摘要

本文主要对已给出的酒店以及景区的 csv 文件进行分析，通过数据的预处理，汇总，引入用户自定义词典，分词等数据处理方法对游客的评论进行处理，建立模型来对酒店和景区进行综合评价，对网络评论的有效性进行分析以及酒店及景区的特色的分析。

针对问题一，为了得出评论热词与热度，根据附件 1 中酒店评论与景区评论数据，利用 jieba 库对每条评论进行分词操作，利用 collections 模块的 Counter 类进行统计，得出出现频率最高的二十个词语和词频，构成印象词云表。

针对问题二，根据评论数据，我们建立综合评分模型。将评论分为了长评与短评，并设不同的权重，利用 snownlp 库直接对短评进行情感分析。对长评，分别对五个模块（服务、位置、设施、卫生、性价比）设定了筛选词，取出筛选词周围的词语构成短句。分别统计五个模块的短句，利用 snownlp 库进行情感分析并求出平均值，得到每个景区或酒店长评的模块化情感得分。根据附件 2 中的酒店评分与景区评分中的专家评分数据的分数构成，对长评得分和短评得分进行修正，之后结合专家评分数据，计算出线上评论的综合得分。利用均方误差模型将修正后的分数与专家评分进行比对，以检验综合评分模型的模型质量。

针对问题三，本文建立了评论有效性分析模型对附件 1 中的酒店评论与景区评论的评论内容进行有效性分析。对虚假评论进行排查后，统计出每条评论的分词数与分词中包含筛选词的词数，设立评分制度进行评分。考虑到部分评论存在成分冗余的情形（如同一词语的多次连续出现），我们对有大量重复内容的评论采取按重复程度降低总分的措施，以得到合理的有效性评分。

针对问题四，本文利用 collections 模块的 Counter 类按照各个酒店和景区统计问题一中的分词结果，得出每个酒店和景区的评论中出现频率最高的二十个词语和词频，构成每个酒店和景区独立的印象词云表。之后我们选取了专家评分中总得分处于高、中、低三个层次的每个层次各 3 家景区和 3 家酒店，根据其各自的印象词云表筛选出每家景区和酒店的优势和特色。

关键字：数据挖掘　分词　情感分析　模糊评价　优势筛选

一、 问题重述

1.1 基本情况

提升景区及酒店等旅游目的地美誉度是各地文旅主管部门和旅游相关企业非常重视和关注的工作。这些工作涉及到如何稳定客源、取得竞争优势、吸引游客到访消费等重要事项。游客满意度与目的地美誉度紧密相关，游客满意度越高，目的地美誉度就越大。因此掌握目的地游客满意度的影响因素，切实提高游客满意度、最终提升目的地美誉度，不仅能够保证客源稳定，而且对于旅游企业科学监管、资源优化配置以及市场持续开拓具有长远而积极的作用。而这一切都可以从对景区和酒店的评论的研究得出一些结论。

1.2 问题的提出

1. 景区及酒店印象分析依据附件 1 中景区及酒店网评文本，计算出目的地 TOP20 热门词，给出每家酒店和景区的印象词云表，按照要求的结构保存为 csv 文件。

2. 景区及酒店的综合评价根据附件 1 景区及酒店网评文本及附件 2 景区及酒店得分建立合理的数学模型及相应算法，按满分为 5 分对景区及酒店的服务、位置、设施、卫生、性价比五个方面进行评分，并按照均方误差（Mean Squared Error,MSE）进行模型评价。

3. 网评文本的有效性分析出于各种原因，网络评论常常出现内容不相关、简单复制修改和无有效内容等现象，妨碍了游客从网络评论中获得有价值的信息，也为各网络平台的运营工作带来了挑战。从文本分析的角度，建立合理的模型，对景区及酒店网络评论的有效性进行分析。

4. 景区及酒店的特色分析旅游业繁荣发展给游客带来了选择困难的问题，评分接近的景区或酒店很难根据评分进行取舍。建立合理的模型和算法从景区及酒店的网评文本中挖掘他们各自的特色和亮点，以吸引游客提升竞争优势。选择综合评价高、中、低三个层次的各 3 家景点和 3 家酒店，结合模型的结果，分析他们各自的特色。

二、 模型的假设

- 假设评论数据的真实性、有效性均有待考究；
- 专家评分只具有一定的可靠性；

三、 符号说明

符号	意义
M_{il}	i 酒店或景区的第 l 个板块长评论的打分
m_{ilh}	第 i 个酒店或景区的长评中第 l 个板块的第 h 个相关的分词模块
MM_{il}	第一次修正后 i 酒店或景区的第 l 个版块长评论的打分
N_{il}	i 酒店或景区 l 板块的综合评价模型打分
T_{il}	短评论打分
D_{il}	专家对 i 酒店或景区 l 板块的打分
X	某条评论的分词数目
Y	某条评论的筛选词的个数
Z	某条评论出现 3 次以上的词语数目

四、 模型建立、求解与数据分析

4.1 问题一：景区及酒店印象分析

我们使用 python 中的 jieba 库对附件一和附件二每一条评论遍历后进行分词，统计每个目的地的 TOP20 热词，制作出了各个目的地的印象词云表。下表为景区 A01 的印象词云表

评论热词	热度
玩	1650
好	1527
取票	1165
动物	1114
不错	947
表演	805
很多	786
好玩	728
马戏	679
开心	618
值得	559
带	558
动物园	557
景区	550
项目	516
排队	512
孩子	510
过山车	504
真的	493
刺激	488

图 1　景区 A01 的印象词云表

4.2 问题二：景区及酒店的综合评价

4.2.1 模型建立与求解

以景区和酒店的评论为基础，我们建立综合评分模型对酒店和景区进行综合评价方式如下：

（1）因为单纯用 jieba 库进行分词副词和形容词会分开，为了把程度副词和形容词放到一起，如“很”与“舒适”连接成“很舒适”我们首先自定义词典，其中包含了绝大多数常用的由程度副词和形容词组成的短语，之后再进行分词操作。分词结果展示如图 2：

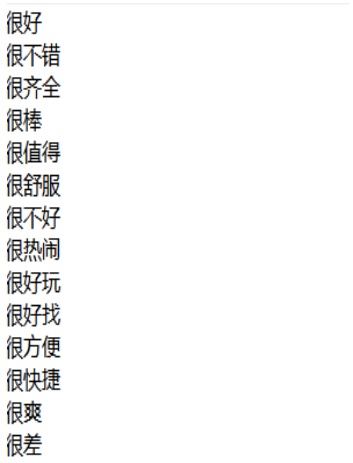

图 2　自定义词典结合 jieba 库分词结果展示

（2）因为长评论细节更多，描述更为详解，且传达信息更多，我们认为长评论和短评论的重要程度不同。所以我们将词语数量 <6 的评论记为短评论，其余的记为长评论。

（3）对于短评论，由于其有效性较小，我们直接将其放入程序中进行情感打分，对于服务、位置、设施、卫生、性价比这五个板块都直接取用此情感打分得到的分值。

情感打分分数越高表明受众的满意程度高，越低说明满意程度低。

（4）对于长评论，其有效性较高，我们需要对各条评论分模块打分取平均值从而得到更精确的评估。

我们针对五个评分板块分别设定了筛选词表，如下：

服务	'服务','工作人员','态度','前台','人员','接待','早餐','取票','购票'
位置	'位置','地理','市中心','郊区','机场','火车','高铁','公交车','地铁','交通','商场','商区','商圈','取票','开车','驾车','自驾
设施	'设施','床','枕头','位置','灯','厕所','马桶','淋浴','配套','电压','wifi','网络','插座','洗漱用具','拖鞋','公共厕所','公厕','停车场','电视机','电视','场馆'
卫生	'卫生','发霉','发黄','脏','不干净','环境','干净','整洁','清爽','空气','清新','消毒','口罩'
性价比	'性价比','实惠','超值','经济','物美价廉','小贵','负担不起','便宜','原价','打折','折扣','不值','贵'

图 3　五个评分板块筛选词表

当程序识别到筛选词时，会将筛选词及其前后的两个分词之后的词语共五个词或组成短句，将其归类到该筛选词对应的评分板块中。

如评论“离东站近，位置很好。相对老一点的五星级酒店。”中如果以“位置”作为定位，“”东站“近”，“位置””很好“”相对“”就被划分到一个模块中，作为位置板块情感打分依据的一部分

从而我们将所有长评论进行了分割并模块化，对于每个模块内的短句用情感分析法分别进行评分。该酒店的服务、位置、设施、卫生、性价比这五个板块的得分由相关对应小模块的情感分析的平均值求得，公式如下：

$$M_{il} = \frac{\sum_1^n m_{ilh}}{n} \tag{1}$$

M_{il} 为 i 酒店或景区的第 l 个板块长评论的打分 m_{ilh} 为第 i 个酒店或景区的长评中第 l 个板块的第 h 个相关的分词模块,n 为一个酒店某个板块的模块总数。

这样之后我们有了每个酒店和景区的情感分析的打分，为了让所打分数利于后续与专家得分进行比较，我们需要将每个酒店或景区的五个模块的短评和长评的情感分析得分（原本介于 0 1 之间的得分），映射到 1 5 之间。

$$MM_{il} = M_{il} * 5 + 1 \tag{2}$$

MM_{il} 为第一次修正后 i 酒店或景区的第 l 个版块长评论的打分

$$N_{il} = T_{il} * 0.2 + MM_{il} * 0.4 + D_{il} * 0.4 \tag{3}$$

其中 N_{il} 为 i 酒店或景区 l 板块的综合评价打分，T_{il} 为短评论打分（五个板块均取情感分析的结果，均相等），D_{il} 为专家对 i 酒店或景区 l 板块的打分

但是在处理数据的过程中我们发现，我们的模型对各板块的评分标准要比专家的评分标准更为严厉，即对于同一水平的表现，模型打分比专家打分更低；因此我们对模型打分进行修正，将模型打分乘以一定的比例使其打分区间与专家打分一致。具体做法如下：

$$D = \frac{MT - MT(min)}{MT(max) - MT(min)} * (D(max) - D(min)) + D(min) \tag{4}$$

MT 为某个酒店或景区的某个板块的评分，D（max）为专家在 50 个酒店中对该模块的打分最大值，D（min）为最小值。对每一个酒店或景区的长评论和短评论均需做此修正修正之后，再带入（5）最终得到我们的综合评分模型打分。

之后我们利用均方误差（MSE）对我们建立的综合模型与专家评分对比检验模型质

酒店名称	服务得分	位置得分	设施得分	卫生得分	性价比得分
H01	4.505152792	4.735036294	4.390031143	4.499162171	4.251833646
H02	4.661784658	4.77177111	4.54157197	4.431981192	4.106674293
H03	4.6415366	4.747652389	4.667196459	4.747290374	4.54
H04	4.716566327	4.761243582	4.662609013	4.70074061	4.37543857
H05	4.516563521	4.661022787	4.524735036	4.607562778	4.452957818
H06	4.664500669	4.393492976	4.531092748	4.759174316	4.519450276
H07	4.347986147	4.685236081	4.628370167	4.564210524	4.386058052
H08	4.677689497	4.601541344	4.572703844	4.758553883	4.388445913

图 4　酒店最终打分结果

景区名称	服务得分	位置得分	设施得分	卫生得分	性价比得分
A01	3.988740568	4.660281406	4.692011287	4.642629408	4.435381341
A02	4.232704129	4.642959817	4.512125426	4.65712133	4.333394104
A03	3.910778946	4.08	4.04	4.462414513	4.327033208
A04	3.974503777	4.319735221	4.286508929	4.49290674	4.411733535
A05	4.040248805	4.450822208	4.301446361	4.408559261	4.395259824
A06	4.308793183	4.4144538	4.458865371	4.732578002	4.275487169
A07	4.297073825	4.664113036	4.234264131	4.899797658	4.360663502
A08	3.96546697	4.454243333	4.757566753	4.184861065	4.314991483

图 5　景区最终打分结果

量:

$$\boldsymbol{MSE} = \frac{\boldsymbol{SSE}}{\boldsymbol{n}} = \frac{1}{n}\sum_{1}^{n} \boldsymbol{w_i}(\boldsymbol{y_i} - \widehat{\boldsymbol{y_i}}) \tag{5}$$

其中 n 为景区或酒店的总数，w_i 设置为 1，y_i 为某酒店的某个板块的得分，$\widehat{y_i}$ 为专家对该酒店该板块的打分。

酒店和景区的均方误差结果如下:

	服务	位置	设施	卫生	性价比
MSE	0.04515	0.04566	0.03521	0.03234	0.11245

图 6　酒店误差

	服务	位置	设施	卫生	性价比
MSE	0.05994	0.10333	0.05727	0.05111	0.04293

图 7　景区误差

4.2.2 模型结果分析

由图 6 和图 7 的结果，我们可做如下分析：

相比于酒店，景区的范围较大，环境较为多样，可变因素多，对于不同的游客个体可能因为游玩时的路线、天气、人流量、服务人员班次等因素产生差别较大的游玩体验，从而在评论中产生较大的差异。而专家在对景区各个板块进行评分时只能根据固定的条件进行静态判断，并不能考虑这些额外的因素，从而也就导致模型根据游客评论给出的最终得分与专家评分有相对更大的差距。相比之下，酒店最终得分与专家评分的差距很小。

酒店最终得分中性价比板块的得分与专家评分相差很大，我们认为这是因为对于经济条件不同的游客而言，判断同一酒店的性价比标准是因人而异的：经济富裕的游客对于高价格的接受度更高，即使是价格偏高的酒店，只要其服务、位置等其他条件达标，他们会认为这是值得的；相对而言，经济水平处于中等及以下的游客会希望以更少的费用换取更好的入住体验，对于性价比的评判会更加苛刻。而专家对于性价比板块的评分是有固定标准的，这也就导致了相对误差较大的结果。因为专家评分为不考虑动态因素的评分，在考虑各类动态因素对游客评分的影响的前提下，我们认为 MSE 的结果体现出模型的质量是较高的。

4.3 问题三：网评文本的有效性分析

4.3.1 模型建立

由于网络评论中常常出现内容不相关，简单复制修改和无效内容等现象，为了使评论的获取更加有效，我们建立评论有效性分析模型。

（1）首先，我们认为长评论和短评论的有效性不同。我们结合 jieba 库设立自定义词典，分词后得到整个评论的分词数目 X

（2）其次，为了对内容的相关性进行评估，根据图 3 的筛选词表，我们遍历后统计每条评论的筛选词的个数 Y

（3）由于评论中会出现词语反复出现，造成无效，我们统计一条评论中出现 3 次以上的词语数目 Z

（4）设定如下打分表

分次数	筛选词出现数字	得分
1	0	1
2~3	1	2
4~5	2	3
>=6	3	4
>=10	>=4	5

图 8　有效性分析模型打分表

评论有效性分析模型计算式如下:

$$S = (0.5X + 0.5Y) * 0.8^{Z} \tag{6}$$

S 为该条评论的有效性得分

4.3.2 求解结果

下面列出部分 H01 酒店评论的有效性分析模型打分

H01	2020-01-14	**酒店非常好	标准客房	['酒店', '非常好']	1.5
H01	2020-01-14	帮朋友订的，应该还不错	标准客房	['帮', '朋友', '订', '不错']	2
H01	2020-01-14	服务一流的	标准客房	['服务', '一流']	2
H01	2020-01-14	酒店不错服务很好不错	标准客房	['酒店', '不错', '服务', '很好', '不错']	2.5
H01	2020-01-15	酒店房间干净卫生，前台韩女士服务非常好，非常满意。	标准客房	['酒店', '房间', '干净', '卫生', '前台', '韩女士', '服务', '非常好', '非常', '满意']	4.5
H01	2020-01-15	停车场停车方便，酒店地理位置优越，出行方便，酒店房间整洁干净，服务员服务意识良好，态度良好，值得入住。	标准客房	['停车场', '停车', '酒店', '地理位置', '优越', '出行', '酒店', '房间', '整洁', '干净', '服务员', '服务', '意识', '态度', '值得', '入住']	5

图 9　有效性分析模型打分表

4.4 问题四：景区及酒店特色分析

4.4.1 模型建立

（1）将专家评分中总得分处于高、中、低三个层次的每个层次各 3 家景区和 3 家酒店筛选出来

（2）对（1）筛选出来的结果，利用 collections 模块的 Counter 类按照各个酒店和景区统计问题一中的分词结果，得出每个酒店和景区的评论中出现频率最高的二十个词语和词频。（3）根据（2）中词频对选出来的 9 个酒店 9 个景区进行优势亮点分析。

4.4.2 模型结果

热词频率统计结果如下：

服务相关词	服务相关词热度	位置相关词	位置相关词热度	设施相关词	设施相关词热度	卫生相关词	卫生相关词热度	性价比相关词	性价比相关词热度
服务	274	位置	133	位置	152	环境	88	性价比	61
很好	150	交通	116	酒店	48	干净	81	高	33
不错	117	便利	68	很好	45	卫生	59	很高	14
前台	90	很好	54	设施	37	不错	43	价格	13
早餐	71	酒店	43	好	27	很好	37	酒店	12
酒店	51	服务	37	便利	26	服务	33	交通	9
好	48	好	30	不错	23	房间	32	实惠	8
房间	24	地铁	30	很方便	21	很	26	好	7
交通	23	很方便	29	服务	20	酒店	23	不错	7
位置	23	东站	29	交通	14	好	20	很好	7
很	23	不错	22	房间	14	交通	16	很	6
环境	21	靠近	15	出行	12	整洁	10	很方便	5
干净	16	房间	14	很不错	9	舒适	9	超高	5
卫生	16	出行	13	东站	8	位置	8	期间	5
热情	16	早餐	12	老	8	非常好	7	推荐	4
很不错	14	卫生	12	非常好	8	服务态度	7	便宜	4
接待	14	商场	12	优越	8	便利	6	贵	4
非常	11	环境	11	很	8	安静	6	出行	4
服务态度	11	很	11	齐全	8	设施	6	环境	4
非常好	11	干净	11	床	8	地理位置	6	服务	4

图 10　H01 酒店热词频率分析结果

酒店优势特色分析结果：

高评价酒店	H04	服务很好、工作人员热情，位于高速路口很好找、交通很方便，酒店设施齐全，有温泉，环境干净卫生，提供自助餐早餐
	H06	服务很好、工作人员热情，位于高速出口、很好找，在流溪河附近、吃饭方便、交通很方便，酒店设施齐全，有温泉，环境干净卫生，提供自助餐早餐，性价比高。
	H07	服务非常好，位置在珠江旁边、有停车场，设施齐全、环境舒适整洁，有广式早茶。

图 11　高评分酒店分析结果

中评价酒店	H08	服务很好，位于市中心，交通便利、有停车场，房间很大，环境干净整洁，性价比超高。
	H09	服务很好，位置很方便，酒店有私人沙滩、环境优美，房间干净卫生，泳池很不错。
	H14	服务很好，位置很方便，酒店有私人沙滩、环境优美，房间干净卫生，泳池很不错。

图 12　中评分酒店分析结果

较低评价酒店	H48	服务很好，位于市中心、交通便利，设施配套齐全，环境干净整洁、很安静，性价比高。
	H50	服务非常好，位于市中心，交通便利、可乘坐大巴，有停车场，房间干净整洁，是五星级。
	H38	酒店位置在镇里，就在海边，房间种类多，有海景房、海鲜好吃，有沙滩、游泳池，有停车场，性价比高。

图 13　较低评分酒店分析结果

景区优势特色分析如下:

高评价景区	A39	公园类景区，取票购票方便，适合带老人孩子，位于郊区空气好，交通方便，门票便宜性价比高。
	A38	温泉类景区，服务很周到，交通非常方便，配套设施齐全，环境优雅，水质干净。
	A34	园林类景区，网上购票很方便，工作人员态度热情，交通非常方便，园内环境心旷神怡，配套设施齐全，园内干净优雅、古色古香。

图 14　高评分景区分析结果

中评价景区	A34	公园类景区，在汕头网评第一名，身份证购票，交通很方便，有水鸭，环境干净卫生，入园佩戴口罩。
	A28	岛屿类景区，取票很方便，岛上有酒店，配套设施齐全，海滩旁边有停车场，岛上可以租车，海水、沙滩干净卫生，价格便宜可以买套票。
	A14	自然类景区，网络购票，取票很方便，交通方便、有停车场，景区内有地下河，风景优美、空气清新。

图 15　中评分景区分析结果

低评价景区	A04	游乐类景区，不用取票可刷身份证进入景点，交通方便可乘地铁，设施很多刺激好玩，环境干净卫生。
	A27	海滩类景区，网络购票、取票方便，工作人员服务态度好，交通方便，配套设施齐全有停车场，海水干净海滩空气好，晚上非常浪漫。
	A11	A11：自然类景区，网上购票，景点有停车场，风景很美、空气清新。 自然类景区，网上购票，景点有停车场，风景很美、空气清新。

图 16　较低评分景区分析结果

参考文献

[1] 唐塞丽，仙树，胡蕾，刘猛，代坤，面向在线产品评论数据的有效性建模与测度研究 [J] 中国运载火箭技术研究院研究发展中心，北京１００７６；中国航天员科研训练中心，北京１０００９４；江西师范大学计算机信息工程学院，南昌３３００２２

[2] 秦欣，基于专家评价与在线评论的影评网站评价方法研究 [J], 毕业论文，东北大学工商管理学院管理科学与工程，2015.7

附录 A　分词–python 源程序

```
import pandas as pd
import jieba
from collections import Counter
jieba.load_userdict('不分开的词典.txt')
data = pd.read_excel('酒店评论.xlsx', encoding='utf-8')
stopwords = pd.read_csv('StopwordsCN.txt', encoding='GBK', names=['stopword'], index_col=False)
stop_list = stopwords['stopword'].tolist()
stop_list.append(' ')
stop_list.remove('很')
stop_list.remove('非常')
stop_list.remove('十分')
data['cut'] = data['评论内容'].apply(lambda x : [i for i in jieba.cut(x) if i not in stop_list])
b = ['服务','位置','设施','卫生','性价比']
e = ['fw','wz','ss','ws','xjb']
def choose_keyword(i):
  keyword = []
  if i == 0:
     keyword = ['服务','工作人员','态度','前台','人员','接待','早餐','取票','购票']
  elif i == 1:
     keyword =
         ['位置','地理','市中心','郊区','机场','火车','高铁','公交车','地铁','交通','商场','商区','商圈','取票','
  elif i == 2:
     keyword =
         ['设施','床','枕头','位置','灯','厕所','马桶','淋浴','配套','电压','wifi','网络','插座','洗漱用具','拖鞋
  elif i == 3:
     keyword =
         ['卫生','发霉','发黄','脏','不干净','环境','干净','整洁','清爽','空气','清新','消毒','口罩']
  elif i == 4:
     keyword =
         ['性价比','实惠','超值','经济','物美价廉','小贵','负担不起','便宜','原价','打折','折扣','不值','贵']
  return keyword
cleanword = ['服务','工作人员','态度','前台','人员','接待','早餐','取票','购票',
          '设施','床','枕头','位置','灯','厕所','马桶','淋浴','配套','电压','wifi','网络','插座',
          '洗漱用具','拖鞋','公共厕所','公厕','停车场','电视机','电视','场馆','性价比']
for i in range (1,51):
   if i < 10:
      dfi = data[data['酒店名称'] == ('H0' + str(i))].reset_index(drop = True)
   else:
      dfi = data[data['酒店名称'] == ('H' + str(i))].reset_index(drop = True)
   df = pd.DataFrame()
   dfshort = pd.DataFrame()
   for k in range(0, 5):
      dfi[b[k]] = ''
      dfi[b[k]+'short'] = ''
```

```
        e = choose_keyword(k)
        for j in range(0, len(dfi['cut'])):
            if len(dfi['cut'][j])<6:
                dfi[b[k]+'short'][j] = dfi['cut'][j]
            for p in range(0,len(e)):
                if e[p] in dfi['cut'][j]:
                    if dfi['cut'][j].index(e[p])<2:
                        if dfi['cut'][j].index(e[p])+3>len(dfi['cut'][j]):
                            dfi[b[k]][j] = dfi['cut'][j][:]
                        else:dfi[b[k]][j] = dfi['cut'][j][:dfi['cut'][j].index(e[p])+3]
                    elif dfi['cut'][j].index(e[p])+3>len(dfi['cut'][j]):
                        dfi[b[k]][j] = dfi['cut'][j][dfi['cut'][j].index(e[p]) - 2:]
                    else:dfi[b[k]][j] = dfi['cut'][j][dfi['cut'][j].index(e[p]) -
                         2:dfi['cut'][j].index(e[p]) + 3]
            # else:data[b[k]][j] = ['']
        # if b[k] not in dfi.columns.tolist():
        #    break
        words = []
        for content in dfi[b[k]]:
            words.extend(content)
        counter = Counter(words)
        # for l in range(0,len(cleanword)):
        #    if cleanword[l] in counter:
        #       del(counter[cleanword[l]])
        wordsshort = []
        for contentshort in dfi[b[k]+'short']:
            wordsshort.extend(contentshort)
        countershort = Counter(wordsshort)
        d = pd.DataFrame(countershort.most_common(20), columns=[b[k] + '相关词', b[k] +
            '相关词热度'])
        dfshort = pd.concat([dfshort, d], axis=1)
        c = pd.DataFrame(counter.most_common(20),columns = [b[k]+'相关词',b[k]+'相关词热度'])
        df = pd.concat([df,c],axis=1)
    if i < 10:
        df.to_excel('酒店excel/long/' + 'H0' + str(i) + '.xlsx', index=False)
        dfshort.to_excel('酒店excel/short/' + 'H0' + str(i) + '.xlsx', index=False)
    else:
        df.to_excel('酒店excel/long/' + 'H' + str(i) + '.xlsx',index=False)
        dfshort.to_excel('酒店excel/short/' + 'H' + str(i) + '.xlsx', index=False)
```

附录 B　综合评分模型–python 源代码

```
import pandas as pd
import numpy as np
import jieba
```

```
from snownlp import SnowNLP
jieba.load_userdict('不分开的词典.txt')
data = pd.read_excel('酒店评论.xlsx', encoding='utf-8')
pscore = pd.read_excel('酒店评分.xlsx',encoding = 'utf-8')
stopwords = pd.read_csv('StopwordsCN.txt', encoding='GBK', names=['stopword'], index_col=False)
stop_list = stopwords['stopword'].tolist()
stop_list.append(' ')
stop_list.remove('很')
stop_list.remove('非常')
stop_list.remove('十分')
data['cut'] = data['评论内容'].apply(lambda x : [i for i in jieba.cut(x) if i not in stop_list])
b = ['服务','位置','设施','卫生','性价比']
e = ['fw','wz','ss','ws','xjb']
ns = pd.DataFrame()
def choose_keyword(i):
   keyword = []
   if i == 0:
      keyword = ['服务','工作人员','态度','前台','人员','接待','早餐','取票','购票']
   elif i == 1:
      keyword =
          ['位置','地理','市中心','郊区','机场','火车','高铁','公交车','地铁','交通','商场','商区','商圈','取票','
   elif i == 2:
      keyword =
          ['设施','床','枕头','位置','灯','厕所','马桶','淋浴','配套','电压','wifi','网络','插座','洗漱用具','拖鞋
   elif i == 3:
      keyword =
          ['卫生','发霉','发黄','脏','不干净','环境','干净','整洁','清爽','空气','清新','消毒','口罩']
   elif i == 4:
      keyword =
          ['性价比','实惠','超值','经济','物美价廉','小贵','负担不起','便宜','原价','打折','折扣','不值','贵']
   return keyword
cleanword = ['服务','工作人员','态度','前台','人员','接待','早餐','取票','购票',
            '设施','床','枕头','位置','灯','厕所','马桶','淋浴','配套','电压','wifi','网络','插座',
            '洗漱用具','拖鞋','公共厕所','公厕','停车场','电视机','电视','场馆','性价比']

for i in range (1,51):
    if i < 10:
        dfi = data[data['酒店名称'] == ('H0' + str(i))].reset_index(drop = True)
        # newpscore = pscore[pscore['景区名称'] == ('A0' + str(i))]
    else:
        dfi = data[data['酒店名称'] == ('H' + str(i))].reset_index(drop = True)
        # newpscore = pscore[pscore['景区名称'] == ('A' + str(i))]
    score = []
    df = pd.DataFrame()
    dfshort = pd.DataFrame()
    shortemotion = []
    emotion = []
```

```
    for k in range(0, 5):
        classifiedemotion = []
        dfi[b[k]] = ''
        e = choose_keyword(k)
        for j in range(0, len(dfi['cut'])):
            if len(dfi['cut'][j]) in range(1,6):
                if k == 0:
                    shortemotion.append(SnowNLP(''.join(dfi['cut'][j])).sentiments)
            else:
                for p in range(0, len(e)):
                    if e[p] in dfi['cut'][j]:
                        if dfi['cut'][j].index(e[p]) < 2:
                            if dfi['cut'][j].index(e[p]) + 3 > len(dfi['cut'][j]):
                                dfi[b[k]][j] = dfi['cut'][j][:]
                            else:
                                dfi[b[k]][j] = dfi['cut'][j][:dfi['cut'][j].index(e[p]) + 3]
                        elif dfi['cut'][j].index(e[p]) + 3 > len(dfi['cut'][j]):
                            dfi[b[k]][j] = dfi['cut'][j][dfi['cut'][j].index(e[p]) - 2:]
                        else:
                            dfi[b[k]][j] = dfi['cut'][j][dfi['cut'][j].index(e[p]) -
                                2:dfi['cut'][j].index(e[p]) + 3]
                        if e[p] in cleanword:
                            del (dfi[b[k]][j][dfi[b[k]][j].index(e[p])])
                        classifiedemotion.append(SnowNLP(''.join(dfi[b[k]][j])).sentiments)
        emotion.append(np.mean(classifiedemotion))
    for o in range(0,5):
         score.append(float((np.mean(shortemotion)*4+1)*0.2 + (emotion[o]*4+1)*0.4) )
    a = np.array(score)
    score = a.reshape(a.shape[0], 1).T
    s = pd.DataFrame(score, columns=['服务得分', '位置得分', '设施得分', '卫生得分',
        '性价比得分'])
    ns = pd.concat([s, ns], axis=0)
ns = ns.reset_index()
ns['index'] = pscore['酒店名称']
ns.rename(columns = {"index": "酒店名称"}, inplace=True)
ns['服务得分'] = ((ns['服务得分'] - min(ns['服务得分'])) / (max(ns['服务得分']) -
    min(ns['服务得分'])) * (4.9 - 3.8) + 3.8)* 0.6 + pscore['服务得分']*0.4
ns['位置得分'] = ((ns['位置得分'] - min(ns['位置得分'])) / (max(ns['位置得分']) -
    min(ns['位置得分'])) * (4.9 - 4.0) + 4.0)*0.6 + pscore['位置得分']*0.4
ns['设施得分'] = ((ns['设施得分'] - min(ns['设施得分'])) / (max(ns['设施得分']) -
    min(ns['设施得分'])) * (4.9 - 4.0) + 4.0)*0.6 + pscore['设施得分']*0.4
ns['卫生得分'] = ((ns['卫生得分'] - min(ns['卫生得分'])) / (max(ns['卫生得分']) -
    min(ns['卫生得分'])) * (4.9 - 4.0) + 4.0)*0.6 + pscore['卫生得分']*0.4
ns['性价比得分'] = ((ns['性价比得分'] - min(ns['性价比得分']))/ (max(ns['性价比得分']) -
    min(ns['性价比得分'])) * (4.9 - 4.0) + 4.0)*0.6 + pscore['性价比得分']*0.4
ns.to_excel('酒店最终得分.xlsx',index = False)
```

附录 C　均方误差–python 源程序

```
from sklearn.metrics import mean_squared_error
import pandas as pd
owns = pd.read_excel('酒店最终得分.xlsx')
pscore = pd.read_excel('酒店评分.xlsx',encoding = 'utf-8')
# '服务得分', '位置得分', '设施得分', '卫生得分', '性价比得分'
print([mean_squared_error(owns['服务得分'],pscore['服务得分']),
mean_squared_error(owns['位置得分'],pscore['位置得分']),
mean_squared_error(owns['设施得分'],pscore['设施得分']),
mean_squared_error(owns['卫生得分'],pscore['卫生得分']),
mean_squared_error(owns['性价比得分'],pscore['性价比得分'])])
```

附录 D　评论有效性分析–python 源程序

```
import pandas as pd
import jieba
from collections import Counter
jieba.load_userdict('不分开的词典.txt')
data = pd.read_excel('景区评论.xlsx', encoding='utf-8')
pscore = pd.read_excel('景区评分.xlsx',encoding = 'utf-8')
stopwords = pd.read_csv('StopwordsCN.txt', encoding='GBK', names=['stopword'], index_col=False)
stop_list = stopwords['stopword'].tolist()
stop_list.append(' ')
stop_list.remove('很')
stop_list.remove('非常')
stop_list.remove('十分')
data['cut'] = data['评论内容'].apply(lambda x : [i for i in jieba.cut(x) if i not in stop_list])

keyword = ['服务','工作人员','态度','前台','人员','接待','早餐','取票','购票','位置',
           '地理','市中心','郊区','机场','火车','高铁','公交车','地铁','交通','商场',
           '商区','商圈','取票','开车','驾车','自驾','设施','床','枕头','位置','灯',
           '厕所','马桶','淋浴','配套','电压','wifi','网络','插座','洗漱用具','拖鞋',
           '公共厕所','公厕','停车场','电视机','电视','场馆','卫生','发霉','发黄','脏',
           '不干净','环境','干净','整洁','清爽','空气','清新','消毒','口罩','性价比',
           '实惠','超值','经济','物美价廉','小贵','负担不起','便宜','原价','打折','折扣','不值','贵']
wordlength = []
wordtimes = []
wordrepeat = []
for i in range (1,2):
    if i < 10:
        # dfi = data[data['酒店名称'] == ('H0' + str(i))].reset_index(drop = True)
        dfi = data[data['景区名称'] == ('A0' + str(i))].reset_index(drop=True)
        # newpscore = pscore[pscore['景区名称'] == ('A0' + str(i))]
```

```
    else:
        dfi = data[data['酒店名称'] == ('H' + str(i))].reset_index(drop = True)
        # newpscore = pscore[pscore['景区名称'] == ('A' + str(i))]
    for j in range(0, len(dfi['cut'])):
        a = 0
        b = 0
        words = []
        for content in dfi['cut'][j]:
            words.extend(content)
            if content in keyword:
                a = a + 1
        counter = Counter(words)
        for value in counter.values():
            if int(value) >= 3:
                b = b + 1
        wordrepeat.append(b)
        wordlength.append(len(dfi['cut'][j]))
        wordtimes.append(a)
def wordlength_score():
    wordlength_score = []
    for i in range (0,len(wordlength)):
        if wordlength[i] == 0:
            wordlength_score.append(0)
        elif wordlength[i] == 1 :
            wordlength_score.append(1)
        elif wordlength[i] <= 3 :
            wordlength_score.append(2)
        elif wordlength[i] <= 5:
            wordlength_score.append(3)
        elif wordlength[i] <= 10:
            wordlength_score.append(4)
        else:
            wordlength_score.append(5)
    return wordlength_score
def wordtimes_score():
    wordtimes_score = []
    for i in range (0,len(wordtimes)):
        if wordtimes[i] == 0:
            wordtimes_score.append(1)
        elif wordtimes[i] == 1 :
            wordtimes_score.append(2)
        elif wordlength[i] == 2 :
            wordtimes_score.append(3)
        elif wordlength[i] == 3:
            wordtimes_score.append(4)
        else:
            wordtimes_score.append(5)
```

```
    return wordtimes_score
a = wordlength_score()
b = wordtimes_score()
c = wordrepeat
d=[]
for i in range(len(a)):
   d.append(round(float((a[i]*0.5+b[i]*0.5)*0.8**c[i]),1))
# dfi = data[data['酒店名称'] == ('H0' + str(1))].reset_index(drop = True)
dfi = data[data['景区名称'] == ('A0' + str(1))].reset_index(drop = True)
dfd = pd.DataFrame(d)
dfi = pd.concat([dfi,dfd],axis = 1)
# print(dfi.columns)
dfi.columns = ['景区名称','评论日期','评论内容','分词结果','评论有效性得分']
# dfi.columns = ['酒店名称','评论日期','评论内容','分词结果','入住房型','评论有效性得分']
# dfi.to_excel('H01有效性得分.xlsx',index = False)
dfi.to_excel('A01有效性得分.xlsx',index = False)
```