# CityLab：基于熵权法和关联度分析的“宜居城市”满意度模型

## 摘要

建设国际一流的和谐宜居之都是北京的发展战略目标，目前围绕此目标展开的各项工作在积极推进中，北京发生了令人鼓舞的深刻变化。这些努力是否转化为广大人民对城市满意度的提升，北京居民是否感受到了和谐宜居水平的变化，是对现有成果的检验和影响城市进一步发展方向的关键因素。

本文构建了一个满意度综合指数来反映居民对城市的主观感受。首先，我们利用 Python 软件对附件中给出的评论进行筛选，然后判断有意义的评论中包含情感的正负向和强烈程度，并进行赋值。评论情感得分范围为-5~5，-5 代表最强烈的负向情感，0 代表中立，5 代表最强烈的正向情感。我们得到 166930 条有意义的评论，66293 条负向评论、699 条中立评论，99938 条正向评论。然后我们根据每条评论的关键词将其分类，得到 8 个衡量满意度的指标，接下来我们对每个指标进行赋权，用以衡量指标对于综合满意度的影响程度，最后用 8 个指标的加权平均表示居民综合满意度。通过计算得出 2015 年—2018 年的居民综合满意度分别为 0.94，0.84，0.93，0.98，由此可见主观满意度均为正值，在 2016 年明显下降后逐年回升，在 2018 年达到新巅峰，但是数值仍较小，北京建设和谐宜居城市仍有较大的可提升空间。此外，我们通过分析各指标的权重大小，得到对主观满意度的影响的关键因素为：住房条件，社会公平，科技文化，交通出行，环境健康。

我们整理了 2015 年—2018 年的部分可能影响综合满意度的客观数据，利用灰色系统模型中的灰色关联度分析，并加以改进，研究客观数据和综合满意度之间的关联性。结果显示，经济富裕度、城市生态环境和社会保障水平和综合满意度之间有较强的联系。对此我们提出五条建议，即优化经济结构，大力发展城市经济；改善生态环境，走可持续发展道路；完善社保制度，福利惠及更多民众；引入高端人才，有计划的发展科技；维持公共设施，适当转移部分资金。

最后，我们向北京市政府写了一份信，信中提出了我们对北京建设宜居城市的发展战略和规划的建议。

我们的模型客观合理，具有很好的推广性，能够广泛应用于全国其他城市中，对于现实生活中的全国各城市建设文明和谐宜居的环境有着一定的参考和指导意义。

本文的详细代码见附录。

**关键词：**多属性决策、自然语言处理、熵权法、灰色系统模型、关联度分析

# 1 问题重述

## 1.1 背景

建设和谐宜居城市，社会、经济、文化、环境和谐发展，满足居民物质和精神生活的双重需求，是对当前城市发展的要求和目标。北京作为现代化的国际大都市，不仅其城市发展路径对全国其他城市具有示范作用，更代表着我们伟大祖国的形象，是向全世界展示中国的首要窗口，因此北京和谐宜居之都的建设更是备受关注。目前，社会各界已经为建设北京成为宜居之都投入了大量努力，取得的一系列有目共睹的成绩，但究竟这些努力是否转换为居民的切身居住感受的提升有待考究。在城市发展建设中首先要考虑人的因素 ,不能只见物不见人,必须把人的需求和根本利益作为城市发展建设的第一需要。因此，了解居民主观感受，能够进一步检验建设工作成果，帮助城市建设者及时调整工作方向，为接下来的城市战略规划指明方向。

## 1.2 问题提出

在本文的研究中，构建居民主观感受评价体系是我们的研究重点，针对以上背景和附件数据，我们需要解决以下问题：

（1） 筛选出与和谐宜居相关的评论，对评论包含的情感的方向及程度进行合理赋值，数值化居民感受；

（2） 根据评论中蕴含的情感数值，建立模型计算反映居民对城市“和谐宜居”的满意度综合指数，分析影响居民对“和谐宜居”主观感受的关键因素；

（3） 查找北京市 2015-2018 年的经济、环境、基础设施建设等客观数据，构建模型衡量客观现实与主观感受之间的关联；

（4） 分析客观现实对居民主观感受的影响，为提高民众对和谐宜居的主观感受，对北京市城市建设规划提出合理有效的建议。

# 2 模型假设

为了使得问题更易于理解，我们作出以下合理假设：

（1） 居民在网络上的评论反映了对于城市“和谐宜居”情况的真实情感。由于居民在社交平台中有言论自由的权利且处于匿名状态，所以他们可以不受限制地发表对城市的观点看法，里面包含了他们的真实主观感受。

（2） 居民在城市不同方面的情感相互独立，互不相关。在面对不同的方面，如住房、交通、环境等，居民对一个方面的情感态度仅取决于他对这个方面的看法，不会影响对其他方面的情感态度，也不会受到其他方面的干扰。

# 3 变量说明

本文建立模型的过程中主要涉及以下变量，变量及说明如下:

表 1 变量以及含义

| 变量 | 含义 |
|---|---|
| $score$ | 每条评论的原始情感打分 |
| $N$ | 每一指标下的评论条数，引入的下标详见模型中的具体描述，下同 |
| $E$ | 每一指标下的评论期望情感打分 |
| $S$ | 每一指标下的评论情感打分标准差 |
| $Y$ | 标准化处理后的矩阵，其中每一个元素记为 $y$ |
| $P$ | （评论期望情感得分的）比重 |
| $H$ | （评论期望情感得分的）熵值 |
| $w$ | （评论期望情感得分的）权重 |
| $f$ | 每一指标的综合权重 |
| $F$ | 居民主观满意度综合得分 |
| $k$ | 每个次级客观指标数据向量，引入的下标详见模型中的具体描述，下同 |
| $K$ | 均值化处理后的向量，其中每一个元素记为 K(j)，向量中每一个元素的表示方法下同 |
| $K_0$ | 参考列（综合满意度向量） |
| $K_0$ | 比较列（各次级各管指标数据的向量） |
| $\triangle_{0,i}$ | 比较列与参考列的绝对差值向量 |
| $\triangle_{max(0,i)}$ | 同一比较列与参考列的绝对差值向量的最大值 |
| $\triangle_{min(0,i)}$ | 同一比较列与参考列的绝对差值向量的最小值 |
| $\triangle_{max}$ | 上述各比较列得到绝对差值向量最大值中的最大值 |
| $\triangle_{min}$ | 上述各比较列得到绝对差值向量最小值中的最小值 |
| $\eta_{0,i}$ | 关联系数（比较列与参考列的关联程度向量） |
| $\rho$ | 分辨系数 |
| $r_{0,i}$ | 关联度（关联系数的平均值） |

# 4. 模型建立与求解

## 4.1 问题一的数据处理

题目要求为评论包含的情感打分，并且分数涉及情感的正负向和强烈程度两方面。附件中的数据给出了某社交网络上 2015 年到 2018 年带有与和谐宜居相关的关键词的评论。我们根据附件中的评论内容，使用 Python 对评进行清洗和赋值。

### 4.1.1 数据清洗

首先对于众多的评论信息数据，需要对数据进行清洗整理。我们使用 python 软件对数据做了以下处理。

①预处理，删除评论中无关成分。

部分评论中出现“@XX 用户”以及表情包的字样，这对下面步骤中判断评论是否有意义和是否与主题相关产生一定的干扰，所以我们首先进行预处理，将评论中与情感表达无关字眼进行删除，保留评论的主干部分。

②使用中文停用词词库对评论关键词进行筛查，删除无意义的词汇组合评论。

评论的关键词可以很大程度上概括评论的意思，所以如果一个句子的关键词包含“然后”、“换句话说”之类的停用词，说明该句子本身是无意义的。首先，我们对每条评论内容进行关键词提取。关键词的提取结果和表中所给的分词结果几乎没有差别，因此我们最终选择表中的分词结果作为我们的评论关键词。然后，我们将每条评论的关键词和中文停用词库进行匹配，如果发现关键词中有停用词，则舍弃掉这条评论。

③构建主题关键词词典，与评论分词进行匹配，删除和当前主题无关的评论。

我们通过整理关键词一栏的数据，将所有评论分为 43 个对应的主题。我们需要衡量每条评论和对应主题的相关程度，这对于我们来说是十分困难的，并没有现成的算法或是机器学习的包可以解决这个问题。我们考虑了两种可能的解决办法：一是，用词向量的语义特征进行聚类。但这是一个计算量十分昂贵的工作，在短时间的建模比赛中并不可行。二是，训练主题相关性的分类器，它需要大量有标签的主题相关语料和主题不相关语料，遗憾的是，题目所给的数据中并没有进行标注。因此，我们提出了一种基于关键词词频统计的概率模型。首先，我们统计每一个主题下出现的所有评论分词，计算它们的词频，按照从高到低的顺序对它们进行排序，取排序在前 20 的词汇（不足 20 个关键词则取全部）作为这个主题的关键词词典。然后我们用和①中同样的方法，对相关主题的评论分词结果和关键词词典进行匹配。若关键词词典中的所有关键词都无法与评论匹配，即评论的分词均不出现在其对应的主题关键词词典中，则丢弃这条评论。这个方法有两点局限性：一是，因为中文词频统计无法全部涵盖有用评论中的词汇表，因此会误删一些有用的评论。我们对前 100 条评论进行了人工筛查，并对比机器筛查的结果，发现有用评论丢失了 5 条。二是，因为中文表达的变化多样，会造成一些误判，包括误删有用的评论和把无用的评论错判为有用的评论。同样我们对前 100 条评论进行对比，发现错判了 3 条无用的评论。附件中提供了 306179 条评

论，由于样本量巨大，且出错率较低，我们筛选数据方法导致的误差对最终结果估计的影响可以忽略不计，本着节约时间和运算合理的原则，建立主题关键词词典是删除与主题无关评论的最佳选择。

### 4.1.2 情感分析和赋值

分析语句中的情感正负向和强烈程度是一个自然语言处理领域的热门研究问题，存在很多优秀的算法和运行良好的机器学习包。这里我们使用了 snownlp，因为它具有强大的中文自然语言处理的能力。snownlp 提供了默认的情感分析的训练语料库，是对一些电商产品的评价，但是这个语料库并没有涵盖题目数据所给出的对于“和谐宜居城市”的评论，因此我们在这个预训练的模型上，用已经清洗过的数据进行迁移学习。以下是具体方法：

①情感正负向分类

虽然预训练的模型可能不能很准确地进行情感分级，但是对于消极和积极情感的识别分类是可靠的。我们通过预训练的模型对评论进行情感偏向的机器标注，正向情感的评论被记录在 pos.txt 中；负向情感的评论被记录在 neg.txt 中。

②建立情感分析模型

我们构建朴素贝叶斯模型进行情感分析。将-5 定义为负向最强烈的情感，将 5 定义为正向最强烈的情感，将 0 定义为中立情感，即-5~0 代表负向情感强度逐渐递减的过程，0 代表中立态度，0~5 代表正向情感强度逐渐递增的过程。我们用得到的正负语料库训练二分类的朴素贝叶斯模型，最后用模型预测得到的后验概率输出作为情感情感强烈程度的量化指标。

具体的过程如下：对于正类(正向情感)p 和负类(负向情感)n，通过一些 NLP 特征提取的方法可以得到特征$w_1, w_2, \cdots, w_n$，这些特征之间是相互独立的。那么对于负类 n 来说，其输出的后验概率为：

$$\mathrm{P}\left(\mathrm{n}\middle|w_1, \cdots，\ w_n\right) = \frac{P\left(w_1, \cdots，\ w_n\middle|n\right)P(n)}{P\left(w_1, \cdots，\ w_n\middle|n\right)P(n) + P\left(w_1, \cdots，\ w_n\middle|p\right)P(p)}$$

我们将网络的权重记录在 my_sentiment.marshal.3 中。

③打分赋值

我们用训练好的情感分析模型，对有用评论重新进行情感打分，并将原数据表的部分数据和打分数据一起记录在 sentiment_analysis.xlsx 中。

### 4.1.3 数据整理结果

经过数据清洗和情感打分，我们最终筛选出有意义数据 166930 条，具体的得分结果见下表。从表中我们可以看出，2015 年有意义的评论数目相对较少，但也有 14000+，样本容量足够大。每一年的评论情感正负向分布相对较为均衡，正向评论略多于负向评论，平均来看正向情感强度略强于负向情感。

表 2 情感得分统计性描述

| 情感得分 | 2015 年 | | 2016 年 | | 2017 年 | | 2018 年 | | 总计 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 数量 | 均值 | 数量 | 均值 | 数量 | 均值 | 数量 | 均值 | 数量 | 均值 |
| -5~0（负向） | 6554 | -3.117 | 19275 | -2.963 | 24297 | -2.960 | 16167 | -2.997 | 66293 | -2.985 |
| 0（中立） | 43 | 0 | 121 | 0 | 375 | 0 | 160 | 0 | 699 | 0 |
| 0~5（正向） | 7587 | 3.243 | 30189 | 3.239 | 38474 | 3.264 | 23688 | 3.269 | 99938 | 3.256 |
| 总计 | 14184 | 0.298 | 49585 | 0.820 | 63146 | 0.850 | 40015 | 0.724 | 166930 | 0.764 |

## 4.2 问题二的模型建立与求解

题目要求我们构建一个反映居民对城市“和谐宜居”的满意度综合指数。我们将 Borda 计数法，首先根据相对应的关键词将评论进行分组，确定衡量“和谐宜居”的主观指标，再计算出每个指标的情感平均得分和权重，最后对所有指标得分进行加权平均，最终得到满意度综合指数

### 4.2.1 模型的构建

（1）对衡量“和谐宜居”的关键词进行分类，构建评价指标。

首先，我们对评论情感分析得到的数据进行统计特征的提取，将具有相同特征的评论归为一组。注意到附件表格中第一列一共有 43 个关键词，其中的一些关键词在描述城市“和谐宜居”情况属性的时候具有较强的相关性，如“北京出行”、“北京通勤”、“北京交通状况”可概括为交通出行方面的指标。因此我们对第一列关键词进行了人为的预分类，将原来的 43 个关键词划分成了 8 个大类指标（交通出行、住房条件、基础设施、休闲娱乐、公共安全、环境健康、科技文化、社会公平）。

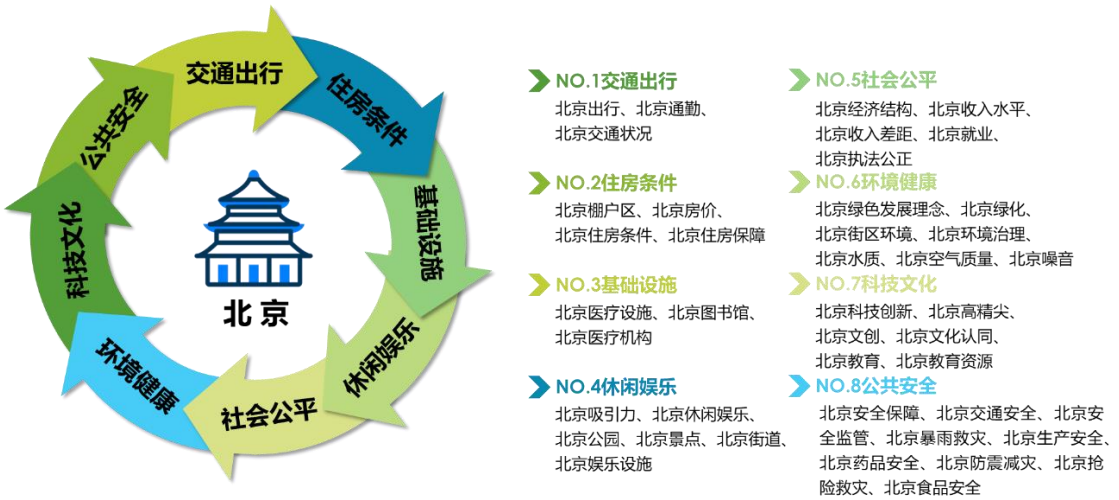


图 1 “和谐宜居”指标分类

事实上，我们可以通过对这些指标进行主成分分析来分类，但由于本题数据量巨大，主成分分析的方法比较昂贵。此外，这种方法进行分类的标准是基于对评论情感打分计算的，可能存在应归于同一大类的关键词对应的评论情感得分无

线性相关，或应归于不同大类的关键词对应的评论情感得分线性相关的情况，例如，经我们检验发现，与关键词“北京出行”和“北京交通状况”相对应的评论的情感得分表现出线性不相关，通过主成分分析的方法会将“北京出行”和“北京交通状况”归为不同类，这样的分类结果不准确。考虑到分类结果的准确性和关键词数量不大，我们抛弃了主成分分析的方法而进行人为分类，同时我们假设划分至不同大类的关键词之间的相关性可以忽略。

（2）计算各类指标得分数学期望。

由于有效评论条数足够多，通过大数定理，我们认为可将各类指标下每条评论得分的均值近似为各类指标得分的数学期望。我们用 $E$ 表示数学期望，即

$$E_j = \frac{1}{N}\sum_{i=1}^{N} score_{ji}$$

其中，N 为在第 j 个指标下对应的评论的条数，score 代表每条评论的得分。

（3）使用改进的熵权法对各个指标进行赋权。

基于评论情感得分数据对人们满意度进行综合评价，需解决的主要问题是确定八类指标分别在综合满意度中的贡献权重。赋权的方法主要包括主观赋权法和客观赋权法，基于主观赋权法如专家匿名反馈（Delphi）法和层次分析（AHP）法给出的参数和赋值者个人感情倾向紧密相关，不具有客观性和通用性，在本题复杂的综合评价系统中并不适用，故我们选择使用客观赋权法。客观赋权法包含熵权法、CRITIC 方法及标准离差法等，根据数据的分布情况来赋权。我们已经根据不同关键词将它们进行分类，构造出 8 个评价指标，不同评价指标可认为在现实中没有关联，其数字上的线性相关性不具有实际意义可以忽略，因此我们选择使用熵权法的来对每项指标赋权。

我们可以计算出每一个指标包含的评论条数 $N$，评论情感得分的期望值 $E$，以及评论情感的分的标准差 $S$。情感打分的区间在（-5,5）之间，其符号表示情感为正面或负面，值的大小表示情感的强烈程度。综合考虑获得的单位步长时间内评论数目及总时间步数，我们将评论的具体时间（评论时间 2015 年 7 月开始至 2018 年 10 月共 40 个月）分为 40 个月，一个月代表一个时期。然后将评论条数、评论得分期望、评论得分标准差，输出为按照评论时间和评论指标为数值特征的 3 个矩阵，分别记为 $N_{40*8}$，$E_{40*8}$，$S_{40*8}$，每个矩阵均有 40 行（40 个月）8 列（8 类指标），分别用 $n_{ij}$，$E_{ij}$，$s_{ij}$ 来表示 3 个矩阵中第 i 行第 j 列的元素。

传统意义上的熵权法一般仅基于每个指标的变异性一个数学特征对 8 个指标在综合满意度中的相对贡献大小进行赋权，此策略在社会科学、管理科学中广为应用，但在本题中存在一定程度的局限性，即每时期内各类指标下的评论条数和评论情感得分的标准差这两个数学特征，也同时参与决定该指标对主观满意度的影响程度。故我们在进行指标赋权时需同时考虑变异性、评论条数、情感得分标准差这三个因素，为了构建出每个指标对应的权重，我们做出以下四条合理的假设。

假设 1：变异性越大的随机变量包含的信息越多，应赋予的权重就越大。这条也是熵权法的基本假设；

假设 2：每个指标下包含的评论条数反映了人们对该问题的重视程度，人们越重视，应赋予的权重越大；

假设 3：每个时期内每个指标下评论情感得分的标准差反映了人们在时期内对该问题认知的离散程度，该值越大，表明人们对此项指标的观点态度越不一致，

该指标可改善的余地越大，应赋予的权重越大；

假设 4：每一指标获得的最终权重与该指标下的评论期望情感得分的变异性指标（矩阵 E 通过熵权法计算得到的熵权）、评论条数、评论情感得分的标准差成正比。

熵权法具体步骤：

①首先对矩阵 $E_{40*8}$ 进行标准化。

由于 $E$ 的值越大，评论的正向程度越强，$E$ 是相对综合评价的正向变量，故对于矩阵中的每一个元素 $E_{ij}$ 和标准化之后的值 $y_{ij}$，标准化的方法为

$$y_{ij} = \frac{E_{ij}-(E_{\min}\ in\ column\ j)}{(E_{\max}\ in\ column\ j)-(E_{\min}\ in\ column\ j)}。$$

②对于每一项指标 j，计算第 i 个时期该指标下的评价情感得分占该指标下所有时期评论情感得分的比重 $P_{ij}$。具体计算方法为

$$P_{ij} = \frac{y_{ij}}{(\sum_{i=1}^{40} y_{ij}\ in\ column\ j)},$$

其中求和上界 40 为矩阵的行数。

③对于每一项指标 j，计算该指标下所有评论期望情感打分的熵值 $H_{jj}$。具体计算方法为

$$H_j = -\frac{\sum_{i=1}^{40} P_{ij}\ln(P_{ij})}{\ln(40)},$$

其中求和上界 40 为矩阵的行数，系数$-\frac{1}{\ln(40)}$中参数 40 同为矩阵的行数，引入该系数是为了将求得的熵值 $H_j$ 限制在区间[0,1]上。特别地，对于$P_{ij}=0$时，我们在计算上取$P_{ij}\ln(P_{ij})=0$。

④对传统意义的熵权法的改进。

在传统意义的熵权法中，熵值 $H_j$ 和对应权重 $w_j$ 的关系为

$$w_j = \frac{1-H_j}{\sum_1^n(1-H_j)},$$

其中 n 为评价指标的数目，对本题中评论分为 8 大类的情况，即 n=8。传统意义的熵权法存在固有不足，当计算得到的熵值 $H_j$ 接近 1 时，$H_j$ 的微小变化会引起权重 $w_j$ 的很大波动（张近乐，任杰，2005）。经过测算我们提出改进的权重计算公式来避免此问题，即

$$w_j = \frac{1-H_j+\frac{1}{10}\sum_1^n(1-H_j)}{\sum_1^n[1-H_j+\frac{1}{10}\sum_1^n(1-H_j)]}。$$

对于每一项指标 j，使用改进的权重计算公式计算 $w_j$，在上述公式中取 n=8。

⑤对矩阵 $N_{40*8}$ 和 $S_{40*8}$ 的每一列 j，求平均值后得到两个向量 $n$ 和 $s$，并对 $n$ 和 $s$ 进行标准化。

标准化的方法与①中略有不同，考虑到向量中 0 元素的存在，为了将标准化后取值能合理得表达权重，由于其数值仅表示相对分布，在数轴上的平移不改变其分布，我们将其取值区间向右平移一个单位，即控制在[1,2]上。对其中的每一个元素 $x_j$，标准化的方法为

$$y_j = \frac{x_j - (x_{\min}\ in\ column\ j)}{(x_{\max}\ in\ column\ j) - (x_{\min}\ in\ column\ j)} + 1,$$

两个向量中求得的每一个值分别记作 $n_j$ 和 $s_j$。

⑥对于每一类 j，计算综合变异性、评论条数、情感得分标准差三个因素得到的综合权重 $f_j$。具体计算方法为

$$f_j = \frac{w_j * n_j * s_j}{\sum_{j=1}^{8} w_j * n_j * s_j},$$

求和上限 8 为综合评价指标的数目。

（3）计算居民主观满意度综合得分 $F$。

具体计算方法为

$$F = \sum_{j=1}^{8} (f_j * E_j),$$

即满意度综合指数为八项评价指标的情感得分期望值的加权平均，其中求和上限 8 为评价指标的数目，$f_j$为指标 j 的权重，$E_j$表示指标 j 下情感得分的平均期望值。

### 4.2.2 模型的求解

（1）计算结果

通过改进的熵权法我们计算得出的 8 个指标的权重分别为：

表 3 评价指标权重

| 指标 | 权重 | 指标 | 权重 |
|---|---|---|---|
| 交通出行 | 0.119204 | 住房条件 | 0.223043 |
| 基础设施 | 0.001385 | 休闲娱乐 | 0.060099 |
| 社会公平 | 0.07748 | 环境健康 | 0.106077 |
| 科技文化 | 0.20122 | 公共安全 | 0.211492 |

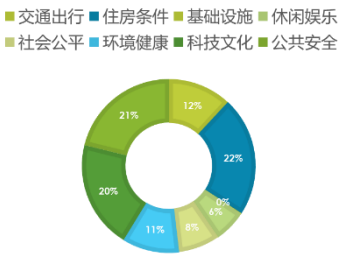


图 2 评价指标权重

根据评论情感得分计算出的 2015 年—2018 年每年各项指标的平均得分如下图所示。从图中我们可以看出，居民对于科技文化的主观感受评价较为积极，对于环境和公共安全的主观感受一直在提升，对于基础设施的主观感受波动较大，对于交通出行的主观感受呈现下降趋势。

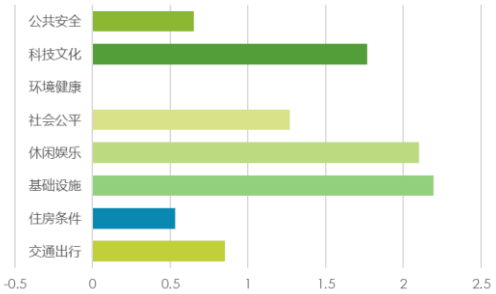


图 3 2015 年各项指标评论得分

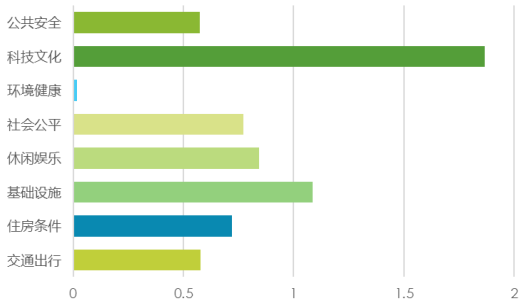


图 4 2016 年各项指标平均得分

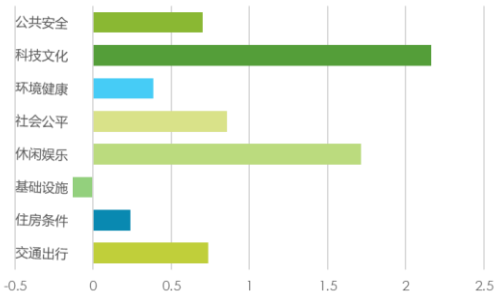


图 5 2017 年各项指标平均得分

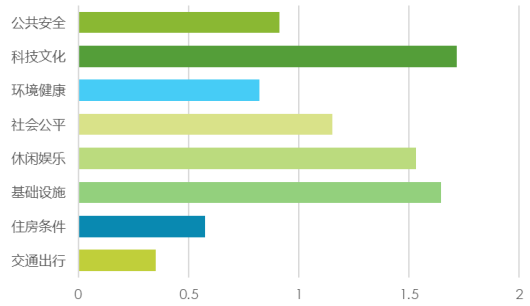


图 6 2018 年各项指标平均得分

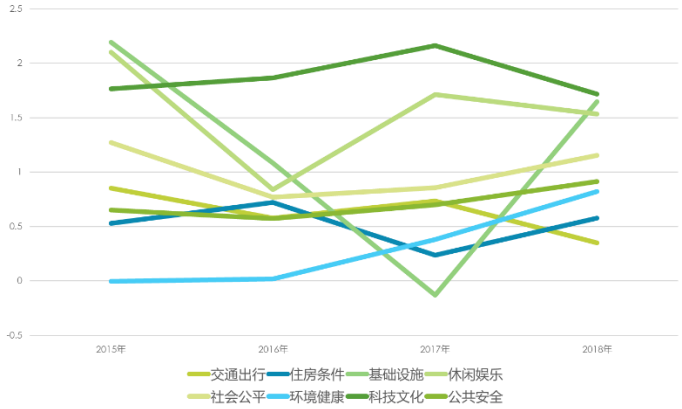


图 7 各指标变化趋势

最终我们得到的 2015 年—2018 年各年的满意度综合指数，从图 8 中可以看出，2016 年的居民综合满意度急剧下降，之后的年份有回升，2017 年基本和 2015 年相持平，2018 年达到新巅峰值。

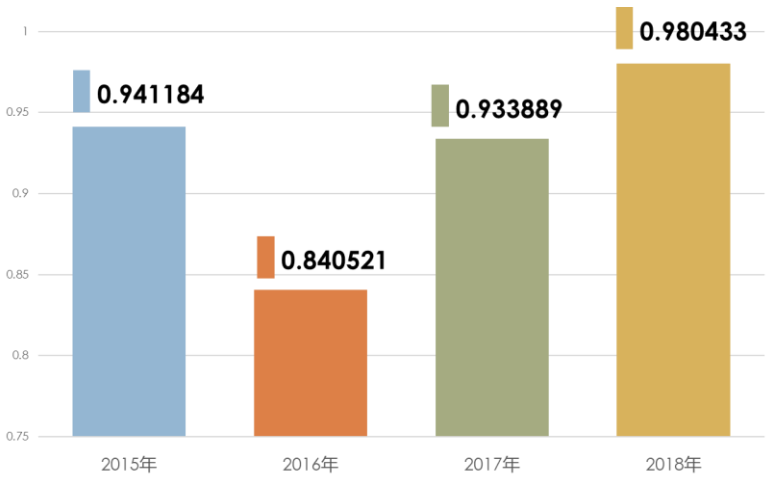


图 8 各年度满意度综合指数

（2）结果分析

权重的大小代表了该项指标对居民综合满意度的影响大小，从结果来看，影响程度由大到小为：住房条件，社会公平，科技文化，交通出行，环境健康，公共安全，休闲娱乐，基础设施。因此我们总结出影响居民对“和谐宜居”主观感受的关键因素有：

①住房条件

古言道：“安居乐业，长养子孙，天下晏然。”可见“安居”即住房条件对于人们生活质量的影响极大，是维护社会和谐稳定的基础。建设和谐宜人城市就是要为人们创造安居乐业的环境条件，首先要能提供足够的住房，同时满足人们居住生活的各种需要。住房条件的好坏和居民生活的基础，是居民对城市的情感的最直接感受。

我们通过图 9 可以看到，人们对于住房条件的情感得分呈现波动，甚至出现负分，这可能是由于北京房价的不断高涨，导致外地劳动者无力负担昂贵的房价，只能无奈选择降低住房标准。我国居民的家庭结构也出现了多元化的趋势，例如单亲家庭、三口之家、单身人群等。不同的收入阶层，不同的家庭结构，对城市居住条件也产生了不同的需求。

宜居城市要使人们在城市里能够住得下、住得起，住得开，住得好，要能够为人们提供人人有其居的条件，并且使人们有可能获得生活居住权和享受到应有的生活居住条件，生活舒适、方便 、安逸。

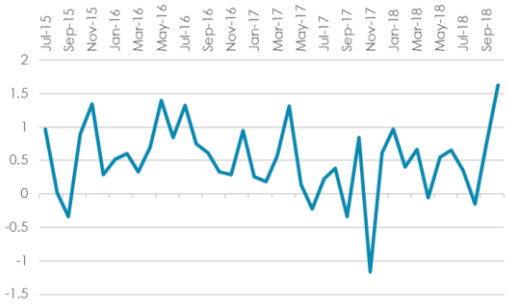


图 9 住房条件情感平均得分变化趋势

②社会公平

城市犹如是一个社会大家庭,大家要共同在法制和道德约束的条件下开展各项社会活动和协调人与人的关系，社会公平是维系社会和谐发展的纽带。

宜居城市建设的核心和根本是为城市中生活的人服务，这就要求政府和规划部门充分考虑不同社会属性人群的居住环境需求差异性，因地因人地推进宜居城市建设。一方面要建立公众参与表达机制。建立健全居民公众利益畅通表达机制，要让不同社会阶层居民的声音都能在宜居城市建设实践中得到反映。另一方面要坚持宜居城市建设的社会公正性。公平公正是宜居城市建设的基本伦理价值，也就是要保障不同社会阶层的居民均能共享到宜居城市建设成果。一般来说，高社会经济地位阶层的居住环境品质相对优越，而低收入人群和外来流动人口等社会弱势群体的居住环境和公共服务享有被边缘化，他们的宜居需求诉求更应该得到优先尊重和表达，从而增加宜居城市建设的人文关怀，减少社会分异。

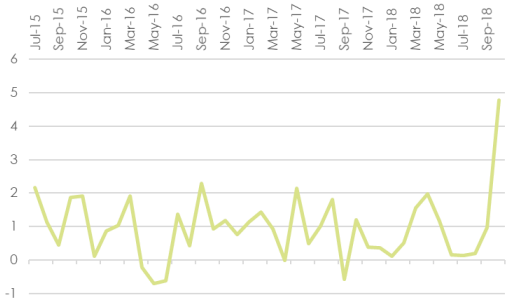


图 10 社会公平情感平均得分变化趋势

③科技文化

经济发展是社会进步的基础，只有经济得到发展，才能解决城市贫困、环境污染、就业不足等一系列城市问题，才能为居民创造良好的城市人居环境，从而促进城市人居软环境的改善。科技是经济发展的驱动力，文化是经济提升的软实力，科技文化创造的人力资本是经济进步的根本原因。

提升城市科技水平，加快产业结构调整并加强基础高新技术的研发，用高科技改变落后状况，不遗余力引进先进人才，可以为宜居城市建设的永续发展提供动力。

同时，随着国家经济实力和国际影响力的提升，民族自信心也开始回升，人们开始追求和重视传统文化的回归，以及传统文化与现代生活价值观的衔接。在这种背景下，传统文化面临多层次的机遇和挑战，传统文化的储存、传承和创新，除了在空间形态上是建设城市可识别性和特色的重要因素，在情感上也是凝聚城市居民认同感和归属感的重要载体。

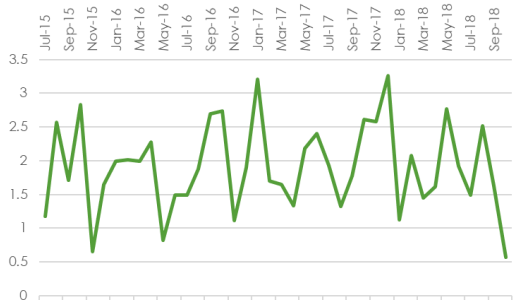


图 11 科技文化情感平均得分变化趋势

④交通出行：

出行便捷是宜居城市建设的基础，优先发展交通则是必然的选择。

随着城市化进程的明显加快，汽车数量急剧上升，交通问题已成为影响城市效率、影响社会经济发展和市民身体健康的突出问题。宜居城市的交通应该是友好的、高效的交通。为了达到这个目标，宜居城市应合理建设交通基础设施，充分完善交通管理系统，大力发展可供选择的公共交通。与此同时，以人为本，建设宜人的、完整的步行休闲网络，方便市民的休闲出行。

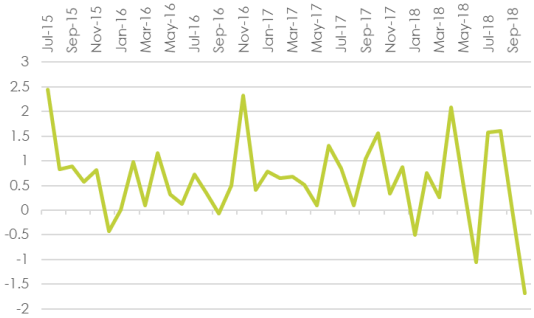


图 12 交通出行情感平均得分变化趋势

⑤环境健康：

优美宜人的生态环境是建设宜居城市最直观的标志和象征。

人口密度的持续增大导致绿地效应快速下降，人均绿地面积持续下降。绿色植物特有的城市美化功能，本应使其成为城市建设重中之重，但对一味追求经济发展的城市建设者将环境视为附属品，进而导致绿地面积同人居需求严重失调，这就成为阻碍宜居城市建设可持续发展的一道屏障。

空气污染是围绕在每个城市上空的难以解决的难题。相关研究结构调查发现，

当人体长时间暴露在污染性空气中时，身体会出现不适。空气污染物浓度突破临界点后，身体健康将会成为空谈。根据对在北京工作的外籍人的调查了解到，北京的雾霾天气已成为外籍人员离京、外籍人员拒绝来京工作的首要因素。

宜居城市要让居民生活得更好，能够满足市民健康生存和繁衍的要求，提供健康的空气、水、住宅、食品等生活环境。

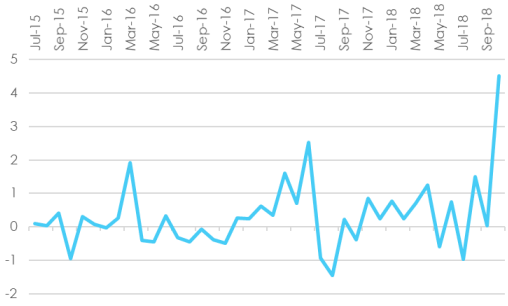


图 13 环境健康情感平均得分变化趋势

## 4.3 问题三的模型建立与求解

题目要求我们研究客观数据对主观情感的影响。我们需要把所研究的客观数据和主观情感按照时间和分类对应起来，由于我们可获得的客观数据和主观情感的范围仅有 2015 年—2018 年，样本数目非常少，无法进行统计回归分析。该情景具有小样本贫信息的灰色系统特征，因此我们考虑使用灰色系统模型中的灰色关联度分析法，并基于此问题的具体情况做了一些改进。

### 4.3.1 模型的构建

（1）整理客观数据。

我们参考中华人民共和国建设部与 2007 年发布的《宜居城市科学评价标准》和第二问中对情感评论分类指标，我们将客观数据分为 5 类，每一类下有若干次级指标。以大类“城市生态环境”为例，我们查找了与之相对应的一些客观数据如“PM2.5 平均浓度（微克/立方米）”，“污水处理率（%）”，“城镇生活垃圾无公害化处理率（%）”，“区域环境噪声平均值（分贝）”等数据。由于部分数据只能查到以年为最小时间步长的均值，为了模型分析的一致性，我们将所有数据都按年为最小时间步长进行统计计算。

我们使用的客观数据详见表 4，所有数据来源于国家统计局和北京统计局发布的统计年鉴：

表 4 客观数据指标

| 指标 | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| **经济富裕度** | | | | |
| 人均 GDP（元/人） | 109603 | 118198 | 128994 | 140748 |
| 城镇调查失业率（%）倒数 | 0.719424 | 0.70922 | 0.699301 | 0.666667 |
| 城镇居民可支配收入 | 52859 | 57275 | 62406 | 62361 |

| | | | | |
|---|---|---|---|---|
| 平均工资（元） | 113073 | 122749 | 134994 | 150748 |
| **城市生态环境** | | | | |
| PM2.5 平均浓度（微克/立方米）倒数 | 0.012407 | 0.014286 | 0.017241 | 0.019608 |
| 污水处理率（%） | 87.9 | 90 | 92.4 | 94 |
| 城镇生活垃圾无公害化处理率（%） | 99.8 | 99.8 | 99.9 | 99.7 |
| 区域环境噪声平均值（分贝）倒数 | 0.018762 | 0.018416 | 0.018797 | 0.018622 |
| 城市绿化覆盖率（%） | 48.4 | 48.4 | 48.4 | 48.44 |
| **社会保障水平** | | | | |
| 城镇居民医疗保险人数占常驻人口比（%） | 0.083391 | 0.087993 | 0.09315 | 0.181413 |
| 企业职工基本养老保险（万人） | 1424.2 | 1459.1 | 1514.3 | 1591.5 |
| 执业（助理）医师数 | 96445 | 100878 | 105732 | 109000 |
| **城市科技发展水平** | | | | |
| 科技活动人员数 | 747461 | 810195 | 958500 | 986352 |
| 研究与试验发展人数（人） | 350721 | 373406 | 397281 | 403698 |
| 研究与试验发展经费（万元） | 13840231 | 14845762 | 15796512 | 15953000 |
| 普通高等学校招收本专科学生（万人） | 15.7 | 15.5 | 15.3 | 15.6 |
| 教育经费支出（万元） | 9998366 | 10937374 | 11171250 | 12786520 |
| **基础设施建设** | | | | |
| 全市公路里程（公里） | 21885 | 22026 | 22242 | 22255.8 |
| 公共交通运营线路长度（公里） | 20740 | 20392 | 19898 | 19245 |
| 基础设施投资占全社会固定资产投资比重 | 27.2 | 28.4 | 33.4 | 33.1 |
| 基础设施投资（亿元） | 2174.5 | 2399.5 | 2984.2 | 2664.9 |
| 公共服务业投资（亿元） | 494.4 | 643.8 | 694.5 | 766.7 |
| 交通运输投资（亿元） | 827 | 973 | 1327 | 1651 |
| 邮政电信投资（亿元） | 172.3 | 147.5 | 194.4 | 189.5 |
| 人均公园绿地面积（平方米/人） | 16 | 16.1 | 16.2 | 16.3 |
| 公共交通客运量（万人次） | 738384 | 734953 | 713396 | 725454 |

我们可以得到了每个次级指标客观数据的向量 $k$，每个向量均包含 2015、2016、2017 和 2018 年数据的四个元素，记为 $k=\{k(1)，k(2)，k(3)，k(4)\}$。以综合满意度为例，综合满意度向量记为（0.94118396878554，0.840520939340113，0.933888794570753，0.980432651338523）。

（2）使用灰色系统模型中的改进灰色关联度分析，研究各客观数据与主观情感的关联度。

灰色关联度分析具体步骤为：

①对每一个向量进行无量纲化。

对此题中的每一个指标 $k_i$ 而言，并非在 2015-2018 年间呈现稳定的增长趋势，故选择均值化的无量纲处理方法，均值化的方法均为

$$K(j) = \frac{k(j)}{\frac{1}{4}\sum_{j=1}^{4} k(j)}，$$

即计算每一年的数据与四年数据平均值的比值。特别地，对于一般意义上，与主观情感综合满意度呈负相关的一些指标，如失业率、噪音平均值等，我们对其预先进行取倒数处理，使得这些指标均转化为相应的正面指标，之后再进行均

值化无量纲处理。

②选定第二问中求得的综合满意度向量为参考列 $K_0$，则其余各客观数据向量均为比较列 $K_i$。

③逐个计算比较列与参考列对应元素的绝对差值，即

$$\Delta_{0,i}(j) = |K_0(\mathrm{j}) - K_i(\mathrm{j})|。$$

在得到绝对差值之中取最大值和最小值，即

$$\Delta_{max(0,i)} = max\{|K_0(\mathrm{j}) - K_i(\mathrm{j})|\}，\ \Delta_{min(0,i)} = min\{|K_0(\mathrm{j}) - K_i(\mathrm{j})|\}。$$

之后在所有比较列的绝对差值最大值和最小值中找到最大值和最小的值，即

$$\Delta_{max} = max\{\Delta_{max(0,i)}\}，\ \Delta_{min} = min\{\Delta_{min(0,i)}\}。$$

④计算指标 i 的比较列 $K_i$ 与参考列 $K_0$ 关于第 j 个元素的关联程度 $\eta_{0,i}(\mathrm{j})$（通常将该指标称为关联系数），即

$$\eta_{0,i}(j) = \frac{\Delta_{min} + \rho * \Delta_{max}}{\Delta_{0,i} + \rho * \Delta_{max}}。$$

式中 $\rho$ 为分辨系数，取值在(0,1)，其作用为削弱因 $\Delta_{max}$ 过大而导致求得的关联系数失真的影响，人为引入这个系数是为了提高关联系数之间的显著差异性。在本题情境中，取 $\rho$ =0.5。

⑤计算每个比较列与参考列之间的关联度。

由于每个比较数列与参考数列的关联程度是通过 j 个关联系数来反映的，关联信息分散，不利于从整体进行比较。因此有必要对关联信息作集中处理，即用比较列与参考列在各个时期的关联系数的平均值来定量反映这两个向量的关联度，记为 $r_{0,i}$。计算公式为：

$$r_{0,i} = \frac{1}{4}\sum_{j=1}^{4}\eta_{0,i}(j)\ ，$$

即计算四年数据的平均值。

⑥计算加权后的百分制标准关联度。

传统的灰色关联度分析中，综合评价系数即与人们常用的百分制评分标准相一致，将每一个参考列的 $r_{0,i}$ 转化为百分数，但并未考虑到每一个因素的权重区别。我们对此缺陷的改进是使用第二问中基于改进熵权法确定的各项指标权重，对于与该项指标相应的客观指标统一按此权重赋权，之后再转化为百分制标准。

⑦按关联度大小对参考列进行排序并进行可视化。

### 4.3.2 模型的求解

通过计算我们得到的灰色相关热力图如下。

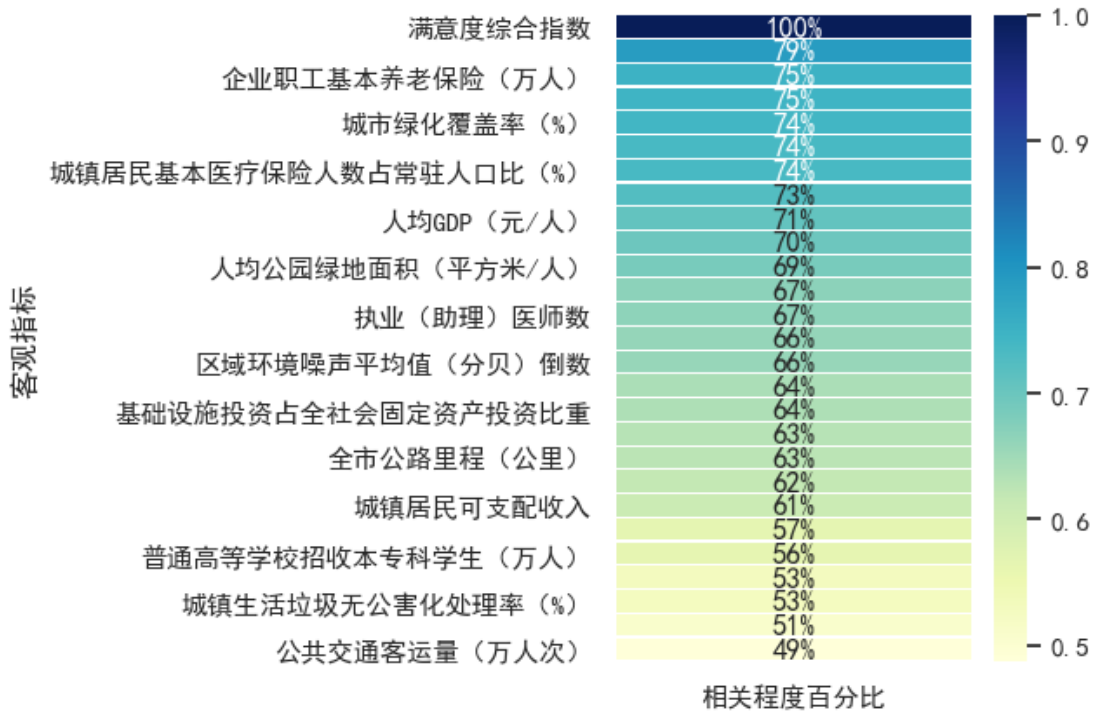


图 14 灰色相关热力图

从计算结果所得客观数据和主观感受的关联度可以看出，关联度较强的几个指标为企业职工基本养老保险、城市绿化覆盖率、城镇居民医疗保险人数占常住人口数比例、人均 GDP 等。由此可以分析出对居民主观感受影响较为强烈的客观指标为经济富裕度、城市生态环境和社会保障水平，城市科技发展水平和基础设施建设影响较小，解释的原因可能是前三个指标涉及的方面可视化程度高，即居民能切身体会到改变，与居民生活密切相关，后两个指标影响人群有针对性，比如有孩子的家庭可能会更关注城市科技发展水平，或者在这四年间变异性小。对于政府如果合理规划北京市城市建设才能有效提高民众对和谐宜居的主观感受，我们提出以下五条建议。

（1）优化经济结构，大力发展城市经济。

宜居城市要求城市具有强劲的经济发展潜力，大力发展经济，创造更多的就业机会，激发经济活力，以确保经济可持续发展，从而提高居民生活水平，持续地为居民营造良好的居住、工作、生活的环境。

（2）改善生态环境，走可持续发展道路。

宜居城市创造美好的生态环境。维护城市的自然环境，让好山好水好风光融入城市，开放和提供更多的绿色空间和休闲空间，同时要控制环境污染，为居民提供健康生活环境。

（3）完善社保制度，福利惠及更多民众。

宜居城市至少应该使每一个居住在该城市的居民能够维持最基本的生活水平。必须建立起包括社会保险、社会救济、社会福利、优抚安置和社会互助等在内的健全的多层次社会保障体系。

同时，城市要提供平等的公民权利体系，城市居民不论种族、肤色、户籍都平等地享有选举权和被选举权，平等地享有教育、医疗、社会保障等公共服务，平等地参与城市管理、接受公共管理。

（4）引入高端人才，有计划的发展科技。

高端人才为城市发展带来新兴动力，政府应出台相关政策吸引人才，如科研扶持政策、住房补贴政策等，降低人才流入的门槛，提高高技术、高学历、高文化水平人才的生活满意度，不仅要将人才吸引到城市来，更重要是留住人才，从而为城市进步注入新鲜动力。

（5）维持公共设施，适当转移部分资金

目前，北京公共设施建设已经趋于成熟，完善的公共设施有利于提高居民生活便宜度。在维持现有公共设施基本不变的情况下，可以考虑将部分投资资金转移到其他正在发展的方面，如建立社保体系等，促进社会资金流的合理转移分配，使社会各层面均衡发展，提高居民综合满意度。

## 4.4 给北京市政府的一封信

尊敬的市政府领导：

您好，感谢您在百忙之中抽出时间来查阅我们的来信。

在过去的五年间，习近平总书记两次视察北京并发表重要讲话，明确北京全国政治中心、文化中心、国际交往中心、科技创新中心的战略定位，提出建设国际一流的和谐宜居之都战略目标。全市广大干部群众在北京政府的带领下，积极有序地展开各项工作，在和谐宜居之都的建设上取得了重大进展，对此我们表示崇高敬意和衷心祝贺。

城市建设应围绕“以人为本”的原则，居民的主观满意度是衡量城市发展进程的重要标准，各项工作推进应以提高居民居住感受的提升为目标，让居民切实感受到城市和谐宜居水平的变化。宜居城市不是自封的，城市的生活居住者最有发言权，城市居民共同给出的答案才是真切的，最具有判断价值。但大家判断城市能不能称得上是“宜居城市”也需要一个客观衡量标准。为此，我们通过整理某社交网络上 2015 年到 2018 年带有与和谐宜居相关的关键词的评论，对每条评论进行情感分析，构建了一个反映居民对城市“和谐宜居”的主观满意度综合指数。

值得庆祝的是，通过我们计算，自 2016 年其居民对于北京城的主观满意度在逐渐提升，比较遗憾的是，2018 年主观满意度巅峰值也仅有 0.98（5 为正向满分），未能实现 1 的突破，建设北京成为“和谐宜居之都”任重道远。我们还分析了影响了主观评价的主要因素和客观事实数据与评价的联系，对北京建设宜居城市的发展战略和规划总结出了以下 3 点建议供参考。

（1）重视生活基础建设，构建安居乐业的和谐社会。

城市不仅是生产的城市、流通的城市财富的城市，更重要的是人们生活的城市、消费的城市、安居乐业的城市。打造宜居城市，必须把安居乐业放到首要的地位上来考虑。

社区是人们长期聚齐的地方。舒适方便卫生安全的社区，使人们的生活状态和乐美满，从而对社区产生归属感、安全感和家园感。在居住区物质设施方面，要有充足的的住宅，有供电、供水、燃气、网络等配套设施，有商店、学校、医院等生活服务设施，有绿化美化；在社区环境方面，要有和睦的邻里关系，充满人际关怀的社区氛围；在交往方面，根据住区的地域特点和居民结构，设计符合社区自身特点的交往环境，开展各种丰富多彩的文化社区活动。

同时，要改善城市居民住房条件，根据城市居民居住需求动态调控经济适

用房、廉租房、商品房比重，完善廉租住房制度，着力解决低收入家庭的住房问题，多渠道解决外来劳动者的居住条件；要加快市政公用基础设施建设，全力推进市政道路建设，实现城市公共交通全覆盖，规划建设合理的城市道路交通体系。

（2）保护环境 ,创造生态良好的绿色城市。

宜居城市中必须要求清新空气、蔚蓝天空、健康人工环境、必要的自然风光。政府部门要加强法规建设，以严格标准合理管控高污染、高耗能新建项目，以强制性效能标准约束产品生产流程，从污染产生源头进行控制和治理。同时要树立公众生态环境保护意识，在城市建设同时做好循环经济宣传工作，让更多居民树立宜居城市建设自觉性，使宜居城市建设具有可持续性。

（3）突出文化个性，建设独具特色的优美城市

过去，由于在城市发展建设中看重城市的使用功能和实用性，主要考虑解决现代化和功能性的问题，所以对城市的个性特色的考虑不多，没有给予城市的历史文化、民族传统、地方风情、风俗习惯、文脉记忆和独特景观应有的重视。城市面貌趋同化，让人们感到生活居住城市里缺乏传承记忆、缺乏文化感染力、缺乏与众不同的骄傲感和留恋感，于是对该城市是否“宜居”就会产生质疑。人们不仅需要物质上的满足 ,更需要精神上的满足。

政府部门应做好城市文化保护工作，加大对重点文物、历史文化街的保护力度，启动城市历史文化遗址抢救修缮工程，恢复历史原貌，并加大对特色传统文化如胡同、四合院、卤煮、京剧等建筑、饮食、服饰、艺术等各方面的宣传力度，充分展示其所具有的历史文化特色。

以上是我们对北京建设宜居城市的发展战略和规划提出的建议。

建设北京成为国际一流的和谐宜居之都，是顺应时代潮流、服务国家发展、造福人民群众、凝聚奋进力量的崇高事业。我们坚信，在党中央的坚强领导下，在政府部门的合理规划下，全市人民的团结奋斗，北京的明天一定会更好！

此致

敬礼

建模团队

2019.05.03

## 5.1 模型的评价

### 5.1.1 模型的优点

（1）模型建立严谨客观，其中改进熵权法模型的所有参数均完全基于数字计算方法得到，无主观赋权数据，考虑的维度和因素较为全面；

（2）模型的计算采用专业软件求解，例如 Python 软件、EXCEL 软件等对数据进行处理和模型求解，用于分析的数据可信度较高；

（3）建立的模型能够与实际紧密联系，结合实际情况对问题进行求解，使得模型具有很好的通用性和推广性。

### 5.1.2 模型的缺点

（1）在用语料库训练情感分析模型时，反复训练应该会得到更加精确的结

果。但训练的时间代价很高，生成语料文件大约需要 1h，训练大约需要 1.5h。在有限的建模时间里，我们的模型仅进行了一次迭代。在时间充足的情况下，我们将用语料库多次反复训练情感分析模型，以提高模型的准确度。

（2）我们的模型忽略了同一个用户发表超过一条评论的影响，这种影响可能造成对于某一和谐宜居情感指标下评论情感的重复计算，也未能分析同一个用户在对各项指标的相关性。我们对评论列表进行分析，其中仅发表一条评论的用户发表的评论数占比约 67.786%，经数据筛选后这个比重将会扩大，所以影响可以忽略。

（3）由于评论的即时性，我们的模型未考虑用户经过较长时间后对于某一和谐宜居情感指标的情感变化。在以后的改进中我们将引入更长时间的评论数据，对长时间的感情趋势进行分析。

## 5.2 模型的推广

在全国推进“和谐宜居城市”建设的背景下，本文建立了完整的居民主观满意度模型，对居民在社交网站上的相关评论进行分析，用居民满意度来衡量建设宜居城市工作的成果，我们也找出了影响民众主观感受变化的主要因素和客观条件，使得政府相关部门和城市建设者在制定城市发展战略和开展工作时能够有效地找准城市发展方向和随时调整发展路径，从而使得建设的各方面投入更有质量地转化为居民对城市生活满意度的提高。

此外，本文建立的模型能够广泛应用于全国其他城市中，对于现实生活中的全国各城市建设文明和谐宜居的环境有着一定的参考和指导意义。

## 参考文献


【1】 张文忠. 中国宜居城市建设的理论研究及实践思考[J]. 国际城市规划, 2016(5):1-6.

【2】 袁锐. 试论宜居城市的判别标准[J]. 经济科学, 2015, Vol.27(4):126-128.

【3】 姜煜华, 甄峰, 魏宗财. 国外宜居城市建设实践及其启示[J]. 国际城市规划, 2009, 24(4).

【4】 黄江松, 鹿春江, 徐唯燊. 基于马斯洛需求理论构建宜居城市指标体系及对北京的宜居评价[J]. 城市发展研究, 2018, v.25；No.201(05):94-98.

【5】 任致远. 关于宜居城市的拙见[J]. 城市发展研究, 2005, 12(4):33-36.

【6】 郭洁. 基于未来生活方式的宜居城市规划设计思考[C]// 2018 中国城市规划年会. 0.

【7】 程启先, 樊哲宇, 秦春艳, et al. 基于层次分析法的信阳市城市宜居性研究[J]. 环境与发展, 2019(1):5-7.

【8】 崔凤琪, 唐海萍, 张钦. 京津冀地区城市宜居性评价及影响因素分析:基于2010—2016 年面板数据的实证研究[J]. 北京师范大学学报(自然科学版), 2018, 54(05):110-117.

【9】 张近乐，任杰．熵理论中熵及熵权计算式的不足与修正[J]. 统计与信息论坛, 2005, (9):5-7.

## 附录一 基于 NLP 情感分析模型的训练代码

```
from snownlp import SnowNLP
from snownlp import sentiment
import pandas as pd
import utils
import os

dirlist = os.listdir()


def models(commands):
    words_list = utils.word_frequency_statistics()
    for comm in commands:
        if comm == 'nan':
            continue
        else:
            comm = utils.Extract_Commands(comm).extract_command()
            for w in words_list:
                if w in comm:
                    score = SnowNLP(comm)
                    # 预训练模型分类
                    if score.sentiments > 0.8:
                        with open('pos.txt', mode='a', encoding='utf-8') as p:
                            p.writelines(comm + '\n')
                    elif score.sentiments < 0.2:
                        with open('neg.txt', mode='a', encoding='utf-8') as n:
                            n.writelines(comm + '\n')
                    break


def train():
    if 'raw_data.csv' in dirlist:
        df = pd.read_csv('raw_data.csv')
    else:
        raise Exception('请先创建 raw_data.csv 文件')
    df.fillna('nan')
    commands = df.评论内容.dropna().tolist()
    models(commands)
    sentiment.train('neg.txt', 'pos.txt')
    sentiment.save('mysentiment.marshal')
    print('得到模型后需拷贝到 snownlp 的 sentiment 文件夹下\
        并修改__init.py__的路径来加载新权重')
```

## 附录二 模型需要使用的工具类

```
import re
import pandas as pd
import numpy as np


class Extract_Commands():
    # 提取微博评论的内容
    def __init__(self, content):
        super(Extract_Commands, self).__init__()
        self.content = content
        if content == '':
            raise Exception('评论内容为空')

    def clear_quotation(self):
        if self.content[0] == '回' or self.content[0] == '@':
            self.content = re.sub('[@回](.*)[: ]', '', self.content)
        else:
            self.content = re.sub('[@回](.*)', '', self.content)

    def clear_special_character(self):
        if '/' in self.content:
            self.content = re.sub('/', '', self.content)
        if '#' in self.content:
            self.content = re.sub('#(.*)#', '', self.content)

    def extract_command(self):
        self.clear_special_character()
        if self.content != '':
            self.clear_quotation()
        return self.content


def word_frequency_statistics(raw_data):
    categorys = raw_data[raw_data.关键词].unique()
    words_list = []
    # 分类统计词频
    for c in categorys:
        df_new = raw_data[(raw_data.分词后评论内容 == c)]
        df_new.dropna(inplace=True)
        cuts = df_new.tolist()

        counts = {}
```

```
            for words in cuts:
                if isinstance(words, str):
                    word_list = words.split(' ')
                for w in word_list:
                    if w == '' or w == '回复' or len(w) == 1:
                        continue
                    else:
                        counts[w] = counts.get(w, 0) + 1
            items = counts.items().tolist()
            items.sort(key=lambda x: x[1], reverse=True)

            print('正在生成 %s 的关键词库' % c)
            for i in range(20):
                w, count = items[i]
                words_list.append(w)
        words_list = list(set(words_list))  # 去重

        # 使用中文停用词表对关键词进行筛查
        with open('中文停用词 1208.txt', 'r') as f:
            for line in f.readlines():
                line = line.strip('\n')
                del words_list[words_list.index(line)]
        return words_list


class EntropyMethod():
    def __init__(self, index, positive, negative, row_name):
        if len(index) != len(row_name):
            raise Exception('数据指标行数与行名称数不符')
        if sorted(index.columns) != sorted(positive + negative):
            raise Exception('正项指标加负向指标不等于数据指标的条目数')

        self.index = index.copy().astype('float64')
        self.positive = positive
        self.negative = negative
        self.row_name = row_name

    def uniform(self):
        uniform_mat = self.index.copy()
        min_index = {column: min(uniform_mat[column])
                     for column in uniform_mat.columns}
        max_index = {column: max(uniform_mat[column])
                     for column in uniform_mat.columns}
        for i in range(len(uniform_mat)):
```

```
            for column in uniform_mat.columns:
                if column in self.negative:
                    uniform_mat[column][i] = (
                        uniform_mat[column][i] - min_index[column]) \
                        / (max_index[column] - min_index[column])
                else:
                    uniform_mat[column][i] = (
                        max_index[column] - uniform_mat[column][i]) \
                        / (max_index[column] - min_index[column])

        self.uniform_mat = uniform_mat
        return self.uniform_mat

    def calc_probability(self):
        try:
            p_mat = self.uniform_mat.copy()
        except AttributeError:
            raise Exception('你还没进行归一化处理，请先调用 uniform 方法')
        for column in p_mat.columns:
            sigma_x_1_n_j = sum(p_mat[column])
            p_mat[column] = p_mat[column].apply(
                lambda x_i_j: x_i_j / sigma_x_1_n_j if x_i_j
                / sigma_x_1_n_j != 0 else 1e-6)

        self.p_mat = p_mat
        return p_mat

    def calc_emtropy(self):
        try:
            self.p_mat.head(0)
        except AttributeError:
            raise Exception('你还没计算比重，请先调用 calc_probability 方法')

        import numpy as np
        e_j = -(1 / np.log(len(self.p_mat) + 1)) * np.array([sum([pij * np.log(
            pij) for pij in self.p_mat[column]])
            for column in self.p_mat.columns])
        ejs = pd.Series(e_j, index=self.p_mat.columns, name='指标的熵值')

        self.emtropy_series = ejs
        return self.emtropy_series

    def calc_emtropy_redundancy(self):
        try:
```

```
                self.d_series = 1 - self.emtropy_series
                self.d_series.name = '信息熵冗余度'
            except AttributeError:
                raise Exception('你还没计算信息熵，请先调用 calc_emtropy 方法')

            return self.d_series

        def calc_Weight(self):
            self.uniform()
            self.calc_probability()
            self.calc_emtropy()
            self.calc_emtropy_redundancy()
            self.Weight = self.d_series / sum(self.d_series)
            self.Weight.name = '权值'
            return self.Weight

        def calc_score(self):
            self.calc_Weight()

            import numpy as np
            self.score = pd.Series(
                [np.dot(np.array(self.index[row:row + 1])[0],
                        np.array(self.Weight))
                 for row in range(len(self.index))],
                index=self.row_name, name='得分'
            ).sort_values(ascending=False)
            return self.score


    class Gray_Relational_Analysis():
        def __init__(self, df, m=0):
            super(Gray_Relational_Analysis, self).__init__()
            self.df = df
            self.m = m

        def gray_analysis(self):
            gray = (self.df - self.df.min()) \
                / (self.df.max() - self.df.min())

            std = gray.iloc[:, self.m]  # 为标准要素
            ce = gray.iloc[:, 0:]  # 为比较要素
            n = ce.shape[0]
            m = ce.shape[1]  # 计算行列
```

```
        # 与标准要素比较，相减
        a = np.zeros([m, n])
        for i in range(m):
            for j in range(n):
                a[i, j] = abs(ce.iloc[j, i] - std[j])

        # 取出矩阵中最大值与最小值
        c = np.amax(a)
        d = np.amin(a)

        # 计算值
        result = np.zeros([m, n])
        for i in range(m):
            for j in range(n):
                result[i, j] = (d + 0.5 * c) / (a[i, j] + 0.5 * c)

        # 求均值，得到灰色关联值
        result2 = np.zeros(m)
        for i in range(m):
            result2[i] = np.mean(result[i, :])
        t_list = result2.tolist()
        del t_list[0]  # 删除年份
        # 相关度向量
        RT = pd.DataFrame(t_list)
        # 用来画图的 RT_plt
        RT = RT.rename(columns={0: "相关程度百分比"})
        RT['客观指标'] = self.df.columns[1:]
        RT_plt = RT.pivot_table(index='客观指标', values='相关程度百分比')
        RT_plt = RT_plt.sort_values(by='相关程度百分比', ascending=False)

        self.RT = RT
        self.RT_plt = RT_plt

    def show_gra_heatmap(self):
        self.gray_analysis()
        import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set(font='simhei')
        plt.title('灰色分析热力图', y=1.05, size=15)
        plt.xlabel('客观指标')
        plt.ylabel('相关程度百分比')
        sns.heatmap(self.RT_plt, linewidths=0.1, vmax=1.0,
                    fmt='.0%', cmap="YlGnBu", linecolor='white', annot=True)
        plt.tight_layout()
```

```
        plt.savefig("灰色分析热力图.jpg",)
        plt.show()
```

**附录三 CSV 数据表生成类，可以生成本文的全部数据**

```
from snownlp import SnowNLP
import pandas as pd
import utils
import os
import numpy as np
import re


class Generate_CSV():
    """docstring for Generate_CSV"""

    def __init__(self):
        super(Generate_CSV, self).__init__()
        self.dirlist = os.listdir()
        if '关键词分类.xlsx' in self.dirlist:
            self.keywords_df = pd.read_excel('关键词分类.xlsx')
        else:
            raise Exception('请先创建关键词分类.xlsx 文件')
        self.date_list = ['2015/7', '2015/8', '2015/9', '2015/10',
                          '2015/11', '2015/12', '2016/1', '2016/2',
                          '2016/3', '2016/4', '2016/5', '2016/6',
                          '2016/7', '2016/8', '2016/9', '2016/10',
                          '2016/11', '2016/12', '2017/1', '2017/2',
                          '2017/3', '2017/4', '2017/5', '2017/6',
                          '2017/7', '2017/8', '2017/9', '2017/10',
                          '2017/11', '2017/12', '1 月', '2 月', '3 月', '4 月',
                          '5 月', '6 月', '7 月', '8 月', '9 月',
                          '10 月', '11 月', '12 月']

    def mark_commonds(self):
        # 这个函数将两个功能合在一起实现
        # 1.过滤掉无用评论 2.对有用评论打分
        if 'raw_data.csv' in self.dirlist:
            raw_df = pd.read_csv('raw_data.csv')
            raw_df.fillna('nan')
        else:
            raise Exception('请先创建 raw_data.csv 文件')
        raw_df = pd.read_csv(self.csv_path)
        word_lists = utils.word_frequency_statistics()
```

```
        df = pd.DataFrame(columns=['keywords', 'commands', 'date', 'mark'])
        for i, row in raw_df.iterrows():
            comm = str(row.评论内容)
            if comm == 'nan' or comm == ''or len(comm) < 2:
                continue
            else:
                comm = utils.Extract_Commands(comm).extract_command()
                if comm == '':
                    # 有可能全部是引用 没有有效评论
                    continue

            for w in word_lists:
                if w in comm:
                    mark = round(SnowNLP(comm).sentiments, 3)
                    df.loc[df.shape[0]] = [row.关键词, comm, row.评论时间,
mark]
                    if df.shape[0] % 100 == 0:
                        print('已经处理%d 条评论' % df.shape[0])
                    break
        print('所有评论打分完毕,正在生成 sentiment_analysis.csv')
        df.to_csv('sentiment_analysis.csv', index=False, encoding='utf_8_sig')

    def commands_attribute(self):
        if 'sentiment_analysis.csv' in self.dirlist:
            sen_df = pd.read_csv('sentiment_analysis.csv')
        else:
            raise Exception('请先创建 sentiment_analysis.csv 文件\
                或调用 mark_commands 方法')
        t_list = self.keywords_df.columns.tolist()
        t_list.insert(0, '日期')
        df_avg = pd.DataFrame(columns=t_list)
        df_std = pd.DataFrame(columns=t_list)
        df_sum = pd.DataFrame(columns=t_list)

        for t in self.date_list:
            df_t = sen_df[sen_df.date.str.contains(t)]
            if df_t.empty:
                continue
            list_avg = []
            list_std = []
            list_sum = []
            for c in self.keywords_df.columns:
                kwd_list = self.keywords_df.c.dropma().tolist()
                kwd_s = ''
```

```
                for i, s in enumerate(kwd_list):
                    if i == 0:
                        kwd_s += s
                    else:
                        kwd_s += '|'
                        kwd_s += s
                df_attr = df_t[df_t.keywords.str.contains(kwd_s)]
                list_avg.append(np.mean(df_attr.mark))
                list_std.append(np.std(df_attr.mark, ddof=1))
                list_sum.append(len(df_attr))
            time_list = re.findall(r'\d+\.?\d*', t)
            if '月' in t:
                t_str = '2018-' + str(time_list[0])
            else:
                t_str = str(time_list[0]) + '-' + str(time_list[1])
            list_avg.insert(0, t_str)
            list_std.insert(0, t_str)
            list_sum.insert(0, t_str)

            print('正在处理%s 的数据', t_str)
            df_avg.loc[df_avg.shape[0]] = list_avg
            df_std.loc[df_std.shape[0]] = list_std
            df_sum.loc[df_sum.shape[0]] = list_sum

        df_avg.to_csv('avg_of_mark.csv', index=False, encoding='utf_8_sig')
        df_std.to_csv('std_of_mark.csv', index=False, encoding='utf_8_sig')
        df_sum.to_csv('sum_of_commands.csv', index=False, encoding='utf_8_sig')

    def factor_weights(self):
        df = pd.DataFrame(columns=['分类', '平均分权重',
                                   '标准差权重', '评论总数权重',
                                   '归一化的总权重'])
        if 'avg_of_mark.csv' in self.dirlist and \
            'std_of_mark.csv' in self.dirlist and\
                'sum_of_commands' in self.dirlist:
            df_avg = pd.read_csv('avg_of_mark.csv')
            df_std = pd.read_csv('std_of_mark.csv')
            df_sum = pd.read_csv('sum_of_commands.csv')
        else:
            raise Exception('请先创建 avg_of_mark.csv, std_of_mark, \
                sum_of_commands.csv 文件  或调用 commands_attribute 方法')

        df_avg_indexs = df_avg.columns[1:].tolist()
        df_avg_positive = df_avg_indexs
```

```
        df_avg_negative = []
        df_avg_date = df_avg['日期']
        df_avg_index = df_avg[df_avg_indexs]

        df_avg_en = utils.EntropyMethod(df_avg_index, df_avg_negative,
                                                df_avg_positive, df_avg_date)
        avg_series = df_avg_en.calc_Weight()
        df.平均分权重 = avg_series

        std_wl = []
        for c in df_std.columns[1:]:
            std_wl.append(np.mean(df_std.c))
        std_ws = pd.Series(std_wl)
        std_ws = std_ws / np.sum(std_ws)
        df.标准差权重 = std_ws.tolist()

        sum_wl = []
        for c in df_sum.columns[1:]:
            sum_wl.append(np.mean(df_sum.c))
        sum_ws = pd.Series(sum_wl)
        sum_ws = std_ws / np.sum(sum_ws)
        df.评论总数权重 = sum_ws.tolist()

        df.分类 = df_avg.columns[1:]
        ws = df.平均分权重 * df.标准差权重 * df.评论总数权重
        ws = ws / np.sum(ws)
        df.归一化的总权重 = ws

        df.to_csv('factor_weights.csv', index=False, encoding='utf_8_sig')

def total_grade(self):
        if 'avg_of_mark.csv' in self.dirlist and \
                'factor_weights.csv' in self.dirlist:
            df_score = pd.read_csv('avg_of_mark.csv')
            df_weight = pd.read_csv('factor_weights.csv')
        else:
            raise Exception('请先创建 avg_of_mark.csv, factor_weights.csv\
                或调用 factor_weights 方法')
        t_list = df_score.columns.tolist()
        t_list.append('满意度综合指数')
        df = pd.DataFrame(columns=t_list)
        years = ['2015', '2016', '2017', '2018']

        for y in years:
```

```
                df_year = df_score[df_score.日期.str.contains(y)]
                score_array = np.array(np.mean(df_year))
                weight_array = np.array(df_weight.归一化的总权重)
                col_list = np.multiply(score_array, weight_array).tolist()
                col_list.append(np.sum(col_list))
                col_list.insert(0, y)
                df.loc[df.shape[0]] = col_list

        df.to_csv('total_grade.csv', index=False, encoding='utf_8_sig')

    def gary_relational_analysis(self):
        if 'total_grade.csv' in self.dirlist and\
                'annual.csv' in self.dirlist:
            df_satisfy = pd.read_csv('total_grade.csv')
            df_annual = pd.read_csv('annual_v2.csv')

            list_df = df_annual.columns.tolist()
            list_df.insert(1, df_satisfy.columns[-1])
            df = pd.DataFrame(columns=list_df)

            df[df_satisfy.columns[-1]] = df_satisfy.iloc[:, -1]
            for col in df_annual.columns:
                df[col] = df_annual[col]
        gra = utils.Gray_Relational_Analysis()
        gra.gray_analysis()
        gra.RT.to_csv('gary_relational_analysis.csv',
                      index=False, encoding='utf_8_sig')
        gra.show_gra_heatmap()

    def process_all(self):
        self.mark_commonds()
        self.commands_attribute()
        self.factor_weights()
        self.total_grade()
        self.gary_relational_analysis()
```

## 附录四 文件结构描述及代码使用

(本程序不能生成-5~5 的情感打分 但-5~5 的数据被存在_v2 中)

1. 所有机器运行得到的数据都被存放在.csv 文件中
2. 带有“v2”表明是将情感评分放大到-5~5 的范围内进行的运算，不带“v2”的是 0~1
3. factor_weights 里面是每项指标对应的权重，对应于 csv 文件的最后一列，“归一化的总权重”

4. index_calculation 里面有三个文件夹，分别是 avg_of_mark, std_of_mark, sum_of_commands。

avg 代表按月计算的，对于 8 个指标的，情感打分的平均值；std 代表标准差（无偏）；sum

代表相关指标的评论总人数。

5. sentiment_analysis 里面包括筛选后的所有评论的情感打分，mark 一栏是 0~1，score 一栏是-5~5

6. total_grade 是四年的满意度综合指数

7. annual 代表考虑负项指标后的结果（对负向指标取倒数）

8. models 存放生成的正负样本 和一次训练得到的模型（反复训练效果更好，因为时间原因，只训了一次）。

要将得到的.marshal 文件放置在 snownlp 的 sentiment 目录下，并修改__init.py__中的权重路径

9. annual.csv 是对北京统计年鉴.xlsx 的整理