义, 就称它是语义完整的. 语义完整性分析, 判别语义完整性, 有助于提高问答系统, 机器翻译以及主观题自动评分等应用系统的准确性.

1 相关工作

近年来深度神经网络发展迅速, 被广泛应用于语音识别、计算机视觉和自然语言处理等邻域中. 目前, 将深度学习技术应用于解决自然语言处理 (NLP) 任务是一个研究热点. 其中循环神经网络 (Recurrent Neural Network, RNN) 的特点是可以将某个时刻隐藏层的输出作为输入用于计算下一时刻隐藏层的输出, 所以适合来解决一些时间序列的问题. 而且不同于以往模型使用的固定序列长度, 它能够处理任意长度的序列.

文献[2]通过使用循环神经网络来训练语言模型, 提出了词向量 Word2Vec, 随后对词的分布式特征的研究不断兴起. 文献[3]使用循环神经网络来生成文本, 提出了一种新的 RNN 结构 MRNN, 提高文本生成能力. 文献[4]和文献[5]分别将循环神经网络和深度学习用在情感分析领域, 并取得了不错的效果. 文献[6]在序列标注的前提下利用双向循环神经网络模型进行中文分词, 通过增加词的上下文信息能够有效地解决梯度爆炸问题, 并取得了相对较好的分词效果. 文献[7]中提出了一种通过抽取句法、词汇、长度等特征分析逗号是否是子句边界的方法, 然而, 在本文语义完整性分析研究中, 若使用传统方法判断句子是否是语义完整, 一方面需要对句子进行句法、语法分析, 另一方面需要从分析结果中抽取合适的特征并且分析特征与结果的因果关系, 当问题较复杂时, 这种方式基本不可行. 文献[8]提出了一种基于循环神经网络的古文自动断句方法, 该方法采用基于 GRU 的双向循环神经网络对古文进行断句. 在大规模古籍语料上的实验结果表明, 该方法能够取得比传统方法更高的断句 F1 值, 但是该方法是针对于单个字进行标注用于断句, 并不适合本文以词语为单位进行标注的语义完整句切分.

本文试图将循环神经网络应用于中文句子语义完整性分析, 将句子语义完整性分析转换为典型的序列标注问题来处理.

2 语义完整性分析方法

本文提出的模型采用基于双层的 Bi-LSTM 循环神经网络, 结构如图 1 所示. 首先, 模型的输入为原始文本经过预处理后的词序列, 将其映射为相应的词向量并标注, 经过循环滑动窗口和欠采样处理后作为 Bi-LSTM 的输入. 然后通过双层 Bi-LSTM 更加准确地学习特征, 最终通过分类器输出相应标签概率.

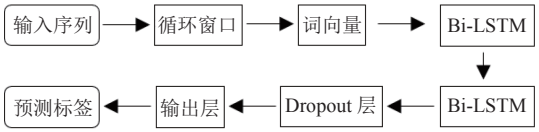

图 1 基于循环神经网络的语义完整性分析方法架构

2.1 分词与词向量

汉语词汇是语言中能够独立运用的最小语言单元, 是语言中的原子结构, 所以对中文进行分词是自然语言处理的基础. 分词的准确率将会直接影响后续词性标注, 句法分析, 词向量等相关工作的质量. 本文采用结巴分词的精确模式进行分词处理.

为了让计算机理解人类的自然语言, 把词表示为计算机适合处理的方式, 这样词向量的概念就应运而生了. 通常, 词向量有两种表示方式: one-hot representation 和 distribution representation.

One-hot representation 是一种离散表示, 它把每个词表示为一个长向量, 向量中只有一个维度的值为 1, 其余维度为 0, 这个维度就代表了当前的词. 例如: “物质”表示为[0, 0, 0, 1, 0, 0, 0, 0, 0, ⋯, 0, 0, 0],这种表示方式简单容易实现, 但缺点就是编码太过于稀疏, 将会导致维度非常高, 同时每个词本身的信息量太小, 无法展示词与词之间的关系.

Distribution representation 是将词转化成一种分布式表示, 是一种既能表示词本身又可以考虑语义距离的词向量表示方法. 它是一种低维实数向量, 例如: “物质”表示为[0.792, −0.177, −0.107, ⋯, 0.109, −0.542]. 这种分布式表示的优点在于它不但解决了维数灾难问题, 并且挖掘了词与词之间的关联属性,每一维度都有特定的含义, 包含了更多的信息, 从而提高了向量语义上的准确度. 因此, 近几年流行的语言模型 Word2Vec 就是采用这种方法表示词向量的.

2.2 LSTM 神经网络

近几年循环神经网络 (Recurrent Neural Network, RNN) 广泛应用于自然语言处理, 它引入了基于时间 (状态) 的循环机制, RNN 神经单元在某一时刻的输出依赖于当前的输入和以往时刻的信息, 同时这一时刻

隐藏层的输出也可以作为下一个神经单元的输入, 这样就能够保持数据的依赖关系, 有效利用信息.

经过大量实验证明, 当相关信息和当前预测位置之间的间隔变得非常大时, 普通循环神经网络就很难学习长期依赖, 原因在于梯度消失和梯度爆炸问题, 所以长短时间记忆 (Long Short-Term Memory, LSTM) 网络这种特定类型的循环神经网络就是专门设计出来解决这个问题的. LSTM 在以往的循环神经元结构基础上进行了改进, 它有四个不同的神经网络层进行信息的交互. LSTM 单个神经元的网络结构如图 2 所示.

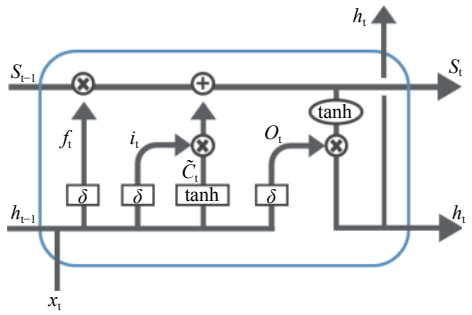

图 2 LSTM 神经元细胞

LSTM 通过“门”的结构来控制细胞状态, 门可以实现让信息选择性的通过, 它主要是包括一个非线性激活函数 Sigmoid 和一个点乘运算 Pointwise 操作来实现. 这样的门有三个, 分别是输入门、遗忘门和输出门, LSTM 通过这三个门来实现信息的存储和更新. 其中 Sigmoid 函数输出的是一个 0 到 1 之间的实数, 表示让对应信息通过的权重, 0 表示“不让任何信息通过”, 1 表示“让所有信息通过”. 它通过公式 (1)~(6) 进行计算,其中 x_t 表示 t 时刻的输入, f_t 表示遗忘门的输出、i_t 表示输入门的输出、o_t 表示输出门的输出、S_t 为 t 时刻的状态、h_t 为 t 时刻的输出.

$$f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right) \tag{1}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{3}$$

$$S_t = f_t * S_{t-1} + i_t * \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * \tanh(S_t) \tag{6}$$

Bi-LSTM 与 LSTM 本质上是一致, 只不过 Bi-LSTM 是在标准 LSTM 上加了一个反向的 LSTM, 这样某一时刻的输出就能由它前面和后面的若干输入共同决定, 充分利用句子的上下文信息, 相比标准 LSTM 信息更加全面.

2.3 改进的双层 Bi-LSTM 网络

本文采用改进的双层 Bi-LSTM 来进行训练[9], 其中每个层包含多个存储器单元, 能够更加准确地学习特征. 第一层 Bi-LSTM 给后一层的 Bi-LSTM 提供序列输出, 而不是单个值输出.

(1) 输入层

首先对经过清洗后的数据集进行分词, 然后采用四元标注集 T={S, B, M, E}进行标注. 定义 B 表示为一个语义完整句的开头词, M 表示为一个语义完整句的中间词, E 表示为一个语义完整句的结尾词, S 表示为特定符号 (, ;、等) 前面和后面最靠近的一个词. 例如: “物质世界的运动是绝对的, 而物质在运动过程中又有某种相对的静止”. 这个语义完整句对应的词序列和正确的标签为“物质/B 世界/M 的/M 运动/M 是/M 绝对/M 的/S 而/S 物质/M 在/M 运动/M 过程/M 中/M 又/M 有/M 某种/M 相对/M 的/M 静止/E”.

经过上述规则标注的标签数量会出现严重的类别不平衡问题[10], M 标签数量远大于其他标签, 我们采用改进的随机欠采样方法对 M 标签进行处理. 对于一个语义完整句子来说, E 和 B 标签的数目为 1, S 标签数目与句中标点符号有关, 一般是 2 个或者 4 个, 而 M 标签的数目可以达到 10 个左右. 其中连续的 M 标签是出现次数最多的, 并且其特征对我们语义完整性分析来说, 不是特别重要. 所以将左边和右边标签都为 M 的词依据一定比率进行丢弃, 丢弃原则根据为 M 标签数目略多于其他标签的数目即可, 具体丢弃比率设置可见 3.3 节中第四个对比实验. 处理前后的标签统计数量如图 3 所示. 虽然处理后的 M 依然大概占有近半的数目, 但通过后文对比实验表明, 本文所提出的改进的随机欠采样方法对类别不平衡问题有很大的改善.

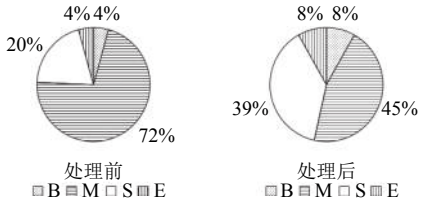

图 3 各个标签采样前后数目统计

随机欠采样后的词序列因为上下文特征改变, 可能会出现欠拟合的现象, 为了既对词序列进行采样, 又不丢失一个词应有的上下文信息, 本文提出滑动窗口的方法, 在随机欠采样前对序列数据进行处理. 对于一个有 n 个词的词序列 $T(1{:}n)$, 用大小为 k 的滑动窗口从首滑动至尾, 每次窗口内的子序列作为 Bi-LSTM 的输入. 假设 k 值为 5, 序列 T 中下标为 i(下标从 0 开始) 的词生成的子序列表示为 $(T_{i-2}, T_{i-1}, T_i, T_{i+1}, T_{i+2})$, 其中 $T_i=T_{[(n+i)\%n]}$.

(2) 双层 Bi-LSTM

为了方便说明, 这里假设滑动窗口的大小为 5 的双层 Bi-LSTM 其内部结构如图 4 所示.

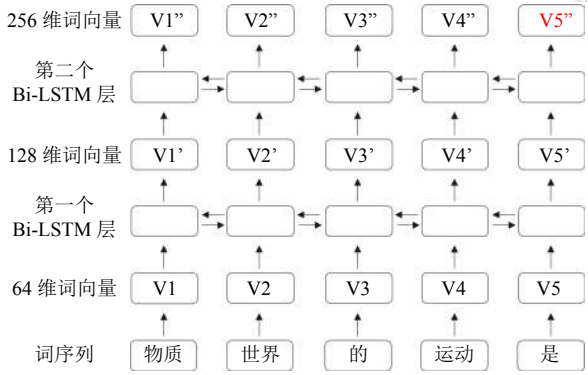

图 4 双层 Bi-LSTM 结构图

在输入层我们已经把输入的词序列转换为维度为 64 的词向量, 图 4 中小矩形的数目即序列的长度. 在第一个 Bi-LSTM 中, 这里输入为维度 64 的词向量, 输出为维度 128 的词向量, 由于其不是最后一层 Bi-LSTM, 这里会输出 5 个 128 维的词向量 V_1’…V_5’. 第二个 Bi-LSTM 输入为 V_1’…V_5’都为 128 维词向量, 经转换后得到 V_1”…V_5”为 256 维词向量, 当前已经是最后一层 Bi-LSTM, 所以这里规定 V_5”为窗口中间词即词向量 V_3 对应的输出.

(3) 输出层

深层神经网络中, 过拟合会使模型泛化性能变差, 为了防止过拟合, 模型中增加了 Dropout 层[11]. Dropout 层将在训练过程中每次更新参数时按一定概率随机断开输入神经元, 这就防止了神经元之间过度的协同适应.

Dropout 层的输出向量维度与 Bi-LSTM 的输出维度相同, 为了将向量维度转换为与标签类别数一致, 所以增加了一个全连接层, 并采用 elu 激活函数, 将 Dropout 层的输出转换为指定维度的向量. 最后对提取的特征采用 Softmax 激活函数得到输出的概率. Softmax 的函数定义如下:

$$S_i = \frac{e^{V_i}}{\sum_i^C e^{V_i}} \tag{7}$$

其中, V_i 是全连接层的输出, i 表示类别索引, 总的类别个数为 C, S_i 表示的是当前元素的指数与所有元素指数和的比值. 一个含任意实数的 K 维向量, 通过 Softmax 层后, 都会“压缩”到另一个 K 维实向量中, 压缩后的向量每个元素都在[0, 1]范围中, 并且所有元素的和为 1.

2.4 训练与预测

本文实际解决的是一个多分类问题, 采用的损失函数为交叉熵损失函数[12], 即模型的训练目标是使如下损失函数最小:

$$C = -\frac{1}{n}\sum_x [y\ln a + (1-y)\ln(1-a)] \tag{8}$$

其中, y 表示真实标签的分布, a 则为训练后模型的预测标签分布, 交叉熵损失函数可以衡量 y 与 a 的相似性. 此外, 交叉熵作为损失函数还有一个好处是能避免均方误差损失函数学习速率降低的问题, 因为学习速率可以被输出的误差所控制. 模型的训练过程采用 GPU 并行加速, 为了使模型达到更好的效果, 本文选取大量神经网络优化算法进行实验.

模型的预测过程即对于任意的输入序列, 深度神经网络输出的是每个时刻标注的条件概率 (参见 Softmax 输出格式). 预测过程中, 模型需要根据该输出值, 进一步输出对应的标签. 本文为了检验模型的准确性, 直接选取概率最大的标签作为预测结果.

3 实验

3.1 实验环境

实验环境如表 1 所示.

表 1 实验环境配置

项目	环境值
服务器	戴尔 PowerEdge T640
处理器	Intel(R) Xeon(R) Bronze 3106 CPU @ 1.70 GHz
内存	128 GB
显卡	4*GeForce RTX 2080 Ti
操作系统	Ubuntu 16.04
开发环境	Python3.6、TensorFlow、Keras

本文采用的数据集是宾州中文树库 (CTB)8.0 语料库, 总字数大概 130 万字, 采用自动标注和人工标注相结合的方法, 先将数据集中的几种标点 (.?!;) 视为语义完整的标志, 然后通过人工检查进一步提高标注的准确性. 最后随机的将数据集切分为 90% 的训练集和 10% 的测试集.

词向量数据来源为百度百科+维基百科+新闻+小说一共 120 g 数据, 词向量维度为 64 维. 本文在获取词向量过程中, 对于未登录词统一用特殊向量代替.

3.2 评估标准

本文属于多分类问题, 我们采用准确率 (*A*)、宏查准率 (*macro-P*)、宏查全率 (*macro-R*) 以及宏 *F*1(*macro-F*1) 作为评价模型效果的指标. *A* 为模型整体的准确率. 其他指标计算方式如下, 其中 n 为类别数, P_i、R_i 分别表示第 i 个类别的 P 值和 R 值.

$$\text{macro}-\text{P}=\frac{1}{n}\sum_{i=1}^{n}P_i \tag{9}$$

$$\text{macro}-\text{R}=\frac{1}{n}\sum_{i=1}^{n}R_i \tag{10}$$

$$macro-F1=\frac{2\times macro-P\times macro-R}{macro-P+macro-R} \tag{11}$$

3.3 模型参数设定

本文提出的模型影响实验结果的主要参数有神经元数目, 激活函数类型以及模型优化器的选择, 为了找到每个参数的较优解, 本文采用控制变量法, 分别作如下实验.

(1) 神经元数目

这里说的神经元数目, 准确来说是循环神经网络的前馈网络层中隐藏神经元的个数, 一般情况下, 隐藏神经元的数目越多, 模型就越复杂, 训练时间越长. 下面分别将隐藏神经元大小设定为 64、128、256、512, 实验结果如图 5 所示. 从实验结果可以看到, 在神经元数目较小时, 随着神经元数目的递增模型各指标递增的比较明显. 当神经元数目达到一定的值后, 各指标增速放缓, 最后甚至有下降的趋势. 为了兼顾模型效果和训练速度, 本文选取的神经元数目为 256.

(2) 激活函数的选择

激活函数的作用是给模型添加非线性因素, 增加模型对数据的适应性, 使得分类更加准确. 神经网络常用的激活函数有 sigmoid、tanh、elu、relu 等, 本文选取这四种激活函数分别进行实验, 实验结果如表 2 所示. 从图中可以看到, 不同激活函数对结果影响还是很大的, 根据实验结果, 本文选择 elu 作为激活函数.

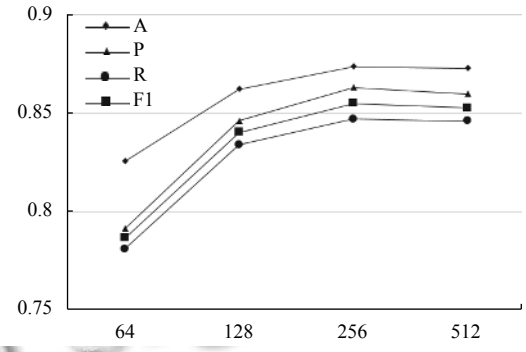

图 5 神经元数目对比实验结果

表 2 激活函数实验对比结果

激活函数	*macro-P*	*macro-R*	*macro-F*1	*A*
elu	**0.8654**	**0.8445**	**0.8548**	**0.8749**
tanh	0.7839	0.7623	0.7729	0.8205
relu	0.8539	0.8483	0.8511	0.8716
Sigmoid	0.7683	0.7228	0.7449	0.8039

(3) 模型优化器设定

模型优化器就是设置深度学习的优化算法, 主要目的是优化参数的更新策略. 不恰当的优化算法会导致模型收敛到局部最优解. 本文选择较常用的 adam 优化算法[13]及 adadelta 算法[14]作对比. 实验结果如表 3 所示. 其中 adam+decay 表示采用 adam 优化算法, 但是每次迭代完成后用 decay 值减小学习率. AMSGrad 为 adam 的变体[15]. 优化算法的参数选用默认值或者论文中的推荐值. 从表中可以看出, adam 的变种 (AMSGrad) 比原生的 adam 的效果要好得多. 而 adadelta 算法比 AMSGrad 算法要略好一点, 因此本文选取的模型优化算法为 adadelta 算法.

表 3 模型优化器对比实验结果

优化算法	*macro-P*	*macro-R*	*macro-F*1	*A*
adadelta	**0.8994**	**0.8917**	**0.8955**	**0.9010**
adam	0.7692	0.7053	0.7359	0.8047
adam+decay	0.8220	0.8071	0.8145	0.8382
AMSGrad	0.8816	0.8793	0.8804	0.8899

(4) 输入欠采样方案比较

对于分类不平衡问题, 本文采取改进的随机欠采样方法对输入数据进行处理. 实验结果如表 4 所示, 实验 1 为普通随机欠采样方式, M 标签占比为 50%. 实

验 2、3、4 为本文提出的改进随机欠采样方式, 采样后 M 标签占比分别为 50%, 45%, 40%. 从实验可以看出, 本文提出的改进随机欠采样方法比普通随机欠采样方法效果要好很多, 并且当 M 标签占比为 45% 时, 实验效果最好, 这可能是因为当 M 标签占比为 45% 时, 各标签分配比例刚好符合模型训练要求.

表 4 优化算法对比实验结果

实验组	*macro-P*	*macro-R*	*macro-F*1	*A*
1	0.8750	0.8782	0.8766	0.8808
2	0.8994	0.8917	0.8955	0.9010
3	**0.9184**	**0.9061**	**0.9122**	**0.9161**
4	0.9116	0.9052	0.9084	0.9084

3.4 模型对比实验

经过上述参数对比试验, 本文主要参数设定如下: Bi-LSTM 神经元数目为 256, Dropout 层设定比率为 0.5, 词序列的滑动窗口大小为 9, 训练神经网络的批次 (batch) 大小设定为 64, 训练循环次数 (epoch) 设定为 20, 每次循环结束将训练数据集进行 shuffle 处理. 采用 AdaDelta 优化算法, 全连接层激活函数为 elu. 为了说明本模型的有效性, 使用相同的数据集, 分别采用 RNN、LSTM、双层 LSTM、双层 Bi-LSTM 进行对比实验, 实验结果如表 5 所示.

表 5 模型对比实验结果

模型	*macro-P*	*macro-R*	*macro-F*1	*A*
RNN	0.6837	0.6328	0.6573	0.7425
LSTM	0.7417	0.7165	0.7289	0.7909
双层 LSTM	0.8255	0.8075	0.8164	0.8369
双层 Bi-LSTM	**0.9184**	**0.9061**	**0.9122**	**0.9161**

从结果可以看出, 本文提出的双层 Bi-LSTM 模型的准确率可以达到 91.61%, 优于其他模型. 一方面, 本文采用的循环窗口和随机欠采样方法可以在欠采样过程中很好的保留上下文特征. 另一方面, Bi-LSTM 能够更好的学习上下文特征, 且双层 Bi-LSTM 模型获取特征更准确, 因此可以达到较好的效果.

4 结束语

本文采用基于双层 Bi-LSTM 的循环神经网络模型, 对长文本实现自动标注, 从而实现语义完整性分析. 从实验结果和项目使用来看, 本方法可以较好的解决标注语义完整性的问题. 后续将模型用到生产环境的过程中, 可以结合标签之间的依赖关系, 对模型输出结果, 按照一定的词性规则进一步提升预测结果.

参考文献

1 林奕欧, 雷航, 李晓瑜, 等. 自然语言处理中的深度学习: 方法及应用. 电子科技大学学报, 2017, 46(6): 913–919. [doi: 10.3969/j.issn.1001-0548.2017.06.021]

2 Mikolov T, Chen K, Corrado G, *et al*. Efficient estimation of word representations in vector space. arXiv: 1301.3781, 2013.

3 Graves A. Generating sequences with recurrent neural networks. arXiv: 1308.0850v5, 2013.

4 Tang DY, Qin B, Liu T. Document modeling with gated recurrent neural network for sentiment classification. Proceedings of 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal, 2015: 1422–1432.

5 何炎祥, 孙松涛, 牛菲菲, 等. 用于微博情感分析的一种情感语义增强的深度学习模型. 计算机学报, 2017, 40(4): 773–790.

6 刁琦, 古丽米拉•克孜尔别克, 钟丽峰, 等. 基于循环神经网络序列标注的中文分词研究. 计算机技术与发展, 2017, 27(10): 65–68. [doi: 10.3969/j.issn.1673-629X.2017.10.014]

7 李艳翠, 冯文贺, 周国栋, 等. 基于逗号的汉语子句识别研究. 北京大学学报 (自然科学版), 2013, 49(1): 7–14.

8 王博立, 史晓东, 苏劲松. 一种基于循环神经网络的古文断句方法. 北京大学学报 (自然科学版), 2017, 53(2): 255–261.

9 Dyer C, Ballesteros M, Ling W, *et al*. Transition-based dependency parsing with stack long short-term memory. arXiv: 1505.08075, 2015.

10 赵楠, 张小芳, 张利军. 不平衡数据分类研究综述. 计算机科学, 2018, 45(6A): 22–27, 57.

11 Srivastava N, Hinton G, Krizhevsky A, *et al*. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014, 15(1): 1929–1958.

12 周志华. 机器学习. 北京: 清华大学出版社, 2016.

13 Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv: 1412.6980, 2014.

14 Zeiler MD. ADADELTA: An adaptive learning rate method. arXiv: 1212.5701, 2012.

15 Reddi SJ, Kale S, Kumar S. On the convergence of Adam and beyond. International Conference on Learning Representations. New York, NY, USA. 2018.