



《自然语言处理》课程设计

新闻文本分类

组长：李页霆 201628015029011

组员：

金融通 201628015029007

王文惠 201628015029020

二〇一七 年 七 月 十 四 日

目录

第一章 绪论.....	3
第二章 获取语料及预处理.....	4
一.语料获取.....	4
二.语料预处理.....	5
第三章 特征工程.....	6
一. 文本表示.....	6
1. 向量空间模型.....	6
2. 分布式表示.....	6
二. 向量空间模型的特征选择.....	7
第四章 分类器概述.....	8
一、传统的机器学习方法.....	8
二. 深度神经网络.....	10
第五章 实验分析.....	13
一. 实验环境.....	13
二. 实验结果.....	13
三、实验分析.....	18
第六章 总结.....	19
参考文献.....	20
附录 成员分工.....	21

第一章 绪论

随着信息技术的快速发展，网络新闻成为人们重要的信息来源之一，而网络新闻信息量大、组织混乱等特点阻碍着人们信息的获取。[1]传统的人工分类整理的方法变得几乎不可能，如何有效地组织和管理这些信息，并快速、准确、全面地从中找到用户所需要的信息是当前信息科学和技术领域面临的一大挑战。因此从 Web 新闻网页中挖掘出有用的知识已经成为当前研究的热点问题，将 Web 新闻网页进行解析爬取，在此基础上进行机器自动文本分类凸显出其重要作用。

基于统计学习的文本分类系统作为处理和组织大量文本的关键技术，能够在给定的分类模型下，根据文本的内容自动对文本分门别类，自动生成便于用户使用的文本分类系统，从而更好地帮助人们组织文本、挖掘文本信息，方便用户准确地定位所需的信息和分流信息。文本分类系统首先通过在预先分类好的文本集上训练，建立一个判别规则或分类器，从而对未知类别的新样本进行自动分类，将一篇文档归入预先定义的几个类别中的一个或几个。大量的结果表明它的分类精度比得上专家手工分类的结果，并且它的学习不需要专家干预，能适用于任何领域的学习，使得它成为目前文本分类的主流方法。[2]

当前文本分类的统计学习方法主要分为基于传统的机器学习算法（如朴素贝叶斯、逻辑回归等）和基于深度神经网络的分类方法（如 CNN、LSTM 等）两类。在中文文本分类问题中，两类算法的第一步都是分词，本文在实验中运用 jieba 分词工具进行分词并去除停用词以及数字等无用信息。传统机器学习方法的特征一般为词的词频、文档频率以及词频-逆文档频率等，涉及特征降维和特征选择；而近年来，神经网络最常用的特征为词向量，词向量在一定程度上能够反映语义信息。

本次实验的目标是将不同的机器学习算法应用于新闻文本分类问题，比较它们的性能，理解并进一步探索这个问题。本次实验中，从凤凰新闻、腾讯新闻爬取 13 类新闻，包括汽车、娱乐、金融、房产、军事以及体育等，使用了包含 9.8 万篇文本的新闻文本集合。在数据的预处理和特征工程之后，将 Naïve Bayes, SVM, CNN 等分类器应用于样本，并且比较它们的性能。数据预处理、特征工程、分类器设计等都使用 Python 语言完成，在实验过程中主要用到的机器学习包有 sklearn、gensim 等。

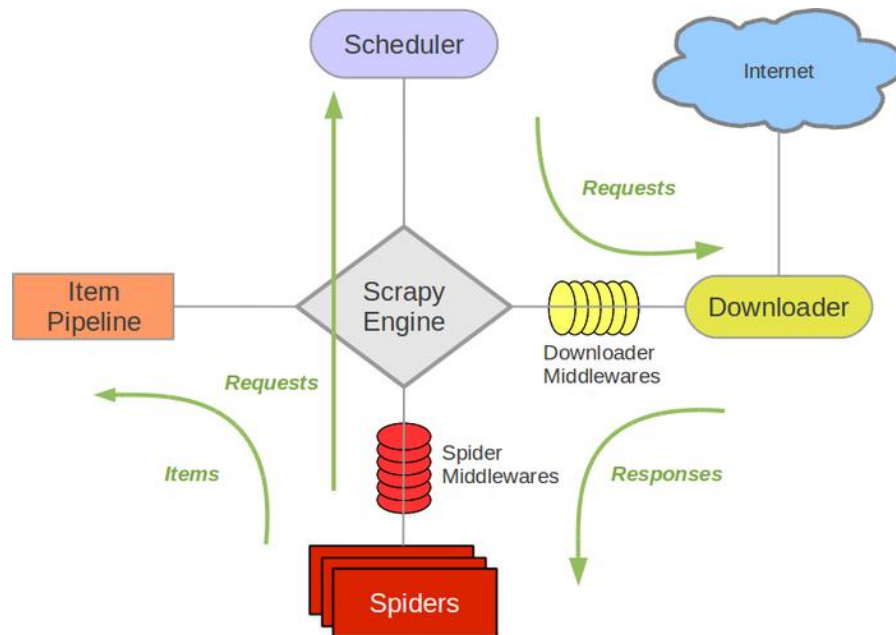
项目报告组织如下：第二章，简要介绍语料获取及预处理的工作；第三章，主要介绍特征工程；第四章，简单介绍各个分类器的基本原理；第五章，从准确度和速度的角度对机器学习方法进行实验分析并比较实验结果；第六章：总结。附录部分为成员分工情况。

第二章 获取语料及预处理

一. 语料获取

本实验的语料主要通过编写网络爬虫对腾讯新闻、凤凰新闻等进行爬取。

网络爬虫采用 Scrapy 框架，Scrapy 使用 Twisted 这个异步网络库来处理网络通讯，架构清晰，并且包含了各种中间件接口，可以灵活的完成各种需求。整体架构如下图所示：



Scrapy Engine

引擎负责控制数据流在系统中所有组件中流动，并在相应动作发生时触发事件。

调度器 (Scheduler)

调度器从引擎接受 request 并将他们入队，以便之后引擎请求他们时提供给引擎。

下载器 (Downloader)

下载器负责获取页面数据并提供给引擎，而后提供给 spider。

Spiders

Spider 是 Scrapy 用户编写用于分析 response 并提取 item(即获取到的 item)或额外跟进的 URL 的类。每个 spider 负责处理一个特定(或一些)网站。

Item Pipeline

Item Pipeline 负责处理被 spider 提取出来的 item。典型的处理有清理、验证及持久化(例如存取到数据库中)。

本实验对凤凰网、腾讯网的文化、历史、军事、娱乐、体育、科技等进行爬取，爬取内容包括每篇新闻的标题、内容、url 等，爬取完成后存入 MySQL 数据库中，等待进一步处理。

二. 语料预处理

爬取后新闻也无法直接用于模型训练。因此，我们都要对此进行预处理。

预处理包括格式转换，分词，去除停用词等步骤。

分词采用 **jieba** 分词，分词过程用从互联网中找到的 **2600** 多个停用词对停用词进行过滤，最终将格式变为每一行为分词后的内容+ `__label__` 类别。

本实验最后一共得到 **13** 类新闻，包括汽车、文化、教育、娱乐、健康、历史、房产、IT、招聘、军事、政治、体育、旅游共 9.8 万篇新闻。语料一共 400M，分完词后为 **1.4G**。最后每一个类别按 **8: 2** 的比例分为训练集和数据

第三章 特征工程

简单地说，特征工程是能够将数据像艺术一样展现的技术，是将数据属性转换为数据特征的过程，属性代表了数据的所有维度。通过特征工程对数据进行预处理，能够减少模型受噪声的干扰。在现实世界中，数据通常是复杂冗余，富有变化的，有必要从原始数据发现有用的特性。人工选取出来的特征依赖人力和专业知识，不利于推广。于是我们需要通过机器来学习和抽取特征，促进特征工程的工作更加快速、有效。[2]特征选择通常选择与类别相关性强、且特征彼此间相关性弱的特征子集，具体特征选择算法通过定义合适的子集评价函数来体现。

一. 文本表示

文档的内容与其包含的词有着必然的联系，同一类文档之间存在多个共同的词，而不同类的文档所包含的词之间差异很大。进一步的，不光是包含哪些词很重要，这些词出现的次数对分类也很重要。这一前提使得向量空间模型（VSM）成了适合文本分类问题的文档表示模型。在这种模型中，一篇文章被看作特征项集合，利用加权特征项构成向量进行文本表示。它实现起来比较简单，并且分类准确度也高，能够满足一般应用的要求。向量空间模型存在两个缺点：1）容易受维数灾难的困扰，尤其是将其用于 Deep Learning 的一些算法时；（2）不能很好地刻画词与词之间的相似性。为了克服这些缺点，引入了分布式表示方法——Word Vector 及 Paragraph Vector。

1. 向量空间模型

- 词袋模型（Bag-Of-Word，简记 BOW）

有两种表示方法，一是词典中的词只要出现就是 1，否则是 0；二是每个特征项的取值等于该词在该文本中出现的次数。

- 词频（Term Frequency, TF）

$$w_{ki} = tf_{ki}$$

- 逆文档频率（Inverse Document Frequency, IDF）

$$w_i = \log\left(\frac{N}{df_i}\right)$$

- TF-IDF

$$w_{ki} = tf_{ki} \times \log\left(\frac{N}{df_i}\right)$$

2. 分布式表示

- Word Vector

Word2vec 采用一个三层的神经网络，对语言模型进行建模，同时也能够获得单词在向量空

间的表示，而这个副产品才是 Word2vec 的真正目标。通过训练将某种语言中的每一个词映射成一个固定长度的短向量（当然这里的“短”是相对于 one-hot-representation 的“长”而言的），将所有这些向量放在一起形成一个词向量空间，而每一向量则为该空间中的一个点，在这个空间上引入“距离”，则可以根据词之间的距离来判断它们之间的（词法、语义上的）相似性了，可以为文本数据寻求更加深层次的特征表示。[4] 该方法的核心技术是根据词频用 Huffman 编码，使得所有词频相似的词隐藏层激活的内容基本一致，出现频率越高的词语，他们激活的隐藏层数目越少，这样有效的降低了计算的复杂度。[5]

- Paragraph Vector

Word2vec 在获得词向量后，对词向量进行平均处理，最终获取到句子向量。然后，利用机器学习的分类算法进行预测。这种方法在微博等短文上的应用效果不错，这是因为微博通常只有十几个单词，所以即使经过平均化处理仍能保持相关的特性。但是分析段落数据时，如果忽略上下文和单词顺序的信息，将会丢掉许多重要的信息。即上述的 word2vec 只是基于词的维度进行“语义分析”的，并不具有上下文的“语义分析”能力。Quoc Le 和 Tomas Mikolov 提出了 Doc2Vec 方法，Doc2vec 是在 Word2vec 的基础上发展而来的，它可以将一段句子表征为实数值向量 Paragraph Vector。除了增加一个段落向量以外，这个方法几乎等同于 Word2Vec。和 Word2Vec 类似，该模型也存在两种方法：Distributed Memory(DM) 和 Distributed Bag of Words(DBOW)。DM 试图在给定上下文和段落向量的情况下预测单词的概率。在一个句子或者文档的训练过程中，段落 ID 保持不变，共享着同一个段落向量。DBOW 则在仅给定段落向量的情况下预测段落中一组随机单词的概率。[3]

在我们的实验中，使用的特征为 TF-IDF、Word Vector 及 Paragraph Vector。

二. 向量空间模型的特征选择

- 文档频率(Document Frequency, DF)

根据训练语料中的文档频率, 对所有特征进行排序。

- 互信息(Mutual Information, MI)

互信息是关于两个随机变量互相依赖程度的一种度量。计算文档中的词项 t 与文档类别 c 的互信息 MI, MI 度量的是词的存在与否给类别 c 带来的信息量。

- 信息增益(Information Gain, IG)

IG 衡量特征能够为分类系统带来多少信息。

- Chi-Square 统计(Chi-Square Statistics, CHI2)

卡方是基于显著统计性来选择特征的。

在我们的实验中，使用的特征选择方法为 Chi-Square 统计。

第四章 分类器概述

在文本分类中，假设我们有一个文档 $d \in X$ ， X 是文档向量空间， $C = \{c_1, c_2, \dots, c_j\}$ 类别集合，类别又称为标签。显然，文档向量空间是一个高维度空间。把带标签的文档集合 $\langle d, c \rangle$ 作为训练样本， $\langle d, c \rangle \in X \times C$ 。期望用某种训练算法，训练出一个函数 γ ，能够将文档映射到某一个类别： $\gamma: X \rightarrow C$ ，这种类型的学习方法叫做有监督学习。

一、传统的机器学习方法

1. MultinomialNB

朴素贝叶斯方法是基于贝叶斯理论的监督学习算法，该方法假设给定类变量 y 时，特征项之间是条件独立的。尽管该假设看上去非常简单，但是在一些应用领域，如文本分类、垃圾邮件过滤等，分类效果非常好。MultinomialNB 假设特征的先验概率为多项式分布，即如下式：

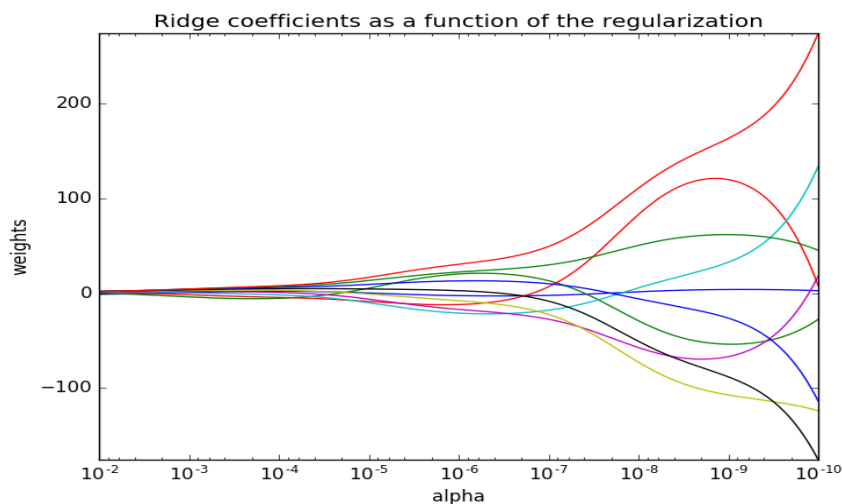
$$P(X_j = x_{jl} | Y = c_k) = \frac{x_{jl} + \lambda}{m_k + n\lambda}$$

其中， $P(X_j = x_{jl} | Y = c_k)$ 是第 k 个类别的第 j 维特征的第 l 个取值的条件概率， m_k 是训练集中输出为第 k 类的样本个数， λ 为一个大于 0 的常数，常常取为 1，即拉普拉斯平滑。也可以取其他值。

2. 岭回归 (Ridge Regression)

岭回归是在平方误差的基础上增加正则项，通过确定 α 的值可以使得方差和偏差达到平衡：随着 α 的增大，模型方差减小而偏差增大。

$$\min_w \|Xw - y\|_2^2 + \alpha \|w\|_2^2$$



3. 感知器 (Perception)

感知机学习旨在求出将训练数据集进行线性划分的分类超平面, 为此, 导入了基于误分类驱动的损失函数, 然后利用随机梯度下降法对损失函数进行极小化, 从而求出感知机模型。

4. 逻辑回归 (Logistic Regression)

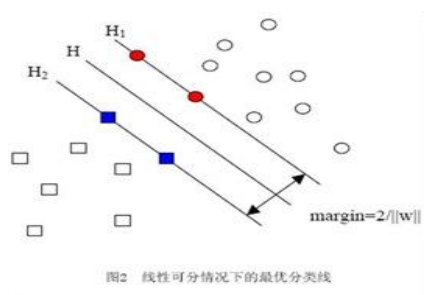
逻辑回归分类器本质上是线性分类器, 带 L2 正则项的最优化问题为:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^n \log(\exp(-y_i (X_i^T w + x)) + 1)$$

逻辑回归分类默认二分类, 在实验中选择 One-vs-Rest 的思想很简单。具体做法是, 对于第 K 类的分类决策, 把所有第 K 类的样本作为正例, 除了第 K 类样本以外的所有样本都作为负例, 然后在做二元逻辑回归, 得到第 K 类的分类模型。其他类的分类模型获得以此类推。

5. SVM

SVM 从线性可分情况下的最优分类面发展而来。最优分类面就是要求分类线不但能将两类正确分开 (训练错误率为 0), 且使分类间隔最大。SVM 考虑寻找一个满足分类要求的超平面, 并且使训练集中的点距离分类面尽可能的远, 也就是寻找一个分类面使它两侧的空白区域 (margin) 最大。过两类样本中离分类面最近的点且平行于最优分类面的超平面上 H_1, H_2 的训练样本就叫做支持向量。在实验中, 对于分类问题仍然使用 One-vs-Rest 的思想。



6. 随机森林

随机森林顾名思义, 是用随机的方式建立一个森林, 森林里面有很多的决策树组成, 随机森林的每一棵决策树之间是没有关联的。在得到森林之后, 当有一个新的输入样本进入的时候, 就让森林中的每一棵决策树分别进行判断, 看看这个样本应该属于哪一类。

在建立每一棵决策树的过程中, 有两点需要注意——采样与完全分裂。首先是两个随机采样的过程, random forest 对输入的数据要进行行、列的采样。对于行采样, 采用有放回的方式, 也就是在采样得到的样本集合中, 可能有重复的样本。假设输入样本为 N 个, 那么采样的样本也为 N 个。这样使得在训练的时候, 每一棵树的输入样本都不是全部的样本, 使得相对不容易出现 over-fitting。然后进行列采样, 从 M 个 feature 中, 选择 m 个 ($m \ll M$)。之后就是对采样之后的数据使用完全分裂的方式建立出决策树, 这样决策树的某一个叶子节点要是

无法继续分裂的，要么里面的所有样本的都是指向的同一个分类。

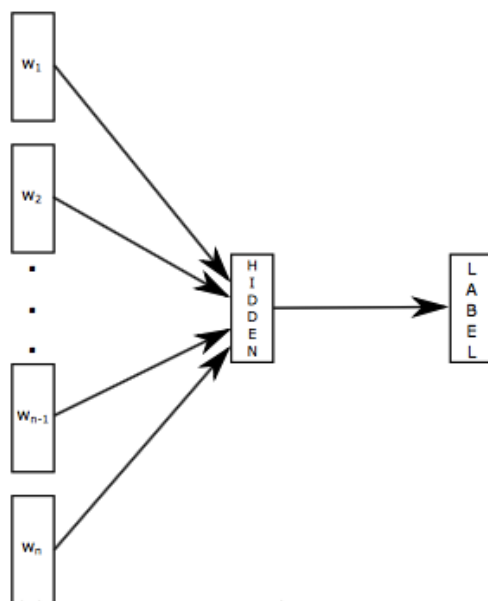
每一棵决策树就是一个精通于某一个窄领域的专家（从 M 个 feature 中选择 m 让每一棵决策树进行学习），这样在随机森林中就有了很多个精通不同领域的专家，对一个新的问题（新的输入数据），可以用不同的角度去看待它，最终由各个专家，投票得到结果。

二. 深度神经网络

1. fastText

fastText 是 facebook 开源的一个文本分类工具，它的特点是训练时间短，但能取得不错的分类效果。

fastText 的网络结构非常简单，如下图所示。模型输入一个词的序列（一段文本或者一句话），输出这个词序列属于不同类别的概率。序列中的词和词组组成特征向量，特征向量通过线性变换映射到中间层，中间层再映射到标签。fastText 在预测标签时使用了非线性激活函数，但在中间层不使用非线性激活函数。



fastText 利用层次 Softmax 来提高运行速度。层次 Softmax 技巧建立在哈弗曼编码的基础上，对标签进行编码，能够极大地缩小模型预测目标的数量。

此外，fastText 加入了 N-gram 特征，将词序的信息考虑在内，提高了分类的准确率。

2. CNN

卷积神经网络是一种前馈神经网络。它是受生物学上感受野的机制而提出的。一个神经元的感受野是指特定区域，只有这个区域内的刺激才能够激活该神经元。卷积神经网络的特点是局部链接，权值共享和下采样。

论文《Convolutional Neural Networks for Sentence Classification》介绍了 CNN 在

文本分类中结构和原理[6]。

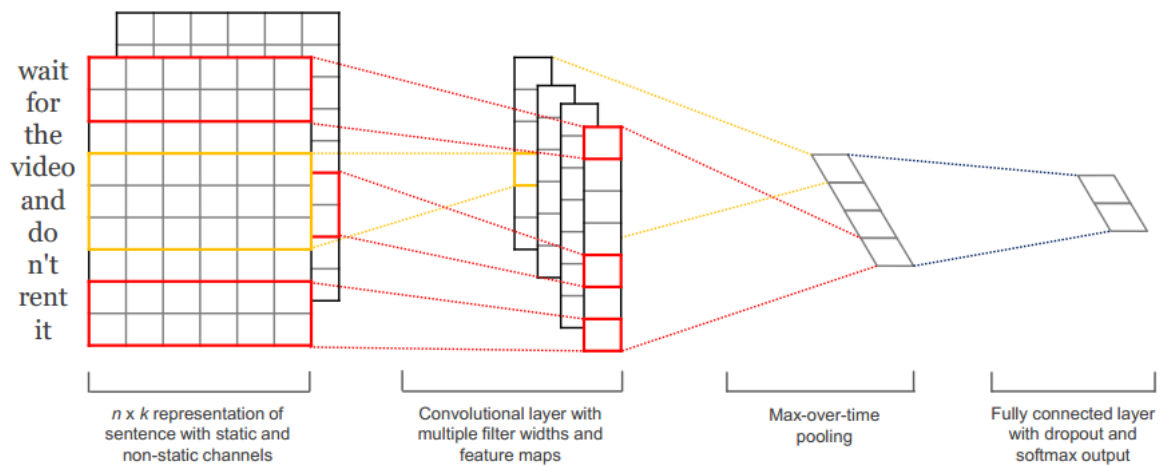
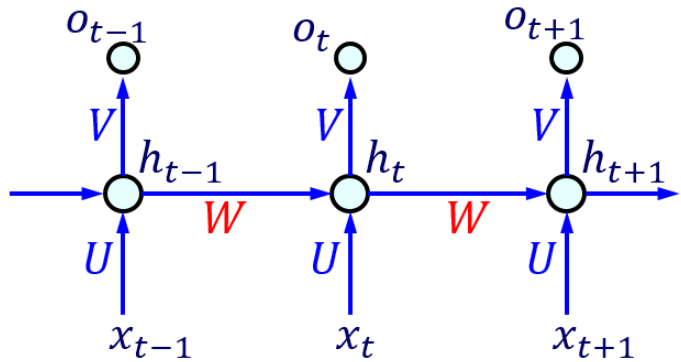


Figure 1: Model architecture with two channels for an example sentence.

输入层为一个词向量的矩阵，每一个词对应一个向量，词向量可以随机初始化也可以利用训练好的 word2vec。输入层通过卷积操作得到若干个 Feature Map。卷积窗口的大小为 $h \times k$ ，其中 h 表示纵向词语的个数，而 k 表示词向量的维数。接下来在池化层中用 MaxPooling 的方法简单地从之前一维的 Feature Map 中提出最大的值可以看出，这种 Pooling 方式可以解决可变长度的句子输入问题（因为不管 Feature Map 中有多少个值，只需要提取其中的最大值）。最终池化层的输出为各个 Feature Map 的最大值们，即一个一维的向量。池化层的一维向量的输出通过全连接的方式，连接一个 Softmax 层，最后输出各种类别的概率。

3. RNN

循环神经网络用来处理序列数据。在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，每层之间的节点是无连接的。RNN 一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。[7]



RNN 的结构如上图所示。其中

$$\begin{aligned}h_t &= f(Ux_t + Wh_{t-1} + b) \\o_t &= f(Vh_t + b')\end{aligned}$$

由于 RNN 能存储前面的信息，因此在文本分类和语音识别等领域都有较好的表现。[8]

4. LSTM

RNN 在训练的时候会存在梯度爆炸和梯度消失等问题。为了解决这些问题，LSTM (Long Short Term Memory) 采用一些控制门来减少梯度累积的长度。一旦信息得到利用，我们希望该结点能释放(遗忘)这种累积效应，从而使网络更具有灵活性和自主学习性。

LSTM 由简单的输入、输出、隐含层自循环增加了三个门单元：遗忘门：控制对细胞内部状态的遗忘程度，输入门：控制对细胞输入的接收程度，输出门：控制对细胞输出的认可程度。三个门均由当前输入信号和隐含层前一时刻的输出共同决定。[9]

第五章 实验分析

一. 实验环境

机器：联想 ThinkPad X1 Carbon 3rd 英特尔 Core i5-5200U @ 2.20GHz 双核

操作系统：Ubuntu 16.04 LTS

编程语言：Python

分词库：jieba

机器学习库：sklearn

深度学习库：keras

Word2Vec 与 Doc2Vec 模型：gensim

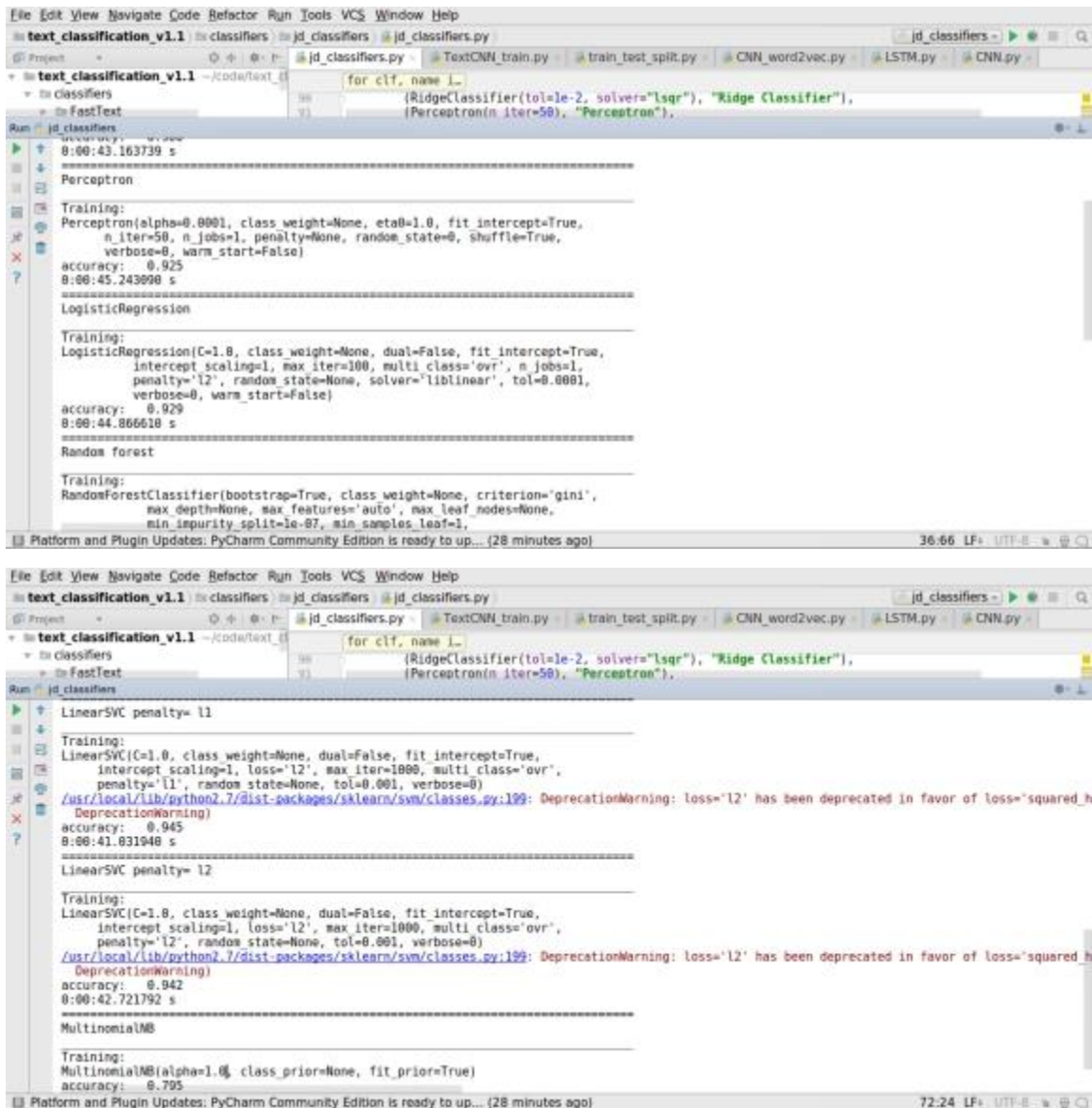
二. 实验结果

方法	准确度	训练时间
Ridge Regression +tfidf	0.900	43.163739 s
Perception +tfidf	0.925	46.822513 s
Logistic Regression +tfidf	0.929	44.866610 s
Random Forest +tfidf	0.949	58.229931 s
LinearSVC penalty= l1 +tfidf	0.945	41.031940 s
LinearSVC penalty= l2 +tdidf	0.942	42.721792 s
MultinomialNB +tfidf	0.795	40.484558 s
Ridge Regression +word2vec	0.952	3 min 3.175557 s
Perception +word2vec	0.964	3 min 2.185452 s
Logistic Regression +word2vec	0.971	3 min 29.068396 s
Random Forest +word2vec	0.950	4 min 14.250944 s
LinearSVC penalty=l1 +word2vec	0.971	6 min 31.680457
LinearSVC penalty=l2 +word2vec	0.971	3 min 3.691352
fastText	0.723	9.051379 s
CNN	0.685	30min
LSTM	0.9457	6320s/epoch*2epoch

传统分类器+tfidf 特征提取的结果截图：

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
text_classification_v1.1 classifiers jd_classifiers jd_classifiers.py
jd_classifiers.py TextCNN_train.py train_test_split.py CNN_word2vec.py LSTM.py CNN.py
text_classification_v1.1 -code/text
FastText
jd_classifiers
Run jd_classifiers
/usr/bin/python2.7 /home/liyt/code/text_classification_v1.1/classifiers/jd_classifiers/jd_classifiers.py
/usr/local/lib/python2.7/dist-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of
This module will be removed in 0.20.
Ridge Classifier
Training:
RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True,
max_iter=None, normalize=False, random_state=None, solver='lsqr',
tol=0.01)
/usr/local/lib/python2.7/dist-packages/sklearn/linear_model/ridge.py:311: UserWarning: In Ridge, only 'sag' solver can currently fit the interce
warnings.warn('In Ridge, only \'sag\' solver can currently fit the '
accuracy: 0.900
0:00:43.163739 s
Perceptron
Training:
Perceptron(alpha=0.0001, class_weight=None, eta0=1.0, fit_intercept=True,
n_iter=50, n_jobs=1, penalty=None, random_state=0, shuffle=True,
verbose=0, warm_start=False)
accuracy: 0.925
0:00:45.243090 s
LogisticRegression
Training:
/usr/local/lib/python2.7/dist-packages/sklearn/linear_model/logistic.py:1218: UserWarning: Default solver has changed from lbfgs to sag.
accuracy: 0.925
0:00:45.243090 s
Platform and Plugin Updates: PyCharm Community Edition is ready to up... (26 minutes ago) 1:1 LF+ UTF-8
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
text_classification_v1.1 classifiers jd_classifiers jd_classifiers.py
jd_classifiers.py TextCNN_train.py train_test_split.py CNN_word2vec.py LSTM.py CNN.py
text_classification_v1.1 -code/text
FastText
jd_classifiers
Run jd_classifiers
Intercept scaling=1, loss='l2', max_iter=1000, multi_class='ovr',
penalty='l1', random_state=None, tol=0.001, verbose=0)
/usr/local/lib/python2.7/dist-packages/sklearn/svm/classes.py:199: DeprecationWarning: loss='l2' has been deprecated in favor of loss='squared_h
DeprecationWarning)
accuracy: 0.945
0:00:41.031940 s
LinearSVC penalty='l2'
Training:
LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, loss='l2', max_iter=1000, multi_class='ovr',
penalty='l2', random_state=None, tol=0.001, verbose=0)
/usr/local/lib/python2.7/dist-packages/sklearn/svm/classes.py:199: DeprecationWarning: loss='l2' has been deprecated in favor of loss='squared_h
DeprecationWarning)
accuracy: 0.942
0:00:42.721792 s
MultinomialNB
Training:
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
accuracy: 0.795
0:00:40.484558 s
Process finished with exit code 0
Platform and Plugin Updates: PyCharm Community Edition is ready to up... (30 minutes ago) 70:52 LF+ UTF-8
```



传统分类器+Word2Vec 截图：


```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
text_classification_v1.1 classifiers jd_classifiers jd_classifiers_word2vec.py jd_classifiers_word2vec
jd_classifiers.py jd_classifiers_word2vec.py train_word2vec.py jd_classifiers_doc2vec.py
jd_classifier(train())
jd_classifier_word2vec
/usr/bin/python2.7 /home/liyt/code/text_classification_v1.1/classifiers/jd_classifiers/jd_classifiers_word2vec.py
Using TensorFlow backend.
/usr/local/lib/python2.7/dist-packages/sklearn/cross_validation.py:44: DeprecationWarning: This module was deprecated in version 0.18 in favor of
    "This module will be removed in 0.20.", DeprecationWarning)
Ridge Classifier
Training:
RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True,
    max_iter=None, normalize=False, random_state=None, solver='lsqr',
    tol=0.01)
/usr/local/lib/python2.7/dist-packages/sklearn/linear_model/ridge.py:311: UserWarning: In Ridge, only 'sag' solver can currently fit the interce
    warnings.warn("In Ridge, only 'sag' solver can currently fit the
accuracy: 0.952
0:03:03.175557 s
Perceptron
Training:
Perceptron(alpha=0.0001, class_weight=None, eta0=1.0, fit_intercept=True,
    n_iter=50, n_jobs=1, penalty=None, random_state=0, shuffle=True,
    verbose=0, warm_start=False)
accuracy: 0.964
0:03:02.185452 s
LogisticRegression
Platform and Plugin Updates: PyCharm Community Edition is ready to up... (17 minutes ago) 25:25 LF+ UTF-8
```

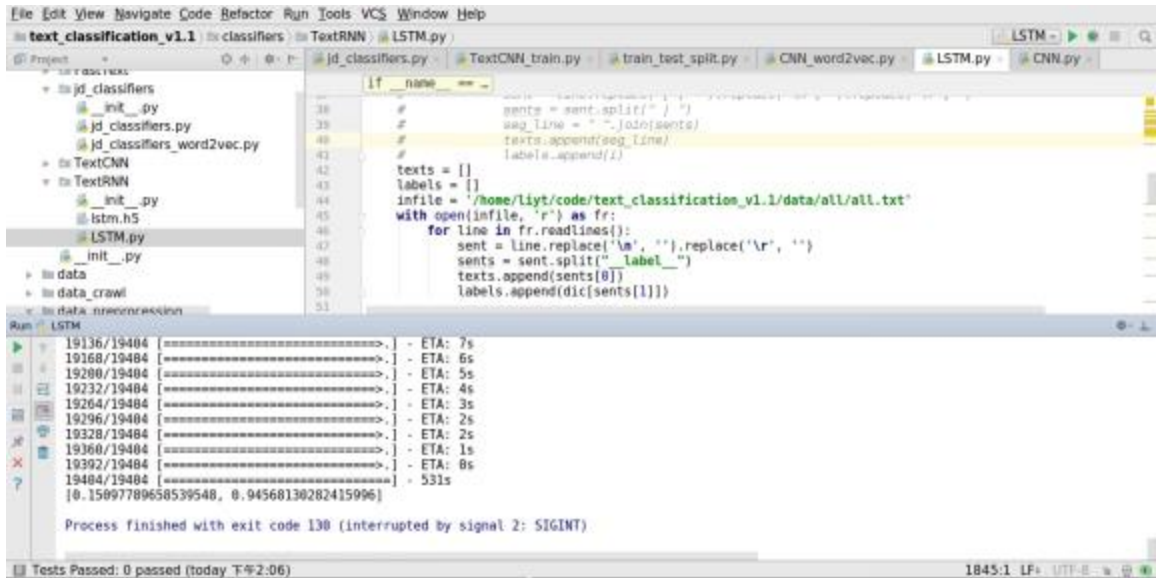
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
text_classification_v1.1 classifiers jd_classifiers jd_classifiers_word2vec.py jd_classifiers_word2vec
jd_classifiers.py jd_classifiers_word2vec.py train_word2vec.py jd_classifiers_doc2vec.py
jd_classifier(train())
jd_classifier_word2vec
for clf, name in
jd_classifier_word2vec
verbose=0, warm_start=False)
accuracy: 0.964
0:03:02.185452 s
LogisticRegression
Training:
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
    intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
    penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
    verbose=0, warm_start=False)
accuracy: 0.971
0:03:29.068396 s
Random forest
Training:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
    max_depth=None, max_features='auto', max_leaf_nodes=None,
    min_impurity_split=1e-07, min_samples_leaf=1,
    min_samples_split=2, min_weight_fraction_leaf=0.0,
    n_estimators=30, n_jobs=1, oob_score=False, random_state=None,
    verbose=0, warm_start=False)
accuracy: 0.950
0:04:14.250944 s
LinearSVC penalty= l1
Platform and Plugin Updates: PyCharm Community Edition is ready to up... (24 minutes ago) 49:1 LF+ UTF-8
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
text_classification_v1.1 classifiers jd_classifiers jd_classifiers_word2vec.py jd_classifiers_word2vec.py train_word2vec.py jd_classifiers_doc2vec.py
jd_classifiers_word2vec.py
Run jd_classifiers_word2vec.py
verbose=0, warm_start=False)
accuracy: 0.950
0:04:14.250944 s
=====
LinearSVC penalty= l1
Training:
LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, loss='l2', max_iter=1000, multi_class='ovr',
penalty='l1', random_state=None, tol=0.001, verbose=0)
/usr/local/lib/python2.7/dist-packages/sklearn/svm/classes.py:199: DeprecationWarning: loss='l2' has been deprecated in favor of loss='squared_hinge'
DeprecationWarning)
accuracy: 0.971
0:06:31.688457 s
=====
LinearSVC penalty= l2
Training:
LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, loss='l2', max_iter=1000, multi_class='ovr',
penalty='l2', random_state=None, tol=0.001, verbose=0)
/usr/local/lib/python2.7/dist-packages/sklearn/svm/classes.py:199: DeprecationWarning: loss='l2' has been deprecated in favor of loss='squared_hinge'
DeprecationWarning)
accuracy: 0.971
0:03:03.691352 s
Process finished with exit code 0
Platform and Plugin Updates: PyCharm Community Edition is ready to up... (33 minutes ago) 68:17 LF+ UTF-8
```

LSTM 结果截图：

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
text_classification_v1.1 classifiers TextRNN LSTM.py LSTM.py CNN.py
jd_classifiers.py TextCNN_train.py train_test_split.py CNN_word2vec.py LSTM.py CNN.py
jd_classifiers.py
jd_classifiers_word2vec.py
TextCNN
LSTM.py
LSTM.py
data
data_crawl
data_preprocessing
Run LSTM
76672/77616 [=====] - ETA: 76s - loss: 0.1141 - acc: 0.9704
76800/77616 [=====] - ETA: 66s - loss: 0.1139 - acc: 0.9704
76928/77616 [=====] - ETA: 55s - loss: 0.1138 - acc: 0.9705
77056/77616 [=====] - ETA: 45s - loss: 0.1137 - acc: 0.9705
77184/77616 [=====] - ETA: 34s - loss: 0.1137 - acc: 0.9705
77312/77616 [=====] - ETA: 24s - loss: 0.1137 - acc: 0.9705
77440/77616 [=====] - ETA: 14s - loss: 0.1137 - acc: 0.9705
77568/77616 [=====] - ETA: 3s - loss: 0.1136 - acc: 0.9705
77616/77616 [=====] - ETA: 6320s - loss: 0.1137 - acc: 0.9705
32/19404 [=====] - ETA: 9979s
64/19404 [=====] - ETA: 5199s
96/19404 [=====] - ETA: 3609s
128/19404 [=====] - ETA: 2811s
160/19404 [=====] - ETA: 2338s
192/19404 [=====] - ETA: 2000s
Tests Passed: 0 passed (today T*2:06) 1227:1 LF+ UTF-8
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
text_classification_v1.1 classifiers TextRNN LSTM.py
Project jd_classifiers.py TextCNN_train.py train_test_split.py CNN_word2vec.py LSTM.py CNN.py
jd_classifiers
  __init__.py
  jd_classifiers.py
  jd_classifiers_word2vec.py
TextCNN
  __init__.py
  lstm_h5
  LSTM.py
data
  data_crawl
  data_preprocessing
Run LSTM
19136/19484 [=====] - ETA: 7s
19168/19484 [=====] - ETA: 6s
19200/19484 [=====] - ETA: 5s
19232/19484 [=====] - ETA: 4s
19264/19484 [=====] - ETA: 3s
19296/19484 [=====] - ETA: 2s
19328/19484 [=====] - ETA: 2s
19360/19484 [=====] - ETA: 1s
19392/19484 [=====] - ETA: 0s
19484/19484 [=====] - 531s
[0.15097789658539548, 0.94568130282415996]
Process finished with exit code 130 (interrupted by signal 2: SIGINT)
Tests Passed: 0 passed (today 下午2:06) 1845:1 LF: UTF-8
```

三、实验分析

1. 在传统机器学习方法中，特征值为 doc2vec 的准确度比 word2vec 高。

分析：在实验中，通过 word2vec 方法获得每个词的词向量后，整篇文档的向量表示为文档中所有词的词向量平均值。这种做法，忽略了上下文和单词顺序的信息，即上述的 word2vec 只是基于词的维度进行“语义分析”的，并不具有上下文的“语义分析”能力。

2. RNN 与 CNN 相比，RNN 的准确度高，而训练时间长。

分析：尽管 CNN 也有不错的表现，但 CNN 的最大问题是固定 filter_size 的视野，一方面无法建模更长的序列信息，另一方面 filter_size 的超参调节也很繁琐。CNN 本质是做文本的特征表达工作，而自然语言处理中更常用的是递归神经网络（RNN），能够更好的表达上下文信息。因此，RNN 的准确度高。由于 CNN 使用了卷积层和 pooling 层，使得 CNN 能够减少训练参数个数，极大地节省了时间。

3. 在传统分类器中，Word2Vec 的特征提取要比 tfidf 特征提取的准确率稍高

分析：Word2Vec 相对于统计的 tfidf 考虑了语义的信息。如“枯燥”和“无聊”，对于 tfidf 来说这是完全不相关的词语。但对于 Word2Vec 来说，两个词却在距离上是相似的。此外，利用 Word2Vec 引入外部训练数据，防止了过拟合。

其他分析：fastText 训练速度远比其他方法快，但在实验结果上准确率并不高，CNN 的准确率也没有超过传统分类器的准确率。

第六章 总结

本实验主要用了传统的机器学习方法和深度学习方法。在传统的机器学习方法中，主要用了 tfidf+CHI2 进行特征提取和特征选择，以及 word2Vec 作为特征提取两种方式。传统机器学习的分类器包括 Ridge Regression Perception、Logistic Regression 、Random Forest、LinearSVC penalty= l1 LinearSVC penalty= l2 、MultinomialNB 等。在深度学习方法中主要用到了 CNN，fastText 和 LSTM。

在语料方面，我们通过网络爬虫爬取 9.8 万篇的新闻，近 400MB 的数据。

在实验结果方面，经过我们的调参从一开始 0.7~0.8 的准确率，最后很多分类器到达了 0.9 以上的准确率。我们可以看出 tfidf 的传统机器学习方法要略低于 word2vec 的，fastText 训练速度远比其他方法快，但在实验结果上准确率并不高，CNN 的准确率也没有超过传统分类器的准确率。因此我们不能盲目的迷恋深度学习。

总的来说通过此次实验加深了我们的动手实践能力，也加深了对课堂知识的理解，收获巨大。

参考文献

- [1] 张国梁, 肖超锋. 基于 SVM 新闻文本分类的研究[J]. 电子技术, 2011, 38(8):16-17.
- [2] <http://www.jianshu.com/p/ab697790090f>
- [3] Le Q V, Mikolov T. Distributed Representations of Sentences and Documents[J]. 2014, 4:II-1188.
- [4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.
- [5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.
- [6] Kim Y. Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.
- [7] Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model[C]// INTERSPEECH 2010, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September. DBLP, 2010:1045-1048.
- [8] Sutskever I, Martens J, Hinton G E. Generating Text with Recurrent Neural Networks[C]// International Conference on Machine Learning, ICML 2011, Bellevue, Washington, Usa, June 28 - July. DBLP, 2011:1017-1024.
- [9] Sutskever I, Vinyals O, Le Q V. Sequence to Sequence Learning with Neural Networks[J]. Advances in neural information processing systems, 2014, 4:3104-3112.

附录 成员分工

成员	工作内容
金融通	与李页霆合作 word2vec、CNN、fastText； 撰写获取语料、深度神经网络、实验结果等文档内容
李页霆	写爬虫获取语料； 完成 word2vec、CNN、RNN 代码； 将代码整合，跑实验结果。
王文惠	完成 doc2vec、传统机器学习方法代码； 撰写绪论、特征工程及传统机器学习方法概述等文档内容