2022-2023 学年秋季学期东北大学

《自然语言处理》课程实践报告

项目名称：	基于预训练词向量和 BiGRU 的微博文本情感分析
姓　　名：	李欣迟
专　　业：	计算机科学与技术
班　　级：	计硕 2203 班
学　　号：	2201822

2022 年　12 月 27 日

基于预训练词向量和 BiGRU 的微博文本情感分析

一、 项目背景及意义

近年来，我国各类公共事件频发，网络社交媒体蓬勃发展，越来越多的人开始并频繁使用网络社交平台来发表自己对于人物、事件、话题等实体及其属性的观点或看法，由此产生的评论文本数据更是呈现爆炸式的增长。作为时下国内受众最广的移动社交媒体，微博（在国内一般指新浪微博）的用户规模巨大。

在这样的背景下，微博上每天都会产生数据量庞大的文本数据，其中隐含的巨大价值将对实际的生产和生活具有很强的指导意义。合理地分析和利用微博文本数据中包含的用户观点信息并，了解用户总体的情感分布情况，将为个人、机构和政府等各类群体的决策行为提供有效帮助，起到辅助和支撑作用。

将文本情感分析方法应用于微博文本数据中，可以有效实现舆论监控与控制，使得相关改进举措更加合理、高效，为各类决策的正确性和有效性提供更加可靠的保障，对于商户、政府部门和微博平台等各类群体来说都具有重要的意义。对于商户来说，可以帮助其对商品质量、营销策略、售后服务等方面的不足进行改善，以提高品牌声誉和销售业绩；对于政府部门来说，对公众情感进行分析为其提供当前社会舆情的及时反馈，让其能够准确把握公众情绪的变化，以迅速做出反应、对相关政策进行合理调整；对于微博平台来说，掌握不同用户的情感倾向，为其制定个性化的推荐与服务，可以逐步完善整个微博平台的构建，提高整体用户满意度，进而吸引更多用户使用微博平台。

二、 实验目的

- 对 weibo_senti_100k 数据集进行预处理，结合中文微博文本的特点，进行文本处理与中文分词等操作；
- 合理使用预训练词向量模型，提高模型准确率；
- 掌握 GRU 模型的原理以及如何利用 Pytorch 架构进行编程实现；
- 结合模型的测试结果，分析实验过程中可能存在的问题，思考如何继续优化模型。

三、 模型框架

结合微博文本数据的特点，使用面向微博领域的预训练中文词向量模型对文

本分词结果进行词嵌入，使用双向门控循环神经网络模型进行训练。实验模型框架主要由输入层、数据预处理层、词嵌入层、BiGRU 层、情感分类层和输出层组成，具体模型框架如图 1 所示。

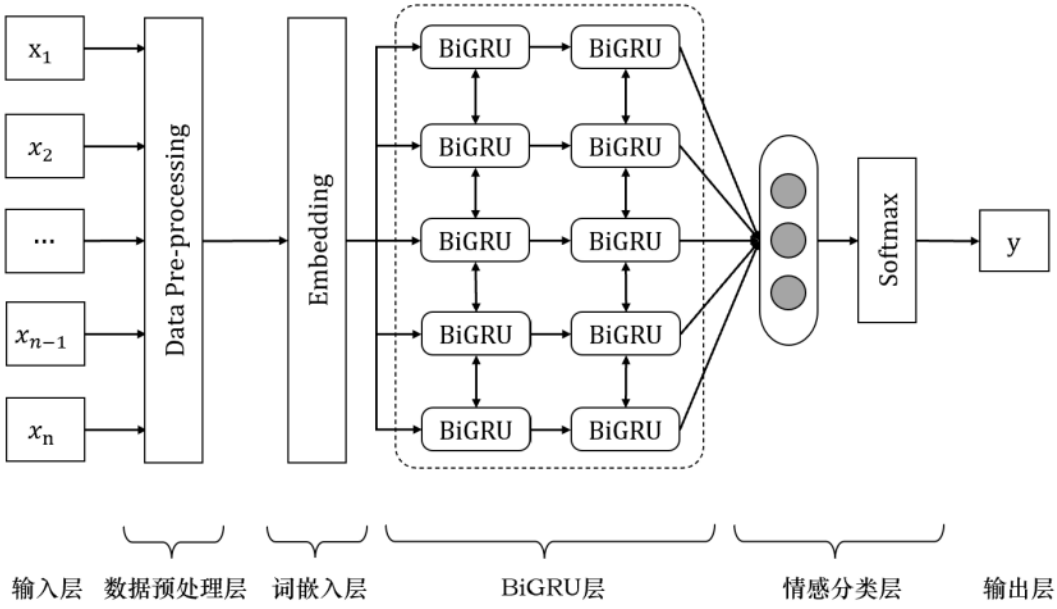

图 1 模型框架

四、 相关理论

(一) 词嵌入

神经网络只能接收数值类型的输入数据，需将文本形式的微博文本数据转化成向量形式。为了更加契合微博评论文本数据的特点，准确地表示语义关系，使用词嵌入技术，将微博评论词汇映射到向量空间中，得到词向量。 由于实验数据集中包含的词汇数量有限，自行训练词向量很有可能导致词汇表达的不准确，进而影响后续模型训练结果的有效性。

使用面向微博领域的预训练中文词向量文件 sgns.weibo.bigram-char 对微博评论中的词进行词向量初始化。 该文件中包含 112728 个利用 Word2vec 算法训练出的 300 维词向量，训练模型为 Skip-Gram，优化方法为 Negative Sampling。预训练模型中的中文词向量基本设置如表 1 所示。

表 1 预训练词向量基本设置

Parameter	Value
Window Size	5
Dynamic Window	Yes
Sub-sampling	1e-5
Low-Frequency Word	10
Iteration	5
Negative sampling*	5
Context features	Word+Character+Ngram

使用主成分分析 PCA 方法将文件中高维度的中文词向量线性映射到三维空间， 对词向量进行可视化，具体结果如图 2 所示。

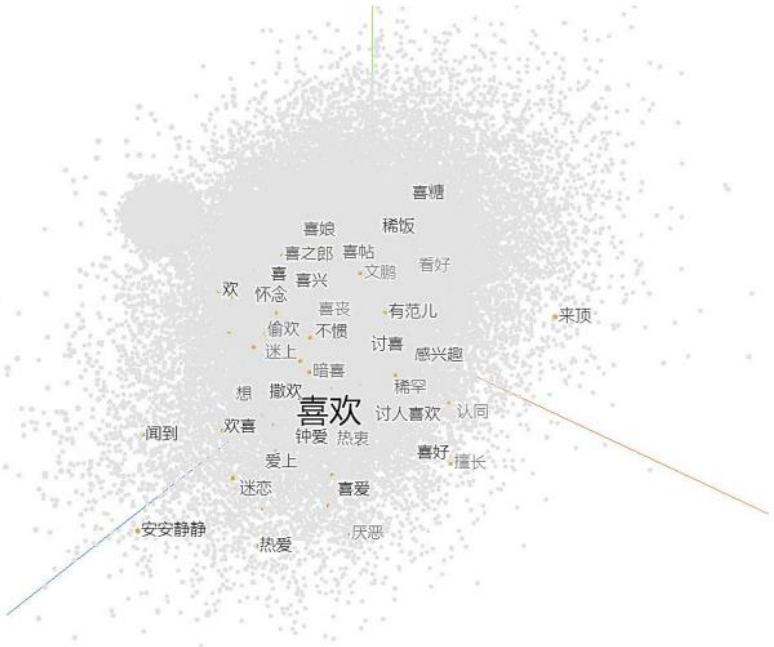

图 2 词向量 3D 可视化结果

(二) GRU

GRU 是对 LSTM 的一种改进，在 LSTM 的基础上减少了“门”的数量，只设置了了更新门 z 和重置门 r，其模型结构如图 3 所示。

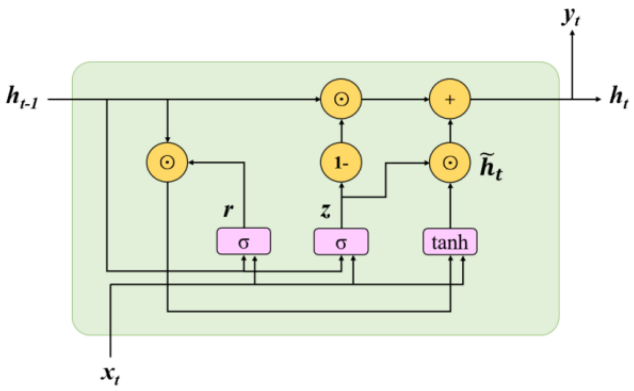

图 3　GRU 模型结构

- 更新门（update gate）。GRU 中的更新门兼顾了遗忘门和输入门的，负责隐藏状态的更新，决定了历史信息的保存量；
- 重置门（reset gate）。重置门决定了新的记忆信息；
- 状态更新与输出。利用更新门、历史隐藏状态和中间隐藏状态决定最终的输出（即隐藏状态）。

(三)　BiGRU

作为门控循环单元神经网络 GRU 的变体，BiGRU 在 GRU 的基础上增加了双向 捕获上下文信息的功能。GRU 接收文本向量时，只能从前向后读入，这使得评论中 靠后的词对整条评论情感倾向判断的影响更大，进而导致结果出现偏差。而 BiGRU 的双向语义捕获功能很好的解决了这个问题。BiGRU 结构中建立了向前传播和向后传播两种连接，使得每个时刻的输出都包含了更加充分的双向上下文信息，具体结构如图 4 所示。

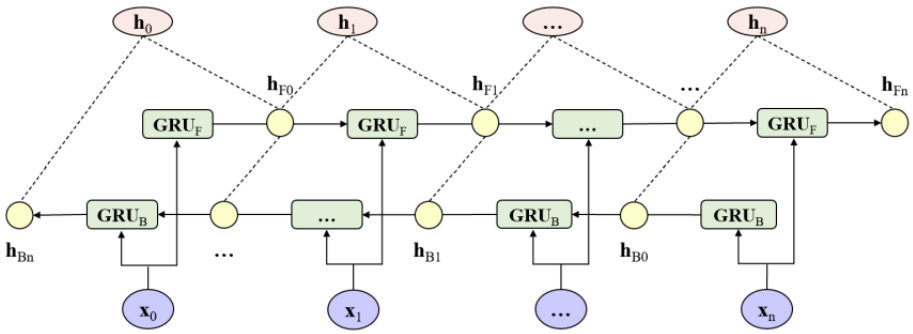

图 4　BiGRU 神经网络单元结构

五、 环境配置

实验环境为 Windows 操作系统，使用的编程语言为 Python，选择具备良好交互性的 Jupyter Notebook 作为编程软件，利用开源工具 Pytorch 深度学习框架进 行神经网络的建模工作。具体的实验环境配置如表 2 实验环境配置和所示。

表 2 实验环境配置

实验环境	环境配置
操作系统	Windows11
编程环境	Jupyter notebook(anaconda3)
编程语言及版本	Python 3.9
深度学习框架及版本	Pytorch-cpu 1.

此外，实验过程中主要使用的模块及扩展库的名称和作用如表 3 所示。

表 3 模块/扩展库名称及作用

模块/扩展库名称	主要作用
Numpy	数组计算
Pandas	数据分析（运算、清洗、特征加工等）
Matplotlib	绘制各种图表
Torch	构建深度学习模型
Torchtext	处理文本数据

六、 实验内容与步骤

(一) 数据集介绍及初步分析

weibo_senti_100k 数据集包含 10 万余条中文微博文本及其情感标签。数据集的整体特征与属性描述如表 4 和表 5 所示。

表 4　weibo_senti_100k 数据集整体特征

数据集名称	数据类型	字段数	行数
weibo_senti_100k	字符、数值数据	2	119988

表 5　weibo_ senti_100k 数据集属性描述

字段名	数据类型	字段描述
label	Numeric	情感标签（0 负向，1 正向）
review	String	微博文本内容

weibo_senti_100k 数据集内容示例如图 5 所示。

	label	review
0	1	更博了，爆照了，帅的呀，就是越来越爱你！生快傻缺[爱你][爱你][爱你]
1	1	@张晓鹏jonathan 土耳其的事要认真对待[哈哈]，否则直接开除。@丁丁看世界 很是细心...
2	1	姑娘都羡慕你呢…还有招财猫高兴……//@爱在蔓延-JC:[哈哈]小学徒一枚，等着明天见您呢/...
3	1	美~~~~~[爱你]
4	1	梦想有多大，舞台就有多大![鼓掌]

图 5　weibo_senti_100k 数据集内容示例

两类评论各占评论总数的 50%左右，整体情感极性分布较为平衡，具体的评论数量分布情况如图 6 所示。

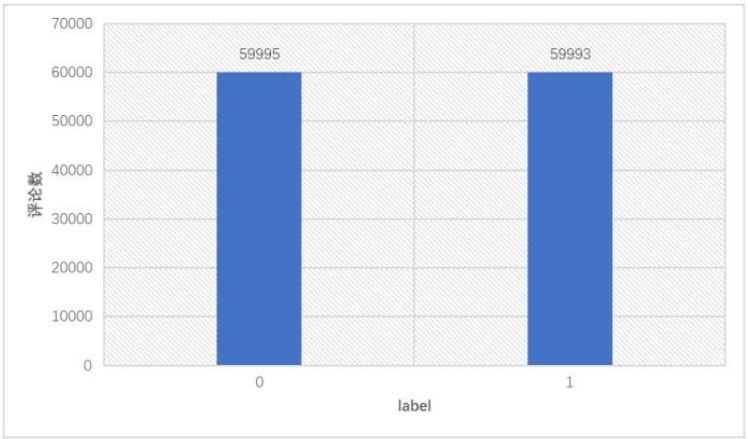

图 6　weibo_senti_100k 数据集微博评论数量统计

(二) 数据预处理

对 weibo_senti_100k 数据集进行了数据清洗和中文 文本处理操作，具体过程如下：

1) 数据集导入

使用 pandas 库中的 csv 文件读取函数导入数据集，加载为 DataFrame 类型的数据集 df。使用 info()方法查看数据集的简要信息和形状，了解数据集的行列数、字段名、数据类型、总体缺失值情况和所在内存大小，运行结果如图 7 所示。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119988 entries, 0 to 119987
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   label   119988 non-null  int64
 1   review  119988 non-null  object
dtypes: int64(1), object(1)
memory usage: 1.8+ MB
```

图 7 weibo_senti_100k 数据集简要信息

2) 数据集初步清洗

由数据集简要信息可知，weibo_senti_100k 数据集中不存在任何的缺失值，因此无需进行缺失值处理，直接对评论数据进行重复值处理。首先，识别并统计“review”字段中的重复数据，结果发现存在 1534 条重复的微博评论文本。然后，查看具体的数据重复情况并进行分析。最后，过滤掉重复的评论文本数据。

3) 中文文本处理

结合中文微博评论文本的格式特点，依次进行特殊符号清洗、英文大小写转换、繁简转换、中文分词和去停用词五项处理操作，获取可以最大程度上体现文本特征的中文分词结果。具体的中文文本处理流程如图 8 所示。

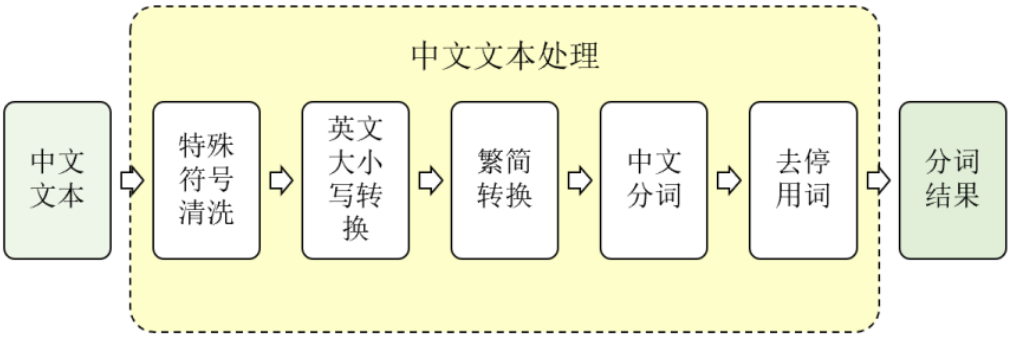

图 8 中文文本处理流程

- **特殊符号清洗。**微博文本中存在的特殊符号无法进行分词，如果不进行处理，会严重干扰后续工作的进行。通过观察现有数据，对特殊符号按书写格式类型进行划分。划分后的特殊符号共计五类，分别为用户名及相关符号、表情符号、网址、非汉字和字母的字符和连续出现的无意义空格符。针对这些特殊符号，利用相应的正则表达式对其进行清洗，各类正则表达式如所示。
- **英文大小写转换。**虽然针对中文微博进行分析，但网络评论中难免出现英文单词或语句。为了统一这些英文字符在微博用户们进行观点信息表达的过程中起到的作用和影响，使用 Python 中的 lower()方法，将所有大写的英文字母转换成小写字母。
- **繁简转换。**在进行文本处理工作时，文本中的繁体字会在一定程度上增加时空复杂度，增加系统开销。OpenCC 是一个针对中文的开源的字体转换项目，支持 Python 语言，兼容 Windows 环境。使用 python 版本的 OpenCC 将微博评论中的繁体字全部转换成简体字，转换方法设置为“t2s”。
- **中文分词。**分词是中文文本处理工作中特有一个环节，是特征提取的关键。受文本主题、文本适应领域等因素的影响，分词依据和规则应结合实际做出相应调整。而微博是一种网络化程度非常高的语言环境，分词的领域倾向性更是十分重要。由北京大学研发的 Pkuseg 中文分词工具为使用者提供了面向多领域的预训练分词模型，分词准确率也高于市面上其他中文分词工具。使用 Pkuseg 中提供的面向网络领域的 web 分词模型对微博评论文本进行有针对性的分词处理工作。
- **去停用词。**文本数据中的停用词大多为对实际任务无意义的词语，去除后可大幅减少计算的工作量。因此，为得到的更好的文本处理效果，对评论中的

停用词进行删除操作。在设置停用词表时，将多种常用停用词表和微博评论实际情况相结合。通过对多个停用词表的内容进行筛选和合并，构建综合停用词表。此外，从微博评论文本的实际特点出发，在已建立的综合停用词表中加入了“转发微博”、“转发”、“回复”等微博评论停用词。

上述操作通过定义中文文本预处理函数 chinese_pre(text)实现。对“review”字段使用函数完成文本处理工作，将处理后的分词结果保存为新字段“cutword”。“cutword”字段部分内容示例如图 9 所示。

```
博 爆照 帅 越来越 爱 生快 傻 缺
土耳其 事 开除 细心 酒店
姑娘 羡慕 招财猫 小学 徒 枚 明天 大佬 范儿
美
梦想 舞台
问答 社区 收到 大学生 发给 私信 偶 喜欢 阿姨 偶是 阿姨控 回 阿姨 稀饭 盆友 偶 盆友 控
吃货们 无不啧啧称奇 好不 喜欢 ps 写错 字
sweet morning love enjoy living smile 爱 享受 生活 微笑 早安 甜心们
```

图 9 “cutword”字段部分内容示例

4) 数据集进一步清洗及构建

查看文本处理后的数据集缺失值和重复值情况。由于数量较少，对两类问题数据进行直接删除操作。将“label”字段和“cutword”字段保存为 csv 格式的新数据集 cut_review。以 6：2：2 的比例将数据划分为训练集、验证集和测试集。

(三) 数据集进一步分析

cut_review 数据集包含 113549 条数据标注好情感标签的中文分词数据，具体内容示例如图 10 所示。

	label	cutword
0	1	博 爆照 帅 越来越 爱 生快 傻 缺
1	1	土耳其 事 开除 细心 酒店
2	1	姑娘 羡慕 招财猫 小学 徒 枚 明天 大佬 范儿
3	1	美
4	1	梦想 舞台

图 10 cut_review 数据集内容示例

cut_review 数据集的整体情感分布情况如表 6 和图 11 所示。

表 6　cut_review 数据集各类数据具体数量统计

情感标签	数据量
负向	56178
正向	57371

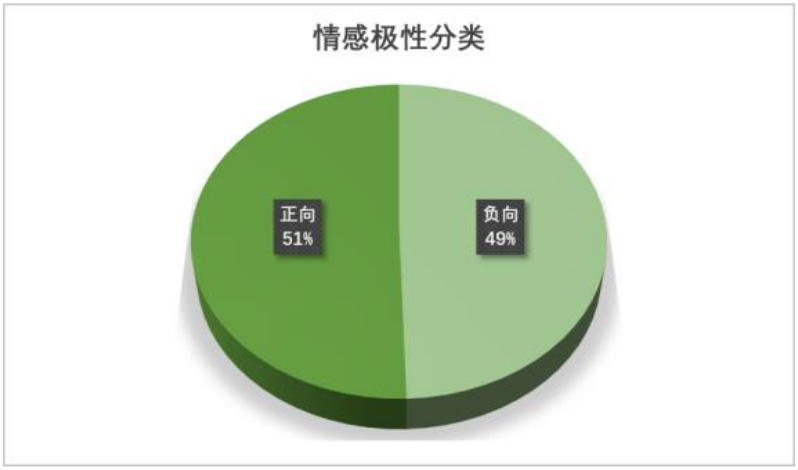

图 11　cut_review 数据集情感分布

对 cut_review 数据集中微博评论文本的分词结果进行词数统计，绘制分布直方图，如图 12 所示。

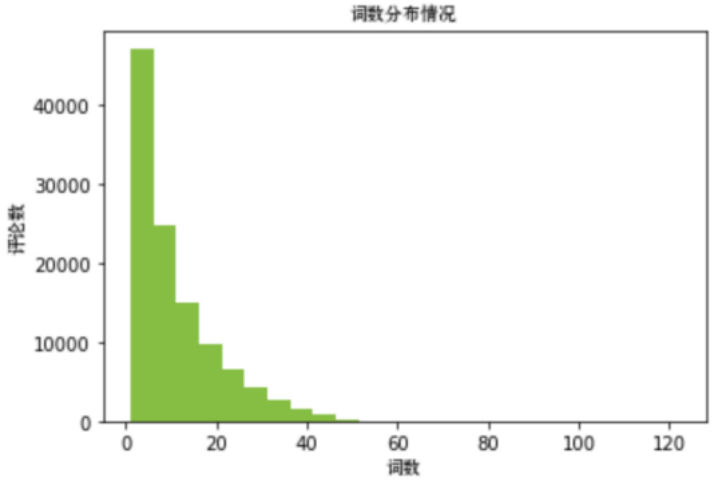

图 12　cut_review 数据集微博评论词数分布

由图可知，cut_review 数据集中每条评论的词数分布主要集中在 20 以下，绝大多数的微博评论文本长度都小于 40，因此本文在生成词嵌入矩阵时对评论文本进行截断与补齐操作，将每条评论的长度都固定为40。使用扩展库 wordcloud 和 matplotlib 绘制微博正向评论和负向评论的词云图，实现文本数据的可视化，词云如图 13 和图 14 所示。

图 13 正向评论文本数据词云图

图 14 负向评论文本数据词云图

由图可知，微博文本的正负向评论中都含有“想”、“说”、“太”、“吃”、“中国”等词。在正向评论中出现频率最高的是“喜欢”、“谢谢”等表达积极情感的词，而“死”、 “哭”、“可惜”等蕴含消极情感的词多集中出现在负向评论中。

(四) 模型定义与训练

使用 torchtext 库中的 data 模块准备模型所需的词向量模型。利用 Vectors() 方法导入预训练词向量模型 sgns.weibo.bigram-char，规定对未命中的词进行 Xavier 初始化，完成词向量表示规则的定义工作。根据定义好的规则构建训练集词汇表。

使用 Pytorch 框架提供的神经网络工具包 nn 定义模型，创建实例对象 bigru_model，具体的模型参数设置如表 7 模型参数设置所示。

表 7 模型参数设置

参数名称	含义	参数值
EMBEDDING_DIM	词向量维度	300
HIDDEN_SIZE	隐藏神经元个数	128
NUM_LAYERS	隐藏层层数	2
NUM_CLASSES	分类数量	2
ACTIVE_FUNCTION	激活函数	ReLU

模型对象实例创建结果如图 15 所示。

```
BiGRUNet(
  (embedding): Embedding(100134, 300)
  (gru): GRU(300, 128, num_layers=2, batch_first=True, dropout=0.5, bidirectional=True)
  (fc): Sequential(
    (0): Linear(in_features=256, out_features=128, bias=True)
    (1): Dropout(p=0.5, inplace=False)
    (2): ReLU()
    (3): Linear(in_features=128, out_features=2, bias=True)
  )
)
```

图 15 模型对象实例

将导入的词向量设置成到 bigru_model 词嵌入的初始权值，嵌入结果如图 16 所示。

```
tensor([[-0.1004,  0.0046, -0.1308,  ..., -0.0477, -0.1237,  0.1133],
        [ 0.0093, -0.0647, -0.0591,  ...,  0.0461,  0.1245, -0.0606],
        [ 0.4581,  0.1740,  0.2048,  ..., -0.0447, -0.4917,  0.0306],
        ...,
        [ 0.3087,  0.1839,  0.1465,  ..., -0.3507,  0.1528, -0.3567],
        [ 0.0502, -0.0758,  0.0318,  ..., -0.0353,  0.1259, -0.1357],
        [ 0.1971,  0.1206,  0.1384,  ..., -0.2737,  0.1661,  0.0379]])
```

图 16　bigru_model 词嵌入矩阵

定义模型训练函数，在训练集上对模型各项参数训练和调整。训练时的超参数设置如表 8 所示。

表 8 超参数设置

参数名称	含义	参数值
LEARNING_RATE	初始学习率	0.003
BATCH_SIZE	批训练样本数	128
NUM_EPOCHS	训练迭代次数	5&10
OPTIMIZER	优化器	Adam
LOSS_FUNC	损失函数	CrossEntropyLoss

对模型进行 10 轮迭代训练，训练过程数据及可视化结果如图 17 和图 18 所示。（每迭代 5 轮学习率减小到原来的 10%）

	epoch	train_loss_all	train_acc_all	val_loss_all	val_acc_all	learn_rate
0	0	0.597311	0.654549	0.523494	0.736583	0.0030
1	1	0.416790	0.817134	0.403114	0.812071	0.0030
2	2	0.231105	0.903112	0.255949	0.888615	0.0030
3	3	0.141432	0.939245	0.149532	0.934982	0.0030
4	4	0.101989	0.954071	0.129290	0.944967	0.0030
5	5	0.062353	0.971888	0.083244	0.963400	0.0003
6	6	0.055035	0.975422	0.064215	0.971274	0.0003
7	7	0.048784	0.977593	0.053329	0.975849	0.0003
8	8	0.043758	0.979514	0.045842	0.979412	0.0003
9	9	0.040878	0.981361	0.045545	0.980732	0.0003

图 17 模型训练过程数据

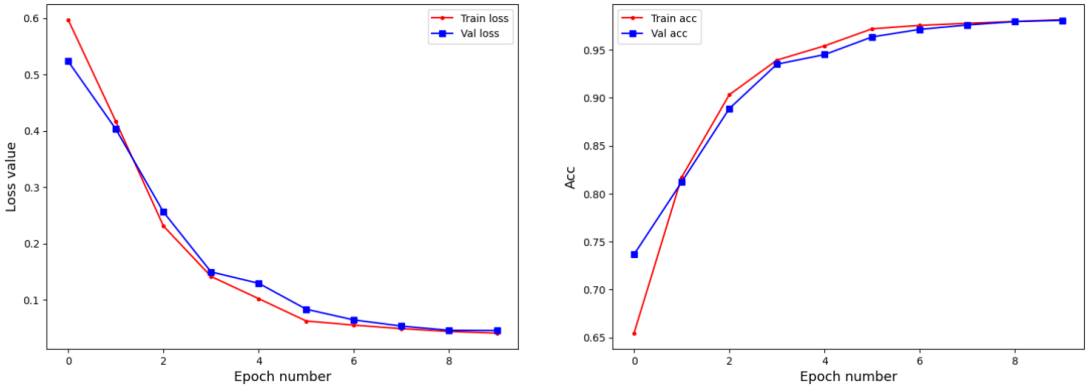

图 18 模型训练过程中的损失与准确率变化

通过观察模型的训练过程可以发现，模型在前 3 次迭代训练中的准确率提升地非常快，从第 4 轮开始速度逐渐放缓。经过 5 次迭代后模型在训练集和验证集上的准确率都达到了 94%以上。整体看来，模型对数据的拟合程度较好。经过 10 轮迭代后，模型在训练集和验证集上的准确率都达到了 98%以上。

(五) 模型测试与评价

为了更加直观地分析实验结果，对测试结果的混淆矩阵进行可视化，具体如图 19 所示。

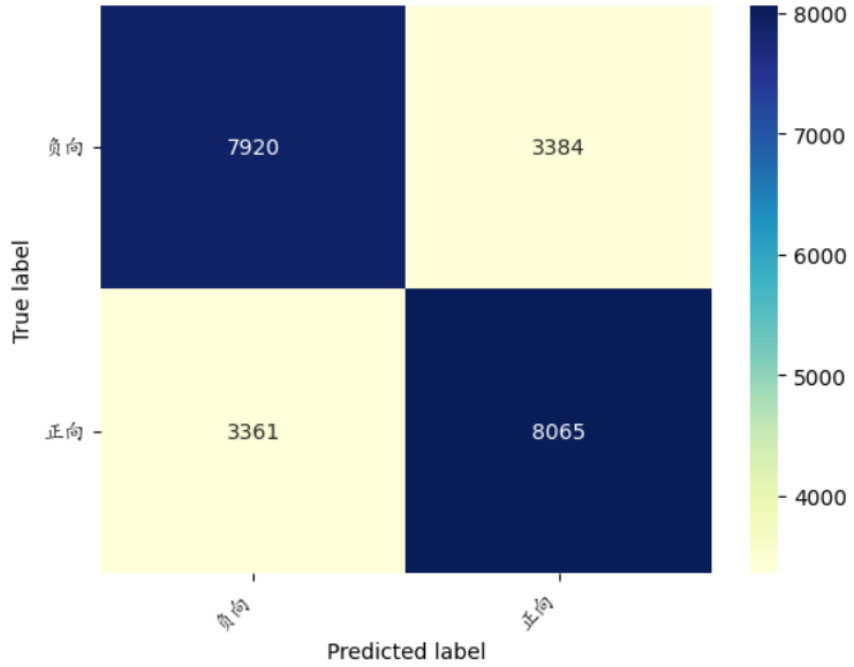

图 19 模型测试结果的混淆矩阵

根据混淆矩阵可以计算出，训练好的模型在测试集中的准确率为 0.6942，精度为 0.69442，召回率为 0.69585，F1 值为 0.6942。很显然，模型在测试集的性能表现并没有到达预期的效果。

七、 实验总结与分析

根据图 17 和图 18 可以确定模型在验证集上拟合的很好，性能表现几乎可以在训练集上的效果媲美。但在测试集中，准确率的下降幅度超过了 20%，已超出可以接受的范围。这种现象说明，模型可能是对验证集过拟合了，所以在验证集上的效果不能代表其真实的泛化能力。关于引起这个问题的原因，我有以下猜想，并思考了相应的解决办法，但由于时间和设备性能受限，未能一一验证。

a) 验证集过小。可以尝试更换验证集或增加验证集中的数据量。
b) 验证集与训练集同分布，但未与测试集同分布。遇到这种情况需要对各个数据集的分布进行检查，如果是，则需要更换验证集，甚至是更换训练集。
c) 验证集也参与了模型的训练。（通过对代码进行检查，排除了该原因）

综上所述，我认为在微博文本情感分析任务中，数据集的选择与处理非常重要，这一点在其他机器学习任务中也同样适用。