第3章

视频讲解

面向中文金融文本情绪分类的预训练模型对比

3.1 模型对比目的

2019—2021年的两年多时间里，中文预训练语言模型爆发性增长，诞生了很多已预训练中文预训练语言模型，既有面向不同任务的通用模型，也有面向特定任务优化的专门模型；预训练使用的语料知识既有通用领域，也有特定领域。句子级短文本情绪分类的已有方法都有一个前提假设：假设一个单句中只表述了一种感情、情绪或观点。这一假设导致模型只能判断句子整体层面陈述级的情绪，会忽略细节，只适用于常规型观点的句子，对包含多种情绪的单句不适用。

本章的目的是针对中文金融文本情绪分类任务，对当前主流的预训练语言模型，旨在通过模型效果对比，分析每个模型的优缺点，更加深入地了解其原理，剖析每个模型效果好坏的内部机制，达到以下模型对比目的。

第一，比较不同的中文预训练语言模型的骨干网络架构、特征提取方法、关键参数、训练语料库等规律和差异。

第二，运用不同的预训练语言模型软件框架实现技术、不同的分类任务关键评估指标的评估方法。

第三，通过评测结果的对比分析与剖析，选择出合适的模型在第3、4章进行模型改进。

3.2　项目技术原理

本章项目技术原理就是预训练语言模型关键技术，详见第 2 章。

3.3　对比实现方法

本章项目运用到的实现方法有描述性统计方法、统计量分析方法、Python 软件库调用方法、软件框架应用程序编程接口编程方法、深度学习预测方法、预训练语言模型微调方法、分类任务评估指标评估方法等。

3.4　标准流程步骤

跨行业数据挖掘标准流程(Cross-Industry Standard Process for Data Mining，CRISP-DM)是数据分析产业应用事实标准，是分析、数据挖掘和数据科学项目中最受欢迎的方法，在各种已有数据库知识发现(Knowledge Discovery in existing Databases，KDD)过程模型中占据重要位置，包括业务理解(business understanding)、数据理解(data understanding)、数据准备(data preparation)、建模(modeling)、评估(evaluation)、部署(deployment)等数据挖掘和分析项目生命周期的 6 个步骤，工作处理流程如图 3.1 所示，所有步骤本质上都是迭代的。按照 CRISP-DM 方法就能够追踪数据挖掘和分析项目，这与具有不同生命周期模型的软件工程项目类似。

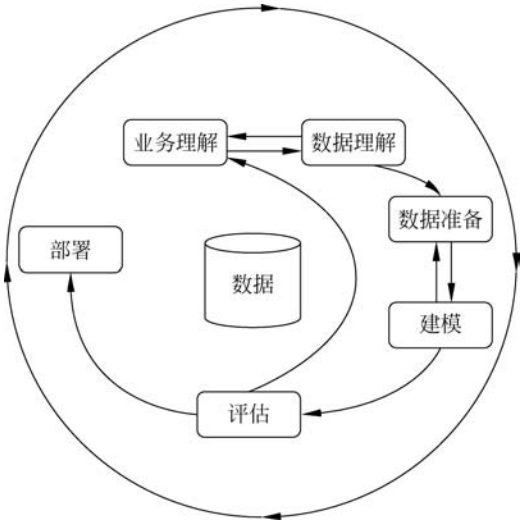

图 3.1　CRISP-DM 参考模型工作处理流程

然而，CRISP-DM 没有指定生产场景中的数据采集阶段(data acquisition phase)，

数据科学家和工程师经常花费大量的时间来定义进行有用实验的技术和质量要求。工程应用的数据挖掘方法（Data Mining Methodology for Engineering applications，DMME）是专为工程应用而设计的 CRISP-DM 方法学的整体扩展，在数据理解步骤前增加了技术理解和概念化（technical understanding & conceptualization）、技术实现和测试（technical realization & testing）两个步骤，在部署步骤中增加了一个技术实施（technical implementation）任务，如图 3.2 所示，为工程领域内的数据分析提供了沟通和规划基础。

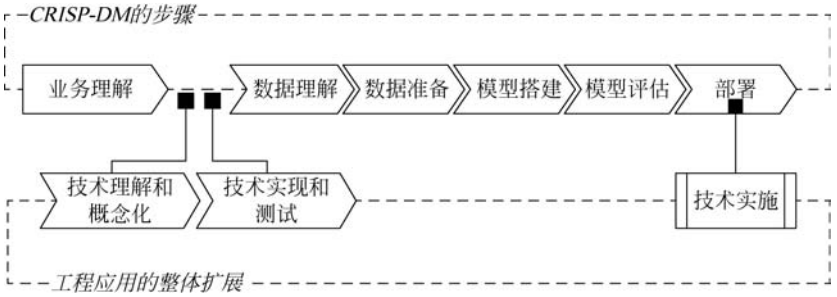

图 3.2　DMME 在 CRISP-DM 上新增的步骤和任务

图 3.3 为 DMME 参考模型工作处理流程。

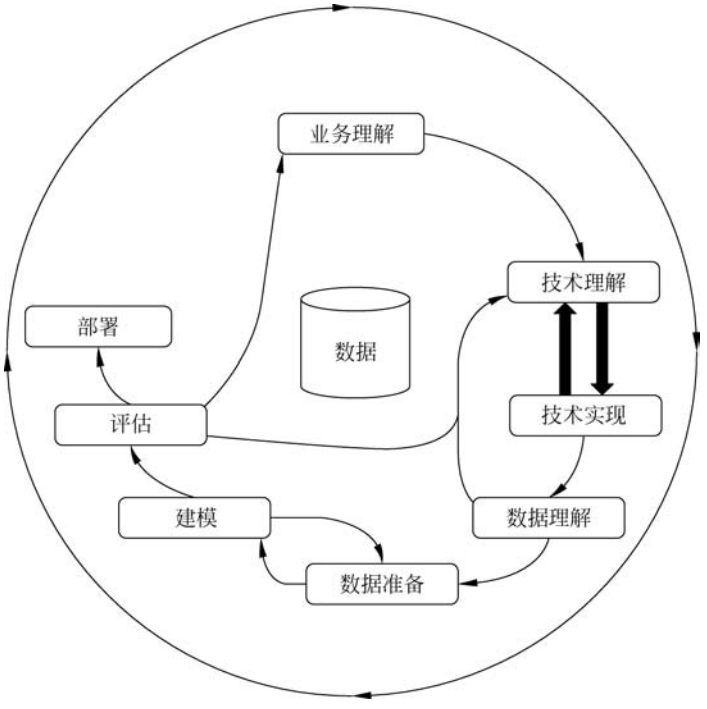

图 3.3　DMME 参考模型工作处理流程

本书符合 CRISP-DM 和 DMME 参考模型的方法。第 1 章是业务理解的步骤，第 2 章是技术理解和概念化的步骤，第 3～5 章涉及技术实现和测试、数据理解、数据准备、评估的步骤，第 4、5 章还涉及建模的步骤。

本章项目具体的操作步骤是，首先对中文金融文本情绪分类标注语料库进行数据采集和数据集成，然后进行数据划分和描述性统计分析，选择合适的开源中文预训练模型，对模型实现和评估，最终对评测结果进行汇总分析。

3.5　自建（评测）标注语料库

按开放方式不同，数据集（dataset）可以分为公开数据集和（非公开）私有（个人）数据集。公开数据集一般是由研究所、比赛举办方或公共机构公开发布，用来协助算法建模研究，适用于模型评测、结果比较和复现；使用公开数据集得出的结论往往更具说服力，便于审稿人对公开发表论文的评审，容易得到业内人士的认可，成为业内公开评测模型性能的基准数据集（benchmark dataset），从而得到当前最佳模型排行榜，数据使用者必须遵守数据集所有者的协议要求。公开数据集的缺点是尽管已经进行脱敏处理来保护个人隐私安全，但仍要避免隐私攻击（恶意攻击者可以发现公开数据集中看似匿名或随机采样的信息链接到特定的个人）；而且，公开数据集不可能覆盖所有应用领域，无法满足所有研究问题的需求，比如特定领域文本或限定条件下的标注或分析等，因此在很多分支领域的应用时仍然需要特定的私有数据集。

语料库（corpus，复数形式 corpora）是自然语言处理和语料库语言学（corpus Linguistics）领域的数据集，指在为特定的应用目标而对真实出现过的语言材料专门收集加工、可被计算机程序分析的文本或文档集合，数据结构类型一般都是非结构化数据或半结构化数据。在自然语言处理实践中，大多数情况下，语料库的选择决定了任务的成败。常用的情绪分类评测基准语料库有：英文情绪语料库有斯坦福情绪树库-情绪二分类（Stanford Sentiment Treebank-binary sentiment classification，SST-2）、康奈尔大学庞博提供的影评语料库（movie review data）、互联网电影资料库电影评论语料库（Internet Movie DataBase，IMDB）、国际语义评测大会-2014（international workshop on Semantic Evaluation-2014，SemEval-2014）任务 4 方面级情绪分析笔记本电脑和餐厅特定领域、子任务 2 方面词语级情绪极性分类、子任务 4 方面类别级情绪极性分类语料库、任务 9 推特情绪分析语料库、Yelp 美国商户点评网站餐厅评论语料库、国际语义评测大会-2016 任务 5 方面级情绪分析笔记本电脑、餐厅和酒店特定领域评论语料库、城市社区问答平台方面级对象抽取语料库 SentiHood、亚马逊笔记本电脑评论语料库、数百万条亚马逊客户评论（输入文本）和星级评级（输出标签）数据集、包含正负向、中性、显/隐式表达、讽刺等多视角情绪语料库、金融短语库（Financial PhraseBank）、分析报告文本（analyst report text）、金融意见挖掘和问答 2018 任务 1 方面级金融情绪分析（FiQA 2018 Task 1）数据集，等等；中文情绪语料库有第一至第八届中文倾向性分析评测的语料库（the first to eighth Chinese Opinion Analysis Evaluation，COAE2008—

2016）、携程网酒店评论语料库（ChnSentiCorp_htl_all）、第三届中国计算机学会自然语言处理与中文计算会议评估任务 1：情感分析中文微博文本（evatask1）、评估任务 2：深度学习的情绪分类（evatask2）、微博情绪分类技术评测通用微博数据集和疫情微博数据集、金融领域实体级细粒度情感分析语料库，等等。然而，目前未能找到公开的已标注中文金融文本情绪分类语料库，因此本书使用私有数据集来评测模型。

本书遵照《GB/T 36344—2018 信息技术 数据质量评价指标》——“规范性、完整性、准确性、一致性、时效性、可访问性”，选择自己创建标注小型句子级情绪分类语料库，既用于模型微调，也用于模型评测（即公开性能评估数据），标注情绪语料库创建过程如下。

第一，采集数据。利用网页超文本标记语言（Hyper Text Markup Language，HTML）源代码中层叠样式表（Cascading Style Sheets，CSS）定义的网页元素信息（如 class、id 等），使用 Python 网页爬虫框架 requests 软件库/工具包提取指定网站的指定原始内容，提取的内容即为样本。本书提取的是互联网上的中文金融或财经网站真实金融新闻（主要是上市公司股票新闻）的主标题，这些网站包括金融界（http://www.jrj.com.cn）、中国经济网（http://www.ce.cn）、证券时报网（http://www.stcn.com）、和讯网（http://www.hexun.com）、中财网（http://www.cfi.cn）等，发布日期为 2018—2020 年，整理去重并保存为每行一个句子、无空行（one-sentence-per-line without empty lines）的一个文本文件。

第二，标注数据。选取部分采集数据，使用人工标注方法，手动将每条新闻主标题归入“利多”（看涨）、“利空”（看跌）和“其他”（持平）三种不同类别的情绪，在每个标题前标注类别并用特殊符号“|||”分隔。

第三，保留少量操作误差、主观误差或脏数据（不完整、含噪声、不一致的数据）。真实的标注数据集往往是不完美的，存在人为造成的误解、误读、误算或者没有意义的数据。本语料库中包含非金融领域的新闻主标题、与金融知识不相关的词和符号（例如，[快讯]、【图解】、(2019)）、空格或未标注的语料。

该评测标注语料库（表 3.1）必须是目标域小样本（few-shot）或处于少资源（low resource）情景，符合金融领域高质量的标注数据十分稀缺的真实情景，这也是当前自然语言处理研究重点关注的领域。同时，避免样本单一，得出结论不牢固。显然，该语料库来自互联网公开数据，不涉及国家秘密、商业秘密、个人隐私和他人知识产权或其他合法权利问题。

表 3.1 标注语料库概览

语料源	语料库格式	语种	领域	任务	是否标注	用途
股票新闻主标题	txt	中文	金融	情绪分类	已标注	模型评测

该标注语料库的示例说明见表 3.2，更多语料库样本示例详见附录 A.1，描述性统计分析见 3.7 节。

表 3.2　标注语料库示例说明

语料库示例	说　明
利多|||天利中期多赚逾 7 倍 现升逾 20%	正向情绪句
利空|||粤传媒：公司收到行政处罚告知书	负向情绪句
其他|||欧陆通：选举李德华为职工代表监事	中性情绪句

3.6　数据集划分

在传统机器学习模型中，通常将原始数据集划分为三个部分：训练集（training set），用于训练模型（计算梯度、更新权重）；验证集（validation set），用于选择模型；测试集（testing set），用于评估模型（判断网络性能优劣）；且划分要尽可能保持数据分布的一致性。其中，验证集的作用是防止过拟合，在训练过程中，用验证集来确定一些超参数。

在预训练语言模型中，标注语料库用于微调，由于标注语料库样本量较小（通常小于 10 000 个），属于小数据，因此采用留出法（holdout method）将数据集划分为训练集和测试集，按文本行数取前 75%行数的语料作为训练集，剩余 25%行数的语料即为测试集，并将两个数据集的数据分布特征调整相似。

3.7　描述统计分析

3.7.1　语料库统计量描述

对标注语料库进行常用统计度量的计算和可视化：各类别标签的样本量和所占百分比、文本长度句数分布、词频排序、词云等。

按照 3.5 节和 3.6 节的方法进行操作，得到本书的标注语料库。标注语料库的样本容量为 2346 条，其中，596 条“利多”类别样本、959 条“利空”类别样本、789 条“其他”类别样本，以及特意保留“未标注”类别样本量 2 条。图 3.4 是 4 个类别样本量的条形图。

标注语料库有 4 个类别，其中，“利多”类别样本所占百分比（以下简称“占比”）为 25.4%、“利空”类别样本占比 40.9%、“其他”类别样本占比 33.6%，以及特意保留的未标注样本占比 0.1%。图 3.5 是 4 个类别样本的条形图。

标注语料库中包含若干个单句，除去“利多”“利空”“其他”三个类别和分隔符“|||”，每个单句以换行符分隔（结尾），先计算出每个单句包含的字数，用横坐标表示字数、纵坐标表示相同字数的句子数量（句数），可以绘制出语料库不同句长度的句数条形（分布）图，如图 3.6 所示。其中，最长句 79 字，最短句 5 字，23 字数的句子最多、有 163 个句子，接近卡方分布（χ^2）。

词数（term count）是指一个词在一个文档中的出现次数，在两个及两个以上文档

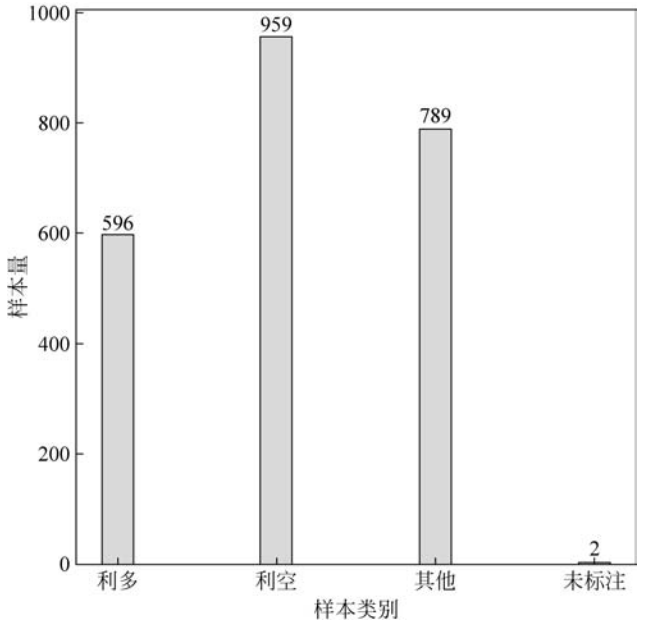

图 3.4　标注语料库各类别样本量条形图

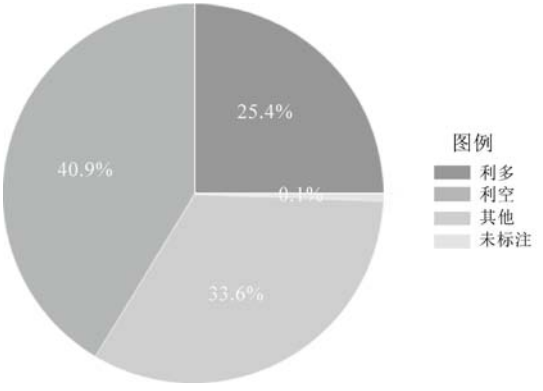

图 3.5　标注语料库各类别所占百分比饼图

比较时，这个度量指标可能会偏向文本长度长的文档。词频（term frequency）是指一个词在一个文档中出现的频率，词频是词数的归一化，避免了一个词虽然在长文档可能会比短文档有更高的词数，却无法准确度量该词重要性的情况。除去高频出现但是却没有真正的实际意义的词汇停用词（例如，中文的“一些”“比如”“对于”，英文的 the、is、are、to，以及标点符号），一个词的词频值随着它在一个文档中出现词数的增加而增加，在文档中出现频率高的词比出现频率低的词更有代表性，出现频率越高的词对文档的影响越大。词频的数学表达式如下。

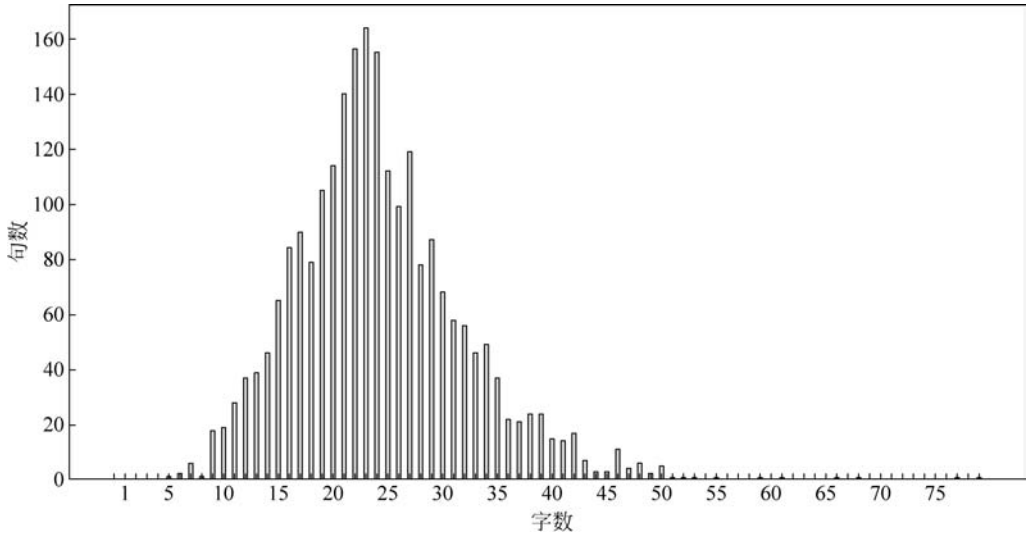

图 3.6　标注语料库不同句长度的句数条形图

$$\mathrm{TF}=\frac{w_{ij}}{\sum_{i=1,j=1}^{n,m} w_{ij}}$$

其中，n 表示一个文档（被中文分词后）包含的词数量，m 表示文档数量，i 表示词编号、取值范围为$[1,n]$，j 表示文档编号、取值范围为$[1,m]$，w_{ij} 表示 i 词在 j 文档中出现的次数，分母表示 m 个文档中所有词出现的次数之和。

将标注语料库中所有词按词频高低降序排列，使用 Python 中文分词工具包结巴中文分词（jieba 0.42.1，2020 年 1 月 20 日发布），输出前 15 位的词，绘制出标注语料库高频词前 15 位排序条形图，如图 3.7 和表 3.3 所示。

表 3.3　标注语料库的高频词前 15 位由高到低排序

排名	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
词语	股份	公司	公告	股东	控股	业绩	质押	科技	中国	集团	SZ	子公司	ST	增长	2018
词频	325	202	185	184	132	121	93	91	80	79	79	77	75	71	69

词云（word cloud）根据文本中关键词的出现频率显示不同的大小和颜色（或灰阶深度），频率越高字体越大颜色越醒目、予以视觉上的突出，形成“关键词云层”或“关键词渲染”，从而过滤掉大量的文本信息，只要一眼扫过词云就可以领略文本的主旨。除去“利多”“利空”“其他”三个类别和分隔符“|||”，标注语料库的词云如图 3.8 所示。

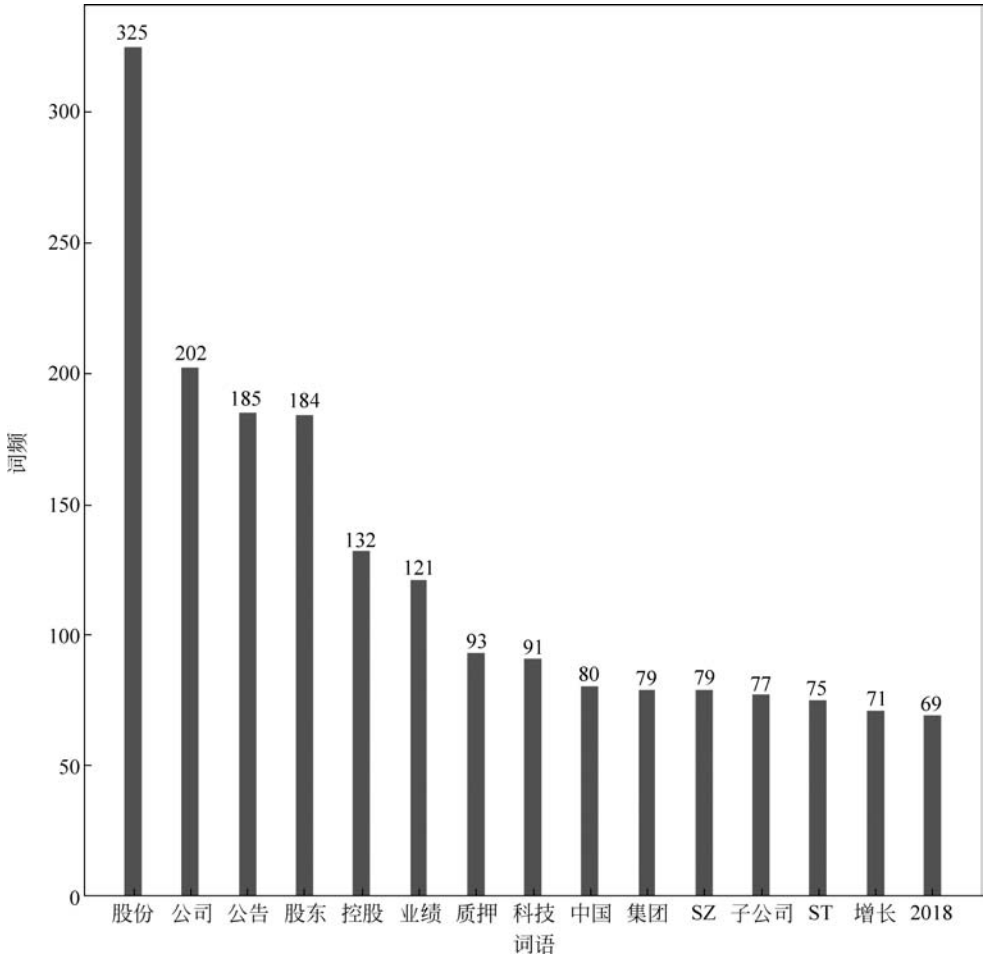

图 3.7　标注语料库高频词前 15 位由高到低排序条形图

图 3.8　标注语料库词云

3.7.2　训练集和测试集统计量描述

按照 3.6 节阐述的数据集划分方法，将标注语料库划分为训练集和测试集，使用与语料库统计量描述同样的方法对训练集和测试集进行统计量描述。

绘制训练集和测试集各类别样本量的对比条形图，如图 3.9 所示。

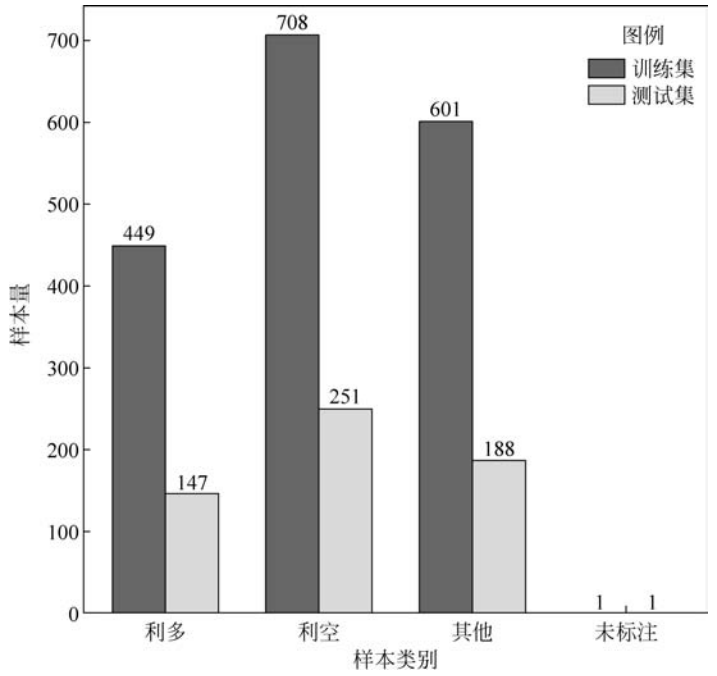

图 3.9　训练集和测试集各类别样本量的对比条形图

分别绘制训练集和测试集各类别所占百分比饼图对比，如图 3.10 所示。

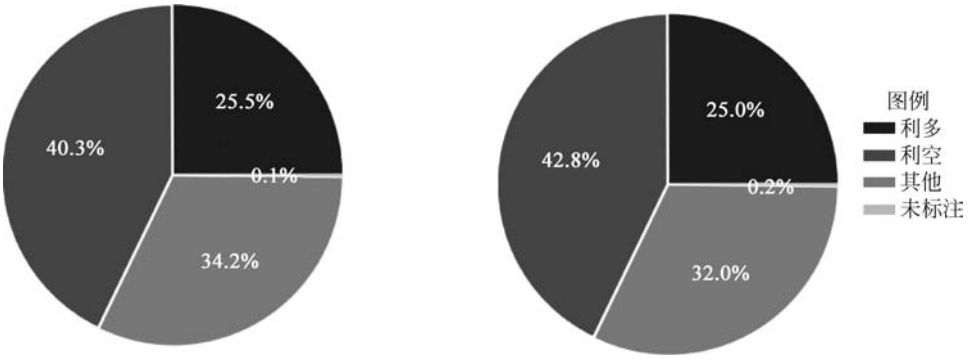

图 3.10　训练集(左图)和测试集(右图)各类别所占百分比饼图对比

绘制训练集和测试集各类别所占百分比的对比条形图，如图 3.11 所示。

将标注语料库及其划分的训练集和测试集的各类别样本量和占比统计数据汇总，

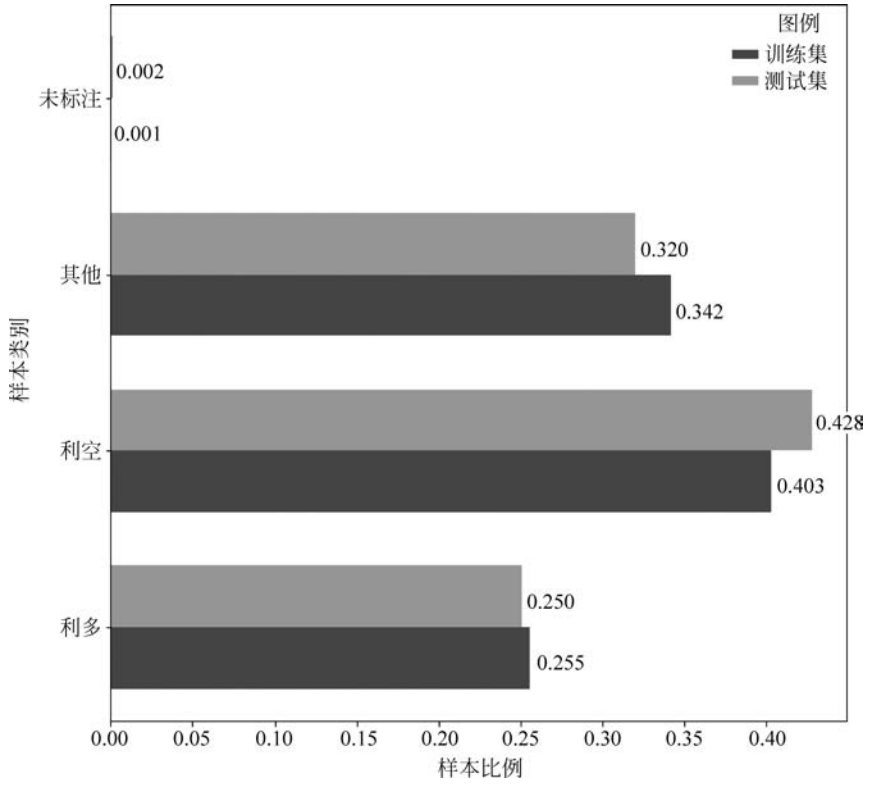

图 3.11　训练集和测试集各类别所占百分比的对比条形图

纳入表 3.4。

表 3.4　语料库及其划分的训练集和测试的样本容量、比例、各类别样本量和占比

统计量 / 数据集	样本容量	利多		利空		其他		未标注	
		样本量	占比	样本量	占比	样本量	占比	样本量	占比
语料库	2346	596	25.4%	959	40.9%	789	33.6%	2	0.1%
训练集	1759	449	25.5%	708	40.3%	601	34.2%	1	0.1%
测试集	587	147	25.0%	251	42.8%	188	32.0%	1	0.2%
两数据集比例	7.5∶2.5	7.5∶2.5		7.4∶2.6		7.6∶2.4		5∶5	

分别绘制训练集和测试集不同句长度句数条形图对比，如图 3.12 所示。

训练集和测试集中所有词按词频由高到低排序，分别绘制出训练集和测试集高频词前 15 位排序条形图进行对比，如表 3.5、表 3.6 和图 3.13 所示。

表 3.5　训练集的高频词前 15 位由高到低排序

排名	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
词语	股份	公司	股东	公告	控股	业绩	科技	集团	质押	子公司	中国	SZ	ST	2018	减持
词频	246	146	135	128	97	94	65	62	61	58	57	57	56	53	52

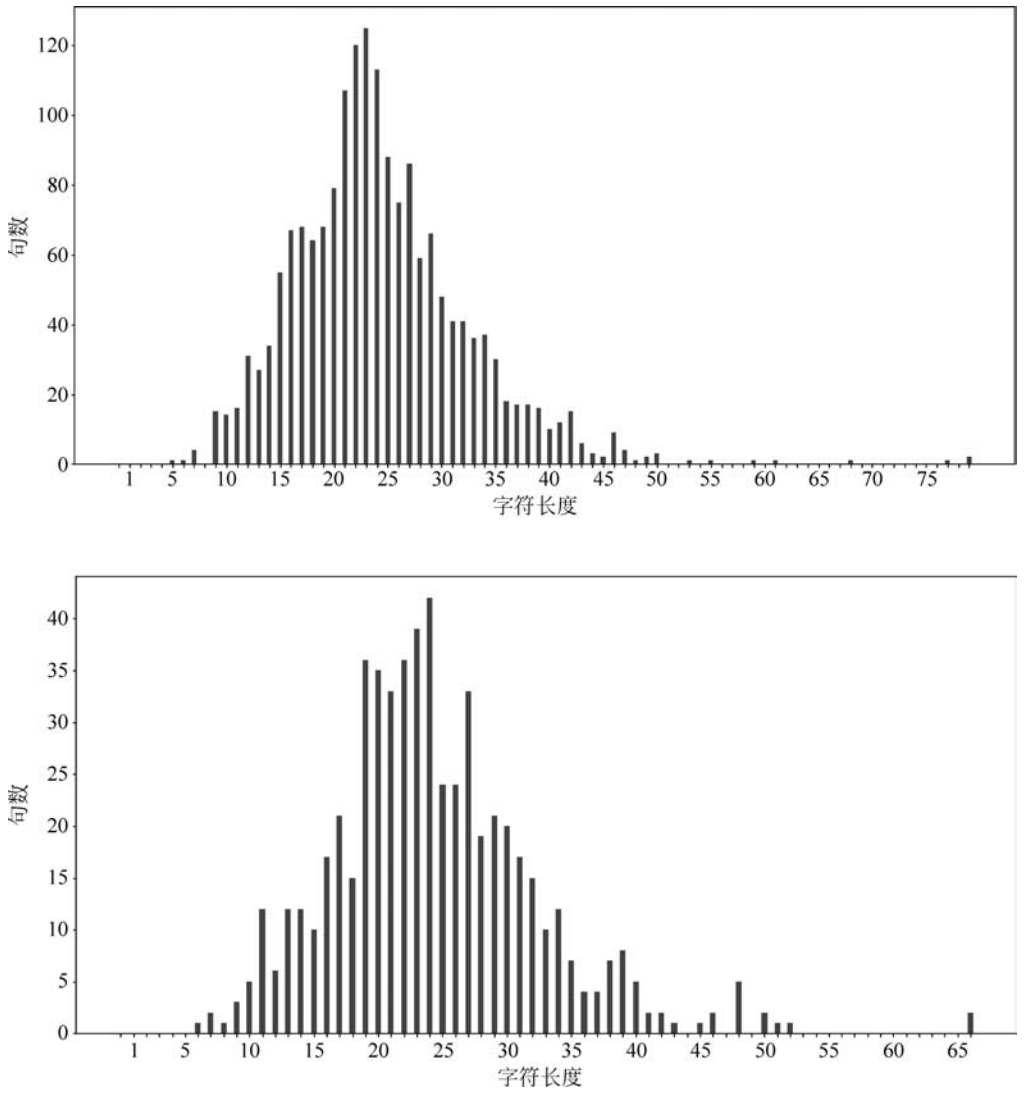

图 3.12　训练集(上图)和测试集(下图)不同句长度的句数条形图对比

表 3.6　测试集的高频词前 15 位由高到低排序

排名	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
词语	股份	公告	公司	股东	控股	质押	业绩	科技	中国	SZ	增长	净利	子公司	ST	资金
词频	79	57	56	49	35	32	27	26	23	22	20	19	19	19	18

分别绘制出训练集和测试集的词云进行对比，如图 3.14 所示。

将标注语料库及其划分的训练集和测试集的高频词前 10 位由高到低排序统计数据汇总，纳入表 3.7。

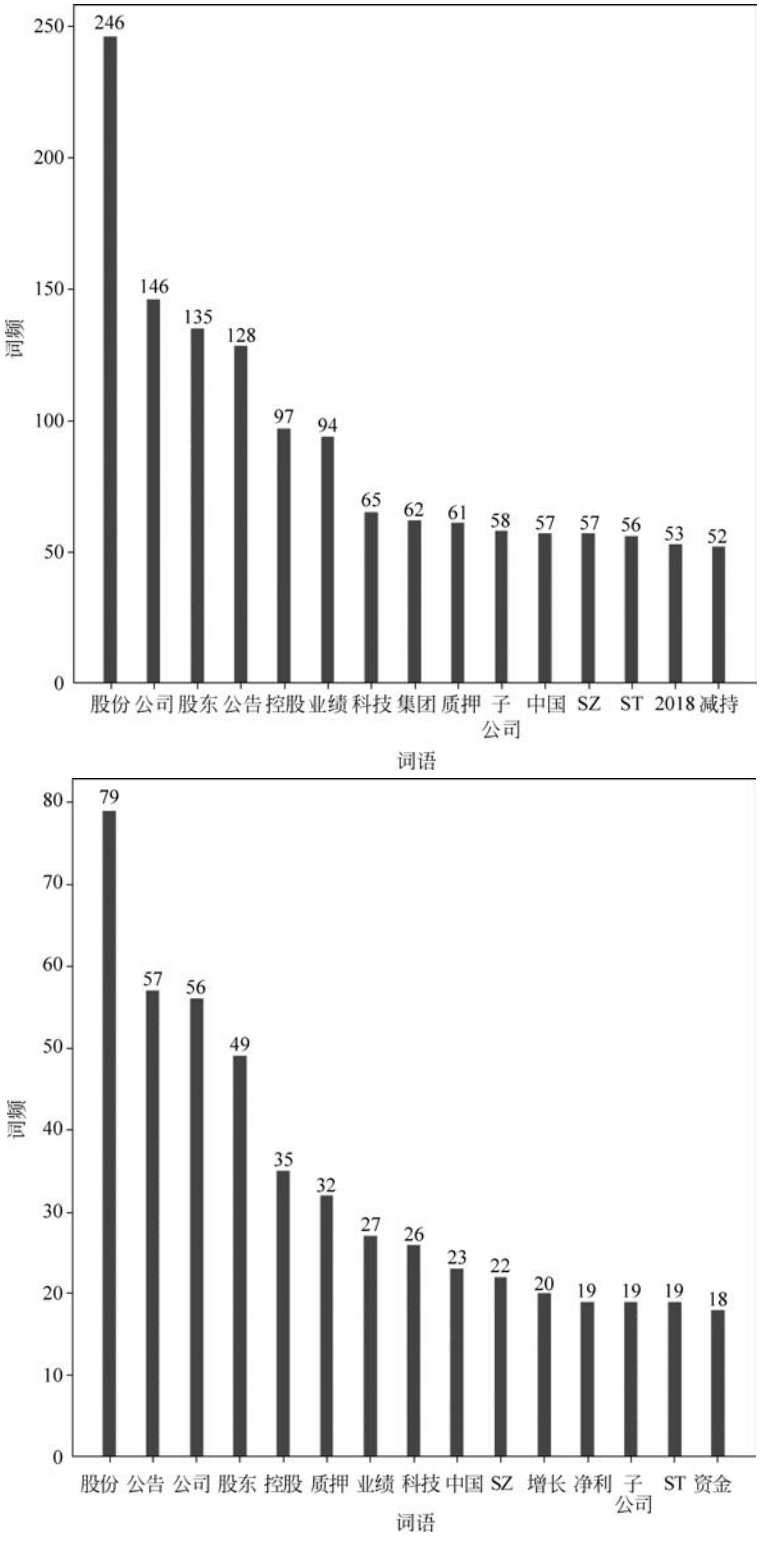

图 3.13　训练集(左图)和测试集(右图)高频词前 15 位排序条形图对比

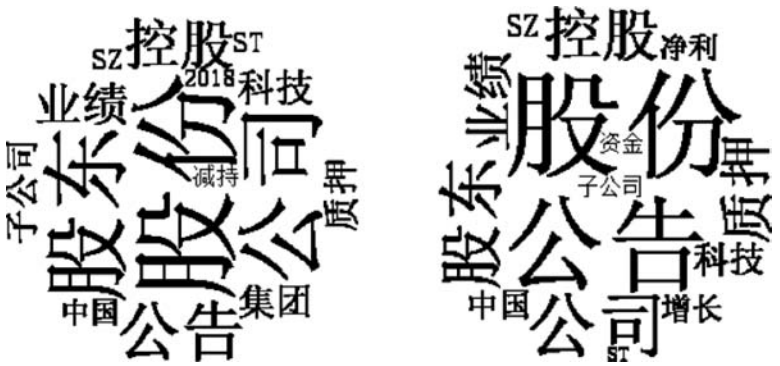

图 3.14　训练集(左图)和测试集(右图)词云对比

表 3.7　语料库及其划分的训练集和测试的高频词前 10 位由高到低排序

排名	1	2	3	4	5	6	7	8	9	10
语料库	股份	公司	公告	股东	控股	业绩	质押	科技	中国	集团
训练集	股份	公司	股东	公告	控股	业绩	科技	集团	质押	子公司
测试集	股份	公告	公司	股东	控股	质押	业绩	科技	中国	SZ

3.7.3　统计分析

通过对标注语料库(即原始数据集)及其划分后的训练集和测试集的统计量计算，得到语料库(数据集)的统计信息，提取到了语料库的一些特征，可以从以下三个方面进行统计分析。

第一，类别分析。从表 3.4 可以看出，各类别样本占比均衡，说明不存在数据的类别不平衡(class imbalance)或类别倾斜(class skew)问题；训练集与测试集两数据集各类别的比例与样本容量的比较接近，说明训练集与测试集的类别样本划分比例合适。

第二，文本长度分析。从图 3.6 和图 3.12 可以看出，各数据集文本长度分布相近但不相同。

第三，词频分析。从表 3.7 或者图 3.8 和图 3.14 的比较中可以看出，各数据集词频基本一致，且都包含金融文本的关键词，但训练集不能完全覆盖测试集。

最终结论，可以认为原始数据集、训练集与测试集的特征空间相同且分布近似。

3.8　对比模型

本章项目选择当前主流的、代码开源的 28 个已预训练中文预训练模型(截至 2021 年 10 月)进行评测，对比模型相关信息说明如表 3.8 所示。对比模型覆盖了原模型、调优模型和压缩模型，模型规模覆盖基础、中号、大号(base，mid，large)，领域包括通用和特定，骨干网络架构包括 BoW＋bigram＋trigram、BiLSTM、Transformer 和高效变换器架构——Longformer(“X-former”的一种)，语言表征的上下文关联程度包括上下文

无感表征和上下文感知表征。

表 3.8 对比模型说明

模型名称	论文发表年份	代码说明
Chinese fastText	2016	来自 Facebook Research
simplified-Chinese ELMo	2018	来自 HIT-SCIR，使用 Chinese gigawords-v5 新华部分，而繁体中文 ELMo 使用中文维基百科
Chinese ULMFiT	2019	来自 bigboNed3（GitHub 账号），使用 2018 年中文维基百科语料（wiki2018-11-14）
$BERT_{BASE}$ Chinese	2018/2019	谷歌官方发布 BERT-Base，Chinese
$BERT_{LARGE}$ Chinese		谷歌官方发布 BERT-Large + vocab
$BERT_{BASE}$-BiGRU-Attention	2020	云南大学信息学院 2019 级硕士研究生袁理在语义评测国际研讨会（SemEval-2020）任务 8 网络表情包情感分析的子任务 A 情绪分类代码
$BERT_{BASE}$-CNN-BiLSTM-Attention	2019	CNN-BiLSTM-Attention 代码来源 PatientEz（GitHub 账号）
$BERT_{BASE}$-BiLSTM-CNN-Attention	2020	BiLSTM-CNN 参考代码 usualwitch（GitHub 账号），引用论文中使用的是 $BERT_{BASE}$-LSTM-CNN 模型
$RoBERTa_{LARGE}$ Chinese	2019	来自中文语言理解测评基准组织
$ALBERT_{BASE}$ Chinese	2019/2020	谷歌官方发布 ALBERT-base
$BERT_{BASE}$-wwm-ext	2019	哈尔滨工业大学社会计算与信息检索研究中心 2018 级博士研究生崔一鸣发布
$RoBERTa_{LARGE}$-wwm-ext		
$MacBERT_{BASE}$	2020	
$ELECTRA_{LARGE}$ Chinese	2020	
$XLNet_{MID}$ Chinese	2019/2020	崔一鸣基于卡内基·梅隆大学语言技术研究所 2020 届博士毕业生戴子行代码发布
$Longformer_{BASE}$ Chinese	2020	来自 Hugging Face
ERNIE1.0 Base	2019	百度 ERNIE
ERNIE Tiny	2019/2020	基于百度 ERNIE 2.0，但 ERNIE 2.0 尚未开源，仅在百度 BML 平台使用
SKEP	2020	百度开源情绪分析系统 Senta
NEZHA-base	2019	来自华为诺亚方舟实验室
ZEN	2019	ZEN_pretrain_base
WoBERT	无	在哈尔滨工业大学开源的 $RoBERTa_{BASE}$-wwm-ext 基础上继续预训练，初始化阶段将每个词用 BERT 自带的分词器切分为字，用字嵌入的平均作为词嵌入的初始化 WoBERT、$WoBERT^{+}$ 和 WoNEZHA 代码来自追一科技
$WoBERT^{+}$（WoBERT Plus）		
WoNEZHA	无	在华为开源 NEZHA-base-WWM 基础上继续预训练
RoFormer	2021	来自追一科技

续表

模型名称	论文发表年份	代　　码
FinBERT 1.0-Base	无	在哈尔滨工业大学开源的 BERT-wwm 基础上采用三大类金融领域语料继续预训练，FinBERT-Large 尚未开源，代码来自熵简科技
Mengzi-$BERT_{BASE}$	2021	300GB 互联网语料训练
Mengzi-$BERT_{BASE}$-fin	2021	基于 Mengzi-$BERT_{BASE}$ 在 20GB 金融新闻、公告、研报等语料继续训练

开源代码仓库中存在不同用户上传的各种不同版本，为了避免模型代码版本不同导致评测结果的差异而对模型对比分析结论产生异议，表 3.8 对本章使用的开源代码进行了说明，它们大多数来自开源社区 GitHub 的源代码仓库，分别下载自脸书研究(Facebook research)、哈尔滨工业大学社会计算与信息检索研究中心(Harbin Institute of Technology-research center for Social Computing and Information Retrieval，HIT-SCIR)、bigboNed3、谷歌人工智能(Google AI，2018 年 5 月 18 日由 Google Research 和 Google.ai 合并而成)、中文语言理解测评基准(CLUE)组织(即中文通用语言理解测评基准(Chinese general Language Understanding Evaluation benchmark，Chinese GLUE benchmark)组织)、哈尔滨工业大学社会计算与信息检索研究中心 2018 级博士研究生崔一鸣、百度飞桨、百度、华为诺亚方舟实验室(Huawei Noah's ark lab)、创新工场人工智能工程院(Sinovation ventures AI institute)、深圳追一科技有限公司、北京熵简科技有限公司、陈元伟、PatientEz 等官方 GitHub 账号上传并公开发布的源代码；间接使用了卡内基·梅隆大学计算机科学学院语言技术研究所 2020 届语言和信息技术博士戴子行(北京循环智能科技有限公司联合创始人杨植麟团队)、北京澜舟科技有限公司官方 GitHub 账号上传并公开发布的源代码。只有中文 Longformer 模型直接下载自抱抱脸(hugging face)官方网站。

所有代码的使用均遵守官方 GitHub 账号指定采用的开源软件授权许可协议，本章对比模型的开发者大多数都选择非盈利开源组织阿帕奇软件基金会制定的授权许可协议 2.0 版本(Apache license 2.0)，可以免费使用、修改、按照自己的方式进行集成，并应该清楚地在产品、网站和市场介绍材料中明确指出使用了开发者的源代码。按照开源软件授权协议规定，本书在此处进行声明。

有些情绪分类特定任务模型虽然已开源，但是未使用中文进行预训练，因此未能纳入到本章模型对比，例如，清华大学人工智能学院交互式人工智能(Conversational AI，CoAI)课题组发布的 SentiLARE 模型。

3.9　模型实现

对于模型的使用者来说，只需要调用已预训练的模型变量和权重值，而不必直接使用未预训练的开源预训练模型源代码从头训练。例如，谷歌官方 GitHub 源代码仓库

既提供了模型源代码，也提供了已预训练的模型变量和权重值。谷歌官方 GitHub 源代码仓库 $BERT_{BASE}$ Chinese 模型源代码有 21 个文件，与模型相关的主要包括 tokenization. py、modeling. py、run_pretraining. py、run_classifier. py 等 Python 代码文件，详见表 3.9；而已预训练的模型变量和权重值则以. zip 格式压缩包形式提供若干检查点（checkpoint）文件下载，详见 3.11 节。本章的模型是基于开源模型检查点文件和开源软件框架来实现的，使用编程语言 Python 版本大于或等于 3.6 或 3.7。

表 3.9 BERT 模型源代码文件

文件名. 后缀名（文件格式）	说　明
__init__. py	模型初始化说明 Python 源代码
tokenization. py	分词类 Python 源代码
tokenization_test. py	分词测试类 Python 源代码
create_pretraining_data. py	创建预训练数据 Python 源代码
extract_features. py	句子转为词向量 Python 源代码
modeling. py	模型训练 Python 源代码
modeling_test. py	模型测试 Python 源代码
optimization. py	模型优化 Python 源代码
optimization_test. py	优化测试类 Python 源代码
run_pretraining. py	运行掩码等建模目标 Python 源代码
run_classifier. py	运行预测的填充示例 Python 源代码
run_classifier_with_tfhub. py	运行分类任务 Python 源代码
predicting_movie_reviews_with_bert_on_tf_hub. ipynb	运行预测电影评论 Python 源代码
run_squad. py	运行自动问答任务 Python 源代码
sample_text. txt	确保 Unicode 被正确处理的测试文本
requirements. txt	运行环境要求 TensorFlow≥1.11.0
multilingual. md	Markdown 格式多语种模型说明
README. md	Markdown 格式自述文件
LICENSE	软件开源授权许可协议
CONTRIBUTING. md	开源贡献管理办法
. gitignore	指定要忽略的故意未跟踪的文件

依托预训练模型框架不需要从头重新构建整个网络，可轻松调用检查点文件，实现多个不同的预训练模型，为实现现有的神经网络模型提供了更快速的方法。本章对比模型直接使用了 4 种不同的框架：PyTorch、Transformers、PaddleHub 和 bert4keras，每种框架分别对应了各自相同名称的 Python 编程语言的软件库，模型通过调用 Python 软件库来直接实现，这样就不必重复编写功能代码。本章模型实现过程中，不同的模型使用了不同的框架，部分程序代码是根据 3.8 节中的开源代码修改的。

脸书人工智能研究院推出的 PyTorch 是一个针对深度学习的张量库（tensor library），提供张量计算和深度神经网络自动求导（autograd）两个高级特性的 Python

包。本章项目使用的 PyTorch 版本为 v1.7.1(stable release)(2020 年 12 月 11 日发布)。项目中使用 PyTorch 库实现的模型如表 3.10 所示。

表 3.10　使用 PyTorch 库实现的模型

模型名称
Chinese fastText
simplified-Chinese ELMo
Chinese ULMFiT

抱抱脸(Hugging Face)公司拥有一个相同名称的人工智能开源社区,可以构建、训练和部署由自然语言处理中参考开源支持的先进模型,超过 2000 家机构在使用抱抱脸社区的产品,包括谷歌、微软、脸书,等。Transformers 库(曾用名 pytorch-transformer 和 pytorch-pretraind-bert)是抱抱脸的自然语言处理库,为自然语言理解和自然语言生成提供通用架构,支持超过 100 个语种的 38 个以上的预训练模型,如 BERT、GPT-2、RoBERTa、XLM、DistilBERT、XLNet 等,并具有 TensorFlow 2.0 和 PyTorch 之间的深度互操作能力。表 3.11 列出了本章项目的每个模型对应在 Transformers 库中的名称,在抱抱脸官网的模型库("Models" or "Model Hub")中可以搜索得到。

表 3.11　使用 Transformers 库实现的模型

模型名称	Transformers 库中名称
$BERT_{BASE}$ Chinese	bert-base-chinese
$RoBERTa_{LARGE}$ Chinese	clue/roberta_chinese_large
$ALBERT_{BASE}$ Chinese	voidful/albert_chinese_base
$BERT_{BASE}$-wwm-ext	hfl/chinese-bert-wwm-ext
$RoBERTa_{LARGE}$-wwm-ext	hfl/chinese-roberta-wwm-ext-large
$MacBERT_{BASE}$	hfl/chinese-macbert-base
$ELECTRA_{LARGE}$ Chinese	hfl/chinese-electra-large-discriminator
$XLNet_{MID}$ Chinese	hfl/chinese-xlnet-mid
$Longformer_{BASE}$ Chinese	schen/longformer-chinese-base-4096

百度飞桨(PaddlePaddle)是集深度学习核心框架、工具组件和服务平台为一体的技术先进、功能完备的开源深度学习平台。PaddleHub 是飞桨生态的预训练模型应用工具,开发者可以便捷地使用高质量的预训练模型结合微调应用程序编程接口(Application Programming Interface,API),快速完成模型迁移到部署的全流程工作。PaddleHub 提供的 200 个以上预训练模型涵盖了图像分类、目标检测、词法分析、语义模型、情感分析、视频分类、图像生成、图像分割、文本审核、关键点检测等主流模型,带来易于推理和服务部署的便利。本章使用的 PaddleHub 版本为 2.0.0rc0(2020 年 12 月 1 日发布)。表 3.12 为使用 PaddleHub 库实现的模型。

表 3.12 使用 PaddleHub 库实现的模型

模型名称	PaddleHub 库中名称
ERNIE 1.0 Base	ernie-1.0
ERNIE Tiny	ernie_tiny
SKEP	ernie_1.0_skep_large_ch

bert4keras 是一个基于谷歌 Keras 预训练模型加载框架，目前支持多种预训练模型（BERT、ALBERT、RoBERTa、NEZHA 等），并支持多种环境（Python 2.7、Python 3.x）和后端（keras 2.2.4/2.3.0/2.3.1、tf.keras、TensorFlow 1.14+、TensorFlow 2.x），谷歌端到端的开源机器学习平台 TensorFlow 2.x 已经深入融合 Keras 作为其高阶 API，通过 tf.keras 接口模块实现使用。本章使用的 bert4keras 版本为 0.9.3（2020 年 11 月 20 日发布）。表 3.13 为使用 bert4keras 库实现的模型。

表 3.13 使用 bert4keras 库实现的模型

模型名称
$BERT_{LARGE}$ Chinese
$BERT_{BASE}$-CNN-BiLSTM-Attention
BERT-BiGRU-Attention
NEZHA-base
ZEN
WoBERT
WoNEZHA
RoFormer
FinBERT 1.0-Base

3.10 运行环境

本章项目是分别在两个深度学习框架云平台环境上进行的——谷歌 Colab 和百度 AI Studio 平台。使用 Transformers 库和 bert4keras 库实现的模型在谷歌 Colab 平台上运行，使用 PaddleHub 库实现的模型在百度 AI Studio 平台上运行。

Colab 和 AI Studio 都是在 Jupyter 基础之上开发的，通过 Colab 和 AI Studio，无须在本地计算机上下载、安装或运行任何软件（例如 Python 解析器、TensorFlow 2.x 等），就可以通过浏览器在线使用 Jupyter Notebook。使用云平台的最大优点在于因不同硬件配置造成的计算结果差异相距较小，公平恰当、容易复现、轻松共享，不偏袒任何一个模型。Jupyter Notebook 是开源的基于网页的交互式计算的应用程序，允许用户创建和共享各种内容，包括实时代码开发、方程式、文档编写、运行代码、展示结果、可视化和叙述文本的文档等计算全过程，支持 Python 编程范式以 Jupyter Notebook 格式（.ipynb）存储。

谷歌 Colaboratory 简称“Colab”，是 Google Research 团队开发的一款产品，网址为 https://colab.research.google.com。在 Colab 中，任何人都可以通过浏览器编写和执行任意 Python 代码。它尤其适合机器学习、数据分析和教育目的。从技术上说，Colab 是一种托管式 Jupyter Notebook 服务，程序和数据存储在 Google 云端硬盘上（默认免费提供 15GB 存储空间，网址为 https://drive.google.com），也可以从 GitHub 加载，用户无须进行任何设置，就可以直接使用，同时还能获得图形处理器（Graphics Processing Unit，GPU）或张量处理器（Tensor Processing Unit，TPU）计算资源的免费使用权限，代码会在分配给用户账号使用的虚拟机中执行。Colab 使用量限额和硬件供应情况采用时有变化的动态限额，并且不会保证资源供应或无限供应资源，可用资源会不时变化，以适应需求的波动性，以及总体需求的增长和其他因素。如果希望获得更高、更稳定的使用量限额，可以订阅 Colab Pro，但仅限美国和加拿大用户使用。Colab 预安装了三百八十多个 Python 软件包，其中默认的深度学习框架开源库版本为 TensorFlow 2.3.0，本章的程序运行中还使用了 TensorFlow 2.2.0、Torch 1.7.1，未使用 TorchServe 云端开源模型服务框架。

百度 AI Studio 是基于百度深度学习平台飞桨的人工智能学习与实训社区，网址为 https://aistudio.baidu.com，提供在线 Jupyter Notebook 编程环境、免费 GPU 算力、海量开源算法和开放数据，帮助开发者快速创建和部署模型。ERNIE（BAIDU）和 SKEP 模型依赖于百度飞桨（PaddlePaddle）深度学习库，因此这两个模型的运行环境以百度平台更方便（用其他平台也可以），本书使用的是百度 AI Studio 基础版（无 GPU、免费使用），而含 GPU 的高级版收取 1 算力卡/小时，宣传推广阶段每日运行即获赠 10 小时 GPU 免费使用时长。AI Studio 预安装了一百六十多个 Python 软件包，其中默认的深度学习框架开源库版本为 PaddlePaddle 2.0.0rc，本章的程序运行中还使用了 1.7.2 或 1.8.0。

如果租赁云 GPU 或本地 GPU 运行建议最低硬件配置为 NVIDIA Tesla V100 16GB，硬件配置低会影响模型训练和评测时间，模型训练轮次过少会影响最终评测结果，每次训练后的评测结果可能略有差异。

3.11　模型加载

本章项目的对比模型加载有两种情况：一种是直接加载已预训练检查点文件，另一种是从软件框架中加载已预训练检查点文件。

谷歌官方基础 BERT 中文模型（$\text{BERT}_{\text{BASE}}$ Chinese）已预训练检查点文件中包含谷歌深度学习框架 TensorFlow 框架保存模型后自动生成 ckpt.meta、ckpt.data、ckpt.index 检查点（checkpoint，ckpt）文件（例如设置每 5s 保存一次检查点，以便训练中断时将训练得到的参数保存下来，不必重新训练，如表 3.14 所示）。.meta 文件是 MetaGraphDef 序列化的二进制文件，保存了网络结构相关的数据，包括 graph_def 和 saver_def 等。.data 保存检查点文件列表，可以用来迅速查找最近一次的检查点文件。

.index 文件存储的核心内容是以 tensor name 为键以 BundleEntry 为值的表格 entries，BundleEntry 主要内容是权值的类型、形状、偏移、校验和等信息。

表 3.14 BERT 中文模型已预训练检查点文件说明

文件名.后缀名(文件格式)	说　明
bert_model.ckpt.meta	模型元图，即计算图的结构
bert_model.ckpt.data-00000-of-00001	已预训练的模型变量名和权重值
bert_model.ckpt.index	检查点文件映射的索引
bert_config.json	模型超参数的配置文件
vocab.txt	中文词汇表，用于词块映射为词编号

抱抱脸 Transformers 库的基础 BERT 中文模型(bert-base-chinese)将谷歌深度学习框架 TensorFlow 框架保存模型的检查点文件转换为脸书深度学习框架 PyTorch 下编译的文件，如表 3.15 所示。

表 3.15 BERT 模型 Transformers 库编译文件说明

文件名.后缀名(文件格式)	说　明
pytorch_model.bin	PyTorch 下的可执行文件
tf_model.h5	模型文件校验码
config.json	模型超参数的配置文件
tokenizer.json	分词后的词编号
tokenizer_config.json	分词模型超参数的配置文件
vocab.txt	中文词汇表，用于词块映射为词编号
README.md	Markdown 格式自述文档
.gitattributes	以行为单位设置一个路径下所有文件的属性

每个软件框架的模型加载 Python 编程具体方法详见官方说明文档。

3.12 微调策略

3.12.1 情绪分类任务微调

微调分为针对下游任务分别(去掉最后一层分类器)仅对语言模型的特定任务微调和仅对最后的分类层(最后一层分类器)的微调。

本章项目对于不同模型执行相同的情绪分类任务，需要针对语言模型的不同网络架构特点进行特定任务微调，将情绪分类任务的输入和输出插入模型中，并对所有参数进行端到端的微调。以 BERT 模型为例，BERT 利用变换器的自我注意机制将输入和输出两个阶段统一起来，通过交换适当的输入和输出，对句子级情绪分类任务进行建模；在每个单句的第一个输入特征前给予一个分类标记([CLS])来区分每个句子，表明了该特征输入后用于分类模型。

3.12.2　分类器超参数调试

面对不同任务，只需要在预训练模型的基础上再添加一个输出层便可以完成对特定任务的微调。对于情绪分类任务把分类器作为输出层进行微调，分类器微调主要是对超参数的配置，主要包括训练轮次（也称为训练循环次数、迭代次数或周期）（epoch）（每个轮次包含若干步数（step））、批大小（batch size）、学习率（learning rate）等。最优超参数值配置是要视下游特定任务而定的。

不同模型的分类器采用不同的超参数调试策略。以 $BERT_{BASE}$ 模型为例，$BERT_{BASE}$ 模型最佳微调超参数如表 3.16 所示。

表 3.16　$BERT_{BASE}$ 模型最佳微调超参数

训练轮次	4	4	4	4	4
批大小	8	16	32	64	128
学习率	3e−4	1e−4	5e−5	5e−5	3e−5

本章项目的模型训练轮次设置为 20。

构建分类器对训练集进行训练、对测试集进行评估，分类器的微调除了对神经网络层数、各隐藏层神经元个数、批量数据集中的样本量调试外，主要是对梯度下降优化器超参数调试。在每次迭代中，梯度下降根据自变量当前位置，沿着当前位置的梯度更新自变量。如果自变量的迭代方向仅取决于自变量当前位置，会导致梯度高度敏感于参数空间的某些方向，面临求解具有病态条件（ill-conditioning）的海森（Hessian）矩阵；当神经网络的条件数很多时，梯度下降法也会表现得很差；批量梯度下降（Batch Gradient Decent，BGD）、随机梯度下降（Stochastic Gradient Descent，SGD）、小批量梯度下降（Mini-Batch Gradient Decent，MBGD）等传统梯度下降算法都存在海森病态矩阵（即条件数很大的非奇异矩阵）问题。

梯度下降优化器（optimizer）可以加速梯度下降，改进学习率，使梯度更快地到达全局最优值处，模型稳定快速收敛。动量梯度下降（momentum gradient decent）算法及其变种牛顿动量（Nesterov momentum）梯度下降算法都可以优化处理使梯度前进方向更加平滑，减小振荡扰动，快速达到全局最优解。自适应提升（Adaptive Boosting，AdaBoost）算法、自适应梯度（Adaptive Gradient，AdaGrad）算法、均方根反向传播（Root Mean Square propagation，RMSprop）算法、自适应累积梯度（AdaDelta）算法、自适应矩估计（Adaptive moment estimation，Adam）算法）等优化算法都可以改进学习率的自适应。AdaGrad 维护一个参数的学习速率，可以提高在稀疏梯度问题上的性能，RMSProp 也维护每个参数的学习速率，根据最近的权重梯度的平均值（例如变化速率）来调整，这意味着该算法在线上和非平稳问题上表现良好（如噪声）。Adam 结合了 AdaGrad 和 RMSProp 的优点，其衰减方式类似动量梯度下降，不同于 AdaDelta 和 RMSprop 的历史梯度衰减方式，数学表达式如下：

$$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$$

$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$$

其中，g 代表梯度，t 代表时间，β_1，$\beta_2 \in [0,1)$代表矩估计的指数衰减速率，m 代表一阶矩向量，v 代表二阶矩向量。第一个表达式表示更新有偏的一阶矩估计，第二个表达式表示更新有偏的二阶矩估计。Adam 利用一阶矩估计和二阶矩估计动态调整每个参数的学习率，加快收敛速度，这可以使每一次迭代的学习率都有一个确定范围，经过偏差校正后，参数平稳随着时间而适应，可以较快地得到估计结果。与 RMSProp 基于一阶矩（即均值或期望）的参数学习速率不同，Adam 既使用了一阶矩也使用了二阶矩（非中心矩、不是方差），计算了梯度和平方梯度的指数移动平均值，并且超参数 β_1 和 β_2 分别控制了两个移动平均值的衰减速率；两个移动平均值和超参数 β_1、β_2 的初始值接近 1（推荐值），通过计算偏差校正的一阶矩估计和偏差校正的二阶原始矩估计，使得矩估计的值接近于 0。很多实证研究结果表明，Adam 优于其他的优化随机目标函数的算法。预训练语言模型微调阶段构建分类器的超参数调试中，动量梯度下降因子 β 表示要在多大程度上保留原来的更新方向，取值为 0～1，预训练模型已经训练过的权重，自变量当前位置已接近全局最优点，动量梯度下降因子 β 初始值建议设置为 0.6～0.9；Adam 算法中默认设置学习率为 0.001，超参数 β_1 初始值为 0.9、β_2 初始值为 0.999、ε 初始值为 1e-8。

在整个神经网络训练过程中，使用固定大小的学习率效果往往不好，可能会造成训练集的损失值下降到一定程度后不再下降，而且可能会造成收敛的全局最优点的时候来回振荡。对于分类器微调中学习率调整的一般原则是随着迭代次数的增加，学习率也应该逐渐减少。预训练模型已经训练过的权重，肯定需要施加学习率衰减（learning rate decay）策略。ULMFit 目标域语言模型采用的倾斜三角形学习速率（Slanted Triangular Learning Rates，STLR）设置了不同层应当采取有区别的学习率。

3.13 数据预处理

3.13.1 数据读取、转换和清洗

本章项目模型为已预训练的预训练模型，数据预处理实际仅发生在微调阶段；也就是说，本章的数据预处理是为微调而做的工作，不是训练模型而做的工作。因此，从程序编写流程角度来看，3.13 节放置在模型加载之后。

文本数据预处理工作首先包括数据读取、数据转换和数据清洗。

微调时，需要读取训练集和测试集，训练集、测试集等数据都是.txt 格式的文件，每一个样本是一个句子（新闻标题）及其对应的类别标签（利多、利空、其他），分隔符为“|||”。根据编程使用的函数不同，也可以转换为.csv 或.tsv 格式文件，再进行数据读取。此外，还需要读取模型.json 格式配置文件。

训练集和测试集的语句基本比较规整，数据清洗时可以采取对特殊字符进行处理、删除未标注数据等操作。

3.13.2　分词、填充和其他

文本数据预处理工作还包括分词、填充(padding)和其他。

第一,分词采用预训练语言模型相同的分词方法。因为是中文,所以不必考虑区分字母大小写。如果是英文,则必须考虑是否区分字母大小写。

第二,对不同字数的句子,将在批创建时动态填充每个批中的最大序列长度,使句子对齐(sentence alignment)。

第三,一些预训练语言模型已设置了标签的使用顺序,所以要确保微调时确实使用了它。

3.14　评估指标

3.14.1　混淆矩阵

本章项目未对预训练语言模型直接进行模型评估,而是对模型执行情绪分类下游任务的结果进行评测。混淆矩阵(confusion matrix)或列联表(contingency table)是分类任务的一级评估指标,准确度、精确度、召回率、真负率、假正率、假负率是二级评估指标,F1 分数、G 分数、G 均值、受试者工作特征曲线(Receiver Operating Characteristic curve,ROC curve)、ROC 曲线下面积(Area Under the Curve of ROC,AUC)值、柯尔莫哥罗夫-斯米尔诺夫(Kolmogorov-Smirnov,KS)值/曲线是三级评估指标。此外,还有损失函数经验风险最小化(Empirical Risk Minimization,ERM)、损失函数结构风险最小化(Structural Risk Minimization,SRM)等其他机器学习分类器(模型)性能评估指标。

按照样本所属类别的正例或负例(positive or negative)与分类器预测结果真(正确)或假(错误)(true or false)的组合,样本数据集中的每个样本经过分类器后可以对应真正例(True and Positive,TP)(正确将本类别预测为本类别)、假正例(False and Positive,FP)(错误将其他类别预测为本类别)、真负例(True and Negative,TN)(正确将其他类别预测为其他类别)和假负例(False and Negative,FN)(错误将本类别预测为其他类别)4 种情况。分类器预测结果的混淆矩阵如表 3.17 所示。

表 3.17　分类器预测结果的混淆矩阵

真实类别	预测结果	
	正例	负例
正例	真正例(TP)	假负例(FN)
负例	假正例(FP)	真负例(TN)

其中,行代表的是真实类别,列代表的是分类器预测结果。

对于多分类任务,k 分类预测结果的混淆矩阵就是 $k \times k$ 维的矩阵。本章项目中,对“利好”“利空”“其他”三个类别进行预测,当以“利好”类别为正例时,“利空”和“其他”两个类别就为负例,三分类预测结果的混淆矩阵如图 3.15 所示。

		预测类别		
		利好	利空	其他
真实类别	利好	TP	FN	FN
	利空	FP	TN	FN'
	其他	FP	FN''	TN

图 3.15　以“利好”类别为正例的三分类预测结果混淆矩阵

其中，FN'和 FN''是区别于 FN 的假负例，不参与“利好”类别二级评估指标的计算。对于不同类别对应不同的混淆矩阵，以“利空”或“其他”类别为正例对应的混淆矩阵与图 3.15 不同，而二级评估指标的计算公式相同。

3.14.2　准确度、精确度、召回度和 F1 分数

本章项目的下游任务是情绪分类任务，准确度或正确度（accuracy）是分类任务中最常用的评估指标，转换为百分比后即为准确率或正确率（accuracy rate），用来度量模型性能。准确度是分类器对整个样本判断正确的比例，表示分类预测正确的样本量占样本容量所占的比例，表达式为：

$$\text{准确度} = \frac{\text{正确预测的样本量}}{\text{样本容量}}$$

一般地，样本容量为 n 的标注数据集 S 为有限集合，表示如下：

$$S\{(x_1, y_1), (x_2, y_2), \cdots, (x_i, y_i), \cdots, (x_n, y_n)\}$$

其中，样本 x 的真实类别标签为 y。

k 分类下所有类别的准确度 Acc 的计算公式为：

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

其中，TP＋TN＋FP＋FN 为样本容量。

精确度、精度、精密度、置信度、精确率（precision ratio）或查准率是被分类器判定为正例中的正样本的比例，表示预测为正的样本中有多少是真的正样本。k 分类下某类别的精确度 Precision 的计算公式为：

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

召回度、召回率（recall ratio）、反馈率、查全率、真正率（True and Positive Ratio，TPR）或灵敏度（sensitivity）是被分类器正确预测为正例的样本量占总的正例的比例，表示样本中的正例有多少被正确预测。k 分类下某类别的召回度 Recall 计算公式为：

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

负召回度（即负样本召回度）、负预测度、真负率（True and Negative Ratio，TNR）或特异度（specificity）是被分类器正确预测为负例的样本量占总的反例的比例，表示样本中的负例有多少被正确预测。k 分类下某类别的负召回度计算公式为：

$$\mathrm{TNR}=\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$$

假正率(False and Positive Ratio,FPR)或虚警率是被分类器错误预测为正例的样本量占总的负例的比例,表示样本中的反例有多少被错误预测。k 分类下某类别的假正率 FPR 计算公式为:

$$\mathrm{FPR}=\frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$$

假负率(False and Negative Ratio,FNR)是被分类器错误预测为负例的样本量占总的负例的比例,表示样本中的反例有多少被错误预测。k 分类下某类别的 FNR 计算公式为:

$$\mathrm{FNR}=\frac{\mathrm{FN}}{\mathrm{TP}+\mathrm{FN}}$$

假正率和假负率之和等于 1,即:

$$\mathrm{FPR}+\mathrm{FNR}=1$$

F1 分数或 F1 度量(F1-Score or F1-measure)是精确度和召回度的调和均值,最大为 1,最小为 0。k 分类下某类别的 F1 值计算公式为:

$$\frac{1}{\mathrm{F1}}=\frac{1}{2}\left(\frac{1}{\mathrm{Precision}}+\frac{1}{\mathrm{Recall}}\right)$$

变换上面的等式:

$$\mathrm{F1}=\frac{2}{\frac{1}{\mathrm{Precision}}+\frac{1}{\mathrm{Recall}}}=2\times\frac{\mathrm{Precision}\times\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}=\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}=\frac{2\mathrm{TP}}{\mathrm{N}+\mathrm{TP}-\mathrm{TN}}$$

其中,N 为样本容量,即 TP+TN+FP+FN。

按照上面的 F1 分数计算公式直接得出的是微平均 F1(micro-F1)分数,先计算出每个类别中的真正例(TP)、假正例(FP)、假负例(FN),分别求和得到 3 个样例的总和,代入 F1 分数计算公式即可。微平均将混淆矩阵中的每个样例同等对待,不考虑样例在不同类别下的预测结果差异。

宏平均 F1(macro-F1)分数是 k 分类下未加权的所有类别 F1 分数之和的平均值,即先分别计算每个类别的 F1 分数,再求和、平均。宏平均将每个类别同等对待,所有类别赋予相同的权重,不考虑类别可能存在的样本量占比差异,计算公式为:

$$\text{maro-F1}=\frac{1}{k}\sum_{n=1}^{k}\mathrm{F1}_k$$

本章项目针对情绪分类任务的主评估指标是准确度(正确度)指标,其他指标作为辅助指标,当准确度指标可以充分地区别模型性能优劣时,其他指标不必计算出结果进行比较。在程序输出结果中,准确度为 acc、eval_acc 或 best_eval_acc,不同的深度学习软件框架输出不同。由于项目数据集不存在类别不平衡的问题,不考虑使用对指标加权平均的计算方法。第 4 章的模型评测使用了准确度、精确度、召回度、微平均 F1 分数、宏平均 F1 分数。

3.14.3 损失值

神经网络训练的目的是使真实值与模型预测估计值之间的误差最小化，建模过程就是将现实问题抽象成类凸优化问题，训练过程就是一个存在约束的凸目标函数的最优化求解过程，损失值(loss)是凸目标函数的最小值。由于无法得到显性数学公式求解，通过计算机程序的数值模拟进行求解。

对于一个多元(变量)函数，所有变量的偏导数称为梯度向量，梯度下降法求解就是让梯度向量沿着最陡梯度下降(steepest descent)，使得神经网络收敛。相应地，在神经网络模型的迭代训练过程中，希望通过对模型参数的设置和优化，减少损失值，直至模型收敛。

按照模型训练过程和评测结果的不同，损失值可以分为训练损失值和评测损失值。随着每个训练轮次中的梯度更新，神经网络模型逐渐收敛，训练损失值(程序输出结果中为 loss)应当逐轮减小，渐趋于 0。评测损失值(evaluation loss，程序输出结果中为 loss、eval_loss 或 best_eval_loss)也是神经网络模型评测的一个重要指标，在准确度相同的情况下，在软件框架、任务、微调、评估的代码一样时，评测损失值越小模型越优。本章项目将损失值作为准确度的辅助评估指标，而不是 F1 分数作为辅助评估指标。

3.15 模型评测

启动模型评估程序时，如果 Python 代码中创建了主函数(main())，可以创建一个 Shell 脚本文本来执行命令(例如 main.sh 或 run.sh)，可执行脚本的主要作用是指定运行 Python 文件、读取训练集和测试集数据路径、模型文件路径、输出路径、日志路径、训练轮次、步数、训练和评测批大小等。

3.16 输出过程

不同的模型和不同的代码编写有各自不同的运行过程和输出结果。本节以 $BERT_{LARGE}$ 模型为例，详细展示模型的运行过程和输出结果。

$BERT_{LARGE}$ 的网络架构摘要如图 3.16 所示。

```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================
Input-Token (InputLayer)        [(None, None)]       0
_________________________________________________________________
Input-Segment (InputLayer)      [(None, None)]       0
_________________________________________________________________
Embedding-Token (Embedding)     (None, None, 1024)   21635072
Input-Token[0][0]
_________________________________________________________________
```

图 3.16 $BERT_{LARGE}$ 的网络架构摘要

```
Embedding-Segment (Embedding)   (None, None, 1024)   2048
Input-Segment[0][0]

__________________________________________________________________________________
Embedding-Token-Segment (Add)   (None, None, 1024)   0
Embedding-Token[0][0]

Embedding-Segment[0][0]

__________________________________________________________________________________
Embedding-Position (PositionEmb (None, None, 1024)   524288
Embedding-Token-Segment[0][0]

__________________________________________________________________________________
Embedding-Norm (LayerNormalizat (None, None, 1024)   2048
Embedding-Position[0][0]

__________________________________________________________________________________
_____________________________
Embedding-Dropout (Dropout)     (None, None, 1024)   0
Embedding-Norm[0][0]

__________________________________________________________________________________
_____________________________
Transformer-0-MultiHeadSelfAtte (None, None, 1024)   4198400
Embedding-Dropout[0][0]

Embedding-Dropout[0][0]

Embedding-Dropout[0][0]

__________________________________________________________________________________
_____________________________
Transformer-0-MultiHeadSelfAtte (None, None, 1024)   0
Transformer-0-MultiHeadSelfAttent

__________________________________________________________________________________
Transformer-0-MultiHeadSelfAtte (None, None, 1024)   0
Embedding-Dropout[0][0]

Transformer-0-MultiHeadSelfAttent

__________________________________________________________________________________
Transformer-0-MultiHeadSelfAtte (None, None, 1024)   2048
Transformer-0-MultiHeadSelfAttent

__________________________________________________________________________________
Transformer-0-FeedForward (Feed (None, None, 1024)   8393728
Transformer-0-MultiHeadSelfAttent

__________________________________________________________________________________
Transformer-0-FeedForward-Dropo (None, None, 1024)   0
```

图 3.16 （续）

```
Transformer-0-FeedForward[0][0]
__________________________________________________________________________
Transformer-0-FeedForward-Add ( (None, None, 1024)   0
Transformer-0-MultiHeadSelfAttent

Transformer-0-FeedForward-Dropout
__________________________________________________________________________
Transformer-0-FeedForward-Norm  (None, None, 1024)   2048
Transformer-0-FeedForward-Add[0][
__________________________________________________________________________
Transformer-1-MultiHeadSelfAtte (None, None, 1024)   4198400
Transformer-0-FeedForward-Norm[0]

Transformer-0-FeedForward-Norm[0]
Transformer-0-FeedForward-Norm[0]
......
......
......
Transformer-23-FeedForward-Norm (None, None, 1024)   2048
Transformer-23-FeedForward-Add[0]
__________________________________________________________________________
CLS-token (Lambda)             (None, 1024)         0
Transformer-23-FeedForward-Norm[0

__________________________________________________________________________
dense (Dense)                  (None, 3)            3075
CLS-token[0][0]
==========================================================================
Total params: 324,475,907
Trainable params: 324,475,907
Non-trainable params: 0
```

图 3.16 （续）

$BERT_{LARGE}$ 模型的预测评估结果如图 3.17 所示。

```
   Epoch 1/20
110/110 [==============================] - ETA: 0s - loss: 0.9567 -
accuracy: 0.2378val_acc: 0.73549, best_val_acc: 0.73549

   110/110 [==============================] - 163s 1s/step - loss: 0.9567
- accuracy: 0.2378
```

图 3.17 $BERT_{LARGE}$ 模型的预测评估结果

```
……

   Epoch 6/20
110/110 [==============================] - ETA: 0s - loss: 0.2231 -
accuracy: 0.3407val_acc: 0.81570, best_val_acc: 0.81570

110/110 [==============================] - 149s 1s/step - loss: 0.2231
- accuracy: 0.3407

……

   Epoch 20/20
110/110 [==============================] - ETA: 0s - loss: 1.0838 -
accuracy: 0.0603val_acc: 0.42833, best_val_acc: 0.81570

110/110 [==============================] - 80s 724ms/step - loss: 1.0838
- accuracy: 0.0603
```

图 3.17　（续）

另外一种程序编写方式，只输出最佳预测评估准确度和/或损失值，$BERT_{BASE}$ 模型的预测评估结果如图 3.18 所示。

```
bert-base-chinese
2%|▌              | 3/120 [00:00<00:04, 28.96it/s]1200
100%|████████████| 120/120 [00:04<00:00, 27.25it/s]Test Accuracy:
86.92%
```

图 3.18　$BERT_{BASE}$ 模型的预测评估结果

以上运行过程和输出结果中，模型训练和模型评测交替，模型训练的准确度应当接近模型评测，如果模型训练的准确度高于模型评测，说明模型过拟合；如果模型训练的准确度低于模型评测的准确度，说明模型欠拟合。

最终得到模型的最佳准确度和/或损失值。

3.17　结果汇总

将 28 个对比模型的预测评估结果汇总展示，并按照效果高低进行排名，如表 3.18 所示。不同的数据预处理和微调方法、不同的程序代码、不同运行环境得到的模型评测结果(evaluation result)都会有差异，因此评测结果可能存在偏差。

表 3.18　对比模型评测结果

效果排名	模型名称	准确度	损失值
1	$RoBERTa_{LARGE}$-wwm-ext	0.882 25	0.685 39
2	RoFormer	0.873 72	0.024 11

续表

效果排名	模型名称	准确度	损失值
3	WoBERT^{+}（WoBERT Plus）	0.872 01	0.032 72
4	$BERT_{BASE}$-BiLSTM-CNN-Attention	0.871 02	0.001 22
5	$Longformer_{BASE}$ Chinese	0.870 31	0.423 66
6	SKEP	0.870 31	0.855 36
7	$ELECTRA_{LARGE}$ Chinese	0.866 89	0.771 44
8	WoBERT	0.863 48	0.000 23
9	$BERT_{BASE}$-wwm-ext	0.863 48	0.839 85
10	FinBERT 1.0-Base	0.861 77	0.505 39
11	$XLNet_{MID}$ Chinese	0.855 43	0.497 50
12	WoNEZHA	0.854 95	0.053 65
13	$BERT_{BASE}$-CNN-BiLSTM-Attention	0.854 72	0.014 15
14	$BERT_{BASE}$-BiGRU-Attention	0.852 10	0.021 65
15	Mengzi-$BERT_{BASE}$	0.848 12	0.000 03
16	Mengzi-$BERT_{BASE}$-fin	0.848 12	0.0886
17	$MacBERT_{BASE}$	0.848 12	0.442 18
18	$RoBERTa_{LARGE}$ Chinese	0.846 42	1.150 42
19	$BERT_{LARGE}$ Chinese	0.842 25	0.912 32
20	$BERT_{BASE}$ Chinese	0.839 59	0.730 89
21	ERNIE1.0 Base	0.828 12	0.544 29
22	NEZHA-base	0.812 29	0.034 68
23	ZEN	0.802 04	0.124 32
24	$ALBERT_{BASE}$ Chinese	0.762 80	1.833 41
25	ERNIE Tiny	0.757 68	0.674 29
26	simplified-Chinese ELMo	0.752 56	0.562 46
27	Chinese fastText	0.720 14	1.109 02
28	Chinese ULMFiT	0.680 89	0.630 62

从表 3.18 可以看出，对于本章创建的中文标注语料库，在 28 个对比模型中，基准模型 $BERT_{BASE}$ Chinese 的准确度为 0.839 59，$RoBERTa_{LARGE}$-wwm-ext 模型效果最佳，预测评估准确度达到 0.882 25，比基准模型准确度高出 0.042 66。WoBERT 和 FinBERT 1.0-Base 的准确度的小数点后 5 位一样，由于两个模型的实现采用了不同的代码和微调，所以无法通过比较损失值来确定模型效果排名。比较小数点后的 15 位值，WoBERT 的准确度是 0.863 481 228 668 942，$BERT_{BASE}$-wwm-ext 的准确度是 0.863 481 223 583 221，因此，WoBERT 的模型效果排名比 $BERT_{BASE}$-wwm-ext 高。

3.18 模型对比项目结论

本节通过从所有模型、语言表征、模型规模大小、骨干网络架构、模型调优、掩码机制、训练样本生成策略、建模目标、组合模型九个方面进行比较分析的总结，对于本章创

建的、接近业界实际的、真实的中文金融情绪标注语料库，面向中文金融文本情绪分类特定任务，可以揭示出以下结论。

第一，数据决定了模型效果上限，而调参只是在逼近模型效果上限。也就是说，数据的样本容量、特征空间、边缘概率分布、条件概率分布都会对模型性能水平造成决定性影响。本章项目所有模型评测结果都没有达到优秀（准确率都未超过 90%），其中一个原因是样本容量小；但是本章项目是为了比较模型效果相对的高低，因此绝对准确率的高低并不妨碍结果对比，不影响项目目的和项目结论。这个结论可以从另一个对比评测结果分析得到支持，中文文本分类 PyTorch 实现从清华大学自然语言处理与社会人文计算实验室发布的中文文本分类数据集 THUCNews 中抽取了 20 万条新闻标题（文本长度为 20～30，一共 10 个类别，每类 2 万条），数据集被划分为训练集 18 万条、验证集 1 万条、测试集 1 万条，训练集样本可能覆盖了绝大多数的测试集样本，几个对比模型效果均达到 90%以上。

第二，从语言表征的角度来看，上下文感知表征模型更优，双向语言模型更优。上下文无感表征模型 fastText 的效果不如其他上下文感知表征模型，单向语言模型 fastText 和两个单向拼接语言模型 ULMFit 不如 ELMo、BERT 等双向语言模型更优。作为浅层预训练模型，fastText 只有一个隐藏层，结合了 n 元（n-gram）和分层 softmax 的优势，对样本量要求少，且训练速度快；但是不能区分上下文语境，比如对“你打给我”和“我打给你”这两句话的理解是一样的。由于不能保证在所有的数据集上的评测结果都是上下文感知表征模型更优，双向语言模型更优，对早先的模型进行评测是有必要的，评测的目的是选择最适合的模型。

第三，从模型大小的角度来看，评测结果表明，预训练语言模型的规模越大效果越好，极小（tiny）模型的效果最差、压缩模型效果较差、基础（base）模型较好、大号（large）模型最好。这个结论是与所有的学术论文中评测结论一致的，这也是预训练语言模型规模越做越大的原因。

第四，从模型骨干网络架构的角度来看，高效变换器架构（“X-former”）模型效果比绝大多数变换器架构更好。中文句子的字数一般比英文句子多，也就是说，序列长度更长，而高效变换器架构更擅长处理长序列，对较长序列的处理能力更强，使采用高效变换器架构的模型对长句子的预测准确度较高。由于本章项目评测数据集中既有短文本也有长文本，因此 Longformer 模型的表现没有超过 $\text{RoBERTa}_{\text{LARGE}}$-wwm-ext 模型。然而，RoFormer 模型主要特点是可以直接处理任意长的文本，是目前唯一一种可用于线性注意力的相对位置编码，在本章评测数据集上准确率排名第二。

第五，从模型调优的角度来看，调优后模型比原模型更优。RoBERTa 模型是由脸书人工智能（Facebook AI）和华盛顿大学的研究团队于 2019 年 7 月共同发布的，对 BERT 模型做了以下几点调整：①动态调整掩码机制，而 BERT 模型的静态掩码是在数据预处理阶段对序列进行掩码，因此输入到模型中的每一个被掩码的句子是相同；②删除了前后句预测建模目标，预训练任务仅为掩码语言模型单一建模目标；③RoBERTa 采用的字节对编码（Byte-Pair Encoding，BPE）是字符级和词表级表征的

混合，是 BERT 模型采用的词块编码的一种特殊形式；④预训练数据更多，由 16GB 增加到 160GB，训练步数更长，从 100K 增长到 500K，批大小更大，从大号模型 256（步数 1M、学习率 1e-4）增大到 2K（步数 125K、学习率 7e-4）、8K（步数 31K、学习率 1e-3）。RoBERTa 模型是 BERT 模型的成熟版，充分发挥了 BERT 性能，它的经验对模型性能提升有很多可借鉴的地方，同时很多论文的评测结论表明，增大批大小会显著提升模型效果。

第六，从掩码机制角度来看，全词掩码比子词掩码更优。大号中文全词掩码 RoBERTa 模型（$RoBERTa_{LARGE}$-wwm-ext）使用 30GB 中文文本，近 3 亿个句子，100 亿个分词标记（token）（即中文的字），产生了 2.5 亿个训练实例（instance）数据，覆盖新闻、社区讨论、多个百科，包罗万象，覆盖数十万个主题等，使用了超大（8K）的批大小。

第七，从训练样本生成策略角度来看，以词为单位输入模型更优。由于谷歌官方发布的 $BERT_{BASE}$ Chinese 中，中文是以字为粒度进行切分，没有考虑到中文分词的特殊性。很多中文预训练模型不能够沿用中文 BERT 模型的做法，仅聚焦于小颗粒度文本单位元（字）的输入，而应当注重字嵌入到词嵌入的转换，同时保持训练序列输入长度较长。$RoBERTa_{BASE}$-wwm-ext 将全词掩码的方法应用在了中文中，使用了中文维基百科（包括简体和繁体）进行训练，并且使用了哈尔滨工业大学语言技术平台（Language Technology Platform，LTP）作为分词工具，即对组成同一个词的汉字全部进行掩码。$WoBERT^{+}$（WoBERT Plus）模型基于 $RoBERTa_{BASE}$-wwm-ext 模型继续预训练，以掩码语言模型为建模目标，初始化阶段将每个词用 BERT 自带的分词器切分为字，然后用字嵌入的平均作为词嵌入的初始化，能够将单个汉字与词的语义信息相融合，采用 30GB 以上大小的通用语料训练，序列最大长度为 512、学习率为 1e-5，批大小为 256，累积梯度 4 步，训练 25 万步。虽然 $WoBERT^{+}$ 是基础模型，但其性能已经超过了 $BERT_{BASE}$ 的组合模型，接近大号模型 $RoBERTa_{LARGE}$-wwm-ext。

第八，从建模目标角度来看，预训练过程中仅使用掩码语言模型单一任务目标，比两个任务目标更好，RoBERTa 样式模型比 BERT 样式模型效果更好。另外，ELECTRA 模型加入了分词替换的对比学习建模目标，这种方法与对抗训练类似，但略有不同：生成器的输入只有潜在空间，没有随机噪声；采用最大似然估计计算极值；生成器会去预测掩码生成的新分词标记的概率值，判别器能更加准确地识别每个位置的标记是否真实。ELECTRA 模型效果优于同等大小的 BERT，且媲美 RoBERTa 模型。由此可以预见到，对抗训练对于预训练语言模型起到一定的性能提升作用。

第九，从组合模型角度来看，组合模型、集成模型或融合模型优于组合中的任意单个模型。双向门控循环单元（BiGRU）和双向编码器从变换器表征相结合，使得 BERT 本编码层学习上下文语境词表征改善了 BiGRU 网络的分类性能。利用 BERT 模型对文本到动态的字符级嵌入进行转换，BiLSTM 和 CNN 组合输出特征，融合充分利用 CNN 提取局部特征的优势和 BiLSTM 具有记忆的优势将提取的上下文特征链接起来，更好地表征文本，从而提高文本分类任务的准确性。本章项目比对的 $BERT_{BASE}$-BiGRU-Attention、$BERT_{BASE}$-CNN-BiLSTM-Attention、$BERT_{BASE}$-BiLSTM-CNN-

Attention 三个组合模型在中文文本中具有强大的语义捕捉能力和远程依赖关系，提高了情绪分类预测的性能，$BERT_{BASE}$-BiLSTM-CNN-Attention 的表现更好。CNN-BiLSTM 和 BiLSTM-CNN 都是 CNN 和 BiLSTM 两者的结合，可以同时利用 CNN 模型识别局部特征和 LSTM 模型处理文本序列的能力。不同的是，CNN-BiLSTM 是 CNN 层接收词向量作为输入、CNN 的卷积层提取局部特征后池化层输出汇集到一个较小尺寸，输入到 BiLSTM 层利用特征来学习文本序列，再经过全连接层输出分类标签；BiLSTM-CNN 是 BiLSTM 层接收 BERT 输出的词向量将所有句子填充到等长、生成新的编码输入到一个完整的 CNN 层、提取局部特征到卷积层、经过池化层输出汇集到一个较小的维度、最终经过全连接层输出分类标签。比较两种不同的结合方式，CNN-BiLSTM 将 BiLSTM 层插入到 CNN 层的池化层后，输出结果与全连接层进行相连，CNN 层可能会丢失一些原始标记信息，而 BiLSTM- CNN 既包含原始标记信息，又包含序列中前后变化信息（先前和即将出现的周围单词），此时 CNN 卷积层中的卷积核（滤波器）真正发挥了作用，从而获得更好的准确性。

以上结论仅对本项目的测试集数据样本负责，不代表其他数据集下模型的效果状况。

小结

本章详细记录了主流的中文预训练模型在中文自然语言理解任务上完整的项目过程、操作要点、输出结果和项目结论。为进一步提高性能模型性能，创建全新的预训练语言模型提供了实践基础，并指出了改进方向。