山 东 科 技 大 学
本科毕业设计（论文）

题 目 基于深度学习与词嵌入的情感分析系统研究与实现

学 院 名 称 计算机科学与工程

专 业 班 级 计算机科学与技术 2015-1

学 生 姓 名 聂国庆

学 号 201501060119

指 导 教 师 赵中英

摘要

近年来，随着网络平台数量以及用户数量的大规模增长，大量很有价值的用户主观性文本需要得到有效的分析和利用。对这些信息的有效分析，能够帮助我们进行更准确的、更及时的响应。本文研究的主要方向是自然语言处理领域的文本情感分析，重点研究句子和段落级的文本，这些文本包括影评、推文等。

本文主要使用基于深度学习的方法，数据集采用论文常用的 IMDB 数据集，研究目标旨在提高最终设计模型的准确性。本文尝试吸收其他深度学习模型优点，自己设计了 7 个深度学习模型。本文主要创新点在于，利用模型集成融合里的堆叠法的思想，实现了 3 个树形的传统机器学习算法与 7 个深度学习模型的集成。本文在 Kaggle 竞赛提供的数据集上，最终给出的 AUC 评分达到前 15%，高于已有的评分。

高效的情感分析方法在实际应用中是很有价值的，随着很多中文词向量以及算法的开源，今后无论中文或者其他语言的情感分析技术必将不断成熟。

关键字： 情感分析 深度学习 机器学习 注意力机制 模型堆叠

Abstract

In recent years, with the large number of network platforms and the large number of users, a large number of valuable user subjective texts need to be effectively analyzed and utilized. An effective analysis of this information can help us to respond more accurately and in a timely manner. The main direction of this paper is text sentiment analysis in the field of natural language processing, focusing on sentence and paragraph-level texts, including film reviews, tweets, etc.

This paper mainly uses a deep learning-based approach. The dataset uses the IMDB dataset commonly used in the paper. The research goal is to improve the accuracy of the final design model. This paper attempts to absorb the advantages of other deep learning models and designs seven deep learning models. The main innovation of this paper is to realize the integration of three tree-shaped traditional machine learning algorithms and seven deep learning models by using the idea of stacking in model integration. In the data set provided by the Kaggle competition, the final AUC score is 15% higher than the existing score. Effective sentiment analysis is very valuable in practical applications. With the open source of many Chinese word vectors and algorithms, sentiment analysis of all languages will mature in the future.

Keywords: sentiment analysis, deep learning, machine learning, attention mechanism, stacking

目录

1 绪论

概述：

本章为绪论部分，介绍情感分析项目的研究背景和研究内容，介绍学习考察到的国内外的相关的研究进展，最后介绍论文的章节安排结构。

1.1 研究背景

科学技术的不断进步，现如今，在“大数据”（Big Data）时代的背景之下，数据呈现爆炸性增长的态势，不管是电子商务平台的海量数据、还是社交媒体的海量数据，都呈现出越来越庞大、越来越复杂的趋势。对这些海量数据的收集、清洗以及之后的挖掘、分析，成为现如今大量公司和学者们共同面对和共同解决的问题。

无论是社交媒体还是电子商务亦或是其他软件社区等等，都是由众多的用户构成，用户的对商品或事件评价、讨论、互动等发布的各种信息在不同时间、不同纬度共同汇聚，来自全世界的数据流量汇成茫茫数据大海。在越来越庞大的数据规模以及复杂度面前，数据的处理难度越来越大，但随着数据的足够庞大，对有效信息的获取变得更加准确。通过庞大规模的数据，我们能够挖掘出以前不能的到信息。比如某知名研究通过超市数据挖掘分析可以得到“男人在给孩子买尿布时，通常也会给自己带一瓶啤酒”[4]的结论，在这个结论的帮助下，可以在超市里把尿布和啤酒摆在一起，达到促进消费的作用。

同样，在众多线上社交媒体社区、电子商务平台里，在众多的用户提交的评价、讨论或媒体新闻等的文本信息里，蕴含着很多丰富且有用的信息，可以从大量的文本信息中提取出准确有效的信息，帮助我们作出相应的反映与改进。

情感分析就是针对大量文本的一种有效的分析维度，通过对文本的处理、挖掘、分析，可以得到用户对某些商品、商家的喜好倾向，可以得到众多媒体对某一件事的报道倾向，可以得到网民对某些热点时间的舆论导向，可以个性化的推荐，可以积极地引导。通过庞大的数据分析，获取我们想得到的情感倾向信息，帮助我们做出正确且及时的响应。通过用户的评价，改进产品；通过对某影院的评价，实现影院推荐；通过对某些事件的讨论的评价情感倾向，把控舆论风向，积极引导舆论导向。

1.2 研究内容

为了能够准确的获取我们想要获取的信息，我们要提高分析的准确性，针对不同的领域、不同问题进行分析处理建模。如果研究方法的准确度无法提高，将会在实际应用中浪费资源且得不到想要分析结果，这样是得不偿失的。分析的准确度能够给我们带来足够准确的信息，我们会做出足够准确的反应，既节省了人力物力资源，又通过对事件的个性化的高效的处理，得到了以前无法得到的响应效果。

同时，在兼顾准确性的同时应该注意方法的泛化能力，提高方法的可移植性，在面对不同领域的问题时也能微调并进行应对。泛化性强的方法，能够更好的应用进而能不断的演进与充分的发挥。

本文重点研究用户评论的情感分析，比如影评、twitter、微博、电商平台商品评价等等评论文本，这些评论不会像文章一样很长，大多数以短文本为主，在几百字的文本里体现用户的情感，体现用户对目标的态度。通过对用户情感的分析，尤其是大数据大规模的用户情感的分析，我们可以更加准确的得到用户的态度，帮助我们更加准确的获取相关信息，做出更加正确的反应。

不管是对商品的评价、对舆论事件的评论、亦或是其他文本，人工的情感分析判断在大数据面前就好像是螳臂当车了，传统的简单的情感分析方法又太过片面。可能在极其简单的情况下简单的方法也可以发挥不错的效果，但是面对越来越庞大且复杂的数据，简单的方法效率不高且准确度无法保证，如果不考虑语义而只是从统计学等方法对词语进行处理，很多语句无法理解，无法处理歧义，准确性不好，从而致使最终结果说服力不够而且有效性不高。

机器学习（Machine Learning）尤其是深度学习（Deep Learning）的快速发展给我们提高情感分析正确性提供了可能，人工智能 AI （Artificial Intelligence）想方设法让机器能够像人一样思考，进而有可能实现让机器理解人类语言，理解一些语义信息，从而更能像人一样去判断语句的情感倾向等等信息。但，传统的机器学习方法并不能很好的理解语义信息，因为他们只能单个词单个词的去处理文本信息，所以很多时候需要深度学习去存储记忆状态（memory），更好的理解文本语义信息，更好的进行情感分析。

对自然语言的理解、分析、分类、预测等等，在 AI 领域中拥有专门的子领域，也就是常说的 NLP（自然语言处理，Neural Language Processing），NLP 重点利用人工智能的方法增强理解语言的能力，以便提高具体任务的效果。

在数据爆炸的今天，每分钟都会产生过去很多很多年才会产生的数据，这也给我们数据分析准确性提供了可能，这在过去也是不可想象的。同时在计算机硬件所遵循的摩尔定律（Moore）的支持下，对海量数据的运算提供了硬件可能。

硬件的支持、数据量的扩张、算法的发展优化，也正是近些年机器学习尤其是深度学习迅速发展的最重要的推动力。通过机器学习方法，尤其是

深度学习方法，在大数据的支持下，获得了很多意想不到的成果。机器学习的强大不仅仅在于自身的完善与效率，同时更在于机器学习对各行各业的深刻影响。硬件的计算能力虽然远超人脑，但是过于死板的特点使其无法超越人脑。不过机器学习在某种程度上给了机械的硬件一些灵活的处理反应，就好像是线性到了非线性一样，实现了必要的突破与进步。近些年机器学习的迅速发展，更是使机械的硬件在机器学习算法的驱动下，在某些特定的领域超越人脑。

所以基于机器学习特别是深度学习的情感分析研究方法，可以给我们带来越来越好的研究与应用前景，值得我们去投入学习与研究。

1.3 章节结构

第一章作为整篇文章的绪论部分介绍了本可以的研究背景和研究内容，接下来第二章将会分章进行系统设计的介绍，即系统整体架构的详细设计和运行部署的流程。第三章将会介绍系统算法部分的原理，也是本文最核心的一部分，详细介绍算法的设计与原理，以及设计算法的原因。第四章将会介绍系统的测试调优过程，最后第五章是本文的总结与展望部分，总结本文并介绍本文做得不够的地方以及接下来工作学习的展望。最终是参考文献和对老师同学的致谢部分。

2 系统设计

概述:

本章介绍毕业设计整体的系统设计及其每一模块的架构设计。重点突出机器学习模块的设计与提升，在此基础上加以封装，实现简单例子（demo）的交互，体现整体系统的感觉，尽可能体现解决领域内问题的学习与努力。本章将会介绍系统功能的整体设计架构，分别介绍机器学习模块的构建与改进、机器学习模型的部署模块。

2.1 系统功能设计

具体的整体的系统功能设计架构如下图所示：

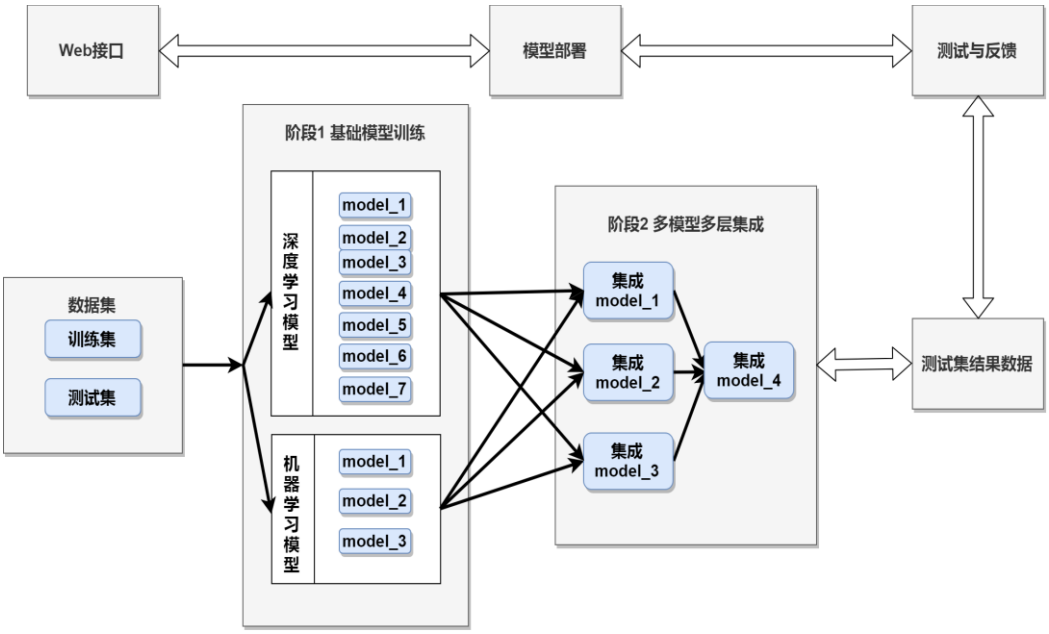

图 2-1 系统总体设计流程图

Figure 2-1 General structure flow chart

本文研究文本的情感分析，每条文本的文本平均长度在 100~200 词左右，最长不超过 500 词，倾向于短文本的情感分析。本文重点在于研究机器学习算法的改进，以国外相关论文为学习和构建模型的验证挑战基础，利用国外论文常用的数据集对自己的模型进行训练与评估，不断改进模型，尝试不同方法，不断学习探索，不断提升自己。通过具体提供上述数据集的比赛进行结果排名与打分，评分方法和结果较为权威。为了更加能够使项目体现系统的概念，利用 Flask 框架对模型进行了简单封装，增加了一个前端 web 页面，可以在线实现对输入句子的情感分析。

2.2 系统各模块详细功能

2.2.1 主模块（机器学习模块）

机器学习模块的训练，使用可以在竞赛网站获得的并有相关论文使用的数据集，使用的两个数据集来自两个竞赛，第一个是来自 Kaggle 竞赛网站，“Bag of Words Meets Bags of Popcorn”，是一个 IMDB 影评的数据集比赛，影评情感分为记记和消极；第二个是 Analytics Vidhya 竞赛网站的“Twitter Sentiment Analysis”比赛。

机器学习模块采用“宽且深（wide and deep）”的模型类型，结合深度学习与传统的浅层学习，联合训练，得到集成的效果比较好的模型。具体的机器学习架构将会在第三章系统实现部分详细介绍。下图是机器学习模型的架构。

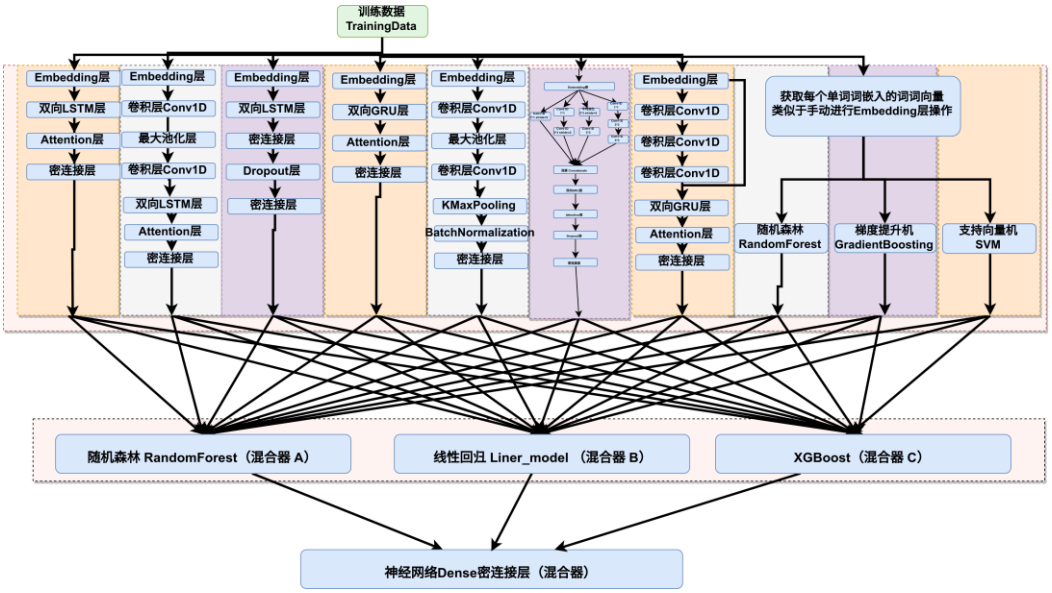

图 2-2 机器学习模块设计结构

Figure 2-2 The structure of machine learning models

2.2.2 模型部署模块

本文重点研究机器学习算法，所以模块部署部分相对较为简单，利用 Python 支持的 Flask 框架实现。首先加载已经训练好并且进行保存的模型，需要分析的语句通过前端 From 进行提交，获取到文本以后进行与训练模型时相同的预处理，接下来调用 model.predict（）对文本情感进行预测，预测值返回 web 网页进行展示。

2.3 主界面设计

2.3.1 提交文本

在文本框里输入想要测试的文本，长度最好不要超过 500 词，超过部分将不会进入模型进行预测。写完之后，点击“Enter”键进行提交。

图 2-3 主界面关键细节展示

Figure 2-3 Web submit interface

2.3.2 结果返回

点击“Enter”键之后，待系统尽情测评，很快将会返回想要的提交文本的情感分析结果，显示情感极性“positive”或“negative”，并显示提交的分析的文本。

图 2-4 情感分析结果

Figure 2-4 Result of sentiment analysis

2.4 本章小结

本章介绍了系统的整体功能架构，展示了系统的整体架构，分别展示了机器学习模块的架构和模型部署部分的架构，对 web 前端页面也进行的展示。本文重点研究机器学习方法的提升与改进，因此系统的重点在于机器学习算法设计实现部分，相关的实现是下一章将要介绍的重点。

3 系统实现

本章将介绍毕业设计的具体实现与改进，第一节是对系统实现的部分更加详细的概述部分，之后介绍数据的处理以及特征提取的选择，接下来分别从深度学习、机器学习两个种类方面来介绍项目使用的每个模型，继续介绍模型的融合方式以及训练。最后介绍部署模型部分的详细实现。

3.1 概述

面对要分析的数据，主要包含训练集和测试集，首先需要进行数据的预处理，选择提取数据特征的方法之后进一步处理数据，接下来就会把提取的特征输入到机器学习模型当中进行训练。

本文的机器学习模型由两个大类的模型组成，包含 10 个具体的模型。两个大类即深度学习模型和传统机器学习模型，两个大类中的模型包含添加 attention 机制的 LSTM/GRU 模型、包含类似于残差网络（residual connection）的一维卷积神经网络模型、包含随机森林和梯度提升机以及 SVM 等传统机器学习模型。

模型广义上采用三层结构，第一层为若干个原始模型，第二层采用多种机器学习模型进行第一次模型融合，第三层对第二层的多种机器学习模型进行再次的堆叠（stacking）。

模型的输入也有两种，一种是利用自己训练的 Word2Vector 模型得到词嵌入空间，另一种是使用预训练的词向量。

模型架构完成之后开始训练、超参调优，保存模型。

利用 Flask 框架对模型进行部署，利用 web 页面实现简单交互。

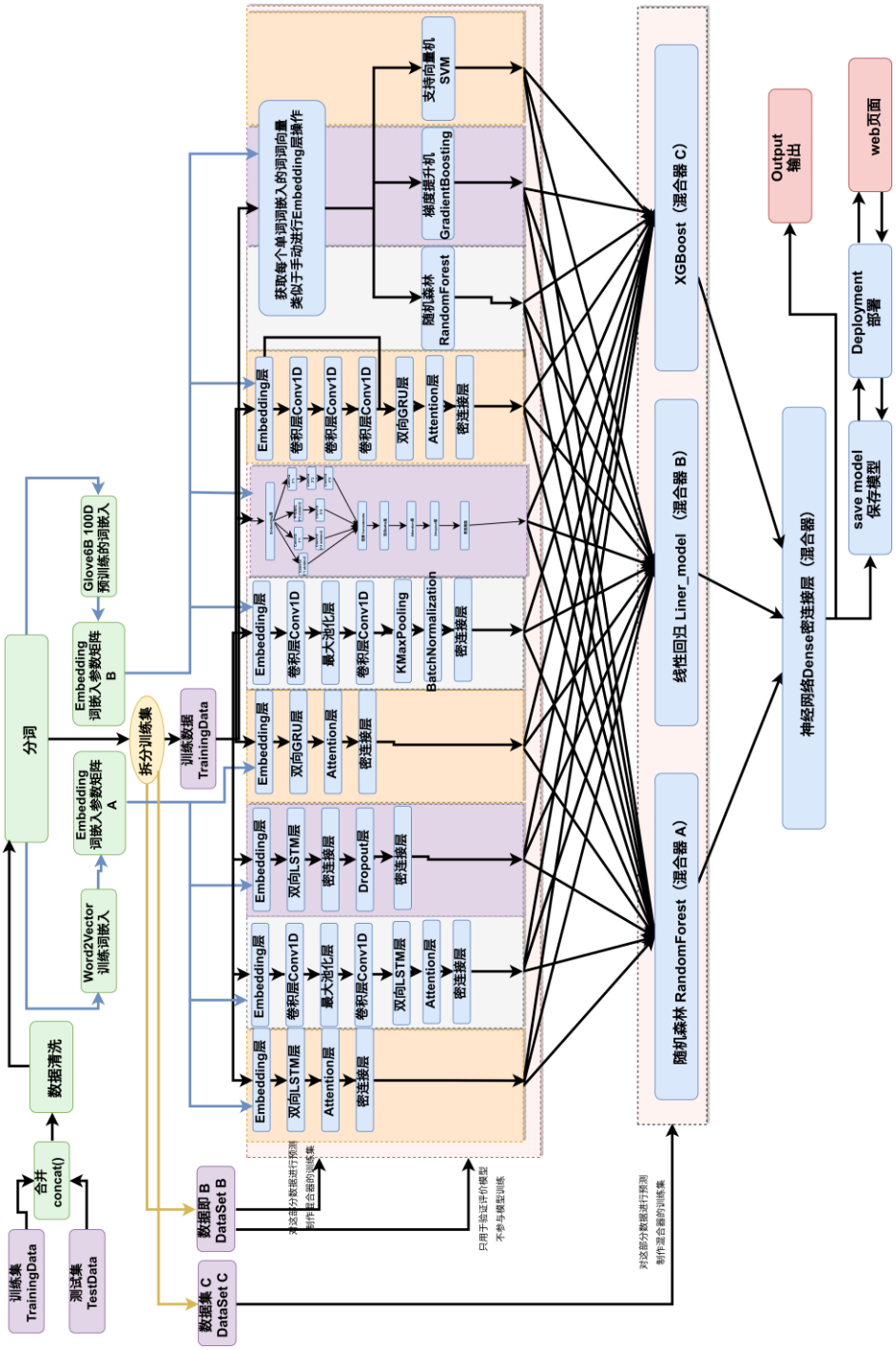

图 3-1 系统实现完整流程图

Figure 3-1 Complete flowchart of the system implementation

3.2 数据处理与特征提取选择

3.2.1 数据预处理

文本数据如果需要被数学化处理的话，其本身肯定也要处理成数学形式才能够被分析和学习。对于数据分析项目来说首先要进行的就是对数据进行数据清洗，清洗掉不必要即没有任何分析价值的信息，清洗掉容易混淆理解分析结果的信息，尽可能的降低噪声，这样才能更好的提高分析效率和准确性。

数据预处理部分我首先使用了正则表达式进行初步的清洗，之后清理了常见的、基本对语言分析没有帮助的停用词（stop words）。接下来利用 NLTK 的语言处理工具进行了“词性恢复”和“同义替换”，为接下来分词训练词向量等做更好的准备。

其中“词性恢复”可以单词恢复到指定的、统一的词性，“同义替换”可以更加同一单词的说法和用法。噪声更小，语言词法句法更简单的语句对接下来的进一步处理起到比较好的作用。

3.2.2 词向量

面对清洗后的条目很多的一行一行的字符串文本数据，首先要做的就是进行分词（tokenization），进行分词后产生的小的结构通常称为标记（token）。文本分词方法有很多，比较常用的分词级别包括单词级、字符级、n-gram，可以利用词袋（bags of words）等方法进行相应分词级别的分词，接下来可以利用 One-Hot 编码、TD-IDF 等等方法进行特征提取，提取词向量。

但是，词袋会忽略顺序，舍弃了句子的整体结构，配合 TD-IDF 方法适合关键词、关键信息的提取，很适合浅层的语言处理模型，但是并不适合深度学习模型，对于文本理解（比如情感分析、机器翻译、智能客服等）来说是不合适的。

同样，One-Hot 编码太过稀疏，对于稀疏矩阵的运算是耗费高、效率低的，而且 One-Hot 编码的词向量之间没有关联关系，彼此之间是离散没有任何语义相关性的。使用 One-Hot 编码很容易造成矩阵维度过高、计算代价太高的问题，效率不高、效果不好。

深度学习相对于传统机器学习模型的一个很大的优势点在于深度学习不需要很复杂的特征工程，不需要死板而且不稳定的方法，因为深度学习可以自己分层的进行学习。所以词向量的处理方法就很简单，可以直接使用分词器（Tokenizer）进行分词，之后直接利用训练好的分词器的 text_to_sequence（）函数生成词向量，同样也可以利用分词器导出的词典（word_index）进行手动生成词向量。

3.2.3 Embedding 词嵌入

对于自然语言处理的问题，Embedding 词嵌入是特别合适的方法，Embedding 会把词向量嵌入到一个相对于其他方法来说要低很多的而且能够自定义的维度当中。

Embedding 词嵌入会把词向量嵌入到一个词嵌入空间当中，在这个空间的当中，词向量之间是有相互关系的，比如:

“king（国王）” - “male（男性）” + “female（女性）” = “queen（女王）”

但是词嵌入并不是如此容易的利用人类语言进行解释，有时候的结果是没法很好的进行解释的。

关于词嵌入，简单的图示实例如下图所示：

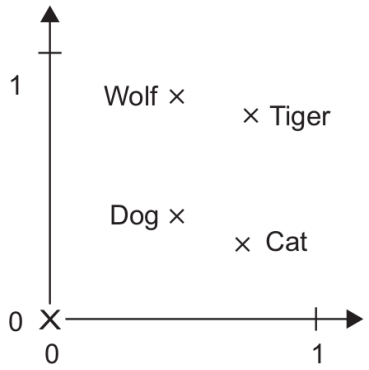

图 3-2 词嵌入空间实例

Figure 3-2 simple example of a word embedding space

猫到老虎和狗到狼的距离可以被解释为“从宠物到野生动物”向量，这两段的向量是一样长度的；同样，猫到狗和老虎到狼的距离可以被解释为“从猫科到犬科”向量，这当然也是相等的[1]。

类似的例子有很多，这些就是词嵌入的有趣的地方和独特的魅力。与此同时，词嵌入的目的得到了实现，词向量之间增加了关联，词向量被嵌入到一个相对很低的维度，这样既方便了计算又体现了词向量之间的关联关系。

本文中采用两种方法，第一种方法是，把分词后的文本使用 Word2Vector 模型训练自定义维度的词向量的词嵌入。第二种方法是使用预训练的词向量，选择适合的嵌入维度。把词嵌入放入在 Embedding 层加载，并且本身 Embedding 词嵌入层在模型训练过程中是同样会进行训练和提升的。

3.2.4 Word to Vector

如果拥有的训练数据量大、或者想自己独立训练词嵌入空间的话就可以尝试使用一个叫做 Word2Vector 的神经网络，它只有两层。我们可以输入一

个自己组合的文本语料库到 Word2vec 当中，它会输出一组向量，即该语料库中单词的特征向量。虽然 Word2vec 的深度不深，但得到的效果使我们所希望的，因为它能够将我们所输入的文本信息转换为深度神经网络需要的、可以理解的数学形式。

Word2Vector 的输入要求很简单，不管是分好词之后存在变量中的文本信息、亦或者是空格分割的本地文本等等，都可以直接输入到 Word2Vecotr 模型中作为 Word2Vecotr 的输入进行训练。Word2Vecotr 有 CBOW 和 Skip-gram 两种不同的模式可以选择，其中 Skip-gram 适合大型数据集[5]。

由于语言不同、领域不同，词嵌入空间并不是一成不变的，并不是每个都适合的，但是使用 Keras 使用反向传播的 Embedding 层可以根据不同的情境实现一个较好的词嵌入空间。

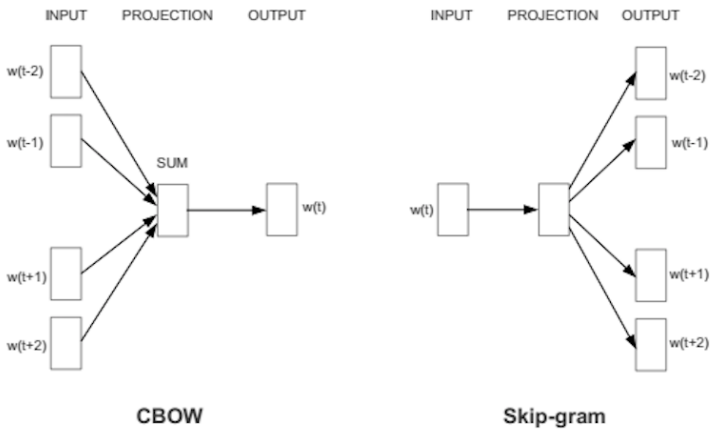

图 3-3 word2vector 的两种方式

Figure 3-3 Two ways of word2vector

3.2.5 预训练的词嵌入

使用词嵌入空间的另一种方式就是使用预训练的词嵌入（pretrained word embedding），当自己的数据量不够学习真正强大的特征时，使用其他

人在极大数据集上训练好的词嵌入是很有用的。他们会在数据量极大、质量很好、范围很大且涉及领域很多的多个文本数据集上进行训练得到预训练的词嵌入。因此预训练的词嵌入的泛化性会比较好，经过自己项目数据集的微调（Fine tuning）之后就会达到很好的效果。

本文尝试使用 Glove 词嵌入，Glove 词嵌入有很多维度可以选择，50、100、150、200、300 等等，选择的词嵌入维度也是很重要的，本文在多次尝试之后最终使用 100 维词向量。

3.2.6 加载到 Embedding 层

不管是 Word2Vector 训练的词嵌入还是预训练的词嵌入，亦或者是前两个都不使用，都需要对 Embedding 层的参数矩阵进行初始化。

第一种方法是，利用训练好的 Word2Vector 模型，逐个输入文本信息，通过模型，逐个返回相应单词的词嵌入，加载到 Embedding 层中。

第二种方法，如果使用预训练的词嵌入只需要把预训练的嵌入加载到相应维度的 Embedding 层当中。

如果说前两种方法都不使用，也必须初始化 Embedding 层的参数矩阵，可以进行全零初始化，在模型训练的时候同步对 Embedding 层进行训练，得到不断进步的词嵌入空间。

3.3 深度学习

3.3.1 概述

本文使用了 10 个自定义的模型，其中包含深度学习模型 7 个。

主要深度学习模型采用了循环神经网络和一维卷积神经网络，在此基础上进行优化创新，加入适当的 Dropout 正则化，recurrent_dropout 循环神经层内的循环正则化，加入注意力机制，提高对关键信息的获取能力。使用

RCNN，结合循环神经网络与卷积神经网络的优点，提高准确性。另外还有自己设计的类似于 Inception 模块和残差网络 ResNet 模块，尝试多种自己设计的模型，建立“宽且深”的网络。

3.3.2 循环神经网络

循环神经网络是专门用来处理时序数据问题的，recurrent neural network，可以用来解决时间时序上的预测问题、可以解决文本问题（可以理解为每个标记（token）是时间步长上的）、同样也是可以用来解决图像问题的。

循环神经网络的精髓在于它拥有一个记忆单元（memory），这个记忆单元可以存储前一个时间步长上的信息，并且加载本次时间步长上的运算信息后传递下去，这样就解决了时间序列问题的前后相关问题。比如说文本问题上，“array Qingdao at ...”和“leave Qingdao at ...”，传统的方法，运算到“Qingdao”时，前边的 arrive/leave 对本次的“Qingdao”是没有任何影响的，但是循环神经网络解决了这个问题。虽然面对的同样是“Qingdao”，但是因为 RNN 拥有记忆，所以会出现两个不一样的结果，“到达青岛”和“离开青岛”。

循环神经网络一般从 simpleRNN 说起，simpleRNN 特点就是简化、便于理解，但缺点也就是过于简化，几乎是没有实际的应用价值的。最大的原因就是 simpleRNN 的记忆单元记录的信息是自始至终的，面对循环次数极多而且没有合适的激活函数，在长期的运算中发生梯度消失或梯度爆炸问题，这样是很可怕的，最后的结果是没有意义的。

所以 LSTM 应运而生，long short-term memory，单词中间的连接符是需要注意的，是 short-term 短时，short-term 前边的 long 只是形容 short-term 相对于其他的方法它的 short-term 是相对稍长的，并不是真正的特别长。

LSTM 解决 simpleRNN 的梯度消失或爆炸问题的方法是加入门控单元，增加中间过程的可操作性，从而完成修改优化。

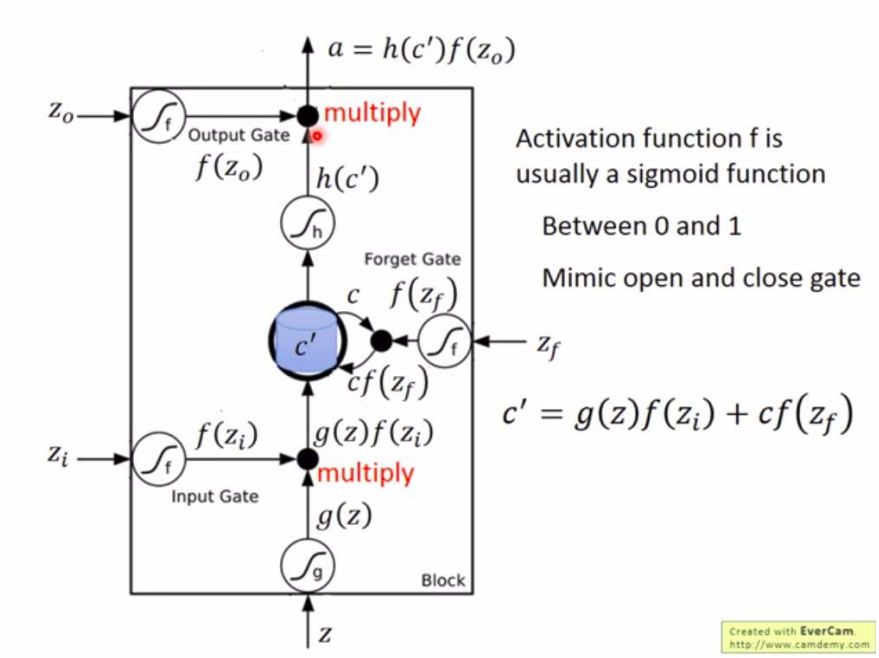

图 3-4 LSTM 门控单元

Figure 3-4 Long Short-Term Memory gate unit

LSTM 拥有三个门控逻辑单元，他们分别是：输入门（Input gate）、输出门（Output gate）、遗忘门（forget gate），三个门控单元在数学实际计算的意义上来说其实是三个参数矩阵，控制着输入输出以及记忆单元的输入输出。其中，最重要的就是遗忘门，遗忘门的开关决定这 memory 记忆单元中的信息是否需要遗忘，需要注意的是遗忘门为 1 是其实不遗忘，为 0 才是遗忘，与普通逻辑相反。

讲完了 LSTM 就需要注意 LSTM 的一个变体，GRU，GRU 是 LSTM 的简化，它只需要两个门控单元即只需要两个参数矩阵，参数更少，训练拟合速度会更快。但是表示能力有时候是不如 LSTM 的，毕竟有得就有失，

很多时候都是表示能力与运算难度的一种折中与取舍。

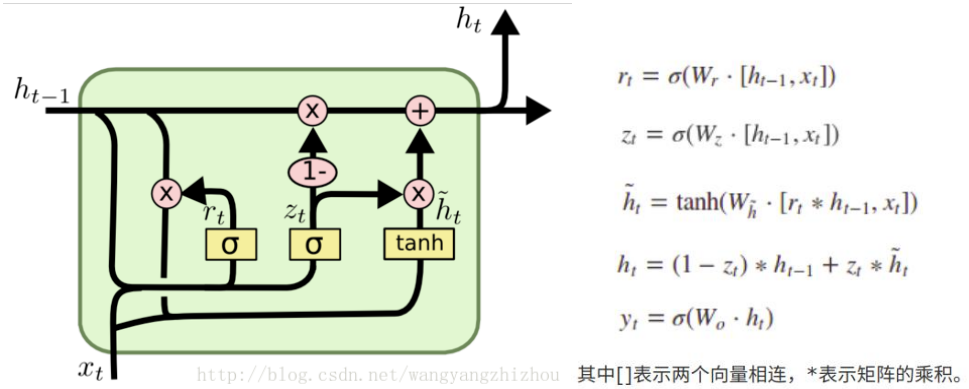

图 3-5 GRU 原理

Figure 3-5 GRU principle

本文使用的主要神经网络支撑就是循环神经网络，用了循环神经网络的一种双向结构，即，每个句子从前往后看并且从后往前看一遍，双向的 RNN 不会比单向的效果差，而且更容易发现一些语义之间的联系。

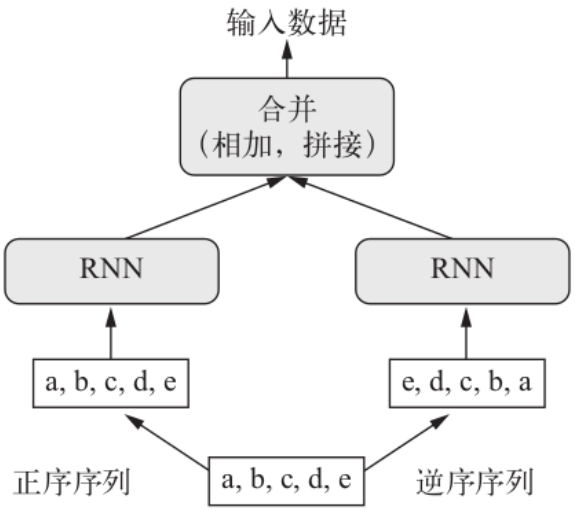

图 3-6 双向 RNN 原理

Figure 3-6 Bi-direction RNN principle

3.3.3 卷积神经网络

一般来说，对于文本处理问题，卷积神经网络并不是最好的解决方法，因为其实卷积神经网络更适合的目标问题是图像处理问题，有用长、宽、深度这三个维度，而文本处理问题是二维的。不过，CNN 并不是不能用来处理文本问题，一维的 CNN 从某些程度上来说，他的很多特性使我们想要去使用的，比如平移不变性，平移不变性可以让模型不用在乎情感出现在哪个部分，就像是二维的它处理图像的时候一样，它不会去在意图片的鼻子、嘴等等必须出现在图像的规定地方。CNN 的另外一个特性就是它的空间上的具有层次性的结构，一开始 CNN 会在意轮廓信息，但是随着空间层次的加深，它会更去在意一些对象的特性。举个例子来说就是，一开始它会在意人脸的轮廓，哪一部分是人脸，随着空间层次的加深，它逐渐会去在意眼睛的形状、鼻孔的大小和形状等等具体的个体的特性信息。这两个特性其实是我们处理文本问题是也想要尝试使用的比较好的特性。

CNN 的原理其实很简单，核心在于自定义大小的窗口在目标上的、以一定步长进行的规则的滑动，滑动的到的结果与卷积核（实际上就是 CNN 的参数矩阵）进行运算，把运算结果进行合并，合并结果大小根据扫描是的 padding 参数来设计，如果“ padding=‘same’ ”，合并后的长和宽与原先相等，只有深度发生变化（这里的深度已经不是图片的颜色深度，已经是图片的参数属性特征了（有时称为过滤器）），如果不设置 padding 的话，运算后的下一层的图片长宽都会缩小。

卷积（convolution）层之后通常会跟一个池化（pooling）层，它的作用是提取关键信息，同时降低图片维度，不会导致随着卷积层数的增加参数量图片大小变得巨大的情况。池化层也是通过窗口实现，通过窗口的扫描，选择本窗口内最有价值的信息，一般会采用最大池化，当然平均池化等在某些情况下也是合适的。

本文使用的是一维的卷积神经网络来处理文本信息，主要还是因为文本本身就只有一个维度，可以对应二维图像的长或宽，时间维度上就是图像意义上的深度。本文尝试通过一维卷积神经网络获得类似于二维卷积神经网络的相关的优秀特性，以此来帮助自然语言的分析理解的问题。

3.3.4 Attention 机制

注意力机制的特点是给 RNN 的运算分析结果增加一个可机器学习的权重参数分布机制，它可以精确到句子和单词两个数量级并且可以从这两个数量级上进行权重分布的机器学习，达到关键信息关键情感的学习与分析处理。

本文采用的是简单的单词级别的注意力思想实现的注意力层（attention layer），借此来进一步提高模型的分析处理能力，更好的进行更适合人类自然语言的分析预处理，最后的相关结果更加合理并且尽可能更加准确。

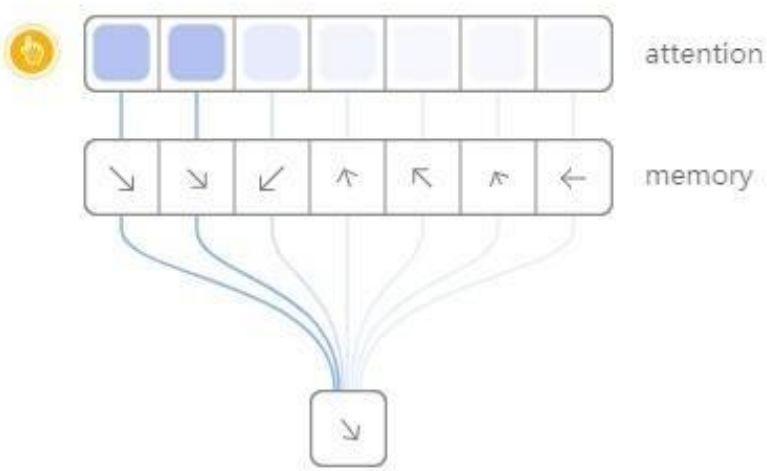

图 3-7 注意力机制原理

Figure 3-7 principle of attention layer

3.3.5 Inception 模块

Inception 模块是一种多模块并联的模型结构模块，每个模块的结构不同，以卷积神经网络为主，每个模块层数不同、步长不同，增加模型的多样性与表示能力。

本文研究的是自然语言文本信息的情感，所以全部采用一维卷积神经网络，本文使用的模块架构如下图所示：

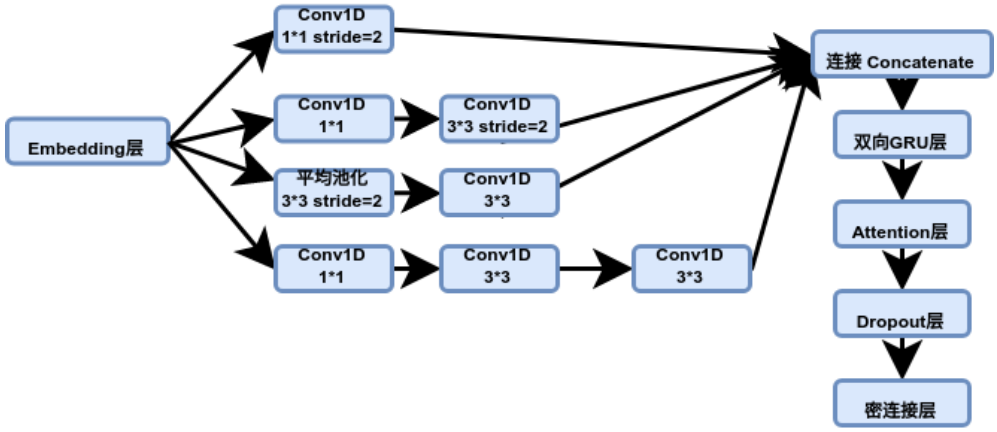

图 3-8 本文 Inception 模型架构

Figure 3-8 Inception structure of this paper

通过本文的架构图可以看到，本模块是一个有向无环图。模块包含了很多 1×1 卷积层，这正是该模块的特色所在，它能够使信息混合在一起而且不会导致不同空间的信息混杂起来。并联的计算路径之间是关联大的，但是其内部的关联性是足够大的，通过这种比较合理的结构往往就能达到不错的效果。

3.3.6 残差网络

残差网络的设计思想在于层与层之间的跨层连接，有时候随着神经网络层数的加多，多层之前的信息可能会消失殆尽而且部分信息对于这一层来说还是很有效果的，最著名的就是 ResNet 网络了。所以就采用了残差连接的结构，上边某层的输入经过其它层，直接接入到本层的输入当中，与本

层原本的输入进行合并操作，这样本层就能实现接收上边某层的直接传递的输入，更好的保留了某些有用的信息。

本文设计的残差网络结构如下图所示：

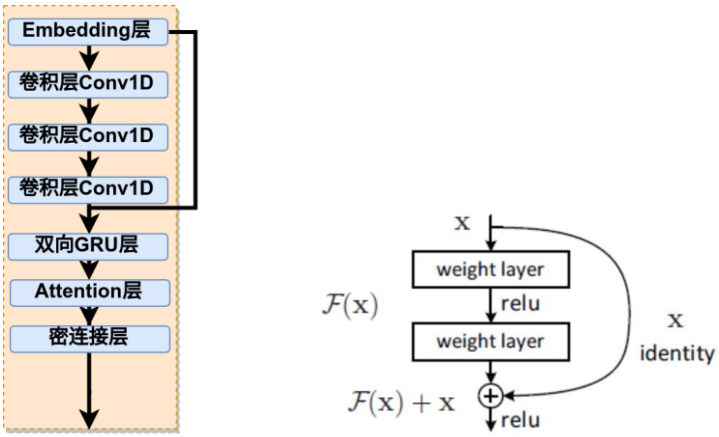

图 3-9 本文残差网络模型架构

Figure 3-9 ResNet structure of this paper

3.3.7 RCNN

在学习相关知识时，通过查阅资料发现在实际的自然语言处理项目（淘宝短文本多分类问题）中，项目拥有者表示在工程的实际应用中使用 RCNN 的效果是比较好的，所以本文也尝试使用了 RCNN 的架构，即 CNN 与 RNN 相结合使用。

RCNN 最明显的优点就是训练速度明显加快，因为 CNN 能够帮助 RNN 提前提取有用信息，所以在训练的时候 RNN 负担会小很多，极大降低了模型整体的训练难度、提升了训练速度。RCNN 的其他优点其实也在于这一点，利用一维的卷积神经网络预先对文本信息进行 CNN 优势特性的处理，利用平移不变性和空间结构层次性，获取更有用、更合适的信息，输入 RNN 当中，利用循环神经网络的语言处理的优势进行下一步的训练学习，达到各取所长的效果。

3.3.8 完整介绍使用的深度学习模型

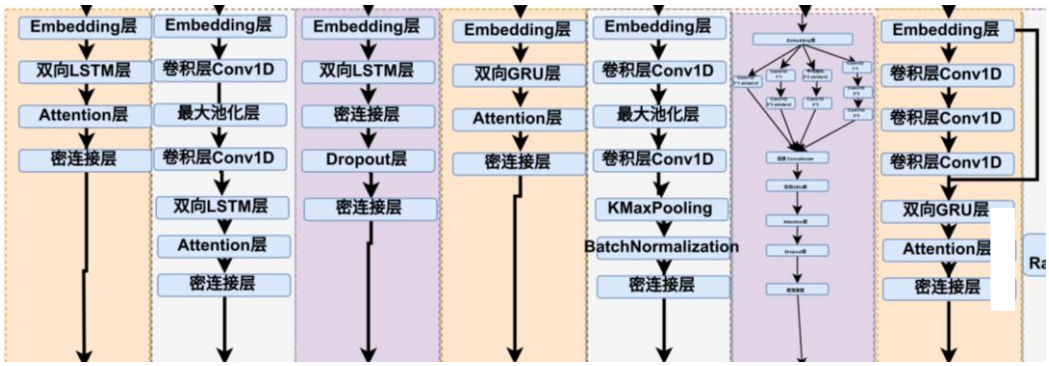

图 3-10 深度学习模块设计结构

Figure 3-10 The structure of deep learning models

Model 1～4 采用自己训练的 Word2Vector 模型进行词嵌入空间的参数矩阵加载，Model5～7 采用预训练的 100 维的 glove 词嵌入进行词嵌入空间参数矩阵的初始化。

所有深度学习 Model 第一层都是 Embedding 层，只是参数矩阵不同，包含两大类。

1. Model_1 在词嵌入之后采用双向 LSTM 层，添加了 Dropout 和本层的层内循环 Dropout（recurrent Dropout），之后引入注意力机制注意力层，进一步优化 RNN 计算结果，更好的提取感情信息。之后是深度学习的密连接层（Dense 层），这一层的神经元数量是 1（二分类问题，输出 0 或 1)，采用 sigmoid 激活函数，约束输出区间在 0~1 之间。

2. Model_2 采用 RCNN 结构，结合了卷积神经网络和循环神经网络，在 Embedding 层之后接入一维卷积层，接上最大池化层，再接上第二个一维卷积层，此时 CNN 信息提取阶段结束，继续接入双向 LSTM 层，同样包含层外与层内两种正则化手段。之后接入最后一个与 Model_1 相同的密连接 Dense 输出层。

3. Model_3 与 Model 类似，不同的是 Model3 放弃使用 attention 机制，在双向 LSTM 之后接入含有 20 个神经元、激活函数为‘relu’的 Dense 密连接层，之后加入必要的 Dropout 正则化层，随机将上一层的 50%初始化为 0，之后接入 sigmoid 密连接输出层。

4. Model_4 与 Model_1 几乎完全相同，将 LSTM 换成了改进的 GRU，调整到合适的超参数。

5. Model_5 采用了纯 CNN 结构，Embedding 加一维卷积加最大池化加一维卷积，之后加入了动态的 KMaxPooling 层，这是一个适合自然语言处理的最大池化层，它是 Max Pooling Over Time 算法的演进，它的作用是只对 Filter 过滤器中大小 TOP-K 个特征值进行提取，这就是它与传统最大池化的最大的区别，尽可能多且有序的保存了之后可能会用到的特征信息。

6. Model_6 根据 Inception 模块的启发建立了一维卷积神经网络的 Inception 模块，Embedding 层之后并行接入四条路径通道，第一条直接入一个步长为 2 的、窗口大小为 1×1 的一维 convnet；第二条接入一个步长为 1 的 1×1 的一维 convnet，接着接入步长为 2 的 3×3 的一维 convnet；第三条路径先直接接入一个步长为 2 的、窗口大小为 3×3 的平均池化层（average pooling），接着接入一个步长为 1 的 3×3 的一维 convnet；第四条路径接入步长为 1 的 1×1 的一维 convnet，接着接入步长为 1 的 3×3 的一维 convnet，之后还是步长为 1 的 3×3 的一维 convnet；随后把四条路径全部连接（concatenate）起来。

在自己构建的一维 Inception 模块之后，接入双向 GRU 层，跟着自注意力层、Dropout 层和 sigmoid 输出密连接层。

7. Model_7 受著名的 ResNet 启发，自己尝试构建一个参差神经网络，首先在 Embedding 层之后串联三个一维 CNN，之后正常接入双向 GRU 层加 attention 层加输出密连接层。与一般网络不同的在于将第一个一维 CNN

的输入（即 Embedding 层的输出）跳跃的直接跳过三个卷积神经网络层，直接进入 GRU 层当中，在一定程度上保存了有用的信息。

以 Model_3 为例，下图是利用 keras 的 plot_model 工具自动绘制的模型详细信息图：

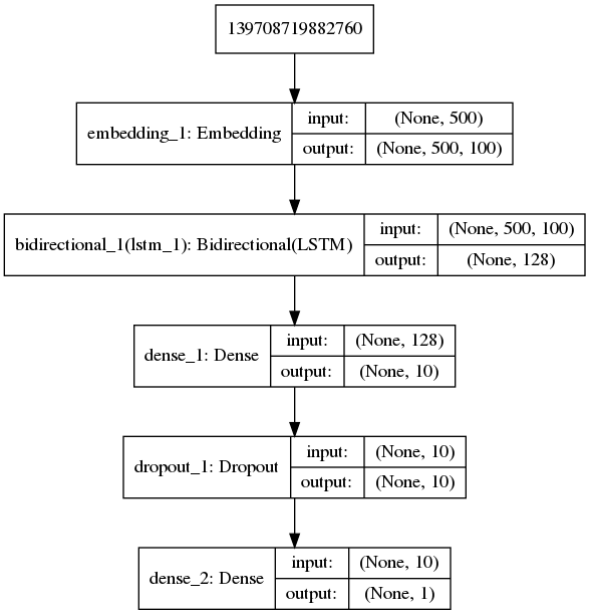

图 3-11 模型详细超参设计

Figure 3-11 The detailed Hyperparameter of model_3

3.4 机器学习

3.4.1 概述

添加传统的机器学习算法与之前的深度学习算法进行结合是受 keras 之父的模型集成建议还有 stacking 算法的启发所想到的，封装好的 stacking 算法通常用于传统机器学习方法的集成学习，但是 Fran ̧ois Chollet 提示说深度学习方法如果能跟传统机器学习树形方法集成起来的话效果是不错的，所以本文在深度学习方法之外创新加入了传统机器学习方法。

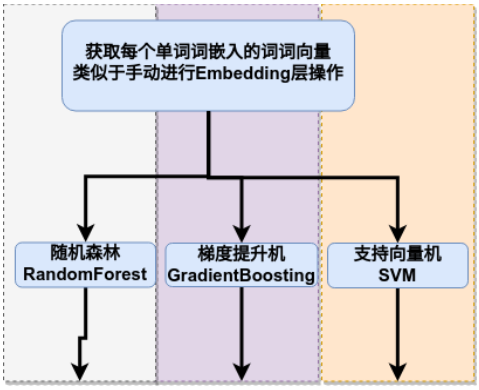

图 3-12 传统机器学习模块设计结构

Figure 3-12 The structure of traditional machine learning models

3.4.2 RandomForest

使用的第一个传统的机器算法是随机森林算法，原因上边已经解释过了，随机森林就是传统机器学习里树形结构的一种，它由很多决策树组成，决策树之间的集成互补也就形成了更为泛化的随机森林。随机森林里需要的随机树的数量可以使用 n_estimators 参数进行设置，根据自己任务的难度进行设置。并且可以通过绘制 ROC 曲线图像的方法进行想用的调优操作，达到计算负载与实际效果的良好折中。

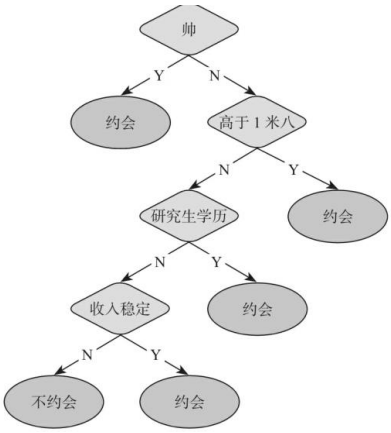

图 3-13 决策树示例

Figure 3-13 Decision tree example

3.4.3 Boosting

又称为提升法，属于机器学习但是也不是特别传统的机器学习方法了，尝试使用了两个 GradientBoosting、Adaboost 方法，最后主要使用梯度提升法。这个机器学习模型是在集成的过程中逐步添加弱学习器，添加不是简单的添加，而是每一次添加都会对之前的做出有效的改正。而且相比于 Adaboost，它对于新的学习器的操作是使它们去拟合上一个学习器的残差。

3.4.4 SVM

支持向量机是比较难理解的一个机器学习模型，难点在于核方法等。简单来说，支持向量机利用核技巧（kernel trick）会把数据映射到高维度的空间内，用核函数（kernel function）计算空间内点间距离，更好的利用超平面来获得表示[2]。支持向量机作为一种比较浅层的方法，结合其他的模型可能会发挥比较好的效果，但是支持向量机本身不算特别适合大型的数据集，表现不算好。

3.5 模型集成融合

3.5.1 概述

前边提到了要做模型集成融合的利用，想把深度学习模型与传统机器学习树形模型进行结合，想尝试达到不错的效果。模型的集成融合其实是机器学习的相关比赛中参赛选手们常用的技巧，一般会利用 bagging 或者 stacking 进行。个人比较喜欢具有机器学习能力的 stacking，但是普遍的方法是用 stacking 集成传统机器学习模型，几乎没有人用来集成深度学习，更没有人利用这种思想集成深度学习和传统机器学习。所以本文尝试使用这样的思想，自己利用 stacking 思想重写模型集成融合的算法，实现深度学习模型和传统机器学习模型的集成融合。

利用 stacking 思想进行算法的重写，运用在自己的模型上，其关键在于利用下几层的混合学习器对多个模型的输出结果进行学习，输出集成之后的输出结果。

输入：训练集$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$；单模型学习算法$\xi_1,\xi_2,\cdots,\xi_T$；融合模型学习算法$\xi$。

过程：

1: $for\ t = 1,2,\cdots,T\ do$ //基于原始数据训练T个单模型

2: $h_t=\xi_t(D)$

3: $end\ for$

4: $D'=\emptyset$

5: $for\ i=1,2,\cdots,m\ do$

6: $for\ t=1,2,\cdots,T\ do$

7: $z_{it}=h_t(x_i)$ //单模型输出作为样本新特征

8: $end\ for$

9: $D'=D'\cup((z_{i1},z_{i2},\cdots,z_{iT}),y_i)$ //单模型输出和样本标记构建新训练集

10: $end\ for$

11: $h'=\xi(D')$ //在新训练集上训练二级模型

输出：$H(x)=h'(h_1(x),h_2(x),\cdots,h_T(x))$。 //二级模型输出作为Stacking输出

$$HH(x)=H(h_1'(h_1(x),h_2(x),...,h_T(x)),h_2'(h_1(x),h_2(x),...,h_T(x)),h_3'(h_1(x),h_2(x),...,h_T(x)))$$

图 3-14 集成学习模块

Figure 3-14 Integrated learning

stacking 原则上可以分成若干层级，但层数多不一定效果越好，一般采用两层结构。因此，我们可以把 stacking 过程看作是两个级别，级别 0 和级别 1。但本文选择使用三层 stacking 结构，原因很简单，因为对于本文的测试结果确实有提升。

0 级：也就是对应第一层，0 级模型就是我们需要堆叠的多个模型，也称为预测器。可以是随机森林 SVM 等等多种模型，也可以只有一种模型。对这些模型在训练集上进行正常训练。

1 级：也就是对应第二层，1 级模型称为混合器或元学习器（blender, or a meta learner）。混合器的训练集的输入特征（x）是上一层多个预测器的预测值的拼接，混合器的训练集的预测值（y）就是训练集原来的预测值（y）。

stacking 一般采用预留集的方法训练 1 级模型（元学习器），如图所示，训练集被拆分为 2 部分（注意，这里不是要几折也不是要拆分成训练集验证集测试集），子集 1 正常训练模型，得到 n 个预测器。让刚刚得到的 n 个预测器在预留集（子集 2）上进行预测，（由于之前一直没有见过子集 2 里的数据，所以可以确保预测是干净的），这样就得到了 n 个预测值。将这些预测值作为输入特征，创建一个新的训练集（n 维），保留原来的 label 作为 label，在这个新的训练集上训练 1 级模型（元学习器/混合器），让它学习使用预测器的预测来预测目标值。

需要注意的是训练数据处理完之后并不是把所有数据全部放入训练，而是分成两个部分，一个作为真正的训练集用于模型的训练，另一个作为作为验证集只用来评价模型并不参与训练（但是也会有部分泄露），此部分的数据不仅可以作为验证集，对于实现 stacking 算法还将发挥最重要作用。

因为本文采用三层的 stacking 结构，所以分出来的数据还需要再分一次，这样就得到了三部分的数据。第一部分数据已经经过训练，不能再用于其他任何用处，因为他们已经不是 clean 的了；第二部分数据拿出来，放入需要集成的 10 个模型当中，让这 10 个模型对这第二部分数据进行预测（model.predict），预测值作为 x，第二部分数据真实的、最初的 y 继续作为 y，这样就构建了包含 x 和 y(label)一个新的训练集。之所以称之为训练集是因为这部分数据要作为 stacking 的第二层（也就是第一层混合器）的训练集进行训练。训练好的模型就可以接受并拼接 10 个模型对测试集的预测值（y）作为本层混合器的 x 特征值，通过已经习得的模型成果，对 x 特征进行本层的预测，三个混合器共输出三组预测值。

第三部分数据则与上边同理，让上一层的三个混合器对第三部分数据的特征进行预测，采用同样的动作构建 x 和 y 的数据集作为最后一层的训练集。最后一层的混合器对这些数据进行学习，接受并拼接上一层的三组预测值，作为 x 进行对 x 的预测，输出最终的结果。

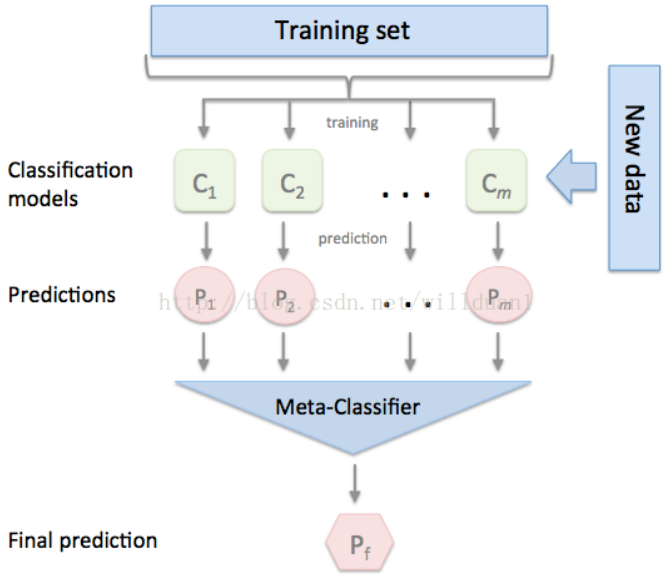

图 3-15 集成学习训练集选用

Figure 3-15 Training set selection of integrated learning

一种更为复杂的方法是使用 k-fold 交叉验证来开发元学习机模型的训练数据集，如下图所示。每个 0 级模型预测器都使用 k-fold 交叉验证(甚至为了达到最大效果使用留一法交叉验证)进行训练;然后模型被丢弃，但是预测被保留。这意味着对于每个模型，都有一个模型版本所做的预测，而这个版本的模型并没有针对这些例子进行训练，例如，有一些在预留的例子。

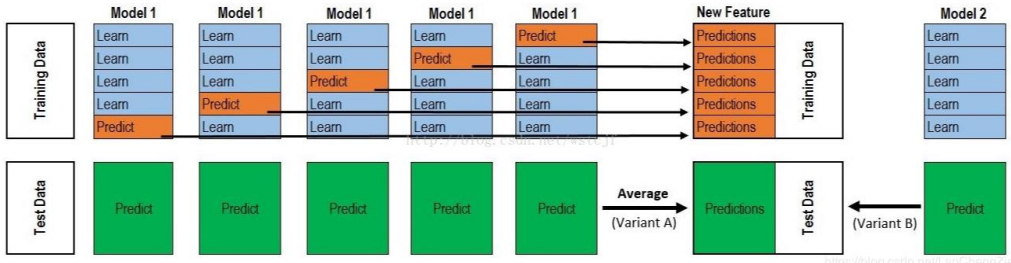

图 3-16 一种五折模型堆叠

Figure 3-16 One five-fold model stack

3.5.2 混合器结构

第一层混合器使用了线性回归模型、随机森林、XGBoost。其中线性回归模型是很符合 stacking 思想的回归模型，进行第一次线性回归。第二个混合器采用了随机森林，这个是比较喜欢的模型所以就也尝试在一层使用了这个模型。而 XGBoost 是比赛中常用的而且效果很好的模型，它与随机森林一样也是与决策树算法相关的。

第二层的混合器采用了神经网络的密连接层，因为在模型测试中发现，stacking 只有传统的两层结构时，密连接层效果不错，所以在三层时也尝试令最后一层为密连接层，最后效果也不错。

通过最终实验测试，这样的效果

3.6 系统部署

3.8.1 模型训练

模型训练主要在 Kaggle 竞赛提供的 kernel 上进行训练，因为本机配置不高，并行训练不可能而且训练使无法进行其他工作，效率很低。所以使用 Kaggle 竞赛提供的 kernel，同时开启多个 kernel 并行训练多个模型，对

训练好的模型继续进行超参修改、层结构修改及调优，继续提交训练，反复循环进行。

3.8.2 模型部署

模型部署在模型训练并测试完成之后，保存要使用的模型到本地。模型部署模块使用 Flask 框架，利用 keras 的 load_model 函数对保存好的模型进行加载。前端提交要测试的文本数据，后端对这个文本数据进行与模型训练前相同的数据预处理，之后就可以利用 model.predict 函数就可以进行最终的输出，最后只需要将结果返回到前端进行展示即可。

3.8.3 开发环境

深度学习开发基于 keras 框架，机器学习主要基于 Scikit-Learn，模型部署基于 Flask 框架。

表 3-1 开发环境详细信息表

开发环境		详细信息
硬件	CPU	Intel® Core™ i5-5200U CPU @ 2.20GHz ×4
	显卡	GeForce 920M/PCIe/SSE2
	内存	7.7 G
软件	操作系统	Ubuntu 18.04 （64-bit）
	开发环境	Python 3.6.8
		Keras 2.2.4
		Scikit-Learn 0.20.3
		TensorFlow-GPU 1.13.1
		CUDA 10.0 / cuDNN 7.3.1
		Flask 1.0.3

P.S. 很多模型训练由于本身开发环境配置限制无法快速完成，
使用了 Kaggle 竞赛提供的 Kernel 进行模型训练调优，
Kaggle Kernel 相关标准配置 Kaggle 网站可查。

3.8.4 Keras 框架

François Chollet 的 Keras 深度学习框架，可以比较方便的去定义、去训练几乎所有的可能的深度学习模型。

之所以选择 Keras 是因为：

1. 接受老师的建议，学习了《Python 深度学习这本书》，这本书的作者就是 François Chollet，本书详细介绍了深度神经网络，介绍了 Keras，并且整本书都是基于 keras 写的，进行了系统的学习与理解。

2. Keras 简答易用，很容易上手，对于使用者来说 keras 的 API 接口十分方便、功能函数封装性强，代码简洁易懂，也就更容易进行学习和理解。

3. Keras 可以选择基于 TensorFlow 进行，相比 TensorFlow 复杂而言，keras 的简洁封装给人很好的感觉，起码对于初学者是很好的，但是后期的学习，TensorFlow 还有 Pytorch 都是必须的，他们的步骤更像是整个步骤的几乎完整的复现，所以更适合入门以后的进一步学习，这些也是我以后一定会去学的。

也就是说，Keras 也是有一些缺点需要注意和避免的。

由于 Keras 对 TensorFlow 的封装太过于优秀，在我们使用方便的同时可能会造成内部原理被忽略的情况，Keras 代码简单简洁，一行函数包含的信息很多很多，所以一定要学会不仅仅是调用函数，更要理解内部的原理，只有这样才能更好的理解深度学习、才能更好的创建和优化自己的模型。有时候虽然看起来 TensorFlow 很复杂、代码量大、逻辑复杂，但是是可以更好的理解神经网络原理和运行训练过程的，近几年很火的 Pytorch 也是同

理。所以不是说 keras 是最好的，只是现阶段比较合适的，今后一定会多多学习其他的框架。

3.8.5 Scikit-Learn 和 Flask 框架

Scikit-Learn 是机器学习最常用的 Python 库之一，其中不仅集成了多种机器学习模型，同时集成了很多有用的工具以及很多质量比较好的数据集，数据集可以直接利用函数进行 load 加载，用于简单的模型测试等很方便。Scikit-Learn 库里的机器学习等模块都是经过高度集成化的，使用时简单几行代码就直接调用，所以这种特点的优缺点有点类似于 keras，提高了开发的效率，但是也在一定程度上限制了开发者的自我学习扩展空间。

Flask 是 Python 语言的后端应用服务器 MVC 框架，架构简单、使用方便。利用路由进行页面以及前后端的连接结合，利用 wtforms 进行表单设计，对路由过来的 get 或 post 请求利用自定义的 Python 函数以及相应的页面进行响应。

3.7 本章小结

算法部分是本文的核心，所以本章篇幅较长、涉及方面较多，本文尝试使用比较易懂的语言介绍了算法的架构与每一部分模型的思想，介绍了自己的理解与为什么要进行这种架构的原因，把自己的理解与学习贯穿于其中。

4 系统测试

概述：

本章将介绍毕业设计的系统运行测试调优，介绍数据集详情，介绍最后的系统整体测试。

4.1 数据集

数据集采用 IMDB 和 twitter 数据集。

IMDB 数据集来自很多论文都使用而且 Kaggle 比赛的比赛公开数据集，有排名得分等。数据集包括 25000 条标注好极性的训练集，未标极性的 25000 条测试数据集。情感标注信息里，0 为消极，1 为积极。Twitter 数据集也是一样的。

表 4-1 IMDB 训练数据集

id	情感	评论
5814_8	1	With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there,…
2381_9	1	\The Classic War of the Worlds\" by Timothy Hines is a very entertaining film that obviously goes to great effort and lengths to faithfully recreate H. G. Wells' classic
7759_3	0	The film starts with a manager (Nicholas Bell) giving welcome investors (Robert Carradine) to Primal Park . A secret project mutating a primal animal using fossilized
3630_4	0	It must be assumed that those who praised this film (\the greatest filmed opera ever,\" didn't I read somewhere?) either don't care for opera, don't care for Wagner,

表 4-2 IMDB 测试数据集[7]

id	评论
12311_10	Naturally in a film who's main themes are of mortality, nostalgia, and loss of innocence it is perhaps not surprising that it is rated more highly by older viewers than younger ones. However there is a craftsmanship and completeness to the film which anyone can enjoy. The pace is steady and constant, the characters full and engaging, the relationships and interactions natural showing that you do not need floods of tears to show emotion, screams to show fear, shouting to show dispute or violence to show anger. Naturally Joyce's short story lends the film a ready made structure as perfect as a polished diamond, but the small changes Huston makes such as the inclusion of the poem fit in neatly. It is truly a masterpiece of tact, subtlety and overwhelming beauty.
8348_2	This movie is a disaster within a disaster film. It is full of great action scenes, which are only meaningful if you throw away all sense of reality. Let's see, …
5828_4	All in all, this is a movie for kids. We saw it tonight and my child loved it. At one point my kid's excitement was so great that sitting was impossible. However, I am a
7186_2	Afraid of the Dark left me with the impression that several different screenplays were written, all too short for a feature length film, then spliced together clumsily into

4.2 模型调优

模型调优部分可以使用简单的利用模型 fit 时返回的 history 对象进行 matplotlib.pyplot 的图像绘制，包括训练和验证的精度和损失值。

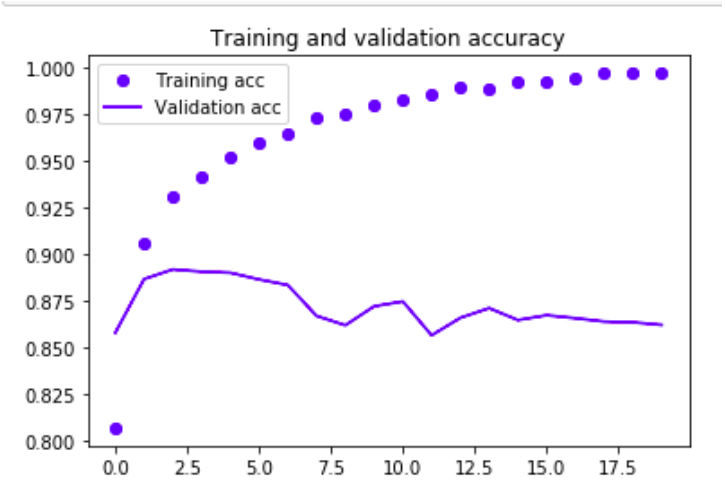

图 4-1 模型训练精确度

Figure 4-1 Accuracy during model training

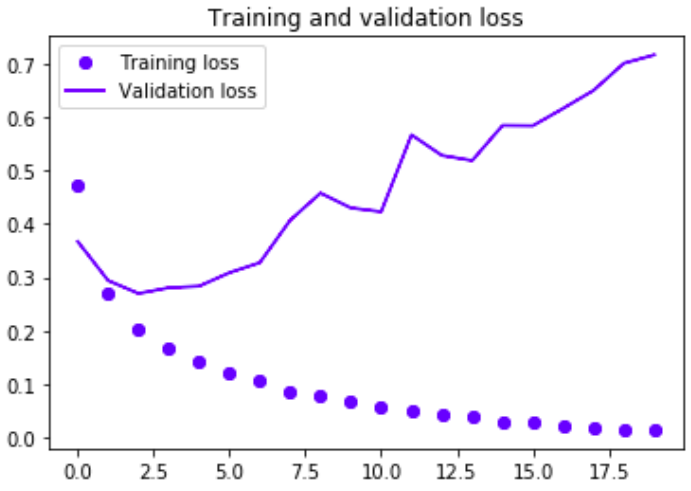

图 4-2 损失值

Figure 4-2 Loss values

更详细和精确的模型训练过程中的数据可以通过 Tensorboard 进行查看，只需要在训练时添加对应数据的回调函数，之后在模型训练时机会记录相应对象的训练参数性能数据。训练之后只需要将 Tensorboard 路径指示到刚才记录数据的文件夹，启动 Tensorboard 服务后就可以通过浏览器打开相应端口进行详细数据的查看，包括模型结构以及每一层的训练情况记录。

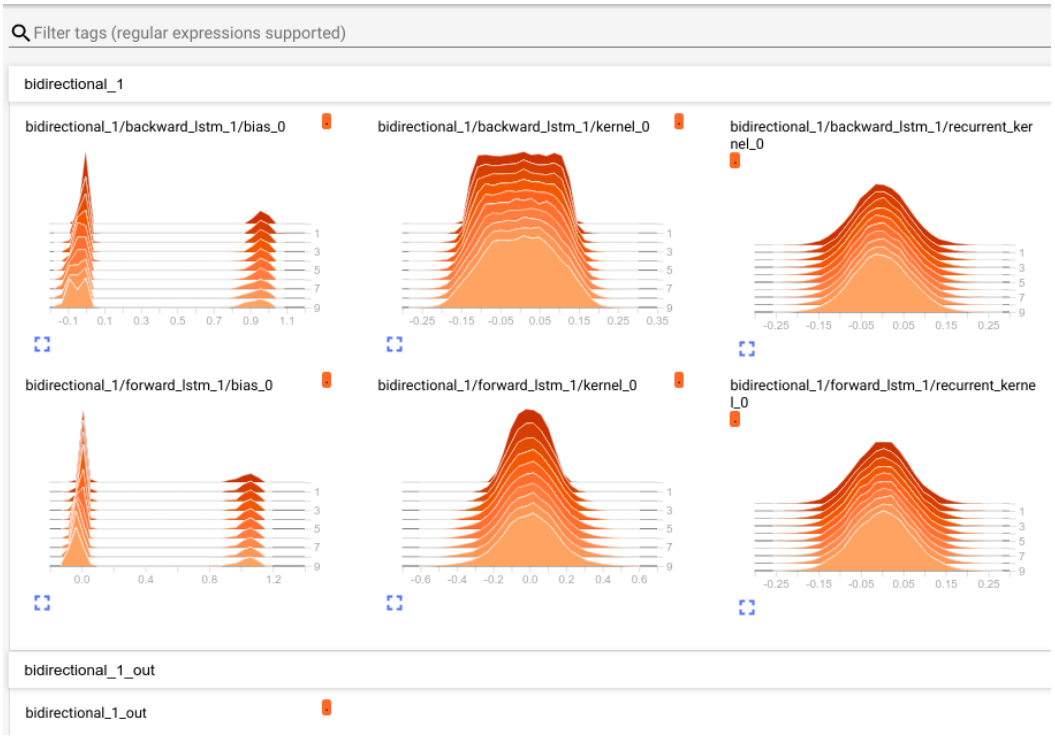

图 4-3 Tensorboard 调优

Figure 4-3 Tensorboard Tuning

4.3 整体测试

在第一个 IMDB 数据集上经过 AUC 评分，计算重合的面积，可以达到 95.97%分，排名能达到前 15%。

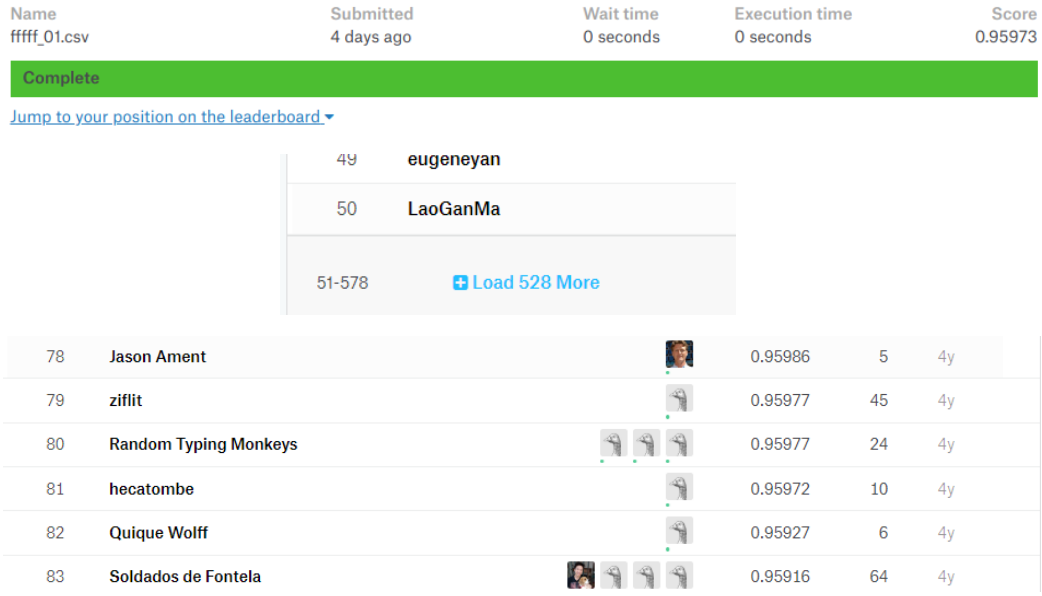

图 4-4 IMDB 比赛分数

Figure 4-4 Score of IMDB competition

在第二个 twitter 数据集上经过 F1 Score 的评分方法，得到了 0.7131280389 的分数，排名 196/614，30%左右。

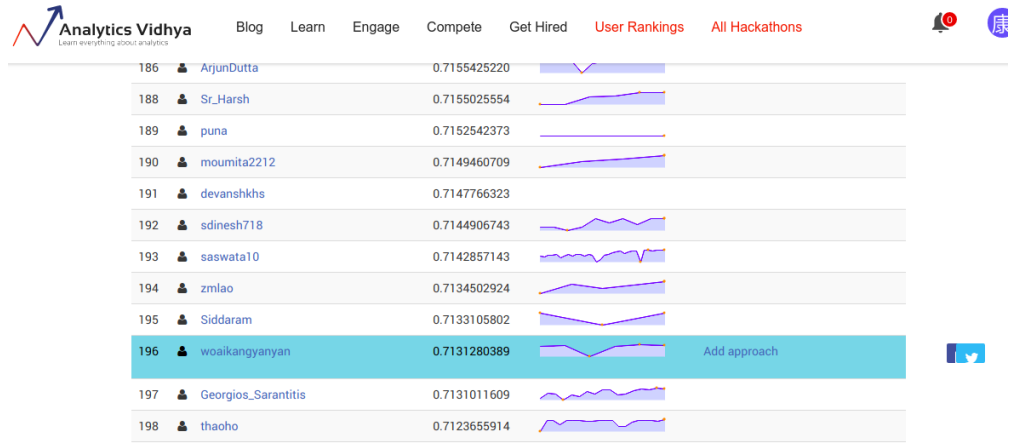

图 4-5 twitter 比赛分数

Figure 4-5 Score of twitter competition

系统整体测试：

图 4-6 情感分析结果（A）

Figure 4-6 Result of sentiment analysis A

图 4-7 情感分析结果（B）

Figure 4-7 Result of sentiment analysis B

4.4 本章小结

本章主要介绍了数据集的情况，以及模型进行调优时的一些方法，最后展示了本文设计模型的最后的在相应数据集上的结果，很多地方还可以继续完善和提高。

5 总结与展望

5.1 总结

本文设计了 7 个深度学习模型和 3 个传统机器学习模型，经过训练集数据的训练拟合，用于测试数据的情感分析结果预测。深度学习模型是经过这一段时间的学习发现的比较好的方法，很多模型都是对其他优秀方法的思想的吸收归纳，把最好的思想用到自己的模型当中去，不断地测试和调优，提升模型性能的同时也提高着自己。

本文的创新点在于实现了对自己构建的深度学习模型与传统机器学习模型的堆叠，因为本文不是多个模型之间的加权或者投票，而是使用了两层的机器学习算法实现了基于机器学习和深度学习的模型选择，相当于三层的模型融合结构。本文的混合算法是对与多个模型输出结果的学习，从而可以更好的提升效果。

模型的集成融合在比赛中是比较常见的，但是一般都只是简单的调用别人封装好的算法，一行代码就完成了集成。但是本文是利用最复杂的基于机器学习的算法思想，构建自己想法的集成融合算法。这样的好处不仅仅是对于算法本身的详细的理解，这样的算法扩展性高，自定义能力强。因为都是自己自定义的结构，所以调优过程更加可操作，更能达到自己想要的效果，而不是没有自主性的依赖于别人的算法。

5.2 展望

下一步的工作将会在以下几个方面展开：

1. 尝试利用自己的算法提升中文文本的情感分析准确度。

2. 彻底弄懂最近很火的 Bert 算法，能够吸收其最精华的优点，把这种思想用于今后的文章和算法当中。直接利用 Bert 模型的话会掩盖其他算法，失去自主性。
3. 创造出真正一点一滴都属于自己的算法，不调用别人任何方法而且能够达到比较好的效果，这样就是真正的自己对算法有很好的理解和感悟。

5.3 本章小结

本章介绍了对于毕业设计论文进行过程的总结，说明了本文的不足之处以便于下一步的工作，介绍了今后如何在学习生活中不断学习、不断对自身进行补强和发展。

参考文献

[1] Francois Chollet,张亮. Python 深度学习[M]. 北京: 人民邮电出版社, 2018, 168-172

[2] Francois Chollet. Deep Learning with Python[M]. USA: Manning Publications, 2017, 180-194

[3] Aurélien Géron,王静源,贾玮,边蕤,邱俊涛. 机器学习实战 基于 Scikit-Learn 和 Tensorflow[M]. 北京: 机械工业出版社, 2018, 181-184

[4] 王子牛,吴建华,高建瓴,陈娅先,王许.基于深度神经网络和 LSTM 的文本情感分析[J].软件,2018,39(12):19-22.

[5] 关鹏飞,李宝安,吕学强,周建设.注意力增强的双向 LSTM 情感分析[J].中文信息学报,2019,33(02):105-111.

[6] 马远浩,曾卫明,石玉虎,徐鹏.基于加权词向量和 LSTM-CNN 的微博文本分类研究[J].现代计算机(专业版),2018(25):18-22.

[7] Pelaez A, Ahmed T, Ghassemi M. Sentiment analysis of IMDb movie[J]. Machine learning, 2015, 198(536):1-7.

[8] Subarno Pal,Soumadip Ghosh,Amitava Nag. Sentiment Analysis in the Light of LSTM Recurrent Neural Networks[J]. International Journal of Synthetic Emotions (IJSE), 2018, 9(1):33-39.

[9] 金志刚,韩玥,朱琦.一种结合深度学习和集成学习的情感分析模型[J].哈尔滨工业大学学报,2018,50(11):32-39.

[10]彭丹蕾,谷利泽,孙斌.基于 SVM 和 LSTM 两种模型的商品评论情感分析研究[J].软件,2019,40(01):41-45.

致谢辞

大学生活转瞬即逝，感谢学院 4 年来的培养以及未来 3 年的培养，感谢学习生活中遇见的所有老师和同学们。

感谢毕业设计指导老师赵中英老师对与本次毕业设计的指导与帮助，感谢老师的悉心解答与教诲。感谢很多老师和同学们在论文以及应用开发过程中的帮助，同样由衷感谢我的研究生导师。感谢舍友几年来的陪伴，更感谢父母对于二十几年来的培养与支持，这是我前进路上最大的支撑与激励。目前进行的学习还是远远不够的，在之后的学习中还要更深入的学习相关的数学知识，进一步提升代码能力，能够实现真正的从自己从公式开始设计的机器学习模型。

大学生活即将结束，在此，祝赵中英老师以及所有老师与同学们，工作顺利、事业有成。