基于深度学习的中文文本情感分析

摘要

文章要解决的问题是提高中文文本情感分析的准确率，传统方法得到的准确率不超过 0.6[5]，故文章结合深度学习的方法解决该问题. 首先需要解决的问题是将语言中的字词转为向量表达，文章引入了深度学习算法——词嵌入. 而后利用循环神经网络训练数据，因为循环神经网络对于处理时间序列模型非常有效. 为了充分使用上下文信息，文章结合了长短期记忆网络，优化了原网络. 最终得到的结果是在不同的数据集上情感分析的准确率达 0.9. 后期优化考虑了先进行文本分类以及测试反讽数据，得到的结论是文本分类意义不大，在反讽数据上表现不佳，未来需进一步优化，可考虑增加中性这一情感标签，优化我们的网络. 这些结果的实际影响是，可对国内某事件进行舆情分析，比如在微薄平台上分析某件商品的用户评价、分析某事件人们的观点.

关键词：情感分析，词嵌入，循环卷积网络，长短期记忆网络，TensorFlow

目录

1. 引言

前些年互联网届日渐兴起的概念名词——Web 2.0，业内人士称它为互联网发展的过渡产物. 而随着信息技术的发展，建立在 2.0 基础之上的 Web 3.0 渐渐浮现其轮廓，为我们展现了未来世界的互联网新模式. 对于 Web 3.0 的具象概念众说纷纭，在此按下不表，本文开篇仅讨论其核心内涵 [1] .

最初的 Web 1.0 模式实现的是任何人可交易，主要代表为某些公司如 eBay，Amazon. 而近些年阿里巴巴在中国市场的成功，证明了 Web 1.0 推动力强大，影响持久. 目前我们身处于 Web 2.0 模式，其主要表现是实现任何人可参与，所有用户可产生内容，整个网络实现合作化、社区化，主要代表如 YouTube、Twitter 以及微博等. 所有的创造仅需连接互联网即可，人人皆是自媒体. 就目前情况来看，Web 2.0 似乎已满足了人类的一切幻想，那么 Web 3.0 将会如何呢？王靓（2008）认为 Web 3.0 将实现信息的高度整合和高度的智能化服务. 举个简单的例子，当用户牙疼需要看牙医时，智能代理（Intelligent Agent）会自动检查用户个人行程，搜索附近合适的牙医，自动与诊所的智能代理沟通并登记，最终成功安排好此次行程. 听起来不可思议，但这就是计算机科学家们一直致力于实现的互联网新模式，实现智能化的人与人和人与机器的交流.

在上述例子中，每一个操作都需要分析整合大量的信息，这些信息多是存储于文本之中，比如不同诊所的环境评价，牙医的患者评价以及朋友的推文（如，是否认识牙医朋友）等. 由此可见，Web 3.0 亟需文本分析辅助，由此诞生出的语义网络（Semantic Web）技术成为的主要技术，甚至有观点认为 [2] ，Web 3.0 等同于智能语义网.

在语义网络 [1] 中，网络内容可以被人类以及智能代理理解. 语义网络的核心包括一种哲学、一系列设计理念、相互协作的工作团体以及各种可用技术. 本文探索的文本情感分析即为其中一项基础的可用技术. 如上述牙医例子中，各种评价所包含的情感对于智能代理最终决策起关键作用.

深度学习作为近些年的热点技术，被各个领域用于解决复杂问题，文本情感分析所属的自然语言处理（Natural Language Processing）领域也不例外. 深度学习在大型数据集上表现优异，而语义网络恰好符合此特性，由此可见，以深度学习算法结合语义网络技术可以作为未来研究的一种思路.

最后，本文着眼于中文文本，因为汉语不同于英语，单词即意义；现代汉语多为双音词，甚至多音词，还包括多音字等等. 故而探索这些问题，对于理解未来语义网络服务中国市场起关键作用. 本文后期分析的主要数据来源为微博，作为全球用户规模最大的独立社交媒体平台 [3] ，我们可以从中获取海量且更新迅速的信息.

2. 相关研究

很多人乐意尝试在社交网站上进行文本情感分析，因为这样做可以了解人们对某件事物的真实看法，目前大多数人基于 Twitter 中的语料进行研究. 有的学者探索该技术在不同领域的应用，比如金融领域，检测某事件的受欢迎程度（Daniel, Neves & Horta, 2017），可以用来预测该事件所属公司的股票价格和业务量. 还有学者对情感分析进行算法优化，比如利用布谷鸟搜索算法（Cuckoo Search）进行 Twitter 上的文本情感分析（Pandey, Rajpoot & Saraswat, 2017）. 也有学者考虑了别的语言，比如西班牙语，并根据其特性进行解构后，再进行情感分析（Tellez, Jiménez, Graff, Moctezuma, Siordia & Villaseñor, 2017）.

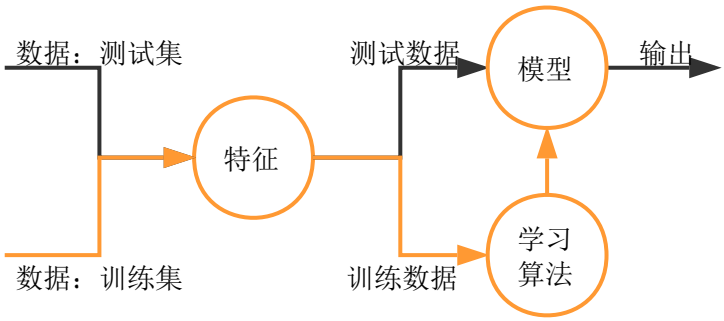
当然这些研究也暴露出了一些待解决的问题. 比如，数据不完全真实. 如今存在各式的手段和技术，使得某一话题下人们的评论并非真实可信，甚至某些账户也不真实，这会降低算法选择用户的质量（Daniel, et al. , 2017）. 还有一些可以进一步优化的地方，比如，进行推文分类时希望能利用上下文信息以及引入特征选择方法和其他优化方法的变体来提高准确率；处理反讽和讽刺还需要进一步改进（Pandey, et al. , 2017）. 还有，Tellez（2017）的实验中各种选择都是默认合适的，比如分类器选择最广泛使用的支持向量机（Support Vector Machine），库参数使用默认值，未来亦需进行调整. 还可以尝试探索转发在事件检测中的重要性，用户的文本中表达了对他人文本的偏好，该偏好值可以向前传递等（Daniel, et al. , 2017）.

3. 预处理

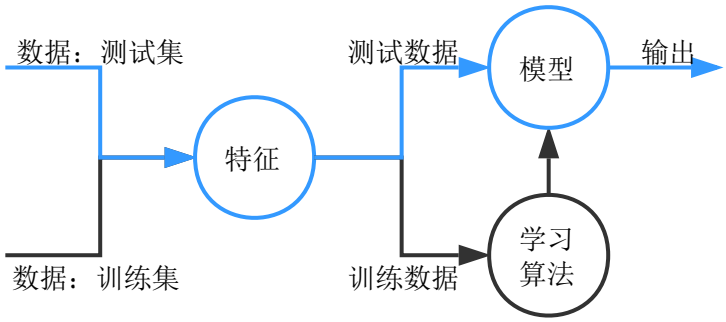
3.1. 预定义

我们本质上最终将利用机器学习来解决问题. 对于该方法而言，不同研究者所采用的术语不尽相同. 本文统一使用 Peter Flach 在 *Machine Leaning* 一书中使用的符号系统 [10] .

Peter 认为，要利用机器学习完成一项任务，需要建立从用特征描述的数据到输出的恰当映射，这个映射即模型. 学习问题的中心任务就是研究如何从训练数据中获取这样的映射. 解决学习问题之后，可以利用得到的模型解决给定任务. 运用机器学习解决给定问题的示意图如 Fig. 1.



（a）学习问题



（b）任务

Fig. 1. 机器学习解决给定问题的过程

3.2. 基于统计的分词算法

在训练之前需要对原始文本进行预处理. 因为对于英文而言，字即是词，有具体含义；但是对于中文，通常多字组词用于表情达意. 所以我们首先需要对中文进行分词.

分词算法主要分三类：基于字符串匹配的方法、基于理解的方法以及基于统计的方法. 本文利用的开源项目结巴分词［12］，就是一种基于统计的方法. 该项目是是国内程序员用 python 开发的一个中文分词模块，其原理描述如 Algorithm 1.

Algorithm 1 基于统计的分词算法

1. 基于 Trie 树结构实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图（DAG）

   e. g. 假如给定词为 {***咖啡, 咖啡豆, 兮, 中间, 中国, 中国心, 中国人民***}，那么生成的 Trie 树如 Fig. 2.

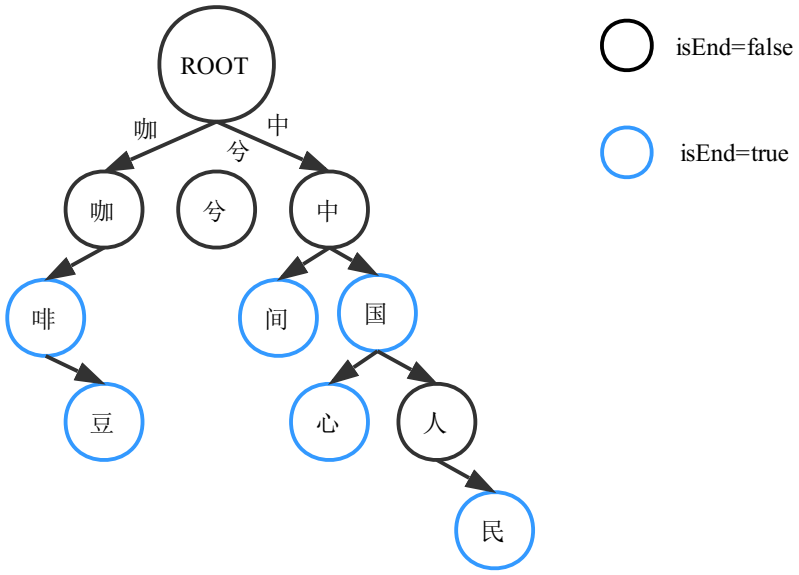


Fig. 2. 中文分词的 Trie 树结构举例

   其中，isEnd 表示该字符是否为词的尾节点.

2. 采用了动态规划查找最大概率路径, 找出基于词频的最大切分组合

   e. g. 假如给定词为 {***咖啡, 咖啡豆, 兮, 中间, 中国, 中国心, 中国人民***}，从根结点开始一次搜索，比如搜索“中间”：

   1) 取得要查找关键词的第一个字符“中”，并根据该字符选择对应的子树并转到该子树继续进行检索
   2) 在相应的子树上，取得要查找关键词的第二个字符“间”，并进一步选择对应的子树进行检索
   3) 迭代第二步，直到树节点“间”满足 isEnd=true 查找结束

3. 对于未登录词，采用了基于汉字成词能力的隐马尔科夫模型（HMM），使用了 Viterbi 算法

至此我们完成了预处理阶段，定义了本文使用的机器学习符号系统，利用结巴分词实现了原始文本的预处理.

4. 本文方法

在预定义中我们提到了机器学习解决问题基本步骤，第一步解决学习问题得到模型，如 Fig. 1. (a)；第二步利用模型解决给定任务，如 Fig. 1. (b). 从该步骤中我们可以提取出机器学习三要素：特征、模型、任务.

本质上，特征（feature）定义了一种用于描述问题中相关对象的“语言”，比如文本情感分析则是需要使用合理的特征描述给定的语句，那么之后的实验中我们可以不用过多的关注语句本身. 模型（model）被称作机器学习的输出，是为解决某个问题从数据中学习到的. 通常在学习之前我们需要选择合适的模型，在文本分析所属的自然语言处理领域内，循环神经网络（RNN）是最常用的一种模型 [9] . 任务（task）是我们期望解决问题的一种抽象表示，本文讨论的文本情感分析则可归纳为两类分类问题，是一种比较常见的任务形式.

我们已经明确了文本情感分析的任务以及模型，所以本小结我们将着重探讨如何构造特征. 特征可被视为一种易于在任意实例上度量的测度. 按照集合论的观点，特征本质上是一个从实例空间到特征空间的映射 [10] . 最常见且方便的特征空间一般为实数域. 所以我们构造特征的本质，就是求一个从文本实例到实数特征的映射.

4.1. 特征提取方法

4.1.1. 传统方法

传统的构造特征的自然语言处理方法——One-hot Encoder，通常将字词转成离散的单独的符号. 每个词对应一个向量，如 Fig.3.

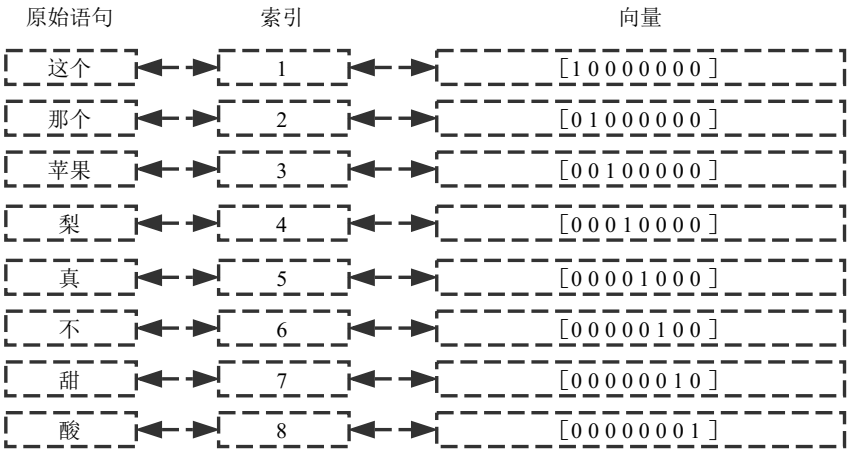


Fig. 3. One-hot Encoder 举例

以“这个苹果真甜”为例，最终整句话将组成一个稀疏矩阵：

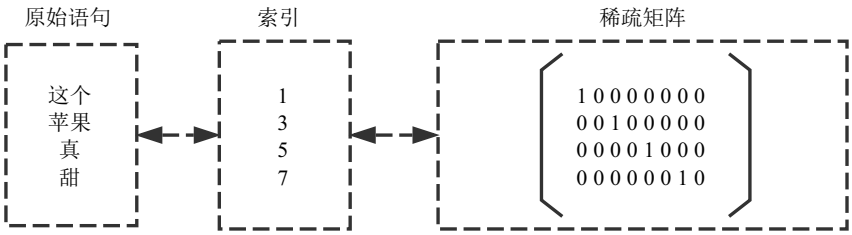


Fig. 4. One-hot Encoder 构成的稀疏矩阵举例

然而利用传统方法存储的信息量较少，比如此时我们无法从各个词对应的向量中了解到“苹果”和“梨”的相似关系. 不仅如此，稀疏矩阵意味着后期训练时需要更多数据，易导致效率低下，甚至造成造成维数灾难［9］.

4.1.2. 词嵌入

而词嵌入（Word2Vec）则是目前一个比较好的能将语言中的字词转为向量表达的预测模型，且计算高效 [9]. 该模型主要依赖于 Harris（1981）的分布假设（Distributional Hypothesis），即相同语境出现的词其语义也相近. 词嵌入分两

种模式：CBOW 和 Skip-Gram. 前者在小型数据集上表现优异，是通过原始语句预测目标语句；后者则相反. 本文是为了处理大量语料，故采用后者.

Model 2 词嵌入的 Skip-Gram 模型

还是以“这个苹果真甜”为例.

1. 构造语境与目标词汇的映射关系

   语境指目标词汇左右两边的词汇，假设滑窗尺寸为 1，那么我们需要构造的映射关系为：［这个，真］→［苹果］，［苹果，甜］→［真］

2. 构造目标语句与原始语句的映射关系作为正样本

   苹果 → 这个，苹果 → 真，真 → 苹果，真 → 甜

3. 制造随机的词汇作为负样本

4. 利用优化算法进行训练

我们利用 TensorFlow 实现 Word2Vec 训练，程序见附录?. 每一个词都得到了一个对应索引以及 50 维的向量. 至此，我们实现了一个从文本实例到实数特征的映射. 依旧以“这个苹果真甜”为例，最终整句话将形成一个稠密的向量：

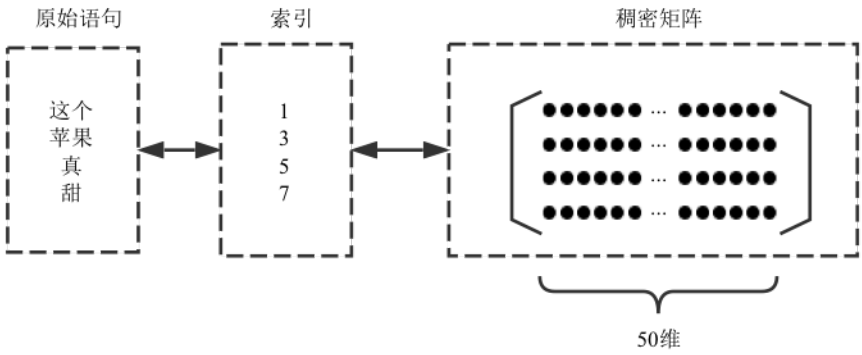


Fig. 4. Word2Vec 构成的稠密向量举例

4.1.3. 处理数据集

上文以“这个苹果真甜”为例对一个句子进行了特征提取，得到了4×50的稠密矩阵. 那么其余句子做同样处理，得到句长 $Sequence\ Length\times50$的稠密矩阵. 每一句话句长不一，为了提高训练效率，我们可以适当删减句长较大的句子. 以服饰类（*clothing*）数据集为例，不同句长的句频（*Frequency*）如 Fig.8.

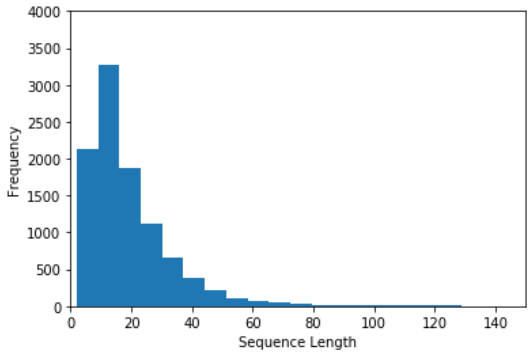


Fig. 8. 服饰类数据集不同词数的句频

从直方图中，我们可以认为将最大句长设置为 60 是可行的.

于是我们对整个数据集共 10000 句话进行同样处理，每一句话得到一个60×50的稠密矩阵，即最终得到一个$10000\times60\times50$的输入. 对于输出，每一句话的情感用［1，0］或［0，1］表示正负，则整个数据集得到$10000\times1\times2$的输出.

4.2. 基于 LSTM 的 RNN 模型

4.2.1. RNN 模型

在文本分析所属的自然语言处理领域内，循环神经网络（RNN）是最常用的一种模型. 因为文本是一种时间序列数据，所谓时间序列是指，我们需要以前的信息来理解一句话，而不是如图像分类一样，只对当前图像进行分类即可. 而 RNN 的基本结构如 Fig.5.

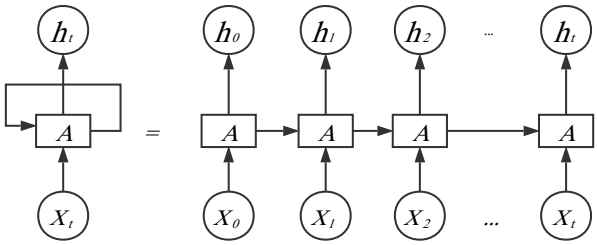


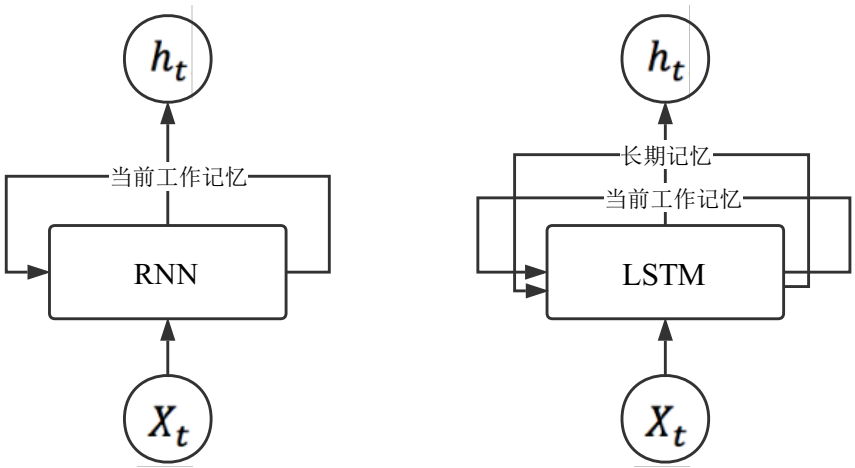
Fig. 5. 循环神经网络的基本结构展开示意图 [9]

其中，左边为 RNN 基本结构：X_t是 RNN 的输入，A是 RNN 的一个节点，h_t是输出. 右边为其展开形式，为一列串联的普通神经网络. 我们对 RNN 输入数据X_t，然后通过网络并计算得到输出结果h_t，再将某些信息传到网络的输入. 对应于右边展开式来解释会更易于理解，我们输入一系列$\{X_i\}$，通过网络并计算得到一系列输出$\{h_i\}$，传递每一层的某些信息到下一层. 这样的结构天然就非常适合时间序列的处理和分析 [9] . 值得注意的是，每一层级的神经网络参数相同，意味着我们只需训练一层 RNN 的参数即可.

我们将输出的h_t与标签（*label*）进行比较可以得到误差. 有了这个误差，我们可以利用梯度下降（Gradient Descent）和 Back Propagation Through Time（BPTT）方法对网络进行训练.

4.2.2. LSTM 组件

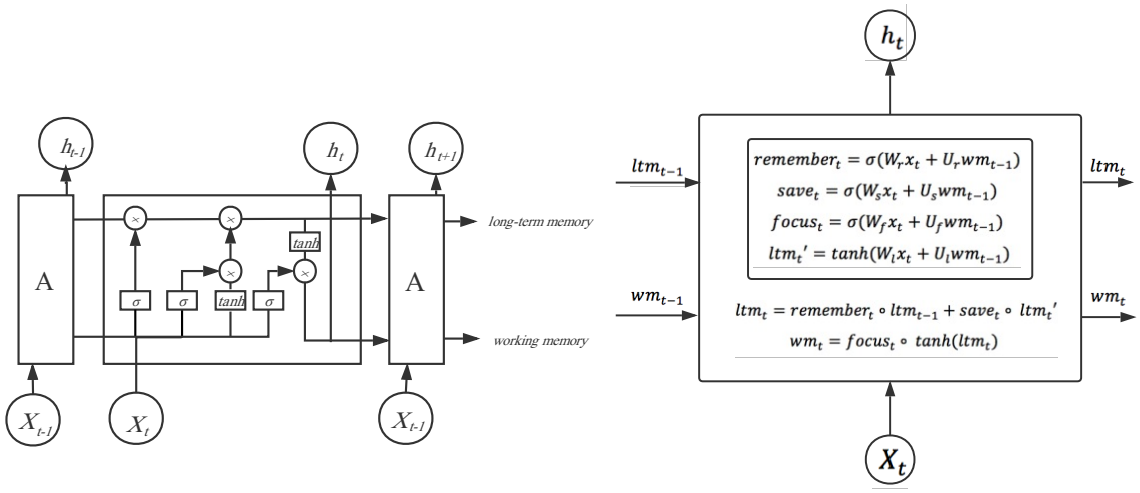
然而 RNN 存在一个问题：最后输入的信号对结果的影响较大，那么自然地，较久远的信息对结果的影响也就微乎其微. 为了解决这个问题，Jürgen Schmidhuber 和 Sepp Hochreiter 共同撰写了一篇论文，提出了一种模拟人类大脑的计算机系统——长短期记忆网络（LSTM）.



a) RNN 模型　　b) 含有 LSTM 的 RNN 模型

Fig. 6. 基础 RNN 模型与含有 LSTM 的 RNN 模型的区别 [14]

可以观察到，含 LSTM 的 RNN 模型与基础 RNN 模型的区别之处在于，含 LSTM 的 RNN 模型不仅传递了当前的工作记忆（*working memory*），还传递了之前存储的长期记忆（*long-term memory*）.其基本结构如 Fig.6.



a) 含有 LSTM 的 RNN 模型　　b) LSTM 的数学表达式

Fig. 7. 含有 LSTM 的 RNN 基本结构示意图 [9] [14]

其基本运作方式 [14] 如下：

1. 增加遗忘机制（$remember$）. 例如当一句话结束时，模型应该重置该段落的相关信息，例如主语、谓语等. 所以，我们希望模型学会一个独立的忘记/记忆机制，当有新的输入时，模型应该知道哪些信息应该丢掉.
2. 增加保存机制（*save*）. 当模型看到新的一句话时，需要学会其中是否有值得使用和保存的信息.
3. 所以当有一个新的输入时，模型首先忘掉哪些用不上的长期记忆信息，然后学习新输入有什么值得使用的信息，然后存入长期记忆中(*ltm*).
4. 把长期记忆聚焦到工作记忆中（*wm*）. 最后，模型需要学会长期记忆的哪些部分立即能派上用场. 不要一直使用完整的长期记忆，而要知道哪些部分是重点.

上文已实现了数据集的处理，所以我们依旧利用 TensorFlow 在训练集上实现含 LSTM 的 RNN 模型学习，程序见附录 1.

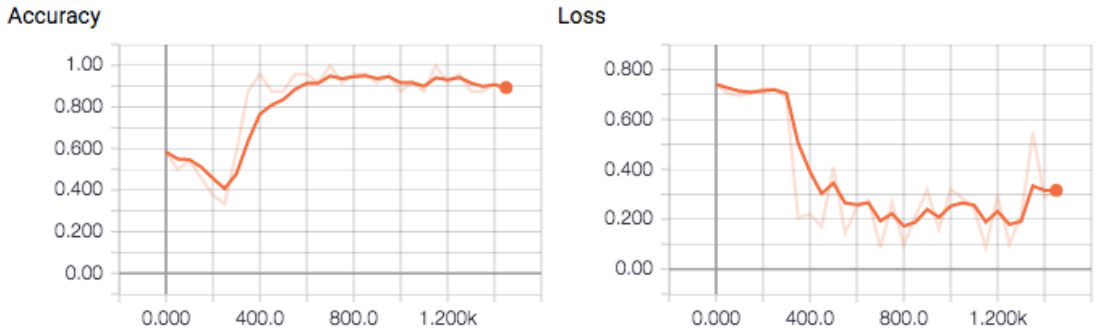
5. 实验结果

文本情感分析的具体任务是分析输入的单词或句子的情绪是积极还是消极的.那么我们利用上述学习到的含 LSTM 的 RNN 模型对测试集进行测试即可.

本文实验主要在？提供的中文数据集上进行. 数据集为商品评论，依据商品种类分为五类——服饰、水果、酒店、数码产品以及洗护用品. 每一类包含正面评价和负面评价. 本文仅服饰类、水果类数据集上进行测试.

5.1. 各个分类独立测试正确率

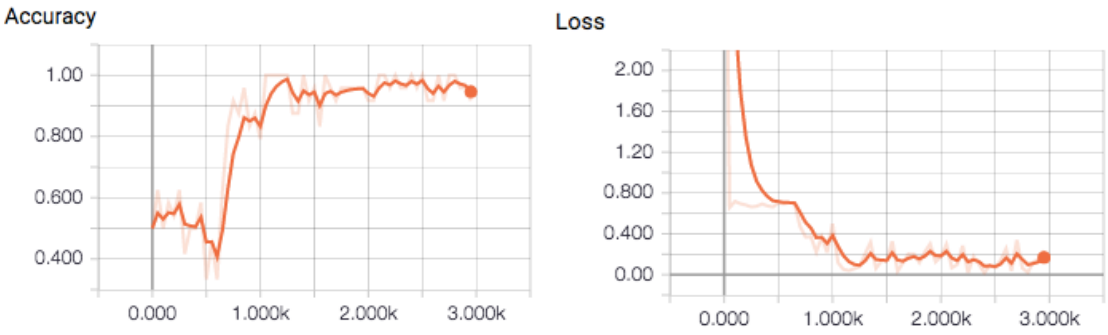
机器学习的参数设置对学习结果影响甚大，其中损失值与选择的优化器有关，学习率（*learning rate*）和网络架构有很大的关系. 其中优化器在以往的研究中没有一个一致的选择，不过 Adam 优化器被广泛的使用. 而学习率对 RNN 训练的意义重大，因为其时间步骤很长. 如果我们将学习率设置的很大，那么学习曲线就会波动性很大，如果我们将学习率设置的很小，那么训练过程就会非常缓慢. 所以我们分别尝试将学习率设置为 0.0001、0.001、0.01 进行训练，得到的训练过程如 Fig.8. 从图中我们可以看出学习率为 0.01 时学习曲线较稳定，准确率不断地在接近 1，损失函数也在稳定地下降. 在训练 1400 次左右以后准确率下降，说明此时我们的网络性能开始退化了，为了防止过拟合现象，我们需要提前终止训练. 最后我们在测试集上进行测试，本文随机选取 90 条语句，得到最终平均准确率.



a) 学习率（*learning rate*）为 0.0001 时的准确率（*accuracy*）和损失函数（*loss*）

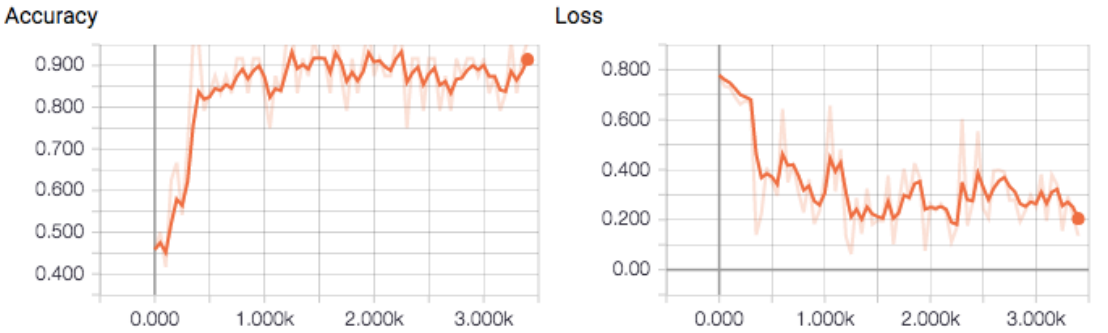
b) 学习率（*learning rate*）为 0.001 时的准确率（*accuracy*）和损失函数（*loss*）

Fig. 8. 服饰类数据集训练过程

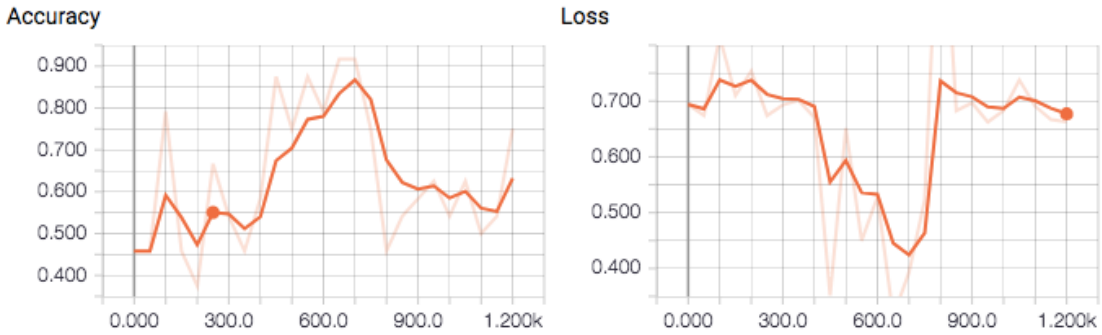


c) 学习率（*learning rate*）为 0.01 时的准确率（*accuracy*）和损失函数（*loss*）

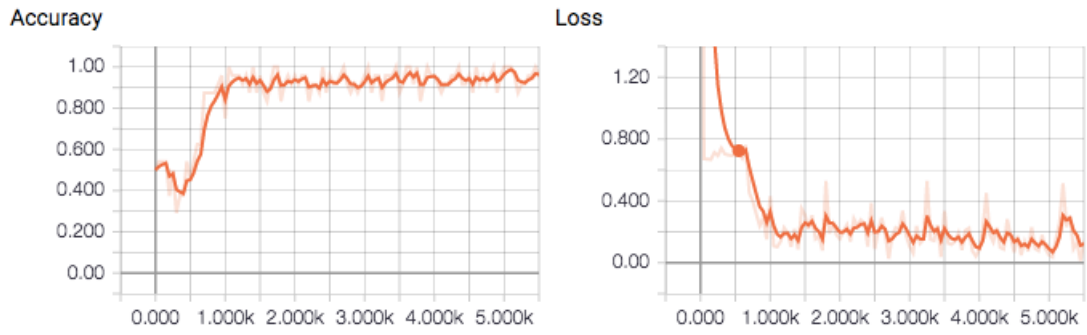
Fig. 8. 服饰类数据集训练过程



a) 学习率（*learning rate*）为 0.0001 时的准确率（*accuracy*）和损失函数（*loss*）



b) 学习率（*learning rate*）为 0.001 时的准确率（*accuracy*）和损失函数（*loss*）



c) 学习率（*learning rate*）为 0.01 时的准确率（*accuracy*）和损失函数（*loss*）

Fig. 9. 水果类数据集训练过程

每一次训练均记录损失函数下降较少或不下降前的步数 *step* 以及此时的准确率 *train_accuracy*，在测试集上测试得到准确率 *test_accuracy*. 该值直观显示了我们学习到的网络模型的性能优劣. 结果如 Tab.1.和 Tab.2.所示.

Tab.1. 服饰类数据集不同学习率下的训练结果

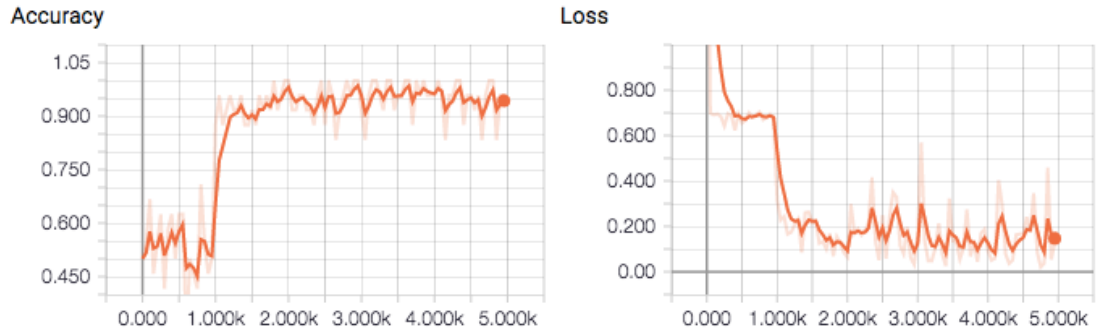
learning rate	*step*	*train_accuracy*	*test_accuracy*
0.0001	*700*	*0.96*	*0.92*
0.001	*1500*	*0.96*	*0.91*
0.01	*2250*	*0.98*	*0.93*

Tab.2. 水果类数据集不同学习率下的训练结果

learning rate	*step*	*train_accuracy*	*test_accuracy*
0.0001	*950*	*0.91*	*0.88*
0.001	*700*	*0.92*	*0.89*
0.01	*1200*	*0.94*	*0.90*

5.2. 不同分类混合测试正确率

我们发现文本在各自数据集上准确率普遍较高，为 0.9 左右. 那么我们进一步测试混合文本，来评估网络的泛化能力. 我们将服饰类与水果类数据集混合得到混合数据集，学习率设置为默认的 0.01. 训练过程如 Fig.10.



测试，迭代 90 次后平均准确率为 0.92. 很明显，数据集混合后的准确集较混合前持平，网络表现依旧优异.

我们测试混合数据集的本意是为了进一步优化此网络，然而当我们发现混合数据集表现依旧优异，说明前期进行文本分类对于此网络后期优化意义不大.

5.3. 测试反讽

讽刺作为人与人交流时非常微妙的一种表达，最能体现人与机器之间的区别. 我们利用之前训练得到的网络，在反讽数据集上进行测试. 结果如 Tab.3.

Tab.3. 反讽语句测试结果

Sentence	$P_{label=[0,1]}$
日常用语	
原定于今天的春游因为下雨而取消了，小东生气地说：“全是因为今天的好天气!”	0.0000
我把一个杯子打碎了，妈妈说：“看你干的好事”	0.0042
大家都说好，我也就呵呵了	0.0333
模拟商品评论	
我买的鞋子一年就邮寄到了，真快	1.0000
这么贵的鞋子，做工却很粗糙，服气	0.0000
我除了羡慕卖家可以卖假货赚这么多钱之外，不敢赞一词	0.0000
我的腿看起来这么粗，真是拜这条好裤子所赐呀	0.0000
我买的这款书包，除了不能使用，真是挑不出一点缺点	0.0000
文学作品	
我们法国人是文明人，中国人是野蛮人，这就是文明对野蛮所做的事情！这次的抢劫是多么漂亮，多么彻底！	0.0000
也有解散辫子，盘得平的，除下帽来，油光可鉴，宛如小姑娘的发髻一般，还要将脖子扭几扭。实在标致极了	0.9958

其中$P_{label=[0,1]}$表示原句经过网络计算后情感为负性的概率，该概率越高，说明情感越为负，反之，情感越为正。

上述文本揭示利用反讽，表达了负性情感，但是测试结果不尽如人意。大部分句子经过网络计算后被判定为正性情感，而且少部分被判定为负性的句子也难提取出规律。这可以是未来研究的一个方向。

6. 结论

本文结合了中文的特色，利用基于统计的分词算法进行研究。抛弃传统词典方式，引入深度学习方法——词嵌入，将语言中的字词转为向量表达。处理数据集时设置了最大句长，以便于提高后期训练效率。利用含 LSTM 的 RNN 模

型，结合了上下文信息，成功将准确率提高到 0.9。深度思考后提出进一步优化的可能性——文本分类、测试反讽。然而测试混合数据集后发现文本分类意义不大；在反讽数据上网络表现不佳，未来会进一步优化，可考虑增加中性这一情感标签，优化我们的网络.

参考文献

[1] 黄婷婷. Web3. 0 时代下协同创新营销的研究与分析[J]. 商,2014(8):54-54.

[2] 王靓. 浅谈 Web3. 0 互联网营销模式[J],科技成果纵横,2008(01):1005-5089.

[3] 圣卡西. 新浪微博 2017 年 Q1 财报:月活跃用户达 3. 4 亿,世界第一![EB/OL]. http://www. yxdown. com/news/201705/352742. html,2017-05-17.

[4] Eric S. Tellez, Sabino Miranda-Jiménez, Mario Graff, Daniela Moctezuma, Oscar S. Siordia, Elio A. Villaseñor. A case study of Spanish text transformations for twitter sentiment analysis [J]. Elsevier Ltd, 2017.

[5] Mariana Daniel, Rui Ferreira Neves, Nuno Horta. Company event popularity for financial markets using Twitter and sentiment analysis[J]. Elsevier Ltd, 2016.

[6] Avinash Chandra Pandey, Dharmveer Singh Rajpoot, Mukesh Saraswat. Twitter sentiment analysis using hybrid cuckoo search method[J]. Pergamon Press, Inc., 2017 , 53 (4) :764-779

[7] Teresa Alsinet, Josep Argelich, Ramón Béjar, Cèsar Fernández, Carles Mateu, Jordi Planes. Weighted argumentation for analysis of discussions in Twitter[J]. International Journal of Approximate Reasoning, 2017, 85 :21-35

[8] Adit Deshpande. Perform sentiment analysis with LSTMs, using TensorFlow[EB/OL]. https://www. oreilly. com/learning/perform-sentiment-analysis-with-lstms-using-tensorflow, 2017-07-13.

[9]黄文坚，唐源. TensorFlow 实战[M]. 北京：电子工业出版社，2017：159-194

[10] Peter Flach. 机器学习[M]. 北京：人民邮电出版社，2016：1-33

[11] JohnSon0722. 中文分词的基本原理以及 jieba 分词的用法[EB/OL]. http://blog. csdn. net/john_xyz/article/details/54645527, 2017-01-21.

[12] fxsjy. Python 中文分词组件 jieba[EB/OL]. https://www. oschina. net/p/jieba?fromerr=ex7thahF, 2012-10-03.

[13] ZS Harris. Distributional structure[M]. Springer Netherlands, 1981: 146–162.

[14] 量子位. 探索 LSTM：基本概念到内部结构[EB/OL]. https://zhuanlan.zhihu.com/p/27345523, 2017-06-11.

[15] Adit Deshpande. Perform sentiment analysis with LSTMs, using TensorFlow[EB/OL]. https://www.oreilly.com/learning/perform-sentiment-analysis-with-lstms-using-tensorflow, 2017-07-13.
[16]光与热的霓虹. 中文情感分析语料库[EB/OL]. http://download.csdn.net/download/u010097581/9919245, 2017-08-02.

附录

附录 1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

索引矩阵：
[[    8     7   106 ...,     0     0     0]
 [   10     8     3 ...,     0     0     0]
 [   52     3     7 ...,     0     0     0]
 ...,
 [   30 1720   157 ...,     0     0     0]
 [ 166    68     1 ...,     0     0     0]
 [   92    19    15 ...,     0     0     0]]
"""

import jieba

# 分词
filePath='/Users/xuyizhou/Desktop/corpus.txt'
fileSegWordDonePath ='/Users/xuyizhou/Desktop/corpusSegDone.txt'
# read the file by line
fileTrainRead = []
#fileTestRead = []
with open(filePath) as fileTrainRaw:
    for line in fileTrainRaw:
        fileTrainRead.append(line)
# define this function to print a list with Chinese
```

```
def PrintListChinese(list):
    for i in range(len(list)):
        print(list[i]),
# segment word with jieba
fileTrainSeg=[]
for i in range(len(fileTrainRead)):
    fileTrainSeg.append([' '.join(list(jieba.cut(fileTrainRead[i][:],cut_all=False)))])
    if i % 1000 == 0 :
        print(i)

# to test the segment result
#PrintListChinese(fileTrainSeg[10])

# save the result
with open(fileSegWordDonePath,'w') as fW:
    for i in range(len(fileTrainSeg)):
        fW.write(str(fileTrainSeg[i][0]))
# 转换成词向量
import word2vec
word2vec.word2vec('/Users/xuyizhou/Desktop/corpusSegDone.txt',
'/Users/xuyizhou/Desktop/corpusWord2Vec.bin', size=300,verbose=True)
model = word2vec.load('/Users/xuyizhou/Desktop/corpusWord2Vec.bin')
print (model.vectors)

#9000 条评论 每条 140 个词 每个词一个索引 一个索引对应一行矩阵

import tensorflow as tf
import numpy as np
m=len(fileTrainSeg)
n=60
firstSen=np.zeros((m,n),dtype='int32')
#for j in range(len(fileTrainSeg)):
for j in range(m):
```

```
        print(j)
        #如果某评论词数小于 60
        if len(fileTrainSeg[j][0].split())<=60:
            for i in range(len(fileTrainSeg[j][0].split())):
                index=np.where(model.vocab==fileTrainSeg[j][0].split()[i])[0]


                if index>=0:
                    if index<=2545:
                        firstSen[j][i]=int(index)
                else:
                    firstSen[j][i]=0

        else:
        #如果某评论词数大于 60
            for i in range(60):
                index=np.where(model.vocab==fileTrainSeg[j][0].split()[i])[0]


                if index>=0:
                    if index<=2545:
                        firstSen[j][i]=int(index)
                else:
                    firstSen[j][i]=0

print("索引矩阵：")
print(firstSen)
#得到每一行的词语数量
numWords=[]
for i in range(len(fileTrainSeg)):
    counter = len(fileTrainSeg[i][0].split())
    numWords.append(counter)
```

```
print('The total number of words in the files is', sum(numWords))

#下面你可以找到几个辅助函数，这些函数在稍后训练神经网络的步骤中会使用到。
from random import randint

def getTrainBatch():
    labels = []
    arr = np.zeros([batchSize, maxSeqLength])
    for i in range(batchSize):
        if (i % 2 == 0):
            num = randint(1,4000)
            labels.append([1,0])
        else:
            num = randint(6001,10000)
            labels.append([0,1])
        ids=firstSen
        arr[i] = firstSen[num-1:num]
    return arr, labels

def getTestBatch():
    labels = []
    arr = np.zeros([batchSize, maxSeqLength])
    for i in range(batchSize):
        num = randint(4001,6000)
        if (num <= 5001):
            labels.append([1,0])
        else:
            labels.append([0,1])
        arr[i] = firstSen[num-1:num]
    return arr, labels
#RNN 模型
batchSize = 24
```

```
lstmUnits = 64
numClasses = 2
iterations = 100000

#LSTM 单元的数量
maxSeqLength=60
#词向量维度
numDimensions=50

import tensorflow as tf
tf.reset_default_graph()

#标签占位符
labels = tf.placeholder(tf.float32, [batchSize, numClasses])
#输入占位符
input_data = tf.placeholder(tf.int32, [batchSize, maxSeqLength])

data = tf.Variable(tf.zeros([batchSize, maxSeqLength,
numDimensions]),dtype=tf.float32)
data = tf.nn.embedding_lookup(model.vectors,input_data)

lstmCell = tf.contrib.rnn.BasicLSTMCell(lstmUnits)
lstmCell = tf.contrib.rnn.DropoutWrapper(cell=lstmCell, output_keep_prob=0.75)
value, _ = tf.nn.dynamic_rnn(lstmCell, data, dtype=tf.float64)

"""
dynamic RNN 函数的第一个输出可以被认为是最后的隐藏状态向量。这个向量
将被重新确定维度，
然后乘以最后的权重矩阵和一个偏置项来获得最终的输出值。
"""
weight = tf.Variable(tf.truncated_normal([lstmUnits, numClasses]),dtype=tf.float32)
bias = tf.Variable(tf.constant(0.1, shape=[numClasses]))
value = tf.transpose(value, [1, 0, 2])
```

```
last = tf.gather(value, int(value.get_shape()[0]) - 1)

last = tf.cast(last, tf.float32)#bug1 thx!

prediction = (tf.matmul(last, weight) + bias)
correctPred = tf.equal(tf.argmax(prediction,1), tf.argmax(labels,1))
accuracy = tf.reduce_mean(tf.cast(correctPred, tf.float32))
```