计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(3):177−183 [doi: 10.15888/j.cnki.csa.007827]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于宽度和词向量特征的文本分类模型①

李雪松

(中国银行总行 个人数字金融部, 北京 100818)
通讯作者: 李雪松, E-mail: lxs0731@mail.ustc.edu.cn

摘　要: 针对词向量文本分类模型记忆能力弱, 缺少全局词特征信息等问题, 提出基于宽度和词向量特征的文本分类模型 (WideText): 首先对文本进行清洗、分词、词元编码和定义词典等, 计算全局词元的词频-逆文档频度 (TF-IDF) 指标并将每条文本向量化, 将输入文本中的词通过编码映射到词嵌入矩阵中, 词向量特征经嵌入和平均叠加后, 和基于 TF-IDF 的文本向量特征进行拼接, 传入到输出层后计算属于每个分类的概率. 该模型在低维词向量的基础上结合了文本向量特征的表达能力, 具有良好的泛化和记忆能力. 实验结果表明, 在引入宽度特征后, WideText 分类性能不仅较词向量文本分类模型有明显提升, 且略优于前馈神经网络分类器.

关键词: Word2Vec; FastText; WideText; 文本分类

引用格式: 李雪松.基于宽度和词向量特征的文本分类模型.计算机系统应用,2021,30(3):177–183. http://www.c-s-a.org.cn/1003-3254/7827.html

Text Classification Model Based on Width and Word Vector Feature

LI Xue-Song

(Digital Personal Banking Department, Bank of China, Beijing 100818, China)

Abstract: To resolve the issues of weak memory ability and no global word feature information in the word-vector-based text classification model, we propose a text classification model (WideText) based on the width and word vector features. Firstly, text cleaning, word segmentation, unit encoding and dictionary definitions are carried out. Secondly, the Term Frequency-Inverse Document Frequency (TF-IDF) index of the global word units is calculated and each text is vectorized. Furthermore, the words in the input text are mapped to the word embedding matrix through encoding. After the word vector features are embedded and averagely superimposed, they are spliced with the text vector features based on TF-IDF and transmitted to the output layer. Finally, the probability of the features belonging to each category is calculated. The proposed model combines the expressive ability of text vector features on the basis of low-dimensional word vectors and has excellent generalization and memory abilities. The experimental results show that after the introduction of the width feature, the WideText classification performance is significantly improved in comparison with that in the word-vector-based text classification model and also slightly better than that in the feedforward neural network classifiers.

Key words: Word2Vec; FastText; WideText; text classification

1　引言

随着社交网络的发展, 各种应用软件层出不穷, 在吸引大量用户的同时, 产生了大量的文本信息, 如何从大量文本中抽取有价值的信息和知识是研究者持续关注的问题. 文本分类作为一种信息获取的方法, 在自然语言处理中发挥着重要的作用, 如, 信息检索、文档分类和排序等. 上世纪 50 年代就有学者对文本分类进行研究, 早期的文本分类主要依靠人工规则, 后来转向基

① 收稿时间: 2020-06-15; 修改时间: 2020-07-14, 2020-08-19; 采用时间: 2020-08-25; csa 在线出版时间: 2021-03-03

于统计的方法, 又逐渐发展到通过深度学习、知识图谱等方式.

文本是知识的载体, 本身就蕴含了大量的语义、词性和时态等隐含信息, 随着通信技术的发展, 文本的处理规模也越来越大. 在此背景下, 词向量技术应运而生, 它通过大量语料的训练, 将词映射到低维度的向量中, 通过求余弦的方式, 可以判断词之间的近似关系. 词向量模型学习了词的上下文关系, 具有良好的泛化能力, 可以揭示海量数据里所承载的复杂而丰富的信息, 因此得到广泛应用. 在近些年, 词向量被应用到文本分类模型中, 其训练的准确性可以和深度学习分类器看齐.

通用搜索引擎面临的问题是如何获取记忆和泛化能力, 一方面用户希望通过检索词可以准确知道包含检索词的文档, 另一方面, 用户也希望获取和检索词相关的文档. 文本分类器也具有类似性, 人们希望通过输入文本可以准确得到分类, 并且通过一些相似的文本也可以得到分类结果. 因此, 提升一个分类器的效果, 可以从泛化和记忆两个方面考虑. 基于词向量的文本分类模型具有良好的特征泛化能力, 但由于文本词分布的稀疏性, 一些重要特征由于缺乏样本而得不到充分学习. 另一方面, 词向量模型将样本中的词变换到一个低维向量上, 其后会不断累加特征信息, 因而在一定程度上会丢失部分特征信息. 本文在词向量文本分类器的基础上, 提出了基于宽度和词向量特征的文本分类模型, 该模型同时具有词向量模型的泛化能力以及线性模型的记忆能力.

2 相关工作

从自然语言产生开始, 它的信息表达和传递方式逐渐发展成具有一种上下文相关的特性. 因此, 要让计算机处理自然语言, 一个主要的问题就是围绕自然语言上下文相关特性建立数学模型. 统计语言模型正是在此背景下产生, 它是当今自然语言处理任务的基础, 20 世纪 70 年代由 Jelinek 提出的 N-gram 模型是最常用的统计语言模型之一, 并在自然语言处理系统中得到广泛应用[1]. Salton 提出的向量空间模型 (Vector Space Model, VSM)[2], 把文本看作是一组特征的随机排列, 被称为词袋 (Bag of Words, BoW) 模型.

Hinton 等在 1986 年首次提出分布式表示 (distributed representation)[3], 用分布式表示来标识词, 通常被称为词向量. 从上世纪 90 年代开始, 基于统计和机器学习的文本分类方法逐渐盛行.

Bengio 等在 2003 年提出神经网络语言模型[4], 并认为如果用传统的稀疏表示法, 如, 常见的 One-hot 编码[5], 在解决某些任务时会造成维度灾难. 因此, 如果在深度学习的任务中使用高维向量, 其复杂度是非常高的, 通常在使用词向量时一般是基于低维的词向量.

Schwenk 等在 2007 年将神经网络语言模型运用到大词汇量连续语音识别中[6], 为了解决自然语言稀疏问题, 通过连续向量来对语言模型概率进行估计, 该模型比传统的 N-gram 语言模型具有更低的字错误率.

Mikolov 等在 2013 年介绍了 Word2Vec 工具[7], 从大量文本语料中以无监督的方式学习语义知识, 其底层采用了跳元 (Skip-Gram) 和连续词袋 (CBoW) 算法的神经网络模型, Skip-Gram 是利用给定的目标词来预测其上下文, CBoW 是根据上下文去预测目标词. 该工具通过将词映射到一个低维向量上, 意义接近的词被映射到相近的位置, 给文本数据的表示提供了一种新的思路, 它被大量地用在自然语言处理 (NLP) 中.

Kim 等在 2014 年将卷积神经网络 (CNN) 运用到文本分类模型[8], 该模型将预训练好的词向量矩阵作为卷积神经网络的输入, 通过一组不同大小的卷积核进行从前向后的卷积运算, 进而提取出局部相关性特征, 再根据标注数据训练出分类模型.

Liu 等在 2016 年将递归神经网络 (RNN) 运用到文本分类模型[9], 由于 RNN 在训练中存在梯度消失的问题, 因此, 该模型引入了擅长学习长距离依赖信息的长短期记忆网络 (LSTM)[10] 以解决该问题.

Cheng 等在 2016 年提出宽深度 (Wide & Deep) 模型[11], 并认为可以通过加强记忆和泛化能力来提升推荐系统的效果, 记忆可以被定义为在推荐系统中将历史数据重现, 而泛化是基于数据相关性的传递性, 探索过去从未或很少出现的特征. 宽深度模型在深度神经网络的基础上利用线性交叉特征有效记忆稀疏特征之间的关联, 结合了深度神经网络的泛化能力和线性模型的记忆能力, 该模型被应用到 Google Play 商业移动应用商店中, 实验效果表明应用下载量有明显增加.

由 Joulin 等在 2016 年提出了 FastText 文本分类模型[12,13], 结合了词袋、N-gram 袋以及子词 (subword) 信息, 具有性能高、分类准的优点. 词袋中词序是固定不变的, 为了获取词序信息, FastText 支持 N-gram 扩

展, 同时使用哈希算法对大量的 N-gram 进行映射, 以降低算法复杂度. 利用 Word2Vec 生成的词分布表示, 忽略了词形变化, 如时态、词根、词缀和人称等, 因此, FastText 模型对此进行改进, 将单词进行更细粒度的拆分, 引入子词信息, 以 panda 为例, 设定子词长度为 3, 其子词分别为<pa,pan,and,nda,da>, 以及其本身<panda>, 可以看出其中的 and 和单词<and>不同, 模型通过子词获取了更加细微的特征. FastText 模型架构如图 1 所示, 它分为输入层、隐藏层和输出层, w_1, ⋯, w_N 为文本的 N-gram 和子词输入信息, 传入隐含层进行向量叠加, 再进入输出层, 输出层使用 Softmax 计算属于各分类的概率, 当分类数很大的时候, 为了提高计算效率, 采用基于哈夫曼编码树的分层 Softmax, 提高了计算效率. Word2Vec 和 FastText 都是利用词向量特征, 不同在于 Word2Vec 预测的训练目标是下一个关联词, 而 FastText 的训练目标是分类.

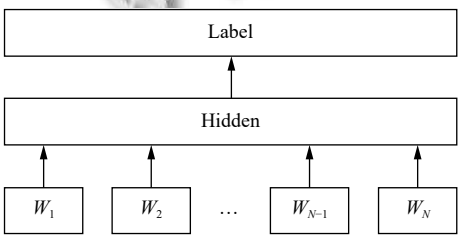

图 1 FastText 分类模型框架

FastText 的子词机制可以学习英文单词中词根和词缀粒度的形态学特征, 一般情况下, 词缀的出现次数要多于词根, 而词根可以带来更多的语义信息, 因此, Choi 等[14] 利用改进的逆文档频率算法对 FastText 子词机制进行了优化, 将词根和词缀区别对待, 加大了对词根的学习权重.

2018 年由 Peters 等[15] 提出 ELMO, 是基于语言模型的动态词向量, 根据上下文的语义去调整相应词向量, 能有效处理一词多义问题. 不同于传统的词向量, 每一个词对应唯一的词向量, ELMO 对于不同上下文的同一个词有不同的表示.

3 WideText 文本分类模型

3.1 词向量特征

在生成词向量之前, 首先要对文本进行表征, 一般使用 N-gram 来表示文本, 常用的如 uni-gram、bi-gram 和 tri-gram. N-gram 是一种基于统计语言模型的算法, 该模型基于这样一种假设, 一个句子的第 n 个词的出现只与前面 $n-1$ 个词相关, 而与其它词无关, 整句的概率就是各个词出现概率的乘积. 假设一个句子 s 由 n 个词组成, $s=(w_1, w_2, \cdots, w_n)$, 那么整句的概率 $p(s)=p(w_1w_2\cdots w_n)=p(w_1)p(w_2|w_1)p(w_3|w_1w_2)\cdots p(w_n|w_1\cdots w_{n-1})$.

语言模型的预测目标是出现下一个词的概率, 在本文模型中, 预测目标是整句属于某个类别的概率. 根据 N-gram 词袋模型, 输入文本可表示为 $x=(w_1,\cdots,w_{len})$, 通过查询词嵌入矩阵, 映射为词向量, 并进入隐含层, 其第 n 个神经元输出结果计算式 (1) 如下:

$$ln = f\left(\sum_{s=1}^{len} W_{\text{embedding}}(s,n)x_s + b_n\right) \tag{1}$$

其中, $\forall n \in (1,\cdots,h)$, h 为隐含层神经元个数, x_s 为输入层 N-gram 词向量, len 为其长度, f 为隐含层激活函数, $W_{\text{embedding}}$ 为词嵌入矩阵权值, b_n 为偏差, l_n 为隐含层输出, 它会进入输出层进行下一步计算.

3.2 宽度特征

经对文本的词频统计发现, 集中在头部的高频词通常是常用词, 这些词样本相对充足, 通过词向量可以很好的拟合, 由于低频词缺乏样本, 词向量得不到充分训练, 因此, 模型对一些重要特征的表达能力较弱. 另外, 在词向量文本分类模型中, 隐含层会对多个词向量按列进行累加, 这个变换进一步抑制了稀疏特征的表达, 而且会抵消部分特征.

在本文的模型中, 采用基于词袋的 TF-IDF 向量作为文本抽取特征, 通过线性模型加强了模型对稀疏特征的记忆能力, 进而提升模型整体效果. TF-IDF 是一种用于信息检索与文本挖掘的常用加权算法, 由词频 (Term Frequency, TF) 和逆文档频率 (Inverse Document Frequency, IDF) 两部分组成, 用以评估一个字词对于一个样本集或语料库的重要程度, 如果某个字词在一文档中出现的频率 TF 高, 并且在其他文档中很少出现, 则认为该字词具有很好的类别区分能力, 适合用来分类. 在本文中, 设定文本输入特征有 d 个, 输入向量 $x=(x_1, x_2, \cdots, x_d)$, 则:

$$y = w^{\mathrm{T}}x + b \tag{2}$$

其中, 输出 $y=(c_1,c_2,\cdots, c_k)$ 是在 k 个分类上的概率,w 为权值矩阵, b 为偏差.

3.3 WideText 模型

WideText 模型结合了词向量和宽度模型的优点, 词向量模型主要用来学习文本中潜在信息, 能够很好

的刻画词与词之间的关联关系, 有利于提高模型的泛化能力. 宽度模型通过包含字词重要程度的文本向量, 从全量样本中获取重要的字词特征, 通过对历史特征的记忆, 弥补了词向量模型记忆能力的不足, 提高了模型的分类能力.

如图 2 所示, WideText 模型由输入层、隐含层和输出层 3 部分组成. 输入层由 Embedding 和 Wide 层组成, Embedding 层将每个样本中的词映射到一个向量上, Wide 层是一个由文本 ID-IDF 值组成一组特征向量; 隐含层类同神经网络模型中的隐含层, 其权值矩阵主要用来学习文本中的隐含信息, 在进入输出层之前, 该模型将对隐含层向量和 Wide 层向量进行拼接; 输出层接受到模型上一层传来的拼接向量, 最后计算出落在每个分类上的概率, 作为模型的输出结果. 如果是一个二分类问题, 那么从中间层到输出层将是一个线性模型, 如果是多分类问题, 从中间层到输出层将是一个多分类器模型.

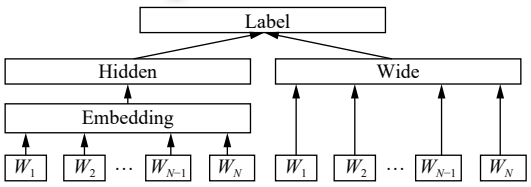

图 2 WideText 分类模型框架

WideText 模型输出层采用 Softmax 激活函数, 预测结果为 $P(Y=i)$, 如式 (3) 至式 (6) 所示:

$$P(Y=i)=\frac{e^{y_i}}{\sum_{r=1}^{k}e^{y_r}} \tag{3}$$

$$V_i=\sum_{m=1}^{d}W_{\text{wide}}(m,i)x_m \tag{4}$$

$$T_i=\sum_{n=1}^{h}W_{\text{text}}(n,i)l_n \tag{5}$$

$$y_i=V_i+T_i+b_i \tag{6}$$

其中, $\forall i\in(1,\cdots,k)$, k 为分类数, x_m 为线性特征, d 为其特征数量, l_n 为隐含层输出, W_{wide} 和 W_{text} 分别为宽度和词向量特征权值矩阵, b_i 为偏差.

在 WideText 中, 进行一次文本分类计算的时间复杂度如式 (7) 所示:

$$S+S\times H+(H+L)\times N+N \tag{7}$$

其中, S 为输入文本长度, H 为词向量维度, L 为宽度特征维度, N 为分类数. 首先进入输入层的是一个固定长度的词编码序列, 用来表示一段语句, 通过查表的方式从词嵌入矩阵中得到序列中每个编码对应的词向量, 这部分计算复杂度为 S, 通常可以忽略. 接下来, 词向量矩阵会进入到隐含层, 这步操作主要是将词向量矩阵按列进行平均叠加, 因此复杂度为 $S\times H$, 之后, 隐含层输出拼接宽度特征进入输出层, 复杂度为$(H+L)\times N$, 最后输出层计算属于每个分类的概率, 其复杂度为 N. 由此, 可以得出 WideText 计算复杂度比 FastText 多出的部分为$L\times N$, 两者的时间复杂度处于同一量级, 在常规的分类任务中, 对模型的性能影响不大.

利用 WideText 模型进行分类时, 主要步骤如下:

(1) 样本清洗. 去除样本中噪声, 如对分类过程无意义的词、停用词及标点, 有拼写错误的词进行修正.

(2) 词元编码. 遍历样本统计 N-gram 词频, 按照 N-gram 词频从高到低的顺序进行排序, 从 0 开始编码, 从 0 开始向后依次加 1 进行编码, 作为语料字典; 再次遍历样本, 按照字典编码将 N-gram 转化为字典编码; 同时将样本的标签转化为类别编码, 类别编码独立于词元编码.

(3) 样本编码. 设定样本最大长度 *maxlen*, 预留 0 为填充码, 1 为样本起始码, 2 为未登录词 (OoV), 3 为 N-gram 起始码. 在样本长度小于 *maxlen* 时, 用 0 进行补齐, 样本长度大于 *maxlen* 时, 截取头部样本, 使所有样本具有固定长度; 在生成样本编码前, 根据 N-gram 起始码对 N-gram 编码向后进行平移; 设定样本 N-gram 特征数 *nf*, 当 N-gram 编码大于 *nf* 时, 视为 OoV, 并将 N-gram 编码设置为 2.

(4) 打乱样本. 由于原始样本是按照相同分类标签集中存放, 为了避免后续训练出现过拟合现象, 需要对样本进行随机化打乱, 并且训练集中的分类标签需要和文本按照同样的顺序打乱.

(5) 生成宽度特征. 先根据样本词频由高到低的顺序生成字典, 再根据设定的宽度特征数对样本中词计算 TF-IDF 值, 其中, 训练集和测试集需要由同一套算法来计算, 以保证特征的一致性.

(6) 训练模型. 将编码后的样本以及 TF-IDF 文本向量分别输入到词向量和宽度特征输入层, 开始训练模型, 在每一轮训练完成时, 利用验证集对模型进行评估, 当迭代次数达到设定值, 完成模型训练.

(7) 模型预测. 将输入文本进行编码, 同时生成宽度特征, 将这些信息输入到模型, 从模型的返回结果中

得到预测结果.

4 实验

4.1 实验环境

实验环境为 Intel Core i5-8250U 处理器、主频 1.6 GHz、内存 8 GB、硬盘 237 GB 的 PC 机, 操作系统为 Win10, 编程语言为 Python3.5, 算法框架采用 Keras.

4.2 实验数据集及评价指标

数据集采用公开的单标签分类数据集 20 Newsgroups, 该数据集收集了 20 000 左右的新闻组文档, 均匀分在 20 个不同主题的新闻组集合, 是用于信息检索、文本分类和文本挖掘研究的国际标准数据集之一. 本实验采用的数据集一共有 18 821 条样本, 其中, 11 293 条样本作为训练集和验证集, 数量分别为 9034 条和 2259 条; 7528 条样本作为测试集, 分属电子、计算机硬件、棒球、曲棍球等 20 个不同新闻类别.

本实验采用准确率 (Accuracy)、精确率 (Precision)、召回率 (Recall) 以及 F 值 (F-measure)4 个指标对实验结果进行评价, 其公式分别如式 (8) 至式 (11) 所示. 准确率表示在预测的样本中被正确预测的样本数量; 精确率表示预测结果为正的样本中有多少实际正例; 召回率表示原样本中的正例有多少被预测为正; F 值综合了精确率和召回率的结果.

$$A_{\text{accuracy}} = \frac{TP+TN}{TP+FP+TN+FN} \tag{8}$$

$$P_{\text{precision}} = \frac{TP}{TP+FP} \tag{9}$$

$$R_{\text{recall}} = \frac{TP}{TP+FN} \tag{10}$$

$$F = \frac{2\times P_{\text{precision}} \times R_{\text{recall}}}{P_{\text{precision}} + R_{\text{recall}}} \tag{11}$$

其中, TP (True Positive) 表示被预测为正实际上是正例的数量, TN (True Positive) 表示被预测为负实际上是负例的数量, FP (False Positive) 表示被预测为正实际上是负例的数量, FN (False Negtive) 表示被预测为负实际上是正例的数量.

由于本实验中数据集有 20 个分类, 因此, 将多分类模型评价转化为多个二分类模型评价,在评价过程中, 分别计算在模型在测试集各个分类上的精确率、召回率和 F 值指标, 再计算各个指标的无加权平均值, 作为多分类模型评价结果. 如, 测试集中一个分类为“棒球”, 在计算该分类的评价指标时, 将其余的多个分类看作非“棒球”类, 通过模型结果分别计算“棒球”类的精确率、召回率和 F 值, 再分别计算其它分类的指标, 最后计算相应指标的平均值作为模型评价结果.

4.3 WideText 实验及参数设置

本实验的主要参数如表 1 所示.

表 1 WideText 模型参数

参数名称	参数值
词表数量	20 000
输入文本长度	400
每批样本数	32
词向量维度	50
迭代次数	15
Wide特征数	10 000
N-gram数	1
分类数	20
学习率	0.001

按照表 1 参数搭建 WideText, 利用训练集和验证集对模型进行训练, 得出分类模型, 再通过测试集对分类结果进行评价, 主要从准确率、精确率、召回率和 F 值 4 个指标进行评价.

如表 2 所示, 测试集分属 20 个类型, 每个类别用编号表示, 分别计算出各个分类的精确率、召回率和 F 值, 再按相应指标计算均值. 通过模型评价得出, 准确率、精确率、召回率和 F 值分别为 82.1%、82.0%、81.2% 和 81.3%.

4.4 实验模型与基本模型的实验对比

为进一步验证 WideText 模型的有效性, 分别选取 FastText、NN (Neural Network)、TextCNN 和 TextRNN 分类模型进行实验对比, NN 为单个隐含层前馈神经网络, TextCNN 为卷积神经网络文本分类模型, TextRNN 为递归神经网络文本分类模型. 在 WideText 和 NN 中, Wide 特征均采用基于 TF-IDF 的词向量来表示. 数据集依然使用 20 Newsgroups, 结果评价采用准确率、精确率、召回率和 F 值.

在对照实验中, 将数据集分别输入到 FastText、NN、TextCNN、TextRNN 和 WideText 中进行训练, 通过评价发现, WideText 在实验组中效果最优. 实验结果如表 3 所示, WideText 模型的准确率、精确率、召回率和 F 值分别为 82.1%、82.0%、81.2% 和 81.3%, 和 FastText 相比在准确率、精确率、召回率和 F 值上分别提高了 3.14%、3.02%、3.18%、3.17%; 和 NN 相

比, 在准确率、精确率、召回率和 F 值上分别提高了 0.98%、0.49%、0.74%、0.49%.

表 2 算法评价结果

类别编号	精确率(%)	召回率(%)	F值(%)
0	96.8	87.5	91.9
1	78.1	91.5	84.3
2	78.3	78.7	78.5
3	92.2	89.9	91.0
4	71.9	71.7	71.8
5	88.4	94.5	91.4
6	97.4	95.2	96.3
7	72.8	74.3	73.6
8	67.3	77.9	72.2
9	75.6	73.0	74.3
10	78.3	57.1	66.0
11	80.3	83.8	82.1
12	87.2	90.1	88.6
13	88.4	86.6	87.5
14	84.5	89.9	87.1
15	79.0	68.8	73.5
16	82.1	79.6	80.8
17	71.9	89.3	79.7
18	74.1	51.4	60.7
19	95.1	92.7	93.9
平均值	82.0	81.2	81.3

表 3 算法性能对比

模型	准确率	精确率	召回率	F值
TextRNN	0.717	0.727	0.709	0.709
FastText	0.796	0.796	0.787	0.788
TextCNN	0.805	0.804	0.800	0.800
NN	0.813	0.816	0.806	0.809
WideText	0.821	0.820	0.812	0.813

4.5 实验结论

通过实验发现, 引入 Wide 特征后, WideText 的性能较优, 证明了引入 Wide 特征是有效的. 和 FastText 相比, 采用 Wide 特征的 NN 性能有所提升, 说明 Wide 特征是一种不同于词向量的特征, 它包含了全局词重要性程度的信息, 并以向量形式来表征输入文本, 有利于提高模型对某些重要特征的记忆能力. WideText 结合了词向量和宽度特征, 在某些特征重现后, 通过模型可以准确识别, 因此, 同 FastText 相比, 模型在引入宽度特征后其特征记忆能力得到增强, 分类性能也得到提升.

5 结论

随着大数据技术的不断发展, 需要处理的文本日趋增多, 如何对文本进行准确有效分类一直是研究的热点. 本文认为文本分类性能的优化在某些方面类似于推荐系统算法优化, 通过加强模型泛化和记忆能力同样可以提升文本分类性能. FastText 利用了低维词向量具有的语义表达能力和泛化能力, 取得了不错的分类效果, 但它记忆能力较弱、缺少全局特征信息, 针对这些问题, 本文提出 WideText, 在词向量特征基础上增加了 Wide 特征, 并通过 TF-IDF 得到全局特征重要性信息, 该模型的输出结果综合了两个不同层面的特征的分类结果. 通过实验发现 WideText 分类性能有所上升, 且略好于单隐含层神经网络. Wide 特征提取简单, 并且对模型整体性能影响不大, 引入该特征不失为提升词向量文本分类性能的有效方法. 本文实验的数据集规模不大, 在海量文本分类模型中引入 Wide 特征表现如何, 是一个值得探索的方向. 另外, 深度学习被越来越多的应用到文本分类任务中, 其高维特征是否可以结合线性文本特征, 进而提升深度学习整体分类效果, 也是一个研究的方向.

参考文献

1 Jurafsky D, Martin JH. Speech and Language Processing. Upper Saddle River, N.J.: Pearson, 2000.9.

2 Salton G, Wong A, Yang CS. A vector space model for automatic indexing. Communications of the ACM, 1975, 18(11): 613–620. [doi: 10.1145/361219.361220]

3 Hinton GE, McClelland JL, Rumelhart DE. Learning distributed representations of concepts. Proceedings of the 8th Annual Conference of the Cognitive Science Society. Amherst, MA, USA. 1986. 1–12.

4 Bengio Y, Ducharme R, Vincent P, *et al*. A neural probabilistic language model. Journal of Machine Learning Research, 2003, 3(6): 1137–1155.

5 奚雪峰, 周国栋. 面向自然语言处理的深度学习研究. 自动化学报, 2016, 42(10): 1445–1465.

6 Schwenk H. Continuous space language models. Computer Speech & Language, 2007, 21(3): 492–518.

7 Mikolov T, Sutskever I, Chen K, *et al*. Distributed representations of words and phrases and their compositionality. Proceedings of the 26th International Conference on Neural Information Processing Systems. New York, NY, USA. 2013. 3111–3119.

8 Kim Y. Convolutional neural networks for sentence classification. Proceedings of 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar. 2014. 1746–1751.

9 Liu PF, Qiu XP, Huang XJ. Recurrent neural network for text

classification with multi-task learning. Proceedings of the 25th International Joint Conference on Artificial Intelligence. New York, NY, USA. 2016. 2873–2879.

10 Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735–1780. [doi: 10.1162/neco.1997.9.8.1735]

11 Cheng HT, Koc L, Harmsen J, *et al*. Wide & deep learning for recommender systems. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. Boston, MA, USA. 2016. 7–10.

12 Joulin A, Grave E, Bojanowski P, *et al*. Bag of tricks for efficient text classification. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Valencia, Spain. 2017. 427–431.

13 Bojanowski P, Grave E, Joulin A, *et al*. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 2017, 5: 135–146. [doi: 10.1162/tacl_a_00051]

14 Choi J, Lee SW. Improving FastText with inverse document frequency of subwords. Pattern Recognition Letters, 2020, 133: 165–172. [doi: 10.1016/j.patrec.2020.03.003]

15 Peters ME, Neumann M, Iyyer M, *et al*. Deep contextualized word representations. Proceedings of 2018 North American Chapter of the Association for Computational Linguistics: Human Language Technologies. New Orleans, LA, USA. 2018. 2227–2237.