1. **实例分析**
   1. **实例简介**：

该智能舆情分析系统实例专注于汽车品牌管理，通过爬虫技术从百度有驾网站采集汽车评论数据，并对数据进行综合处理和分析。系统利用分词和 TF-IDF 算法从评论中提取关键词，并通过热度图和词云展示关键词的频率。通过 text-CNN 模型进行情感分析，系统能够评估各维度的消费者情绪，并通过时间序列分析监测市场趋势。整个实例为汽车品牌提供了全面的市场分析和管理策略支持。

* 1. 数据收集与预处理：
     1. 数据采集与去重：

数据采集与去重： 本实例通过爬虫技术从百度有驾平台采集汽车品牌评价数据。由于平台为了防止竞争性质的商业爬虫，限制了单分类只能读取760条评论，因此我们通过收集不同标签下不同排序方式的信息，再去重合并，尽量获取更多更全面的评价。

**数据采集代码**：

DESKTOP\_TAGS = [

'all', 'space', 'fuel', 'cruising', 'workmanship,internal,external',

'powersteering,power,control', 'comfort', 'cp', 'select'

]

def main(series\_ids, tags):

for series\_id in series\_ids:

all\_comments = []

for tag in tags:

# 爬取不同标签下的评论

...

**数据去重代码**：

import hashlib

def hash\_comment(comment):

return hashlib.md5(comment.encode('utf-8')).hexdigest()

def remove\_duplicates(input\_file, output\_file, backup\_folder):

...

for comment in comments:

comment\_text = comment['comment']

comment\_hash = hash\_comment(comment\_text)

if comment\_hash in seen\_hashes:

# 处理重复评论

...

**数据去重结果**：

* 宋PLUS新能源：
  + 总评论数：5195
  + 唯一评论数：2584
  + 重复评论数：2611
  + 重复但ID不同的评论数：27
  + 备份文件数：7
* 秦PLUS：
  + 总评论数：5848
  + 唯一评论数：2602
  + 重复评论数: 3246
  + 重复但ID不同的评论数: 30
  + 备份文件数: 14

去重完成后，去重后的评论分别保存到 output\desktop\_宋PLUS新能源.json 和 output\desktop\_秦PLUS.json。通过去重处理，确保数据的唯一性和准确性，提高了分析结果的可靠性。

* + 1. 评论角度分类：

在本实例中，我们对汽车评论进行角度分类，以便分析用户对不同方面的评价。分类过程如下：

1. **加载与保存JSON文件**：
   * 使用load\_json函数加载包含评论的JSON文件，使用save\_json函数保存分类结果，确保目录存在。
2. **切分评论**：
   * 评论切分是基于特定的分隔符，例如句号、感叹号、换行符等。切分函数split\_comment依据不同的特征进行句子分割，返回分割后的句子列表和分割方法的标识符。
   * 示例代码（省略部分细节）：

python

复制代码

def split\_comment(comment\_text):

sentences = None

if len(re.findall(r'[。 ！] ', comment\_text)) >= 3:

sentences = re.split(r'[。 ！] ', comment\_text)

split\_id = 1

elif len(re.findall(r'\n【', comment\_text)) >= 2:

sentences = re.split(r'\n【|\】', comment\_text)

split\_id = 2

# 其他切分方案省略

elif len(re.findall(r'(。|\.\.\.|\\n)|((?<![0-9a-zA-Z]) (?![0-9a-zA-Z]))|；|;', comment\_text)) >= 1:

sentences = re.split(r'(。|\.\.\.|\\n)|((?<![0-9a-zA-Z]) (?![0-9a-zA-Z]))|；|;', comment\_text)

split\_id = 6

else:

return -1, [comment\_text]

sentences = [sen.strip() for sen in sentences if sen and sen.strip() and count\_chinese\_char(sen) > 4]

return split\_id, sentences

1. **提取分类**：
   * 根据预定义的关键词表，将句子分类到不同的评价角度，如外观与内饰、操控与性能、舒适性等。提取分类的函数extract\_aspects会扫描句子中是否包含任何与某个角度相关的关键词。
   * 示例代码（省略部分细节）：

python

复制代码

def extract\_aspects(sentence):

aspect\_keywords = {

"操控与性能": ["刹车", "加速", "驾驶", "制动", "超车"],

"操控与性能": ["刹车", "加速", "驾驶", "制动", "超车", "操控", "动力", "手感", "性能", "起步", "转向", "打滑", "抓地", "速度", "启动", "马力", "底盘", "油门", "灵敏", "提速","过弯"],

"舒适性": ["舒适", "悬挂", "减震", "隔音", "安静", "静音", "平稳", "平顺", "气味", "异味", "体验", "舒服", "噪音", "震动", "颠", "风噪", "胎噪", "手感", "沉稳", "顺滑", "味道", "通风", "震感", "减速带", "偏硬", "异响", "声响", "塑料味", "声音"],

"空间": ["空间", "后备箱", "宽敞", "的个子", "有点小", "高度", "轴距"],

# 其他分类和关键词省略

}

aspects = []

for aspect, keywords in aspect\_keywords.items():

if any(keyword in sentence for keyword in keywords):

aspects.append(aspect)

return aspects

分类与统计

1. **分类评论**：
   * 使用classify\_comments函数对每条评论进行切分和分类。对切分后的每个句子，根据其包含的关键词确定其分类角度。
   * 对于无法切分或未能分类的评论，分别记录下来以便进一步分析。
   * 核心代码（省略部分细节）：

python

复制代码

def classify\_comments(comments):

classified\_comments = []

no\_split\_comments = []

# ... other definitions

for comment in comments:

comment\_text = comment['comment']

split\_id, sentences = split\_comment(comment\_text)

if split\_id != -1:

for index, sentence in enumerate(sentences):

aspects = extract\_aspects(sentence)

sen\_id = f"{comment['id']}\_{index}"

unique\_comments.append({

'sen\_id': sen\_id,

'car\_name': comment['car\_name'],

'sentence': sentence,

'aspects': ', '.join(aspects),

'split\_id': split\_id

})

if aspects:

for aspect in aspects:

classified\_comment = {

'aspect': aspect,

'sen\_id': sen\_id,

# ... other similiar fields

}

classified\_comments.append(classified\_comment)

aspect\_count[aspect] = aspect\_count.get(aspect, 0) + 1

else:

no\_aspect\_comments.append({

# ... similar fields

})

else:

no\_split\_comments.append({

'id': comment['id'],

'car\_name': comment['car\_name'],

'comment': comment\_text

})

分类结果：

**| 分类 | 秦PLUS | 宋PLUS 新能源 |**

**|-----------------------|------------------|------------------|**

**| 总评论数 | 2602 | 2584 |**

**| 总分条评论数 | 18649 | 17274 |**

**| 未成功切分的评论数 | 1 | 0 |**

**| 找不到分类的条评论数 | 606 (3.25%) | 745 (4.31%) |**

**| 总分类评论数 | 36572 | 34379 |**

**| 操控与性能 | 6891 | 6474 |**

**| 舒适性 | 6217 | 6080 |**

**| 外观与内饰 | 7235 | 7355 |**

**| 能耗与续航 | 3728 | 3011 |**

**| 空间 | 3816 | 3877 |**

**| 车机硬件与智能化 | 4087 | 3848 |**

**| 安全 | 689 | 734 |**

**| 性价比 | 3167 | 2273 |**

**| 品牌与服务 | 521 | 532 |**

**| 维保 | 221 | 195 |样例输出**

示例输出展示：

json

复制代码

[

{

"sen\_id": "356507\_2",

"car\_name": "2024款 荣耀版 DM-i 150km 旗舰PLUS",

"sentence": "这车的配置简直太智能了，每次用都觉得像在跟未来对话一样。那些高科技配置虽然我说不太清楚，但用起来真的特别方便。比如语音控制系统，我只要说句话就能控制好多功能，太省事了",

"aspects": "车机硬件与智能化",

"split\_id": 1

},

{

"sen\_id": "356507\_3",

"car\_name": "2024款 荣耀版 DM-i 150km 旗舰PLUS",

"sentence": "驾驶这辆车真的挺爽的，转向手感很好，油门响应也特别快，轻轻一点就有。制动也很稳，开起来心里特别踏实。就像有个经验丰富的舞伴带着你跳舞一样，轻松又愉快。",

"aspects": "操控与性能, 舒适性",

"split\_id": 1

},

{

"sen\_id": "354689\_0",

"car\_name": "2024款 荣耀版 DM-i 150km 旗舰PLUS",

"sentence": "这车的外观真的挺霸气的，大灯设计犀利，进气格栅也显得很大气。每次开出去都觉得很拉风。内饰方面，虽然有些地方用了塑料材质，但考虑到价格，我觉得也还可以接受。细节处理上如果能再提升一下就更好了，不过整体来说还是挺满意的",

"aspects": "外观与内饰, 性价比",

"split\_id": 1

},

{

"sen\_id": "354689\_1",

"car\_name": "2024款 荣耀版 DM-i 150km 旗舰PLUS",

"sentence": "宋PLUS新能源的空间真的给我留下了深刻印象。我身高172cm，体重140kg，坐进去感觉特别宽敞，头顶空间也很足，比我以前的车舒服多了。后座空间也很大，坐三个人也不会觉得拥挤。",

"aspects": "空间, 舒适性",

"split\_id": 1

}

]

通过上述过程，我们实现了对汽车评论的细化分类，为后续的热词分析和情感分析提供了基础。

经验证，未成功分类的条评论几乎全部为无效信息，如开场白、小标题、简单经历分享等等，可以忽略。样例：

json:

{

"id": "294028\_0",

"car\_name": "2023款 冠军版 DM-i 110KM 旗舰PLUS",

"comment": "🚗2023比亚迪宋DM-i，开启我的出行新篇章！🚗",

"split\_id": 3

},

{

"id": "294028\_1",

"car\_name": "2023款 冠军版 DM-i 110KM 旗舰PLUS",

"comment": "Hey小伙伴们！今天我要给大家分享一下我的新宠，2023款比亚迪送DM-i。 不管是选车、买车、用车还是玩车，我都要毫不保留地和你们聊聊我的亲身感受，一切都是真实的哦！✨",

"split\_id": 3

},

{

"id": "302141\_0",

"car\_name": "2021款 EV 旗舰型",

"comment": "【攒够买车的首付，买了宋PLUS，享受幸福时光😍😍",

"split\_id": 2

},

{

"id": "301684\_0",

"car\_name": "2021款 DM-i 51KM 尊荣型",

"comment": "一直以来家人外出都需要乘坐公交车或者是打车，可是一旦道路高峰期，每一项都很困难。便想着购买一台汽车，然而我一直都喜欢宋PLUS，老婆也推荐了零跑C11，后来对比之后，我还是更爱宋PULS，便购买了这款车。",

"split\_id": 3

}

* 1. 热点分析

在本实例中，我们使用文本处理和数据可视化技术对汽车评论进行热点分析。此分析通过提取评论中的高频关键词并生成词云图来直观展示用户对不同方面的关注点。以下是实现过程的详细步骤：

1. **生成 n-gram 关键词**：
   * 我们首先使用 jieba 库对评论文本进行分词，并通过自定义停用词和严格停用词进行过滤。
   * 使用 generate\_ngrams 函数生成 n-gram 关键词，并根据这些关键词计算每个词组的出现频率。
   * 停用词分为两种：严格停用词（strict\_stop），用于过滤掉无关紧要的词语；整体停用词（stopwords），用于去除无意义的高频词。

核心代码如下：

python

复制代码

def generate\_ngrams(text, n, stopwords, strict\_stop):

words = jieba.lcut(text)

words = [word.upper() for word in words]

words = [word for word in words if word not in stopwords]

ngrams = zip(\*[words[i:] for i in range(n)])

ngrams = [ngram for ngram in ngrams if not any(ssword in word for ssword in strict\_stop for word in ngram)]

ngrams = [''.join(ngram) for ngram in ngrams]

ngrams = [ngram for ngram in ngrams if ngram not in stopwords]

return ngrams

1. **计算关键词频率**：
   * 使用 extract\_keywords 函数从评论中提取高频关键词。该函数支持 n-gram 的生成，并通过 Counter 计算每个关键词的出现频率。
   * 为了确保关键词的有效性，我们对生成的 n-gram 关键词进行进一步过滤，去除无效字符。

核心代码如下：

python

复制代码

def extract\_keywords(texts, topK=20, ngram\_range=(1, 2, 3)):

combined\_text = ' '.join(texts)

keywords = []

stopwords = set([line.strip().upper() for line in open('src\\cn\_stopwords.txt', 'r', encoding='utf-8').readlines()])

custom\_stopwords = set([

"这款车", "这辆车", "开起来","这台车子","款车子", "比亚迪宋", "宋PLUSMINI", "宋PLUS", "PLUS新能源",

"燃油车", "-I", "不会觉得", "比较高", "比较喜欢",

"比较", "过程中", "后备箱空间", "后排空间", "乘坐空间", "储物空间", "动力方面","中控大屏", "式尾灯", "挺高","满意地方"

])

stopwords.update(custom\_stopwords)

strict\_stop = set(["比亚迪", "车", "【", "】", "0","I"])

for n in ngram\_range:

ngrams = generate\_ngrams(combined\_text, n, stopwords, strict\_stop)

ngrams = [ngram for ngram in ngrams if ngram.strip() and not any(c in ngram for c in ['\n', '\r', ' '])]

keywords += ngrams

keywords = [kw for kw in keywords if kw.strip() and not any(c in kw for c in ['\n', '\r', ' '])]

keyword\_counts = Counter(keywords)

most\_common\_keywords = keyword\_counts.most\_common(topK)

return most\_common\_keywords

1. **生成词云图**：
   * 使用 create\_wordcloud 函数生成词云图，以便直观展示评论中高频关键词的分布。
   * 我们使用汽车形状的遮罩图像，使得词云图更加符合汽车主题。

核心代码如下：

python

复制代码

def create\_wordcloud(keywords, image\_name, car\_image\_path):

car\_mask = np.array(Image.open(car\_image\_path))

wordcloud = WordCloud(font\_path='src\\simsun.ttc', background\_color='white', width=2560, height=1600, mask=car\_mask)

wordcloud.generate\_from\_frequencies(dict(keywords))

plt.figure(figsize=(12.8, 8))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.savefig(image\_name, dpi=200, bbox\_inches='tight')

plt.show()

#### 示例词云图

下面是基于汽车评论生成的词云图，通过这些图我们可以直观地看到用户关注的热点话题。



#### 总结

通过上述步骤，我们实现了对汽车评论的热点分析。这一分析不仅帮助我们提取了用户对不同方面的关注点，还通过词云图的方式直观地展示了这些关注点的分布。这对于了解用户需求、改进产品设计以及制定市场策略都具有重要意义。

* 1. 情感分析：

在本实例中，我们采用Text-CNN模型进行情感分析，该模型基于jieba分词和Embedding/Chinese-Word-Vectors知乎回答训练词向量。以下是详细的过程描述：

#### 7.4.1 数据预处理、分词和词向量

首先，我们需要对文本数据进行预处理，包括分词和生成词向量。jieba分词是一个常用的中文分词工具，能够将中文文本切分成一个个词语。我们使用jieba分词来处理我们的文本数据。以下是预处理、分词和词向量的具体步骤：

1. **数据预处理**：
   * 读取原始的文本数据，并将其转化为统一的格式。
   * 移除无关字符、空格等噪声数据。
2. **分词**：
   * 使用jieba库对文本数据进行分词，将句子分解成词语序列。
   * 示例代码：

python

复制代码

import jieba

def word\_cut(text):

text = regex.sub(' ', text)

return [word for word in jieba.cut(text) if word.strip()]

1. **词向量生成**：
   * 我们选择使用预训练的Chinese-Word-Vectors知乎回答训练的词向量，这些词向量是通过大规模知乎数据训练而成，能够捕捉到丰富的上下文信息。
   * 使用词向量将分词后的词语序列转化为向量序列，以便输入到模型中进行训练。

#### 7.4.2 Text-CNN模型

由于时间关系，我们选择了纯CNN模型进行情感分析，而没有使用结合BLSTM的CNN模型。结合BLSTM的模型可能效果会更好，但纯CNN模型已经足够满足当前任务的需求。以下是CNN模型的具体原理和参数设置：

1. **模型原理**：
   * Text-CNN模型通过卷积操作提取文本数据中的局部特征。模型中使用了多个不同尺寸的卷积核（filter sizes），以捕捉不同长度的n-gram特征。
   * 卷积层之后是一个最大池化层（Max-Pooling），用于提取重要的特征并降低特征维度。
   * 最后，通过全连接层（Fully Connected Layer）将特征映射到分类标签空间。
2. **模型参数**：
   * 滑动窗口和卷积层：
     + 滑动窗口大小（filter sizes）：2, 3, 4, 5, 6
     + 卷积核数量（filter num）：150
     + 多通道（multichannel）：静态和非静态词向量结合使用，以便模型能够根据训练数据微调词向量，从而更好地适应汽车话题。
   * 训练参数：
     + 使用Python命令运行训练脚本，参数设置如下：

bash

复制代码

python main.py -static=true -non-static=true -multichannel=true -device 0 -pretrained-name sgns.zhihu.bigram -filter-num 150 -filter-sizes 2,3,4,5,6 -early-stopping 1000

* + - 预训练词向量选择知乎bigram，以提高上下文能力。

1. **数据集**：
   * 我们使用了车评相关数据集进行训练从而针对性地提高模型对车评的情感分析能力。数据集含有6300条测试数据及56714条训练数据。示例如下：

label text

0 1 外观确实非常霸气，钻石切工有棱有角，3.0的动力在城市里绰绰有余，内饰考究，空间比较大，bose的音响非常给力，小众品牌不像德系三架马车那样成为街车，为个性代言。

1 1 外观漂亮，安全性佳，动力够强，油耗够低

2 1 后备箱大！！！

3 1 空间大。外观大气，中控台用料讲究简洁

4 1 外观漂亮，空间够大，动力家用也ok

#### 7.4.3 预测方法

在训练过程中，我们保存了训练参数（args）、模型参数（best\_model）和词汇表（text\_field\_vocab）。为方便模型的加载和使用，我们封装了一个预测类Predictor，该类负责加载模型、词汇表和训练参数，并提供预测方法。以下是该类的主要功能和实现概述：

##### 1. 加载模型和词汇表

Predictor 类在初始化时加载训练好的模型参数、词汇表和训练参数，并设置相应的设备（CPU或CUDA）。

python

复制代码

class Predictor:

def \_\_init\_\_(self, model\_path='saved/best\_model.pt', args\_path='saved/args.pkl', vocab\_path='saved/text\_field\_vocab.pkl', device=None):

print('Loading model...')

self.device = device

with open(args\_path, 'rb') as f:

self.args = pickle.load(f)

if(self.device):

self.args.device = self.device

with open(vocab\_path, 'rb') as f:

text\_field\_vocab = pickle.load(f)

self.text\_field = data.Field(lower=True, tokenize=dataset.word\_cut)

self.text\_field.vocab = text\_field\_vocab

if self.args.static:

self.args.embedding\_dim = self.text\_field.vocab.vectors.size()[-1]

self.args.vectors = self.text\_field.vocab.vectors

self.model = MD.TextCNN(self.args)

self.model.load\_state\_dict(torch.load(model\_path))

if self.args.cuda:

torch.cuda.set\_device(self.args.device)

self.model = self.model.cuda()

print('Model loaded')

该类初始化时：

* 加载训练参数和词汇表。
* 根据训练参数和词汇表配置模型。
* 加载训练好的模型参数，并将模型移动到指定设备上（如CUDA）。

##### 2. 预测方法

Predictor 类提供 predictstr 方法，该方法接收一系列句子，并返回每个句子的情感预测结果。

python

复制代码

class Predictor:

# ... (上文的初始化代码)

def predictstr(self, senlist):

self.model.eval()

processed\_texts = [self.text\_field.preprocess(text) for text in senlist]

tokenized\_texts = self.text\_field.process(processed\_texts).to(self.args.device).t\_()

predictions = []

with torch.no\_grad():

logits = self.model(tokenized\_texts)

probabilities = F.softmax(logits, dim=-1)

predicted\_labels = torch.max(probabilities, 1)[1]

predictions.extend(predicted\_labels.cpu().numpy())

return predictions

该方法具体步骤如下：

1. **预处理**：
   * 使用 self.text\_field.preprocess 方法对输入的每个句子进行预处理，包括分词、去除标点等操作。
   * 示例代码：

python

复制代码

processed\_texts = [self.text\_field.preprocess(text) for text in senlist]

* + preprocess 将输入文本转化为词语列表，便于后续处理。

1. **词向量化**：
   * 将预处理后的文本转换为词向量，并转移到指定设备（如GPU）。
   * 示例代码：

python

复制代码

tokenized\_texts = self.text\_field.process(processed\_texts).to(self.args.device).t\_()

* + process 方法将词语列表转换为数值化的词向量序列。

1. **模型预测**：
   * 将词向量输入模型，计算输出的logits。
   * 使用Softmax函数将logits转化为概率分布。
   * 通过 torch.max 方法选择最大概率对应的标签作为预测结果。
   * 对应代码：

python

复制代码

with torch.no\_grad():

logits = self.model(tokenized\_texts)

probabilities = F.softmax(logits, dim=-1)

predicted\_labels = torch.max(probabilities, 1)[1]

* + logits 是模型的原始输出值。
  + softmax 将logits转化为概率分布，确保输出的值在0到1之间，且总和为1。
  + torch.max 从概率分布中选取最大值对应的索引，即预测的标签。

1. **结果收集**：
   * 将预测结果收集并转换为CPU格式的NumPy数组。
   * 对应代码：

python

复制代码

predictions.extend(predicted\_labels.cpu().numpy())

* + 该步骤将GPU上的张量转换为CPU上的NumPy数组，便于后续处理。

##### 3. 情感分析函数

analyze\_sentiments 函数用于分批次对条评论进行情感分析，并返回带有情感标签的评论列表。该函数利用 Predictor 类中的 predictstr 方法分批处理评论，并在处理过程中显示进度条。

python

复制代码

def analyze\_sentiments(predictor, comments, batch\_size=64):

sentiments = []

total\_batches = (len(comments) + batch\_size - 1) // batch\_size

for i in tqdm(range(total\_batches), desc="Analyzing sentiments", unit="batch"):

batch\_comments = comments[i \* batch\_size:(i + 1) \* batch\_size]

sentences = [comment['sentence'] for comment in batch\_comments]

batch\_sentiments = predictor.predictstr(sentences)

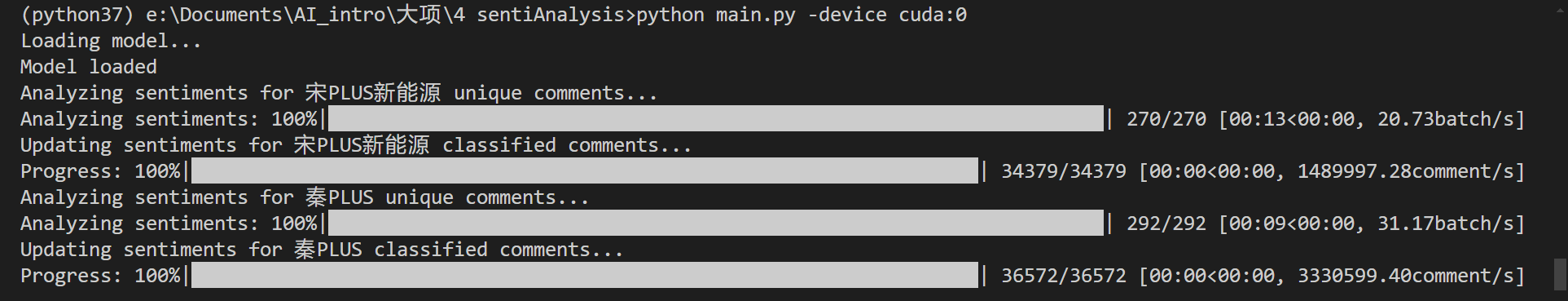
sentiments.extend(batch\_sentiments)

for comment, sentiment in zip(batch\_comments, batch\_sentiments):

comment['sentiment'] = "正面" if sentiment == 1 else "负面"

return comments

#### 7.4.4 运行结果



分析成功，得到分类情绪判断，部分实例如下：

json

{

"sen\_id": "296256\_1",

"car\_name": "2023款 冠军版 EV 605KM 旗舰PLUS",

"comment": "方向盘有点轻，感觉开下来没有太好的手感。",

"aspect": "舒适性",

"split\_id": 3,

"sentiment": "负面"

},

{

"sen\_id": "295971\_1",

"car\_name": "2023款 冠军版 EV 520KM 豪华型",

"comment": "后备箱的空间不大，只能稍微放点小东西，装载能力不太好。",

"aspect": "空间",

"split\_id": 3,

"sentiment": "负面"

},

{

"sen\_id": "139026\_4",

"car\_name": "2021款 DM-i 110KM 旗舰PLUS",

"comment": "动力很强，尤其是在起步的瞬间。美中不足的就是操控很糟糕",

"aspect": "操控与性能",

"split\_id": 1,

"sentiment": "正面"

},

{

"sen\_id": "139026\_5",

"car\_name": "2021款 DM-i 110KM 旗舰PLUS",

"comment": "辅助配置非常齐全，360全景也很清晰好用。",

"aspect": "车机硬件与智能化",

"split\_id": 1,

"sentiment": "正面"

},

#### 7.4.5总结

在情感分析过程中，我们首先对评论文本进行了预处理和分词，并生成了词向量。这些词向量使用了预训练的Chinese-Word-Vectors知乎回答训练模型，以捕捉丰富的上下文信息。接下来，我们训练了一个Text-CNN模型，通过卷积操作提取文本的局部特征，并使用最大池化层和全连接层进行情感分类。训练过程中，我们使用了多通道（multichannel）的词向量设置，以便模型能够根据训练数据微调词向量，从而更好地适应汽车话题。

为了方便模型的加载和使用，我们封装了一个预测类Predictor，该类负责加载训练好的模型参数、词汇表和训练参数，并提供预测方法。通过 Predictor 类，我们能够对输入的评论进行情感预测，并将预测结果添加到评论数据中。整个情感分析过程通过批处理和进度条显示，确保高效和透明。

通过上述步骤，我们实现了对汽车评论的情感分析，为后续的数据展示和综合分析提供了坚实的基础。

### 7.5 综合报告（GPT生成）：【待填充】

* + **基于时间线的可视化**：通过时间线图表展示评论中关键词和情感倾向的变化，直观显示舆情的动态变化。
  + **综合分析方法**：将提取的热词、评论情感取向及时间线数据喂给GPT等大模型，生成更详细的分析报告，提供更全面的舆情洞察。