1、 项目主要进展（附已发表论文首页复印件）

在网购商品的过程中，难以衡量商品的真正品质，需要参考其他购买者的评价。商品购买者的评论在相当程度上能直接影响潜在购买者的购买意愿，但其本身质量良莠不齐，有价值的评价数量少，易混杂。为此，有必要对商品的评价进行分析汇总。目前，并不存在对商品评价信息自动化收集，处理的平台，消费者的购买存在着相当大的盲目性与不确定性。本项目致力于开发一个基于 NLP 技术的商品评价系统，具体功能包括商品评价信息收集，评价信息录入与处理，分析评价合理性并给出合理的综合评价，帮助用户筛选掉无用或不真实的评价，提供一个更有效、可靠的评价体系。

目前我们已经完成了如下功能：

1.数据集的建立

- **爬虫获取商品评价**

我们利用爬虫技术，爬取到了特定商品的相关评价信息，具体代码在 catch.py 文件中。在实现过程中遇到了反爬机制，采用了 headers 来反爬虫，具体的链接获取被封装在 Get_Url()函数中。

代码实现如下所示：

```
def GetInfo(num):
    # 循环获取每一页评论
    headers = {...}
    for i in range(num):
        # 头文件，没有头文件会返回错误的js
        params={...}
        # 解析JS文件内容
        content= requests.get(COMMENT_PAGE_URL[i], params,headers=headers).text[12:-1]
        comment = re.findall('rateContent":"(.*?)"', content)
        nk=re.findall('displayUserNick":"(.*?)"',content)
        #Id=re.compile('id:"(\d*?)"',re.S)
        #userId.extend(re.findall(Id,content))
        Id=re.findall(r'id":"(\d+)"',content)
        ratedate.extend(re.findall('"rateDate":"(.*?)"', content))
        #Id=re.findall(r'\"id\"\:\"[\d\.]*?\"', content)
        #print(Id)
        ratecontent.extend(comment)
        nickname.extend(nk)
        userId.extend(Id)
        print(len(userId))#此处数字还未爬取成功
    # auctionSku.extend(re.findall('"auctionSku":"(.*?)"', content))#商品类别，可以作为附加的功能
        print(ratedate)
```

- **数据库建立**

我们根据设计出的数据库关系图，利用 SQL 技术建立了相应的数据库，并把爬虫爬取的评价信息和商品信息等放入对应的数据库中。部分代码如下所示：

```python
def save_data():
    db = pymysql.connect(
        host="localhost",
        user="root",
        passwd="20010316",
        database="nlp_test")
    cursor = db.cursor()  # 创建一个游标对象
    # 插入对象
    sql="INSERT INTO comment(comment_id,goods_id,user_id,user_nk,content,comment_time) VALUES(%d,%d,%d,%s,%s,%s)"%
    #SQL="""INSERT INTO goods_item(goods_id,goods_name) VALUES(1,'红米手机')"""
    for every in range(0,len(ratecontent)):
        comment_id=every+1
        goods_id=1
        user_nk=nickname[every]
        content=ratecontent[every]
        comment_time=ratedate[every]
        value=(comment_id,goods_id,user_nk,content,comment_time)
        try:
            sql = "INSERT INTO comment(comment_id,goods_id,user_nk,content,comment_time) VALUES(%s,%s,%s,%s,%s)"
            cursor.execute(sql,value)
            db.commit()
            print("successful!")
        except:
            db.rollback()
    db.close()
```

数据库的结构及部分内容如下所示：

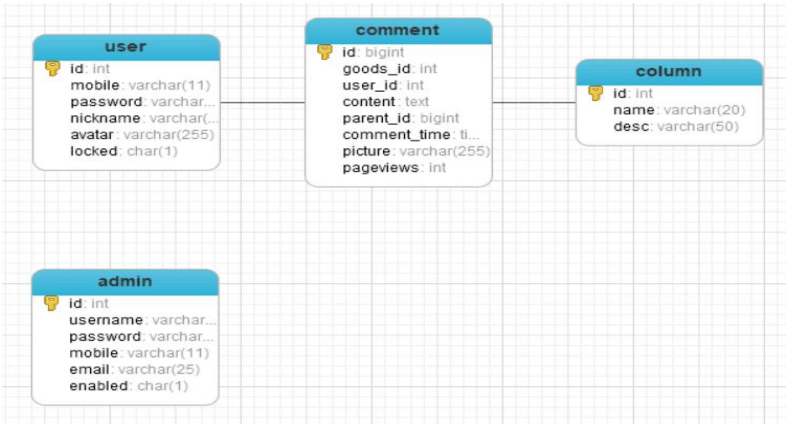

图 1.1 数据库关系图

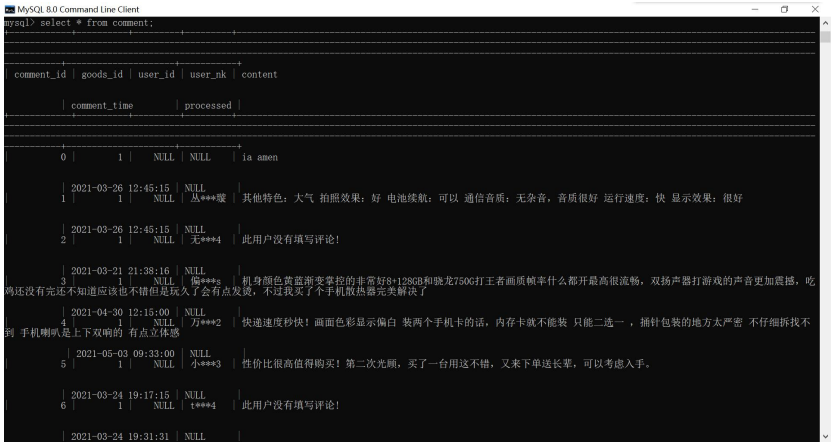

图 1.2 数据库评价信息存储

到此，我们实现了商品评价信息的数据集构建。

2.模型构建

基于本项目的数据特点，我们尝试了两种网络结构：GRU 与 BiLSTM。GRU 能够获得 87%左右的准确率，在实际样本的测试中，发现 GRU 的判定时常会出现失误，不能达到我们想要的效果。BiLSTM 的准确率稍低，多次测试后稳定在在 85%左右，能够较为准确地判定文本内容的情感，但模型的判断仍然存在不准确的情况。模型训练使用的优化器为 Adam，学习率设置为 0.001。

- **文本数据分词与进一步处理**

爬虫获取的文本信息中会夹杂着 Unicode 字符，Emoji 符号等需要处理的信息。首先利用 jieba 库进行分词处理，示例如图。随后去除标点符号，并去除停用词。其中停用词库来自哈尔滨工业大学的停用词库。

```
结巴分词测试
<generator object Tokenizer.cut at 0x000002739841E970>
精确模式输出：
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\Crimson\AppData\Local\Temp\jieba.cache
Loading model cost 0.693 seconds.
Prefix dict has been built successfully.
结巴，分词，测试
```

图 1.3 jieba 分词处理

对数据的处理是为了能够在使用模型预测是尽可能地减小可能带来的负面影响。初步处理的示例如图：

```
手链 看 很 精致 设计 感强 做工 好 款式 新颖 时尚 戴 非常 好看
非常 好看 送礼 很 不错 款 水星 手链 确实 挺 气质 值 入手
手链 款式 很 好看 纯银 亮亮 买 女神 节 礼物 满意
太 好看 款式 新颖致 做工 精细 很漂亮
手链 质量 好 设计 独特 戴 非常 好看 价格 实惠
手链 蛮 精致 天气 暖 戴 美美 哒
手链 很 细 小巧玲珑 很 精致 戴 显 气质
买 送 闺蜜 简单 精致 款式 大小 调节 很 方便 手里 分量 挺不错
```

图 1.4 分词处理结果

- **模型构建与训练**

构建模型，首先导入中文词向量库。考虑到评价文本的内容特点，我们选用了基于知乎问答数据的中文词向量库，同时训练的语料集选用了谭松波的酒店评论语料，以减少训练时可能产生的误差。训练的语料进行了处理，以便索引化。索引化的目的是使其能够与词向量相对应，考虑到评论文本的长度不一，尝试了两种方案：在末尾进行填充与裁剪长度。填充会消耗大量的资源，训练耗时极长，不适宜我们的训练环境；在对所有的评论长度进行了统计之后，我们取用了一个折衷的长度，经过统计计算能够涵盖 95.65%的语料，以此为最大索引长度。在文本的索引前部填充 0 值，使长度达到最大索引长度。

根据 Keras 的要求，准备一个(numwords,embedding-dim)的矩阵 Embedding Matrix，作为模型的第一层。代码如下：

```
model.add(Embedding(num_words,
                    embedding_dim,
                    weights=[embedding_matrix],
                    input_length=max_tokens,
                    trainable=False))
```

全连接层，其中激活函数选择 ReLU，在模型的训练中能取得较好的结果，代码如下：

```
model.add(Dense(1, activation='relu'))
optimizer = Adam(lr=1e-3)
```

LSTM 层与 BiLSTM 层，参数设置如图：

```
model.add(Bidirectional(LSTM(units=64, return_sequences=True)))
model.add((LSTM(units=16, return_sequences=False)))
```

导出模型结构，可训练参数共 196177。

```
_________________________________________________________________
___
Layer (type)                 Output Shape              Param #
=================================================================
===
embedding_1 (Embedding)      (None, 236, 300)          1500000
0
_________________________________________________________________
___
bidirectional_1 (Bidirection (None, 236, 128)          186880
_________________________________________________________________
___
lstm_3 (LSTM)                (None, 16)                9280
_________________________________________________________________
___
dense_1 (Dense)              (None, 1)                 17
=================================================================
===
Total params: 15,196,177
Trainable params: 196,177
Non-trainable params: 15,000,000
```

图 1.5 模型网络图

模型训练的输出如图：

```
Epoch 00021: ReduceLROnPlateau reducing learning rate to 1.000
0001518582595e-15.
Epoch 22/2000
26/26 [==============================] - 56s 2s/step - loss:
0.3167 - accuracy: 0.8843 - val_loss: 0.3257 - val_accuracy:
0.8750

Epoch 00022: val_loss did not improve from 0.32568

Epoch 00022: ReduceLROnPlateau reducing learning rate to 1.000
0001095066122e-16.
Epoch 23/2000
26/26 [==============================] - 55s 2s/step - loss:
0.3015 - accuracy: 0.8949 - val_loss: 0.3257 - val_accuracy:
0.8750

Epoch 00023: val_loss did not improve from 0.32568

Epoch 00023: ReduceLROnPlateau reducing learning rate to 1.000
0000830368326e-17.
Epoch 24/2000
26/26 [==============================] - 56s 2s/step - loss:
0.3199 - accuracy: 0.8830 - val_loss: 0.3257 - val_accuracy:
0.8750

Epoch 00024: val_loss did not improve from 0.32568

Epoch 00024: ReduceLROnPlateau reducing learning rate to 1.000
0000664932204e-18.
Epoch 00024: early stopping
```

图 1.6 训练结果示意

为了在节省计算时间的同时保持良好的整体性能，我们尝试对训练过程进行优化，经过试验后发现以下参数调整在提升效率方面较为有效：

1. **random_state**：对模型训练精度影响最大，经过多次的训练与测验，最终选择 random_state

为15，能够获得最好的精度。

2. **学习率：**学习率的设置影响着训练的速度与精度。我们选定学习率为0.001，取得了速率与精度的平衡。

- **训练结果**

```
运行内存6+128足够老年人用了 像素跟分辨率偏差，不过也对得起899这个价格了
是一例中性评价 output=0.51
机子轻薄，拿上手刚好，通话声音清晰，一键大字体，适合老年人用，给五星好评。
是一例正面评价 output=0.77
手机美观大气，上网畅通，触摸灵敏，特喜欢！
是一例正面评价 output=0.75
拍照效果：很清晰 电池续航：电池很够用，充电也快。 通信音质：没有杂音 运行速度：很快，一点都不卡。 显示效果：
是一例正面评价 output=0.70
已经收到了，非常不错的手机，首先物流很快，顺丰的两天就到了，十分给力，然后满怀期待的心情打开了包装袋，拿出来
是一例正面评价 output=0.73
没感觉 这个价位，流畅性已经很棒了。而且还是新机器，用个一年多卡了，被女儿摔烂了也不心疼再换就是。 还行 没有
是一例负面评价 output=0.19
可以还不错，买给老妈用的，速度还快，字体大，续航强，不错的一次购物
是一例正面评价 output=0.92
```

图1.7 模型评价展示

可以看出，模型大致能够对情感的极性进行分析，但是也可以看到模型的判别标准与现实语境有差异，造成了判断的不准确。进一步的优化考虑使用BERT，提高判别精确度。

- **训练环境概述**

模型训练基于Keras与Tensorflow，考虑到训练集仍然相当可观，为了节省时间，训练使用Colab完成。远程配置GPU为：NVIDIA Tesla T4。生成的模型文件大小为59.5MB。

3. web交互平台建立

我们使用js建立了一个模拟淘宝页面，用来展示模型对实际商品评价数据的评价效果。具体页面如下所示：

2、 下一步工作计划

由于项目开发时间有限，本系统在开发过程中还存在着一些不足。因此有待之后进行更深入的改进和开发。我们下一步的工作计划如下：

① **改进算法，以得到更高的准确率**

目前本系统的评价模型的准确率还不够高，因此后期还可以继续优化算法，提升模型准确率，在已有的评价效果上实现进一步提升。

② **完善前后端交互功能**

目前前端页面尚未完成，后期将继续完善前端的搭建，并将训练好的准确率高的模型部署到后端服务器上，实现前后端的交互。经过测试后，供用户使用。

③ **项目成果转化**

继续对项目进行开发，优化项目的目前所得成果，积极利用项目产出成果参加各种比赛；并计划根据项目成果去申请专利。

3、 存在问题、建议及需要说明的情况

存在问题：

目前的模型准确率还不够高，还需要寻找其他具有更好效果的模型；

前端开发速度较慢，页面功能尚未完全实现。