基于字符的卷积神经网络文本分类方法

陆晨昱
义语智能科技（上海）有限公司
DeepBrain
chenyu@deepbrain.ai

引言

文本分类作为人机对话系统中的重要一环，其性能好坏直接影响后续功能模块的执行，进而影响整个系统的表现。传统的文本分类算法很大程度上依赖于精心挑选的特征和设置的规则，依靠人为加入的对某一特定语言的先验知识来达到“理解”文本的效果。随着深度学习的兴起，各种基于深度神经网络的分类模型在图像、语音等领域都取得了很大的成功。近年来，深度学习也被广泛应用于自然语言处理领域，文本分类便是其中一个热门课题。

文本分类是一个典型的序列分类问题。根据给定的字符序列，我们需要输出其对应的类别标签。常见的用于文本分类的深度学习模型有卷积神经网络、循环神经网络、注意力机制等。由于深度学习最大程度上简化了特征工程和预处理，模型结构与训练方法的选择对最后结果好坏来说至关重要。

关键技术

在 smp2017-ecdt 测评中，我司基于深度学习的文本分类模型在封闭与开放测试中均取得了较好的成绩。本文将系统的介绍该文本分类模型的具体实现。我们认为以下罗列的三点是该模型取得较好表现的关键：

字向量预训练

与传统方法不同的是，基于深度神经网络的文本分类模型是数据驱动的。给定训练样本及标签，模型会学习到对分类任务有用的特征及规则。由于整个模型的参数是随机初始化的，且我们没有人为的引入任何语言的先验知识，在训练样本较少的情况下得到的模型可能泛化能力不足。

我们的模型采用字作为建模单元，因此为了让模型对输入的字有一个初步的“认识”，我们采用无监督训练的方式对字向量做了预训练。我们尝试了多种预训练的方法，最终采用的方法是用无标签的短文本语料训练一个变分自编码器[1]，而后取其中的字嵌入矩阵作为分类模型中字向量的初始值。

在分类模型的初始化阶段，我们加载经过预训练得到的字嵌入矩阵，模型的其它参数仍采用随机初始化。随后我们采用通常的监督训练的方式对所有参数进行训练，包括预训练后的字向量。后续的实验表明，对字向量做预训练有助于分类模型更快的收敛，并且同时降低模型的训练及测试误差。

残差卷积网络

卷积神经网络[2]在序列分类任务中有着出色的表现。针对短文本分类任务的特点，模型不需要长跨度的记忆能力。同时，相比循环神经网络，卷积神经网络有着更好的并行性能。因此在该任务中卷积网络是一个合适的选择。

由于自然语言处理中的输入是离散且稀疏的，建模能力较强的模型在文本分类任务上很容易发生过拟合。所以通常我们在模型的选取上偏向小而简单的模型，例如较少层数的卷积网络，甚至是一个简单的线性分类器。然而在防止过拟合的同时，我们也限制了分类器的性能。

为了提升模型的性能，我们尝试使用更深的卷机网络来做分类。我们发现，在文本分类任务上，简单的增加网络层数反而会引起模型的性能下降。我们进一步尝试了残差网络[3]，在层与层之间引入跳跃连接。经过实验，我们发现残差网络的特性使得多层的卷机网络模型表现有了显著的改善。

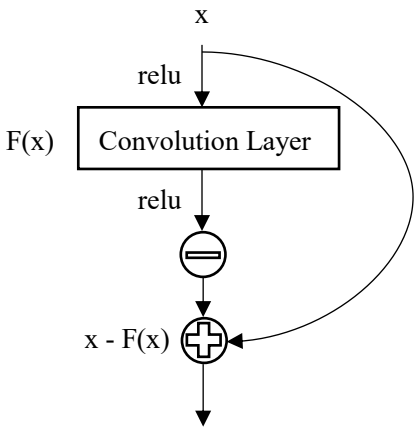

Figure 1 自抑制残差卷积模块结构

我们进一步尝试了不同结构的残差模块，提出了如图 1 所示的自抑制残差卷积模块（**S**elf-**I**nhibiting **R**esidual Convolution Block）。与一般残差模块不同的是，我们将卷积操作的输出通过修正线性单元（relu）激活函数，然后取其负值与该模块的输入相加。由于 relu 函数的特性，在经过这样一个模块的处理后，输出相对输入是被抑制过的。实验表明，该自抑制残差卷积模块相比传统的残差模块有更好的泛化能力。

集成学习

实验中，我们进行了多次重复的训练过程，得到的模型在测试中的表现有小幅浮动。进一步检视模型分类错误的句子，我们发现每个模型在出错的句子上不尽相同。为了进一步提升模型的性能，我们引入了集成学习[4]。在集成学习中，我们综合多个模型的决策来生成最终的结果。该方法可以避免单个模型的不足，显著提升最终结果的可靠性。

我们采用的是同构集成。在训练阶段，我们并行训练了一组结构完全相同的文本分类模型。在测试阶段，我们将所有模型输出的概率分布的平均值作为最终的输出，并取其中的最大值所在类作为输出类别。

模型结构

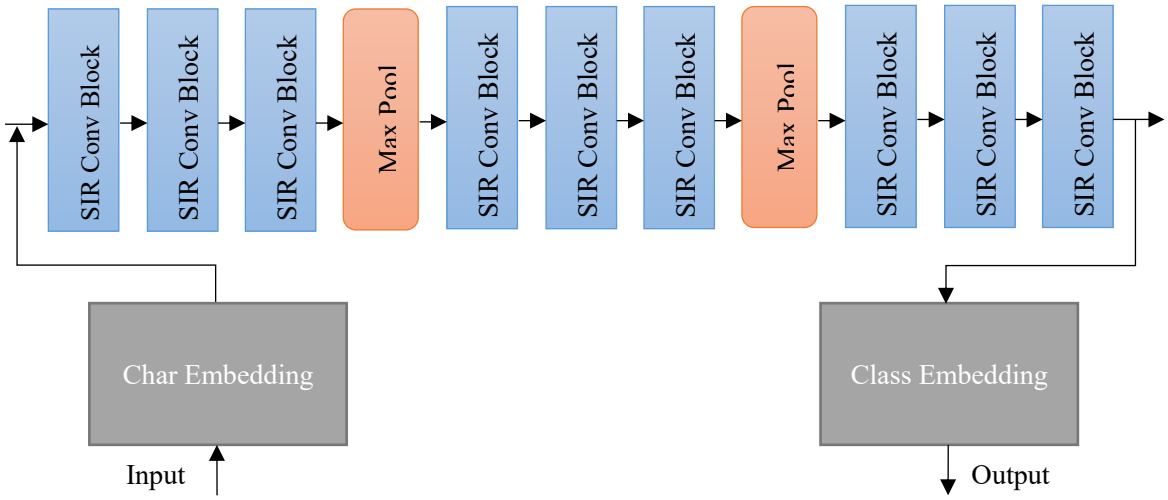

Figure 2 文本分类模型结构

如图 2 所示，整个文本分类模型由多个串联的自抑制残差卷积模块与最大池化层交替构成。其中，卷积与最大池化均为一维操作，作用于输入序列的时间轴上。在整个模型中，卷积核的大小统一为 3，最大池化的窗长统一为 2。模型的隐层大小与字嵌入、类嵌入大小均为 512 维。在卷积与最大池化操作中，我们用 0 值补齐输入的两端，使得这两种操作的输出与输入有相同的形状。

训练流程

首先，我们对分类模型中的字向量进行预训练。我们准备了 6 亿量级的无标签短文本语料，采用无监督训练的方式来获得所需的字嵌入矩阵。我们构建了一个变分自编码器，将输入文本映射到连续的语义空间后，再用一个生成器重现原句子。同时，我们用一个标准的高斯分布作为先验来约束句子的语义表示。训练后的模型可用来采样生成短句子。当然，我们需要的只是这个模型中训练得到的字嵌入矩阵。

接下来是文本分类训练数据的预处理。由于我们采用“字”作为输入单元，因而无需对文本做分词。在预处理中，我们仅做了去除标点符号这一操作。加入这一步骤的原因有二：其一是我们认为标点符号并不包含对文本分类有用的信息；另外，我们用于预训练的文本是不包含标点符号的，因此预训练得到的字向量中也不包含标点符号。

准备好训练数据后，我们用传统的监督学习方式训练文本分类模型。训练的损失函数为输出分布与真实标签的交叉熵。我们使用 adam 优化器[5]来做迭代优化。训练中我们将学习率设定为 0.001，批(batch)大小设为 64。同时，为了减少训练时间，我们对不同长度的训练文本做了分组(bucketing)。

考虑到训练集中各类别样本的数量可能存在不均衡的情况，我们在训练中调整了采样的策略。我们将不同类别标签的样本分组，采样时先随机挑选类别组，再从选中的类别组中选出一条样本。这样做可以保证最终每个批中不同类别的样本出现的概率是均等的。

训练中，我们构建了 32 个相同结构的分类模型并随机初始化以不同的参数。所有模型的字向量则统一加载由预训练得到的字嵌入矩阵。训练中的每一步我们都随机采样得到 32

个不同的批，分别输入不同模型并行训练。模型训练时长固定为 500 步。

结果分析

Table 1 模型分类错误的部分样本

输入文本	真实类别	输出类别
我想看电影台北飘雪	cmd_video	cmd_cinemas
可以帮买火车票吗	cmd_chat	cmd_train
今天什么天气	cmd_chat	cmd_weather
打开猫扑	cmd_website	cmd_app
三安光电	cmd_stock	cmd_video
鼻息肉	cmd_health	cmd_cookbook

为了分析模型的性能，我们用训练集训练模型后，用验证集作为测试集来检验模型。表 1 中列出了部分具有代表性的分类错误的样本。我们粗略的将这些出错样本分为三类：

1) 我们可以看到，其中像是“可以帮买火车票吗”、“今天什么天气”应该属于样本有误。结合训练数据与类别含义来看，模型输出的类别更准确一些。
2) “我想看电影台北飘雪”、“打开猫扑”这两个样本则是可能属于多个类别。如用户是想在线播放，则前一句应属于 cmd_video；若是想去电影院看，则应属于 cmd_cinemas。同理，猫扑可以是指网站，也可以是手机上的 app。在这种情况下，仅根据文本的信息无法做出可靠的正确分类。我们认为在实际产品中，需结合额外的场景信息或用户的上下文信息来做出判断。如信息缺失，则可以通过询问用户的方式来进一步确认。
3) 由于训练样本数量较少，而某些类别是有大量的领域相关词汇的，是给定的训练样本远远无法覆盖的。因而导致模型无法分清“三安光电”是股票还是视频，错认为“鼻息肉”是某一种菜名。针对这类错误，提升训练样本的数量是一种方法。我们认为更好的办法是引入领域相关知识，例如构建股票名与常见病症名的数据库来做检索。

深度学习是一类强力的工具。然而在解决实际问题中，仅依赖深度学习是远远不够的。我们认为基于对问题的深刻理解，将深度学习与传统方法、与场景相结合，是进一步提升表现的正确途径。

参考文献

[1] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” arXiv:1312.6114, 2013.
[2] X. Zhang, J. Zhao, and Y. LeCun, “Character-level Convolutional Networks for Text Classification,” 2015 Conference on Neural Information Processing Systems, pp. 3057–3061, 2015.
[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.
[4] L. K. Hansen and P. Salamon, “Neural Network Ensembles,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993–1001, 1990.
[5] D. P. Kingma and J. L. Ba, “Adam: a Method for Stochastic Optimization,” 2015 International Conference on Learning Representations, pp. 1–15, 2015.