

目 录

第一章 绪 论	1
1.1 研究工作的背景与意义	1
1.2 国内外研究现状	2
1.3 课题主要内容	3
1.4 本论文的结构安排	3
第二章 相关技术研究	4
2.1 中文文本分词	4
2.1.1 基于字符串匹配的分词算法	4
2.1.2 基于统计的分词算法	6
2.2 传统文本表示方法	7
2.2.1 向量空间模型	7
2.2.2 TF-IDF 特征提取	8
2.3 传统文本分类方法	8
2.3.1 贝叶斯分类器	8
2.3.2 支持向量机	9
2.4 基于神经网络的短文本分类方法	10
2.4.1 短文本的分布式表示方法	10
2.4.2 卷积神经网络	13
2.4.3 循环神经网络	15
2.5 本章小结	17
第三章 中文文本表示方法的研究和改进	19
3.1 汉字偏旁部首的语义信息	19
3.2 中文文本表示方法	20
3.3 部首信息增强的中文词向量构建方法	21
3.3.1 文本预处理	21
3.3.2 部首信息增强的中文词向量模型	22
3.4 部首信息增强的中文字向量构建方法	25
3.5 实验及其结果分析	25
3.5.1 部首信息增强的中文词向量模型实验对比	26
3.5.2 部首信息增强的中文字向量模型实验对比	30

3.6 本章小结	30
第四章 Attention-Based 字词结合卷积循环中文短文本分类模型	31
4.1 卷积循环神经特征提取网络	31
4.1.1 卷积循环神经网络整体结构	31
4.1.2 卷积网络层	31
4.1.3 循环网络层	33
4.2 Attention Model	33
4.3 Attention-Based 字词结合卷积循环中文短文本分类网络	35
4.3.1 Attention-Based 卷积循环特征提取网络	35
4.3.2 双通道字词结合短文本分类模型	36
4.4 实验设计和结果分析	38
4.4.1 实验语料数据	38
4.4.2 实验环境及相关工具	38
4.4.3 分类模型评价方法	39
4.4.4 对比实验设计	40
4.4.5 实验结果和分析	41
4.5 本章小结	42
第五章 短文本分类系统设计和实现	43
5.1 系统总体概述	43
5.2 系统各模块的设计和实现	44
5.2.1 网络爬虫	45
5.2.2 队列管理	48
5.2.3 语料数据存储	48
5.2.4 字/词向量训练	50
5.2.5 模型训练	53
5.2.6 分类服务	55
5.2.7 日志记录	56
5.3 系统测试与分析	57
5.3.1 新闻标题数据搜索	57
5.3.2 汉字信息库	57
5.3.3 分类测试结果	57
5.4 本章小节	57
参考文献	58

第一章 绪论

1.1 研究工作的背景与意义

随着信息化进程的不断推进，我国互联网市场空前繁荣。从 WEB2.0 时代开始，到如今的“互联网+”时代，互联网已经融入了人们的生活中，也对各行各业产生了深远的影响。根据中国互联网络信息中心发布的《第 40 次中国互联网络发展状况统计报告^①》显示，截止到 2017 年 6 月，我国网民规模达到 7.51 亿，其中手机网民更是达到 7.24 亿，占总体网民的 96.3%，使用率排名前三的互联网应用分别是即时通信（92.1%）、网络新闻（83.1%）、搜索引擎（81.1%），使用率最高的三个 app 应用则是微信（84.3%）、QQ（65.8%）、微博（38.7%）。可以明显看出，人们使用互联网以及获取信息的方式，已经从以前的桌面台式电脑，转变为移动端的手机、IPAD 等掌上设备。

移动设备的盛行，使得人们发布和接收信息都更加方便，据统计，在 2012 年，微博用户已经增长到 3.68 亿，其中 69% 通过移动设备登陆，每天能产生 1.17 亿条微博。手机用户的大量增加，让互联网信息爆炸式增长，并且产生了大量碎片化的信息，如微博、QQ 说说、留言、商品评论等。据有关部门统计，互联网全体文本信息中，80% 以上属于内容较少的短文本信息。

如果能够有效分析这些短文本，对其进行精确的分类，可以方便用户有效的梳理这些浩如烟海的文本并掌握对自身有用的信息，商家也能够根据信息的分类提供更加优质的服务。例如，内容提供商，如新浪微博、知乎等，可以对其提供的信息进行分类，让用户快速获取自己感兴趣的某一类信息，同时商家也可以统计用户浏览过的信息类，精确投放用户感兴趣的广告，减少无关广告对用户体验的伤害；政府相关部门可以根据一段时间内大众发布的微博或朋友圈等信息，掌握当前的热门话题、集中关注点，监控当前的社会舆情；电子商务平台，如淘宝、京东等，可以利用情感分类技术，提取商品的正面评论与负面评论给用户，让用户能够更好的筛选与判断优质商品。

但是和传统文章相比，短文本过于短小（通常在 100 字以内，一般是一句话的长度），不能提供足够的词共现（word co-occurrence）或上下文，以至于很难从中提取出有效的文本特征^[1]。因此，常规机器学习技术与文本分类算法很难直接应用在短文本之上。那么如何准确高效的对短文本进行分类，成为了互联网从业者与互联网技术学者所面临的一个重要难题，其突破也会具有重要的商业价值与

^① <http://www.cnnic.net.cn/hlwfzyj/hlwxxzb/hlwtjbg/201708/P020170807351923262153.pdf>

使用价值。

1.2 国内外研究现状

随着互联网中短文本信息的增多，短文本分类领域得到了广泛的关注，越来越多的学者投入到短文本分类的研究之中。但由于短文本短小、信息分散的特性，传统基于词频、词共现的分类方法通常得不到较好的效果，比如贝叶斯方法 (Naive Bayes)、最大熵模型 (Maximum Entropy Model)、K-邻近算法 (K Nearest Neighbors) 以及支持向量机 (Support Vector Machines, SVMs)。因此学者们开始尝试从其他方面来改进短文本分类算法，比如语义分析 (semantic analysis)、半监督 (semi-supervised) 算法和集中模型 (ensemble models)。

在语义分析方面，纽约大学的 Sarah Zelikovitz 等人^[2]通过隐含语义索引算法 (Latent Semantic Index, LSI) 对短文本的语义分析，将文本中的词映射到潜在语义空间，来捕获文本单词之间的相关性，提升分类效果；清华大学的 Chen 等人^[3]通过改进的隐含狄利克雷分布 (Latent Dirichlet allocation, LDA) 模型，将短文本中的单词与多个粒度的话题相关联，从而拓展短文本特征。

在半监督学习方面，Juan Manuel Cabrera 等人^[4]根据分布式词语表示算法 (Distributional Term Representations, DTRs)，用半监督的方式统计语料中的文档出现特征以及词语共现信息，形成每个词语的上下文信息，最后强化文本表示，以此来克服短文本处理中长度过短、信息高度分散的难点。

中国科学院自动化研究所的冯晓等人^[5]则构造了一种集中学习模型，直接确立短文本实例与某一主题直接的相关性，而不是将短文本表示为权重向量，取得了超过向量空间模型 (Vector Space Model, VSM) 的效果。

随着神经网络与深度学习逐渐兴起，并在计算机视觉、语音识别等领域取得了不错的成果，越来越多的学者注意到深度学习模型对于特征提取与数据建模上的优势，开始尝试在自然语言处理问题中引入。而通过神经网络提取出的文本特征向量，可以直接用于其他任务，例如输入传统分类器进行分类。斯坦福大学的 Richard Socher 等人^[6]构造了一个使用矩阵向量的循环神经网络 (MV-RNN)，从长度不一致的句子中学习语义信息，形式长度统一的特征向量，最后放入分类器分类，取得了超过传统文本分类方法的结果；牛津大学的 Nal Kalchbrenner 等人^[7]提出了动态卷积神经网络 (DCNN)，利用动态 K-max 池化的方法，直接获取文本中单词直接的距离关系，避免了对语法分析树的依赖；哈佛大学的 Yoon Kim 等人^[8]将动态词向量与预训练好的静态词向量相结合作为同一段文本的两个表示，输入卷积神经网络的两个通道中进行分类，也取得的较好的效果。

但是，随着深度学习使用的逐渐扩大，学者们发现，对于语法复杂、需要分词的中文文本，深度学习模型并不能直接应用，也无法取得英文语料一样的优秀效果。因此，国内学者开始探寻适合中文文本的深度学习模型，也获得了很多不错的成果。

中国科学院自动化研究所的来斯惟等人^[9]设计了一个循环卷积神经网络，在中文长文本分类任务中取得了较好的效果。纽约大学的张翔等人^[10]实现了一个字符级别的卷积网络（ConvNets），将中文语句转化为汉语拼音，对中文语料进行分类。北京大学的李嫣然等人^[11]通过在文本表示中引入部首信息，利用汉字中部首也包含一定语义信息的特点，在连续词袋模型（Continuous Bag-of-Words, CBOW）的基础上，构造出了一个新的字向量模型，在中文新闻标题分类上获得了明显的效果。

近几年，自然语言处理领域不断发展，一些新的方法也相继出现，2017年，Google公司开创性的提出注意力（Attention）观点，认为模型的结果并不是和每一个模型提取出的特征都有密切的关系，往往一个结果只是由某一个或某几个关键特征所决定的。这给了学者们新的启示，一些学者于是将注意力观点应用在文本分类之中。在文献[12]中，南洋理工大学的 Meng Joo Er 等人开发的基于注意力池化的卷积神经网络，利用平行的双向长短期记忆网络构造了一个输入文本的中间向量表示，以此作为卷积神经网络生成的文件特征向量的注意力权重，最后将经过处理后的文本特征向量输入分类器进行分类；卡耐基梅隆大学的 Zichao Yang 等人^[13]则实现了一个分层注意力网络，整个网络由两层循环神经网络组成，第一层网络对句子进行建模，经过注意力机制处理之后得到句子的向量表示，然后第二层根据所有的句子向量得到文章的向量表示，最后根据这个文章向量获得分类结果。

尽管基于注意力机制的文本分类在近几年获得了研究人员的关注并得到了很多成果，但是这些成果大多是基于英文语料的，如何将该方法推广到其他语言中，并且如何通过注意力机制提升中文文本分类，特别是中文短文本的分类依然是研究人员需要继续的课题。

1.3 课题主要内容

1.4 本论文的结构安排

第二章 相关技术研究

随着互联网技术的发展与国民生活水平的提供,手机持有率也直线上升,这使得手机网民大规模的增长,人们在网上产生的信息、发布的文本日益碎片化。为了有效利用这些片段化的信息,短文本分类技术,作为自然语言处理的关键技术之一,也得到了充分的关注与发展,短文本分类应用的影子在互联网中随处可见。本章将对传统文本分类技术进行论述,并介绍本文涉及到的相关技术问题。

2.1 中文文本分词

词是最小的能够独立活动的有意义的语言成分,在英文文本中,每个单词天然的由空格分割开来,使得研究人员可以毫无难度的获取文本中所有的单词,然后进行接下来的研究工作。但对于中文这样由字组成的连续文本,不存在这样的语言优势,词语直接没有明确的区分标记,要想进行语义分析,必须先将中文文本切分,因此中文分词是中文信息处理的基础和关键。同时,大量实验表明,分词的好坏直接关系着后续分类算法的最终效果,所以选择一个快速并准确的分词算法尤为重要。目前常用的分词方法主要有两大类:基于字符串匹配的算法,基于规则的算法。

2.1.1 基于字符串匹配的分词算法

基于字符串匹配的分词算法又称为机械分词算法,主要依据外部提供的词典,按照一定的策略将待切分的中文文本与词典中的词条逐一匹配,若在词典中找到该词条,则匹配成功,否则做其它相应的处理。查找词库的匹配策略分为两种:长单词优先的最大匹配法以及短单词优先的最小匹配法。在实际使用中,人们发现单词切分次数越短,切分出的单词越长,分词效果越好,所以目前一般使用最大匹配法。按照匹配顺序的不同,最大匹配算法又可分为:正向最大匹配算法^[14]、逆向最大匹配算法^[15]、双向最大匹配算法^[16]。

(1) 正向最大匹配算法

正向最大匹配算法的基本思想是:已知词典中最长单词的长度为 N ,则以 N 最为截取单词初始长度。对于带切分的中文文本 S ,首先从左向右截取长度为 N 的字符串 W_1 ,然后在词典中寻找是否有和该字符串 W_1 匹配的单词。如果找到,则将 W_1 标记为成功切分出的单词,再继续从待切分文本的 $N+1$ 处开始下一次匹配;如果没找到,则将截取长度减1重新截取,即在 S 的原来位置重新截取长度为

$N-1$ 的字符串，然后重复之前匹配操作，直到截取长度为 1。算法流程如图2-1所示：

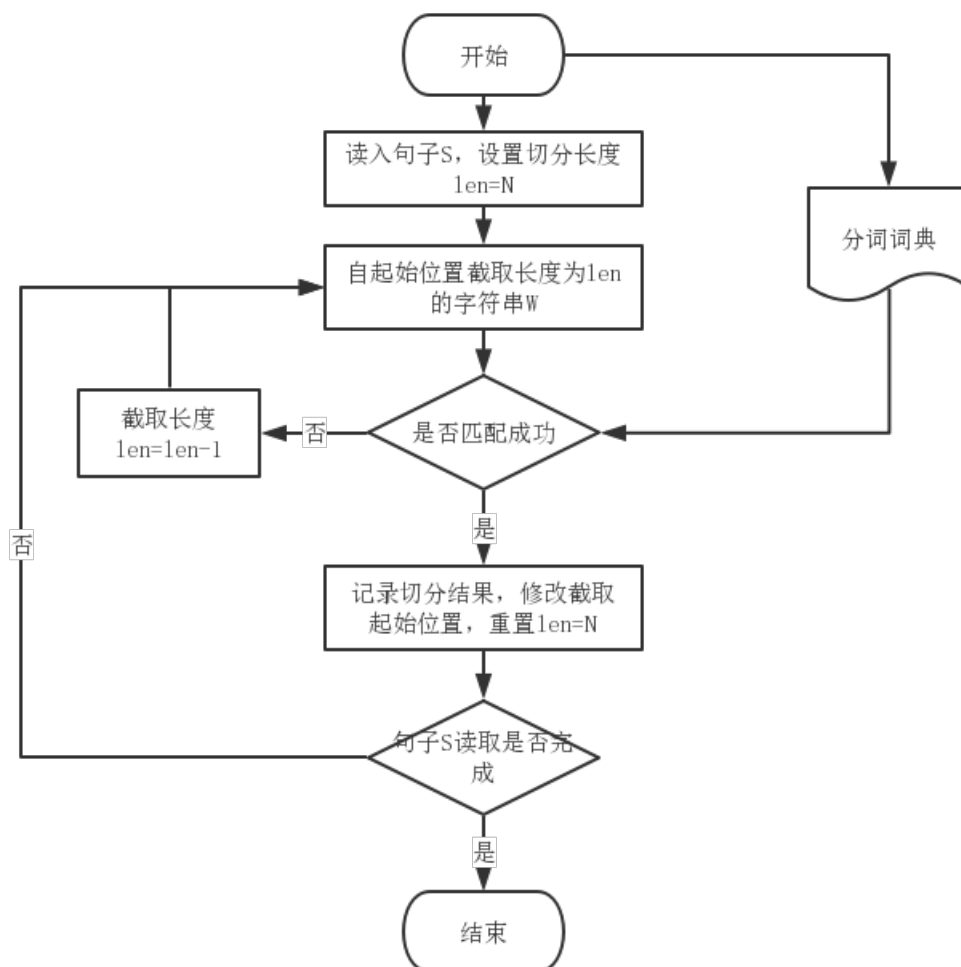


图 2-1 正向最大匹配算法流程

(2) 逆向最大匹配算法

逆向最大匹配算法思路和正向最大匹配算法大致相同，不同之处在于截取字符串时的方向由从左向右换成了从右向左。也就是说，当对句子分词时，根据词典中最长单词的长度，从句子末尾开始向左截取字符串与词典中的单词匹配，直到切分到句子的开始位置为止。

(3) 双向最大匹配算法

双向最大匹配算法是上述两种最大匹配算法的结合，侧重于分词过程中的检错和纠错，其基本思路是对待分词文本分别采用正向最大匹配和逆向最大匹配进行初步切分，然后将得到的正向分词结果和逆向分词结果进行比较，如果两种方

法的结果一致，则认为分词结果正确，如果结果存在出入，则认为分词存在着分错误，需要采用其他技术手段消除结果中的歧义。

从上面的分析可以明显看出，无论哪种最大匹配算法都极度依赖词典，只有在词典覆盖的领域内，才有较理想的分词效果。对与词典没有覆盖的陌生领域语料，极端情况下甚至会出现单字切分的分词结果。

2.1.2 基于统计的分词算法

这类分词算法并不依赖具体的词典，而是根据语料中的统计信息，识别句子中的单词。即把单词看做是特定的字的结合，在语料中邻近的字共同出现的次数越多越可能是一个词。所以计算句子中特定字的组合的出现概率，可以判断这个组合是否是一个词。通过以概率论为理论基础，将中文文本中的每一个词的出现抽象成随机过程，分词算法不会被待分词文本的内容所影响，对所有领域的中文语料都有统一的效果，这是极度依赖词典的基于字符串匹配的分词算法所没有优势。根据采用的统计模型不同，基于统计的分词算法又可分为互信息算法、N 元统计模型等。

(1) 互信息算法

在概率论和信息论中，两个随机变量的互信息是这两个变量彼此之间依赖性的一个度量。更确切的说，它是根据另一个随机变量来量化从一个随机变量中可以获得的信息量。互信息分词算法是互信息理论在分词中的应用，通过计算两个相邻字符串的互信息值，判断它们之间的结合程度。

对于两个相邻字符串 x 和 y ，它们的互信息值计算公式如下：

$$I(x, y) = \log \frac{p(x, y)}{p(x)p(y)} \quad (2-1)$$

其中 $p(x, y)$ 表示字符串 x 和 y 在语料中共同出现的频率， $p(x)$ 与 $p(y)$ 分别表示字符串 x 与 y 的出现频率。当 $I(x, y) > 0$ 时，表示 x 和 y 具有一定的相关性，并且这个值越大，它们联系的就越紧密，超过某一个阈值时即可判定为一个词；当 $I(x, y) = 0$ 时，表示 x 和 y 的关系不明确；当 $I(x, y) < 0$ 时，表示 x 和 y 直接几乎没有相关性，基本不会组成一个词。

(2) N 元统计模型

N 元统计模型又称为 N 元语言模型 (n-gram language model)，本质上是对语言建模的一种统计模型。该模型假定语言满足马尔科夫性，句子中的单词的出现与其前面出现的单词紧密相关，即第 n 个词的出现只与前面 $n - 1$ 个词的出现相关，而和其他任何词都不相关。假设句子 S 由单词序列 (w_1, w_2, \dots, w_m) 组成，则 N

元语言模型可表示为：

$$\begin{aligned} P(S) &= P(w_1 w_2 \dots w_m) \\ &= P(w_1) P(w_2 | w_1) \dots P(w_i | w_{i-n+1} \dots w_{i-1}) \dots P(w_m | w_{m-n+1} \dots w_{m-1}) \end{aligned} \quad (2-2)$$

理论上来说， N 取值越大，模型就越精确，越能揭示出语言的内在结构。但随着 N 的增加，模型的计算复杂度也呈几何式上升，所以在实际应用中，通常将 N 取值为 2、3、4，而 N 取 2 的 N 元统计模型称为 bigram 模型， N 取 3 的则称为 trigram 模型。

在分词应用中，算法首先对句子 S 进行全切分，得到若干分词结果，再根据公式 2-2 计算这些分词结果的概率 $P(S)$ ，最后选择概率最高的分词结果作为最终结果。

2.2 传统文本表示方法

分词处理之后，文本信息转化为一个单词序列，但对于计算机与程序来说依然是一段没有意义的字符串。为了让程序能够理解文本信息，继续之后的分类工作，我们需要对其再次处理，提取出其中的有效信息，将字符串文本映射成结构化的数字信息。

2.2.1 向量空间模型

向量空间模型（Vector Space Model, VSM）最早由 Salton 等人^[17]于 20 世纪 70 年代提出，是一种将文本信息表示为向量的代数模型。模型的主要思想是将文本看做由单词的简单组合，通过统计语料中不同单词的个数，构建一个 n 维的向量，向量中每一个纬度都代表一个不同的单词，单词在文本中存在，则此位为 1，否则为 0，以此将一段文本信息转换为一个 n 维的数学向量。

可以明显看出，虽然向量空间模型建立了一个从文本到向量的快速转换，让程序能够容易地对文本进行处理计算，但这种映射方式太过简单。将文本表示为单词的组合，忽略了词语的位置关系以及词语之间的相互联系，而将相同单词都统计为一类，也忽略了一词多义的情形，让程序难以进行进一步的语义分析。而且，当统计的语料足够多后，模型产生的向量会拥有一个巨大的维度，造成后续计算的维度灾难。

为了解决上述问题，实际使用中，通常使用文本的关键词作为文本向量，而不是使用所有单词。因此，如何选择关键词就变得尤为重要。

2.2.2 TF-IDF 特征提取

TF-IDF (term frequency-inverse document frequency) 是一种常用的关键词提取技术, 它表明了一个词对于语料库中一份文本重要程度。算法的中心思想是: 一个词的重要性与它在文本中出现的次数成正比, 但同时也与它在整个语料库中出现的次数成反比, 即如果某个词在一份文本中出现频率很高, 同时在语料库中其他文本中出现频率很少, 那么就认为这个词对于这份文本非常重要。

在 TF-IDF 中, TF (term frequency) 代表词频, 对于文本 j 的单词 i , 它的 TF 值可以通过公式2-3计算。

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (2-3)$$

其中 $n_{i,j}$ 表示单词 i 在文本 j 中出现的次数, $\sum_k n_{k,j}$ 表示文本 j 中所有单词的出现次数之和。

IDF (inverse document frequency) 代表逆文档频率, 是对一个词可以提供的信息量的一个度量, 可以体现这个词在所有文档中是否重要。详细的计算方法如公式2-4所示。

$$IDF_{i,D} = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|} \quad (2-4)$$

其中 $|D|$ 表示语料库中的文本总数, $|\{d \in D : t \in d\}|$ 表示包含单词 i 的文件数量。最后, 根据公式2-3和2-4可以得到单词 i 的 TF-IDF 值, 如公式2-5所示。

$$TF - IDF_{i,j} = TF_{i,j} \cdot IDF_{i,D} \quad (2-5)$$

2.3 传统文本分类方法

传统文本分类方法使用机器学习中的分类器对文本特征向量进行分类, 这类分类器本质上是通过设定一个能够将任何特征向量映射到某一具体类别的理想目标函数 γ ($\gamma : D \rightarrow C$), 然后根据学习算法减少自身的误差不断接近目标函数, 最终实现分类的目的。按照设定的目标函数的不同, 分类器可以分为线性分类器与非线性分类器, 下面将对几种常见分类器作简要介绍。

2.3.1 贝叶斯分类器

朴素贝叶斯分类器^[18] 是一种典型的线性分类器, 并且也是最古老及最简单的分类器之一。朴素贝叶斯理论是该分类器的基本理论, 它在分类任务中对输入数据有一个基本假设: 数据的各特征之间是条件独立的。虽然这个假设看起来可能

明显是错的，但依据朴素贝叶斯理论实现的朴素贝叶斯分类器在结构化或半结构化数据上都有比较理想的表现。同时，朴素贝叶斯分类器对 CPU 和内存的消耗也少于其他分类器。

朴素贝叶斯理论是指在上面提到的基本假设下，对于事件 A 和 B ，它们之间的概率关系满足公式2-6。

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2-6)$$

而在文本分类任务中，假设一篇文本的特征向量为 $D = \{d_1, d_2, \dots, d_m\}$ ，它的类别为 C ，则上式可以写为公式2-7。

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)} \quad (2-7)$$

其中 $P(C)$ 和 $P(D)$ 是先验概率 (prior probability)，分别表示类别 C 和特征向量 D 出现的概率， $P(D|C)$ 代表在类别 C 中出现特征向量 D 的概率， $P(C|D)$ 则是分类器的结果，称为后验概率 (posterior probability)，代表特征向量 D 是类别 C 的概率。实际应用中， $P(C)$ 和 $P(D)$ 以及 $P(D|C)$ 都可以根据训练语料库直接计算得到。

2.3.2 支持向量机

支持向量机 (Support Vector Machine, SVM) 是一种感知机的改进算法，在机器学习算法中有非常广泛的应用。

在感知机模型中，分类器通过寻找一个可以将数据正确分为两类的超平面来实现二元分类任务。但是，如图2-2所示，这样的超平面往往不是唯一，并且不同的超平面选择会导致感知机的分类准确率截然不同。

为了选择一个分类效果最佳的超平面，支持向量机将距离超平面最近的点设定为支持向量，然后让这些支持向量和超平面直接的距离最大，从而选择出一个最优的超平面，如图所示。并且对于线性分类不适用的非线性数据（如文本数据），支持向量机利用核函数的技巧，通过选择一个恰当的转换函数，将非线性数据映射到一个更高维的向量空间当中，让数据重新分布为线性可分的形式。但是这种做法增加了算法复杂度，复杂的核函数会使算法的计算量成倍的提高，而且如何选择合适的核函数让数据线性可分也没有一个统一的方法。

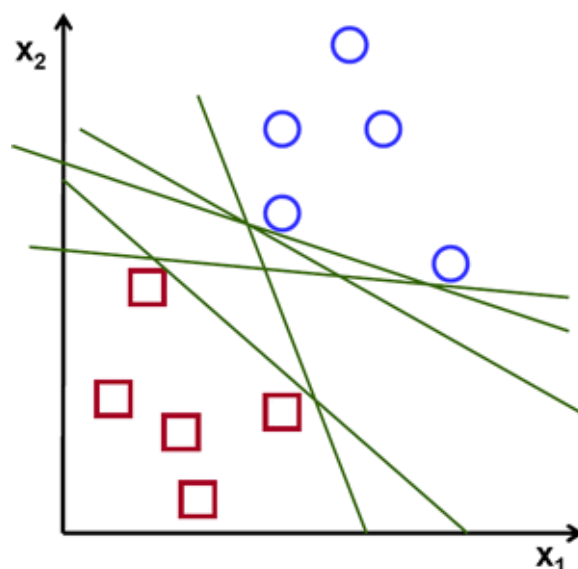


图 2-2 超平面

2.4 基于神经网络的短文本分类方法

传统文本分类方法虽然在篇章级的长文本上能够有较好的分类效果，但是对于文本信息量较少的短文本来说，却难以发挥效用。基于统计的特征提取方法会在训练时生成庞大的特征词库，使得文本特征向量的维度依然很高，一旦文本长度过短，特征向量就会变为稀疏向量，即数据只集中在向量中的某几维上，其他维度都为零。文本特征向量的稀疏性一直是短文本分类上最难以解决的问题之一，为了解决这个难题，学者们开始尝试引入神经网络与深度学习技术。本小结主要介绍基于深度学习的文本表示方法，以及两种常见的深度学习网络。

2.4.1 短文本的分布式表示方法

短文本的分布式表示方法也称为词向量表示法。在神经网络与深度学习模型中，输入数据通常需要转化为矩阵的形式。对于自然语言处理任务，则需要将文本中的词语表示为一个向量，从而让模型理解文本并进一步提取特征，完成分类任务。

One-hot Representation 是词向量表示法最简单的实现方式，具体方法与前文中介绍的向量空间模型大致相同，针对具体语料可以快速得到每个单词的词向量。但与向量空间模型一样，也存在维度过高与稀疏性的问题，并且从词向量之中无法得到词语之间的关系，对短文本分类任务并不适用。

为了得到连续稠密的词向量，Hinton 等人提出了一种称之为词嵌入（Word Embedding）的词向量实现方法^[19]。该方法旨在根据语料库中词语的分布式属性，

来量化不同词语之间的语义相似度。通过一系列转换函数，将词语从高纬度文本空间映射到一个维度相对较低的向量空间，表示为低维实数向量，从而有效的解决了向量的稀疏问题。并且这种映射方法还保留了词语之间的关系，即意思相近的词语它们产生的词向量也会在向量空间中彼此靠近。

目前最常用的词向量构建工具是 word2vec，由 Google 公司于 2013 年发布^[20]。工具基于 2.1.2 中介绍的 N 元语言模型思想，假定文本中的一个单词只和其相邻的 N（word2vec 中称为窗口大小）个单词相关，构建了两套词向量模型——CBOW（Continuous Bag-Of-Words Model）模型与 Skip-gram 模型。

CBOW 模型仿照神经网络语言模型的思想（Neural Network Language Model, NNLM），根据当前词的上下文环境来预测当前词，如图 2-3 所示。不同的是，

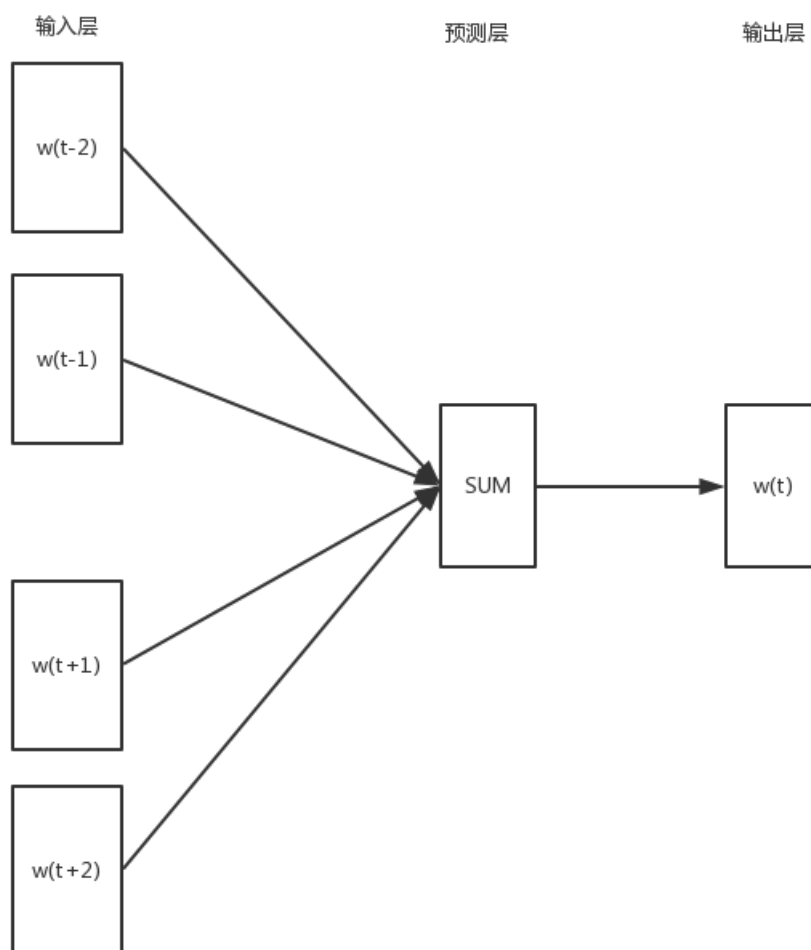


图 2-3 CBOW 模型

CBOW 删除了 NNLM 中前向反馈神经网络的隐藏层，将中间层直接和输出层的

softmax 节点连接，并且忽略当前词上下文的顺序信息，把输入的全部词向量都汇聚在同一个隐藏层节点上。因此从数学上看，CBOW 模型相当于是把一个词袋模型的向量乘以一个嵌入矩阵，来得到一个连续的词嵌入向量。

与 CBOW 模型相反，Skip-gram 模型的思想是根据当前词预测与当前词相邻的词，模型结构如图2-4所示。对于句子 S ，Skip-gram 模型的目标函数如2-8所示。

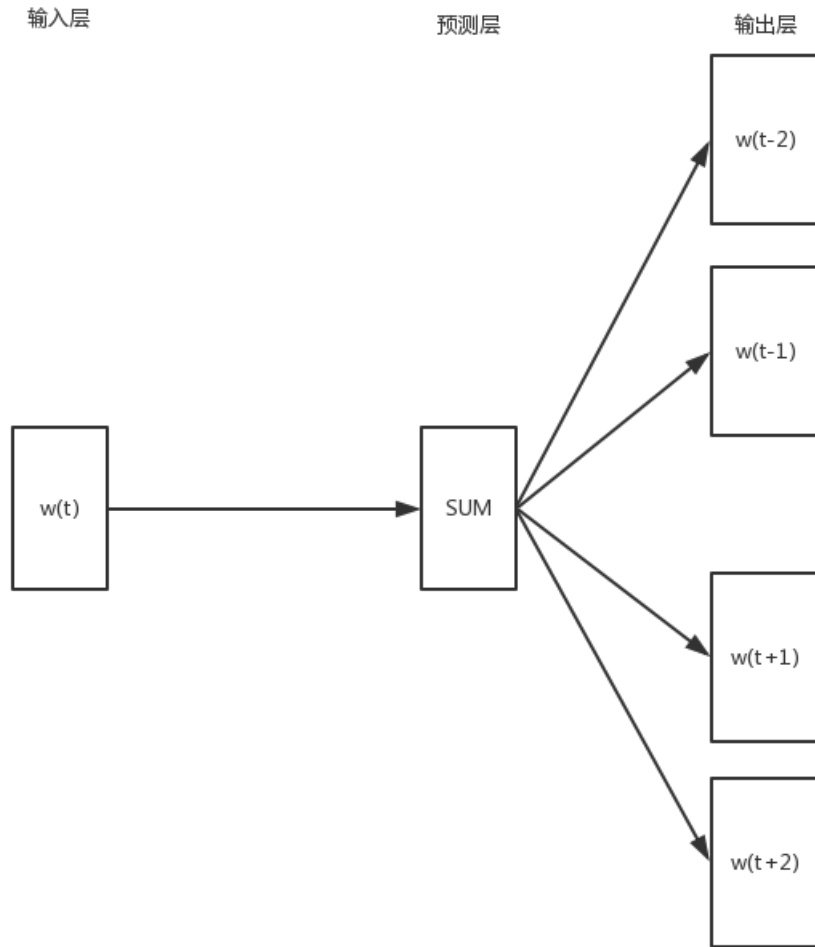


图 2-4 Skip-gram 模型

$$\arg \max_{\theta} \prod_{(w,c) \in S} p(c|w; \theta) \quad (2-8)$$

其中 w 为当前词， c 为当前词 w 周围的一些单词，长度由模型的窗口大小决定， θ 是隐藏层参数。

word2vec 工具的优点主要体现在计算快速，通过简化传统 NNLM 的结构，引

入分层 softmax (Hierarchical softmax) 和负采样 (Negative Sampling) 技术, 极大的减少了预测层到输出层之间的计算量。同时模型采用非监督训练, 适用于无标签的大型语料库。

word2vec 工具的缺点也十分明显: 针对英语的设计, 只使用单词信息进行训练, 使得模型不能够理解单词内在的一些语义信息。因此本文在原始 CBOW 模型之上, 增加中文的字和部首信息, 致力于实现一个适用于中文的词向量模型。

2.4.2 卷积神经网络

卷积神经网络 (Convolutional Neural Network, CNN) ^[21] 是一种前馈神经网络, 通过模仿猫脑皮层中用于局部感知和方向选择神经元结构, 有效降低了普通反馈神经网络的复杂性。

文本分类任务中, 普通的神经网络模型通过构建一个全连接的网络对句子进行处理, 输入层的神经元直接接收文本的词向量, 计算时隐藏层的某些节点就会被输入的特征词激活 (例如情感分类中”不”、”喜欢”等具有较强情感倾向的词), 从而实现最终的分类。但是在这样的网络结构中, 单词之间的位置信息被忽略了, 而同样的特征词, 顺序与位置不同很可能导致不同的分类结构。继续以情感分析为例, 两个都包含”不”和”讨厌”这两个特征词的句子, ”我不讨厌这部电影”表示的是好的情感, 而”我讨厌这部电影, 不会买它的票”则表示坏的情感。

虽然可以通过 n-gram 等关注上下文的多元语言模型来解决这个问题, 但在实际训练中, 模型的窗口大小会是一个非常大的问题。窗口过小会使得模型效果不佳, 窗口过大则导致计算量爆炸。并且, 相似的单词在模型中不能共享参数、权重, 相似单词无法获得交互信息。

经过一系列探索, 学者们发现卷积神经网络能够有效解决上述问题: 基于局部感知野理论, 一个隐藏层神经元只与部分词语连接, 可以达到多元语言模型类似的效果, 学习到更多的文本特征, 并且在特征的识别与学习时只关注词语的相对位置, 忽略它们的绝对位置, 也加速了学习过程; 基于参数共享理论, 相似特征使用同样的参数来提取, 极大的降低了网络模型的总体参数数量, 加快了模型的训练。

卷积神经网络由三层网络组成: 卷积层、池化层、光栅层, 整体结构如图2-5所示。

(1) 卷积层

卷积层主要用于特征提取, 通过多个卷积核, 对输入数据进行卷积计算, 最后得到相应的特征图 (Feature Map)。

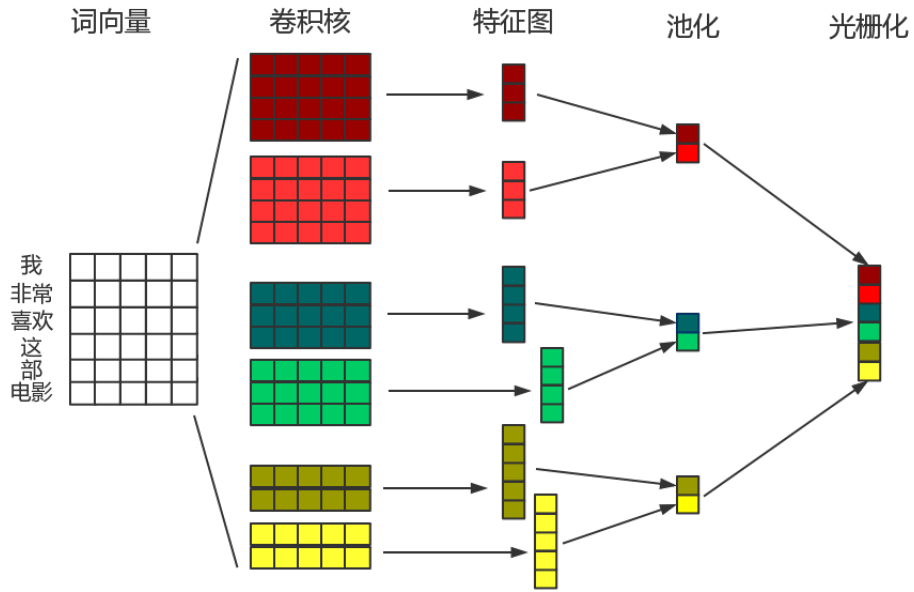


图 2-5 卷积神经网络结构

卷积计算是通信领域中常见的计算方法，假设有二维离散函数 $f(x,y)$ 和 $g(x,y)$ ，那么它们的卷积定义如公式2-9所示。

$$f(x,y) * g(x,y) = \sum_u \sum_v f(u,v) g(x-u, y-v) \quad (2-9)$$

卷积核是提取特征的主要工具，相当于图像处理领域中的滤波器，可以从输入数据中提取一类特征，其他类特征则需要另外的卷积核来提取，所以卷积层通常包含有多个卷积核。一个卷积核包含参数矩阵 W 和偏置项 b ，计算过程如公式2-10所示。

$$h = f(W * x + b) \quad (2-10)$$

其中 h 表示产生的特征图， f 是激活函数。

具体到文本分类任务，输入数据为文本的词向量，卷积核通常覆盖上下几行单词，并且宽度与词向量宽度相同（如图中卷积核部分所示），这样就能够捕捉到多个连续词之间的特征，并且能够在同一类特征计算时共享参数。

(2) 池化层 (Pooling layer)

卷积层输出的特征图在数据纬度上与原始输入数据相比并没有减少很多，并且由于存在多个卷积核，实际上输出数据往往会增多，如果不做处理，无疑会增加网络的复杂度，产生巨大的计算。

池化层也称为下采样层，在卷积层之后，对特征图进行降维操作（即池化操作）。常见的池化操作有最大池化、最小池化和平均池化。其中最大池化是现在最常见的池化方式，该方法将特征图分为多个子块，每个子块内部取最大的一个数据输出，从而达到降低特征图纬度的目的。

对于文本数据，池化操作可以将长度不同的句子转化为长度相同的特征向量，方便后续统一处理。另外最大池化还能够保留显著的特征。卷积核提取特征时，每一种卷积核会专注检测某一种含义的词组，如果这种类型的词组出现了，该卷积核此时的输出值就会非常大，通过最大池化就能够尽可能的将这些信息保留，同时忽略其他无关的信息。

（3）光栅层

输入数据经过卷积层与池化层计算之后，得到的是一系列特征图，而卷积神经网络之后的结构接收的输入可能是一个特征向量。因此需要将这些特征图中的数据依次取出，排列成一个向量，如网络结构图2-5中的光栅化部分所示。

2.4.3 循环神经网络

循环神经网络（Recurrent Neural Network, RNN）也叫做递归神经网络，是深度学习另一个重要的网络模型。RNN 的结构如图2-6所示，其特点是隐藏层节点不仅接收输入层的数据，还会接收上一个节点的输出数据作为另一个输入源，这让 RNN 可以非常自然的处理序列数据，如文本、语音、视频等。RNN 中隐藏层

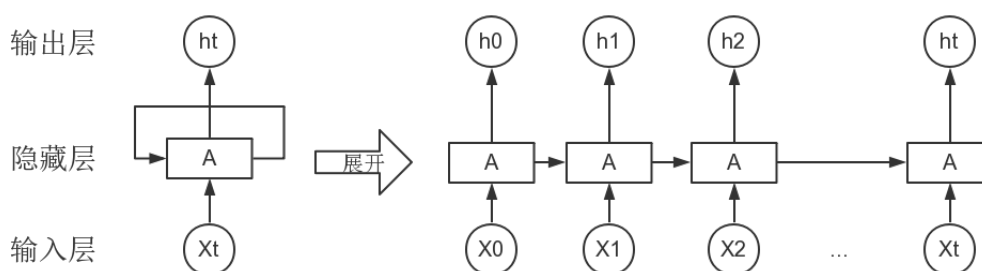


图 2-6 循环神经网络结构

节点包含三个部分，输入参数矩阵 U 、记忆参数矩阵 W 以及输出矩阵 V ，对于节点 t ，计算公式如2-11所示。

$$s_t = f(Ws_{t-1} + Ux_t) \quad (2-11)$$

其中 s_t 为中间向量，同时送往下一个隐藏层节点 $t+1$ 与输出节点 h_t ， f 一般为非线性激活函数， s_{t-1} 。输出节点通过公式2-12得到输出向量。

$$h_t = \varphi(Vs_t) \quad (2-12)$$

其中 h_t 是输出向量， φ 是输出函数，对于分类任务，一般为 *softmax* 函数。

RNN 模型的优点是对文本这类顺序数据有优秀的建模能力，能够突破 n-gram 模型中窗口大小的限制，提取长间隔的单词之间的特征，隐藏层最后一个节点的输出甚至可以包含前面所有单词的特征信息。同时模型长度可变，对任意长度的文本都可以直接处理。

RNN 模型的缺点也十分明显，就是模型训练十分困难。根据反向传播算法，RNN 模型训练时会将梯度在隐藏层之间以乘积的形式传播，如果梯度大于 1，则多次相乘会使其指数级上升，引起梯度爆炸 (Gradient Exploding)；相反如果小于 1，都次相乘后梯度逐渐为 0，又引起梯度消失 (Gradient Vanishing)。

长短期记忆网络 (Long Short Term Memory networks, LSTM) ^[22] 是对传统 RNN 模型的改进，它将每个隐藏层节点看做“记忆细胞”，在里面增加一种称为“门”的结构，对输入输出数据加以控制，从而有效解决梯度爆炸与梯度消失的问题。

LSTM 的结构如图2-7所示，每个节点包含三个门结构 (图中 σ 部分)，分别为遗忘门 (Forget Gate)、输入门 (Input Gate) 与输出门 (Output Gate)。每个门由一个 *Sigmoid* 神经网络层和一个相乘操作组成，其中 *Sigmoid* 层输出 0 1 之间的值，表示对应的部分信息是否应该通过，并利用相乘操作作用在输入或输出信息上，0 值表示不允许信息通过，1 值表示让所有信息通过。在一次计算过程中，LSTM 节点首先计算遗忘门，公式如2-13所示。

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2-13)$$

其中 W_f 与 x_t 是遗忘门中 *Sigmoid* 层的参数矩阵和偏置矩阵， f_t 为遗忘向量，里面的每一位都对应 C_{t-1} 中的一个数据，用来控制信息的流入。

第二步是将输入数据 x_t 存储到节点中，并更新节点状态。这一步首先计算第二个门结构——输入门的 *Sigmoid* 层与一个 *tanh* 层，得到两个候选值 i_t 与 \tilde{C}_t ，如公式2-14与2-15所示。

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2-14)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (2-15)$$

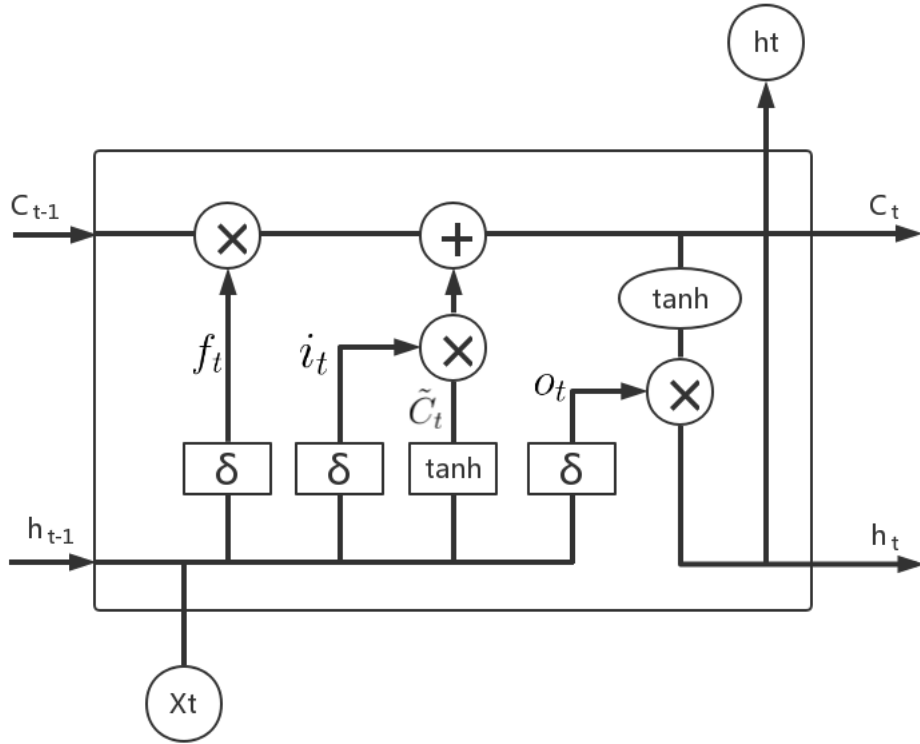


图 2-7 长短期记忆网络

然后根据公式2-16得到节点的新状态 C_t 。

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (2-16)$$

最后，节点通过第三个门结构——输出门，得到输出向量，如公式2-17所示。

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \quad (2-17)$$

通过增加门结构，LSTM 有效避免了传统 RNN 在训练上的问题，但与 CNN 相比，其训练效率依然十分低下，实际使用中常常造成问题。因此，本文对短文分类算法的优化，将融合 RNN 与 CNN，结合它们的优点，构建一个对于中文短文快速有效的分类模型。

2.5 本章小结

本章首先对传统的文本表示方法与文本分类算法的基本理论做了简要介绍，并指出了传统方法在中文短文本分类上的缺陷与不足。接着引出了基于神经网络

的短文本分类方法，详细介绍文本的词向量表示法以及卷积神经网络和循环神经网络在文本分类任务中的应用。这些基础内容为本文的后续工作提供了坚实的理论基础，后续章节将根据这些知识搭建一个中文短文本分类模型。

第三章 中文文本表示方法的研究和改进

文本的向量化表示是中文短文本分类研究的核心之一。传统的 One-hot 表示法虽然简单快速，但忽略的太多的语义信息，在实际使用中效果较差，达不到人们的预期。词嵌入技术将文本中的单词映射到一个连续的向量空间之中，让意思相近的单词能够在向量空间中彼此靠近，这让深度学习模型能够直接通过文本向量获取相关的语义信息，更有利与进一步的文本分类工作。因此，本章节将根据汉字的特点，充分利用汉字中蕴含的丰富信息，设计一种新颖的适合中文的词向量模型。

本章节首先介绍汉字中部首的一些特性，然后简要概述现阶段中文文本表示的一些方法，再详细介绍本文设计的基于部首信息的词向量与字向量模型，最后根据实验结果论述模型的有效性。

3.1 汉字偏旁部首的语义信息

汉字是四大古老文字之一，传说起源于黄帝的史官仓颉。根据内在结构的不同，汉字可分为独体字和合体字。独体字源于最早的甲骨文，是象形文字的延续。如图3-1所示，每一个独体字都可以看做是现实事物的抽象，本身不可分割，整体表达一个语义。合体字则是在独体字之上发展而来的。汉字系统初期几乎都为独体字，数量相对较少，大量事物以通假字来表示，使文字表述存在较大歧义。为了能更精确的表述，人们以基本的象形独体字为基础，通过组合的方式构造大量新的文字。比如最早海上的交通工具就只有：“舟”一种，但演化到现在，细分成“舢、舟、艇、船、舰”等不同小大规模与形制的“舟”。到了现代，合体字成为了汉字的主体，占 90% 以上。

部首是构建合体字的主要单元。这一概念最早由公元 2 世纪汉朝的许慎提出，他在其著作《说文解字》中将合体字中重复出现的部分加以归类，分成 540 个“部”。“部”之后发展为现在的部首，现在通用的部首由清朝康熙五十五年（1716 年）成书的《康熙字典》所制定，共 214 个。

汉字的部首通常由某一个独体字演化而来，而且与英语等字母语言不同的是，部首是汉字语义的重要组成部分。很多时候，部首能够让我们在任何上下文的情况下大致理解或推测一个汉字的意义。这也就是说，在学习汉字语义表示的时候，部首所固有的语义特征能够提供额外的信息。举例来说，通过由“人”字演变而来的部首“亻”，我们可以很清楚的知道“你”、“他”、“伙”、“侣”、“们”等字全部都和



图 3-1 鱼字的演化过程

人有关。并且这种根据部首获得的语义信息与 N-gram 模型这类基于相邻 N 个单词获得的信息本质上是不同的。因此在词向量模型中添加部首的语义特征能够给每个词语增加丰富的语义信息，提升中文词向量的质量。

偏旁是构建合体字的另外一个重要单元。虽然在合体字特别是形声字中，偏旁往往表示事物的读音，但是对于会意字与部分形式字，偏旁任然含有可利用的语义信息。比如汉字“武”，它的部首为“止”，是“趾”的本字，表示脚，偏旁为“戈”，表示武器，整体表示人拿着武器行走；对于汉字“茱”，部首的“艹”表示植物，偏旁“朱”则表示红色，兼表示字音。所以只要善加利用，偏旁也能够为词向量的构建提供很多语义特征。

3.2 中文文本表示方法

中文文本的词向量构建方法一直是中文自然语言处理领域的焦点。在研究初期，研究者们尝试直接使用英文词向量工具（如2.4.1中介绍的 CBOW 模型和 Skip-gram 模型）在经过分词之后的中文语料上训练词向量。但是这种做法很明显存在问题：大多数英文词向量工具在训练时都将词作为最小的操作单位，而忽略了词语内在的一些形态信息（morphological information）。和英语等字母文字不同，中文词语之中的汉字任然含有丰富的语义信息。比如“智能”这个词，它的语义信息一方面我们可以从语料中相关的上下文学习，就像 word2vec 模型一样。另一方面我们也可以根据组成这个词的两个汉字“智”和“能”推测出来（“智”为智慧，“能”为能力）。

为了解决上述问题，充分利用中文词语的语义信息，陈志勇等人^[23]在普通 CBOW 模型上加入汉字信息，提出了 CWE 模型。随后 Xu 等人^[24]在 CWE 模型

的基础上，赋予词中每个字权重，优化整个词向量模型。但这些模型都没有利用汉字中的偏旁部首，忽略了所有文字内部的信息。

另一方面，Sun 等人^[25]在 CBOW 模型的基础上增加了部首信息来训练中文字向量；Yu 等人^[26]在 CWE 模型中结合“部首-汉字”与“汉字-词”的信息，设计了一个多粒度的词向量模型。但这些方法都只是简单的在模型中添加了部首信息，没有考虑到部首的演化，即没有将部首与意义对应的汉字联系起来（比如“氵”-“水”），这使得模型从部首获得的信息非常有限，影响生成的词向量的质量。

3.3 部首信息增强的中文词向量构建方法

根据上一小节中介绍的其他学者在中文词向量模型上的种种成果以及汉字中偏旁部首的特性，本文设计了一种新的中文词向量模型——部首信息增强的中文词向量模型（Radical Enhanced Chinese Word Embedding, RECWE）。RECWE 以 word2vec 工具的 CBOW 模型为基础，在模型的预测层中增加一个代表“偏旁部首-汉字”信息的隐藏层，增强每个词的语义信息。同时，RECWE 将输入文本从简体转换为繁体，并对每个字的部首进行语义转换，最大程度上丰富输入词语的语义信息。

3.3.1 文本预处理

由于实验所用的语料大多来自于互联网，其格式编码往往没有统一，所以需要首先进行相关处理，得到一个相对整齐一致的文本数据。同时为了让词向量模型能够获取部首信息，还需要进行语义相关的一些处理。整体的预处理流程包括：

（1）特殊字符过滤

网络搜集的数据常常是用户随意产生的非规范文本，里面会包含大量和文本语义无关的特殊字符与标点符号，因此需要对这些符号进行过滤，只保留文本以及相关英文专有名词。

（2）全半角转换

由于使用的语料来源广泛，相互之间没有共同的格式规范，文本中的数字、英文字母等同时存在全角和半角的格式，为了让之后的分词工作能够准确快速，需要将所有数字及字母统一转换为半角字符。

（3）分词

本文所使用的分词工具为 java 语言实现的 Ansj 中文分词开源库。该库基于 n-Gram、CRF 与 HMM 算法，分词速度达到每秒钟大约 200 万字左右，准确率能达到 96% 以上，并且实现了中文分词、中文姓名识别、用户自定义词典、关键字

提取、自动摘要、关键字标记等重要功能，对于短文本语料有较好的表现。

(3) 繁简转换

新中国成立后进行了一系列汉字简化工作，现阶段中国大陆地区使用的简体汉字是 1956 年发布的^[27]。但在汉字简化工作中，存在一些不适当的简化，使得简化字的部首与原始繁体字的部首不同，出现汉字语义上的偏差。例如繁体字中的“熊”字，分解后的偏旁部首为“𧢲”和“熊”，意指一种很大的熊，可是简化后却变成了“黑”，相应的偏旁部首为“罢”和“灬”，失去了原来的含义。而且简化工作还将多个繁体字归并为一个简体字，即“一简对多繁”，这就造成了理解上的困难和歧义，并生成了一批多音字，让简体词语中的字不能很好的反应这个词的语义，增加了词向量模型学习词语语义的难度。比如繁体的词语“干(gān)涉”、“乾(gān)燥”、“幹(gàn)部”中的“干”、“乾”和“幹”，简化后统一变为“干”字，让模型很难区分。

因此，为了更好的表达原始语料中文本的语义，让模型能够正确捕获中文词语中的部首的语义信息，需要将原本的简体文本全部转换为繁体文本。本文使用汉语言处理包 HanLP 作为繁简转换工具，该处理包能够有效区分同一个简体字对应的多个繁体字，并且可以识别简繁分歧词，如“打印机”与“印表機”。

(4) 部首提取与语义转换

为了让 RECWE 模型能够直接获取汉字的部首信息，预处理阶段会将词语中的汉字拆解为对应的偏旁部首。但是，根据 3.1 节中关于部首的介绍，由某一个独体字演化而来的部首，在合体字中会为了书写美观往往需要进行一定的拉伸或收缩，来适应合体字整体的字型。例如“水”字在合体字中拉伸成了部首“氵”，“食”字在合体中收缩为了部首“饣”。因此本文提取了常见字的部首并构建了一个转换表(如表 3-1 所示)，将部首转换为对应的独体字，从而让 RECWE 模型能够有效识别出这种内在联系，比如“木”字与“案”、“板”、“森”、“李”等字，“水”字与“泉”、“河”、“海”等字。

3.3.2 部首信息增强的中文词向量模型

本文提出的 RECWE 模型结合了中文的单词、汉字以及偏旁部首的语义信息，旨在构建适合中文的高效词向量。RECWE 模型基于 CBOW 模型^[20]，通过词的平均上下文向量和字与部首的平均上下文向量预测目标词，并且使用这两个上下文向量的预测损失的和作为目标函数。

RECWE 模型结构如图 3-2 所示，其中 w_i 表示需要预测的目标词， w_{i-1} 和 w_{i+1} 分别表示文本中目标词 w_i 左边和右边的词。 c_{i-1} 和 c_{i+1} 表示 w_{i-1} 和 w_{i+1} 中的汉字， r_{i-1} 和 r_{i+1} 则分别表示 c_{i-1} 和 c_{i+1} 对应的部首(经过表 3-1 转换)， s_{i-1} 和 s_{i+1}

表 3-1 常见部首转换表

部首	对应汉字	部首	对应汉字
艹	艸	亻	人
刂	刀	犴	犬
灠	火	金	金
麥	麥	食	食
示	示	月	肉
攴	攴	𦉳	网
扌	手	氺	水
糸	糸	耂	老
牛	牛	忄	心
衤	衣	王	玉
辵	走	疒	病

表示 c_{i-1} 和 c_{i+1} 的偏旁。通过在预测层将两个上下文向量并列以及共享隐藏层参数，字向量与偏旁部首向量在训练时能够很好的影响词向量，从而让最终训练得到的词向量能够充分反映出词语内部汉字与部首的语义信息。

与 CBOW 模型类似的，RECWE 模型的目标函数是两个上下文向量对于目标词 w_i 的条件概率的对数似然函数，如公式3-1所示。

$$L(w_i) = \sum_k^2 \log P(w_i | h_{i_k}) \quad (3-1)$$

其中 h_{i_1} 和 h_{i_2} 分别表示词的上下文向量和字与部首的上下文向量。通过这样

每个上下文向量对于目标词 w_i 的条件概率 $p(w_i | h_{i_k})$ 可以利用 softmax 函数求解，如公式3-2所示。

$$p(w_i | h_{i_k}) = \frac{\exp(h_{i_k}^T \hat{v}_{w_i})}{\sum_{j=1}^N \exp(h_{i_k}^T \hat{v}_{w_j})}, k = 1, 2 \quad (3-2)$$

其中 \hat{v}_{w_i} 表示目标词 w_i 的“输出”向量， \hat{v}_{w_j} 表示输入语料中每个词的“输出”向量， N 表示输入语料的长度。

上下文向量 h_{i_1} 是上下文窗口中每个词的“输入”向量的平均值，通过式3-3得到：

$$h_{i_1} = \frac{1}{2T} \sum_{-T \leq j \leq T, j \neq 0} v_{w_{i+j}} \quad (3-3)$$

式中的 T 表示上下文窗口的大小， $v_{w_{i+j}}$ 是上下文窗口中单词的“输入”向量。

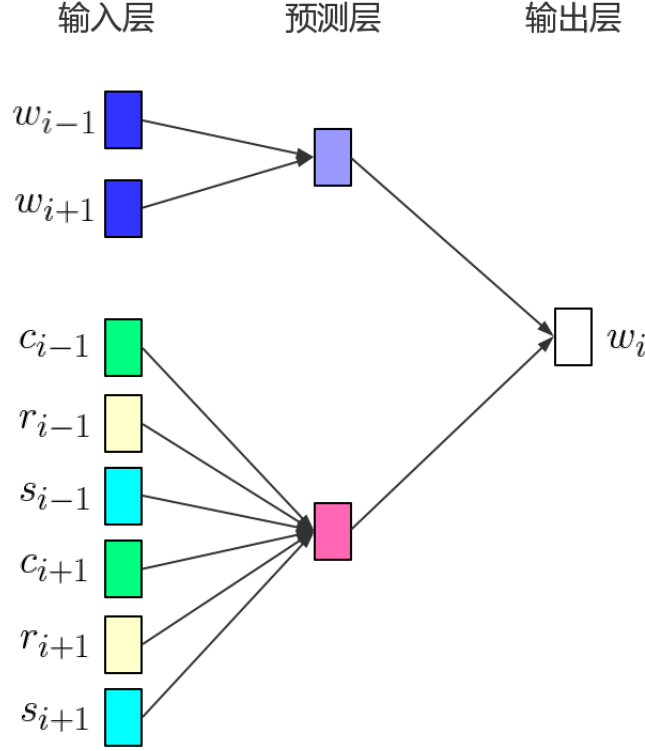


图 3-2 RECEW 模型结构

类似的，字与部首的上下文向量 h_{i_2} 是上下文窗口中每个词对应的字及其偏旁部首的“输入”向量的平均值，计算公式如3-4所示。

$$h_{i_2} = \frac{1}{X} \sum_{-T \leq j \leq T, j \neq 0} v_{c_{i+j}} + v_{r_{i+j}} \quad (3-4)$$

其中 $v_{c_{i+j}}$ 和 $v_{r_{i+j}}$ 分别表示字和偏旁部首的“输入”向量， X 为 $v_{c_{i+j}}$ 和 $v_{r_{i+j}}$ 的数量。

于是，对于语料库 D ，RECWE 模型的整体对数似然函数如公式3-5所示。

$$L(D) = \sum_{w_i \in D} L(w_i) \quad (3-5)$$

RECWE 模型的训练算法采用 CBOW 模型实现的负采样 (Negative Sampling, NEG) 算法^[20]，该算法是 NCE (Noise Contrastive Estimation, NCE) 算法的一个简化版本，其中心思想是将当前窗口内的词语分为正样本 (需要预测的目标词) 及负样本 (目标词之外的其他词)，然后加权随机选取负样本进行更新，每个负样本的权重与其在语料库中出现的次数成正比，以此来提高模型的训练速度并改善所得词向量的质量。结合随机梯度上升技术，基于负采样的 RECWE 模型能够快速获得中文词向量。

3.4 部首信息增强的中文字向量构建方法

中文分词技术虽然已相对成熟，但准确率仍然达不到 100%，分词结果依然存在一定的错误。特别是对于本文主要研究的中文短文本，由于其大多来自于用户在互联网中的互动，如腾讯空间说说、新浪微博、淘宝商品评价等，相比于长文本具有很强的随意性，充斥着大量的口语化表达方式，有些句子甚至存在语病，这更进一步增加了分词的难度，使得分词结果很不理想。而错误的分词结果会极大影响后续的词向量模型，降低词向量的质量。

字向量是解决分词问题的一个方向，来源于英文文本处理中字符级别 (Character-level) 的文本表示方式^[10]。国内学者通过给汉字编码^[23,28]，或将汉字转换为汉语拼音^[10]，将文本表示为连续的向量，从而绕过分词的步骤，然后通过循环神经网络等深度学习技术直接从字向量中学习文本特征，取得了不错的成果。但这些方法得到字向量类似于 One-hot 向量，没有反应出每个字直接语义上的联系，忽略了大量的语义信息。因此，作为 RECWE 词向量模型的补充，本文设计了一个能够利用汉字部首信息的字向量模型——部首信息增强的中文字向量模型 (Radical Enhanced Chinese Character Embedding, RECCE)，为后续的分类模型提供更多的文本信息。

RECCE 模型是上节中介绍的 RECWE 模型的一个变形，它替换了输入层的词向量，将模型的主要训练对象转换为字向量，网络结构图如图3-3所示，其中 c_i 表示当前需要预测的目标字，其他符号与图3-2相同。

通过 RECCE 模型将汉字映射到连续的向量空间，克服了 One-hot 类型的字向量的种种问题，同时由于添加了偏旁部首的语义信息，具有相同语义的字向量能够在空间上彼此靠近（如“病”、“疤”、“癌”等字），增强了字向量对于文本的表现力。并且对于某些非常短小的文本，如新闻标题、微信聊天记录等，使用词向量进行表示得到的数据长度往往不超过 10，还可能存在分词错误，不利于后续分类模型的特征提取工作。使用字向量则一定程度上丰富了文本的信息量，让分类模型能够获取到更多的特征，避免分词错误带来的影响，提升分类准确率。

3.5 实验及其结果分析

为了验证算法获得的词向量与字向量的可行性与有效性，本节将根据来源于“今日头条”新闻网的新闻标题数据进行实验。实验实现前面介绍的 RECWE 模型与 RECCE 模型，同时实现了3.2小节中提到的 word2vec 工具、CWE 模型^[23]、SCWE 模型^[24]、JWE 模型^[26]与1.2小节中提到的 charCBOW 模型^[11]用于对比实验。词向量、字向量的有很多种评价方法，本实验主要使用词相似度 (Word Similarity)

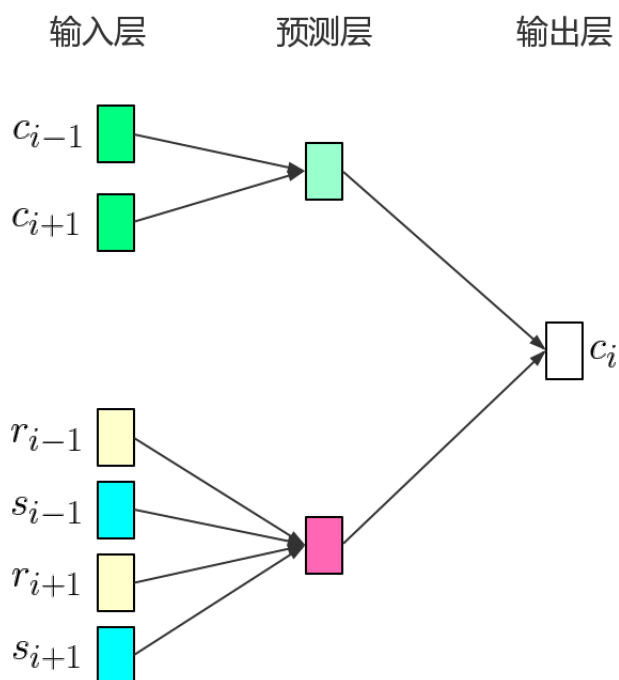


图 3-3 RECCW 模型结构

与词类比（Word Analogy）方法来评价不同模型训练出的词向量，使用字相似度（Character Similarity）方法来评价不同模型训练出的字向量。

3.5.1 部首信息增强的中文词向量模型实验对比

训练语料使用 RECWE 模型使用的 Ansj 工具分词，哈工大以及百度停用词表过滤掉新闻中的停用词，所有词向量模型采用统一的训练参数，具体如表3-2所示。

表 3-2 词向量训练参数表

参数名	参数意义	参数值
size	词向量维度	200
alpha	学习速率	0.025
mincount	语料中低频词的出现词频	5
sample	高频词汇的随机降采样的配置阈值	1e-4
workers	训练的并行数	4
iter	迭代次数	5
window	当前词与预测词的最大距离	5

(1) 词相似度

词相似度主要用来评估词向量判断语义相近的单词对的能力^[26]。本实验选择了文献 [23] 提供的两个不同的相似词数据库 wordsim-240 和 wordsim-296，它们分别包含 240 个中文单词对与 296 个中文单词对，每个单词对都包含一个人工标注的相似度。wordsim-240 主要针对语义上有联系的单词，wordsim-296 则针对同义词，部分数据如表3-3与表3-4所示。

表 3-3 wordsim-240 数据节选

单词 1	单词 2	相似度
李白	诗	9.2
浏览器	网页	8.95
医生	职责	8.85
白天	晚上	8.8
学生	学校	8.71
法官	法律	8.65
员工	公司	8.65
...
教授	黄瓜	0.5
鸟	自行车	0.5
蛋白质	文物	0.15

表 3-4 wordsim-296 数据节选

单词 1	单词 2	相似度
足球	足球	4.98
老虎	老虎	4.8888888889
恒星	恒星	4.7222222222
入场券	门票	4.5962962963
...
签名	暂停	0.304
股票	美洲豹	0.304
延迟	种族主义	0.262962963

对于每个词向量模型，单词对的相似度通过这两个单词相应的词向量的余弦距离来表示。实验最后依靠计算词向量得到的相似度与相似词数据库中人工标注的相似度之间的 spearman 相关系数 (Spearman Correlation)^[20] 判断词向量的好坏，spearman 相关系数越高，代表该词向量越能体现单词之间的相关性。具体的

实验结果如表3-5所示。

表 3-5 词向量评估结果

模型	wordsim-240	wordsim-296
word2vec	0.4221	0.4479
CWE	0.43635	0.4750
SCWE	0.4311	0.4648
JWE	0.4467	0.5831
RECWE	0.4962	0.5849

从结果中可以看出，在两个相似词库上，RECWE 模型的表现都优于其他模型。CWE 模型和 SCWE 模型相比 word2vec 结果有所提升，但由于只利用了单词中的汉字信息，所以相较于添加了部首信息的 JWE 模型与 RECWE 模型结果要差。还可以看到，在 wordsim-240 数据库上，RECWE 模型与 JWE 模型相比有较大提升。这主要是因为该数据库中包含的大多是语义上有一定从属关系的单词对，而拥有部首转义机制的 RECWE 模型能够很好的找到这样的关系。比如单词对“淋浴”与“水”，RECWE 模型会在预处理阶段把“淋”字和“浴”字的偏旁“氵”转义为“水”，然后在模型训练阶段就可以发现“淋浴”与“水”之间的内在联系。

(2) 词类比

词类比是另一种衡量词向量好坏的技术，它主要判断词向量是否能够体现出单词对之间的语言规律^[29]。举例来说，给定单词关系组“北京-中国: 东京-日本”，一个好的词向量就应该让单词“日本”的词向量 $vec(\text{日本})$ 接近式子“ $vec(\text{中国}) - vec(\text{北京}) + vec(\text{东京})$ ”产生的向量。也就是说，给定一个类比关系“ $a-b:c-d$ ”，如果词向量能够通过公式3-6在词表中寻找向量 x 来使关系“ $a-b:c-x$ ”成立，那么就判定这个词向量包含这个类比关系。

$$\arg \max_{x \neq a, x \neq b, x \neq c} \cos(\vec{b} - \vec{a} + \vec{c}, \vec{x}) \quad (3-6)$$

本次实验使用文献 [23] 提供的中文词类比数据库，包含了 1124 组词类比关系，每组类比关系包含 4 个单词，所有类比关系分为三个类别：“首都” (677 组)、“省会” (175 组)、“人员” (272 组)。具体数据如表3-6所示。

实验用准确率作为结果指标，准确率越高，表明词向量包含的词类比关系越多，三类词类比的实验结果如表3-7所示，总体结果见图3-4所示。

从这两个结果中可以看出，同其他词向量模型相比，RECWE 模型在三类类比关系上都取得了最优的结果，表明部首信息能够增强模型在语言规律上的表现

表 3-6 词类比数据库（部分）

类别	单词对 1	单词对 2
首都	曼谷-泰国	北京-中国
首都	东京-日本	柏林-德国
首都	伦敦-英国	开罗-埃及
省会	长春-吉林	哈尔滨-黑龙江
省会	南昌-江西	南京-江苏
省会	广州-广东	成都-四川
人员	爸爸-妈妈	兄弟姐妹
人员	国王-王后	父亲-母亲
人员	爷爷-奶奶	叔叔-阿姨

表 3-7 各类类比关系实验结果

词向量模型	首都	省会	人员
word2vec	0.11816	0.11428	0.34558
CWE	0.20787	0.14285	0.32720
SCWE	0.21381	0.14022	0.33021
JWE	0.22101	0.16	0.32352
RECWE	0.23632	0.18285	0.38603

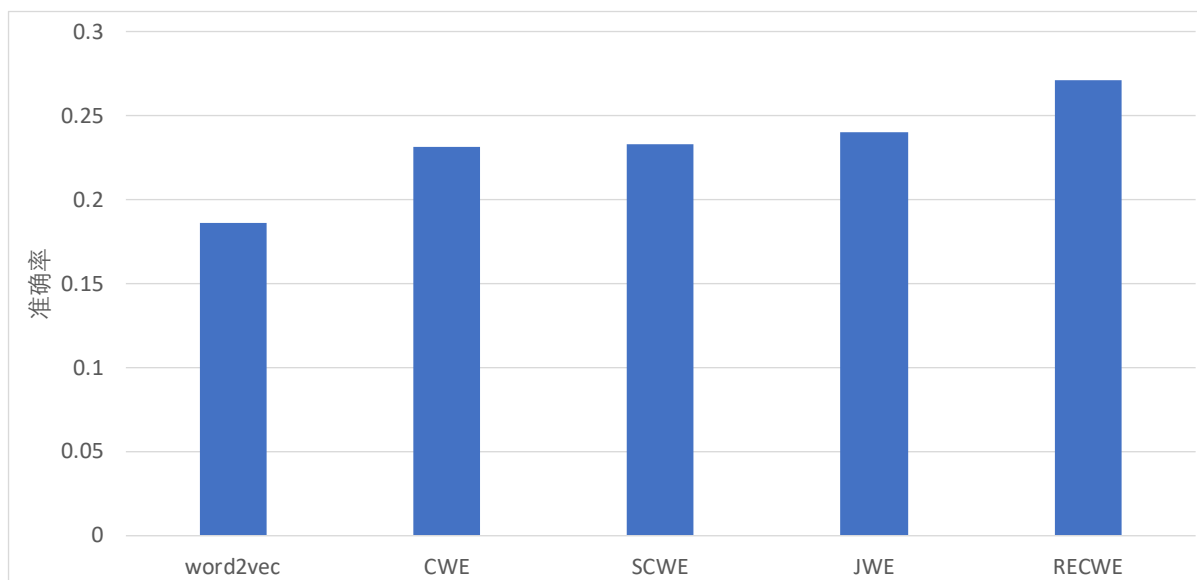


图 3-4 词类比整体实验结果

力。特别在“人员”类上，部首转义机制让 RECWE 模型能够相对容易的根据部首

发现词对之间的关系，如通过部首“女”联系“妈妈”、“姐姐”等词，所以结果中相对 JWE 模型准确率有较大提升。而对于“首都”这类包含大量音译词的类别，其内部部首不能提供额外语义，则提升较小。

3.5.2 部首信息增强的中文字向量模型实验对比

本实验通过字相似度技术比较不同模型训练得到的词向量，使用的字相似度数据库类似上一个小节中使用的词相似度数据库，描述了 400 个常见字的关联关系，部分数据如表3-8所示。字向量模型的训练参数与上一小结中的训练参数相

表 3-8 字相似度数据库（部分）

汉字 1	汉字 2	相似度
犬	狗	11.425
癌	病	10.525
七	列	0.5
车	鸟	0.427

同，实验结果见表3-9所示。

表 3-9 字向量评估结果

模型	spearman 相关系数
word2vec	0.47693
charCBOW	0.48852
JWE	0.56075
RECCE	0.59409

可以看到，RECCE 模型训练的字向量总体上有了一个较优秀的结果，表明添加部首的语义信息对于字向量的训练同样有效。

3.6 本章小结

本章节主要介绍中文短文本的表示方法，首先分析了汉字的演变以及汉字部首对于汉字语义的贡献，接着总结了近几年相关学者在中文文本表示上的一些研究，然后在此基础上提出了两种新的文本表示模型——部首信息增强的词向量模型与部首信息增强的字向量模型，最后设计实验与其他模型进行对比，实验结果表明模型能够有效提升词向量与字向量的质量，为后续分类模型提供充实的基础。

第四章 Attention-Based 字词结合卷积循环中文短文本分类模型

短文本分类在自然语言处理领域中扮演着重要的角色，在垃圾信息过滤、语义分析、自动问答等任务中有着广泛的应用。本节将通过结合基于局部感知域理论设计的卷积神经网络以及基于时间记忆理论设计的循环神经网络，组合两种网络的优点，提出一种新型的中文短文本分类模型。该模型还引入了 Attention 机制等现在最新的深度神经网络技术，并结合字向量与词向量，优化神经网络提取出的文本特征向量，防止特征向量信息冗余或信息缺失，增强模型在中文短文本上的分类效果。

4.1 卷积循环神经特征提取网络

根据2.4.2小节与2.4.3小节的介绍，我们知道卷积神经网络和循环神经网络在特征提取方面都有优秀的表现，各有其优缺点。一方面，卷积神经网络能够从序列数据（如文本数据）和空间数据（如图片数据）之中快速的学习本地特征，得到分类结果，但忽略了特征的顺序，对某些数据并不适合；另一方面，循环神经网络则专门对序列数据进行建模，但却无法并发的提取数据特征，使得模型运算较慢，无法大规模应用^[30]。针对这样的特点，本文将组合卷积神经网络与循环神经网络，把它们堆叠在一个系统中共同进行特征提取工作，形成一个新的特征提取网络，有效的吸收两种模型的优点，同时避免其缺点。

4.1.1 卷积循环神经网络整体结构

卷积循环神经网络由卷积网络层和循环网络层两个主要模块组成，总体结构如图4-1所示。文本数据编码后，卷积网络层会对其进行处理，得到文本的初级特征图，特征图之后输入循环网络层，进行更高一级的特征提取，并将网络中最后一个隐藏节点的输出向量作为最终的特征向量。

4.1.2 卷积网络层

卷积网络层的结构与普通卷积神经网络大致相同，分为卷积模块、池化模块与光栅模块。为了适应本文研究的中文短文本数据，本文对每个模块的具体实现做出了一些调整，下面对各个模块进行说明：

（1）卷积模块

基于文本数据的信息特征，卷积网络层的卷积模块使用窄卷积（Narrow Convolution）作为卷积策略，避免宽卷积（Wide Convolution）的补零操作对提取

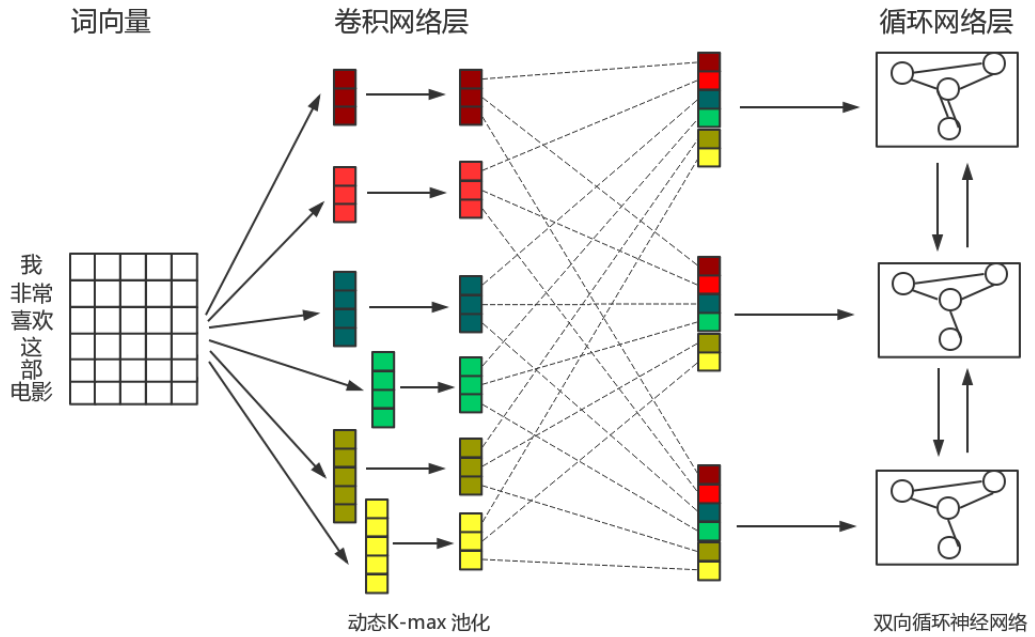


图 4-1 卷积循环神经网络结构图

文本特征造成影响。卷积核长度分为 3、4、5 三种，宽度为词向量的长度，滑动步长为 1，保证特征图能够包含尽可能多的文本特征。每种卷积核生成 128 个，保障特征图的多样性。

(2) 池化模块

由于短文本数据的长度以及卷积核的长度并不统一，卷积层产生的特征图的长度各有不同，为了让接下来的循环网络层能够处理特征图，需要在池化模块对其进行处理，将长度统一。本文使用 k-max 池化（k-max pooling）作为池化算法，该算法是 Kalchbrenner 提出的动态 k-max 算法的前置算法^[31]，主要思想是对于一个给定的 k 值与特征序列 p ($p \in R^p, length(p) \geq k$)，选择序列 p 中前 k 个最大值，且保留原来序列的次序（即生成序列是原序列 p 的一个子序列）。通过 k-max 池化算法，卷积网络层不仅能够接收变长的输入，同时生成的特征图也保留了相对的位置信息，提升了特征的质量。

(3) 光栅模块

光栅模块主要用来整合生成的各个特征图，将其中相同位置的特征值进行合并，形成最终的特征向量，输入下面的循环网络层，如网络结构图4-1中的虚线所示。

4.1.3 循环网络层

循环神经网络依据时间对序列数据进行建模，上一个节点的数据不仅会进行输出，还会作为同一隐藏层下一个节点的输入，隐藏层最后一个节点则能够接收到全部的文本特征数据。但是短文本分类任务需要更多的考虑文本上下文信息，单纯使用正向循环神经网络，模型只能依据上文信息进行分类，无法利用下文的的信息，从而影响最终的分类效果。因此，本文使用双向长短时记忆循环神经网络 (Bi-directional Long Short-Term Memory, Bi-LSTM) 作为循环网络层的实现，将网络分为前向传递层与后向传递层，分别获取输入短文本的上文信息与下文信息。

Bi-LSTM 网络的整体流程如图4-2所示，对于从上层网络传来的输入向量，网络首先将其转变为正序序列及逆序序列两个向量，然后分别输入一个单向 LSTM 网络进行特征提取，得到正序特征向量与逆序特征向量，之后将两个特征向量合并，形成最终的文本特征向量并输出到下一层网络。经过这样的处理，网络提取的特征向量既包含上文信息又包含下文信息，能够给之后的分类网络提供更加丰富的语义信息，有效缓解了短文本数据语义信息不足的问题，提升了分类的准确率。

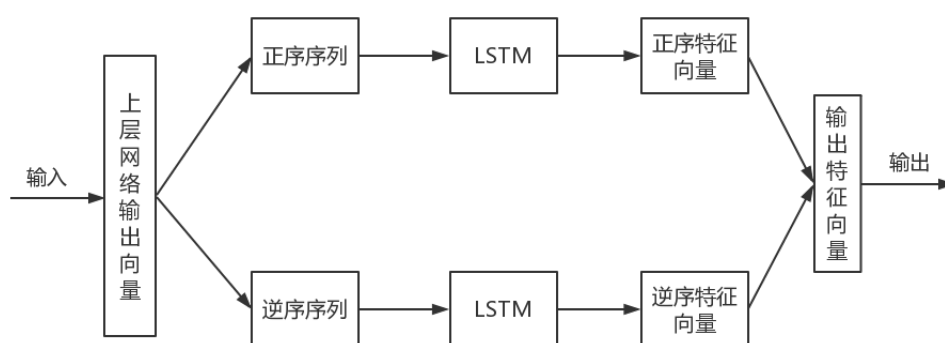


图 4-2 Bi-LSTM 网络流程图

4.2 Attention Model

Attention Model（注意力机制）是近几年兴起的一个新型模型，在很多场景被证明有效。模型以认知心理学中的人脑注意力理论为核心，认为人脑在处理具体任务时，对于相关事物的注意力会集中在某一个特定的地方，忽略其他无关的地方，而不是对每个地点分配相同的注意力。通过这样的理论，Attention Model 能够合理的分配模型的计算资源，并且还可以避免非关键数据对结果的影响。Attention Model 最先在计算机视觉领域被应用于图片识别等问题，取得了很好的成果，随后

在自然语言处理领域也获得了优秀的成绩。下面将以编码-解码模型中的 Attention 机制为例,说明 Attention Model 的基本知识,然后在下一小节把 Attention Model 应用在卷积循环神经网络当中。

编码-解码模型是自然语言处理领域中一个非常普遍的模型,它通过编码器将输入向量编码为中间变量,再用解码器解码成输出向量,完成输入数据到另一组输出数据的转变,从而实现多种内容转换任务,如机器翻译、文本复述等。同时编码-解码模型的编码器与解码器没有具体限定,可以使用各种各样的深度学习模型,如 CNN、RNN 等,让模型具有很强的泛用性,能够适合各种应用场景。编码-解码模型的一般架构如图4-3所示。

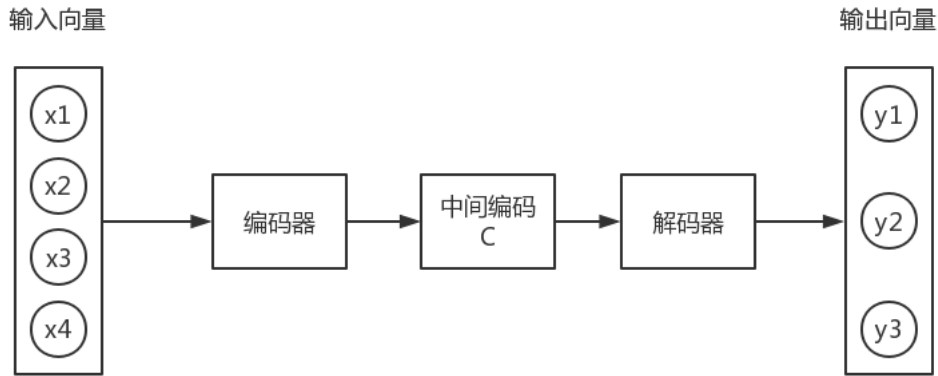


图 4-3 编码-解码模型架构图

编码-解码模型的输入一般是一个长度为 n 向量序列 X ($X = \{x_1, x_2, x_3, \dots, x_n\}$), 输出则是一个长度为 m 的向量序列 Y ($Y = \{y_1, y_2, y_3, \dots, y_m\}$)。模型运行时, 编码器根据一定的映射规则对输入序列进行编码, 得到中间编码 C , 表达式如4-1所示。

$$C = \text{Encode}(x_1, x_2, x_3, \dots, x_n) \quad (4-1)$$

之后解码器对中间编码 C 进行解码, 得到输出序列 Y , 计算公式如4-2所示。

$$y_i = \text{Decode}(C, y_1, y_2, y_3, \dots, y_{i-1}), i = 1, 2, 3, \dots, m \quad (4-2)$$

可以看出解码器在计算输出序列 Y 时, 对于每一个输出子项 y_i 用到的数据信息是一样的, 都是输入序列 X 编码后得到的中间编码 C , 即输入序列 X 对输出序列 Y 中每一个子项的影响是相同的。这样的方式无疑忽略了很多细节, 在一些任务中甚至直接影响了输出结果。比如机器翻译任务, 一个词的翻译虽然与上下文

和词本身同时有关，但对于人名机构名等名词，翻译时词本身的影响必定较重，而其他的动词形容词等则可能上下文的影响较大，所以简单输入序列同比例的作用在输出序列之上并不是一个好的方法。

Attention Model 的出现很好的解决了这个问题，在 Attention-Based 编码-解码模型中，编码器的输出不再是一个完整的中间编码 C ，而是输入序列 X 在编码阶段的历史状态 C_i ，如公式4-3所示。

$$C_i = \sum_{j=1}^n \alpha_{ij} \text{Encode}(x_j) \quad (4-3)$$

其中 α_{ij} 表示当前输入子项 x_j 的注意力权重，这个值越高，表明 x_j 对结果越重要。

解码器根据历史状态 C_i 得到优化后的输出序列，如公式4-4所示。

$$y_i = \text{Decode}(C_i, y_1, y_2, y_3, \dots, y_{i-1}), i = 1, 2, 3, \dots, m \quad (4-4)$$

可以发现，Attention Model 本质上既是在模型的输出结果上添加一个注意力权重，筛选对结果重要的数据项，同时过滤对结果影响较低的数据项，以此优化模型的输出效果。这一思想在短文本分类任务中同样适用，因为对于短文本分类任务，确定分类结果的往往是文本的几个关键词，如情感分析中的“喜欢”、“讨厌”等词，如果能够判断出这些关键词，就可以极大的增强分类模型的结果。

4.3 Attention-Based 字词结合卷积循环中文短文本分类网络

4.3.1 Attention-Based 卷积循环特征提取网络

根据 Attention Model 的思想，本文改进了4.1小节提出的卷积循环特征提取网络，增加注意力权重，优化原始网络产生的特征向量。

Attention Model 的核心在于注意力权重的获取，有多种实现方式，如 Soft Attention、Hard Attention 等^[32]。为了减少模型的计算负担，本文选择参数化的 Soft Attention 实现方式，该实现方式能够让整个 Attention 层直接嵌入模型，使梯度可以经过 Attention Mechanism 模块，反向传播到其他地方，以此简化模型的训练过程。总体计算流程如图4-4所示，首先把双层 LSTM 的输出 h ($h = \{h_1, h_2, \dots, h_n\}$) 送入一个单层全连接神经网络，根据4-5公式将其转换为中间向量 u ，作为原始输出 h 的隐藏表示。

$$u_i = \tanh(W_w h_i + b_w) \quad (4-5)$$

然后通过 softmax 函数计算中间向量 u 与文本上下文向量 u_w 的相似度 α ，如

公式4-6所示。

$$\alpha_i = \frac{\exp(u_i^\top u_w)}{\sum_n \exp(u_i^\top u_w)} \quad (4-6)$$

其中 u_w 是记忆向量，可以视为筛选重要特征的高层抽象参数，代表着整个语料库的上下文信息^[33]，在模型初始阶段随机初始化，并在训练时同其他参数一起调整。计算结果 α 则是标准化的注意力权重，用于下一步计算。

最后，把原始向量 h 和 α 加权相加，就得到 Attention 优化后的特征向量 S ，如公式4-7所示。

$$S_i = \sum_n \alpha_i h_i \quad (4-7)$$

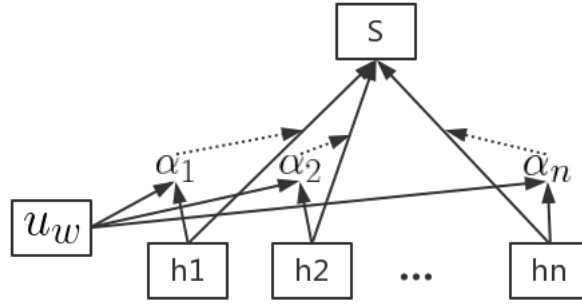


图 4-4 Soft Attention 计算流程

4.3.2 双通道字词结合短文本分类模型

在短文本分类任务中，词向量与字向量都能够作为文本的向量表示，但都有其固有的缺点：词向量极度依赖分词结果，错误的分词会极大的影响词向量的性能；字向量虽然不存在分词的问题，但有些词语和内部的汉字意思可能南辕北辙，如“东西”与其内部的汉字“东”和“西”，单纯从字向量很难推测出词的含义。为了解决这个问题，本文设计了双通道字词结合短文本分类模型，同时接受短文本的词向量表示与字向量表示，充分结合两者的特征信息，克服了常规分类模型的不足。

双通道字词结合短文本分类模型主要是构建一个包含两个平行的特征提取网络的分类模型，分别提取词向量特征与字向量特征，总体结构如图4-5所示。

模型总共分为三层：编码层、特征提取层与全连接层。编码层根据相应的词向量与字向量模型，将输入文本解析为词向量序列 W 和字向量序列 C 。特征提取

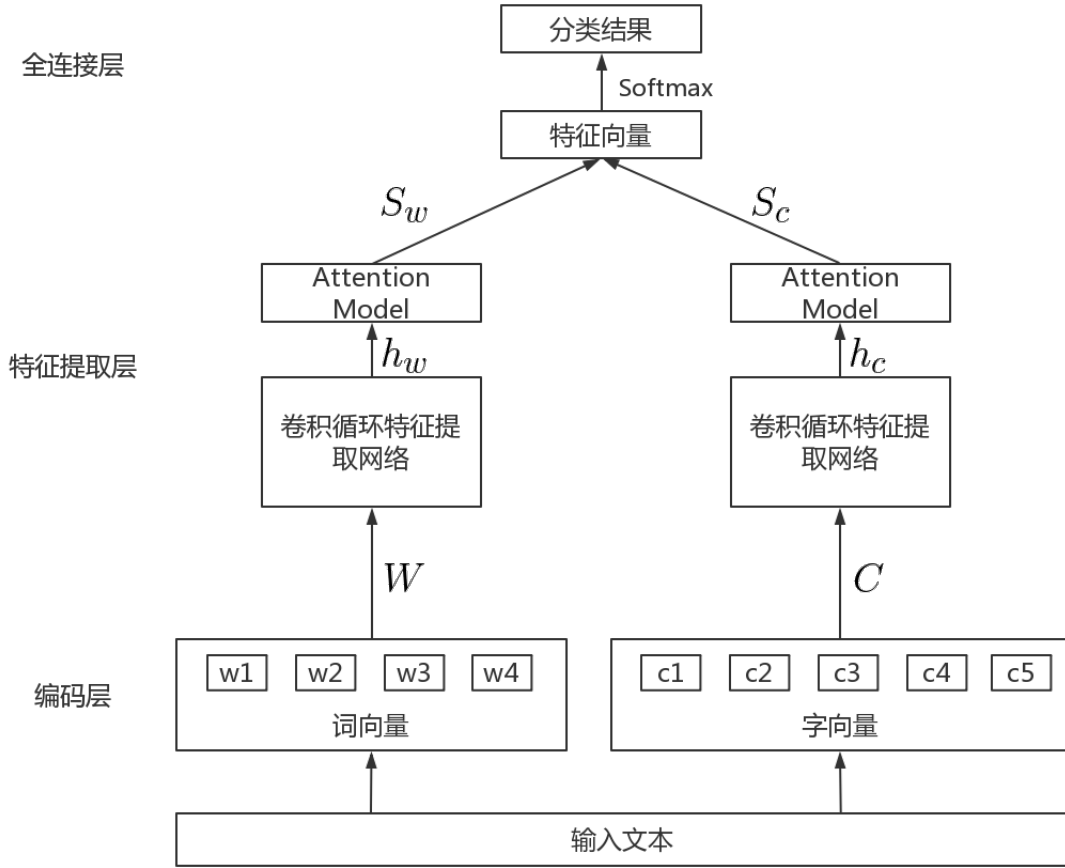


图 4-5 双通道字词结合短文本分类模型

层分为两个平行的神经网络模块，由前文中的 Attention-Based 卷积循环特征提取网络组成，分别提取词向量序列和字向量序列的文本特征，即向量 S_w 、 S_c ，然后根据公式4-8将其合并，得到最终的文本特征向量 S 。

$$S = [S_w \oplus S_c] \quad (4-8)$$

全连接层由线性转换层和 Softmax 层组成，其中线性转换层将特征向量 S 转换为一个维度与分类类别相当的实值向量，然后 Softmax 层将这些实值映射为最终的条件概率，计算公式如4-9所示。

$$P = \text{softmax}(W_s S + b_s) \quad (4-9)$$

本文使用公式4-10作为模型的损失函数，来最小化模型的分类误差。其中 N_t 表示训练语料库大小， N_c 表示类别数量， $P_j^s(s_i)$ 表示当前文本的真实类别为 j 的概

率，属于类别 j 则为 1，否则为 0。

$$Loss = - \sum_{i=1}^{N_t} \sum_{j=1}^{N_c} P_j^g(s_i) \cdot \log(P_j(s_i)) \quad (4-10)$$

4.4 实验设计和结果分析

为了验证 Attention-Based 字词结合卷积循环中文短文本分类网络模型的可行性及有效性，本文设计并实现了字向量/词向量两类总计 8 组对比实验模型，对新闻标题数据进行分类，并对结果进行分析。

4.4.1 实验语料数据

本实验所用的短文本数据由两部分组成，一部分来自“今日头条”网站，另一部分为清华大学 THUCTC 新闻语料^[34]。语料库总计包含 830396 条新闻标题数据，共分为财经、科技、体育等 12 个类别（具体信息如表4-1所示），长度都在 60 字以内。实验随机抽取 80% 数据作为模型训练数据集，另外 20% 则为验证集。

表 4-1 实验语料库分类详情

类别	数量	类别	数量
社会	57860	体育	134680
时政	62658	股票	149853
教育	41827	娱乐	92886
财经	41085	家居	32188
游戏	24313	房产	18440
时尚	13150	科技	161456

4.4.2 实验环境及相关工具

本实验模型采用机器学习库 Tensorflow 作为分类模型的实现工具。Tensorflow 是 Google 公司与 2015 年 11 月 9 日发布并宣布开源的机器学习框架，是现在最流行的深度学习项目之一。Tensorflow 支持 Linux 平台、Windows 平台、Mac 平台等多种主流平台，提供了非常丰富的深度学习相关 API，包括基本的向量矩阵计算、各种优化算法、各种卷积神经网络和循环神经网络基本单元的实现、以及可视化的辅助工具等等。Tensorflow 将所有用户输入的计算结构都视为数据流图（data flow graphs），图中的节点（nodes）表示数学操作，连接节点的线表示在节点间相互输送数据的多维数据数组，即张量（tensor）。Tensorflow 可以适应任何硬件环

境，无论台式机、服务器、手机移动设备都可以直接运行计算代码，并且拥有自动求微分功能，在模型训练时自动计算相关的微分导数，极大的简化了深度学习代码的开发。

本实验使用英伟达 GeForce 显卡作为模型训练的辅助工具。通过英伟达推出的 CUDA 技术（Compute Unified Device Architecture，统一计算架构），使用 GeForce 显卡能够在 GPU（GP GPU）上使用图形 APIs 进行传统通用计算，从而加速模型训练。

本实验具体的实验环境如表4-2所示：

表 4-2 实验环境具体配置

操作系统	Ubuntu 16.04
开发语言	Python 3.6
开发平台	Google Tensorflow 深度学习框架
CPU	Intel I5
内存	8G
硬盘	1T
显卡	NVIDIA GeForce 1080

4.4.3 分类模型评价方法

分类模型的有效性有多种评价方法，如准确率（Precision）、召回率（Recall）和 F1-Measure。对于分类模型的分类结果，总体可以分为 4 中情况，如表4-3所示。其中 TP 表示实际属于该类且被模型分到该类的样例数。FP 表示实际不属于该类，

表 4-3 分类结果示例表

	实际属于该类	实际不属于该类
被分到该类	TP	FP
未被分到该类	FN	TN

但被模型分到该类的样例数。FN 表示实际属于该类，但未被分到该类的样例数。TN 表示实际不属于该类，且未被分到该类的样例数。

根据表4-3，准确率计算公式可写为式4-11。

$$Accuracy = \frac{TP + TN}{TP + FP} \quad (4-11)$$

召回率可定义为式4-12。

$$Recall = \frac{TP + TN}{TP + FN} \quad (4-12)$$

准确率反映了模型分类结果的正确率，召回率反映了模型找到正确列表的能力，两者都可以体现分类模型有效性，但很多时候会相互矛盾。为了正确评价本实验中实现的所有模型，本文使用这两个的加权调和平均值作为评价标准，即 F1-Measure，计算公式如4-13所示。

$$F1 - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4-13)$$

4.4.4 对比实验设计

为了验证本文设计的模型对于短文本分类的效果，本实验设计了 4 组对比实验，每组又词向量分类模型与字向量分类模型两种，总计 8 个对比实验：

(1) 传统 LSTM 模型

模型结构与实现方式由文献 [35] 提供，词向量版本使用 Ansj 中文分词开源库以及 word2vec 工具将训练语料映射为 200 维的词向量文本数据，字向量版本使用 word2vec 工具将训练语料映射为 200 维的字向量文本数据。隐藏层的节点个数设置为 200。模型训练的学习率 (learn rate) 取 0.001，优化算法使用随机梯度下降算法 (Stochastic Gradient Descent, SGD)。分类器使用多类分类的逻辑回归分类器，使用 LSTM 模型隐藏层最后一个节点的输出值作为分类特征向量。训练循环次数为 50，批尺寸 (batch size) 为 200。

(2) 传统 CNN 模型

模型结构与实现方式同样由文献 [35] 提供，词向量版本使用 Ansj 中文分词开源库以及 word2vec 工具将训练语料映射为 200 维的词向量文本数据，字向量版本使用 word2vec 工具将训练语料映射为 200 维的字向量文本数据。卷积层使用大小为 3、4、5 的三类卷积核，每类卷积核的个数为 128。池化层使用最大池化算法。模型训练的学习率 (learn rate) 取 0.001，优化算法使用随机梯度下降算法 (Stochastic Gradient Descent, SGD)。分类器使用多类分类的逻辑回归分类器。训练循环次数为 50，批尺寸 (batch size) 为 200。

(3) 传统卷积循环网络模型 (CLSTM)

模型依据文献 [36] 实现，词向量版本使用 Ansj 中文分词开源库以及 word2vec 工具将训练语料映射为 200 维的词向量文本数据，字向量版本使用 word2vec 工具将训练语料映射为 200 维的字向量文本数据。卷积层使用大小为 3、4、5 的三类卷积核，每类卷积核的个数为 128。池化层使用最大池化算法。LSTM 层的隐藏节

点个数设置为 200。模型训练的学习率 (learn rate) 取 0.001, 优化算法使用随机梯度下降算法 (Stochastic Gradient Descent, SGD)。分类器使用多类分类的逻辑回归分类器。训练循环次数为 50, 批尺寸 (batch size) 为 200。

(4) Attention-Based 卷积循环网络模型 (Attention-CLSTM)

模型依据4.3.1小节的设计实现, 词向量版本使用 Ansj 中文分词开源库以及 word2vec 工具将训练语料映射为 200 维的词向量文本数据, 字向量版本使用 word2vec 工具将训练语料映射为 200 维的字向量文本数据。卷积层使用大小为 3、4、5 的三类卷积核, 每类卷积核的个数为 128。池化层使用 k-max 池化算法。LSTM 层的隐藏节点个数设置为 200。Attention 层隐藏参数向量长度为 200。模型训练的学习率 (learn rate) 取 0.001, 优化算法使用随机梯度下降算法 (Stochastic Gradient Descent, SGD)。分类器使用多类分类的逻辑回归分类器。训练循环次数为 50, 批尺寸 (batch size) 为 200。

本文设计的分类模型同时使用词向量文本数据与字向量文本数据, 都为 200 维。两个通道的卷积层都使用大小为 3、4、5 的三类卷积核, 每类卷积核的个数为 128, 池化层都使用 k-max 池化算法, LSTM 层的隐藏节点数目与 Attention 层的隐藏参数向量长度也都为 200。模型训练的学习率 (learn rate) 取 0.0001, 优化算法使用随机梯度下降算法 (Stochastic Gradient Descent, SGD)。分类器使用多类分类的逻辑回归分类器。训练循环次数为 50, 批尺寸 (batch size) 为 200。

4.4.5 实验结果和分析

本文对上一个小节设计的实验模型在新闻标题语料库上进行实验, 每个模型都训练 5 次并选取平均结果。实验对比结果如表4-4所示。从实验结果的 1 到 2 行

表 4-4 实验对比结果

模型	字向量	词向量
传统 CNN	0.855679	0.835943
传统 LSTM	0.869336	0.849313
CLSTM	0.894824	0.872391
Attention-Based CLSTM	0.89947	0.87346
双通道 Attention-Based CLSTM (本文设计)	0.913441	

中可以看出, 和字向量相比, 词向量在短文本分类任务中效果更好, 这表明词向量在文本特征表达上更加有效, 并且使用 LSTM 结构的循环神经网络比卷积神经网络更加适合提取文本特征。表格的第 3 行与第四行证明结合 CNN 与 RNN 网络能够有效提升模型对文本数据的表达能力。最后, 对比其余模型, 本文设计的双

通道 Attention-Based CLSTM 分类模型取得了最好的结果，证明了字向量文本特征能够很好的强化原本的词向量文本特征，从而让模型能够进一步的获取短文本的语义，有效提高短文本分类的准确率。

4.5 本章小结

本章主要提出了 Attention-Based 字词结合卷积循环中文短文本分类模型。首先介绍了用于提取文本特征的卷积循环神经网络，然后再分析了 Attention Model 的相关概念并引入卷积循环神经网络。之后加入字向量数据，设计了一个双通道的短文本分类模型，弥补了使用单一词向量/字向量的不足。最后通过对比实验，验证了模型的可行性及有效性。

第五章 短文本分类系统设计和实现

本章主要阐述短文本分类系统的功能设计与代码实现，并详细介绍系统包含的各个功能模块。整个系统可以分为离线网络训练与在线短文本分类两大部分，离线网络训练用于语料库更新以及分类模型学习与参数调优，在线短文本分类则是系统实际提供的功能，用于给外部系统调用。这两部分相互依赖，只有通过离线网络训练不断更新分类模型数据，在网络训练时将模型参数调整到最优，才能让在线分类系统获得最好的效果。

5.1 系统总体概述

该短文本分类系统设计的主要目的是给其他用户提供一个持续可靠的短文本分类服务，主要思想是依据使用领域不断通过网络爬虫搜集标注好的训练语料，训练出相应的词向量与字向量。然后系统使用这些文本表示信息训练分类网络，并根据在验证集上的表现动态调节相关参数让模型效果达到最优。最后使用训练得到的网络参数给前台用户提供短文本分类服务。系统的总体架构逻辑上大致可以分为三个子系统：训练语料更新子系统、分类模型训练子系统和短文本分类子系统，整个流程如图5-1所示。

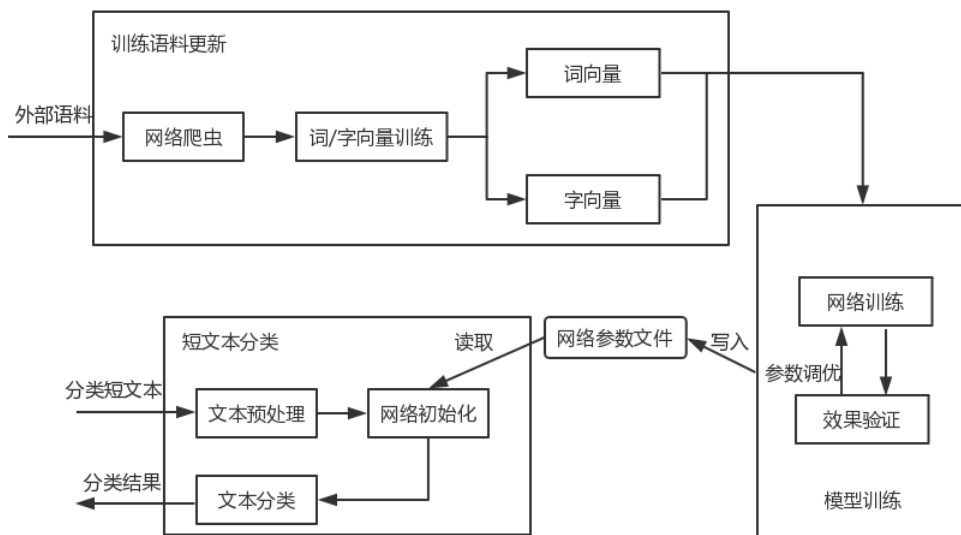


图 5-1 短文本分类系统流程

为了保证系统的运算速度以及服务的稳定性，后台的语料更新子系统和短文

本分类子系统都由包含多台计算机的集群提供服务，分类模型训练子系统则为高性能运算服务器，配备有能够加速矩阵、向量计算的 NVIDIA GeForce 显卡。系统的整体物理部署如图5-2所示。

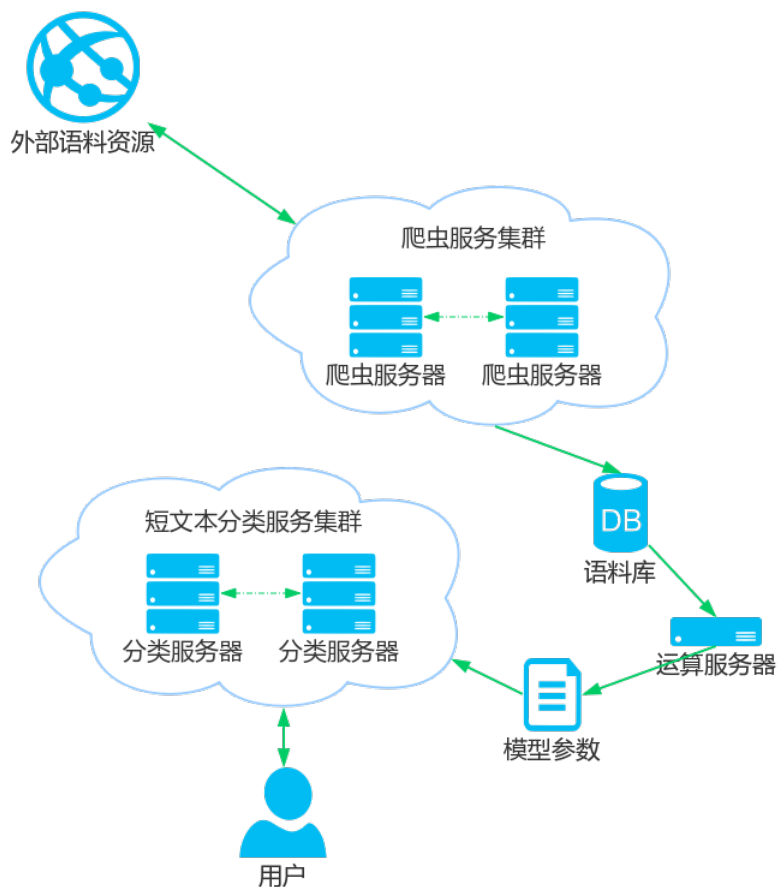


图 5-2 分类系统物理部署

5.2 系统各模块的设计和实现

系统各个子系统的功能模块如图5-3所示，训练语料更新子系统分为网络爬虫、队列管理、语料数据存储、词/字向量训练四个模块，模型训练子系统分为分类网络训练、参数调优两个模块，短文本分类子系统分为分类服务、日志记录两个模块。其中训练语料更新子系统和模型训练子系统为后台模块，由系统定时启动，用于实时更新系统的模型参数，短文本分类子系统为前台模块，用于与用户交互，给外部用户提供短文本分类服务。下面对每一个功能模块进行详细阐述。

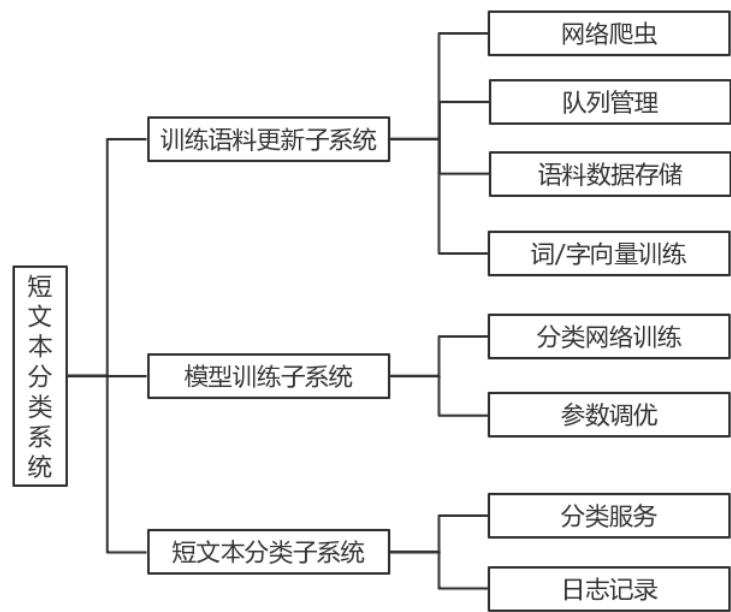


图 5-3 系统功能模块图

5.2.1 网络爬虫

网络爬虫模块采用分布式爬虫架构，由多个网络节点中的爬虫程序共同完成数据抓取任务。爬虫程序基于 Python 的 Scrapy 框架开发，该框架是一个快速、高层次的屏幕抓取和 web 抓取框架，用于抓取 web 站点并从页面中提取结构化的数据，并且轻量灵活，能够快速的二次开发，适应不同爬虫需求。

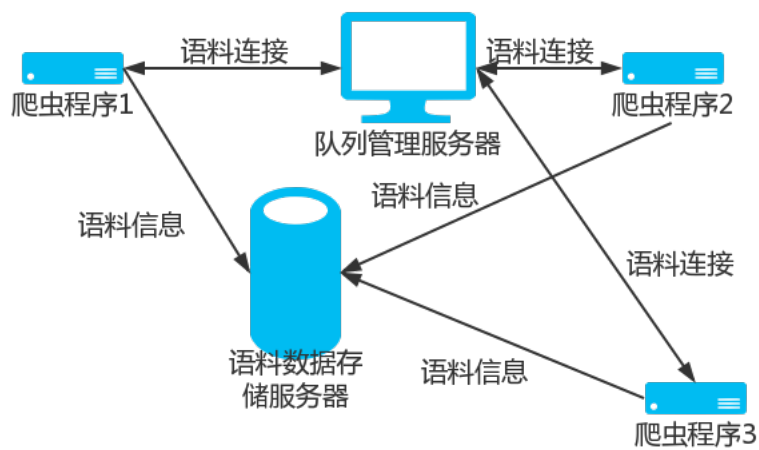


图 5-4 网络爬虫模块结构图

爬虫模块的整体结构如图5-4所示，爬虫任务由队列管理模块分发，爬虫程序

负责数据的获取，然后将获取的数据写入语料数据存储模块。每个爬虫程序包含 4 个组件：

- 下载器：负责通过 http/https 协议从外部 WEB 服务器下载 WEB 页面，以便后续的处理。
- 页面解析器：负责解析下载的 WEB 页面，抽取所需的信息，以及发现新的语料链接并发送到队列管理模块。
- 存储器：负责将获取的有用信息存储到语料数据存储模块。
- IP 池：负责给下载器提供代理 IP，防止外部 WEB 服务器感知并禁止爬虫模块的请求。

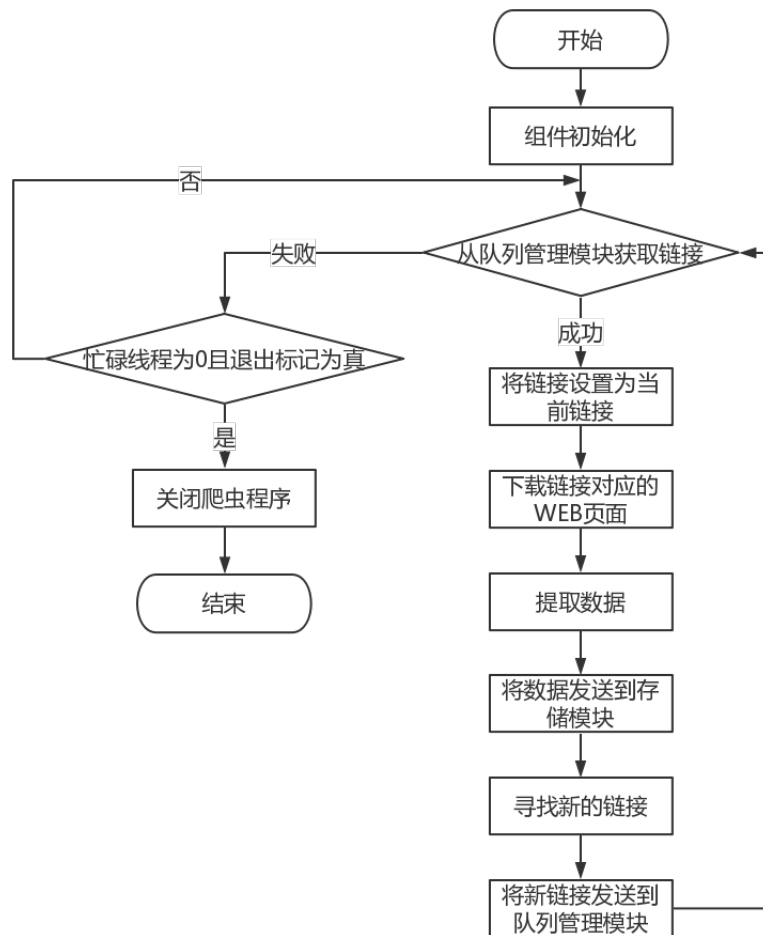


图 5-5 爬虫程序运行流程

爬虫程序运行流程如图5-5所示，首先，程序对各个组件进行初始化工作，如生成下载器、设置线程池线程数量等。初始化结束之后，程序的主线程开始尝试从队列管理模块获取下载链接，如果获取成功，则生成一个 Task 任务块并将其设

置为当前任务，然后从线程池中取出一个线程进行下一步的处理，如果获取失败，则认为当前任务可能已经结束，并检测退出标志是否为真以及是否还有工作中的线程，两者都为真时表示系统已经确定退出，程序会立即关闭线程池，结束工作。子线程接收到 Task 任务块会立刻运行下载器下载对应的 WEB 页面，下载完成后则通过页面解析器进行解析，然后将解析得到的所需数据以及新的连接分别发送到语料信息存储模块和队列管理模块。爬虫程序的核心页面处理代码如下所示：

```
def parse(self, response):
    news = json.loads(response.body.decode('utf-8'))
    if news["message"] == "error":
        # 服务器返回错误数据，当前IP地址可能被禁止
        if "proxy" in response.meta:
            self.ban_ip[response.meta["proxy"]] = 1
        else:
            self.ban_ip[""] = 1
        print("ban_ip: " + str(len(self.ban_ip)) + "/" +
              str(len(IP_POOL)))
        if len(self.ban_ip) >= len(IP_POOL):
            raise CloseSpider("ban")
        yield Request(response.url, dont_filter=True)
    else:
        # 获取新闻语料
        news_list = news["data"]
        for news_item in news_list:
            item = NewsSpiderItem()
            if "chinese_tag" not in news_item or news_item[
                "chinese_tag"] == "其他" or \
                news_item["chinese_tag"] == "视频":
                continue
            item["tag"] = news_item["chinese_tag"]
            item["title"] = news_item["title"]
            item["hash"] = self.__get_md5(news_item["title"]
                                           ).encode("utf-8"))
        # 返回获取的数据
```

```

        yield item
        next_time = news["next"]["max_behot_time"]
        sign = lib.get_sign(next_time)
        ascp_dict = lib.get_as_cp()

# 返回新发现的链接
yield Request(self.api_format % (self.__get_url_tag
                                (response.url), next_time, next_time,
                                ascp_dict["as"],
                                ascp_dict['cp'], sign))

```

5.2.2 队列管理

队列管理模块主要用来给整个爬虫网络提供一个可靠的任务管理服务，统一分发数据爬取任务，平衡各节点的工作负担，负载均衡，防止忙碌节点的出现。模块以稳定性、实时性较好的 Redis (REmote DIctionary Server) 分布式数据库为基础实现，通过将爬虫程序上传的新链接重新分配给网络中的其他节点，不断推进整体的数据爬取进度，工作示意图大致如图5-6所示。

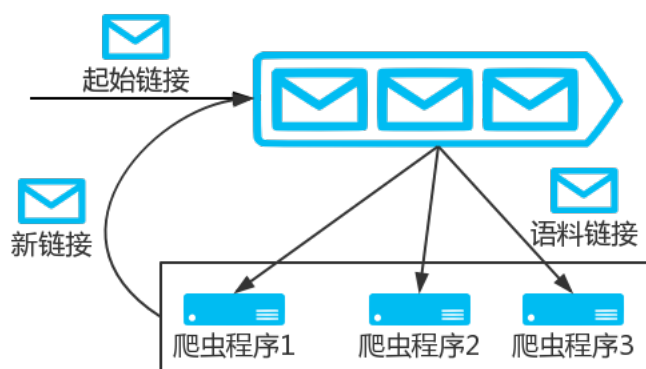


图 5-6 队列管理模块工作示意图

5.2.3 语料数据存储

语料数据存储模块用于持久化保存爬虫程序提取的语料信息，使用 Mysql 数据库实现。为了获取语料的全面信息，语料数据存储模块除了存储语料文本，还

存储了来源、URL 链接、发布时间、爬取时间、文本 MD5 等重要信息，以新闻标题语料为例，相关数据格式如表5-1所示，爬取的相应数据见表5-2所示。

表 5-1 新闻标题语料数据格式

title	新闻的标题	url	新闻的 url 地址
time	新闻的发布时间	get_time	新闻的爬取时间
resource	新闻的来源	news_id	新闻的 id
news_md5	新闻标题文本的 md5 值	category	新闻所属的类别

表 5-2 爬取的新闻标题数据

title	男子出狱创业招 500 名犯人，为帮员工戒毒半年吃素
url	http://toutiao.com/group/6504029172468285966/
time	1511756299
get_time	1511759149
category	社会
resource	人民网
news_id	6504029172468285966
news_md5	1ce5761e192c42e76e4bb45ac22be5ed

同时，为了后续模块能够方便的处理爬取的语料数据，存储模块还需要对这些数据进行预处理工作，包括标点符号剔除、分词处理、繁简转换、分词筛选、去除停用词、汉字统计。其中标点符号剔除、分词处理、繁简转换和之前小节介绍的一样，这里不再赘述，分词筛选是用于统计语料中发现的新词，汉字统计则是统计新汉字，这两者是判断是否需要更新词向量/字向量的重要指标。去除停用词使用的是哈工大、百度、川大的停用词列表，能够有效去除语料中对语义特征没有帮助的词。同样以新闻标题数据为例，预处理之后的数据如表5-3所示。

表 5-3 预处理过后的新闻数据

id	1303086993
news_id	6504029172468285966
words	男子, 出狱, 创业, 招, 犯人, 帮, 员工, 戒毒, 半, 年, 吃, 素
char	男, 子, 出, 狱, 创, 业, 招, 犯, 人, 帮, 员, 工, 戒, 毒, 半, 年, 吃, 素

5.2.4 字/词向量训练

字/词向量训练模块主要用来获取分类模型使用的词向量与字向量，采用增量训练的方式，只有在语料数据或新词/新字数量超过一定阈值的时候，才开始训练，更新当前使用的词/字向量。

由3.3.2节的目标公式以及负采样算法可以继续推导得式5-1：

$$L = \sum_{w \in I} \sum_{u \in \{w\} \cup NEG(w)} \left\{ L^w(u) \cdot \log \left[\sigma \left(X_w^\top \theta^u \right) \right] + [1 - L^w(u)] \cdot \log [1 - \sigma \left(X_w^\top \theta^u \right)] \right\} \\ + \sum_{c \in I} \sum_{u \in \{c\} \cup NEG(c)} \left\{ L^c(u) \cdot \log \left[\sigma \left(X_c^\top \theta^u \right) \right] + [1 - L^c(u)] \cdot \log [1 - \sigma \left(X_c^\top \theta^u \right)] \right\} \quad (5-1)$$

再根据式5-1分别对各个参数求导，即可得到词向量/字向量的梯度更新方法，如公式所示5.2.4。

$$q = \sigma \left(X_w^\top \theta^u \right) \\ g = \eta \left(L^w(u) - q \right) \\ e := e + g \theta^u \\ \theta^u := \theta^u + g X_w \\ v_u := v_u + e \quad (5-2)$$

词/向量训练代码由 C 语言实现，核心部分分为正向推导、梯度计算、反向更新三个部分。正向推导主要是从模型底部统计所有输入向量，得到当前预测的隐藏层向量，对应的代码如下所示：

```
// 叠加窗口内的词向量
for(c = 0; c < layer1_size; c++)
    neu1[c] += syn0[c + last_word * layer1_size];
// 叠加字向量
for(c = 0; c < vocab[last_word].character_size; c++)
{
    char_id = vocab[last_word].character[c];
    char_id_list[char_list_cnt++] = char_id;
    for(d = 0; d < layer1_size; d++) neucomp[d] +=
        synchar[d + char_id * layer1_size];
    for(d = 0; d < char2comp[char_id].comp_size; d++)
    {
```

```

        comp_id = char2comp[char_id].comp[d];
        if(comp_id != char_id)
        {
            comp_id_list[comp_list_cnt++] = comp_id;
            // 叠加部首向量
            for(e = 0; e < layer1_size; e++) neucomp[e]
                += synchar[e + comp_id * layer1_size];
        }
    }
}

```

梯度计算部分是根据公式计算当前输入对应的梯度，代码如下所示。

```

for(d = 0; d < negative + 1; d++)
{
    // 一些初始化操作
    ...
    // 开始计算梯度
    l2 = target * layer1_size;
    real f1 = 0, f3 = 0, g1 = 0, g3 = 0;
    for(c = 0; c < layer1_size; c++)
    {
        f1 += neu1[c] * synlneg[c + l2];
        f3 += neucomp[c] * synlneg[c + l2];
    }
    // f1
    if(f1 > MAX_EXP)
        g1 = (label - 1) * alpha;
    else if(f1 < -MAX_EXP)
        g1 = (label - 0) * alpha;
    else
        g1 = (label - expTable[(int)((f1 + MAX_EXP) * (
            EXP_TABLE_SIZE / MAX_EXP / 2))]) * alpha;
}

```

```

//f3
if(f3 > MAX_EXP)
    g3 = (label - 1) * alpha;
else if(f3 < -MAX_EXP)
    g3 = (label - 0) * alpha;
else
    g3 = (label - expTable[(int) ((f3 + MAX_EXP) * (
        EXP_TABLE_SIZE / MAX_EXP / 2))]) * alpha;
// 统计梯度
for(c = 0; c < layer1_size; c++)
{
    neu1_grad[c] += g1 * synlneg[c + 12];
    neucomp_grad[c] += g3 * synlneg[c + 12];
}
//更新 synlneg
for(c = 0; c < layer1_size; c++)
    synlneg[c + 12] += g1 * neu1[c] + g3 * neucomp[c];
}

```

反向更新部分则是用上个部分得到的梯度更新输入的词向量与字向量及部首向量，完成一次训练，相关代码如下所示：

```

for(a = b; a < window * 2 + 1 - b; a++)
    if(a != window)
    {
        c = sentence_position - window + a;
        if(c < 0) continue;
        if(c >= sentence_length) continue;
        last_word = sen[c];
        if(last_word == -1) continue;
        for(c = 0; c < layer1_size; c++)
        {
            //更新词向量
            syn0[c + last_word * layer1_size] += neu1_grad[

```

```

        c] / cw;
    }
}
for(a = 0; a < char_list_cnt; a++)
{
    char_id = char_id_list[a];
    for(c = 0; c < layer1_size; c++)
    {
        //更新字向量
        synchar[c + char_id * layer1_size] +=
            neucomp_grad[c] / (char_list_cnt +
                comp_list_cnt);
    }
}
for(a = 0; a < comp_list_cnt; a++)
{
    comp_id = comp_id_list[a];
    for(c = 0; c < layer1_size; c++)
    {
        //更新部首向量
        synchar[c + comp_id * layer1_size] +=
            neucomp_grad[c] / (char_list_cnt +
                comp_list_cnt);
    }
}
}

```

5.2.5 模型训练

模型训练子系统包含分类网络训练与参数调优两个模块，是整个系统的核心部分，用于训练并获取分类性能最优的网络模型参数。详细过程设计如图5-7所示。

模型训练子系统的详细流程如下：

1. 根据配置文件中的相关配置信息使用 Google Tensorflow 深度学习框架构建第四章中提出的短文本分类网络结构。

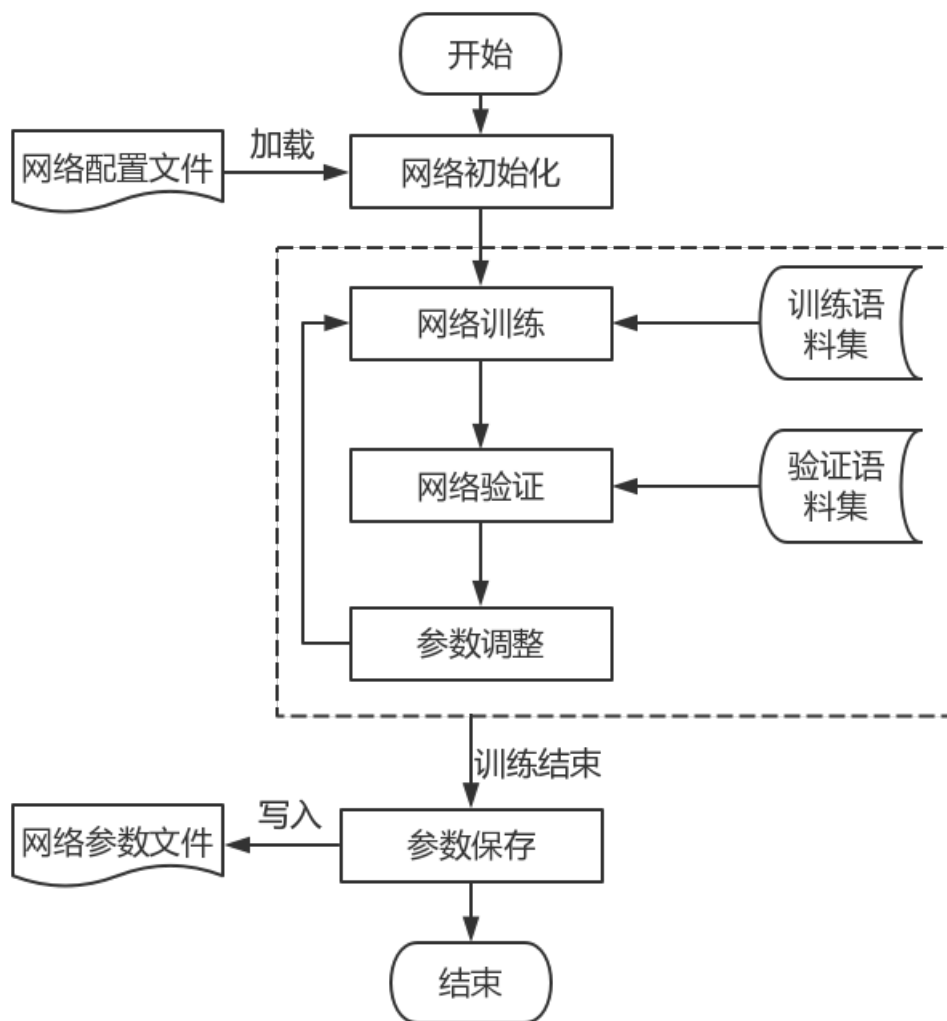


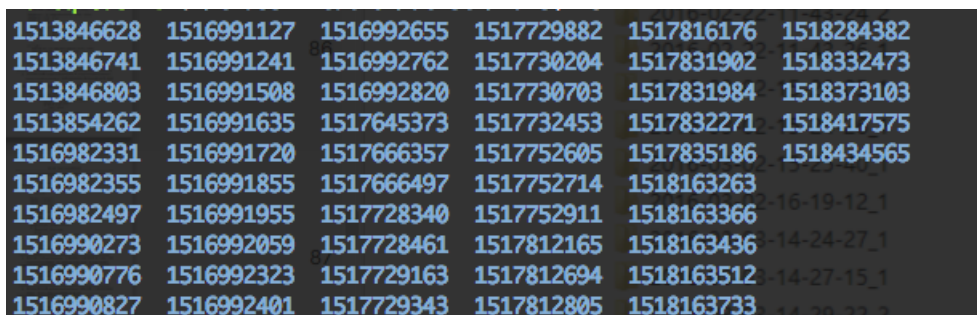
图 5-7 模型训练子系统过程示意图

2. 使用误差反向传播算法（Error Back Propagation, BP）不断进行前向计算与后向传导，更新网络参数。
3. 使用验证语料集对训练好的网络进行测试，确定网络的性能以及判断网络是否正确收敛。验证语料集不参与网络训练，仅仅在其上进行 BP 算法的前向过程。
4. 根据相关配置文件，在一定范围内微调网络的参数并使用新参数重新训练玩了个，尝试优化网络。在调整一定次数之后，取性能最好的网络参数作为最后的有效参数。
5. 保存完成调优之后的网络参数，以便后续阶段使用。

表5-4展示了需要调节的网络参数及其调节范围，参数微调时会在范围内对相关参数重新取值。图5-8展示了系统保存的网络参数，训练过程中产生所有模型参数都会被保存并统一编号，方便日后管理人员分析。

表 5-4 微调参数列表

参数名称	调节范围
LSTM 隐藏层大小	64-256
CNN 卷积核数量	64-256
CNN 卷积核高度	2-10
Dropout 层丢弃率	0.2-0.6
学习率	0.001-0.0001
Attention 层大小	65-256



1513846628	1516991127	1516992655	1517729882	1517816176	1518284382
1513846741	1516991241	1516992762	1517730204	1517831902	1518332473
1513846803	1516991508	1516992820	1517730703	1517831984	1518373103
1513854262	1516991635	1517645373	1517732453	1517832271	1518417575
1516982331	1516991720	1517666357	1517752605	1517835186	1518434565
1516982355	1516991855	1517666497	1517752714	1518163263	-16-19-12_1
1516982497	1516991955	1517728340	1517752911	1518163366	-14-24-27_1
1516990273	1516992059	1517728461	1517812165	1518163436	-14-27-15_1
1516990776	1516992323	1517729163	1517812694	1518163512	-14-27-15_1
1516990827	1516992401	1517729343	1517812805	1518163733	-14-27-15_1

图 5-8 模型保存的网络参数文件

5.2.6 分类服务

分类服务模块主要是给外部用户提供一个持续有效的即时短文本分类服务，整个服务依托于 HTTP 协议，通过传递 Json 进行数据交换，外部用户只要按照格式要求给系统发送 JSON 数据，系统即会返回分类结果。服务网络由多台服务节

点组成，并使用 Nginx 反向代理服务器提供链接分发服务，保证系统的稳定性以及可拓展性，具体框架如图5-9所示。

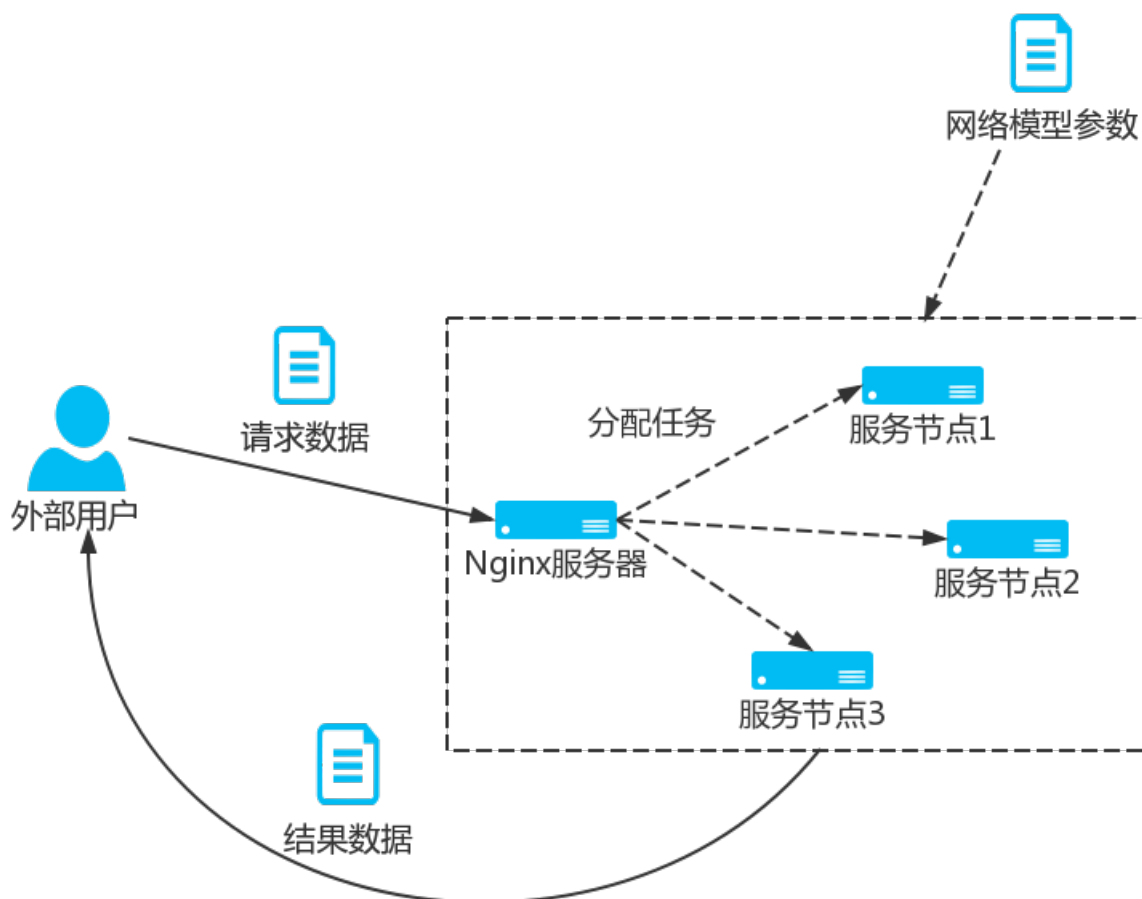


图 5-9 分类服务模块基本框架示意图

用户的分类请求首先发往 Nginx 代理服务器，由代理服务器统一分配服务节点。服务节点接收到分类文本之后，需要对其分词、去除标点与停用词，然后按照词/字向量表转换为对应的词向量/字向量数据，同时忽略词表/字表中没有词或字。最后将数据送入分类网络，并将分类结果返回给用户。

5.2.7 日志记录

日志记录模块用于记录外部用户的相关信息，记录内容包括用户编号、用户 IP、分类文本等，同时系统的分类结果也会被保存，方便以后的查阅以及错误排

查。日志存储模块使用 Mongodb 作为存储数据库。Mongodb 与常见的 Mysql 数据库等关系数据库不同，采用非关系结构存储数据，同时侧重大量数据写入性能，非常适合日志信息存储。用户日志的具体信息如表5-5所示。

表 5-5 用户日志字段

字段名称	字段值
_id	ObjectId("41d2c2913761a2258c7c4e79")
user_id	dca8815bef5e1150
time	2016-01-27 05:55
model_arg	1517812805
content	如果一个人临去世之前把信用卡全部套现，银行会怎么办？
classification_result	财经

5.3 系统测试与分析

为了验证系统的有效性以及在最新短文本语料数据上的分类效果，需要在实际网络环境中进行系统测试，并对系统的测试结果进行统计、分析。本文将在今日头条新闻网站的标题数据上对系统进行实际的测试。

5.3.1 新闻标题数据搜索

5.3.2 汉字信息库

5.3.3 分类测试结果

5.4 本章小节

参考文献

- [1] G. Song, Y. Ye, X. Du, et al. Short text classification: A survey[J]. ISSN 1796-2048 Volume 9, Number 5, May 2014, 2014, 9(5): 635
- [2] S. Zelikovitz, F. Marquez. Transductive learning for short-text classification problems using latent semantic indexing[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2005, 19(02): 143-163
- [3] M. Chen, X. Jin, D. Shen. Short text classification improved by learning multi-granularity topics[C]. IJCAI, 2011, 1776-1781
- [4] J. M. Cabrera, H. J. Escalante, M. Montes-y Gómez. Distributional term representations for short-text categorization[C]. International Conference on Intelligent Text Processing and Computational Linguistics, 2013, 335-346
- [5] X. Feng, Y. Shen, C. Liu, et al. Chinese short text classification based on domain knowledge.[C]. IJCNLP, 2013, 859-863
- [6] R. Socher, B. Huval, C. D. Manning, et al. Semantic compositionality through recursive matrix-vector spaces[C]. Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, 2012, 1201-1211
- [7] P. Blunsom, E. Grefenstette, N. Kalchbrenner. A convolutional neural network for modelling sentences[C]. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, 655-665
- [8] Y. Chen. Convolutional neural network for sentence classification[C]. Proceedings of the 2014 Conference on EMNLP, 2014, 1746--1751
- [9] S. Lai, L. Xu, K. Liu, et al. Recurrent convolutional neural networks for text classification.[C]. AAAI, 2015, 2267-2273
- [10] X. Zhang, J. Zhao, Y. LeCun. Character-level convolutional networks for text classification[C]. Advances in neural information processing systems, 2015, 649-657
- [11] Y. Li, W. Li, F. Sun, et al. Component-enhanced chinese character embeddings[C]. Proceedings of the 2015 Conference on EMNLP, 2015, 829--834
- [12] M. J. Er, Y. Zhang, N. Wang, et al. Attention pooling-based convolutional neural network for sentence modelling[J]. Information Sciences, 2016, 373: 388-403

- [13] Z. Yang, D. Yang, C. Dyer, et al. Hierarchical attention networks for document classification.[C]. HLT-NAACL, 2016, 1480-1489
- [14] 张劲松, 袁健. 回溯正向匹配中文分词算法 [J]. 计算机工程与应用, 2009, 45(22): 132-134
- [15] 李娟, 周贤善. 一种改进的逆向匹配快速切分算法 [J]. 信息系统工程, 2010, 1(2): 133-134
- [16] 陈耀东, 王挺. 基于有向图的双向匹配分词算法及实现 [J]. 计算机应用, 2005, 25(06): 1442-1444
- [17] 薛苏琴, 牛永洁. 基于向量空间模型的中文文本相似度的研究 [J]. 电子设计工程, 2016, 24(10): 28-31
- [18] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification[C]. AAAI-98 workshop on learning for text categorization, 1998, 41-48
- [19] G. E. Hinton. Learning distributed representations of concepts[C]. Proceedings of the eighth annual conference of the cognitive science society, 1986, 12
- [20] T. Mikolov, I. Sutskever, K. Chen, et al. Distributed representations of words and phrases and their compositionality[C]. Advances in neural information processing systems, 2013, 3111-3119
- [21] Y. LeCun, B. Boser, J. S. Denker, et al. Backpropagation applied to handwritten zip code recognition[J]. Neural computation, 1989, 1(4): 541-551
- [22] S. Hochreiter, J. Schmidhuber. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780
- [23] X. Chen, L. Xu, Z. Liu, et al. Joint learning of character and word embeddings.[C]. IJCAI, 2015, 1236-1242
- [24] J. Xu, J. Liu, L. Zhang, et al. Improve chinese word embeddings by exploiting internal structure[C]. Proceedings of NAACL-HLT, 2016, 1041-1050
- [25] Y. Sun, L. Lin, N. Yang, et al. Radical-enhanced chinese character embedding[C]. International Conference on Neural Information Processing, 2014, 279-286
- [26] J. Yu, X. Jian, H. Xin, et al. Joint embeddings of chinese words, characters, and fine-grained sub-character components[C]. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, 286-291
- [27] 苏文正. 汉字简化问题研究 [D]. 兰州: 兰州大学, 2007, 53
- [28] 胡浩, 李平, 陈凯琪. 基于汉字固有属性的中文字向量方法研究 [J]. 中文信息学报, 2017, 31(3): 32-40

- [29] T. Mikolov, W.-t. Yih, G. Zweig. Linguistic regularities in continuous space word representations[C]. Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013, 746-751
- [30] J. Schmidhuber. Deep learning in neural networks: An overview[J]. Neural networks, 2015, 61: 85-117
- [31] N. Kalchbrenner, E. Grefenstette, P. Blunsom. A convolutional neural network for modelling sentences[C]. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, 655-665
- [32] K. Xu, J. Ba, R. Kiros, et al. Show, attend and tell: Neural image caption generation with visual attention[C]. International Conference on Machine Learning, 2015, 2048-2057
- [33] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks[C]. Advances in neural information processing systems, 2015, 2440-2448
- [34] J. Li, M. Sun. Scalable term selection for text categorization[J]. EMNLP-CoNLL 2007, 2007, 1: 774
- [35] Y. Zhou, B. Xu, J. Xu, et al. Compositional recurrent neural networks for chinese short text classification[C]. Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on, 2016, 137-144
- [36] C. Zhou, C. Sun, Z. Liu, et al. A c-lstm neural network for text classification[J]. Computer Science, 2015, 1(4): 39-44