmoid(dot(x, self.vars['weights_att']) + self.vars['bias_att'])
    emb = self.act(dot(x, self.vars['weights_emb']) + self.vars['bias_emb'])

    N = tf.reduce_sum(self.mask, axis=1)
    M = (self.mask-1) * 1e9

    # graph summation
    g = self.mask * att * emb
    g = tf.reduce_sum(g, axis=1) / N + tf.reduce_max(g + M, axis=1)
    g = tf.nn.dropout(g, 1-self.dropout)
```

4.2 实验环境搭建

在 anaconda 中下载代码运行需要的扩展库，然后在 pycharm 中完成代码编写和调试。

5、 实验结果分析

```
Epoch: 0187 train_loss= 0.19554 train_acc= 0.92435 val_loss= 0.58627 val_acc= 0.80563 test_acc= 0.79178 time= 3.93364
Epoch: 0188 train_loss= 0.19778 train_acc= 0.92044 val_loss= 0.58890 val_acc= 0.79718 test_acc= 0.79207 time= 3.90991
Epoch: 0189 train_loss= 0.19481 train_acc= 0.92013 val_loss= 0.60191 val_acc= 0.79296 test_acc= 0.78869 time= 3.93743
Epoch: 0190 train_loss= 0.19483 train_acc= 0.92216 val_loss= 0.62420 val_acc= 0.79296 test_acc= 0.79122 time= 4.00501
Epoch: 0191 train_loss= 0.19899 train_acc= 0.91826 val_loss= 0.62543 val_acc= 0.79718 test_acc= 0.78953 time= 4.01775
Epoch: 0192 train_loss= 0.19807 train_acc= 0.91904 val_loss= 0.62436 val_acc= 0.78592 test_acc= 0.78925 time= 3.90791
Epoch: 0193 train_loss= 0.18898 train_acc= 0.92216 val_loss= 0.61385 val_acc= 0.78451 test_acc= 0.78362 time= 3.90727
Epoch: 0194 train_loss= 0.18995 train_acc= 0.92435 val_loss= 0.59905 val_acc= 0.79437 test_acc= 0.78334 time= 3.94132
Epoch: 0195 train_loss= 0.19979 train_acc= 0.91841 val_loss= 0.60570 val_acc= 0.79296 test_acc= 0.78953 time= 3.84500
Epoch: 0196 train_loss= 0.19490 train_acc= 0.91638 val_loss= 0.60415 val_acc= 0.79718 test_acc= 0.79066 time= 4.02671
Epoch: 0197 train_loss= 0.20188 train_acc= 0.91529 val_loss= 0.62920 val_acc= 0.78310 test_acc= 0.78953 time= 4.07687
Epoch: 0198 train_loss= 0.20930 train_acc= 0.91200 val_loss= 0.60619 val_acc= 0.80000 test_acc= 0.79038 time= 4.01748
Epoch: 0199 train_loss= 0.19883 train_acc= 0.91982 val_loss= 0.60524 val_acc= 0.79296 test_acc= 0.78756 time= 4.08576
```

在 train.py 文件中展示了如何键入不同的数据集进行训练。以上结果是数据集 MR 的运行结果，分类准确率能达到 79.2%。

6、 总结与展望

文章实验结果表明以单个文档构建图比全局图方法的分类准确率更高，特别是在新词比率大的数据集中，由此表现了该方法的强归纳性。但仍然可以考虑单个文本和全局文本对词交互的不同作用，比如可以设想两种方式按照 1：1 的比例进行节点信息融合是否会具有更好的性能。

参考文献

[1]Yao L, Mao C, Luo Y. Graph Convolutional Networks for Text Classification[C]. Proceedings of the AAAI conference on Artificial Intelligence, 2019: 7370-7377.

[2]Huang L, Ma D, Li S, et al. Text Level Graph Neural Network for Text Classification[C]. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019: 3444-3450.

[3]Linmei H, Yang T, Shi C, et al. Heterogeneous Graph Attention Networks for Semi-Supervised Short Text Classification[C]. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing ,2019: 4821-4830.