“信息茧房”现象的破除与信息平衡策略分析

摘要

近年来，随着信息传播格局的全新升级，“尖叫效应”与“回声室效应”在信息传播中层出不穷，导致“信息茧房”的出现，使人们的眼界与思想趋于窄化，甚至变得偏执与自固.为了破解这种效应带来的危害，寻找合适的策略，在主流信息与个性化信息之间达到平衡是重要的.

对于问题一，构造了相应的社交网络模型，借助贝叶斯公式进行参数表示，并将邻接矩阵运用到类似于传染病模型的**$SEIR$信息传播微分方程模型**中.使用离散化方法进行求解，并作出阅读量变化图像，在参数$\lambda = 0.8$，$\rho = 0.2$时，拟合曲线与实际贴近程度最高.最后分析了传播率、网络拓扑结构等因素在信息传播过程中的影响.

对于问题二，为了研究用户节点的不同属性对信息传播的影响，对用户评论的**语料特征**进行情感分析，将用户划分为不同具有情感标签、传染系数的5个**子群体**，以子群体为单位进行$SEIR$信息传播微分方程模型拟合，提升了与实际情况的贴合程度。为了刻画中立与极化的形成机制，将I节点拆分为激进的I_1节点与中立的I_2节点，引入**话题吸引度μ**，算出不同类型话题下I_1与I_2比例的变化，在激进的高吸引度话题下I_1的比例为优势位；为了描绘“尖叫效应”，引入**脉冲函数**：δ函数，并对微分方程进行修正，再离散化求解拟合出新的图像，图像在传播初期阅读量增长更加显著；在**基于模型的协同过滤**中，使用ALS**矩阵分解算法**获取用户打分矩阵以了解偏好，从而推荐用户高评分的信息，并采集用户的数据行为对用户打分矩阵进行更新，像回声一样循环往复，得到一个趋于稳定的用户打分矩阵，此时的信息推荐最贴合用户偏好.两种效应互相结合形成“信息茧房”.分析发现高话题吸引度等**确定性因素**与节点用户高活跃度、从众心理等**不确定因素**都会促进了“信息茧房”的产生.

对于问题三，从“尖叫效应”与“回声室效应”的**形成机理开始**分析。用户行为层面列举了如定期清除浏览记录cookie、避免浏览同质化信息等能够削弱这些效应的行为。算法设计方面，设计了传播速度监测与信息内容识别两种方法识别尖叫效应 ，再启用**宏观的流量优化**和**局部“定向免疫”控制**两种方案以遏制信息快速传播，从而遏制了“尖叫效应”；通过使用**$RMSE$-阈值**控制ALS算法的迭代次数，避免“过度学习”用户喜好,从而遏制“回声室效应”。

对于问题四，针对现在网络的“信息茧房”格局，从政府、主流媒体的引领、广大用户自身三个角度提出具体可行的建议，如出台法规，打击哗众取宠和传播恶意消息的行为，并借助数据验证支持了方案的有效性和应用性.

综上所述，基于建立在合理假设之上的分析与模型所提供的策略是实用而准确的.

关键词：微分方程 网络 ALS矩阵分解 语料特征

目录：

一、问题重述

问题背景：

随着全新的信息传播格局的出现，“尖叫效应”与“回声室效应”以及“信息茧房”等现实问题层出不穷.对于以上问题的解决，应从信息传播过程，算法推荐，网络用户等方面入手，从而帮助公众尽可能客观、清晰地了解社会热点问题.

问题要求：

1. 通过建立数学模型，刻画信息传播的过程，并根据在社交媒体收集的数据，分析影响信息传播的因素.
2. 建立数学模型刻画出中立性、两极分化机制的形成，同时研究“尖叫效应”、“回声室效应”以及“信息茧房”形成的过程，并研究这些机制中受到话题吸引度等因素的影响大小.
3. 根据问题 2 建立的数学模型，制定破除“尖叫效应”和“回声室效应”、规避“信息茧房”的策略.
4. 基于上述数据分析与数学模型，针对如何破除“信息茧房”撰写 1~2 页报告，分别对政府的顶层设计、主流媒体的引领和广大网络用户的责任担当提出相应的解决方案或建议.

二．问题分析

2.1 问题一分析

对于信息传输过程的描述，我们选择借鉴信息传播与疾病传播上的相似性，采用系统动力学模型进行研究.针对某一特定信息，首先使用社交网络图论模型描绘出全体网络用户的在相应网络节点上的 4 种状态分类，“S：易传播节点”、“E：潜伏节点”、“I：感染节点”、“R：免疫节点”.使用有向赋权图描绘状态之间的转化关系与概率，同时借助贝叶斯公式确定出有向赋权图的邻接矩阵中的各个权重的取值以定量出信息传播模型的主要参数[1].

我们再使用基于微分方程的系统动力学模型对信息传播过程中的各个网络节点状态的此消彼长变化进行模型构建，建立起了$SEIR$信息传播模型.并从微博

选择热搜话题进行阅读量、讨论区数据采集. 再进行模型计算，拟合出信息传播过程中的阅读量变化曲线并不断调整参数以尽可能贴近原始数据图，做出相应的讨论说明. 最后从模型组成的角度剖析了影响传播过程的一些常见因素.

2. 2 问题二分析

一系列网络传播现象的出现，离不开具体每个网络用户的个人属性带来的影响，在此引入每个网络节点（用户）的属性刻画来对传统微分方程传播模型作为补充

为了研究具体到不同用户对于接收到信息后产生的不同反应，我们对用户针对某一信息下的发言进行整理，提取词句特征并进行分析，以判断情感倾向并将原总易感人群样本依据情感倾向的类似性，划分诸多小型的易感节点群体，并赋予不同的传播系数，借助$SEIR$模型进行模拟并汇总

对于中立共识与观点极化产生的机制探讨，我们在第一问信息传播模型的基础上，考虑网络平台的推荐算法、用户个人因素在信息传播过程中影响对话题最终走向的影响. 同时我们使用疾病传播过程中的类似现象类比，“尖叫效应”可以使用脉冲函数来对其特性进行刻画，“回声室效应”可以看作是在对于用户喜好的不断学习下的信息定向推送.

在自然社交网络信息传播规律、平台的倾向性推荐算法、用户的意识三方面动力共同作用下，当出现了“尖叫效应”、“回声室效应”两个现象时，“信息茧房”很快就会形成. 我们把话题的吸引度、用户的活跃度等一系列因素看作是作用参量，通过灵敏度分析的方法来研究它们的作用.

2. 3 问题三分析

问题三要求是分别从“尖叫效应”与“回声室效应”两个方面，给出破除“回声室”效应的策略. 从两种效应的形成与特征入手，分别找出影响这些效应发展的一些主要因素，再去具体诠释破解这些要素的具体方法，即可从来源上阻止“信息茧房”的形成.

2. 4 问题四分析

将问题三中的一些措施按照信息传输、系统算法、用户行为等方面进行分类，综合阐述并给出一个合理的报告，即为第四问.

三．模型假设

1. 网络信息的传播仅考虑正常情况，不计网络延迟、网络故障等不可抗力因素带来的干扰；
2. 处于潜伏状态的节点E不具有信息传播性；
3. 信息不会在传播过程中出现信息歪曲或失真；
4. 存在于该网络下的网络用户总数是不变的.

四．符号说明

符号	符号说明
$S(t)$	t时刻易传播节点占总节点数比例
$E(t)$	t时刻潜伏节点占总节点比例
$I(t)$	t时刻感染节点占总节点比例
$Q(t)$	t时刻免疫节点占总节点比例
$\lambda(t)$	日接触率
D_t	t时刻话题总阅读量
μ	话题吸引度系数

五．模型的建立与求解

5.1 数据采集与预处理

5.1.1 数据采集

根据题目需求，收集热点话题的相关数据.要求的话题数据一共有两种类型：一种话题的参与者观点趋于中立性共识，另一种话题的参与者观点趋于两极分化的极化分歧.

我们选择在微博上下载相关数据，为了使研究信息传播的过程更具有一般性，我们信息选择过程中遵循了以下原则：

（1）在一个非特定时间从微博热搜榜上随机选取热搜话题；

（2）尽可能避免选择一些明显易受人为因素控制的话题，如：有炒作嫌疑的高阅读量榜前三、冷门话题、有明显粉丝圈子指向的话题.

基于提出的选择原则，在确定某话题是我们需要的类型后，在该话题数据中下载相关信息，信息包括的具体数据有三种：

（1）话题实时阅读量；

（2）话题实时讨论量；

（3）话题实时原创人数.

最终我们挑选出了两个话题.一个话题是#成都免费取水冰柜里的水越取越多#，开始于 2022 年 8 月 24 日，记为话题一，通过浏览评论区发现：该话题下的用户最终观点趋于温和与同化；另一个是#上海全面推行轻微违法行为不予行政处罚#，开始于 2022 年 8 月 17 日，记为话题二，该话题下的用户观点最终趋于激进与极化.

信息受众的不同情感表现等自然语言特征在“中立共识”和“两极分化”产生机制以及信息传播的影响因素中发挥着至关重要的作用，因此需要对信息受众 S 节点的情感类型进行划分，在信息传播过程中表现为用户对话题的评论.

挑选一个开始于 2022 年 11 月 19 日的新话题#大熊猫团团不幸离世#，该话题满足了 5.1.1 中我们对于话题选择的要求，截至 2022 年 11 月 20 日 18：00，该话题下的所有帖子共计有 5454 条对话题下用户的讨论发言进行收集，为了避免大量同质化发言的简单重复给收集整理工作带来的不必要麻烦，同时尽可能使数据具有代表性和典型性，选择热度最高的、有代表性的帖子，将其评论数据爬取下来，以方便后续对评论情感的处理.

5.1.2 数据预处理

5.1.2.1 话题走势数据可视化

根据话题走势，为了直观分析话题传播走势，我们绘制出三个话题的阅读量-时间坐标图（程序见附录 1），如图 1、图 2 和图 3 所示.

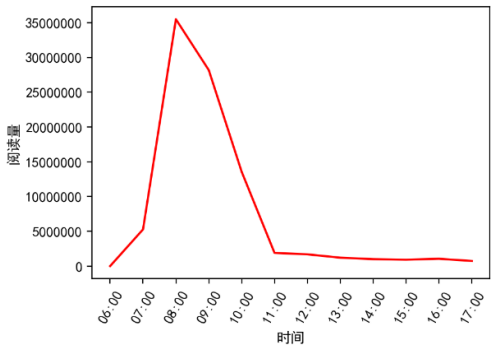


图 1：话题一阅读趋势-时间坐标图

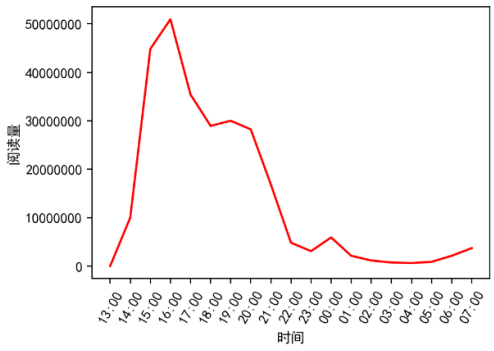


图 2：话题二阅读趋势-时间坐标图

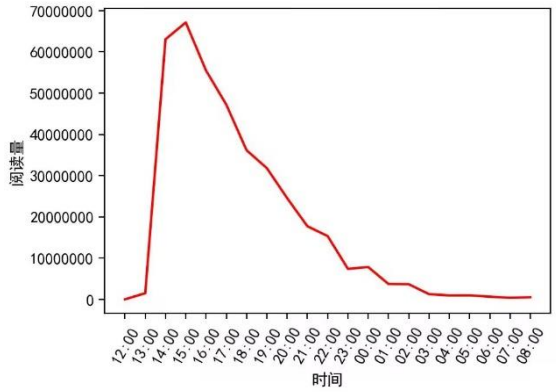


图 3：话题三阅读趋势-时间坐标图

由图 1 图 2 和图 3 可见，信息传播过程中，新信息的出现到被淹没基本都会遵循一个加速上升-减速登顶-加速下降-缓慢埋没的阅读量-单位时间变化过程.

5.1.2.2 评论数据预处理

爬取得到用户评论数据（程序见附录 3），我们将用户 id、用户发言时间、用户发言内容等汇总到表格中，整理结果如附录 17 所示.

为了后续对用户评论的情感信息进行处理，需要预先将完整的发言转化为词块的状态这里使用词袋分词算法，将收集到的评论进行拆词处理，形成词汇向量

以第一条评论“RIP 还记得当年在春晚给大熊猫起名投票“团团圆圆”” 为例，经过词袋分词算法的处理后，原句转化为词汇向量：

[RIP, 还, 记得, 当年, 在, 春晚, 给, 大熊猫, 起名, 投票, “, 团团圆圆, ”]

该词汇向量用于后续语料分析.

5.2 问题一的处理与求解

问题一要求我们建立一个关于描述信息传播过程的模型，并研究影响信息传播的一些因素.

步骤一：构造出一个针对某特定信息、能够刻画出不同用户状态的网络结构拓扑图，并使用图论语言进行关于不同状态转化的描述.

步骤二：依托于实际的用户行为，对图论模型中的有向权进行分析与数学表示.

步骤三：根据表达出来的图论模型中的有向权，建立微分方程模拟信息的传播全过程，同时对其影响因素做分析.

5.2.1 信息传播模型构建

5.2.1.1 图模型与社交网络

我们可以把社交网络理解为一个“网络结构”，即：组成网的节点是一个看作是互联网用户，而连接节点之间的线段就是用户之间的关系.因此在具体进行问题分析时，我们往往选择“节点状态”作为研究对象，根据信息传播与疾病传播上的相似性得出，针对某一特定信息，我们可以把互联网上的节点分为四类状态，如表 1 所示.

表 1：互联网四类状态节点含义表

符号	符号名称	符号含义
S	易传播节点	对于网络上的信息，用户尚未浏览到
E	潜伏节点	用户阅读过相关信息，但不会传播出去
I	感染节点	用户阅读过相关信息，且会将信息传播出去
R	免疫节点	用户已阅读过相关信息，或者对该信息无兴趣

且在任意时刻t，网络用户只能为以上四种之一.

为了研究这种网络模型的特点，并能够进一步利用其性质而求解，我们将图形转化为矩阵语言，并使用有向赋权图进行刻画.

我们能够使用有向赋权图完成达到以下目的：
（1）具体表达出各类节点之间的转化关系，如图 4 所示.

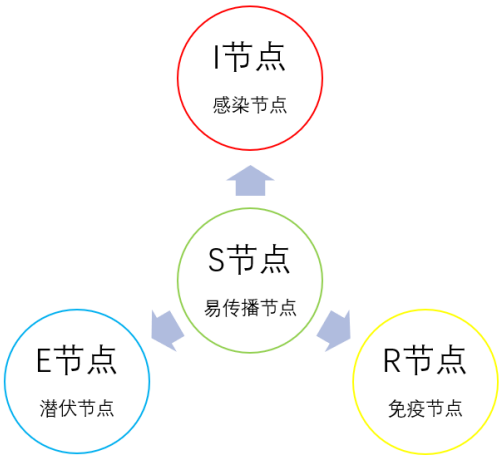


图 4：网络关联关系图

（2）完成不同状态转化的描述过程

对于已有的各节点之间转化关系，我们使用有向赋权图进行描述，即：向不同的节点定向转化过程赋予介于 0 到 1 取值的“权”，以表示这种转化可能发生的概率. 为此，我们定义有向赋权图$G =< V, E, w >$，其中：

① $V = \{S, E, I, R\}$，表示节点状态的集合；

② $E = V \times V$，表示各节点的转换关系，各边的权的集合；

③ 邻接矩阵$W = [w_{ij}] \in R^{4\times4}$，表示节点转化的概率，其中$i \in V$，$j \in V$，矩阵元素$w_{ij}$表示相应的边在矩阵中的取值，即有向边的权值，可表示为：

$$w_{ij} = \begin{cases} (0,1) & i\text{与 }j\text{间存在正向状态转换} \\ 0 & i\text{与 }j\text{间不存在正向状态转换} \end{cases}$$

因为节点状态的转化方式目前只为“S”状态向其他三种状态转化，每一种转化都存在相应的概率.因此邻接矩阵中只有w_{SE}、w_{SI}、w_{SR}三个矩阵元素不为 0，其余均为零.

5.2.1.2 权值取值表示与贝叶斯公式

为了表示出权值，我们首先要了解权值所代表的物理意义. 权值解释的是在信息传播过程中，指的是针对某种特定信息，尚未了解该信息的用户通过某种信息传播途径了解到信息，并做出不同的反应相应的概率. 因此，权值可以用不同情境下所有用户行为的概率的累加和来表示.

一般而言，网络用户接收到某特定信息的途径一般有两种：

（1）随机浏览媒体，包括但不限于哔哩哔哩平台、微博、微信公众号等，收到了系统推送. 记为事件X，其发生对应的概率为ρ；

（2）与已浏览过信息，且要将信息传播出去的人接触（I节点）. 记为事件

A，其发生对应的概率为λ.

每一个途径中也可以根据用户的行为做具体的划分. 事件X中，处于节点S的用户随机浏览到该信息后转化为E状态，即浏览信息后不会传播出去是事件X_1，概率为ρ_1；转化为I状态，即浏览信息，会进一步传播是事件X_2，概率为ρ_2；转化为R状态，即对该信息无兴趣不会浏览是事件X_3，概率为ρ_3. 事件A的分类依此类推. 分类如图 5 所示.

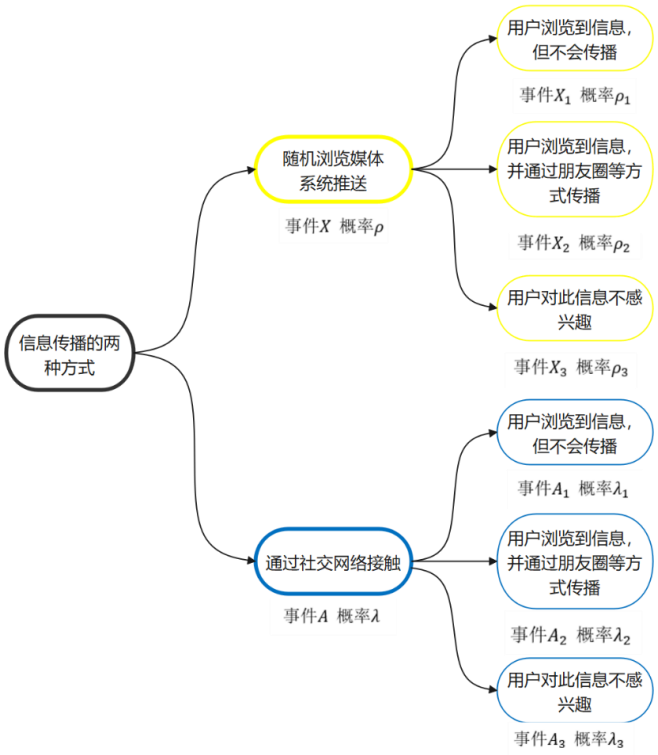


图 5：信息传播路径与行为划分图

同时可知每一种所求权值等于在不同途径下的相同行为的概率加和，根据以上事件样本划分，由贝叶斯公式可以得到：

$$\begin{cases} \rho_i = P(X_i|X) = \dfrac{P(X|X_i)P(X_i)}{\sum_{j=1}^{3} P(X|X_j)P(X_j)} & i = 1,2,3 \\ \lambda_i = P(A_i|A) = \dfrac{P(A|A_i)P(A_i)}{\sum_{j=1}^{3} P(A|A_j)P(A_j)} & i = 1,2,3 \end{cases}$$

其中，$P(X|X_i)$表示在某类途径中，已知某种用户行为发生的条件下该途径传播信息的概率. $P(X_i|X)$表示在已知某类信息传播途径发生的条件下，某种具体用户行为发生的概率. $P(X_i)$表示某类用户行为发生的概率，事件A以此类推.

因此，我们可以确定每一种状态转化的对应权值的表示方式，即：

$$\begin{cases} w_{SE} = \rho_1 + \lambda_1 \\ w_{SI} = \rho_2 + \lambda_2 \\ w_{SR} = \rho_3 + \lambda_3 \end{cases}$$

邻接矩阵$W = [w_{ij}] \in R^{4\times4}, i \in V, j \in V$可以相应确定表示为:

$$\begin{bmatrix} 0 & \rho_1+\lambda_1 & \rho_2+\lambda_2 & \rho_3+\lambda_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

且由概率的公理化定义可知：

$$\sum_{i=1}^{3}\rho_i = \rho \qquad \sum_{j=1}^{3}\lambda_j = \lambda$$

$$\rho + \lambda = 1$$

这一公式表示的是所有情况发生的概率之和是等于1的，用于约束参数的取值.

5.2.1.3 信息传播的微分方程模型

针对某一特定信息，为了描绘出在一个网络系统中从刚开始传播到该信息被淹没的全过程，以及针对该消息的各类网络节点变化情况，根据已有的邻接矩阵W，列出微分方程，因此信息传播模型可以表示为模型方程组：

$$\begin{cases} \dfrac{\mathrm{d}S(t)}{\mathrm{d}t} = -(\lambda+\rho)IS \\ \dfrac{\mathrm{d}E(t)}{\mathrm{d}t} = (\lambda_1+\rho_1)IS \\ \dfrac{\mathrm{d}I(t)}{\mathrm{d}t} = (\lambda_2+\rho_2)IS \\ \dfrac{\mathrm{d}R(t)}{\mathrm{d}t} = (\lambda_3+\rho_3)IS \end{cases}$$

和初值条件方程组:

$$\begin{cases} S+E+I+R = 1 \\ S(t_0) = 0.99 \\ E(t_0) = 0.005 \\ I(t_0) = 0.005 \\ R(t_0) = 0 \\ N = 103000000 \end{cases}$$

其中，N表示在该网络中活跃用户总数，$S(t)$、$E(t)$、$I(t)$、$R(t)$分别表示在t时刻下S、E、I、R类人群占总人数的比例，若$t = 0$，即表示在初始条件下各类人群在总活跃用户N中所占的人群比例，它们的加和为1.

5.2.2 信息传播模型求解与因素分析

5.2.2.1 信息传播模型求解

上述微分方程组是一个满足初值条件的一阶非齐次常微分方程组.对于该类问题而言，寻找其符号解有时候是困难甚至不可行的.为了对上述模型进行仿真模拟，采用龙格-库塔方法求解微分方程组的数值解（程序见附录 2）.一般情况下，4 阶龙格-库塔公式（RK4）即可达到较高的计算精度且保证高的计算效率.

将之前的微分方程组抽象为符号的形式.

$$\begin{cases} \dfrac{\mathrm{d}S(t)}{\mathrm{d}t} = f_1(I,S) \\ \dfrac{\mathrm{d}E(t)}{\mathrm{d}t} = f_2(I,S) \\ \dfrac{\mathrm{d}I(t)}{\mathrm{d}t} = f_3(I,S) \\ \dfrac{\mathrm{d}R(t)}{\mathrm{d}t} = f_4(I,S) \end{cases}$$

微分方程的解可以看作从初值条件出发，按照一定的步长h，依照某种方法逐步计算相隔步长时间的时刻的解函数的值.设置初始时刻$t=0$为t_0.前后两时刻之间的关系为$t_{n+1}=t_n+h, n=0,1,2,\cdots$在每一个时刻$t_n$,函数都有对应的取值$S_n, E_n, I_n, R_n$.

方程组求解的 RK4 由如下公式给出：

$$\begin{cases} S_{n+1} = S_n + \dfrac{h}{6}(f_{11}+2f_{12}+2f_{13}+f_{14}) \\ E_{n+1} = E_n + \dfrac{h}{6}(f_{21}+2f_{22}+2f_{33}+f_{44}) \\ I_{n+1} = I_n + \dfrac{h}{6}(f_{31}+2f_{32}+2f_{33}+f_{34}) \\ R_{n+1} = R_n + \dfrac{h}{6}(f_{41}+2f_{42}+2f_{43}+f_{44}) \end{cases}$$

首先求解$\{f_{11}, f_{21}, f_{31}, f_{41}\}$，由如下等式给出

$$\begin{cases} f_{11} = f_1(I_n, S_n) \\ f_{21} = f_2(I_n, S_n) \\ f_{31} = f_3(I_n, S_n) \\ f_{41} = f_4(I_n, S_n) \end{cases}$$

再求解$\{f_{12}, f_{22}, f_{32}, f_{42}\}$，由如下等式给出

$$\begin{cases} f_{12} = f_1\left(I_n + \frac{h}{2}f_{31}, S_n + \frac{h}{2}f_{11}\right) \\ f_{22} = f_2\left(I_n + \frac{h}{2}f_{31}, S_n + \frac{h}{2}f_{11}\right) \\ f_{32} = f_3\left(I_n + \frac{h}{2}f_{31}, S_n + \frac{h}{2}f_{11}\right) \\ f_{42} = f_4\left(I_n + \frac{h}{2}f_{31}, S_n + \frac{h}{2}f_{11}\right) \end{cases}$$

接下来求解$\{f_{13}, f_{23}, f_{33}, f_{43}\}$，由如下等式给出

$$\begin{cases} f_{13} = f_1\left(I_n + \frac{h}{2}f_{32}, S_n + \frac{h}{2}f_{12}\right) \\ f_{23} = f_2\left(I_n + \frac{h}{2}f_{32}, S_n + \frac{h}{2}f_{12}\right) \\ f_{33} = f_3\left(I_n + \frac{h}{2}f_{32}, S_n + \frac{h}{2}f_{12}\right) \\ f_{43} = f_4\left(I_n + \frac{h}{2}f_{32}, S_n + \frac{h}{2}f_{12}\right) \end{cases}$$

最后求解$\{f_{14}, f_{24}, f_{34}, f_{44}\}$，由如下等式给出

$$\begin{cases} f_{14} = f_1\left(I_n + \frac{h}{2}f_{33}, S_n + \frac{h}{2}f_{13}\right) \\ f_{24} = f_2\left(I_n + \frac{h}{2}f_{33}, S_n + \frac{h}{2}f_{13}\right) \\ f_{34} = f_3\left(I_n + \frac{h}{2}f_{33}, S_n + \frac{h}{2}f_{13}\right) \\ f_{44} = f_4\left(I_n + \frac{h}{2}f_{33}, S_n + \frac{h}{2}f_{13}\right) \end{cases}$$

经过多次迭代，即可得到在整个时间段上的微分方程数值解.

而当需要对这个模型进行仿真时，还需要设定一些初始数据和参数. 根据微博的数据统计显示，在2022年上半年其每日活跃用户数在1.03亿人左右. 于是设定总人数$N = 103000000$. 在最初的时刻一个新的话题只有小部分人群知道，即E类和I类人群所占比例很小. 且在话题传播的一开始是不存在R类人群的. 因此有如下设定：

$$R(0) = 0$$

$$E(0) = I(0) = 0.005$$

其余人群部分全部属于易传播节点，即$S(0) = 0.99$.

在模拟信息传播过程中，用D来表示一个话题的阅读量. 在传播的初始阶段，

其阅读量D_0可以表示为：

$$D_0 = (S(0) + I(0)) \cdot N$$

在第k个时间段内的阅读量D_k可以表示为：

$$D_k = (I_k + S_k - I_{k-1} - S_{k-1}) \cdot N$$

通过此方式我们便可以仿真模拟出各个时间段内的话题阅读量.

为了获取最真实的拟真效果，对于不同的话题，拟真过程的参数的调节必不可少. 对于λ_i和ρ_i的不同取值，可以得到不同的阅读量过程曲线.

首先，假设S 状态的节点通过社交网络和随机浏览接触某一信息的机会是均等的，接触到信息后转化到其他状态的概率也是均等的，即：

$$\lambda = \rho = 0.5$$

$$P(A_i) = P(X_i) = \frac{1}{6}$$

绘制阅读量随时间变化拟合图，仿真效果如图 6 所示.

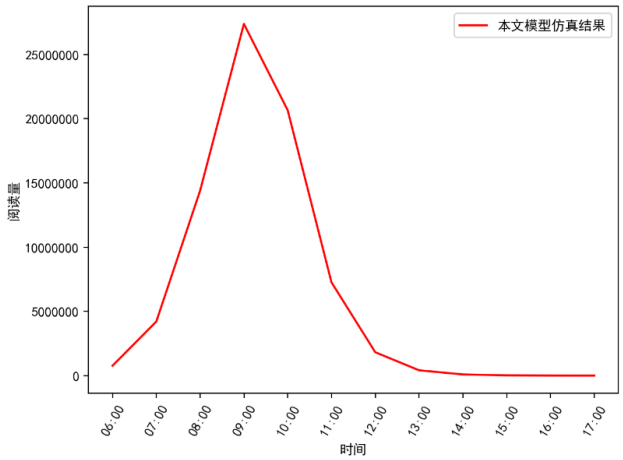


图 6：初次拟真阅读量曲线

由图像可知，仿真图线一条变化率不太平缓的曲线，这与采集到的数据相违背. 因此我们对参数进行了多次调整. 结果发现，$\lambda = 0.8$，$\rho = 0.2$时的拟真曲线效果与实际贴合程度最高. 对于#成都免费取水冰柜里的水越取越多#这个话题，针对开展实证对比实验，将仿真结果数据和真实统计数据绘制成对比分析图，其效果如图7所示.

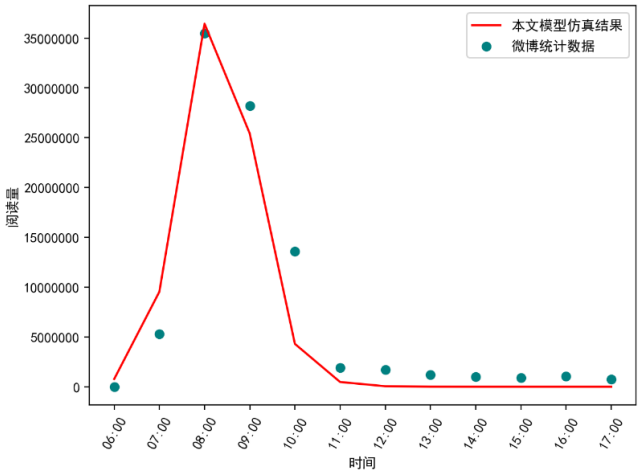


图 7：“话题 1”信息传播数据对比图

同样，对于“上海全面推行轻微违法行为不予行政处罚”这个话题进行类似的拟真过程，其拟合效果如图 8 所示.

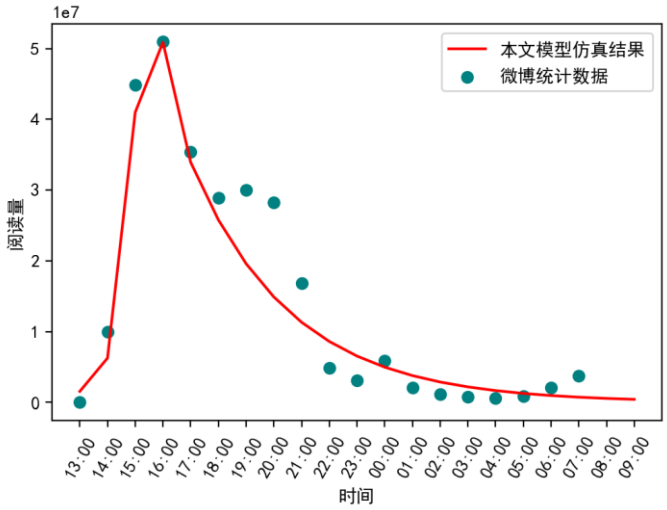


图 8：“话题 2”信息传播数据对比图

由图可知，此时选取的模型仿真数据与真实统计数据变化趋势吻合程度高，拟合性能好，较准确地反映了网络信息扩散的趋势.

5.2.2.2 信息传播过程因素分析

衡量信息传播过程的影响因素，即找出影响信息传播环节中各部分人群比例变化的因素，我们可以借助模型中的数学方程进行讨论：

（1）传播率λ

传播率λ是衡量节点I传播信息至S节点的能力强弱. 当λ值较大时，节点I可以将特定信息传播给更多的易感节点，使其成为节点E或I.

（2）网络拓扑结构

即与S状态节点相连的I节点个数，与I节点相连接的S节点数量越多，I节点更容易直接将消息传播出去. 例如：同样情况下，一个处于I节点状态的网络用户将消息传播到社群中，比私聊一次进行传播的效果要显著更强.

（3）网络用户浏览媒体信息的活跃性

活跃度更高的用户浏览社交媒体更加频繁，更易被推送到特定的信息，加快了信息的传播.

（4）用户网络安全意识或网络素质

若用户网络安全意识较好，在收到某种新消息推送，尤其是虚假信息，就会有意识进行判断，并不会轻易将消息传播出去，即：

$$\begin{cases} \rho_1 > \rho_2 \\ \lambda_1 > \lambda_2 \end{cases}$$

S节点在接触到信息后更容易转化为无传播能力的E节点，消息传播的速度受到了极大的遏制.

5.3 第二问的处理与求解

在本问中，要求我们刻画出中立性、两极分化机制的形成，同时研究“尖叫效应”、“回声室效应”以及“信息茧房”的形成过程，并研究这些机制中受到如话题吸引度等因素的影响大小.

步骤一：论用户属性对信息传播的影响，并对语料特征进行处理，将用户节点差异的因素融入到微分方程中.

步骤二：将I节点进行具体的拆分，化分为拥有不同属性I_1、I_2节点，分别代表激进者与中立者，重新使用微分方程模型计算.

步骤三：为了刻画“尖叫效应”，引入瞬间力量巨大但极短暂的脉冲函数，放入到微分方程中进行修正.

步骤四：从推荐算法的角度出发，分析网络平台如何将用户喜欢的信息进行推送并不断进行精度修正，使得信息愈发贴合用户的偏好，从而形成“回声室”闭环.

步骤五：综合宏观信息传输、推荐算法、用户行为与相互作用等方面，描绘出“信息茧房”的形成过程，并找到一系列影响因素的作用环节，将这些因

素划分为确定性因素和不确定阴损分析它们在形成“信息茧房”过程中的作用.

5.3.1 用户个人属性研究与语料处理

5.3.1.1 用户个人属性

在问题一的解决中，我们使用的是$SEIR$常微分方程动力学模型来描绘信息在网络空间中传播的宏观行为，这一方法能够有力地在宏观层面对信息传播进行刻画[2]然而，该模型在信息传播时未能考虑到系统内局部特征对总体的影响，在本题的研究中，表现为未能考虑到不同节点的差异，具体表现在：

（1）不同易感节点由于各自的属性差异而具有的潜在信息传播能力的不同.

（2）传播节点内部存在属性差异，导致 I 节点产生群体分化（具体见 5.3.2）.

（3）网络节点往往呈现社区分布在不同社区内，不同节点之间的联系会更密切.

在现实网络空间中，每个网络用户都是独一无二的，拥有各自的浏览爱好、兴趣标签、发言倾向、分享习惯、思考方式等显然，在一个大的系统内，不可能面面俱到地对每个用户的每个特征都进行较好地刻画，因为这往往是低效和高成本的，且常常对所研究问题而言意义不大.

研究某条特定消息的传播过程中，不同用户的差异对信息传播造成的影响，这里考虑的是用户的情感倾向仍然以微博平台为例，这里从某条特定话题下的用户评论开始考虑，从用户的发言中分析用户针对此消息的情感倾向，并将有类似的情感倾向的用户划分为同一子群体，并近似忽略子群体内部的所有用户间细微情感差异对传播的影响.

完成划分后的子群体由于各自的情感立场各不相同，这里认为他们在消息传播过程中的传播能力也是具有差异性的因此，假设这些子群体在接收到消息后要对外进行传播，则认为子群体间的传播系数是普遍不同的，这样就完成了对于 S 节点的初步划分，假如要化为 n 个子群体，那么可以记为.

$$S = S_1 \cup S_2 \cup \ldots \cup S_k$$

其中，$S_t \quad t = 1,2,3,\ldots,k$ 表示的是子群体，这些节点在接受到消息后，若转化为感染节点$I_t \quad t = 1,2,3,\ldots,,k$,则针对感染节点$I_t$,存在一个映射$\tau$，使得.

$$\tau(I_t) = (\lambda_t + \mu_t)$$

其中，每一个感染节点一一对应一对传播系数λ_t、μ_t，S表示总的易传播节点群

体.

再对每个细分的的子S节点群体使用$SEIR$模型模拟信息传播过程，再累加每个群体的模拟效果，即可得到考虑了节点不同属性对于信息传播的影响的修正模型.

完成以上工作的详细过程中最重要的情感倾向分析，则离不开对于语料特征分析与文本分类，将在以下 5.3.1.2 部分呈现.

5.3.1.2 语料处理

在本部分中，将要对原始的评论数据进行处理（程序见附录 4），最终完成对于用户的子群体分类大体分为加载训练集和验证集-设置特征工程-分类器分类三部分[3].

（1）加载训练集和验证集

针对预先选择好的所有评论区数据，经过数据预处理，每一条用户评论全部转化为词汇向量，这一部分已在 5.1.2 中完成，并将这些预先处理后的生成的词汇向量作为验证集.

同时准备语料库作为训练集，选择“中文情感词汇本体库”[4]并进行加载该语料库中展示了常见的 7 种情感大类词，其中共累计下属了 21 种感情小类每一个小类都有一个特定的字母编号，如“PH、NE”且下属了一系列常见中文词汇，并标明了情感强度、褒贬极性等一系列标签.

（2）特征工程设置

鉴于评论内容较为简短，且信息量集中等特点，我们着重于其中一些词汇出现和出现次数因此我们使用计数向量η作为特征，即.

$$\eta = [c_1, c_2, \dots, c_i, \dots, c_n]$$

其中，c_i表示在语料库中第i位的词汇在本评论中出现的次数，该向量的长度n等于语料库的大小.

因此，最终得到该评论的特征向量是一个高维稀疏的向量.

（3）分类器分类

常用的有监督分类方式有 KNN 近邻，支持向量机（SVM），朴素贝叶斯等由于本问中使用的“中文情感词汇本体库”中每个词汇普遍有且只有一类情感分类，这里我们近似忽略一些词汇具有的辅助情感分类因此，每个词汇都可以找到一种不同于其他词汇的独特的情感分类和情感强度，即每个词汇都有独立于其他词汇的标签特征基于这种特点，我们借用朴素贝叶斯分类器完成所有S节点的划分.其计算方法为.

$$P(l_t|D) = \frac{P(D|l_t)P(l_t)}{P(D)} \quad t = 1,2,\dots,k$$

$P(l_t|D)$表示已知该评论中的词汇属于语料库的条件下，该评论属于第t类S节点子群体的概率，$P(D)$等于该评论中所有能在语料库中找寻的到的词汇在语料库中的占比，$P(l_t)$表示第t类S节点子群体中包含的所有可能词汇在语料库词汇中的占比，$P(D|l_t)$表示该评论中词汇在某类子群体下属的词汇下的占比. 由词汇的标签特征独立性可知.

$$P(D|l_t) = \prod_{j=1}^{p_t} P(D_j|l_t) \quad t = 1,2,\dots,k$$

其中，p_t表示第t类子群体下属词汇的总数量；$P(D_j|l_t)$表示评论中从属于t类子群体的词汇的偏向强度，使用情感强度来衡量.

最终，比较某评论的在t不同取值下的$P(l_t|D)$的值，最大时对应的t值为该评论所属的S节点子群.

5.3.1.3 带有节点属性的微分方程求解

以 5.1.1 中选取的话题#大熊猫团团不幸离世#作为测试话题，选取其下由媒体《北京青年报》、《环球网》所发布的贴子作为样本，并爬取其下所有用户针对该样本帖子的评论。按照标准数据预处理方法得到所有评论的词汇向量作为测试集，并加载训练集，并通过特征工程生成所有评论向量，使用分类器进行分类后，得到的关于S节点子群分类集$S_1,\dots,S_k$, 对应情感标签、总S节点占比，绘制饼状图（程序见附录 5），如图 9.

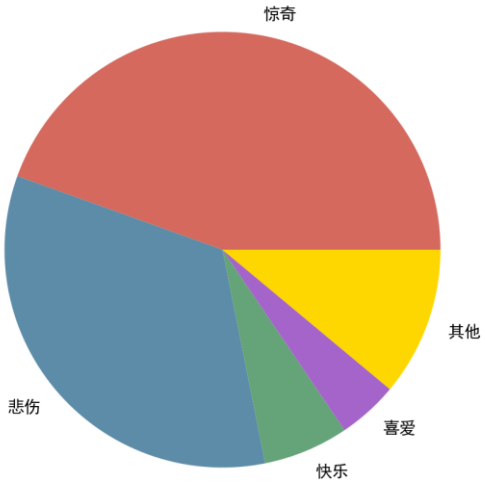


图 9. 分类器人群分类结果饼状图

分别以每类小群体为初始易传播人群，分别赋予不同的传播系数，使用第一问微分方程传播方法求解(程序见附录 6)，分别得到阅读量曲线图 10-14.

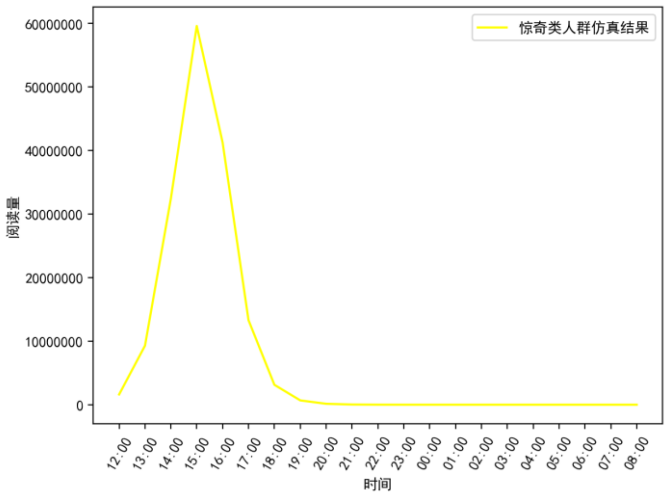


图 10：惊奇类人群$SEIR$模型仿真结果

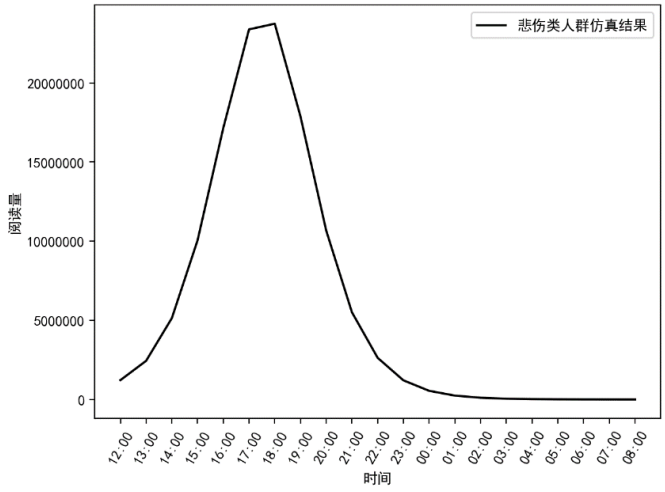


图 11：悲伤类人群$SEIR$模型仿真结果

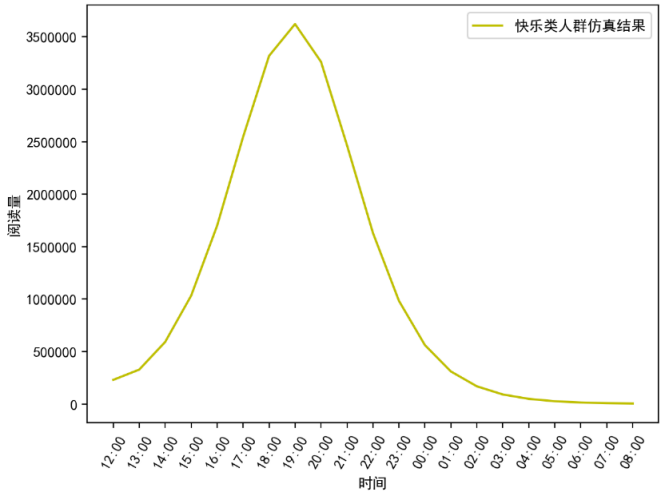


图 12：快乐类人群$SEIR$模型仿真结果

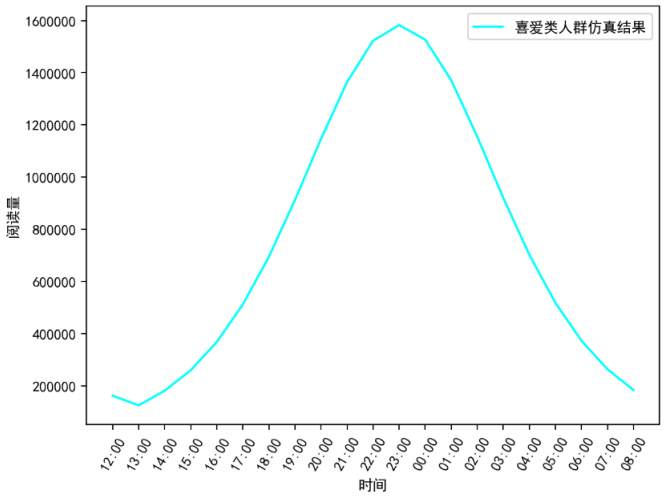


图 13：喜爱类人群$SEIR$模型仿真结果

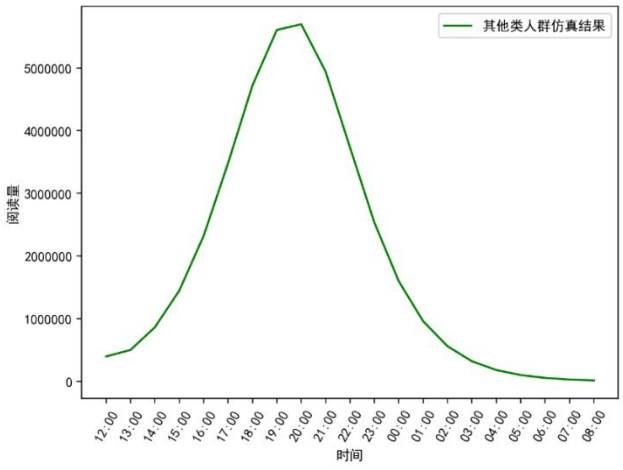


图 14：其他类人群$SEIR$模型仿真结果

再将所有小群体内的阅读量曲线图合并，得到如下的总阅读量曲线图 15.

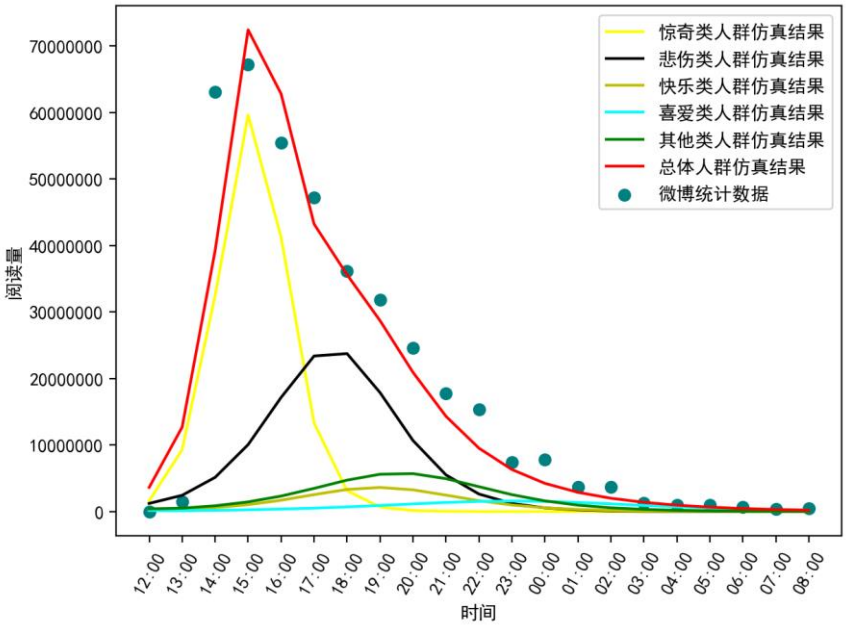


图 15. 总体人群、各类人群阅读量曲线图对比

5.3.2 中立机制与极化机制刻画

上文完成了对于用户个体不同属性的讨论，并针对易传播节点依照用户特性进行了划分。实际情况中，话题本身的性质也会对传播造成影响。那么考虑：当话题本身存在中立或两极分化的语料特点时，是否也会一定程度上影响信息传播的过程.

首先，中立与两级分化指的是话题讨论趋于的两种不同发展趋势，而信息传播过程中这两种现象的出现，与话题吸引度有极大关系，主要由浏览过该话题信息的网络用户，产生不同观点，且向外传播导致. 因此为了刻画这两种机制，应以问题一建立起的信息传播模型为基础进行分析，并根据实际情况进行复杂化.

（一）拆分 I 节点为I_1和I_2节点，具体含义如表 2 所示.

表 2：I_1，I_2节点含义表

符号	符号含义
I_1	用户阅读过该信息后产生激进的极化观点，且将其传播出去
I_2	用户阅读过该信息后产生温和的中立观点，且将其传播出去

根据现实中的网络信息传播，I_1节点将激进的观点传播出去后，若有S节点受

到其接收到该信息，并受到其影响也意图继续将信息传播出去，即转化为I节点，则该新I节点必然是同样携带激进的观点，即为I_1节点. 同理，I_2节点传播信息后，接受到其信息的S节点若转化为I节点，则认为是I_2节点. 长此以往，I_1，I_2节点不断向外“传染”，且形成分别由I_1、I_2所主导的“种群”，最终在某话题下占据优势比例的“种群”，能够在整个信息网络中主导最终观点的走向.

（二）引入新系数话题吸引度μ

一个话题的传播能力与其吸引度息息相关，随着话题吸引度的变化，由S节点转化到I节点的概率会发生变化. 该系数可以表示话题吸引度对S状态到I状态转化的概率的影响，话题吸引度越高，发生转化的概率越大. 且中立和极化两种观点不同，传播时的话题吸引度也不同，因此对应的话题吸引度系数也不相同.

将μ引入到微分方程中，新的信息传播模型可以表示为模型方程组：

$$\begin{cases} \dfrac{\mathrm{d}S(t)}{\mathrm{d}t} = -(\mu_1+\mu_2)(\lambda+\rho)IS \\ \dfrac{\mathrm{d}E(t)}{\mathrm{d}t} = (\lambda_1+\rho_1)IS \\ \dfrac{\mathrm{d}I_1(t)}{\mathrm{d}t} = \mu_1(\lambda_2+\rho_2)IS \\ \dfrac{\mathrm{d}I_2(t)}{\mathrm{d}t} = \mu_2(\lambda_2+\rho_2)IS \\ \dfrac{\mathrm{d}R(t)}{\mathrm{d}t} = (\lambda_3+\rho_3)IS \end{cases}$$

和初值方程组：

$$\begin{cases} S+E+I+R=1 \\ S(t_0) = 0.99 \\ E(t_0) = 0.005 \\ I(t_0) = 0.005 \\ R(t_0) = 0 \\ N = 103000000 \end{cases}$$

其中，N表示在该网络中活跃用户总数，$S(t)$、$E(t)$、$I(t)$、$R(t)$分别表示在t时刻下S、E、I、R类人群占总人数的比例，若$t=0$，则表示在初始条件下各类人群在总活跃用户N中所占的人群比例，它们的加和为1.

（三）数值仿真与拟合

当关于一个话题人们的极端观点具有更强大的吸引力时，这个话题往往容易走向极端. 极化话题情况下，I_1类人群占I的比例会更大一些. 因此提出：

$$p=\frac{I_1}{I_1+I_2}$$

以此刻画话题的极端情况. 当p接近于 1 时，话题的极端化程度越大. 若设定

$\mu_1 = 0.8$，$\mu_2 = 0.2$，得到I_1所占比例p变化曲线（程序见附录 9），如图 16 所示：

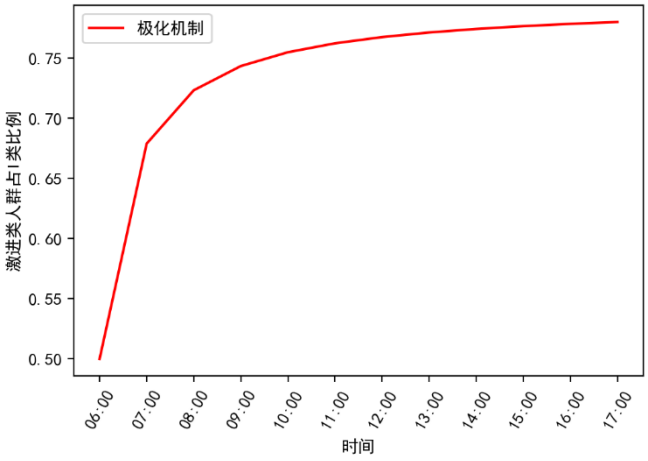


图 16：激进人群所占比例变化

随着时间的增长，I_1类人群所占比例急剧增大，接近0.8，此时持有激进极化观点的I_1种群在该话题下占据了主导地位，说明该话题最终演化为一个极化话题.

当一个话题的讨论中，人们的温和观点具有更强大的吸引力时，这个话题往往容易走向中立. 当p接近于0时，话题温和化程度越大. 如果设定$\mu_1 = 0.16$，$\mu_2 = 0.84$，我们可以得到I_1所占比例p变化曲线，如图 17 所示.

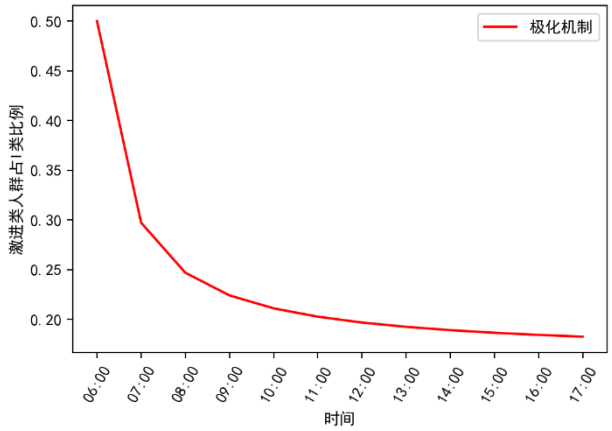


图 17：激进人群所占比例变化

随着时间的增长，I_1类人群所占比例急剧减小，接近0.16.此时持有温和中立观点的I_2种群在该话题下占据了主导地位，说明该话题最终演化为一个中立话题.

5.3.3“尖叫效应”形成机制

在信息传输领域，尖叫效应指的是突然性输出大量引人注目的内容，从而实现吸引他人注意力并将信息传播出去的目的. 这类效应在目前的自媒体平台等十分常见. 例如：某些特定新闻在传播初期，借助平台的推流、某些大 V 的有意宣传，再辅以恶搞性或者猎奇性的标题与内容，从而实现阅读量的快速增长. 这种

效应使得某些特定信息在传播初期短时间内的阅读量增长远大于常规信息传播规律，从而使得信息的一些影响效果发生陡然不同的变化.

5.3.2.1 效应特征与刻画

一般而言，“尖叫效应”的实现离不开两个关键性因素：

（1）突然性

在某一个时间点，某些信息忽然出现在信息传播系统中，这些信息是全新的. 这也意味着，尖叫效应往往是某类特定信息开始向周围扩散的开端.

（2）高频率性

尖叫效应必须是在短时间内系统大量迸发某特定信息. 只有在这种情况下，单位时间内的信息强度和密度才会足够强大，使得信息传播的影响力足够大.

在这两种因素的互相促进下，“尖叫效应”便产生了.

为了刻画这两种特征对于常规信息传播模型的影响，我们引入新的函数来对第一问的模型进行修正. 首先，我们引入δ函数[5]：

$$\delta(t-t_0)=\lim_{\varepsilon\to 0}\delta_\varepsilon(t-t_0)$$

其中，ε为任意大于0的整数.

定义脉冲函数：

$$(1)\,\delta_\varepsilon(t-t_0)=\begin{cases}1/2\varepsilon & t_0-\varepsilon<t<t_0+\varepsilon\\ 0 & else\end{cases}$$

$$(2)\int_0^\infty \delta_\varepsilon(t-t_0)\,dt=1$$

当ε无限趋于零时，在信息爆发的某个t_0时刻的无限小邻域内，脉冲函数的取值无限趋于正无穷，这样我们就完成了“尖叫效应”的特征刻画. 通过计算机仿真（程序见附录 8）可得到一条阅读量随时间变化曲线，如图 18 所示.

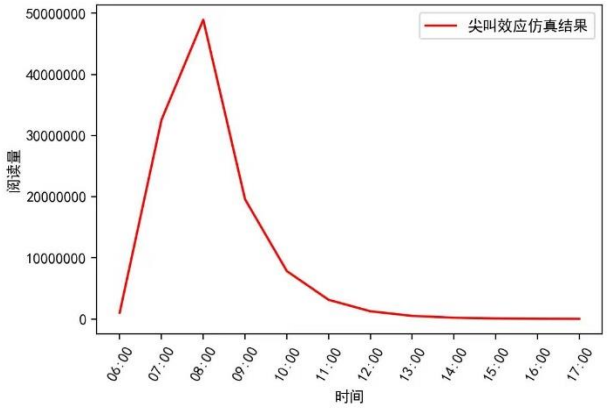


图 18：尖叫效应效果图

可以看到，相较于之前的拟合曲线，加入尖叫效应效果后在话题传播的初期，其阅读量增长率有了一个明显的增长. 话题在初始阶段的传播能力得到了巨大的加强. 虽然在此之后阅读量随时间还有一段的小幅度增长，但其之后就进入到了下降阶段.

在现实生活中，经常会产生这样的话题，一出现受到了大众群体的热烈关注. 这些话题在信息传播的初期，I节点人数占比相较其他话题多，此时尖叫效应便产生了. 我们发现之前一般的模型仿真往往无法较好贴合这种实际情况，而使用脉冲函数则能够改良原先的模型. 这就是“尖叫效应”的应用效果.

5.3.2.2 前期爆发传播模型

“尖叫效应”的数学表示通过影响微分方程的已知函数组成这一方式.

一般而言，在信息传播的初始阶段，“尖叫效应”的爆发会在短时间内大量消除社交网络中易感节点S的数量. 按照生活经验，绝大多数看到信息的网络用户还会将信息进行对外转发，即转化为I节点，少部分阅读到信息的网络用户选择浏览但不转发.

综上，我们对微分方程中$I(t)$与$E(t)$的变化率做出修正：

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -(\lambda+\rho)IS - N\sum_{i=1}^{m}\delta(t-t_i)$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = S(\lambda+\rho)I + N\sum_{i=1}^{m}\delta(t-t_i)$$

可以看出，随着时间戳远离信息爆发的时间节点，脉冲函数的影响程度愈发减弱. 因此，在不同时间戳上，脉冲函数的强度是不同的.

5.3.4“回声室效应”形成机制

回声室效应的形成主要是因为系统推荐算法根据用户的喜好，不断向某用户推向自己偏好的内容，且用户对于其偏好的内容浏览强度不断上升，进而在推荐算法的推动下接收到更多的喜好内容推荐. 在这种正反馈调节的长期作用下，用户几乎最后只能收到自己所偏好的信息推荐. 研究该形成机制，主要分为以下几个步骤：

步骤一，刻画出系统是如何学习用户的喜好，了解用户的习惯并做出预测.

步骤二，建立系统如何给用户推荐信息、并根据用户反馈进行修正的核心模型，这里一般认为是使用主流的“协同过滤”模型.

步骤三，根据以上两个过程，描绘出“回声室效应”形成闭环的动态过程.

（一）基于随机生成的ALS矩阵分解算法[6][7]

在机器学习领域，尤其是大数据下关于人的行为与偏好特征的学习，一般是通过构建出一个集成了多个用户信息的“用户偏好矩阵”，矩阵中记录着用户们的偏好信息，这些信息一般是通过用户对某些特定指标进行打分来体现，可以将该矩阵记为用户喜好特征矩阵U：

$$U = [u_1, u_2, \dots, u_j] \in R^{k \times j}$$

其中，u_x表示用户x的评分向量 $(x \in [1, j])$，其是一个k维列向量.

同时，为了确定出究竟什么样的信息适合被推送给用户，为了定量地衡量出是否“合适”，这就涉及到对于信息的评价打分，打分一般是通过构建一个物品偏好矩阵，矩阵中记录着某些特定信息关于某些特定指标的得分，可以将该矩阵记为产品特征矩阵M：

$$M = [m_1, m_2, \dots, m_p] \in R^{k \times p}$$

其中，m_y表示用户y的评分向量 $(y \in [1, p])$，其是一个k维列向量.

为了完成一个消息的推送，在确定好用户的喜好特征和信息的特征后，最终要确认的是一些特定信息中对哪个更具偏好，为了定量表征，引入关于信息的用户打分矩阵R：

$$R = [r_{xy}] \in R^{j \times p}$$

其中，矩阵中的指标r_{xy}表示第x个用户对于第y个信息的打分.

在本模块中的所有打分都是介于0到10之间的整数，且打分越高，表示偏好性越强.

一般来说，系统开发者不可能完全拥有关于所有用户对于所有类型的信息的打分矩阵，一定是存在相当数量的打分信息空白，即：矩阵R中存在大量空白元素，只有将这些空白信息补全，才能进行下一步的信息推送工作. 为了达到该目标，我们使用 ALS 矩阵分解算法.

（二）ALS 矩阵分解算法的实现

ALS 的核心基于假设：打分矩阵R是近似低秩的. 即：一个打分矩阵R可以用两个小矩阵U和M的乘积来近似，通过该方法就把这个打分系统的自由度降低了. 用户的喜好和电影的特征都投到一个低维的空间，用户的喜好特征映射到了一个低维向量，一个电影的特征变成了纬度相同的向量，那么用户和这个电影的相似

度就可以表述成这两个向量之间的内积.把打分理解成相似度，那么“用户打分矩阵”就可以由“用户喜好特征矩阵U”和“产品特征矩阵M”的乘积来近似.

如上所述，使用 ALS 算法的目的就是将打分矩阵R拆解为U和M的乘积形式.根据U^TM所得到的矩阵来模拟之前打分矩阵R.这样就可以将原来矩阵R的元素进行填充并把填充后的评分进行排序.协同过滤系统会根据评分排名为用户推荐评分高的产品.

把一个充满空白元素的矩阵进行分解就是 ALS 算法的精华所在.ALS 全称为交替最小二乘法.其是使用迭代的方法，在每一个迭代过程中使用目标函数进行非线性规划，将两个原始矩阵的乘积逼近到理想的结果.

根据定义可知：

$$R_{j\times p}\approx U_{k\times j}^{T}M_{k\times p}$$

为了使U^TM接近于矩阵R，定义目标函数：

$$Z=\min_{U,M}\sum_{r_{xy}\neq 0}\left(r_{xy}-u_x^Tm_y\right)^2+\lambda\left(\sum_x\|u_x\|_2^2+\sum_y\|m_y\|_2^2\right)$$

其中$r_{xy}\neq 0$表示第x个用户已经给第y个物品进行了打分.为了保持数值计算稳定性，防止过度拟合，在目标函数中引入正则化参数λ.

至此，将 ALS 算法转化为一个非线性规划问题，将 U 和 M 中的列向量视作决策变量其中列向量中的每个元素取值应该位于$(0,10)$的区间内

首先随机生成一个大小为$k\times p$矩阵并赋予M.但是要求矩阵M中任意元素m_{ij}的取值为 0 到 10 之间.现将矩阵M固定来优化矩阵U中的列向量

$$Z=\min_{u_x}\sum_{r_{xy}\neq 0}\left(r_{xy}-u_x^Tm_y\right)^2+\lambda\sum_x\|u_x\|_2^2$$

先将目标函数转化为矩阵表达式

$$J(u_x)=(R_x-M_x^TU_x)^T(R_x-M_x^TU_x)+\lambda u_x^Tu_x$$

在极小值位置处目标函数关于所有变量的导数为 0.因此，将J关于所有u_x求导并令导数为0（$x\in[1,j]$）.

$$\frac{\partial J(u_x)}{\partial u_x}=-2M_x(R_x-M_x^Tu_x)+2\lambda u_x=0$$

求解可得

$$u_x = (M_x M_x^T + \lambda I)^{-1} M_x R_x, \quad x = 1,2,\dots,j$$

从而求得在最优解下矩阵U所有元素的值. 这样就完成了第一次迭代. 再用求解得到的U作为已知矩阵，矩阵M中的列向量视为决策变量，使用相同的规划模型去计算新的矩阵M. 同理可推倒出结果

$$m_y = \left(U_y U_y^T + \lambda I\right)^{-1} U_y R_y, \quad y = 1,2,\dots,p$$

再用新矩阵M为已知量，继续进行规划. 迭代多次后就可以得到最后理想的矩阵U和M.将两个矩阵相乘得到预测打分矩阵，即：

$$R_{j\times p} \approx U_{k\times j}^T M_{k\times p}$$

这便是 ALS 矩阵分解算法实现的基本过程.

（三）动态描述与仿真（程序见附录 7）

首先找到一组用户对一组话题的打分情况表. 为了讨论信息茧房效应形成的原因，将所有话题大概分成了“追星”“军事”“体育”“电视剧”四大类. 对于每一类话题，找到了 3 位用户对本话题打高分且对其他话题没有打分. 而仿真的目的就是为了探究每一类话题的喜欢阅读的用户其属于其他类别的话题和对本类别话题的打分情况是否有显著性差异, 如表 3 所示.

表 3：初始打分表

	爱豆1	爱豆2	爱豆3	军事1	军事2	军事3	政治1	政治2	政治3	电视剧1	电视剧2	电视剧3
用户1	9		9									
用户2		9	9									
用户3	9	9										
用户4				9		9						
用户5					9	9						
用户6				9	9							
用户7							9	9				
用户8								9	9			
用户9							9		9			
用户10										9	9	
用户11											9	9
用户12										9		9

经过对 ALS 模型的计算，最后得到的预测打分结果如表 4 所示.

表 4：仿真预测打分表

	爱豆1	爱豆2	爱豆3	军事1	军事2	军事3	政治1	政治2	政治3	电视剧1	电视剧2	电视剧3
用户1	3.935216	3.935216	3.935216	0.793581	0.793581	0.793581	-0.77555	-0.77555	-0.77555	2.362708	2.362708	2.362708
用户2	3.935216	3.935216	3.935216	0.793581	0.793581	0.793581	-0.77555	-0.77555	-0.77555	2.362708	2.362708	2.362708
用户3	3.935216	3.935216	3.935216	0.793581	0.793581	0.793581	-0.77555	-0.77555	-0.77555	2.362708	2.362708	2.362708
用户4	0.793581	0.793581	0.793581	5.302942	5.302942	5.302942	0.355458	0.355458	0.355458	-1.08291	-1.08291	-1.08291
用户5	0.793581	0.793581	0.793581	5.302942	5.302942	5.302942	0.355458	0.355458	0.355458	-1.08291	-1.08291	-1.08291
用户6	0.793581	0.793581	0.793581	5.302942	5.302942	5.302942	0.355458	0.355458	0.355458	-1.08291	-1.08291	-1.08291
用户7	-0.77555	-0.77555	-0.77555	0.355458	0.355458	0.355458	5.319287	5.319287	5.319287	1.058296	1.058296	1.058296
用户8	-0.77555	-0.77555	-0.77555	0.355458	0.355458	0.355458	5.319287	5.319287	5.319287	1.058296	1.058296	1.058296
用户9	-0.77555	-0.77555	-0.77555	0.355458	0.355458	0.355458	5.319287	5.319287	5.319287	1.058296	1.058296	1.058296
用户10	2.362708	2.362708	2.362708	-1.08291	-1.08291	-1.08291	1.058296	1.058296	1.058296	2.442555	2.442555	2.442555
用户11	2.362708	2.362708	2.362708	-1.08291	-1.08291	-1.08291	1.058296	1.058296	1.058296	2.442555	2.442555	2.442555
用户12	2.362708	2.362708	2.362708	-1.08291	-1.08291	-1.08291	1.058296	1.058296	1.058296	2.442555	2.442555	2.442555

由表 4 可知，在预测打分情况中.算法对于每个用户还是对于之前关注的话题更加感兴趣，打分水平跟高.而对于其他的话题其打分情况有着明显的降低的状态.也就是说，用户看到之前感兴趣的话题的新的消息的概率势比别的信息群体大很多的.这就是 ALS 矩阵分解算法的基本原理.

以信息#上海全面推行轻微违法行为不予行政处罚#为例，如果要将信息推送给目标用户并形成回声室闭环，要经过如下过程：

（1）通过 ALS 迭代算法形成了R矩阵，计算得到了用户对于此条信息的打分.

（2）若超过预先设定的打分阈值，那么将该消息推送至用户；否则不推送.

（3）系统将信息推送至用户，并记录用户针对此信息的一系列行为，如：浏览时长、浏览次数等.综合该信息的一些属性指标，如：时效性、趣味性等，对用户偏好矩阵中的打分作出修改，并重新计算矩阵中的一系列打分.

（4）重复流程，直到推送的信息类别都类似用户偏好的信息.

流程如图 19 所示.

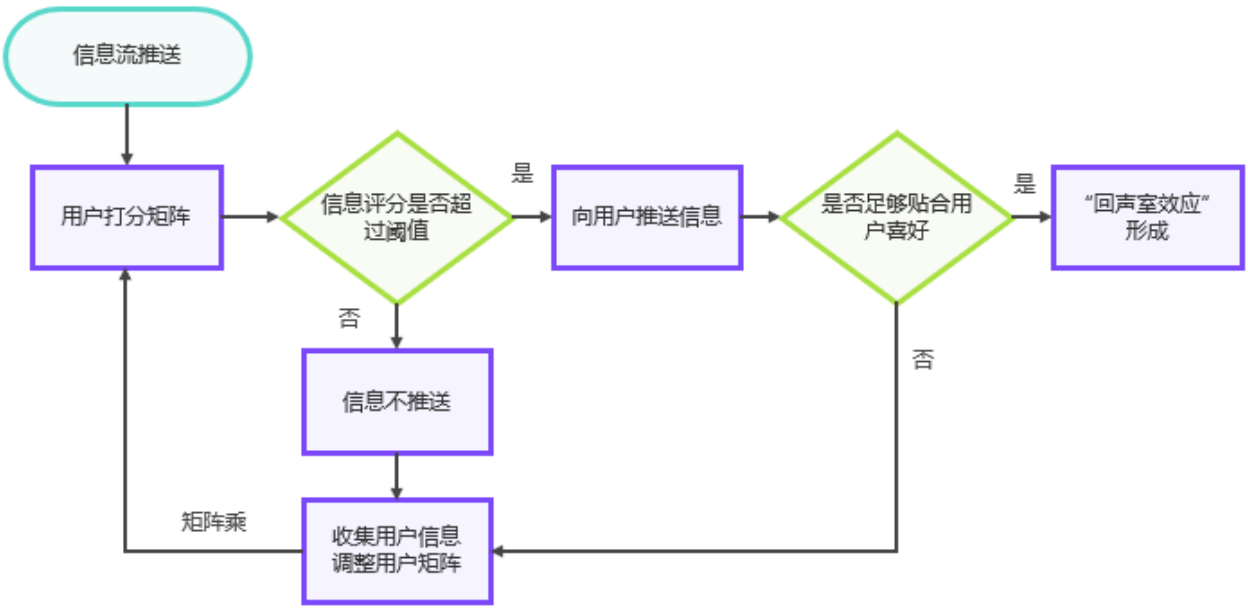


图 19：回声室形成流程图

5.3.5“信息茧房”的形成与多因素分析

（一）“信息茧房”的形成：

“回声室效应”与“尖叫效应”的共同作用是“信息茧房”被搭建起来的诱因，其形成主要分为以下几个步骤：

（1）由“尖叫效应”开始，向某一用户短时间内推送大量信息，即信息大量从I节点扩散到S节点的过程.用户根据自己的偏好，在信息流中进行选择性浏览，一段时间后形成一个初步的不完全用户打分矩阵R_0.

（2）系统根据不完全用户打分矩阵R_0，在后台使用 ALS 迭代计算出一个初步的完整用户打分矩阵R_1,并根据事先设定好的阈值，决定向某用户推送何类的信息.

（3）系统记录下用户得到推送的信息后的一系列行为特征，如浏览时间等，并根据评价指标对用户喜好特征矩阵U进行修改，重新改变某些打分，得到新的矩阵U.

（4）根据公式$R = U^T M$,更新用户打分矩阵R_1,得到R_2再次向用户推送偏好信息.

（5）循环步骤（3）（4），R矩阵趋向于稳定，此时用户得到的信息几乎是无偏于其喜好的，此时形成了一个“信息茧房”.

流程如图 20 所示.

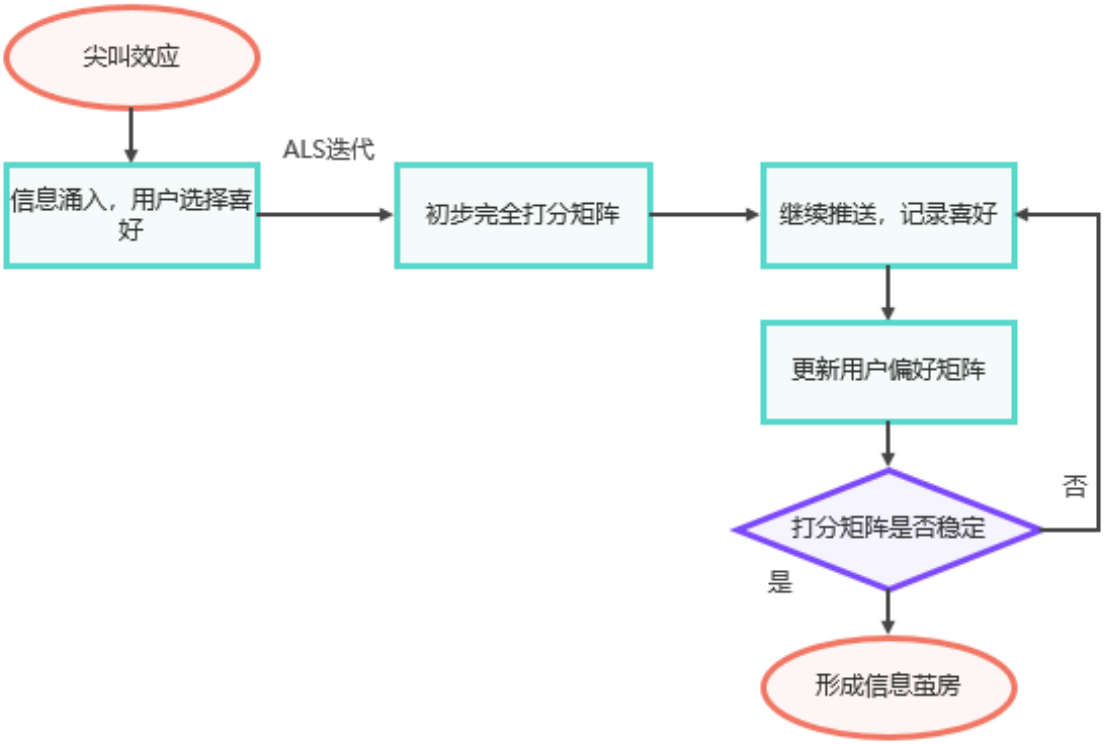


图 20：信息茧房形成流程

（二）“影响因素”分析：

促成信息茧房形成的因素有很多。有一些是由于媒体信息特性、网络空间结构、平台设计算法等事先预设好的因素，这些因素可能对信息传播造成的影响是

在某一条消息开始传播前即可预知的，这样的一些因素我们理解为确定性因素。另外一些因素，如用户的网络习惯与心理，用户社区差异等，这些因素对信息传播这一宏观过程的影响往往是非线性的，又对一定程度上依赖于一些其他因素的影响，且并不在各个信息传播系统中完全体现出来，这样的因素我们理解为非确定因素。

（1）确定性因素：

①话题吸引度：

话题吸引度μ用于描述一个话题是否更容易激发起用户的浏览兴趣.倘若话题吸引度μ更大，根据 5.3.2 中的讨论，I节点的传播性将更强，用户的信息浏览行为会更多.这意味着在茧房形成的第一步中，初代R_0矩阵的信息的空白会减少，系统在 AIS 迭代的过程中可以更精确地识别用户偏好，从而加快了“信息茧房”的形成.

②平台推荐算法：

在目前的主流媒体平台，如微博、B 站、抖音等，在推荐信息系统上不断设计优化信息推送机制，这些算法能够基于事先学习好的目标用户的特点，定向推送其更为感兴趣的内容。以 2012 年在 Facebook 的 F8 开发者大会上提出的 EdgeRank 排序算法[8]。该算法使得平台在向用户推送信息时，综合信息源与用户亲密度、信息质量、信息时效性三个维度，决定该信息被优先推送的概率，即传播系数ρ的大小。若某信息来自于节点用户关注的话题或博主、且该信息具有优质的内容或符合用户的偏好习惯、且为不久发生的事，则该消息极易被优先推送给用户，这一操作使得“尖叫效应”迸发出的大量信息得到了筛选，更快地投“用户”所好。同时，用户对这些感兴趣的内容浏览频率会增大，加速了系统的学习速度，从而加快了“信息茧房”的形成.

③节点网络：

将节点连接的边数定义为节点的度。节点的度越大，说明与某节点存在联系的其他节点数量越多，这里认为某节点的度越大，网络传播影响力越大[9]。假定某消息从一个度非常大的节点开始传播，那么能够在一个较短时间内蔓延至周围大量的网络节点，即消息的初始传播速度会非常巨大，尖叫效应便形成。

（2）不确定因素；

①用户活跃度：

用户活跃度表征的是用户进行信息浏览与参与到网络讨论中的概率.用户越活跃，意味着S节点向I节点转化的概率越大.对于单个用户而言，效果与话题吸引度类似，最终会加快其个体的“信息茧房”的形成.但是对于一群网络用户而言，高活跃度的用户群意味着群体内部会经常传播不同类别的信息，反而利于打破群体内部其他用户的“信息茧房”.

②用户心理：

用户心理决定着用户面对信息所做出行为. 例如，若一个用户有凑热闹、从众心理，那么就会更容易浏览被推送的信息，同时面对他人输出的一些观点（尤其是激进类观点）更容易跟从接受，即更容易转化为E, I状态，使得其初代用户打分矩阵R_0更加完善. 因为其从众心理，所以对于被推送到的偏好内容会更容易产生认同感，其产生的用户行为数据加快了R矩阵迭代的速度. 最终“信息茧房”会更快更准确地形成.

③不同用户间互相影响：

用户活跃度在群体内部对于其他用户的影响可以被看作是一个典型的用户互相影响案例. 此外，I_1节点特有的激进特性，更会使S节点产生盲从心理，与I_2传播的观点相比较，更会使一些用户偏好所被系统推荐的内容，从而快速形成稳定的R矩阵，这意味着“信息茧房”会更快地形成.

5.4 第三问的处理与解决

第三问要求我们给出合理的方案，能够破除“尖叫效应”与“回声室效应”的影响，进而打破“信息茧房”的桎梏.

步骤一：从“尖叫效应”的特征分析，有意避免此类情景下的信息浏览.

步骤二：从“回声室效应”的一些成因和影响因素分析，提出合理的实际举措.

5.4.1 “尖叫效应”的破除

“尖叫效应”有两大特征，突然性与高频率性. 前者决定了信息的爆发是仓促而无征兆的；后者决定信息是重磅与高强度的. 满足“尖叫效应”的情景，例如：某大 V 忽然爆出 XX 重磅消息、突然发生重大网络事件等. 在这种情景下，对这些短暂而高频率的信息保持一个清醒和理智的判断是极为重要的，保持原有的认识，不要刻意去浏览一些带有强烈吸引性标题的帖子（往往是哗众取宠）等. 这些都是从用户角度可以避免被“尖叫效应”带来的强信息流影响的一些方法.

同时，基于算法设计等宏观层面，也有能够破除“尖叫效应”的有效手段，下面介绍的是基于流量优化的“限流算法”与基于控制的“定向免疫”策略。

5.4.1.1 “尖叫效应”识别

在施行优化和控制策略之前，需要系统识别出某消息的传播存在“尖叫效应”的传播趋势。识别方法主要基于信息传播速度的监测和基于信息内容的识别两方面。

首先是对目标信息进行信息传播速度监测，这一方法可以有效遏制特定信息的大规模传播。如 5.2.2 中所提到的，在进行微分方程求解的过程中，将不同时间戳下的节点比例进行离散化，按照步长进行递推求解。因此，利用任意一个步长两端的时间戳上易传播节点比例，计算变化量，根据“尖叫效应”的爆发性增长的特征，来检测其存在。当走过某次步长，比例变化大于事先设定的阈值，则认为可能有“尖叫效应”出现，需要进一步判断确定，于是需要进入下面的信息内容识别。

其次是信息内容的识别，可以有效遏制恶意信息的散播。与模块 5.3.1 中的语料识别的方法类似，识别恶意信息步骤主要为：

（1）对信息内容进行预处理，使用词袋模型，将其转化为词汇向量，形成验证集。

（2）加载语料库，可以选择面向文本主题分类、消息属性、隐含义发掘的语料库，形成训练集.

（3）设置特征工程

（4）使用分类器进行分类

如流程图 21 所示.

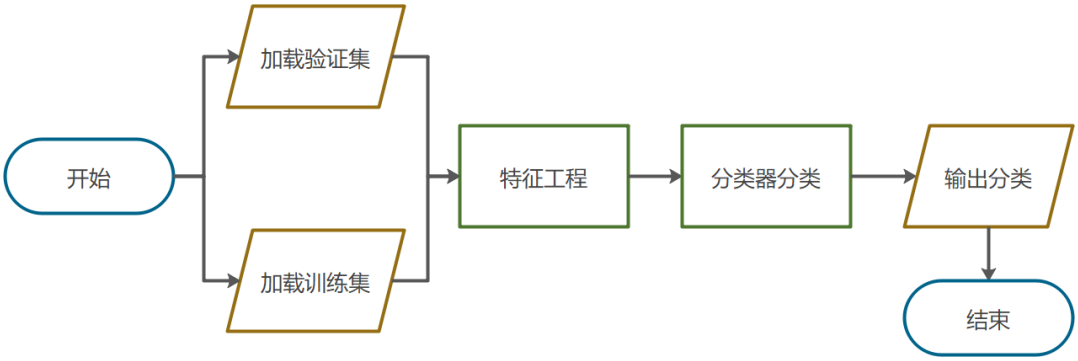


图 21:信息内容识别判断尖叫效应

具体计算原理与过程与 5.2.2 类似，在此不做详细展开.

经过以上两步的识别，若发现某信息是恶意信息、或其信息流量过大可能存在异常，则启动相应的优化或控制策略进行遏制.

5.4.1.2“尖叫效应”的算法应对策略

（1）基于流量优化的限流算法

限流算法作为一种在各大平台广泛应用的舆论控制方案，在宏观层面的流量的控制上具有较强的有效性。目前，常见的限流算法主要有固定窗口、滑动窗口、漏桶算法、令牌桶算法[10].这里选用固定窗口算法进行限流实现，算法处理的核心是优化调整流量的变化率，使不明的爆发消息变得温和或一般化、或对恶意

信息进行压制或完全压制（程序见附录 10）.

滑动算法仍然基于 5.2.2 微分方程的求解过程，其算法实现为：

①记录初始时间戳 t0，待限流段总长度$length$，$length = (t_0 + n) - t_0, n = 1,2,\ldots$

②记录时间戳t其后一步长内的比例变化量，记为ΔS_t，表示从时间戳t到$t+1$这段步长内，S节点比例变化了ΔS

③令$\Delta R = (1 - \frac{1}{e})\Delta S$ $\Delta S = \frac{\Delta S}{e}, e = 1,2,3\ldots$，当$e$取足够大时，$\Delta S$趋向于 0

④令$t = t + 1$，进入下一步长内，跳入步骤 2；若完成总长度内所有步长的限流，则停止.

其流程图如图 22.

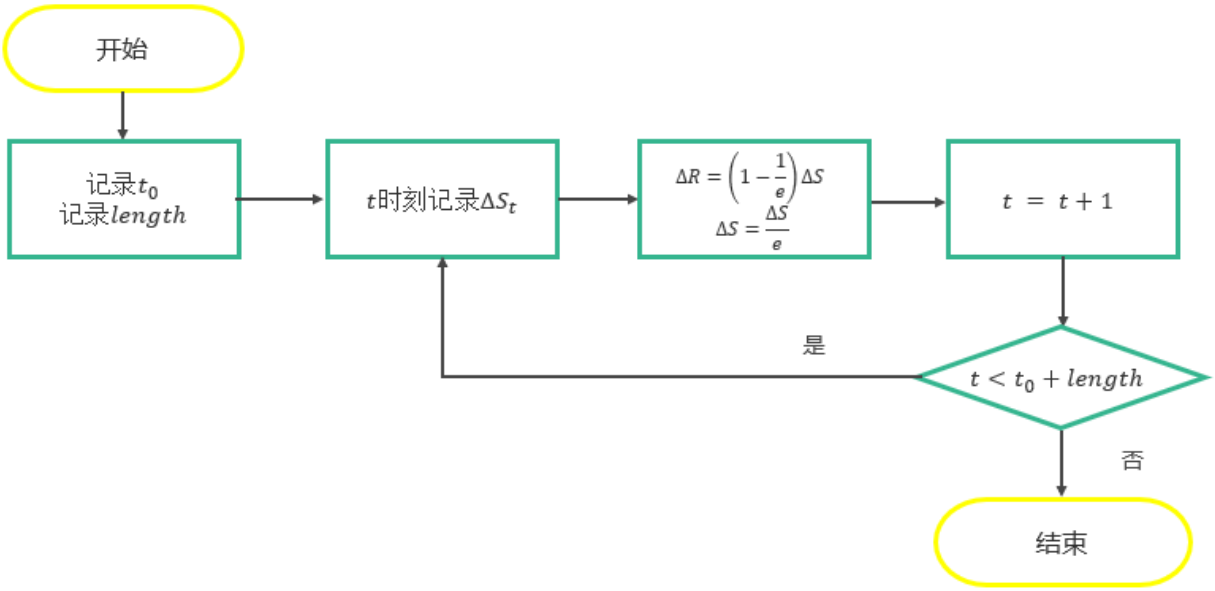


图 22：流量优化限流算法流程图

最终，经过限流算法的处理后，在“尖叫效应”爆发时期，信息阅读流量爆发增长能够在极短的时间内得到优化调整.

（2）基于控制的“定向免疫”策略

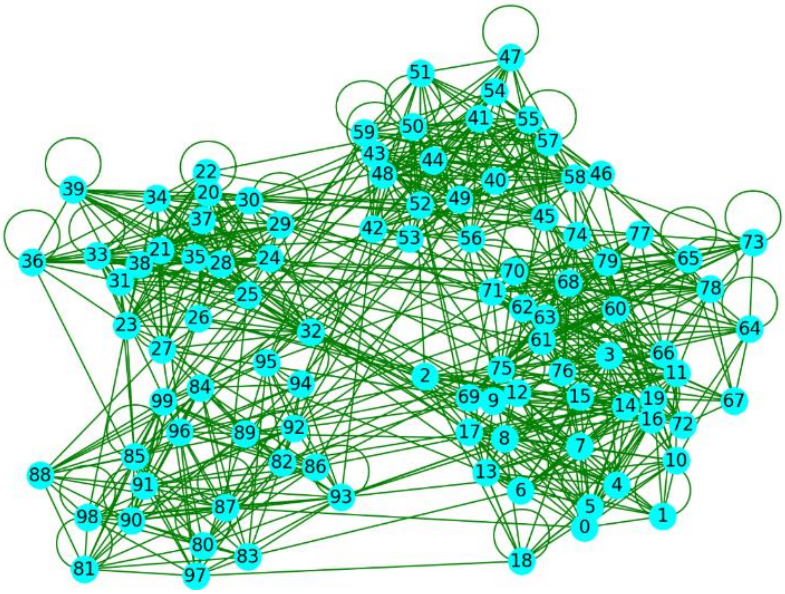
在信息传播模型的局部特征中，节点的属性是一个影响消息传播的重要因素（在 5.2.1 中已分析）。“定向免疫”方法就是针对于某些具有特殊性质的节点而出现的控制策略。与限流算法的宏观控制不同，“定向免疫”是从微观角度发挥作用，针对某些特定人群，进行精确地传播控制（程序见附录 11）.

引用图模型研究信息传播模型的局部性质，生成一个网络模型G_1。该模型初始状态由众多节点$n_1, n_2, \ldots n_r$，以及节点之间的无向边$(n_1, n_2), \ldots, (n_{r-1}, n_r)$组成，我们记为.

$$V_1 = \{n_1, n_2, \ldots, n_r\}$$
$$E_1 = \{(n_1, n_2), (n_1, n_3), \ldots, (n_{r-1}, n_r)\}$$
$$G_1 = (V_1, E_1)$$

初始状态下，V_1其下所有节点都是S节点，如图 23 所示.

图 23：初始状态网络图



其中，青色表示S节点，所有的边用绿色表示.

当信息出现时，随机选择某个节点变为I节点，随后记录每个节点的属性变化情况，并计量每种属性的节点随时间变化情况。

本策略的核心在于，一般情况下网络都具有节点度不均匀的特性，存在少部分点的节点度非常大,与众多其他节点存在连接关系，这部分节点的在网络中传播能力极强。因此，选择性地免疫这些节点，可以大大减少可能的传播途径。当信息即将开始传播时，立刻选择网络模型中的高节点度节点，将其定向免疫，使其从S节点R节点。以此达到压制信息蔓延的目的。

最终，整个图模型完成传播后，绘制网络图（程序见附录 11）如图 24 所示.

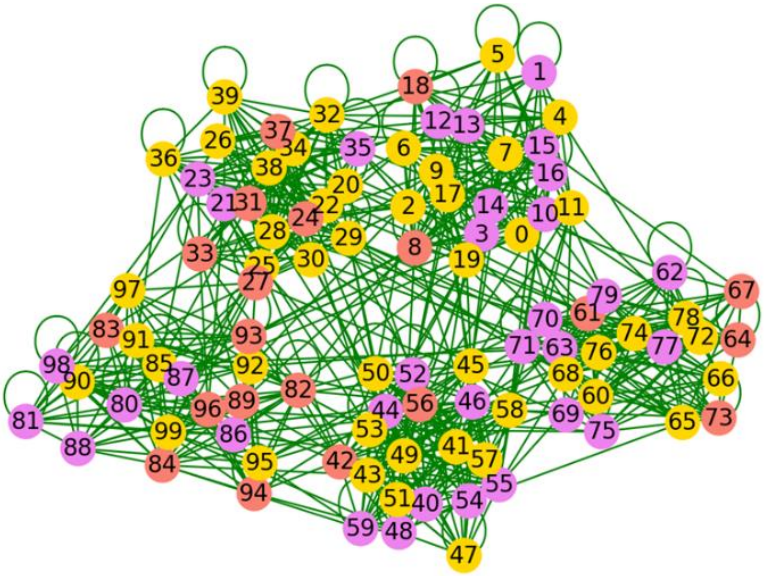


图 24：传播完成网络图

其中，紫罗兰色代表I节点，橙红色代表R节点，金色代表E节点.可以从图中看出，相当数量的免疫节点存在，由于这些免疫节点的节点度普遍较大，因此大量边因为免疫节点的设置而在图模型中丧失传播能力,因此可以有效遏制信息通过不同边进行的传播.

5.4.2“回声室效应“的破除

“回声室效应”的建立主要依托于系统建立的用户打分矩阵与用户行为数据之上.一个多元化的、非单调的用户矩阵是不利于系统偏好算法的.可以从用户行为与算法顶层设计两个层次来进行策略指定.

(1) 用户自身行为

用户可以避免长期浏览一些类别单调、情景单一的信息；面对网络上一些炒作的“标题党”不予浏览；面对他人提出的一些与自己观点相仿的激进的评论，要加以判断与甄别.从 app 操作上来看，可以定期清除自己的一些浏览记录 cookie 等.

（2）算法顶层设计

在 ALS 迭代计算 R 矩阵过程中，通常是直到矩阵中的元素不再发生变化时，认为矩阵近似收敛，停止迭代，根据第二问中的计算，一般情况下，用户矩阵在经过 9-10 次迭代之后，就会近似收敛。完成收敛后的矩阵，极度贴合用户的喜好。

为了避免用户只接受到自己喜好的信息，我们对矩阵的收敛进行限制，使得矩阵不能精确学习用户的爱好,从而会有一定的概率向用户推送偏离其喜好的内容。具体在 ALS 中的实现方法如下.

ALS 算法使用$RMSE$(root-mean-square-error)来评估误差是否收敛.

$$RMSE = \sqrt{\frac{\sum_{r_{xy}\neq 0}\left(r_{xy} - u_x^T m_y\right)^2}{N}}$$

其中，N为原始打分矩阵中不等于 0（即用户已对物品打分）的元素个数.

通过计算每一次 ALS 迭代后的$RMSE$，来判断当前状态下矩阵的收敛情况.设定一个特定的$RMSE$阈值R_0，当误差值满足$RMSE < R_0$时，即停止 ALS 迭代得到最后的打分矩阵.从而防止矩阵过度收敛,推荐物品过度贴近用户喜好使得推

荐系统出现回声室效应. 如图 25 所示.

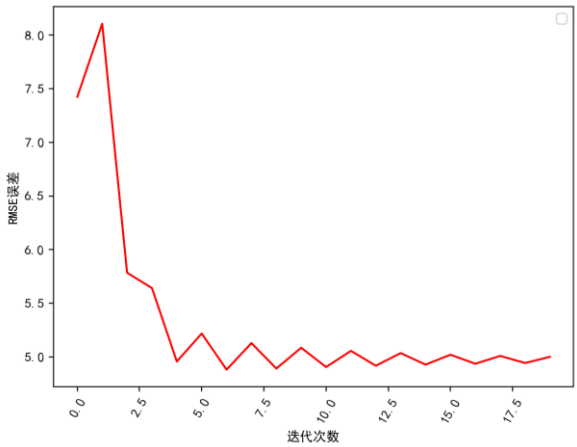


图 25 $RMSE$随迭代次数变化图

如图所示，在迭代多次后，$RMSE$值收敛到5.0左右. 为了防止过度收敛，可设定$R_0 = 6.0$. 当$RMSE$小于6.0时，即迭代 2 次后即可停止迭代，认定此时的打分矩阵已经可以适度满足用户的喜好. 此时的打分预测结果如表

表 5 改进后 ALS 算法给出的打分表

	爱豆1	爱豆2	爱豆3	军事1	军事2	军事3	政治1	政治2	政治3	电视剧1	电视剧2	电视剧3
用户1	0.850339	0.874432	3.147069	-1.40089	0.366461	0.275577	-1.48113	-1.57201	0.013448	1.923847	-0.09743	0.898403
用户2	0.856357	1.220951	3.165144	-1.66081	-0.32092	-0.10843	-1.63267	-1.42018	0.548949	1.844578	-0.36838	1.789806
用户3	0.435504	0.78804	1.289267	-0.56947	-0.28601	-0.01729	-0.53578	-0.26706	0.587557	0.761719	-0.07563	1.174288
用户4	0.028578	-0.61534	-1.15389	2.011037	2.179572	1.645851	1.666142	1.132421	-0.83275	-0.28001	1.482049	-2.32553
用户5	0.384039	-0.68734	0.257999	1.633155	3.158074	2.192269	1.124405	0.1586	-1.52844	0.680971	1.718933	-3.03907
用户6	0.045905	-0.90139	-1.08033	2.198603	2.799223	1.992006	1.755412	0.948195	-1.31132	-0.15554	1.717	-3.09439
用户7	-0.16609	0.217869	-1.30159	0.840758	-0.39667	-0.00736	0.90776	1.297065	0.706259	-0.76199	0.234157	0.469838
用户8	-0.40156	-0.40127	-2.65158	2.109583	0.594025	0.68898	2.026356	2.121312	0.178469	-1.39029	0.931516	-1.01818
用户9	-0.18342	0.503916	-1.37515	0.653192	-1.01632	-0.35352	0.81849	1.481291	1.184834	-0.88646	-0.00079	1.238699
用户10	0.561936	1.37407	2.260314	-1.74868	-1.50125	-0.85685	-1.52643	-0.88204	1.234276	1.13226	-0.84288	2.766294
用户11	0.518156	0.167938	1.308265	0.1818	1.379661	1.020242	-0.04968	-0.4091	-0.47816	1.019749	0.701794	-0.73037
用户12	0.239104	0.859558	0.561872	-0.45037	-0.92702	-0.39311	-0.31029	0.223622	1.014283	0.243904	-0.27248	1.67951

根据该打分表进行物品推荐，系统将会给原本痴迷于追星话题的用户 1 推荐和电视剧有关的话题；给原本痴迷于电视剧话题的用户 11 推荐和政治有关的话题. 可见这样的 ALS 算法在破除“回声室效应”上发挥作用.

5.5 第四问：解决方案与建议

关于破除“信息茧房”的建议报告

在全新的信息传播格局下，“尖叫效应”与“回声室效应”这类现象愈演愈烈，这两种效应的叠加，使得广大网络用户很容易被最终困于一个单一化的信息囚笼中，称为“信息茧房”. 这种囚笼的存在使得网络用户对于信息的接触愈发单一，久而久之失去了接触“不同的声音”的机会

每当有重大的社会事件发生时，我们能注意到：总有一些“标题党”写下片面而又激进的文章，阅读与转发者不计其数，从而使不客观、不准确的信息在网

络上“泛滥”.机械的解读、情绪化的理解，使得无数围观网民也纷纷从众，继而丧失了自己的观点，最终只能在一片相似的观点中互相认同，丧失了与其他观点交流与发现的机会.类似的现象屡见不鲜，这就是无处不在的“信息茧房”. 走出“信息茧房”，就是当前迫切需要解决的现实问题.

从广大网络用户的个人担当的角度来说，应当加强个人网络素养的养成，术产生的后果最终还是看运用技术的人类本身，在茧房效应的影响下用户个人需要主动拓宽自我的信息接收获取渠道，试着从多方面接收来自不同媒介的信息.算法推送下，使接收信息的体验感更加丰富，而在这一过程中难免会出现与自我价值观相冲突的部分.用户个人应该构建起理性思维，积极培养自身的媒介素养[11].例如：用户在面对一些有意蹭热度、抹黑甚至引战的内容时、不应当去转发甚至评论、应该以理性克制的态度为恶意信息“限流”，否则只会助长该信息的热度，从而使其被更多人看到.

以一个一千人的网络群体模拟全体网络用户，当一个“片面而激进”的消息开始在网络中传播时，在用户素质整体优良同用户素质整体欠佳的两种不同情况下，该网络的信息传播趋势对比如图 26.

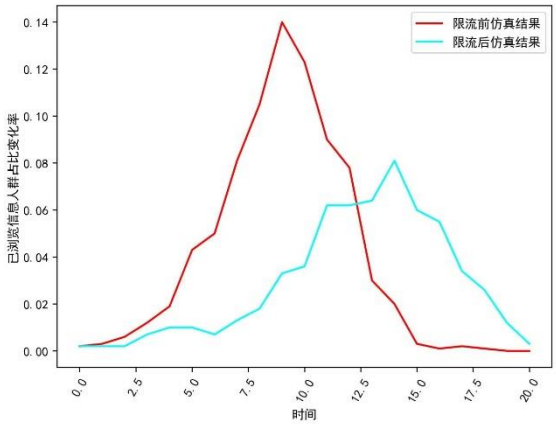


图 26：限流前后（用户素质低、高）已浏览信息人群占比变化率对比

由图 26 可知，在用户素质整体优良的情况下，网络中受到恶意信息影响的用户比例下降的用户占比变化率最大到了总用户数的 6%，有效降低了恶意消息对于网络媒体和用户民众的冲击，这种从用户本身出发的破解“尖叫效应”，从而对抗“信息茧房”的方式，是最本质且最长远的方案.

从政府的顶层设计角度上来讲，政府可以有意出台一些管理网络平台的法律法规，打击一些恶意传播消息、哗众取宠的一系列行为，限制诸如此类的“尖叫效应”的产生，同时，可以对网络媒体的推荐系统作出限制，禁止使用一些学习性过强的偏好算法，不允许过度采集用户数据、拟合用户特征，给用户的网络画像保留一层“面纱”，从而阻止“回声室效应”的作用.

政府出台法规要求网络平台中的一些信息进行防范性控制时，可以有效降低

信息传播的规模，以防范可能出现的不良信息对于舆论导向的控制。以 2021 年 8 月份微博话题#上海全面推行轻微违法行为不予行政处罚#一话题为例，推行限流效果前后对比如图 27 所示.

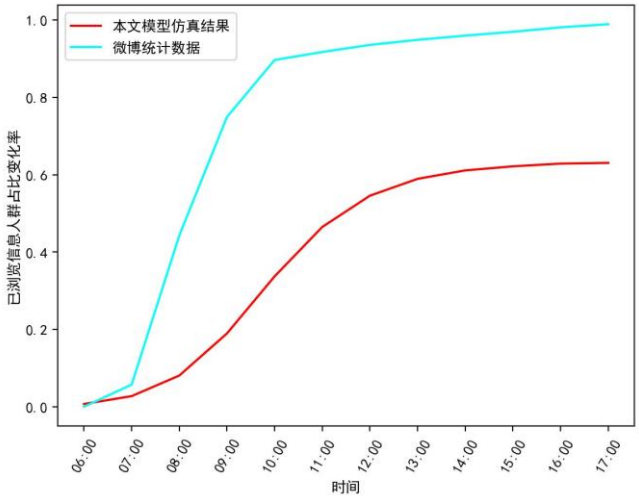


图 27：限流前后已浏览信息人群占比与时间关系对比

由图 27 可知，开启防范性控制之后，阅读量较之前减少了 40%，有效控制了舆论的走向，这种基于政府意志层面的调控，能够从宏观调控有效破除“信息茧房”现象.

从主流媒体的引领角度来看，主流媒体应当注重信息推送的多元化，同时加强平台的规范化管理，对于某些故意传播虚假或者低俗内容的大 V 博主予以封号处罚等. 同时借助自身媒体优势，多加宣传多样化的正能量信息，对一些即将开始传播的恶意信息予以制止. 以此能够优化信息传输的顶层路径.

以上一系列不同层面的措施，是破除“信息茧房”有力措施，同时基于数据分析和实际情况综合考量了应用效果，希望政府部门能够考虑或采纳.

六．模型的检验与分析

6.1 灵敏度分析

在前面的仿真实验中，对于一个过程的刻画需要用到一些参数来调整结果来达到一个较好的效果.但是，如果现在对这个参数进行一定的变化，其结果是否还能保持一个理想的状态？这就需要对其去做灵敏度分析.在前面的实验中，假设$\lambda = 0.8$，$\rho = 0.2$.现在对这两个参数取不同的数值，第二次取$\lambda = 0.3$，$\rho = 0.5$，第三次取$\lambda = 0.7$，$\rho = 0.5$.其效果如图 28 所示（程序见附录 15）.

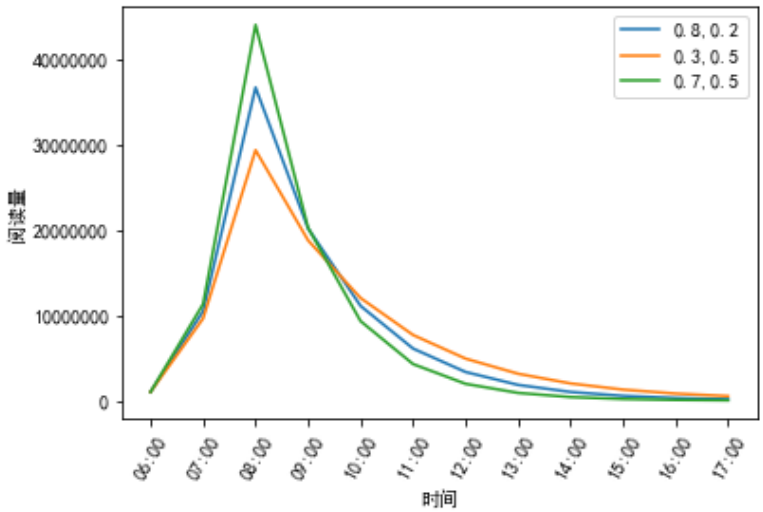


图 28：灵敏度分析实验图

对模型做灵敏度分析是测试模型稳定性的一个重要方式.可以看到，阅读量随时间变化的序列在不同参数下的结果呈相似的态势.尽管在不同位置的阅读量发生了变化，但是曲线的走势是不变的.这就证明了模型的灵敏度是良好的，即模型可以较好适应各种各样的情况.

再以 5.4.1.2 中的图网络模型为例，不断调整图网络模型在开始传播时被“定向免疫”的节点个数，采用话题#上海全面推行轻微违法行为不予行政处罚#的数据进行仿真，绘制不同免疫节点数时阅读量随时间变化曲线图（程序见附录 16），如图 29.

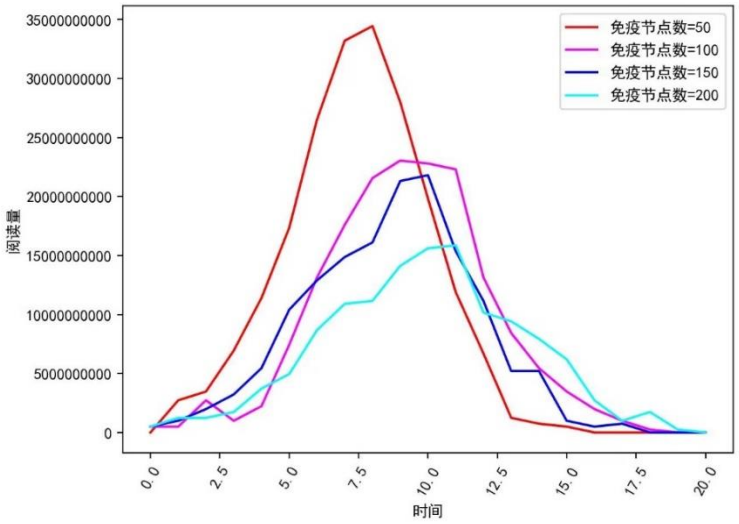


图 29. 不同免疫节点数时阅读量随时间变化曲线图

同样可以发现，在不断增加免疫节点数量时，限流效果逐渐增强，且这种增强的趋势是稳定变化而非骤增的。最终各条曲线都收敛到同一纵坐标值，证明了图模型在不同条件的参数下是稳定的.

6.2 误差分析

（1）打分的原始数据是离散化的，存在精度缺失，造成误差；

（2）微分方程模型的参数是通过拟合而得到的，最终的解为数值解而非解析解.

（3）语料分析部分。选定训练集时所需要的语料库对于最终文本分类的精确度存在极大的影响，若语料库中语料丰富度较低，或与测试集相关度较小，则极有可能出现分类召回率低等精度问题；由于实际网络情境中，用户的汉语表达含义与情景存在相当大的相关性，如“反语”现象，导致分类器分类效果差。在识别此类信息时，还要依赖语义扩展、联系上下文等其他方式提高识别精度.

（4）在问题三的图模型中，仅选用 1000 个节点进行传播模拟，同时节点之间的有向边是基于随机生成的建立方式，导致节点度分布可能与实际常见的网络形态存在差异，造成仿真误差.

七. 模型的评价

（1）优点

①借鉴信息传播与病毒传播过程中的相似性，并综合考虑现实生活信息传播过程中推荐算法，网络用户心理等方面的影响，将模型进行复杂化，使模型更贴

合实际.并引入基于语料特征分析的子群体划分，图网络模型，综合刻画了信息在群体传播中的宏观行为和微观特征.

②使用 ALS 矩阵分解模型，避免了主观打分，所得到的结果较为准确客观.同时引用了$RMSE$方法控制矩阵的迭代次数，矩阵学习可控性强.

2）缺点

本文基于历史数据展开分析与预测，模型预测情况与未来真实情况可能有一定出入.

八．参考文献

[1]杜鸿毅,杨华,刘艳红,杨鸿鹏.基于网络媒体的非线性动力学信息传播模型[J].计算机科学,2022,49(S1):280-284.

[2]曹玉林. 移动机会网络中的不良信息传播动力学模型与控制策略研究[D].陕西师范大学,2019.DOI:10.27292/d.cnki.gsxfu.2019.001178.

[3]阿里云.手把手教你在 Python 中实现文本分类（附代码、数据集）
https://developer.aliyun.com/article/593627

[4]信息检索研究室.
https://ir.dlut.edu.cn / EmotionOntology Download.aspx

[5]韩中庚.数学建模方法及其应用（第二版）.北京：高等教育出版社.2009.6

[6]简书.基于模型的协同过滤算法 ALS 简单实现.
https://www.jianshu.com/p/7f71a6b8a98a . 2019.11.12

[7]知乎.数据挖掘-协同过滤（ALS）.
https://zhuanlan.zhihu.com/p/231841722 .2020.09.10

[8]CSDN. EdgeRank 杂谈
https://blog.csdn.net/fangaoxin/article/details/7283168

[9] 宋会敏. 社区网络中信息传播控制技术研究[D].沈阳航空航天大学,2017.

[10]CSDN. 主流的几种限流策略，我都可以通过 python+redis 实现
https://blog.csdn.net/chenchen5152/article/details/121705061

[11]白云祥,汪航.针对算法推送强化茧房效应的分析与对策研究[J].视听,2020(02):204-205.DOI:10.19395/j.cnki.1674-246x.2020.02.106.

附录

本文使用 python 3.9.7，spyder 开发环境进行编写.编写程序过程中调用了 numpy、matplotlib、pandas、pyscipopt 第三方库.

附录 1：绘制测试数据图像

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import gensim
data=pd.read_excel("上海数据.xlsx")

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

x=np.array(['                ']*len(data))
for i in range(len(data)):
    x[i]=str(data.iloc[i,0])[0:5]
y=np.array(data.iloc[:,1])

fig, ax=plt.subplots(figsize=(5,3.5) )
ax.plot(x,y,'r')
ax.set(xlabel='时间',ylabel='阅读量')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
plt.show()
```

附录 2：SEIR 仿真模型

```
from scipy.integrate import odeint
from sympy.abc import t
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#载入现有数据
real=pd.read_excel("上海数据.xlsx")
x=np.array(['                ']*len(real))
for i in range(len(real)):
    x[i]=str(real.iloc[i,0])[0:5]
y=np.array(real.iloc[:,1])
```

```
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

#常量赋值
lambda_=0.8
rou=0.2
lambda_1=lambda_*0.5
lambda_2=lambda_*0.3
lambda_3=lambda_*0.2
rou_1=rou*0.5
rou_2=rou*0.3
rou_3=rou*0.2
N=82000000

#定义微分方程组为一类方程
class functions():
    def __init__(self,parameter1):
        self.parameter1=parameter1

    result=0

    def calculate(self,S,I):
        self.result=self.parameter1*S*I

f1=functions(-(rou+lambda_))
f2=functions(rou_1+lambda_1)
f3=functions(rou_2+lambda_2)
f4=functions(rou_3+lambda_3)
f1.calculate(1,5)
a=f1.result
print(a)
#定义微分方程
def Pfun(y,x):
    S, E, I ,R= y;
    fun1=-(lambda_+rou)*S*I
    fun2=(lambda_1+rou_1)*S*I
    fun3=(lambda_2+rou_2)*S*I
    fun4=(lambda_3+rou_3)*S*I
    return np.array([fun1, fun2, fun3, fun4])
```

```
#定义时间区间
endtime=90
interval_length=endtime/len(real)
t=np.arange(0, endtime, interval_length)  #创建时间点

#寻求微分方程数值解
sol1=odeint(Pfun, [0.99, 0.005, 0.005, 0], t)

#得到每段时间间隔内的阅读量
cum_result=(sol1[:,1]+sol1[:,2]+sol1[:,3])*N
result=np.array([0]*len(real))
for i in range(len(real)):
    if(i==0):
        result[i]=cum_result[i]
    else:
        result[i]=cum_result[i]-cum_result[i-1]

#数据可视化
fig, ax = plt.subplots(figsize=(7,5.25))
ax.plot(x, result, label='本文模型仿真结果',color='r')
ax.scatter(x, y, label='微博统计数据',color='teal')
ax.set(ylabel='阅读量', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 3：微博评论爬虫

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import os

def fetchUrl(pid, uid, max_id):

    url = "https://weibo.com/ajax/statuses/buildComments"

    headers = {
        "user-
agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36
",
```

```
    }

    params = {
        "flow" : 0,
        "is_reload" : 1,
        "id" : pid,
        "is_show_bulletin" : 2,
        "is_mix" : 0,
        "max_id" : max_id,
        "count" : 20,
        "uid" : uid,
    }

    r = requests.get(url, headers = headers, params = params)
    return r.json()

def parseJson(jsonObj):

    data = jsonObj["data"]
    max_id = jsonObj["max_id"]

    commentData = []
    for item in data:
        # 评论 id
        comment_Id = item["id"]
        # 评论内容
        content = BeautifulSoup(item["text"], "html.parser").t
ext
        # 评论时间
        created_at = item["created_at"]
        # 点赞数
        like_counts = item["like_counts"]
        # 评论数
        total_number = item["total_number"]


        dataItem = [comment_Id, created_at,  like_counts, tota
l_number, content]
        print(dataItem)
        commentData.append(dataItem)

    return commentData, max_id
```

```
def save_data(data, path, filename):

    if not os.path.exists(path):
        os.makedirs(path)

    dataframe = pd.DataFrame(data)
    dataframe.to_csv(path + filename, encoding='utf_8_sig', mo
de='a', index=False, sep=',', header=False )

if __name__ == "__main__":

    pid = 4838907944109106  # 微博id，固定
    uid = 1259110474   # 用户id，固定
    max_id = 0
    path = "D:/Data/"            # 保存的路径
    filename = "comments.csv"    # 保存的文件名

    csvHeader = [["评论id", "发布时间", "点赞数", "回复数", "评论
内容"]]
    save_data(csvHeader, path, filename)

    while(True):
        html = fetchUrl(pid, uid, max_id)
        comments, max_id = parseJson(html)
        save_data(comments, path, filename)
        # max_id 为 0 时，表示爬取结束
        if max_id == 0:
            break;
```

附录 4：语料分析

```
import numpy as np
import pandas as pd
import jieba
import math
#载入语料库
df = pd.read_excel("F:\建模\深圳杯\情感词汇本体\情感词汇本
体.xlsx")
df = df[['词语', '词性种类', '词义数', '词义序号', '情感分类', '强
度', '极性']]

#载入微博数据
```

```
weibo_df=pd.read_csv("F:\建模\深圳杯\comments.csv")

#情绪分类
PC = []
PE = []
PD = []
PH = []
PG = []
PB = []
PK = []
PA = []
NA = []
NB = []
NJ = []
NH = []
PF = []
NI = []
NC = []
NG = []
NE = []
ND = []
NN = []
NK = []
NL = []

PC2 = []
PE2 = []
PD2 = []
PH2 = []
PG2 = []
PB2 = []
PK2 = []
PA2 = []
NA2 = []
NB2 = []
NJ2 = []
NH2 = []
PF2 = []
NI2 = []
NC2 = []
NG2 = []
NE2 = []
ND2 = []
```

```
NN2 = []
NK2 = []
NL2 = []

for idx, row in df.iterrows():
    if row['情感分类'] in ['PA']:
        PA.append(row['词语'])
        PA2.append(row['强度'])
    if row['情感分类'] in ['PE']:
        PE.append(row['词语'])
        PE2.append(row['强度'])
    if row['情感分类'] in ['PD']:
        PD.append(row['词语'])
        PD2.append(row['强度'])
    if row['情感分类'] in ['PH']:
        PH.append(row['词语'])
        PH2.append(row['强度'])
    if row['情感分类'] in ['PG']:
        PG.append(row['词语'])
        PG2.append(row['强度'])
    if row['情感分类'] in ['PB']:
        PB.append(row['词语'])
        PB2.append(row['强度'])
    if row['情感分类'] in ['PK']:
        PK.append(row['词语'])
        PK2.append(row['强度'])
    if row['情感分类'] in ['PC']:
        PC.append(row['词语'])
        PC2.append(row['强度'])
    if row['情感分类'] in ['NA']:
        NA.append(row['词语'])
        NA2.append(row['强度'])
    if row['情感分类'] in ['NB']:
        NB.append(row['词语'])
        NB2.append(row['强度'])
    if row['情感分类'] in ['NJ']:
        NJ.append(row['词语'])
        NJ2.append(row['强度'])
    if row['情感分类'] in ['NH']:
        NH.append(row['词语'])
        NH2.append(row['强度'])
    if row['情感分类'] in ['PF']:
        PF.append(row['词语'])
```

```
            PF2.append(row['强度'])
        if row['情感分类'] in ['NI']:
            NI.append(row['词语'])
            NI2.append(row['强度'])
        if row['情感分类'] in ['NC']:
            NC.append(row['词语'])
            NC2.append(row['强度'])
        if row['情感分类'] in ['NG']:
            NG.append(row['词语'])
            NG2.append(row['强度'])
        if row['情感分类'] in ['NE']:
            NE.append(row['词语'])
            NE2.append(row['强度'])
        if row['情感分类'] in ['ND']:
            ND.append(row['词语'])
            ND2.append(row['强度'])
        if row['情感分类'] in ['NN']:
            NN.append(row['词语'])
            NN2.append(row['强度'])
        if row['情感分类'] in ['NK']:
            NK.append(row['词语'])
            NK2.append(row['强度'])
        if row['情感分类'] in ['NL']:
            NL.append(row['词语'])
            NL2.append(row['强度'])

dictionary =np.array(PC+PE+PD+PH+PG+PB+PK+PA+NA+NB+NJ+NH+PF+NI
+NC+NG+NE+ND+NN+NK+NL)
dictionary_density =np.array(PC2+PE2+PD2+PH2+PG2+PB2+PK2+PA2+N
A2+NB2+NJ2+NH2+PF2+NI2+NC2+NG2+NE2+ND2+NN2+NK2+NL2)
print('情绪词语列表整理完成')

#生成向量
def vectorbuilder(vector_text):
    vector1=np.array([0]*len(dictionary))
    vector2=np.array([0]*len(dictionary))
    for i in range(len(vector_text)):
        for j in range(len(dictionary)):
            if (vector_text[i]==dictionary[j]):
                vector1[j]=1
                vector2[j]=dictionary_density[j]
    return vector1,vector2
```

```
#进行分词
def spilt(word):
    if(type(word)==float):
        word=''
        return word
    wordset=jieba.lcut(word)
    return wordset


#概率计算
probin_PC=len(PC)/len(dictionary)
probin_PE=len(PE)/len(dictionary)
probin_PD=len(PD)/len(dictionary)
probin_PH=len(PH)/len(dictionary)
probin_PG=len(PG)/len(dictionary)
probin_PB=len(PB)/len(dictionary)
probin_PK=len(PK)/len(dictionary)
probin_PA=len(PA)/len(dictionary)
probin_PA=len(NA)/len(dictionary)
probin_NB=len(NB)/len(dictionary)
probin_NJ=len(NJ)/len(dictionary)
probin_NH=len(NH)/len(dictionary)
probin_PF=len(PF)/len(dictionary)
probin_NI=len(NI)/len(dictionary)
probin_NC=len(NC)/len(dictionary)
probin_NG=len(NG)/len(dictionary)
probin_NE=len(NE)/len(dictionary)
probin_ND=len(ND)/len(dictionary)
probin_NK=len(NK)/len(dictionary)
probin_NL=len(NL)/len(dictionary)
probin_NN=len(NN)/len(dictionary)
print('概率计算完成')



#生成标准向量
standardvector1=np.empty([21, len(dictionary)], dtype=int)
standardvector2=np.empty([21, len(dictionary)], dtype=int)
vector1,vector2=vectorbuilder(PC)
standardvector1[0,:]=vector1
standardvector2[0,:]=vector2
vector1,vector2=vectorbuilder(PE)
```

```
standardvector1[1,:]=vector1
standardvector2[1,:]=vector2
vector1,vector2=vectorbuilder(PD)
standardvector1[2,:]=vector1
standardvector2[2,:]=vector2
vector1,vector2=vectorbuilder(PH)
standardvector1[3,:]=vector1
standardvector2[3,:]=vector2
vector1,vector2=vectorbuilder(PG)
standardvector1[4,:]=vector1
standardvector2[4,:]=vector2
vector1,vector2=vectorbuilder(PB)
standardvector1[5,:]=vector1
standardvector2[5,:]=vector2
vector1,vector2=vectorbuilder(PK)
standardvector1[6,:]=vector1
standardvector2[6,:]=vector2
vector1,vector2=vectorbuilder(PA)
standardvector1[7,:]=vector1
standardvector2[7,:]=vector2
vector1,vector2=vectorbuilder(NA)
standardvector1[8,:]=vector1
standardvector2[8,:]=vector2
vector1,vector2=vectorbuilder(NB)
standardvector1[9,:]=vector1
standardvector2[9,:]=vector2
vector1,vector2=vectorbuilder(NJ)
standardvector1[10,:]=vector1
standardvector2[10,:]=vector2
vector1,vector2=vectorbuilder(NH)
standardvector1[11,:]=vector1
standardvector2[11,:]=vector2
vector1,vector2=vectorbuilder(PF)
standardvector1[12,:]=vector1
standardvector2[12,:]=vector2
vector1,vector2=vectorbuilder(NI)
standardvector1[13,:]=vector1
standardvector2[13,:]=vector2
vector1,vector2=vectorbuilder(NC)
standardvector1[14,:]=vector1
standardvector2[14,:]=vector2
vector1,vector2=vectorbuilder(NG)
standardvector1[15,:]=vector1
```

```
standardvector2[15,:]=vector2
vector1,vector2=vectorbuilder(NE)
standardvector1[16,:]=vector1
standardvector2[16,:]=vector2
vector1,vector2=vectorbuilder(ND)
standardvector1[17,:]=vector1
standardvector2[17,:]=vector2
vector1,vector2=vectorbuilder(NN)
standardvector1[18,:]=vector1
standardvector2[18,:]=vector2
vector1,vector2=vectorbuilder(NK)
standardvector1[19,:]=vector1
standardvector2[19,:]=vector2
vector1,vector2=vectorbuilder(NL)
standardvector1[20,:]=vector1
standardvector2[20,:]=vector2

#生成句向量
vectorset1=np.empty([len(weibo_df), len(dictionary)], dtype=in
t)
vectorset2=np.empty([len(weibo_df), len(dictionary)], dtype=in
t)
for k in range(len(weibo_df)):
    wordset=spilt(weibo_df.iloc[k,4])
    vector1,vector2=vectorbuilder(wordset)
    vectorset1[k,:]=vector1
    vectorset2[k,:]=vector2
print('特征向量生成完成')

class likehood():
    def __init__(self,vector):
        self.prob_array1=vector*0.1
        self.prob_array2=vector
        for i in range(len(vector)):
            if(self.prob_array1[i]==0):
                self.prob_array1[i]=0.5
    def multiplication(self,sample):
        result1=1
        result=0
        for i in range(len(sample)):
            result1=result1*(2*self.prob_array1[i]*sample[i]+1
-self.prob_array1[i]-sample[i])*2
            result=result+self.prob_array2[i]*sample[i]
```

```
        return result


PCl = likehood(standardvector2[0,:])
PEl = likehood(standardvector2[1,:])
PDl = likehood(standardvector2[2,:])
PHl = likehood(standardvector2[3,:])
PGl = likehood(standardvector2[4,:])
PBl = likehood(standardvector2[5,:])
PKl = likehood(standardvector2[6,:])
PAl = likehood(standardvector2[7,:])
NAl = likehood(standardvector2[8,:])
NBl = likehood(standardvector2[9,:])
NJl = likehood(standardvector2[10,:])
NHl = likehood(standardvector2[11,:])
PFl = likehood(standardvector2[12,:])
NIl = likehood(standardvector2[13,:])
NCl = likehood(standardvector2[14,:])
NGl = likehood(standardvector2[15,:])
NEl = likehood(standardvector2[16,:])
NDl = likehood(standardvector2[17,:])
NNl = likehood(standardvector2[18,:])
NKl = likehood(standardvector2[19,:])
NLl = likehood(standardvector2[20,:])
def Bayesian(vector):
    rating=np.array([0]*21)
    rating[0]=PCl.multiplication(vector)
    rating[1]=PEl.multiplication(vector)
    rating[2]=PDl.multiplication(vector)
    rating[3]=PHl.multiplication(vector)
    rating[4]=PGl.multiplication(vector)
    rating[5]=PBl.multiplication(vector)
    rating[6]=PKl.multiplication(vector)
    rating[7]=PAl.multiplication(vector)
    rating[8]=NAl.multiplication(vector)
    rating[9]=NBl.multiplication(vector)
    rating[10]=NJl.multiplication(vector)
    rating[11]=NHl.multiplication(vector)
    rating[12]=PFl.multiplication(vector)
    rating[13]=NIl.multiplication(vector)
    rating[14]=NCl.multiplication(vector)
    rating[15]=NGl.multiplication(vector)
```

```
    rating[16]=NEl.multiplication(vector)
    rating[17]=NDl.multiplication(vector)
    rating[18]=NNl.multiplication(vector)
    rating[19]=PKl.multiplication(vector)
    rating[20]=NLl.multiplication(vector)
    type_=np.argmax(rating)
    return type_

typevector=np.array([0]*len(weibo_df))
for i in range(len(weibo_df)):
    typevector[i]=Bayesian(vectorset1[i,:])

proportion=np.array([0]*21)
for i in range(len(typevector)):
    proportion[typevector[i]]+=1
proportion=proportion/len(typevector)

dic = {"情感类别": ['惊起','安心','尊敬','赞扬','相信','喜爱','祝愿','快乐','愤怒','悲伤','失望','疚','思','慌','恐惧','羞','烦闷','憎恶','贬责','嫉妒','怀疑'], #列表
       "比例": proportion} #元组
data = pd.DataFrame(dic) # 创建Dataframe
data.to_csv("D:/Data/result.csv",encoding='utf_8_sig', mode='a', index=False, sep=',', header=False)
```

附录 5：饼状图绘制

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("F://建模//深圳杯//result.csv",header=None)
a=data.iloc[:,1]
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False


rest=0
for i in range(len(data)):
    if(data.iloc[i,1]<=0.04):
        rest=rest+data.iloc[i,1]

y = np.array([0.445127305, 0.336259877, 0.063213345, 0.044776119,rest])
```

```
plt.pie(y,
        labels=['惊奇','悲伤','快乐','喜爱','其他'], # 设置饼图标签
        colors=["#d5695d", "#5d8ca8", "#65a479", "#a564c9","gold"], # 设置饼图颜色
       )
# 设置标题
plt.show()
1.
```

附录 6：预料分析 SEIR 模型

```
1. from scipy.integrate import odeint
2. from sympy.abc import t
3. import numpy as np
4. import matplotlib.pyplot as plt
5. import pandas as pd
6.
7. #载入现有数据
8. real=pd.read_excel("F://建模//深圳杯//大熊猫数据.xlsx")
9. x=np.array([']*len(real))
10.for i in range(len(real)):
11.    x[i]=str(real.iloc[i,0])[0:5]
12.y=np.array(real.iloc[:,1])
13.
14.plt.rcParams['font.sans-serif']=['SimHei']
15.plt.rcParams['axes.unicode_minus'] = False
16.
17.
18.def SEIR(lambda_c,rou_c,N_p,t):
19.    #常量赋值
20.    lambda_=lambda_c
21.    rou=rou_c
22.    lambda_1=lambda_*0.5
23.    lambda_2=lambda_*0.3
24.    lambda_3=lambda_*0.2
25.    rou_1=rou*0.5
26.    rou_2=rou*0.3
27.    rou_3=rou*0.2
28.    N=177000000*2.05*N_p
29.
30.    #定义微分方程组为一类方程
```

```
    class functions():
        def __init__(self,parameter1):
            self.parameter1=parameter1

        result=0

        def calculate(self,S,I):
            self.result=self.parameter1*S*I

    f1=functions(-(rou+lambda_))
    f2=functions(rou_1+lambda_1)
    f3=functions(rou_2+lambda_2)
    f4=functions(rou_3+lambda_3)
    f1.calculate(1,5)
    a=f1.result
    print(a)
    #定义微分方程
    def Pfun(y,x):
        S, E, I ,R= y;
        fun1=-(lambda_+rou)*S*I
        fun2=(lambda_1+rou_1)*S*I
        fun3=(lambda_2+rou_2)*S*I
        fun4=(lambda_3+rou_3)*S*I
        return np.array([fun1, fun2, fun3, fun4])


    #定义时间区间
    endtime=t
    interval_length=endtime/len(real)
    t=np.arange(0, endtime, interval_length)  #创建时间点

    #寻求微分方程数值解
    sol1=odeint(Pfun, [0.99, 0.005, 0.005, 0], t)

    #得到每段时间间隔内的阅读量
    cum_result=(sol1[:,1]+sol1[:,2]+sol1[:,3])*N
    result=np.array([0]*len(real))
    for i in range(len(real)):
        if(i==0):
            result[i]=cum_result[i]
        else:
            result[i]=cum_result[i]-cum_result[i-1]
```

```
    return result

jing_result=SEIR(0.7,0.2,0.445127305,120)
bei_result=SEIR(0.55,0.2,0.336259877,75)
le_result=SEIR(0.25,0.3,0.063213345,80)
ai_result=SEIR(0.6,0.3,0.044776119,30)
else_result=SEIR(0.6,0.4,0.1106233538191391,40)
total_result=jing_result+bei_result+le_result+ai_result+else_result
#数据可视化
fig, ax = plt.subplots(figsize=(7,5.25))
ax.plot(x, jing_result, label='惊奇类人群仿真结果',color='yellow')
ax.plot(x, bei_result, label='悲伤类人群仿真结果',color='black')
ax.plot(x, le_result, label='快乐类人群仿真结果',color='y')
ax.plot(x, ai_result, label='喜爱类人群仿真结果',color='cyan')
ax.plot(x, else_result, label='其他类人群仿真结果',color='green')
ax.plot(x, total_result, label='总体人群仿真结果',color='r')
ax.scatter(x, y, label='微博统计数据',color='teal')
ax.set(ylabel='阅读量', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 7：协同过滤

```
import pandas as pd
import numpy as np
import math

#注意：改程序运行需要安装SCIP库，否则难以运行
#读入打分数据
R=np.array(pd.read_csv("F:\建模\深圳杯\打分表.csv"))[:,1:13]#建立打分矩阵
m=np.size(R,0)
n=np.size(R,1)
k=3
lambda_=1
```

```
#将nan 换成0
for i in range(m):
    for j in range(n):
        if (math.isnan(R[i,j])==True):
            R[i,j]=0
        else:
            R[i,j]=int(R[i,j])

#创建一个分解矩阵
X=np.random.randint(0,11,(k,m))

#对于Y 矩阵创建als 算法
def als1(X,k,n):
    X_transpose = X.transpose()
    for i in range(n):
        wawa=X@X_transpose+lambda_*np.eye(k)
        y=np.linalg.inv(wawa)@X@R[i,:]

        if (i==0):
            Y=y
        else:
            Y=np.vstack([Y, y])
    Y=Y.transpose()
    return Y

#对于X 矩阵创建ALS 算法
def als2(Y,k,m):
    Y_transpose = Y.transpose()
    for i in range(m):
        wawa=Y@Y_transpose+lambda_*np.eye(k)
        x=np.linalg.inv(wawa)@Y@(R[:,i].transpose())

        if (i==0):
            X=x
        else:
            X=np.vstack([X, x])
    X=X.transpose()
    return X

for i in range(400):
    Y=als1(X,k,n)
    Y=Y.astype(float)
```

```
    X=als2(Y,k,m)
    X=X.astype(float)

result=pd.DataFrame(X.transpose()@Y)

result.to_csv("F:\建模\深圳杯\预测打分表.csv")
```

附录 8：尖叫效应

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

real=pd.read_excel("F://建模//深圳杯//上海数据.xlsx")

x=np.array(['                ']*len(real))
for i in range(len(real)):
    x[i]=str(real.iloc[i,0])[0:5]
y=np.array(real.iloc[:,1])

#常量赋值
lambda_=0.8
rou=0.2
lambda_1=lambda_*0.3
lambda_2=lambda_*0.5
lambda_3=lambda_*0.2
rou_1=rou*0.3
rou_2=rou*0.5
rou_3=rou*0.2

#定义迭代函数
def f(S,E,I,R,t):
    S_N=S-t*(lambda_+rou)*S
    E_N=E+t*(lambda_1+rou_1)*S
    I_N=I+t*(lambda_2+rou_2)*S
    R_N=R+t*(lambda_3+rou_3)*S
    return S_N,E_N,I_N,R_N

#定义初始状态
S=0.99
```

```
E=0.005
I=0.005
R=0
N=103000000
#选择时间跨度
Wilcoxon=1.8
t=0.60

#迭代

a=np.array([0]*21)
a[0]=(E+I)*N
a[1]=6252762+N/Wilcoxon*0.46
for i in range(2,21):
    S_N,E_N,I_N,R_N=f(S,E,I,R,t)
    #print((I_N+E_N-I-E)*N)
    a[i]=(I_N+E_N-I-E)*N
    S,E,I,R=S_N,E_N,I_N,R_N
    #print(I_new)


#画图
fig, ax = plt.subplots(figsize=(6,4))
ax.plot(x, a[0:12], label='尖叫效应仿真结果',color='red')
ax.set(ylabel='阅读量', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 9：中立机制和极化机制刻画

```
1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4.
5. plt.rcParams['font.sans-serif']=['SimHei']
6. plt.rcParams['axes.unicode_minus'] = False
7.
8. real=pd.read_excel("F://建模//深圳杯//上海数据.xlsx")
9.
10.x=np.array['                ']*len(real))
11.for i in range(len(real)):
```

```
    x[i]=str(real.iloc[i,0])[0:5]
y=np.array(real.iloc[:,1])

#常量赋值
lambda_=0.8
rou=0.2
lambda_1=lambda_*0.3
lambda_2=lambda_*0.5
lambda_3=lambda_*0.2
rou_1=rou*0.3
rou_2=rou*0.5
rou_3=rou*0.2
miu_1=0.8
miu_2=0.2

#定义迭代函数
def f(S,E,I1,I2,R,t):
    S_N=S-t*(lambda_+rou)*S*(miu_1+miu_2)
    E_N=E+t*(lambda_1+rou_1)*S
    I1_N=I1+t*(lambda_2+rou_2)*S*miu_1
    I2_N=I2+t*(lambda_2+rou_2)*S*miu_2
    R_N=R+t*(lambda_3+rou_3)*S
    return S_N,E_N,I1_N,I2_N,R_N

#定义初始状态
S=0.985
E=0.005
I1=0.005
I2=0.005
R=0
N=103000000
#选择时间跨度
t=0.03

#迭代

a=np.array([0.00000000]*21)
a[0]=I1/(I2+I1)
for i in range(1,21):
    S_N,E_N,I1_N,I2_N,R_N=f(S,E,I1,I2,R,t)
    #print((I_N+E_N-I-E)*N)
    S,E,I1,I2,R=S_N,E_N,I1_N,I2_N,R_N
    a[i]=I1/(I2+I1)
```

```
    #print(I_new)


#画图
fig, ax = plt.subplots(figsize=(6,4))
ax.plot(x, a[0:12], label='极化机制')
ax.set(ylabel='I_2 类人群占 I 类比例', xlabel='时间')
plt.xticks(rotation=60)
ax.legend()#添加图例标注
plt.show()
```

附录 10：基于流量优化的限流算法

```
from scipy.integrate import odeint
from sympy.abc import t
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

#常量赋值
lambda_=0.8
rou=0.2
lambda_1=lambda_*0.5
lambda_2=lambda_*0.3
lambda_3=lambda_*0.2
rou_1=rou*0.5
rou_2=rou*0.3
rou_3=rou*0.2
N=177000000*1.4

#定义微分方程组为一类方程
class functions():
    def __init__(self,parameter1):
        self.parameter1=parameter1

    result=0

    def calculate(self,S,I):
        self.result=self.parameter1*S*I
```

```
f1=functions(-(rou+lambda_))
f2=functions(rou_1+lambda_1)
f3=functions(rou_2+lambda_2)
f4=functions(rou_3+lambda_3)
f1.calculate(1,5)
a=f1.result

#定义微分方程
def Pfun(y,x):
    S, E, I ,R= y;
    fun1=-(lambda_+rou)*S*I
    fun2=(lambda_1+rou_1)*S*I
    fun3=(lambda_2+rou_2)*S*I
    fun4=(lambda_3+rou_3)*S*I
    return np.array([fun1, fun2, fun3, fun4])



#定义时间区间
interval_length=3
time_length=20
#创建时间点

#寻求微分方程数值解
thorehold=0.04#设定阈值
cum_result=np.empty([time_length+1,4],dtype=float)
sol0=np.array([0.99, 0.005, 0.005, 0])
cum_result[0,:]=sol0
for i in range(time_length):
    sol1=odeint(Pfun, sol0, [0,interval_length])
    if(abs(sol1[1,0]-sol1[0,0])>thorehold):
        sol1[1,:]=sol1[0,:]-(sol1[0,:]-sol1[1,:])/1.4
        sol0=sol1[1,:]
    else:
        sol0=sol1[1,:]
    cum_result[i+1,:]=sol0

 #得到每段时间间隔内的阅读量
cum_result2=cum_result[:,1]+cum_result[:,2]+cum_result[:,3]
result2=np.array([0.000]*len(cum_result2))
for i in range(len(cum_result2)):
    if(i==0):
```

```
        result2[i]=cum_result2[i]
    else:
        result2[i]=cum_result2[i]-cum_result2[i-1]
result2=result2*N
#获取原始数据与之对比
#定义时间区间
endtime=21
interval_length=1
t=np.arange(0, endtime, interval_length)  #创建时间点

#寻求微分方程数值解
sol1=odeint(Pfun, [0.99, 0.005, 0.005, 0], t)

#得到每段时间间隔内的阅读量
cum_result=(sol1[:,1]+sol1[:,2]+sol1[:,3])*N
result1=np.array([0]*endtime)
for i in range(endtime):
    if(i==0):
        result1[i]=cum_result[i]
    else:
        result1[i]=cum_result[i]-cum_result[i-1]


#数据可视化
fig, ax = plt.subplots(figsize=(7,5.25))
ax.plot(result2, label='限流前仿真结果',color='r')
ax.plot(result1, label='限流后仿真结果',color='cyan')
ax.set(ylabel='阅读量', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 11：基于“定向免疫”的限流算法

```
import collections
import networkx as nx
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
```

```
import time

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

G1=nx.Graph()

#add nodes
for i in range(1000):
    G1.add_node(i, classic='s')

#add edges
for i in range(200):
    for j in range(200):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(200,400):
    for j in range(200,400):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(400,600):
    for j in range(400,600):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(600,800):
    for j in range(600,800):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(800,1000):
    for j in range(800,1000):
        if(random.random()>0.6):
```

```
            G1.add_edge(i, j)
        else:
            continue

for i in range(1000):
    for j in range(1000):
        if(random.random()>0.98):
            G1.add_edge(i, j)
        else:
            continue

G2=nx.Graph()

#add nodes
for i in range(1000):
    G2.add_node(i, classic='s')

#add edges
for i in range(200):
    for j in range(200):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(200,400):
    for j in range(200,400):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(400,600):
    for j in range(400,600):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(600,800):
    for j in range(600,800):
        if(random.random()>0.6):
            G2.add_edge(i, j)
```

```
        else:
            continue

for i in range(800,1000):
    for j in range(800,1000):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(1000):
    for j in range(1000):
        if(random.random()>0.98):
            G2.add_edge(i, j)
        else:
            continue




def countnumber(G):
    classic = nx.get_node_attributes(G, 'classic')
    counter6=0
    for i in range(len(classic)):
        if((classic[i]=='I') or (classic[i]=='E')):
            counter6+=1
    return counter6

def speard(G,setof_I):
    for j in range(len(setof_I)):
        classic = nx.get_node_attributes(G, 'classic')
        neighbourhood=G[setof_I[j]]._atlas
        for v in neighbourhood.keys():
            if(classic[v]=='s'):
                choice=random.random()
                if(choice<prob*0.5):
                    G.add_node(v, classic = 'E')
                elif(prob*0.5<choice<prob*0.8):
                    G.add_node(v, classic = 'I')
                    setof_I.append(v)
                elif(prob*0.8<choice<prob):
                    G.add_node(v, classic = 'R')
```

```
def cum_to_in(list_):
    result=np.array([0]*len(list_))
    for i in range(len(list_)):
        if(i==0):
            result[i]=list_[i]
        else:
            result[i]=list_[i]-list_[i-1]
    return result


#限流前
s_node1=random.randint(1,1001)
s_node2=random.randint(1,1001)

G1.add_node(s_node1 , classic='I')
G1.add_node(s_node2 , classic='I')

setof_I1=[]
setof_I1.append(s_node1)
setof_I1.append(s_node2)
counter1=np.array([0]*21)
counter1[0]=countnumber(G1)


prob=0.02

for i in range(1,len(counter1)):
    speard(G1,setof_I1)
    counter1[i]=countnumber(G1)

pre_result=cum_to_in(counter1)


#限流后
prob=0.016
mianyigeshu=250
sorted_list=sorted(G2.degree, key=lambda x: x[1], reverse=True
)
for i in range(mianyigeshu):
    G2.add_node(sorted_list[i][0] , classic='R')

s_node3=random.randint(1,1001)
```

```
s_node4=random.randint(1,1001)

G2.add_node(s_node3 , classic='I')
G2.add_node(s_node4 , classic='I')

setof_I2=[]
setof_I2.append(s_node3)
setof_I2.append(s_node4)
counter2=np.array([0]*21)
counter2[0]=countnumber(G2)


for i in range(1,len(counter2)):
    speard(G2,setof_I2)
    counter2[i]=countnumber(G2)

aft_result=cum_to_in(counter2)

N=177000000*1.4  #定义总人口数
pre_result=pre_result*N
aft_result=aft_result*N
#数据可视化
fig, ax = plt.subplots(figsize=(7,5.25))
ax.plot(pre_result, label='限流前仿真结果',color='r')
ax.plot(aft_result, label='限流后仿真结果',color='cyan')
ax.set(ylabel='已浏览信息人群占比变化率', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 12：社交网络图绘制

```
import collections
import networkx as nx
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import time
```

```
G=nx.Graph()

#add nodes
for i in range(100):
    G.add_node(i, classic='s')

#add edges
for i in range(20):
    for j in range(20):
        if(random.random()>0.6):
            G.add_edge(i, j)
        else:
            continue

for i in range(20,40):
    for j in range(20,40):
        if(random.random()>0.6):
            G.add_edge(i, j)
        else:
            continue

for i in range(40,60):
    for j in range(40,60):
        if(random.random()>0.6):
            G.add_edge(i, j)
        else:
            continue

for i in range(60,80):
    for j in range(60,80):
        if(random.random()>0.6):
            G.add_edge(i, j)
        else:
            continue

for i in range(80,100):
    for j in range(80,100):
        if(random.random()>0.6):
            G.add_edge(i, j)
        else:
            continue
```

```
for i in range(100):
    for j in range(100):
        if(random.random()>0.98):
            G.add_edge(i, j)
        else:
            continue


nx.draw(G, with_labels=True, node_color='cyan',edge_color='gre
en')
plt.show()



def countnumber(G):
    classic = nx.get_node_attributes(G, 'classic')
    counter2=0
    for i in range(len(classic)):
        if(classic[i]!='s'):
            counter2+=1
    return counter2

def speard(G):
    for j in range(len(setof_I)):
        classic = nx.get_node_attributes(G, 'classic')
        neighbourhood=G[setof_I[j]]._atlas
        for v in neighbourhood.keys():
            if(classic[v]=='s'):
                choice=random.random()
                if(choice<0.1875):
                    G.add_node(v, classic = 'E')
                elif(0.1875<choice<0.3):
                    G.add_node(v, classic = 'I')
                    setof_I.append(v)
                elif(0.3<choice<0.375):
                    G.add_node(v, classic = 'R')


def cum_to_in(list_):
    result=np.array([0]*len(list_))
    for i in range(len(list_)):
        if(i==0):
            result[i]=list_[i]
```

```
        else:
            result[i]=list_[i]-list_[i-1]
    return result

s_node1=random.randint(1,101)
s_node2=random.randint(1,101)

G.add_node(s_node1 , classic='I')
G.add_node(s_node2 , classic='I')

setof_I=[]
setof_E=[]
setof_R=[]
setof_I.append(s_node1)
setof_I.append(s_node2)
counter=np.array([0]*21)
counter[0]=countnumber(G)



for i in range(1,len(counter)):
    speard(G)
    counter[i]=countnumber(G)
    color_list=np.array(['      ']*100)

classic = nx.get_node_attributes(G, 'classic')

for i in range(len(color_list)):
    lei=classic.get(i)
    if (lei=='S'):
        color_list[i]='cyan'
    elif(lei=='E'):
        color_list[i]='gold'
    elif(lei=='I'):
        color_list[i]='violet'
    elif(lei=='R'):
        color_list[i]='salmon'

#数据可视化
nx.draw_networkx(G,node_color=color_list,edge_color='green' ,w
ith_labels=True)
plt.show()
```

```
pre_result=cum_to_in(counter)
```

附录 13：政府报告 1

```
import collections
import networkx as nx
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import time

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

G1=nx.Graph()

#add nodes
for i in range(1000):
    G1.add_node(i, classic='s')

#add edges
for i in range(200):
    for j in range(200):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(200,400):
    for j in range(200,400):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(400,600):
    for j in range(400,600):
```

```
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(600,800):
    for j in range(600,800):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(800,1000):
    for j in range(800,1000):
        if(random.random()>0.6):
            G1.add_edge(i, j)
        else:
            continue

for i in range(1000):
    for j in range(1000):
        if(random.random()>0.98):
            G1.add_edge(i, j)
        else:
            continue

G2=nx.Graph()

#add nodes
for i in range(1000):
    G2.add_node(i, classic='s')

#add edges
for i in range(200):
    for j in range(200):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(200,400):
    for j in range(200,400):
        if(random.random()>0.6):
```

```
            G2.add_edge(i, j)
        else:
            continue

for i in range(400,600):
    for j in range(400,600):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(600,800):
    for j in range(600,800):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(800,1000):
    for j in range(800,1000):
        if(random.random()>0.6):
            G2.add_edge(i, j)
        else:
            continue

for i in range(1000):
    for j in range(1000):
        if(random.random()>0.98):
            G2.add_edge(i, j)
        else:
            continue



def countnumber(G):
    classic = nx.get_node_attributes(G, 'classic')
    counter6=0
    for i in range(len(classic)):
        if((classic[i]=='I') or (classic[i]=='E')):
            counter6+=1
    return counter6
```

```
def speard(G,setof_I):
    for j in range(len(setof_I)):
        classic = nx.get_node_attributes(G, 'classic')
        neighbourhood=G[setof_I[j]]._atlas
        for v in neighbourhood.keys():
            if(classic[v]=='s'):
                choice=random.random()
                if(choice<prob*0.5):
                    G.add_node(v, classic = 'E')
                elif(prob*0.5<choice<prob*0.8):
                    G.add_node(v, classic = 'I')
                    setof_I.append(v)
                elif(prob*0.8<choice<prob):
                    G.add_node(v, classic = 'R')


def cum_to_in(list_):
    result=np.array([0]*len(list_))
    for i in range(len(list_)):
        if(i==0):
            result[i]=list_[i]
        else:
            result[i]=list_[i]-list_[i-1]
    return result


#限流前
s_node1=random.randint(1,1001)
s_node2=random.randint(1,1001)

G1.add_node(s_node1 , classic='I')
G1.add_node(s_node2 , classic='I')

setof_I1=[]
setof_I1.append(s_node1)
setof_I1.append(s_node2)
counter1=np.array([0]*21)
counter1[0]=countnumber(G1)


prob=0.02

for i in range(1,len(counter1)):
```

```
    speard(G1,setof_I1)
    counter1[i]=countnumber(G1)

pre_result=cum_to_in(counter1)


#限流后
prob=0.016
mianyigeshu=250
sorted_list=sorted(G2.degree, key=lambda x: x[1], reverse=True
)
for i in range(mianyigeshu):
    G2.add_node(sorted_list[i][0] , classic='R')

s_node3=random.randint(1,1001)
s_node4=random.randint(1,1001)

G2.add_node(s_node3 , classic='I')
G2.add_node(s_node4 , classic='I')

setof_I2=[]
setof_I2.append(s_node3)
setof_I2.append(s_node4)
counter2=np.array([0]*21)
counter2[0]=countnumber(G2)


for i in range(1,len(counter2)):
    speard(G2,setof_I2)
    counter2[i]=countnumber(G2)

aft_result=cum_to_in(counter2)

N=177000000*1.4  #定义总人口数
pre_result=pre_result/1000
aft_result=aft_result/1000
#数据可视化
fig, ax = plt.subplots(figsize=(7,5.25))
ax.plot(pre_result, label='限流前仿真结果',color='r')
ax.plot(aft_result, label='限流后仿真结果',color='cyan')
ax.set(ylabel='已浏览信息人群占比变化率', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
```

```
ax.legend()#添加图例标注
plt.show()
```

附录 14：政府报告 2

```
from scipy.integrate import odeint
from sympy.abc import t
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#载入现有数据
real=pd.read_excel("F://建模//深圳杯//上海数据.xlsx")
x=np.array(['                ']*len(real))
for i in range(len(real)):
    x[i]=str(real.iloc[i,0])[0:5]
y=np.array(real.iloc[:,1])


cum_real=np.array([0.00]*len(y))
for i in range(len(y)):
    if (i==0):
        cum_real[i]=y[i]
    else:
        cum_real[i]=y[i]+cum_real[i-1]

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

#常量赋值
lambda_=0.4
rou=0.2
lambda_1=lambda_*0.5
lambda_2=lambda_*0.3
lambda_3=lambda_*0.2
rou_1=rou*0.5
rou_2=rou*0.3
rou_3=rou*0.2
N=92000000

#定义微分方程组为一类方程
```

```
class functions():
    def __init__(self,parameter1):
        self.parameter1=parameter1

    result=0

    def calculate(self,S,I):
        self.result=self.parameter1*S*I

f1=functions(-(rou+lambda_))
f2=functions(rou_1+lambda_1)
f3=functions(rou_2+lambda_2)
f4=functions(rou_3+lambda_3)
f1.calculate(1,5)
a=f1.result
print(a)
#定义微分方程
def Pfun(y,x):
    S, E, I ,R= y;
    fun1=-(lambda_+rou)*S*I
    fun2=(lambda_1+rou_1)*S*I
    fun3=(lambda_2+rou_2)*S*I
    fun4=(lambda_3+rou_3)*S*I
    return np.array([fun1, fun2, fun3, fun4])


#定义时间区间
time_length=len(real)
interval_length=90/len(real)
#创建时间点

#寻求微分方程数值解
thorehold=0.02#设定阈值
cum_result=np.empty([time_length,4],dtype=float)
sol0=np.array([0.99, 0.005, 0.005, 0])
cum_result[0,:]=sol0
for i in range(time_length-1):
    sol1=odeint(Pfun, sol0, [0,interval_length])
    if(abs(sol1[1,0]-sol1[0,0])>thorehold):
        sol1[1,:]=sol1[0,:]-(sol1[0,:]-sol1[1,:])/1.4
        sol0=sol1[1,:]
    else:
        sol0=sol1[1,:]
```

```
    cum_result[i+1,:]=sol0

#得到比例
cum_result_=cum_result[:,1]+cum_result[:,2]/2.3
cum_real=cum_real/N
#数据可视化
fig, ax = plt.subplots(figsize=(7,5.25))
ax.plot(x, cum_result_, label='本文模型仿真结果',color='r')
ax.plot(x, cum_real, label='微博统计数据',color='cyan')
ax.set(ylabel='已浏览信息人群占比变化率', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 15：灵敏度分析 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

real=pd.read_excel("F://建模//深圳杯//上海数据.xlsx")

x=np.array(['                ']*len(real))
for i in range(len(real)):
    x[i]=str(real.iloc[i,0])[0:5]
y=np.array(real.iloc[:,1])


#定义迭代函数
def f(S,E,I,R,t,lambda_,rou):
    lambda_1=lambda_*0.3
    lambda_2=lambda_*0.5
    lambda_3=lambda_*0.2
    rou_1=rou*0.3
    rou_2=rou*0.5
```

```
    rou_3=rou*0.2
    S_N=S-t*(lambda_+rou)*S
    E_N=E+t*(lambda_1+rou_1)*S
    I_N=I+t*(lambda_2+rou_2)*S
    R_N=R+t*(lambda_3+rou_3)*S
    return S_N,E_N,I_N,R_N

#定义初始状态
S=0.99
E=0.005
I=0.005
R=0
N=103000000
#选择时间跨度
t=0.45

#迭代
def s(S,E,I,R,t,lambda_,rou):
    a=np.array([0]*21)
    a[0]=(E+I)*N
    a[1]=6252762+(lambda_+rou)*4222222
    for i in range(2,21):
        S_N,E_N,I_N,R_N=f(S,E,I,R,t,lambda_,rou)
        #print((I_N+E_N-I-E)*N)
        a[i]=(I_N+E_N-I-E)*N
        S,E,I,R=S_N,E_N,I_N,R_N
        #print(I_new)
    return a

a1=s(S,E,I,R,t,0.8,0.2)
a2=s(S,E,I,R,t,0.3,0.5)
a3=s(S,E,I,R,t,0.7,0.5)

#画图
fig, ax = plt.subplots(figsize=(6,4))
ax.plot(x, a1[0:12],label='0.8,0.2')
ax.plot(x, a2[0:12], label='0.3,0.5')
ax.plot(x, a3[0:12], label='0.7,0.5')
ax.set(ylabel='阅读量', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 16：灵敏度分析 2

```
import collections
import networkx as nx
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import time

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

G1=nx.Graph()
G2=nx.Graph()
G3=nx.Graph()
G4=nx.Graph()

#add nodes
for i in range(1000):
    G1.add_node(i, classic='s')
    G2.add_node(i, classic='s')
    G3.add_node(i, classic='s')
    G4.add_node(i, classic='s')

#add edges
for i in range(200):
    for j in range(200):
        if(random.random()>0.6):
            G1.add_edge(i, j)
            G2.add_edge(i, j)
            G3.add_edge(i, j)
            G4.add_edge(i, j)

        else:
            continue

for i in range(200,400):
    for j in range(200,400):
        if(random.random()>0.6):
            G1.add_edge(i, j)
            G2.add_edge(i, j)
            G3.add_edge(i, j)
```

```
            G4.add_edge(i, j)
        else:
            continue

for i in range(400,600):
    for j in range(400,600):
        if(random.random()>0.6):
            G1.add_edge(i, j)
            G2.add_edge(i, j)
            G3.add_edge(i, j)
            G4.add_edge(i, j)
        else:
            continue

for i in range(600,800):
    for j in range(600,800):
        if(random.random()>0.6):
            G1.add_edge(i, j)
            G2.add_edge(i, j)
            G3.add_edge(i, j)
            G4.add_edge(i, j)
        else:
            continue

for i in range(800,1000):
    for j in range(800,1000):
        if(random.random()>0.6):
            G1.add_edge(i, j)
            G2.add_edge(i, j)
            G3.add_edge(i, j)
            G4.add_edge(i, j)
        else:
            continue

for i in range(1000):
    for j in range(1000):
        if(random.random()>0.98):
            G1.add_edge(i, j)
            G2.add_edge(i, j)
            G3.add_edge(i, j)
            G4.add_edge(i, j)
        else:
            continue
```

```
def countnumber(G):
    classic = nx.get_node_attributes(G, 'classic')
    counter6=0
    for i in range(len(classic)):
        if((classic[i]=='I') or (classic[i]=='E')):
            counter6+=1
    return counter6

def speard(G,setof_I):
    for j in range(len(setof_I)):
        classic = nx.get_node_attributes(G, 'classic')
        neighbourhood=G[setof_I[j]]._atlas
        for v in neighbourhood.keys():
            if(classic[v]=='s'):
                choice=random.random()
                if(choice<prob*0.5):
                    G.add_node(v, classic = 'E')
                elif(prob*0.5<choice<prob*0.8):
                    G.add_node(v, classic = 'I')
                    setof_I.append(v)
                elif(prob*0.8<choice<prob):
                    G.add_node(v, classic = 'R')


def cum_to_in(list_):
    result=np.array([0]*len(list_))
    for i in range(len(list_)):
        if(i==0):
            result[i]=list_[i]
        else:
            result[i]=list_[i]-list_[i-1]
    return result


#G1
prob=0.02
mianyigeshu1=50
sorted_list=sorted(G1.degree, key=lambda x: x[1], reverse=True
)
for i in range(mianyigeshu1):
    G2.add_node(sorted_list[i][0] , classic='R')
```

```
s_node1=random.randint(1,1001)
s_node2=random.randint(1,1001)

G1.add_node(s_node1 , classic='I')
G1.add_node(s_node2 , classic='I')

setof_I1=[]
setof_I1.append(s_node1)
setof_I1.append(s_node2)
counter1=np.array([0]*21)
counter1[0]=countnumber(G2)

for i in range(1,len(counter1)):
    speard(G1,setof_I1)
    counter1[i]=countnumber(G1)


#G2
prob=0.02
mianyigeshu2=150
sorted_list=sorted(G2.degree, key=lambda x: x[1], reverse=True
)
for i in range(mianyigeshu2):
    G2.add_node(sorted_list[i][0] , classic='R')

s_node3=random.randint(1,1001)
s_node4=random.randint(1,1001)

G2.add_node(s_node3 , classic='I')
G2.add_node(s_node4 , classic='I')

setof_I2=[]
setof_I2.append(s_node3)
setof_I2.append(s_node4)
counter2=np.array([0]*21)
counter2[0]=countnumber(G2)

for i in range(1,len(counter2)):
    speard(G2,setof_I2)
    counter2[i]=countnumber(G2)

#G3
```

```
prob=0.02
mianyigeshu3=250
sorted_list=sorted(G3.degree, key=lambda x: x[1], reverse=True
)
for i in range(mianyigeshu3):
    G3.add_node(sorted_list[i][0] , classic='R')

s_node5=random.randint(1,1001)
s_node6=random.randint(1,1001)

G3.add_node(s_node5 , classic='I')
G3.add_node(s_node6 , classic='I')

setof_I3=[]
setof_I3.append(s_node5)
setof_I3.append(s_node6)
counter3=np.array([0]*21)
counter3[0]=countnumber(G3)

for i in range(1,len(counter3)):
    speard(G3,setof_I3)
    counter3[i]=countnumber(G3)

#G4
prob=0.02
mianyigeshu4=350
sorted_list=sorted(G4.degree, key=lambda x: x[1], reverse=True
)
for i in range(mianyigeshu4):
    G4.add_node(sorted_list[i][0] , classic='R')

s_node7=random.randint(1,1001)
s_node8=random.randint(1,1001)

G4.add_node(s_node7 , classic='I')
G4.add_node(s_node8 , classic='I')

setof_I4=[]
setof_I4.append(s_node7)
setof_I4.append(s_node8)
counter4=np.array([0]*21)
counter4[0]=countnumber(G4)
```

```
for i in range(1,len(counter4)):
    speard(G4,setof_I4)
    counter4[i]=countnumber(G4)



G1_result=cum_to_in(counter1)
G2_result=cum_to_in(counter2)
G3_result=cum_to_in(counter3)
G4_result=cum_to_in(counter4)

N=177000000*1.4  #定义总人口数
G1_result=G1_result*N
G2_result=G2_result*N
G3_result=G3_result*N
G4_result=G4_result*N

#数据可视化
fig, ax = plt.subplots(figsize=(7,5.25))
ax.plot(G1_result, label='免疫节点数=50',color='r')
ax.plot(G2_result, label='免疫节点数=100',color='magenta')
ax.plot(G3_result, label='免疫节点数=150',color='blue')
ax.plot(G4_result, label='免疫节点数=200',color='cyan')
ax.set(ylabel='阅读量', xlabel='时间')
plt.xticks(rotation=60)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.legend()#添加图例标注
plt.show()
```

附录 17：微博评论原始数据

评论 id	发布时间	评论内容
4.84E+15	Sat Nov 19 16:53:00 +0800 2022	RIP 还记得当年在春晚给大熊猫起名投票“团团圆圆”…
4.84E+15	Sat Nov 19 17:19:51 +0800 2022	#大熊猫团团不幸离世# 2008 年全民投票取名团团圆圆带着美好期盼赴台，在台北动物园生活了 14 年，和圆圆育有两个女儿，圆仔和圆宝，如果不得病 18 岁还不算太高龄，之前还有病情好转，这两天又

		反复多次发作，癫痫疾病团团受苦了🙏，团团回胖达星球了
4.84E+15	Sat Nov 19 16:27:10 +0800 2022	团团和圆圆的名字当初还是我们一起发短信取的
4.84E+15	Sat Nov 19 17:47:34 +0800 2022	是不是当年送给台湾省的团团圆圆？没记错的话是全国征集名字，那时候小，也给俩国宝起了名字，我麻麻说，最后一定会叫团团圆圆，是祖国统一的愿景。
4.84E+15	Sat Nov 19 17:07:52 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 17:10:31 +0800 2022	想到当初征集名字的时候
4.84E+15	Sat Nov 19 17:20:07 +0800 2022	团团回去熊猫星了
4.84E+15	Sat Nov 19 16:56:17 +0800 2022	唉，可以把它带回来安葬吗？
4.84E+15	Sat Nov 19 17:12:36 +0800 2022	每次看到大熊猫 心里就莫名其妙的稀罕就像稀罕自己家宠爱着的猫猫一样
4.84E+15	Sat Nov 19 16:46:01 +0800 2022	记得当时的春晚全国征集姓名，最后选了团团、圆圆这两个名字，希望台湾与祖国大陆早日团圆
4.84E+15	Sat Nov 19 16:30:09 +0800 2022	小天使一路走好吧
4.84E+15	Sat Nov 19 16:31:58 +0800 2022	希望团团在熊猫星上能够不再有病痛，有吃不完的竹子、竹笋和水果。
4.84E+15	Sat Nov 19 17:11:48 +0800 2022	回熊猫星了🙏
4.84E+15	Sat Nov 19 18:18:00 +0800 2022	宝贝，回潘达星要开开心心，再见
4.84E+15	Sat Nov 19 17:06:51 +0800 2022	痛心!!
4.84E+15	Sat Nov 19 17:06:57 +0800 2022	团团
4.84E+15	Sat Nov 19 16:53:41 +0800 2022	团团
4.84E+15	Sat Nov 19 17:02:56 +0800 2022	小时候在新闻里所熟知的团团啊
4.84E+15	Sat Nov 19 16:34:35 +0800 2022	团团那么可爱的团团去了熊猫星球
4.84E+15	Sat Nov 19 17:10:23 +0800 2022	
4.84E+15	Sat Nov 19 16:17:04 +0800 2022	真难过，希望送出去的熊猫都平平安安
4.84E+15	Sat Nov 19 18:18:06 +0800 2022	愿胖达星球没有痛苦
4.84E+15	Sat Nov 19 16:38:01 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:32:42 +0800 2022	再见了，团团
4.84E+15	Sat Nov 19 16:19:42 +0800 2022	我们的团团😭😭😭
4.84E+15	Sat Nov 19 17:11:25 +0800 2022	🙏🙏
4.84E+15	Sat Nov 19 17:06:35 +0800 2022	宝贝，再见
4.84E+15	Sat Nov 19 17:10:39 +0800 2022	团团在天上要幸福呀
4.84E+15	Sat Nov 19 17:09:14 +0800 2022	团团，一路走好
4.84E+15	Sat Nov 19 17:06:31 +0800 2022	去属于你自己的竹林吧
4.84E+15	Sat Nov 19 18:24:54 +0800 2022	团团走好，愿你在熊猫星球自由的奔跑
4.84E+15	Sat Nov 19 16:52:27 +0800 2022	🙏🙏🙏

4.84E+15	Sat Nov 19 16:33:25 +0800 2022	团团再见 希望另一个世界没有痛苦
4.84E+15	Sat Nov 19 16:29:09 +0800 2022	在潘达星要开心哦
4.84E+15	Sat Nov 19 16:17:17 +0800 2022	好难过，团团走好
4.84E+15	Sat Nov 19 18:03:36 +0800 2022	宝贝再见
4.84E+15	Sat Nov 19 17:16:29 +0800 2022	团团圆圆是大家共同投票决定的名字没有人不知道的 太爱它们了
4.84E+15	Sat Nov 19 17:08:13 +0800 2022	555555 团团
4.84E+15	Sat Nov 19 17:06:00 +0800 2022	转发微博
4.84E+15	Sat Nov 19 17:05:59 +0800 2022	天呢
4.84E+15	Sat Nov 19 16:52:47 +0800 2022	希望它一路走好，在胖星球开心
4.84E+15	Sat Nov 19 16:51:22 +0800 2022	哭了，还记得当初取名字的时候，我叫圆圆，所以好喜欢团团，没想到
4.84E+15	Sat Nov 19 16:49:47 +0800 2022	一路走好吧，团团！
4.84E+15	Sat Nov 19 16:49:14 +0800 2022	团团🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:49:01 +0800 2022	我今天早上看到团团的视频了
4.84E+15	Sat Nov 19 16:46:14 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:44:04 +0800 2022	团团宝宝
4.84E+15	Sat Nov 19 16:43:54 +0800 2022	团团一路安好，祝你未来在那边依旧生活美好
4.84E+15	Sat Nov 19 16:43:21 +0800 2022	月初还说好好
4.84E+15	Sat Nov 19 16:42:23 +0800 2022	我们的团团那么可爱
4.84E+15	Sat Nov 19 16:41:55 +0800 2022	我在熊猫星好好生活吧
4.84E+15	Sat Nov 19 16:39:53 +0800 2022	希望每只滚滚都能健健康康开开心心
4.84E+15	Sat Nov 19 16:39:37 +0800 2022	好难过
4.84E+15	Sat Nov 19 16:38:15 +0800 2022	愿天堂没有苦痛
4.84E+15	Sat Nov 19 16:36:43 +0800 2022	乖乖，去熊猫星要好好的。
4.84E+15	Sat Nov 19 16:35:17 +0800 2022	亲眼见过的团。
4.84E+15	Sat Nov 19 16:34:35 +0800 2022	才在电视里看到新闻
4.84E+15	Sat Nov 19 16:32:21 +0800 2022	现在都还记得团团圆圆
4.84E+15	Sat Nov 19 16:27:20 +0800 2022	看到这个报道，有点伤感
4.84E+15	Sat Nov 19 16:25:10 +0800 2022	团团一路走好 天堂不再有痛苦
4.84E+15	Sat Nov 19 16:25:07 +0800 2022	我的宝贝心碎了
4.84E+15	Sat Nov 19 16:24:50 +0800 2022	还记得，当初海选名字呢
4.84E+15	Sat Nov 19 16:24:44 +0800 2022	团团宝贝
4.84E+15	Sat Nov 19 16:17:53 +0800 2022	🙏🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:16:48 +0800 2022	团团一路走好，回到潘达星开开心心🙏
4.84E+15	Sat Nov 19 16:16:27 +0800 2022	再见，团团
4.84E+15	Sat Nov 19 17:16:04 +0800 2022	对的，最后入选的有“和和美美”“团团圆圆”，取名“团团圆圆” //@-阿芋芋芋圆酱Zzz: 我至今还记得那年春晚让全国人民观众们短信投票两只大可爱🐼的名字
4.84E+15	Sat Nov 19 17:10:08 +0800 2022	回来乖乖的那边好吃好喝 也没有病痛

		乖乖的去玩吧
4.84E+15	Sat Nov 19 16:51:55 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 16:42:22 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:18:09 +0800 2022	难过
4.84E+15	Sat Nov 19 16:16:47 +0800 2022	团团
4.84E+15	Sat Nov 19 17:31:31 +0800 2022	团团回潘达星球了，那里没有病痛
4.84E+15	Sat Nov 19 16:49:35 +0800 2022	挺难过的🙏
4.84E+15	Sat Nov 19 16:44:51 +0800 2022	一路走好团团宝贝
4.84E+15	Sat Nov 19 16:26:37 +0800 2022	团团天堂里只有快乐没有病痛
4.84E+15	Sat Nov 19 16:22:57 +0800 2022	记得小时候 团团圆圆 团团一路走好天堂没有病痛
4.84E+15	Sat Nov 19 16:12:56 +0800 2022	小时候第一个记住的熊猫名字 团团和圆圆心疼你就这样离开了。
4.84E+15	Sat Nov 19 16:17:04 +0800 2022	真难过，希望送出去的熊猫都平平安安
4.84E+15	Sat Nov 19 18:18:06 +0800 2022	愿胖达星球没有痛苦
4.84E+15	Sat Nov 19 16:38:01 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:32:42 +0800 2022	再见了，团团
4.84E+15	Sat Nov 19 16:19:42 +0800 2022	我们的团团😭😭😭
4.84E+15	Sat Nov 19 17:11:25 +0800 2022	🙏🙏
4.84E+15	Sat Nov 19 17:06:35 +0800 2022	宝贝，再见
4.84E+15	Sat Nov 19 17:10:39 +0800 2022	团团在天上要幸福呀
4.84E+15	Sat Nov 19 17:09:14 +0800 2022	团团，一路走好
4.84E+15	Sat Nov 19 17:06:31 +0800 2022	去属于你自己的竹林吧
4.84E+15	Sat Nov 19 18:24:54 +0800 2022	团团走好，愿你在熊猫星球自由的奔跑
4.84E+15	Sat Nov 19 16:52:27 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:33:25 +0800 2022	团团再见 希望另一个世界没有痛苦
4.84E+15	Sat Nov 19 16:29:09 +0800 2022	在潘达星要开心哦
4.84E+15	Sat Nov 19 16:17:17 +0800 2022	好难过，团团走好
4.84E+15	Sat Nov 19 18:03:36 +0800 2022	宝贝再见
4.84E+15	Sat Nov 19 17:16:29 +0800 2022	团团圆圆是大家共同投票决定的名字没有人不知道的 太爱它们了
4.84E+15	Sat Nov 19 17:08:13 +0800 2022	555555 团团
4.84E+15	Sat Nov 19 17:06:00 +0800 2022	转发微博
4.84E+15	Sat Nov 19 17:05:59 +0800 2022	天呢
4.84E+15	Sat Nov 19 16:52:47 +0800 2022	希望它一路走好，在胖星球开心
4.84E+15	Sat Nov 19 16:51:22 +0800 2022	哭了，还记得当初取名字的时候，我叫圆圆，所以好喜欢团团，没想到
4.84E+15	Sat Nov 19 16:49:47 +0800 2022	一路走好吧，团团！
4.84E+15	Sat Nov 19 16:49:14 +0800 2022	团团🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:49:01 +0800 2022	我今天早上看到团团的视频了
4.84E+15	Sat Nov 19 16:46:14 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:44:04 +0800 2022	团团宝宝

4.84E+15	Sat Nov 19 16:43:54 +0800 2022	团团一路安好，祝你未来在那边依旧生活美好
4.84E+15	Sat Nov 19 16:43:21 +0800 2022	月初还说好好
4.84E+15	Sat Nov 19 16:42:23 +0800 2022	我们的团团那么可爱
4.84E+15	Sat Nov 19 16:41:55 +0800 2022	我在熊猫星好好生活吧
4.84E+15	Sat Nov 19 16:39:53 +0800 2022	希望每只滚滚都能健健康康开开心心
4.84E+15	Sat Nov 19 16:39:37 +0800 2022	好难过
4.84E+15	Sat Nov 19 16:38:15 +0800 2022	愿天堂没有苦痛
4.84E+15	Sat Nov 19 16:36:43 +0800 2022	乖乖，去熊猫星要好好的。
4.84E+15	Sat Nov 19 16:35:17 +0800 2022	亲眼见过的团。
4.84E+15	Sat Nov 19 16:34:35 +0800 2022	才在电视里看到新闻
4.84E+15	Sat Nov 19 16:32:21 +0800 2022	现在都还记得团团圆圆
4.84E+15	Sat Nov 19 16:27:20 +0800 2022	看到这个报道，有点伤感
4.84E+15	Sat Nov 19 16:25:10 +0800 2022	团团一路走好 天堂不再有痛苦
4.84E+15	Sat Nov 19 16:25:07 +0800 2022	我的宝贝心碎了
4.84E+15	Sat Nov 19 16:24:50 +0800 2022	还记得，当初海选名字呢
4.84E+15	Sat Nov 19 16:24:44 +0800 2022	团团宝贝
4.84E+15	Sat Nov 19 16:17:53 +0800 2022	🙏🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:16:48 +0800 2022	团团一路走好，回到潘达星开开心心🙏
4.84E+15	Sat Nov 19 16:16:27 +0800 2022	再见，团团
4.84E+15	Sat Nov 19 17:16:04 +0800 2022	对的，最后入选的有“和和美美”“团团圆圆”，取名“团团圆圆” //@-阿芋芋芋圆酱Zzz: 我至今还记得那年春晚让全国人民观众们短信投票两只大可爱🐼的名字
4.84E+15	Sat Nov 19 17:10:08 +0800 2022	回来乖乖的那边好吃好喝 也没有病痛乖乖的去玩吧
4.84E+15	Sat Nov 19 16:51:55 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 16:42:22 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:18:09 +0800 2022	难过
4.84E+15	Sat Nov 19 16:16:47 +0800 2022	团团
4.84E+15	Sat Nov 19 17:31:31 +0800 2022	团团回潘达星球了，那里没有病痛
4.84E+15	Sat Nov 19 16:49:35 +0800 2022	挺难过的🙏
4.84E+15	Sat Nov 19 16:44:51 +0800 2022	一路走好团团宝贝
4.84E+15	Sat Nov 19 16:26:37 +0800 2022	团团天堂里只有快乐没有病痛
4.84E+15	Sat Nov 19 16:22:57 +0800 2022	记得小时候 团团圆圆 团团一路走好天堂没有病痛
4.84E+15	Sat Nov 19 16:12:56 +0800 2022	小时候第一个记住的熊猫名字 团团和圆圆心疼你就这样离开了。
4.84E+15	Sat Nov 19 16:26:37 +0800 2022	团团天堂里只有快乐没有病痛
4.84E+15	Sat Nov 19 16:22:57 +0800 2022	记得小时候 团团圆圆 团团一路走好天堂没有病痛

4.84E+15	Sat Nov 19 16:12:56 +0800 2022	小时候第一个记住的熊猫名字 团团和圆圆心疼你就这样离开了。
4.84E+15	Sat Nov 19 16:17:04 +0800 2022	真难过，希望送出去的熊猫都平平安安
4.84E+15	Sat Nov 19 18:18:06 +0800 2022	愿胖达星球没有痛苦
4.84E+15	Sat Nov 19 16:38:01 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:32:42 +0800 2022	再见了，团团
4.84E+15	Sat Nov 19 16:19:42 +0800 2022	我们的团团😭😭😭
4.84E+15	Sat Nov 19 17:11:25 +0800 2022	🙏🙏
4.84E+15	Sat Nov 19 17:06:35 +0800 2022	宝贝，再见
4.84E+15	Sat Nov 19 17:10:39 +0800 2022	团团在天上要幸福呀
4.84E+15	Sat Nov 19 17:09:14 +0800 2022	团团，一路走好
4.84E+15	Sat Nov 19 17:06:31 +0800 2022	去属于你自己的竹林吧
4.84E+15	Sat Nov 19 18:24:54 +0800 2022	团团走好，愿你在熊猫星球自由的奔跑
4.84E+15	Sat Nov 19 16:52:27 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:33:25 +0800 2022	团团再见 希望另一个世界没有痛苦
4.84E+15	Sat Nov 19 16:29:09 +0800 2022	在潘达星要开心哦
4.84E+15	Sat Nov 19 16:17:17 +0800 2022	好难过，团团走好
4.84E+15	Sat Nov 19 18:03:36 +0800 2022	宝贝再见
4.84E+15	Sat Nov 19 17:16:29 +0800 2022	团团圆圆是大家共同投票决定的名字没有人不知道的 太爱它们了
4.84E+15	Sat Nov 19 17:08:13 +0800 2022	555555 团团
4.84E+15	Sat Nov 19 17:06:00 +0800 2022	转发微博
4.84E+15	Sat Nov 19 17:05:59 +0800 2022	天呢
4.84E+15	Sat Nov 19 16:52:47 +0800 2022	希望它一路走好，在胖星球开心
4.84E+15	Sat Nov 19 16:51:22 +0800 2022	哭了，还记得当初取名字的时候，我叫圆圆，所以好喜欢团团，没想到
4.84E+15	Sat Nov 19 16:49:47 +0800 2022	一路走好吧，团团！
4.84E+15	Sat Nov 19 16:49:14 +0800 2022	团团🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:49:01 +0800 2022	我今天早上看到团团的视频了
4.84E+15	Sat Nov 19 16:46:14 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:44:04 +0800 2022	团团宝宝
4.84E+15	Sat Nov 19 16:43:54 +0800 2022	团团一路安好，祝你未来在那边依旧生活美好
4.84E+15	Sat Nov 19 16:43:43 +0800 2022	
4.84E+15	Sat Nov 19 16:43:21 +0800 2022	月初还说好好
4.84E+15	Sat Nov 19 16:42:23 +0800 2022	我们的团团那么可爱
4.84E+15	Sat Nov 19 16:39:53 +0800 2022	希望每只滚滚都能健健康康开开心心
4.84E+15	Sat Nov 19 16:39:37 +0800 2022	好难过
4.84E+15	Sat Nov 19 16:38:15 +0800 2022	愿天堂没有苦痛
4.84E+15	Sat Nov 19 16:36:43 +0800 2022	乖乖，去熊猫星要好好的。
4.84E+15	Sat Nov 19 16:35:17 +0800 2022	亲眼见过的团。
4.84E+15	Sat Nov 19 16:34:35 +0800 2022	才在电视里看到新闻

4.84E+15	Sat Nov 19 16:32:21 +0800 2022	现在都还记得团团圆圆
4.84E+15	Sat Nov 19 16:27:20 +0800 2022	看到这个报道，有点伤感
4.84E+15	Sat Nov 19 16:25:10 +0800 2022	团团一路走好 天堂不再有痛苦
4.84E+15	Sat Nov 19 16:25:07 +0800 2022	我的宝贝心碎了
4.84E+15	Sat Nov 19 16:24:50 +0800 2022	还记得，当初海选名字呢
4.84E+15	Sat Nov 19 16:24:44 +0800 2022	团团宝贝
4.84E+15	Sat Nov 19 16:17:53 +0800 2022	🙏🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:16:48 +0800 2022	团团一路走好，回到潘达星开开心心🙏
4.84E+15	Sat Nov 19 16:16:27 +0800 2022	再见，团团
4.84E+15	Sat Nov 19 17:16:04 +0800 2022	对的，最后入选的有“和和美美”“团团圆圆”，取名“团团圆圆”//@-阿芋芋芋圆酱Zzz: 我至今还记得那年春晚让全国人民观众们短信投票两只大可爱🐼的名字
4.84E+15	Sat Nov 19 17:10:08 +0800 2022	回来乖乖的那边好吃好喝 也没有病痛乖乖的去玩吧
4.84E+15	Sat Nov 19 16:51:55 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 16:42:22 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:18:09 +0800 2022	难过
4.84E+15	Sat Nov 19 16:16:47 +0800 2022	团团
4.84E+15	Sat Nov 19 17:31:31 +0800 2022	团团回潘达星球了，那里没有病痛
4.84E+15	Sat Nov 19 16:49:35 +0800 2022	挺难过的🙏
4.84E+15	Sat Nov 19 16:45:59 +0800 2022	
4.84E+15	Sat Nov 19 16:44:51 +0800 2022	一路走好团团宝贝
4.84E+15	Sat Nov 19 16:42:23 +0800 2022	我们的团团那么可爱
4.84E+15	Sat Nov 19 16:41:55 +0800 2022	我在熊猫星好好生活吧
4.84E+15	Sat Nov 19 16:39:53 +0800 2022	希望每只滚滚都能健健康康开开心心
4.84E+15	Sat Nov 19 16:39:37 +0800 2022	好难过
4.84E+15	Sat Nov 19 16:38:15 +0800 2022	愿天堂没有苦痛
4.84E+15	Sat Nov 19 16:36:43 +0800 2022	乖乖，去熊猫星要好好的。
4.84E+15	Sat Nov 19 16:35:17 +0800 2022	亲眼见过的团。
4.84E+15	Sat Nov 19 16:34:35 +0800 2022	才在电视里看到新闻
4.84E+15	Sat Nov 19 16:32:21 +0800 2022	现在都还记得团团圆圆
4.84E+15	Sat Nov 19 16:27:31 +0800 2022	
4.84E+15	Sat Nov 19 16:27:20 +0800 2022	看到这个报道，有点伤感
4.84E+15	Sat Nov 19 16:25:10 +0800 2022	团团一路走好 天堂不再有痛苦
4.84E+15	Sat Nov 19 16:25:07 +0800 2022	我的宝贝心碎了
4.84E+15	Sat Nov 19 16:24:50 +0800 2022	还记得，当初海选名字呢
4.84E+15	Sat Nov 19 16:24:44 +0800 2022	团团宝贝
4.84E+15	Sat Nov 19 16:17:53 +0800 2022	🙏🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:16:48 +0800 2022	团团一路走好，回到潘达星开开心心🙏
4.84E+15	Sat Nov 19 16:16:27 +0800 2022	再见，团团

4.84E+15	Sat Nov 19 17:16:04 +0800 2022	对的，最后入选的有“和和美美”“团团圆圆”，取名“团团圆圆”//@-阿芋芋芋圆酱Zzz: 我至今还记得那年春晚让全国人民观众们短信投票两只大可爱🐼的名字
4.84E+15	Sat Nov 19 17:10:08 +0800 2022	回来乖乖的那边好吃好喝 也没有病痛乖乖的去玩吧
4.84E+15	Sat Nov 19 16:51:55 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 16:42:22 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:18:09 +0800 2022	难过
4.84E+15	Sat Nov 19 16:16:47 +0800 2022	团团
4.84E+15	Sat Nov 19 17:31:31 +0800 2022	团团回潘达星球了，那里没有病痛
4.84E+15	Sat Nov 19 16:49:35 +0800 2022	挺难过的🙏
4.84E+15	Sat Nov 19 16:44:51 +0800 2022	一路走好团团宝贝
4.84E+15	Sat Nov 19 16:26:37 +0800 2022	团团天堂里只有快乐没有病痛
4.84E+15	Sat Nov 19 16:22:57 +0800 2022	记得小时候 团团圆圆 团团一路走好天堂没有病痛
4.84E+15	Sat Nov 19 16:12:56 +0800 2022	小时候第一个记住的熊猫名字 团团和圆圆心疼你就这样离开了。
4.84E+15	Sat Nov 19 18:24:54 +0800 2022	团团走好，愿你在熊猫星球自由的奔跑
4.84E+15	Sat Nov 19 16:17:04 +0800 2022	真难过，希望送出去的熊猫都平平安安
4.84E+15	Sat Nov 19 18:18:06 +0800 2022	愿胖达星球没有痛苦
4.84E+15	Sat Nov 19 16:38:01 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:32:42 +0800 2022	再见了，团团
4.84E+15	Sat Nov 19 16:19:42 +0800 2022	我们的团团😭😭😭
4.84E+15	Sat Nov 19 17:11:25 +0800 2022	🙏🙏
4.84E+15	Sat Nov 19 17:06:35 +0800 2022	宝贝，再见
4.84E+15	Sat Nov 19 17:10:39 +0800 2022	团团在天上要幸福呀
4.84E+15	Sat Nov 19 17:09:14 +0800 2022	团团，一路走好
4.84E+15	Sat Nov 19 17:06:31 +0800 2022	去属于你自己的竹林吧
4.84E+15	Sat Nov 19 16:52:27 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:33:25 +0800 2022	团团再见 希望另一个世界没有痛苦
4.84E+15	Sat Nov 19 16:29:09 +0800 2022	在潘达星要开心哦
4.84E+15	Sat Nov 19 16:17:17 +0800 2022	好难过，团团走好
4.84E+15	Sat Nov 19 18:03:36 +0800 2022	宝贝再见
4.84E+15	Sat Nov 19 17:16:29 +0800 2022	团团圆圆是大家共同投票决定的名字没有人不知道的 太爱它们了
4.84E+15	Sat Nov 19 17:08:13 +0800 2022	555555 团团
4.84E+15	Sat Nov 19 17:06:00 +0800 2022	转发微博
4.84E+15	Sat Nov 19 17:05:59 +0800 2022	天呢
4.84E+15	Sat Nov 19 16:52:47 +0800 2022	希望它一路走好，在胖星球开心
4.84E+15	Sat Nov 19 16:51:22 +0800 2022	哭了，还记得当初取名字的时候，我叫圆圆，所以好喜欢团团，没想到

4.84E+15	Sat Nov 19 16:49:47 +0800 2022	一路走好吧，团团！
4.84E+15	Sat Nov 19 16:49:14 +0800 2022	团团🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:49:01 +0800 2022	我今天早上看到团团的视频了
4.84E+15	Sat Nov 19 16:46:14 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:44:04 +0800 2022	团团宝宝
4.84E+15	Sat Nov 19 16:43:54 +0800 2022	团团一路安好，祝你未来在那边依旧生活美好
4.84E+15	Sat Nov 19 16:43:21 +0800 2022	月初还说好好
4.84E+15	Sat Nov 19 16:17:04 +0800 2022	真难过，希望送出去的熊猫都平平安安
4.84E+15	Sat Nov 19 18:18:06 +0800 2022	愿胖达星球没有痛苦
4.84E+15	Sat Nov 19 16:38:01 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:32:42 +0800 2022	再见了，团团
4.84E+15	Sat Nov 19 16:19:42 +0800 2022	我们的团团😭😭😭
4.84E+15	Sat Nov 19 17:11:25 +0800 2022	🙏🙏
4.84E+15	Sat Nov 19 17:06:35 +0800 2022	宝贝，再见
4.84E+15	Sat Nov 19 17:10:39 +0800 2022	团团在天上要幸福呀
4.84E+15	Sat Nov 19 17:09:14 +0800 2022	团团，一路走好
4.84E+15	Sat Nov 19 17:06:31 +0800 2022	去属于你自己的竹林吧
4.84E+15	Sat Nov 19 18:24:54 +0800 2022	团团走好，愿你在熊猫星球自由的奔跑
4.84E+15	Sat Nov 19 16:52:27 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:33:25 +0800 2022	团团再见 希望另一个世界没有痛苦
4.84E+15	Sat Nov 19 16:29:09 +0800 2022	在潘达星要开心哦
4.84E+15	Sat Nov 19 16:17:17 +0800 2022	好难过，团团走好
4.84E+15	Sat Nov 19 18:03:36 +0800 2022	宝贝再见
4.84E+15	Sat Nov 19 17:16:29 +0800 2022	团团圆圆是大家共同投票决定的名字没有人不知道的 太爱它们了
4.84E+15	Sat Nov 19 17:08:13 +0800 2022	555555 团团
4.84E+15	Sat Nov 19 17:06:00 +0800 2022	转发微博
4.84E+15	Sat Nov 19 17:05:59 +0800 2022	天呢
4.84E+15	Sat Nov 19 16:52:47 +0800 2022	希望它一路走好，在胖星球开心
4.84E+15	Sat Nov 19 16:51:22 +0800 2022	哭了，还记得当初取名字的时候，我叫圆圆，所以好喜欢团团，没想到
4.84E+15	Sat Nov 19 16:49:47 +0800 2022	一路走好吧，团团！
4.84E+15	Sat Nov 19 16:49:14 +0800 2022	团团🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:49:01 +0800 2022	我今天早上看到团团的视频了
4.84E+15	Sat Nov 19 16:46:14 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:44:04 +0800 2022	团团宝宝
4.84E+15	Sat Nov 19 16:43:54 +0800 2022	团团一路安好，祝你未来在那边依旧生活美好
4.84E+15	Sat Nov 19 16:43:21 +0800 2022	月初还说好好
4.84E+15	Sat Nov 19 16:42:23 +0800 2022	我们的团团那么可爱
4.84E+15	Sat Nov 19 16:41:55 +0800 2022	我在熊猫星好好生活吧

4.84E+15	Sat Nov 19 16:39:53 +0800 2022	希望每只滚滚都能健健康康开开心心
4.84E+15	Sat Nov 19 16:39:37 +0800 2022	好难过
4.84E+15	Sat Nov 19 16:38:15 +0800 2022	愿天堂没有苦痛
4.84E+15	Sat Nov 19 16:36:43 +0800 2022	乖乖，去熊猫星要好好的。
4.84E+15	Sat Nov 19 16:35:17 +0800 2022	亲眼见过的团。
4.84E+15	Sat Nov 19 16:34:35 +0800 2022	才在电视里看到新闻
4.84E+15	Sat Nov 19 16:32:21 +0800 2022	现在都还记得团团圆圆
4.84E+15	Sat Nov 19 16:27:20 +0800 2022	看到这个报道，有点伤感
4.84E+15	Sat Nov 19 16:25:10 +0800 2022	团团一路走好 天堂不再有痛苦
4.84E+15	Sat Nov 19 16:25:07 +0800 2022	我的宝贝心碎了
4.84E+15	Sat Nov 19 16:24:50 +0800 2022	还记得，当初海选名字呢
4.84E+15	Sat Nov 19 16:24:44 +0800 2022	团团宝贝
4.84E+15	Sat Nov 19 16:17:53 +0800 2022	🙏🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:16:48 +0800 2022	团团一路走好，回到潘达星开开心心🙏
4.84E+15	Sat Nov 19 16:16:27 +0800 2022	再见，团团
4.84E+15	Sat Nov 19 17:16:04 +0800 2022	对的，最后入选的有“和和美美”“团团圆圆”，取名“团团圆圆” //@-阿芋芋芋圆酱Zzz: 我至今还记得那年春晚让全国人民观众们短信投票两只大可爱🐼的名字
4.84E+15	Sat Nov 19 17:10:08 +0800 2022	回来乖乖的那边好吃好喝 也没有病痛乖乖的去玩吧
4.84E+15	Sat Nov 19 16:51:55 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 16:42:22 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:18:09 +0800 2022	难过
4.84E+15	Sat Nov 19 16:16:47 +0800 2022	团团
4.84E+15	Sat Nov 19 17:31:31 +0800 2022	团团回潘达星球了，那里没有病痛
4.84E+15	Sat Nov 19 16:49:35 +0800 2022	挺难过的🙏
4.84E+15	Sat Nov 19 16:44:51 +0800 2022	一路走好团团宝贝
4.84E+15	Sat Nov 19 16:26:37 +0800 2022	团团天堂里只有快乐没有病痛
4.84E+15	Sat Nov 19 16:22:57 +0800 2022	记得小时候 团团圆圆 团团一路走好天堂没有病痛
4.84E+15	Sat Nov 19 16:12:56 +0800 2022	小时候第一个记住的熊猫名字 团团和圆圆心疼你就这样离开了。
4.84E+15	Sat Nov 19 16:44:51 +0800 2022	一路走好团团宝贝
4.84E+15	Sat Nov 19 16:26:37 +0800 2022	团团天堂里只有快乐没有病痛
4.84E+15	Sat Nov 19 16:22:57 +0800 2022	记得小时候 团团圆圆 团团一路走好天堂没有病痛
4.84E+15	Sat Nov 19 16:12:56 +0800 2022	小时候第一个记住的熊猫名字 团团和圆圆心疼你就这样离开了。
4.84E+15	Sat Nov 19 16:17:04 +0800 2022	真难过，希望送出去的熊猫都平平安安
4.84E+15	Sat Nov 19 18:18:06 +0800 2022	愿胖达星球没有痛苦
4.84E+15	Sat Nov 19 16:38:01 +0800 2022	🙏🙏🙏

4.84E+15	Sat Nov 19 16:32:42 +0800 2022	再见了，团团
4.84E+15	Sat Nov 19 16:19:42 +0800 2022	我们的团团😭😭😭
4.84E+15	Sat Nov 19 17:11:25 +0800 2022	🙏🙏
4.84E+15	Sat Nov 19 17:06:35 +0800 2022	宝贝，再见
4.84E+15	Sat Nov 19 17:01:12 +0800 2022	
4.84E+15	Sat Nov 19 17:10:39 +0800 2022	团团在天上要幸福呀
4.84E+15	Sat Nov 19 17:09:14 +0800 2022	团团，一路走好
4.84E+15	Sat Nov 19 17:06:31 +0800 2022	去属于你自己的竹林吧
4.84E+15	Sat Nov 19 18:24:54 +0800 2022	团团走好，愿你在熊猫星球自由的奔跑
4.84E+15	Sat Nov 19 16:52:27 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:33:25 +0800 2022	团团再见 希望另一个世界没有痛苦
4.84E+15	Sat Nov 19 16:29:09 +0800 2022	在潘达星要开心哦
4.84E+15	Sat Nov 19 16:17:17 +0800 2022	好难过，团团走好
4.84E+15	Sat Nov 19 18:03:36 +0800 2022	宝贝再见
4.84E+15	Sat Nov 19 17:16:29 +0800 2022	团团圆圆是大家共同投票决定的名字没有人不知道的　太爱它们了
4.84E+15	Sat Nov 19 17:08:13 +0800 2022	555555 团团
4.84E+15	Sat Nov 19 17:06:00 +0800 2022	转发微博
4.84E+15	Sat Nov 19 17:05:59 +0800 2022	天呢
4.84E+15	Sat Nov 19 16:52:47 +0800 2022	希望它一路走好，在胖星球开心
4.84E+15	Sat Nov 19 16:51:22 +0800 2022	哭了，还记得当初取名字的时候，我叫圆圆，所以好喜欢团团，没想到
4.84E+15	Sat Nov 19 16:49:47 +0800 2022	一路走好吧，团团！
4.84E+15	Sat Nov 19 16:49:14 +0800 2022	团团🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:49:01 +0800 2022	我今天早上看到团团的视频了
4.84E+15	Sat Nov 19 16:46:14 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:44:04 +0800 2022	团团宝宝
4.84E+15	Sat Nov 19 16:43:54 +0800 2022	团团一路安好，祝你未来在那边依旧生活美好
4.84E+15	Sat Nov 19 16:43:21 +0800 2022	月初还说好好
4.84E+15	Sat Nov 19 16:42:23 +0800 2022	我们的团团那么可爱
4.84E+15	Sat Nov 19 16:41:55 +0800 2022	我在熊猫星好好生活吧
4.84E+15	Sat Nov 19 16:39:53 +0800 2022	希望每只滚滚都能健健康康开开心心
4.84E+15	Sat Nov 19 16:39:37 +0800 2022	好难过
4.84E+15	Sat Nov 19 16:38:15 +0800 2022	愿天堂没有苦痛
4.84E+15	Sat Nov 19 16:36:43 +0800 2022	乖乖，去熊猫星要好好的。
4.84E+15	Sat Nov 19 16:35:17 +0800 2022	亲眼见过的团。
4.84E+15	Sat Nov 19 16:34:35 +0800 2022	才在电视里看到新闻
4.84E+15	Sat Nov 19 16:32:21 +0800 2022	现在都还记得团团圆圆
4.84E+15	Sat Nov 19 16:27:20 +0800 2022	看到这个报道，有点伤感
4.84E+15	Sat Nov 19 16:25:10 +0800 2022	团团一路走好 天堂不再有痛苦
4.84E+15	Sat Nov 19 16:25:07 +0800 2022	我的宝贝心碎了

4.84E+15	Sat Nov 19 16:24:50 +0800 2022	还记得，当初海选名字呢
4.84E+15	Sat Nov 19 16:24:44 +0800 2022	团团宝贝
4.84E+15	Sat Nov 19 16:17:53 +0800 2022	🙏🙏🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:16:48 +0800 2022	团团一路走好，回到潘达星开开心心🙏
4.84E+15	Sat Nov 19 16:16:27 +0800 2022	再见，团团
4.84E+15	Sat Nov 19 17:16:04 +0800 2022	对的，最后入选的有“和和美美”“团团圆圆”，取名“团团圆圆” //@-阿芋芋芋圆酱Zzz: 我至今还记得那年春晚让全国人民观众们短信投票两只大可爱🐼的名字
4.84E+15	Sat Nov 19 17:10:08 +0800 2022	回来乖乖的那边好吃好喝 也没有病痛乖乖的去玩吧
4.84E+15	Sat Nov 19 16:51:55 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 16:42:22 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 16:18:09 +0800 2022	难过
4.84E+15	Sat Nov 19 16:16:47 +0800 2022	团团
4.84E+15	Sat Nov 19 17:31:31 +0800 2022	团团回潘达星球了，那里没有病痛
4.84E+15	Sat Nov 19 16:49:35 +0800 2022	挺难过的🙏
4.84E+15	Sat Nov 19 16:44:51 +0800 2022	一路走好团团宝贝
评论 id	发布时间	用户城市
4.84E+15	Sat Nov 19 17:22:55 +0800 2022	潘达星球没有病痛 全都是吃不完的笋笋苹果窝窝头
4.84E+15	Sat Nov 19 17:18:57 +0800 2022	还记得那时候才小学，参加起名活动，选的团团圆圆，后来还中了张动物园的门票
4.84E+15	Sat Nov 19 17:19:52 +0800 2022	是春晚上征集名字的大熊猫吗
4.84E+15	Sat Nov 19 17:23:18 +0800 2022	团团回“家”了，接下来希望台湾能回归祖国，实现真正的团圆。
4.84E+15	Sat Nov 19 17:21:32 +0800 2022	天哪，我可爱的团团，好舍不得你
4.84E+15	Sat Nov 19 17:29:45 +0800 2022	在春晚上征集名字的团团圆圆，这么多年过去了，我们的团团
4.84E+15	Sat Nov 19 17:26:28 +0800 2022	团团走好，在喵星保佑圆圆健健康康的!
4.84E+15	Sat Nov 19 17:20:55 +0800 2022	一路走好
4.84E+15	Sat Nov 19 17:37:29 +0800 2022	我可爱的团团，一路走好 潘达星球没有病痛，全都是吃不完的笋笋苹果窝窝头
4.84E+15	Sat Nov 19 17:20:26 +0800 2022	团团
4.84E+15	Sat Nov 19 17:28:20 +0800 2022	宝贝一路走好
4.84E+15	Sat Nov 19 18:11:46 +0800 2022	我记得我们全家都说叫团团圆圆，后来元宵晚会的时候公部就叫团团圆圆，我们一家高兴好久
4.84E+15	Sat Nov 19 17:40:55 +0800 2022	希望在那边也会有很多人爱团团
4.84E+15	Sat Nov 19 17:20:43 +0800 2022	一路走好🙏
4.84E+15	Sat Nov 19 18:32:48 +0800 2022	熊有生老病死，惜别“团团”。也愿“圆圆”和“圆仔”“圆宝”一切安好!

4.84E+15	Sat Nov 19 17:48:52 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 18:15:32 +0800 2022	在另一个星球好好生活
4.84E+15	Sat Nov 19 17:37:12 +0800 2022	希望大熊猫们能健健康康快快乐乐的活着
4.84E+15	Sat Nov 19 16:26:22 +0800 2022	团团
4.84E+15	Sat Nov 19 17:28:33 +0800 2022	团团啊…是名字来源于春晚的团团啊…你，怎么…就走了呢……
4.84E+15	Sat Nov 19 17:35:06 +0800 2022	团团下辈子做个健康的大熊猫
4.84E+15	Sat Nov 19 17:53:53 +0800 2022	星球上会有很多好吃的，团团别把我们忘记了
4.84E+15	Sat Nov 19 18:30:32 +0800 2022	团团走好//@4oi_D:潘达星球没有病痛全都是吃不完的笋笋苹果窝窝头
4.84E+15	Sat Nov 19 18:18:15 +0800 2022	2008 年有了 qq 号第一个网名就叫做团团圆圆
4.84E+15	Sat Nov 19 18:13:47 +0800 2022	希望你是一颗闪亮的星星
4.84E+15	Sat Nov 19 18:17:58 +0800 2022	大宝贝团团
4.84E+15	Sat Nov 19 18:04:53 +0800 2022	团团去熊猫星要幸福
4.84E+15	Sat Nov 19 17:35:45 +0800 2022	我们的国宝宝宝又少了一员
4.84E+15	Sat Nov 19 19:32:25 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 18:31:42 +0800 2022	圆圆，你要好好的
4.84E+15	Sat Nov 19 18:02:56 +0800 2022	寶貝再見，别忘了我們很愛你
4.84E+15	Sat Nov 19 17:53:24 +0800 2022	团团走好，舍不得!
4.84E+15	Sat Nov 19 19:35:29 +0800 2022	想哭，毕业旅行的时候还去看过它，那时候它还躺着在晒太阳
4.84E+15	Sat Nov 19 18:26:03 +0800 2022	2008 年赠台湾的大熊猫之一，是大家一起起名的“团团”和“圆圆”，团团生病是之前就听到的消息了，一直在积极治疗，很顽强的坚持了好多天。愿 panda 星球没有病痛
4.84E+15	Sat Nov 19 17:35:06 +0800 2022	抱抱
4.84E+15	Sat Nov 19 17:29:44 +0800 2022	团团
4.84E+15	Sat Nov 19 19:35:19 +0800 2022	啊团团圆圆里面的团团
4.84E+15	Sat Nov 19 19:32:43 +0800 2022	睡吧，梦里还有好多好多香甜的竹子～
4.84E+15	Sat Nov 19 19:30:02 +0800 2022	呜呜呜团团宝宝走好在熊猫星球一定要健康快乐
4.84E+15	Sat Nov 19 18:31:56 +0800 2022	好宝宝
4.84E+15	Sat Nov 19 17:39:18 +0800 2022	救命，我很多年以前第一次见的大熊猫就是团团圆圆
4.84E+15	Sat Nov 19 17:36:48 +0800 2022	团团
4.84E+15	Sat Nov 19 17:17:23 +0800 2022	走好
4.84E+15	Sat Nov 19 17:26:13 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 17:22:42 +0800 2022	走好团团

4.84E+15	Sat Nov 19 17:13:23 +0800 2022	我好难过😫
4.84E+15	Sat Nov 19 19:29:47 +0800 2022	团团一路走好🙏
4.84E+15	Sat Nov 19 18:30:24 +0800 2022	看到这个消息，真的，瞬间泪奔了，团团，有这个名字的时候我还小呢，上初中呢
4.84E+15	Sat Nov 19 18:22:45 +0800 2022	看到那年春晚上征集的名字，大家都有参与，就很泪目
4.84E+15	Sat Nov 19 18:02:33 +0800 2022	宝贝一路走好 在熊猫星球要开心快乐
4.84E+15	Sat Nov 19 17:56:51 +0800 2022	希望圆圆好好的
4.84E+15	Sat Nov 19 17:56:17 +0800 2022	愿天堂没有病痛，一路走好🐼。
4.84E+15	Sat Nov 19 17:28:55 +0800 2022	团团
4.84E+15	Sat Nov 19 17:25:17 +0800 2022	一路走好团团
4.84E+15	Sat Nov 19 17:21:09 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 18:24:32 +0800 2022	是的，对这个场景记忆犹新，那时候不明白年龄小，根本不明白团团圆圆这个很普通的名字背后饱含的深意//@Bannerobby:是春晚上征集名字的大熊猫吗
4.84E+15	Sat Nov 19 18:23:00 +0800 2022	我们的国宝团团
4.84E+15	Sat Nov 19 18:19:50 +0800 2022	团团宝贝一路走好
4.84E+15	Sat Nov 19 18:19:23 +0800 2022	看完真的很难受，心里说不出来的感觉，希望它在下一个星球开开心心，没有病痛
4.84E+15	Sat Nov 19 18:13:14 +0800 2022	啊团团
4.84E+15	Sat Nov 19 17:57:02 +0800 2022	一路走好
4.84E+15	Sat Nov 19 17:56:26 +0800 2022	团团宝贝一路走好
4.84E+15	Sat Nov 19 17:51:28 +0800 2022	团团……
4.84E+15	Sat Nov 19 17:49:10 +0800 2022	团团妹妹，一路走好！姐姐爱你呦
4.84E+15	Sat Nov 19 17:47:44 +0800 2022	天呐！是小时候“团团圆圆”的团团么！
4.84E+15	Sat Nov 19 17:47:36 +0800 2022	我们团团去潘达星球了
4.84E+15	Sat Nov 19 17:47:11 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 17:39:42 +0800 2022	团团走好…在熊猫星也要健康幸福。
4.84E+15	Sat Nov 19 17:39:03 +0800 2022	可爱的团团
4.84E+15	Sat Nov 19 17:38:06 +0800 2022	可爱的团团，去看过你两次都可可爱爱，在潘达星球也要快快乐乐哦
4.84E+15	Sat Nov 19 17:37:19 +0800 2022	一路走好我们的国宝们
4.84E+15	Sat Nov 19 17:36:32 +0800 2022	🙏
4.84E+15	Sat Nov 19 17:36:03 +0800 2022	团团
4.84E+15	Sat Nov 19 17:29:55 +0800 2022	团团
4.84E+15	Sat Nov 19 17:27:52 +0800 2022	好难过
4.84E+15	Sat Nov 19 17:27:02 +0800 2022	去潘达星要开心幸福哈，宝贝～
4.84E+15	Sat Nov 19 17:26:46 +0800 2022	🙏🙏🙏团团一路走好
4.84E+15	Sat Nov 19 17:26:41 +0800 2022	团团
4.84E+15	Sat Nov 19 17:25:47 +0800 2022	好宝贝 来世再无病痛

4.84E+15	Sat Nov 19 17:25:45 +0800 2022	小团团，一路走好!!
4.84E+15	Sat Nov 19 17:23:41 +0800 2022	小宝贝
4.84E+15	Sat Nov 19 17:21:24 +0800 2022	团团
4.84E+15	Sat Nov 19 17:20:55 +0800 2022	团团走好天堂没有疾病的困扰
4.84E+15	Sat Nov 19 17:20:45 +0800 2022	一路走好团团
4.84E+15	Sat Nov 19 17:20:26 +0800 2022	团团宝宝，天堂没有痛苦
4.84E+15	Sat Nov 19 17:20:23 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 17:19:34 +0800 2022	团团去了那边一定要快乐
4.84E+15	Sat Nov 19 17:19:00 +0800 2022	团团圆圆没有了团团
4.84E+15	Sat Nov 19 17:18:38 +0800 2022	团团走好
4.84E+15	Sat Nov 19 17:18:30 +0800 2022	可爱的团团就这样离开我们了
4.84E+15	Sat Nov 19 17:18:29 +0800 2022	一路走好
4.84E+15	Sat Nov 19 17:17:47 +0800 2022	团团宝贝一路走好
4.84E+15	Sat Nov 19 17:14:31 +0800 2022	团团
4.84E+15	Sat Nov 19 17:14:07 +0800 2022	🙏团团一路走好
4.84E+15	Sat Nov 19 16:59:45 +0800 2022	团团一路走好，去潘达星好好的
4.84E+15	Sat Nov 19 16:53:45 +0800 2022	一路走好团团生病的时候肯定也很痛苦吧
4.84E+15	Sat Nov 19 16:51:09 +0800 2022	团团终于还是离开了我们，愿在天堂没有病痛缠身，一路走好
4.84E+15	Sat Nov 19 16:40:00 +0800 2022	团团一路走好🕯🕯🕯
4.84E+15	Sat Nov 19 16:30:13 +0800 2022	还记得那会征集名字的活动
4.84E+15	Sat Nov 19 19:32:10 +0800 2022	团团走好 在潘达星球好好的
4.84E+15	Sat Nov 19 18:33:06 +0800 2022	我的宝喵星健健康康!!!!
4.84E+15	Sat Nov 19 18:11:24 +0800 2022	想当初团团圆圆可是一对，那个时候我还很小，但它俩的名字一直记得十多年了
4.84E+15	Sat Nov 19 17:55:49 +0800 2022	团团宝贝 好喜欢你的
4.84E+15	Sat Nov 19 17:45:15 +0800 2022	团团宝贝要在潘达星球开心哦 ~ 我们大家都会想你的
4.84E+15	Sat Nov 19 17:36:59 +0800 2022	哦天呐，我昨天晚上还梦到熊猫了
4.84E+15	Sat Nov 19 17:28:16 +0800 2022	再见团团。。
4.84E+15	Sat Nov 19 17:23:20 +0800 2022	一路走好
4.84E+15	Sat Nov 19 17:20:54 +0800 2022	天堂没有病痛🙏🙏🙏🙏🙏
4.84E+15	Sat Nov 19 16:52:27 +0800 2022	小宝贝一路走好
4.84E+15	Sat Nov 19 18:15:03 +0800 2022	panda 星球也要快乐
4.84E+15	Sat Nov 19 18:13:26 +0800 2022	希望去了熊猫星只有快乐没有病痛
4.84E+15	Sat Nov 19 17:54:19 +0800 2022	团团你辛苦了，好好睡一觉吧🙏
4.84E+15	Sat Nov 19 17:42:03 +0800 2022	一路走好，可爱的小宝贝
4.84E+15	Sat Nov 19 17:36:36 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 17:35:56 +0800 2022	泪目，一路走好
4.84E+15	Sat Nov 19 17:30:01 +0800 2022	以前还写过团团圆圆的作文
4.84E+15	Sat Nov 19 17:27:27 +0800 2022	团团

4.84E+15	Sat Nov 19 17:17:16 +0800 2022	我的团团呀
4.84E+15	Sat Nov 19 17:16:18 +0800 2022	太心疼团团了，愿 ta 再也没有痛苦
评论 id	发布时间	用户城市
4.84E+15	Sat Nov 19 14:12:11 +0800 2022	希望带他回四川落叶归根
4.84E+15	Sat Nov 19 14:25:52 +0800 2022	当年送它去台湾的时候，我还参加了给他俩取名的活动了呢，团团圆圆！
4.84E+15	Sat Nov 19 14:13:03 +0800 2022	希望来世能自由自在。
4.84E+15	Sat Nov 19 14:41:49 +0800 2022	把团团接回来四川吧！那怕只是骨灰…
4.84E+15	Sat Nov 19 14:11:47 +0800 2022	愿不在有痛苦
4.84E+15	Sat Nov 19 14:41:28 +0800 2022	带它回四川吧
4.84E+15	Sat Nov 19 14:16:03 +0800 2022	团团宝贝一路走好…
4.84E+15	Sat Nov 19 15:01:48 +0800 2022	团团一路走好🙏把团团接回四川吧
4.84E+15	Sat Nov 19 15:03:30 +0800 2022	让它回来吧！
4.84E+15	Sat Nov 19 14:11:46 +0800 2022	no。。。。。
4.84E+15	Sat Nov 19 15:55:22 +0800 2022	请让它落叶归根吧！
4.84E+15	Sat Nov 19 15:10:27 +0800 2022	和人一样，走了就没有痛苦了。
4.84E+15	Sat Nov 19 17:56:13 +0800 2022	一晃团团圆圆都离家 14 年了
4.84E+15	Sat Nov 19 14:18:36 +0800 2022	团团多少岁了呀
4.84E+15	Sat Nov 19 16:41:22 +0800 2022	团团必须回老家雅安宝兴
4.84E+15	Sat Nov 19 14:34:00 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 14:59:45 +0800 2022	让团团回来了吧
4.84E+15	Sat Nov 19 17:43:03 +0800 2022	回家吧，熊猫也要回来落叶归根
4.84E+15	Sat Nov 19 15:11:46 +0800 2022	可以带回来吗
4.84E+15	Sat Nov 19 16:59:51 +0800 2022	可爱的团团🙏
4.84E+15	Sat Nov 19 14:47:20 +0800 2022	团团
4.84E+15	Sat Nov 19 14:37:15 +0800 2022	希望最后可以接团团回家
4.84E+15	Sat Nov 19 18:11:41 +0800 2022	接回来吧，让他们回家吧
4.84E+15	Sat Nov 19 15:39:03 +0800 2022	把它带回四川啊
4.84E+15	Sat Nov 19 15:16:12 +0800 2022	去了胖达星自由了🙏🙏🙏
4.84E+15	Sat Nov 19 18:33:16 +0800 2022	熊有生老病死，惜别“团团”。也愿“圆圆”和“圆仔”“圆宝”一切安好！
4.84E+15	Sat Nov 19 17:29:29 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 14:47:15 +0800 2022	团团宝贝
4.84E+15	Sat Nov 19 14:33:11 +0800 2022	团团
4.84E+15	Sat Nov 19 16:51:42 +0800 2022	当年我初中，还是发短信过年的时候给团团圆圆起的名，唉
4.84E+15	Sat Nov 19 15:10:20 +0800 2022	接回来吧
4.84E+15	Sat Nov 19 15:49:03 +0800 2022	那圆圆呢？情况如何？
4.84E+15	Sat Nov 19 15:32:30 +0800 2022	團團一路走好！ 感謝你給台灣的小朋友，帶來無數的童年歡樂回憶！
4.84E+15	Sat Nov 19 18:19:43 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 17:41:54 +0800 2022	不舍

4.84E+15	Sat Nov 19 17:10:17 +0800 2022	早前还说有好转，团团安息吧！🙏🙏
4.84E+15	Sat Nov 19 17:07:36 +0800 2022	宝贝，可以回家了
4.84E+15	Sat Nov 19 17:05:05 +0800 2022	早前还说有好转，团团安息吧🙏🙏
4.84E+15	Sat Nov 19 16:54:21 +0800 2022	
4.84E+15	Sat Nov 19 16:44:40 +0800 2022	带回故乡吧 落叶归根
4.84E+15	Sat Nov 19 15:58:06 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 15:21:28 +0800 2022	、哭死
4.84E+15	Sat Nov 19 15:12:44 +0800 2022	团团宝贝一路走好
4.84E+15	Sat Nov 19 15:03:45 +0800 2022	团团..
4.84E+15	Sat Nov 19 14:56:45 +0800 2022	🌿
4.84E+15	Sat Nov 19 14:21:26 +0800 2022	omg
4.84E+15	Sat Nov 19 14:17:57 +0800 2022	安息吧
4.84E+15	Sat Nov 19 14:12:27 +0800 2022	这么突然?
4.84E+15	Sat Nov 19 17:59:42 +0800 2022	再见了！团团宝贝
4.84E+15	Sat Nov 19 16:56:34 +0800 2022	呜呜呜宝宝团你怎么……
4.84E+15	Sat Nov 19 16:18:54 +0800 2022	团团，谢谢你来过
4.84E+15	Sat Nov 19 16:12:30 +0800 2022	还记得当年给团团圆圆征名的活动……
4.84E+15	Sat Nov 19 15:44:37 +0800 2022	团团
4.84E+15	Sat Nov 19 15:31:14 +0800 2022	把它接回来吧
4.84E+15	Sat Nov 19 15:27:13 +0800 2022	哎～落叶归根，给送回来吧。
4.84E+15	Sat Nov 19 14:40:55 +0800 2022	好心疼，能不能带回四川家乡……
4.84E+15	Sat Nov 19 15:28:32 +0800 2022	团儿 回家咯
4.84E+15	Sat Nov 19 18:36:35 +0800 2022	一路走好
4.84E+15	Sat Nov 19 18:32:45 +0800 2022	让团团回四川吧
4.84E+15	Sat Nov 19 18:12:16 +0800 2022	带团团回家吧
4.84E+15	Sat Nov 19 17:59:01 +0800 2022	请归还，落叶归根
4.84E+15	Sat Nov 19 17:44:46 +0800 2022	亲爱的宝贝一路走好
4.84E+15	Sat Nov 19 17:36:48 +0800 2022	一路走好在熊猫星要快乐
4.84E+15	Sat Nov 19 17:29:19 +0800 2022	让团团回四川吧
4.84E+15	Sat Nov 19 16:52:36 +0800 2022	愿来世在无痛苦
4.84E+15	Sat Nov 19 16:27:29 +0800 2022	团团一路走好，熊猫星球没有病痛，愿来世健康平安喜乐。
4.84E+15	Sat Nov 19 16:22:58 +0800 2022	好可怜
4.84E+15	Sat Nov 19 16:19:20 +0800 2022	团团宝贝 以后没有痛苦
4.84E+15	Sat Nov 19 16:00:09 +0800 2022	带回四川
4.84E+15	Sat Nov 19 15:58:12 +0800 2022	送回来吧 落叶归根 回到家乡吧
4.84E+15	Sat Nov 19 15:56:39 +0800 2022	团团一路走好！在星球没有就没有病痛，只有数不尽的笋子～
4.84E+15	Sat Nov 19 15:48:03 +0800 2022	带团团回家吧
4.84E+15	Sat Nov 19 15:36:50 +0800 2022	希望带团团会四川
4.84E+15	Sat Nov 19 15:35:47 +0800 2022	希望把它带回家
4.84E+15	Sat Nov 19 15:35:21 +0800 2022	骨灰能带回家乡嘛

4.84E+15	Sat Nov 19 15:35:06 +0800 2022	这是最没有痛苦的安排，希望你一路走好
4.84E+15	Sat Nov 19 15:28:18 +0800 2022	🙏
4.84E+15	Sat Nov 19 15:27:16 +0800 2022	寶貝 走好
4.84E+15	Sat Nov 19 15:06:47 +0800 2022	宝贝一路走好
4.84E+15	Sat Nov 19 15:02:49 +0800 2022	哭泣
4.84E+15	Sat Nov 19 14:59:56 +0800 2022	带他回家，回四川，他想家了
4.84E+15	Sat Nov 19 14:38:19 +0800 2022	团团
4.84E+15	Sat Nov 19 14:23:28 +0800 2022	安息吧！
4.84E+15	Sat Nov 19 17:31:09 +0800 2022	唉，这个病确实也治不了。带它回四川吧，让它回到卧龙的竹林里。
4.84E+15	Sat Nov 19 17:53:29 +0800 2022	落叶归根 魂归故里
4.84E+15	Sat Nov 19 17:39:23 +0800 2022	本来以为四川的专家去了会有好转，没想到哎
4.84E+15	Sat Nov 19 15:04:51 +0800 2022	带回四川吧，落叶归根，孩子应该回家💔
4.84E+15	Sat Nov 19 15:04:49 +0800 2022	带回来吧
4.84E+15	Sat Nov 19 15:02:30 +0800 2022	带回四川吧
4.84E+15	Sat Nov 19 14:39:45 +0800 2022	团团一路走好
4.84E+15	Sat Nov 19 17:38:27 +0800 2022	好难过
4.84E+15	Sat Nov 19 15:49:24 +0800 2022	落叶归根，回来吧。独在异乡为异客~~~愿天堂没有病痛
4.84E+15	Sat Nov 19 15:45:17 +0800 2022	希望它在那边也可以好好的
4.84E+15	Sat Nov 19 15:36:33 +0800 2022	一路走好
4.84E+15	Sat Nov 19 15:33:03 +0800 2022	一路走好
4.84E+15	Sat Nov 19 15:12:53 +0800 2022	团团一路走好，愿天堂没有痛苦🙏🙏🙏
4.84E+15	Sat Nov 19 15:09:24 +0800 2022	团团
4.84E+15	Sat Nov 19 14:59:05 +0800 2022	🙏🙏🙏团团。
4.84E+15	Sat Nov 19 14:24:21 +0800 2022	团团一路走好!! 愿胖达星球没有病痛
4.84E+15	Sat Nov 19 14:13:04 +0800 2022	我可怜的🐼，团团走好
4.84E+15	Sat Nov 19 14:12:38 +0800 2022	啊
评论 id	发布时间	用户城市
4.84E+15	Sat Nov 19 14:55:12 +0800 2022	把圆圆和孩子都接回来。
4.84E+15	Sat Nov 19 14:43:34 +0800 2022	所以之前说病情转好…那个就是 回光返照 吧……
4.84E+15	Sat Nov 19 14:50:49 +0800 2022	从 06 年开始，小时候的我就知道了它们的名字，多少年了？提起熊猫，首先就会想到它们这一对！它们是带着使命和愿望去的！如今团团宝贝走了，突然好想哭！
4.84E+15	Sat Nov 19 14:58:38 +0800 2022	国宝碎了国宝离世赶紧统一吧
4.84E+15	Sat Nov 19 14:50:02 +0800 2022	團團走了，哭死我了
4.84E+15	Sat Nov 19 14:44:45 +0800 2022	那圆圆怎么办呢
4.84E+15	Sat Nov 19 15:04:41 +0800 2022	两岸关系就此终结的一个标志
4.84E+15	Sat Nov 19 14:54:28 +0800 2022	能安排它回四川吗？

4.84E+15	Sat Nov 19 15:37:14 +0800 2022	痛彻心扉啊！痛彻心扉啊！！痛彻心扉啊！！！
4.84E+15	Sat Nov 19 15:36:32 +0800 2022	这冥冥之中好像预示着什么。。。。。唉
4.84E+15	Sat Nov 19 15:05:32 +0800 2022	看不见两岸和统的希望
4.84E+15	Sat Nov 19 14:48:11 +0800 2022	任何国宝到这帮败家子手里，能有好？赶紧把圆圆接回来
4.84E+15	Sat Nov 19 20:56:11 +0800 2022	说来也是很沉重，何止是熊猫🐼，很多人也没能等到那一天就离世了。两岸还有多久能够团团圆圆呢？
4.84E+15	Sat Nov 19 14:59:02 +0800 2022	暗示团不了了？
4.84E+15	Sat Nov 19 14:44:44 +0800 2022	黄智贤星球超话
4.84E+15	Sat Nov 19 14:59:15 +0800 2022	有问题的是政客 我相信动物园的相关工作人员对团团的照料肯定是没问题的
4.84E+15	Sat Nov 19 15:10:37 +0800 2022	带回熊猫基地安葬吧
4.84E+15	Sat Nov 19 14:53:16 +0800 2022	象征和平团圆的最后纽带没了！
4.84E+15	Sat Nov 19 15:43:03 +0800 2022	美好的期盼终成遗憾
4.84E+15	Sat Nov 19 15:45:56 +0800 2022	赶紧把圆圆接回来吧！！！
4.84E+15	Sat Nov 19 15:43:58 +0800 2022	团团的离开就是某种凶险的预兆，该来的还是要来
4.84E+15	Sat Nov 19 15:31:36 +0800 2022	拿起我们的枪炮，把台湾收回来！
4.84E+15	Sat Nov 19 14:47:48 +0800 2022	这么快？
4.84E+15	Sat Nov 19 14:52:58 +0800 2022	那就把圆圆接回来吧！
4.84E+15	Sat Nov 19 15:01:59 +0800 2022	可憐
4.84E+15	Sat Nov 19 14:44:59 +0800 2022	中午还在新闻中看到它！怎么这么快就没了？
4.84E+15	Sat Nov 19 17:21:58 +0800 2022	评论区有的人，了解清楚再来开麦吧。
4.84E+15	Sat Nov 19 16:04:49 +0800 2022	所以说台湾根本就不配，表示着现在这个形式走下去。还团圆个啥。不服就干了
4.84E+15	Sat Nov 19 15:47:26 +0800 2022	还我熊猫🐼
4.84E+15	Sat Nov 19 15:43:33 +0800 2022	团团
4.84E+15	Sat Nov 19 14:59:25 +0800 2022	下次在送圆圆熊猫🐼过去
4.84E+15	Sat Nov 19 14:49:31 +0800 2022	难过
4.84E+15	Sat Nov 19 14:45:31 +0800 2022	🙏
4.84E+15	Sat Nov 19 14:42:59 +0800 2022	唉~~~
4.84E+15	Sat Nov 19 14:42:35 +0800 2022	好难过
4.84E+15	Sat Nov 19 21:16:50 +0800 2022	心痛，超難過，很多等不到統一就走了，何時到來.....
4.84E+15	Sat Nov 19 15:16:13 +0800 2022	把它老婆和两个孩子给我们带回来，娘家人知道疼
4.84E+15	Sat Nov 19 22:43:36 +0800 2022	木柵動物園照顧的很好，同胞們不必過度聯想
4.84E+15	Sat Nov 19 18:15:11 +0800 2022	太傷心了

4.84E+15	Sat Nov 19 15:42:16 +0800 2022	好难过智贤姐
4.84E+15	Sat Nov 19 15:07:20 +0800 2022	这是告示放弃和平团圆是吧?
4.84E+15	Sat Nov 19 14:59:20 +0800 2022	我勒个去，罪过罪过
4.84E+15	Sat Nov 19 14:58:19 +0800 2022	够了 开打吧
4.84E+15	Sat Nov 19 14:54:00 +0800 2022	带着团圆使命的宝贝，完成了自己的使命，期待两岸尽早团圆!
4.84E+15	Sat Nov 19 14:51:34 +0800 2022	很痛心……
4.84E+15	Sat Nov 19 14:51:08 +0800 2022	本来期待着两岸团圆，现在圆没了
4.84E+15	Sat Nov 19 14:47:42 +0800 2022	说明两岸快团圆了!
4.84E+15	Sat Nov 19 14:47:15 +0800 2022	好心痛💔
4.84E+15	Sat Nov 19 14:46:30 +0800 2022	难过
4.84E+15	Sat Nov 19 14:46:14 +0800 2022	愿早日团圆
4.84E+15	Sat Nov 19 14:46:11 +0800 2022	悲痛
4.84E+15	Sat Nov 19 14:45:48 +0800 2022	🕯️🕯️🕯️
4.84E+15	Sat Nov 19 14:45:40 +0800 2022	我们两专家不是说好转了吗？哎
4.84E+15	Sat Nov 19 14:45:17 +0800 2022	不祥之兆
4.84E+15	Sat Nov 19 14:44:15 +0800 2022	难过!
4.84E+15	Sat Nov 19 14:43:30 +0800 2022	好伤心💔
4.84E+15	Sat Nov 19 14:43:10 +0800 2022	真的吗！非常遗憾!!
4.84E+15	Sat Nov 19 14:42:25 +0800 2022	伤心
4.84E+15	Sat Nov 19 18:38:08 +0800 2022	团团可怜的团团，让回到故土吧！把圆圆还回来
4.84E+15	Sat Nov 19 18:37:23 +0800 2022	唉!!
4.84E+15	Sat Nov 19 18:29:41 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 17:17:39 +0800 2022	团团不在就只剩圆圆了
4.84E+15	Sat Nov 19 15:55:29 +0800 2022	🙏🙏🙏🙏
4.84E+15	Sun Nov 20 07:53:34 +0800 2022	遗憾
4.84E+15	Sun Nov 20 01:02:19 +0800 2022	可能是巴子下毒
4.84E+15	Sat Nov 19 23:16:45 +0800 2022	该来的总会来
4.84E+15	Sat Nov 19 22:55:52 +0800 2022	希望能好好照顾她的孩子们，还有圆圆
4.84E+15	Sat Nov 19 22:44:08 +0800 2022	🙏
4.84E+15	Sat Nov 19 22:33:29 +0800 2022	今天看到这个新闻的时候 哭了
4.84E+15	Sat Nov 19 21:13:35 +0800 2022	在这说些什么感谢不舍有什么用 要被解剖要被剥皮有人阻止吗？！惺惺作态有什么用？！把我们的熊猫还给我们!!!!!!!!!!!!!!!!!
4.84E+15	Sat Nov 19 21:10:37 +0800 2022	不能点赞了，又被限制了?
4.84E+15	Sat Nov 19 20:54:48 +0800 2022	再见了，团团
4.84E+15	Sat Nov 19 20:35:46 +0800 2022	统一才是真团圆 tyty
4.84E+15	Sat Nov 19 20:04:57 +0800 2022	中华民族注定不能团圆吗
4.84E+15	Sat Nov 19 19:55:11 +0800 2022	唉好伤心
4.84E+15	Sat Nov 19 19:53:36 +0800 2022	还要做成标本?圆圆和圆仔还是回大陆的

		好.
4.84E+15	Sat Nov 19 19:20:20 +0800 2022	真的很不捨
4.84E+15	Sat Nov 19 19:18:54 +0800 2022	该走的就要走了，熊猫都留不住的地方，可想而知有毒！台湾地区生病了，要治一治
4.84E+15	Sat Nov 19 18:55:19 +0800 2022	象征和统的“团圆”已破，只有武统!
4.84E+15	Sat Nov 19 18:53:07 +0800 2022	转发微博
4.84E+15	Sat Nov 19 18:15:33 +0800 2022	心痛！😭
4.84E+15	Sat Nov 19 17:54:36 +0800 2022	希望把剩下的接回来，其他的别送了
4.84E+15	Sat Nov 19 17:50:49 +0800 2022	天呐，这么快，不是说病情乐观吗？怎么就
4.84E+15	Sat Nov 19 17:44:32 +0800 2022	好难过啊
4.84E+15	Sat Nov 19 17:44:04 +0800 2022	我就想知道是什么原因啊？…
4.84E+15	Sat Nov 19 17:43:13 +0800 2022	伤心💔
4.84E+15	Sat Nov 19 17:39:03 +0800 2022	是否预示两岸和平时代的结束?
4.84E+15	Sat Nov 19 17:21:11 +0800 2022	一方水土养一方人
4.84E+15	Sat Nov 19 17:10:29 +0800 2022	请将团团送回雅安，落叶归根!
4.84E+15	Sat Nov 19 17:09:20 +0800 2022	两岸关系也没变好
4.84E+15	Sat Nov 19 16:51:23 +0800 2022	快把团团接回来吧
4.84E+15	Sat Nov 19 16:36:18 +0800 2022	把她带回家。
4.84E+15	Sat Nov 19 16:33:29 +0800 2022	团团都熬不住死了 看来注定团不了
4.84E+15	Sat Nov 19 16:30:38 +0800 2022	很
4.84E+15	Sat Nov 19 16:29:37 +0800 2022	把圆圆接回来吧
4.84E+15	Sat Nov 19 16:20:05 +0800 2022	唉终究没等来团圆
4.84E+15	Sat Nov 19 16:17:44 +0800 2022	冥冥之中是否早已注定某些定数
4.84E+15	Sat Nov 19 16:07:01 +0800 2022	不是好兆头 故宫宝物 熊猫。。。。
4.84E+15	Sat Nov 19 15:57:58 +0800 2022	好心疼哦😭把圆圆和它们的孩子还回来🙏
4.84E+15	Sat Nov 19 15:57:30 +0800 2022	感觉好像是一个时代结束了
4.84E+15	Sat Nov 19 15:55:16 +0800 2022	啊，团团怎么就离世了呢
4.84E+15	Sat Nov 19 15:02:58 +0800 2022	请问你是台独嘛?
4.84E+15	Sat Nov 19 14:48:25 +0800 2022	再不统一不光是大熊猫，台北故宫里的珍贵文物不毁坏掉早晚也会被调包或偷运出境流失
4.84E+15	Sat Nov 19 15:20:46 +0800 2022	太难过了团团，我们永远爱你
4.84E+15	Sun Nov 20 04:19:17 +0800 2022	评论区部分评论跟民智没有开化一样
4.84E+15	Sat Nov 19 18:08:34 +0800 2022	把圆圆接回来吧
4.84E+15	Sat Nov 19 16:30:05 +0800 2022	哎,
4.84E+15	Sat Nov 19 15:19:05 +0800 2022	往生淨土,超生出苦,阿彌陀佛,阿彌陀佛
4.84E+15	Sat Nov 19 15:41:40 +0800 2022	伤了大陆人的心🙏🙏
4.84E+15	Sat Nov 19 15:32:01 +0800 2022	蔡八婆一臭万年
4.84E+15	Sat Nov 19 15:12:42 +0800 2022	何时月圆

4.84E+15	Sat Nov 19 15:09:03 +0800 2022	客死异乡
4.84E+15	Sat Nov 19 15:01:26 +0800 2022	让圆圆回来吧，别让他们嚯嚯了
4.84E+15	Sat Nov 19 14:42:16 +0800 2022	！
4.84E+15	Sat Nov 19 19:02:55 +0800 2022	国宝“团团”使命走完了也没等来……
4.84E+15	Sat Nov 19 18:58:57 +0800 2022	
4.84E+15	Sat Nov 19 18:18:16 +0800 2022	啊……
4.84E+15	Sat Nov 19 17:17:23 +0800 2022	没有团圆了
4.84E+15	Sat Nov 19 16:37:50 +0800 2022	可怜的孩子
4.84E+15	Sat Nov 19 16:29:18 +0800 2022	团团不在了，可以灭台了！支独的太多了
4.84E+15	Sat Nov 19 16:00:03 +0800 2022	终究没有活到两岸统一
4.84E+15	Sun Nov 20 09:20:11 +0800 2022	希望圆圆回家
4.84E+15	Sat Nov 19 22:29:48 +0800 2022	送别
4.84E+15	Sat Nov 19 21:54:39 +0800 2022	灭绝吧！呆湾的公务员们
4.84E+15	Sat Nov 19 19:39:32 +0800 2022	🙏🙏🙏
4.84E+15	Sat Nov 19 18:43:27 +0800 2022	它不该成为牺牲品
4.84E+15	Sat Nov 19 18:26:31 +0800 2022	台湾带衰
4.84E+15	Sat Nov 19 18:09:12 +0800 2022	说明要打仗了
4.84E+15	Sat Nov 19 17:27:34 +0800 2022	团团是不是因为耽误了救治而死的？如果是希望团团的亡魂不要放过那些杀人犯！
4.84E+15	Sat Nov 19 17:07:50 +0800 2022	无团难圆！
4.84E+15	Sat Nov 19 16:37:14 +0800 2022	就是湾湾害的，死了的湾湾才是好湾湾！
4.84E+15	Sat Nov 19 16:30:14 +0800 2022	还是把国宝给整死了
4.84E+15	Sat Nov 19 16:13:52 +0800 2022	岛娃的医疗水平就是垃圾，在外国那么多大熊猫，还是第一次几年就就死了！
4.84E+15	Sat Nov 19 16:09:43 +0800 2022	我还记得当年在学校里看新闻给大熊猫征名，最高票就是团团圆圆.
4.84E+15	Sat Nov 19 16:03:59 +0800 2022	没别的期望。能给送回来安葬吧
4.84E+15	Sat Nov 19 16:03:35 +0800 2022	团团的死，吹响了解放台湾的号角！一定要解放台湾
4.84E+15	Sat Nov 19 15:54:15 +0800 2022	台湾不配拥有我们的国宝！
4.84E+15	Sat Nov 19 15:52:59 +0800 2022	赶紧把圆圆，圆仔，圆宝母女们接回大陆，让悲剧不再上演~
4.84E+15	Sat Nov 19 15:50:34 +0800 2022	难团圆？
4.84E+15	Sat Nov 19 15:32:06 +0800 2022	很难过，团团完成了它的历史使命！
4.84E+15	Sat Nov 19 15:28:40 +0800 2022	有些人和物离开了就永远回不来了
4.84E+15	Sat Nov 19 15:15:57 +0800 2022	不会再送了
4.84E+15	Sat Nov 19 15:15:02 +0800 2022	给拜登披上熊猫皮凑数
4.84E+15	Sat Nov 19 15:05:48 +0800 2022	是啊，团团都过了一生还没统一。
4.84E+15	Sat Nov 19 15:02:01 +0800 2022	好难过
4.84E+15	Sat Nov 19 14:55:27 +0800 2022	还是死了，可怜的娃。所托非人啊，就看柯文哲的冷血言论，熊猫在蛙岛，过得定

		不如意。
4.84E+15	Sat Nov 19 14:52:57 +0800 2022	
4.84E+15	Sat Nov 19 14:52:34 +0800 2022	不幸的消息，不好的征兆!
4.84E+15	Sat Nov 19 14:52:07 +0800 2022	唉，可怜的小家伙
4.84E+15	Sat Nov 19 14:50:19 +0800 2022	曾经美好的愿望
4.84E+15	Sat Nov 19 14:50:10 +0800 2022	快让圆圆回来吧!
4.84E+15	Sat Nov 19 14:48:52 +0800 2022	没了团字只有统字了
4.84E+15	Sat Nov 19 14:46:57 +0800 2022	智贤姐，当年阿扁执政时候曾经阻挠团团和圆圆来台湾，后来马英九执政后，团团和圆圆两只大熊猫才来台北与台湾民众见面。
4.84E+15	Sat Nov 19 14:46:07 +0800 2022	哎
4.84E+15	Sat Nov 19 14:45:21 +0800 2022	气死（台湾的有些人不配我们的国宝🐼）不是智贤姐啦，只是有些人真不配
4.84E+15	Sat Nov 19 14:44:44 +0800 2022	智贤姐，还有一只熊猫圆圆和圆仔送回大陆。
4.84E+15	Sat Nov 19 14:43:52 +0800 2022	天哪
4.84E+15	Sat Nov 19 14:42:39 +0800 2022	还有一只，赶快接回来
评论 id	发布时间	用户城市
4.84E+15	Sat Nov 19 18:17:45 +0800 2022	你是怎么知道团团走了台湾兴高采烈的？你这故意让两岸互起敌意吧？
4.84E+15	Sat Nov 19 18:17:23 +0800 2022	沒錯，我剛剛在奇摩新聞裡看到一堆渾蛋 1450 在興高采烈叫囂，民進党真的渾蛋至極
4.84E+15	Sat Nov 19 19:03:55 +0800 2022	我可沒興高采烈，我很難過的
4.84E+15	Sat Nov 19 19:38:42 +0800 2022	我们这边的言论也一样，看到团团去世了一味的骂台湾，但其实根本不了解实际的情况，团团生病已经有一段时间了，而且台湾那边一直都尽心尽力的对待，不止是团团，还有圆圆和她的女儿们圆仔和圆宝，看过视频的都知道台湾动物园对它们照顾的一直都很好
4.84E+15	Sat Nov 19 18:18:02 +0800 2022	不知道别人怎么想，反正我是不能吧那群“人”看成是隔壁的湖北人河南人的，从小就觉得是他们侵占了台湾岛，是侵略者。
4.84E+15	Sat Nov 19 18:15:36 +0800 2022	那些年还对和平统一存在诸多期盼 要是现在 就该起名叫武武统统
4.84E+15	Sat Nov 19 18:49:27 +0800 2022	那只是一些人好吧　咱们这边同样也有分裂派 能代表大众吗
4.84E+15	Sat Nov 19 19:35:27 +0800 2022	台灣大多數都很難過不要亂說
4.84E+15	Sat Nov 19 18:23:59 +0800 2022	别总用我们的想法去想别人，我们总在幻想什么两岸一家亲，人家真的没那么亲

		想统一只有武力解决！[我喜欢]
4.84E+15	Sat Nov 19 19:06:41 +0800 2022	我家人還有周遭的人都很難過哦。你說的不是全部的真相
4.84E+15	Sat Nov 19 19:34:20 +0800 2022	圈养大熊猫一般都能活到 30 岁 团团 18 岁去世 相当于人类 50 岁就去世 弯弯在不做人方面一向是不做人的。
4.84E+15	Sat Nov 19 20:20:14 +0800 2022	團團走了我們都很心疼難過好不…
4.84E+15	Sat Nov 19 19:57:44 +0800 2022	熊猫不适合台湾，导弹最合适
4.84E+15	Sat Nov 19 19:45:31 +0800 2022	现在大陆的年轻人对湾湾也没好感了～就这样吧～希望我有生之年能看到这地收回来～
4.84E+15	Sat Nov 19 18:03:47 +0800 2022	原来有人想法和我一致哈哈哈，拿回那块地突破第一岛链就行了，以后愿意继续在哪儿生活就生活，不愿意的就去台湾爸爸（美国），哥哥（日本），表哥（澳洲)家吧。自 1949 年以来那片土地上出生的人也七十三岁了，他们对没多少认同同宗同源。
4.84E+15	Sat Nov 19 21:58:08 +0800 2022	據我所知大部分的人還是因為團團的離開而感到傷心難過，他確確實實在動物園陪伴我們度過很多年快樂的時光，請不要以少部分人的言論來以偏概全
4.84E+15	Sat Nov 19 20:04:57 +0800 2022	我知道我和家人都感到難過，看到微博的某些評論 更覺得心寒
4.84E+15	Sat Nov 19 19:17:23 +0800 2022	为台湾洗地的人真多
4.84E+15	Sat Nov 19 21:54:43 +0800 2022	很突然，中午出門買東西，回到宿舍打開手機就是團寶走了的消息，我還懵了一下，大陸專家來台的時候，看新聞說團團還有起來走動、吃東西，本以為牠有好轉的跡象，怎麼想得到團團就這麼離開了
4.84E+15	Sat Nov 19 20:16:44 +0800 2022	尽快让中国人打起来，是美国鹰派的重要使命
4.84E+15	Sun Nov 20 09:49:21 +0800 2022	本来就是，留岛不留人（仅指 taidu）。
4.84E+15	Sat Nov 19 18:45:27 +0800 2022	你在幻想啥呢
4.84E+15	Sun Nov 20 07:39:37 +0800 2022	你真的很无聊，熊猫不在，你说什么邪门歪道的话
4.84E+15	Sat Nov 19 20:21:26 +0800 2022	哪里都有极端言论的人别以偏概全
4.84E+15	Sat Nov 19 18:03:33 +0800 2022	没人拦着你游过去
4.84E+15	Sat Nov 19 17:56:43 +0800 2022	他们爱去哪去哪,我们更想要的是土地
4.84E+15	Sun Nov 20 02:40:42 +0800 2022	基本上 95%的台灣民眾都在悼念，但人群中一定有惡言，如果硬要去看、去截圖那些剩下的 5%我們也沒辦法.....在 FB 上基本上都看到正面的。

4.84E+15	Sat Nov 19 22:15:44 +0800 2022	台灣沒有興高采烈，網友留言都很難過、不捨
4.84E+15	Sat Nov 19 21:51:01 +0800 2022	比較很少看到興高采烈的，但我想這種少數人還是存在的，但不代表大多數人
4.84E+15	Sat Nov 19 21:33:23 +0800 2022	這麼可愛的大寶貝怎麼可能興高采烈…，說會興高采烈不知道是怎麼想的…，平時沒怎麼關注政治和新聞，但很喜歡熊貓的！上網看到時震驚難受不已，眼眶都濕了，怎麼能覺得會興高采烈呢？！畢竟他們的到來也許是很多政治因素的原因，但他們本身不是政治！只是可愛的大熊貓而已！
4.84E+15	Sat Nov 19 21:11:20 +0800 2022	團團離世我們很難過，不想再看到有人消費牠
4.84E+15	Sat Nov 19 19:36:12 +0800 2022	沒有人興高采烈好嗎別亂說
4.84E+15	Sun Nov 20 07:40:43 +0800 2022	誰興高采烈叫囂了，我是台灣人我怎麼不知道🤦‍♀️
4.84E+15	Sat Nov 19 23:01:49 +0800 2022	当面谁把团团圆圆送出去的，需要切腹自尽
4.84E+15	Sat Nov 19 19:03:30 +0800 2022	本来要的就是地，不是人
4.84E+15	Sun Nov 20 10:15:52 +0800 2022	还有，来微博的台湾同胞只会让你看到友善的那面，因为不友善的那面，他们根本不会出来，而不出来的部分，非常庞大。。。。
4.84E+15	Sun Nov 20 10:07:55 +0800 2022	你才是最壞那個人吧 藉由團團過世來製造分化獲得你自己的快樂
4.84E+15	Sat Nov 19 23:23:52 +0800 2022	太过分了，台湾还想扒皮做标本
4.84E+15	Sat Nov 19 22:02:11 +0800 2022	回复@ys 是真愛:我也很難過 T_T
4.84E+15	Sat Nov 19 20:38:23 +0800 2022	那是因为台湾的教材没有大陆概念 故意隐瞒
4.84E+15	Sat Nov 19 20:03:45 +0800 2022	认同就留下，不认同，驱离
4.84E+15	Sat Nov 19 19:03:17 +0800 2022	没错
4.84E+15	Sat Nov 19 18:57:23 +0800 2022	是真的
4.84E+15	Sat Nov 19 18:27:52 +0800 2022	回复@GoCi_w:前排看打架 快开始
4.84E+15	Sat Nov 19 20:35:44 +0800 2022	那是别人生活的地方，你凭什么觊觎
4.84E+15	Sat Nov 19 19:15:31 +0800 2022	那些沒血沒淚的大多是綠側翼。正常的台灣同胞一定很不捨，像我就很難過。
4.84E+15	Sat Nov 19 22:07:38 +0800 2022	“谁跟你一家亲”
4.84E+15	Sat Nov 19 19:27:29 +0800 2022	別胡說了，我們論壇上面大家都很難過的，甚至許多人哭了。不用這樣撕兩岸感情!

4.84E+15	Sat Nov 19 22:52:18 +0800 2022	@DominicWei：一樣米養百樣人，同胞們不必在意，政治只是一時，相信很多人會醒過來的，想想馬英九時的兩岸和平相處，期待和平統一的到來
4.84E+15	Sat Nov 19 22:48:33 +0800 2022	好多台湾 ip 感觉已经慌了，在求饶了，好像之前没有那么软骨头哦，我在外网混了三四年了，对他们的嘴脸可是一清二楚
4.84E+15	Sat Nov 19 22:45:31 +0800 2022	😂😂😂我怎麼不知道我們興高采烈叫囂 作為獸醫 真的是好難過丫
4.84E+15	Sat Nov 19 21:35:51 +0800 2022	从没在意过那些人，就算在意，也是在意从大陆过去的那些老一辈。
4.84E+15	Sat Nov 19 20:33:25 +0800 2022	叫囂的都是一群乌合之众
4.84E+15	Sat Nov 19 20:21:53 +0800 2022	有一說一，大部分台灣同胞還是很難過，網路、fb 都是各種希望團團一路好走！也許有少數 1450，但還是希望大陸同胞多正確看待喔！ 當然可能大家都對於不好的言論會特別注意，這就是 1450 要的分裂兩岸！
4.84E+15	Sat Nov 19 20:09:11 +0800 2022	诈骗岛是人类世界最下限，只能适合蛙类生存，它们只要有诈骗就够了
4.84E+15	Sat Nov 19 19:18:21 +0800 2022	留岛不留蛙
4.84E+15	Sat Nov 19 18:26:51 +0800 2022	严惩凶手！
评论 id	发布时间	用户城市
4.84E+15	Sat Nov 19 23:58:49 +0800 2022	可爱啊。。。配乐很好，有人知道这是什么曲子嘛？
4.84E+15	Sat Nov 19 21:25:35 +0800 2022	它玩的好开心。幸福的宝宝。每一个小动物都应该被善待。不仅仅是熊猫
4.84E+15	Sun Nov 20 01:34:20 +0800 2022	比美国和台湾的熊猫要幸福，遇到了温柔的人
4.84E+15	Sat Nov 19 21:11:59 +0800 2022	同熊不同命
4.84E+15	Sun Nov 20 00:58:30 +0800 2022	是被人爱着的大可爱谢谢保育员爷爷
4.84E+15	Sat Nov 19 22:11:46 +0800 2022	撒了欢了
4.84E+15	Sun Nov 20 01:12:02 +0800 2022	总是会想到永远失去团团，就好难过
4.84E+15	Sat Nov 19 21:10:12 +0800 2022	好可爱
4.84E+15	Sun Nov 20 08:39:27 +0800 2022	它一看就被照顾的很好，皮毛就很有光泽很顺滑。
4.84E+15	Sun Nov 20 08:16:53 +0800 2022	替福宝开心，有个疼爱他的爷爷
4.84E+15	Sun Nov 20 08:08:19 +0800 2022	这一地红叶很有古早韩剧的浪漫感～
4.84E+15	Sat Nov 19 20:59:11 +0800 2022	它好开心啊
4.84E+15	Sun Nov 20 10:10:38 +0800 2022	回中国后交不到朋友，太可爱了
4.84E+15	Sun Nov 20 08:21:16 +0800 2022	枫叶🍁中的熊猫，惬意又自在，笔芯宠它爱它的保育员～

4.84E+15	Sun Nov 20 08:20:21 +0800 2022	好幸福的熊猫宝宝
4.84E+15	Sun Nov 20 01:37:18 +0800 2022	它好开心啊
4.84E+15	Sun Nov 20 10:08:26 +0800 2022	仿佛一只大傻子，哈哈哈哈，可爱的
4.84E+15	Sun Nov 20 10:07:44 +0800 2022	好可爱啊!
4.84E+15	Sun Nov 20 00:42:11 +0800 2022	多美好啊宝贝。[落叶][落叶][落叶]
4.84E+15	Sun Nov 20 01:37:48 +0800 2022	还会抓起来闻
4.84E+15	Sun Nov 20 10:13:57 +0800 2022	把秋叶看成了泡菜🙏
4.84E+15	Sun Nov 20 01:01:01 +0800 2022	下辈子做国宝
4.84E+15	Sun Nov 20 10:13:22 +0800 2022	韩国爷爷满满的都是爱🙏🙏🙏
4.84E+15	Sun Nov 20 10:12:04 +0800 2022	如果能有个伴就更好了！
4.84E+15	Sun Nov 20 10:07:04 +0800 2022	用心爱着大宝宝🙏🙏🙏
4.84E+15	Sun Nov 20 10:03:42 +0800 2022	无忧无虑
4.84E+15	Sun Nov 20 10:01:26 +0800 2022	好快乐的福宝
4.84E+15	Sun Nov 20 09:59:07 +0800 2022	它还先试试有危险不
4.84E+15	Sun Nov 20 09:57:43 +0800 2022	他在吃脚
4.84E+15	Sun Nov 20 03:21:54 +0800 2022	看着像被宠大的孩子。树叶都好干净。
4.84E+15	Sun Nov 20 02:48:20 +0800 2022	哎呀～怎么这么可爱呀
4.84E+15	Sun Nov 20 01:43:31 +0800 2022	它玩得好欢乐啊
4.84E+15	Sat Nov 19 18:03:25 +0800 2022	好可爱
4.84E+15	Sun Nov 20 01:44:43 +0800 2022	wa 回复@bridgeofgod:哇！感谢
4.84E+15	Sat Nov 19 20:16:51 +0800 2022	画面真美好！福宝要一直好好的！
4.84E+15	Sun Nov 20 10:10:55 +0800 2022	一堆叶子就能玩的那么开心，我们福宝真可爱
4.84E+15	Sun Nov 20 10:27:01 +0800 2022	@比丢不是丢 这猫子开心得像树叶子里藏着个电鳗
4.84E+15	Sun Nov 20 10:13:16 +0800 2022	福宝吃泡菜不?
4.84E+15	Sun Nov 20 10:12:39 +0800 2022	幸福的福宝
4.84E+15	Sun Nov 20 10:09:35 +0800 2022	@张大豆 A 感觉得到被照顾的很好，过的很开心呢
4.84E+15	Sun Nov 20 10:06:22 +0800 2022	@就爱 ci 火锅 好可爱啊
4.84E+15	Sun Nov 20 09:53:14 +0800 2022	遇到对的人了
4.84E+15	Sun Nov 20 09:52:39 +0800 2022	满满的幸福感
4.84E+15	Sun Nov 20 09:20:53 +0800 2022	它真会玩
4.84E+15	Sun Nov 20 00:04:10 +0800 2022	@肉包豆浆茶叶蛋 哇哇哇哇哇哇
4.84E+15	Sun Nov 20 09:24:26 +0800 2022	幸福的宝子
4.84E+15	Sun Nov 20 09:24:52 +0800 2022	看着好治愈
4.84E+15	Sun Nov 20 09:06:18 +0800 2022	我好像在抖上看到这个爷爷给福宝检查jiojio，还趁机拍了拍福宝的肚子，duangduang 的，我好酸，我也想拍
4.84E+15	Sun Nov 20 08:43:01 +0800 2022	哇 真的好幸福的猫
4.84E+15	Sun Nov 20 08:40:20 +0800 2022	可爱的福宝
4.84E+15	Sun Nov 20 07:37:51 +0800 2022	它好干净啊

4.84E+15	Sun Nov 20 07:27:33 +0800 2022	她就是就在韩国也不错，过的挺好
4.84E+15	Sun Nov 20 10:36:26 +0800 2022	我想变成叶子
4.84E+15	Sun Nov 20 09:26:09 +0800 2022	哈哈哈太可爱了 谢谢保育员爷爷
4.84E+15	Sun Nov 20 07:23:02 +0800 2022	@薄荷糖饶
4.84E+15	Sun Nov 20 10:28:01 +0800 2022	好可爱的保育员爷爷
4.84E+15	Sun Nov 20 10:26:52 +0800 2022	高下立判
4.84E+15	Sun Nov 20 10:20:07 +0800 2022	但 还是拒绝熊猫外交
4.84E+15	Sun Nov 20 09:55:12 +0800 2022	韩国爷爷会说中文?
4.84E+15	Sun Nov 20 09:54:03 +0800 2022	他好快乐~
4.84E+15	Sun Nov 20 09:34:32 +0800 2022	熊猫开心日常
4.84E+15	Sun Nov 20 08:57:37 +0800 2022	可可爱爱的人可可爱爱的福宝
4.84E+15	Sun Nov 20 08:30:19 +0800 2022	送到各个国家的🐼都被当成宝，只有自己省份的不拿他当回事
4.84E+15	Sun Nov 20 06:51:44 +0800 2022	它好幸福
4.84E+15	Sun Nov 20 04:57:44 +0800 2022	它好开心啊
4.84E+15	Sun Nov 20 04:13:04 +0800 2022	保育员“爷爷”？年纪有那么大吗看着就一中年大叔的样子啊
4.84E+15	Sun Nov 20 09:12:55 +0800 2022	最温暖的事物！熊猫的憨态，妈妈的饭菜，王一博的微笑😁
4.84E+15	Sun Nov 20 08:54:34 +0800 2022	动物不应该关在笼子里
4.84E+15	Sun Nov 20 10:36:04 +0800 2022	幸福的福宝公主
4.84E+15	Sun Nov 20 10:35:28 +0800 2022	它好可爱，好开心啊
4.84E+15	Sun Nov 20 10:35:02 +0800 2022	它玩的好开心啊
4.84E+15	Sun Nov 20 10:33:04 +0800 2022	虽然但是，你得教成都话啊
4.84E+15	Sun Nov 20 10:32:52 +0800 2022	福宝真的太开心啦！“我永远知道你需要什么，而我所做的一切你永远都捧场“这就是最好的缘分。
4.84E+15	Sun Nov 20 10:31:10 +0800 2022	太可爱了
4.84E+15	Sun Nov 20 10:30:36 +0800 2022	它喜欢
4.84E+15	Sun Nov 20 10:20:55 +0800 2022	福宝是真的很开心。“我永远知道你需要什么，而我所做的一切你都欣然接受”这是最好的相遇。
4.84E+15	Sun Nov 20 10:20:42 +0800 2022	它好开心，像孩子一样
4.84E+15	Sun Nov 20 10:19:45 +0800 2022	🙏
4.84E+15	Sun Nov 20 10:19:43 +0800 2022	好可爱的人还有它
4.84E+15	Sun Nov 20 10:19:39 +0800 2022	这位保育员很温暖
4.84E+15	Sun Nov 20 10:16:41 +0800 2022	好可爱嘤嘤嘤
4.84E+15	Sun Nov 20 09:56:44 +0800 2022	好可爱
4.84E+15	Sun Nov 20 09:50:26 +0800 2022	挺好，只可惜美国的跟台湾的
4.84E+15	Sun Nov 20 09:47:46 +0800 2022	好幸福呀~~
4.84E+15	Sun Nov 20 09:42:51 +0800 2022	真好
4.84E+15	Sun Nov 20 09:41:48 +0800 2022	真可爱

4.84E+15	Sun Nov 20 09:38:39 +0800 2022	看到它开心 我也好开心哈哈哈哈
4.84E+15	Sun Nov 20 09:34:36 +0800 2022	这么识趣的韩国人可比大熊猫还稀有了
4.84E+15	Sun Nov 20 09:30:53 +0800 2022	好开心呀
4.84E+15	Sun Nov 20 09:26:39 +0800 2022	@荷风堂主
4.84E+15	Sun Nov 20 09:24:33 +0800 2022	遇到了对的人
4.84E+15	Sun Nov 20 09:22:54 +0800 2022	真的，这个爷爷对福宝特别好，跟亲孩子一样!
4.84E+15	Sun Nov 20 09:22:34 +0800 2022	来个枫叶🍁浴
4.84E+15	Sun Nov 20 09:21:19 +0800 2022	比如你母上大人扫了一堆落叶，你跑去打滚，后果不堪设想啊人和熊命都不同
4.84E+15	Sun Nov 20 09:17:58 +0800 2022	从不说韩语也太夸张了吧，爷爷的视频都是韩语啊
4.84E+15	Sun Nov 20 09:15:32 +0800 2022	它玩的好开心啊!
4.84E+15	Sun Nov 20 09:10:51 +0800 2022	熊生幸运
4.84E+15	Sun Nov 20 09:08:39 +0800 2022	撒欢
4.84E+15	Sun Nov 20 09:02:01 +0800 2022	福宝是一只幸福的小熊
4.84E+15	Sun Nov 20 08:59:19 +0800 2022	给那么多树叶任它玩，可见保育员有多宠，它举起树叶撒手的时候仿佛是在说我知道这些叶子原来长在树上，风一吹就这样这样掉下来了。
4.84E+15	Sun Nov 20 08:55:40 +0800 2022	这是一只幸福满满的熊
4.84E+15	Sun Nov 20 08:53:41 +0800 2022	台的那两只咋样了我想问
4.84E+15	Sun Nov 20 08:53:04 +0800 2022	他好开心啊
4.84E+15	Sun Nov 20 08:52:09 +0800 2022	洗个快乐的树叶澡，闻闻叶子里有没有家乡的味道
4.84E+15	Sun Nov 20 08:50:52 +0800 2022	@Summer 爱傻笑
4.84E+15	Sun Nov 20 08:50:16 +0800 2022	请您欣赏表演：红叶枫了
4.84E+15	Sun Nov 20 08:46:54 +0800 2022	也是有好的韩国人的
4.84E+15	Sun Nov 20 08:45:41 +0800 2022	好开心
4.84E+15	Sun Nov 20 08:45:28 +0800 2022	宠上天
4.84E+15	Sun Nov 20 08:42:42 +0800 2022	治愈系顶流：玩乐的大熊猫！肖战的笑容！羽生结弦的花滑！
4.84E+15	Sun Nov 20 08:41:30 +0800 2022	像极了自拍撒落叶的我
4.84E+15	Sun Nov 20 08:41:20 +0800 2022	福公主真是太幸福了
4.84E+15	Sun Nov 20 08:35:58 +0800 2022	可爱
4.84E+15	Sun Nov 20 08:35:46 +0800 2022	幸福的福宝
4.84E+15	Sun Nov 20 08:35:13 +0800 2022	好宠啊，完全当自己孩子宠
4.84E+15	Sun Nov 20 08:31:57 +0800 2022	@我就是阿陌
4.84E+15	Sun Nov 20 08:29:46 +0800 2022	原来还有这么活泼好动的一面
4.84E+15	Sun Nov 20 08:28:22 +0800 2022	善良
4.84E+15	Sun Nov 20 08:24:52 +0800 2022	肆意玩耍，像一个被宠爱的孩子
4.84E+15	Sun Nov 20 08:23:59 +0800 2022	谢谢

4.84E+15	Sun Nov 20 08:20:19 +0800 2022	下辈子我也要做熊猫🐼
4.84E+15	Sun Nov 20 08:14:16 +0800 2022	作为四川人，心疼每一只滚宝宝，福宝真有福气
4.84E+15	Sun Nov 20 08:13:09 +0800 2022	大熊猫就是这个世界上最最最最最最最最最可爱的生物
4.84E+15	Sun Nov 20 08:04:40 +0800 2022	看这毛发就知道养的多好，保育员真的用心在照顾。
4.84E+15	Sun Nov 20 08:04:31 +0800 2022	玩得好开心啊
4.84E+15	Sun Nov 20 07:53:22 +0800 2022	他们对待福宝，真的像对待一个人类的宝宝
4.84E+15	Sun Nov 20 07:51:32 +0800 2022	好开心呀呀呀
4.84E+15	Sun Nov 20 07:49:28 +0800 2022	它玩树叶玩这么开心，好羡慕它
4.84E+15	Sun Nov 20 07:49:01 +0800 2022	棒子国还是有好人的!
4.84E+15	Sun Nov 20 07:41:57 +0800 2022	善良的人，可爱的墩墩儿
4.84E+15	Sun Nov 20 07:29:47 +0800 2022	幸福的福宝宝(ɔˆ ³(ˆ⁀ˆc)
4.84E+15	Sun Nov 20 07:01:13 +0800 2022	好可爱啊
4.84E+15	Sun Nov 20 06:58:58 +0800 2022	好乖哦
4.84E+15	Sun Nov 20 06:58:04 +0800 2022	国宝好可爱，爷爷也好可爱
4.84E+15	Sun Nov 20 05:34:14 +0800 2022	我时刻提醒自己，它是猛兽，猛兽！但是，好可爱
4.84E+15	Sun Nov 20 05:04:56 +0800 2022	然后它真的玩儿起来了 好可爱!
4.84E+15	Sun Nov 20 03:42:45 +0800 2022	我觉得这爷爷好懂浪漫…秋天给福宝一床的红叶做玩具
4.84E+15	Sun Nov 20 08:45:41 +0800 2022	希望蔡英文去陪团团
4.84E+15	Sun Nov 20 03:02:45 +0800 2022	别在让西巴爱豆靠近熊猫
4.84E+15	Sun Nov 20 09:05:04 +0800 2022	去密码的👃 ，赠送大熊猫，就是间接虐待。
4.84E+15	Sun Nov 20 08:35:47 +0800 2022	台湾的动物园过来看看，这才是国宝的待遇，这么珍贵的宝贝不知道珍惜。
4.84E+15	Sun Nov 20 10:32:59 +0800 2022	幸福死了
4.84E+15	Sun Nov 20 09:57:11 +0800 2022	你就胡编乱造吧
4.84E+15	Sun Nov 20 09:50:06 +0800 2022	@脑袋困掉了 aa 啊啊啊啊啊啊啊啊
4.84E+15	Sun Nov 20 09:36:27 +0800 2022	@hearstzhang 他有这么多红叶子🙈
4.84E+15	Sun Nov 20 09:35:14 +0800 2022	幸运
4.84E+15	Sun Nov 20 09:15:04 +0800 2022	@王企鹅呃 我想说教中文也不一定能交到朋友 要交四川话才得
4.84E+15	Sun Nov 20 09:13:23 +0800 2022	看哭了。团团好可怜。冰冷的地板，没有植物，没有玩具
4.84E+15	Sun Nov 20 08:36:28 +0800 2022	@泡面加蛋挞
4.84E+15	Sun Nov 20 08:29:18 +0800 2022	台湾省还不如一个泡菜国爱熊猫。
4.84E+15	Sun Nov 20 08:28:35 +0800 2022	太可爱了
4.84E+15	Sun Nov 20 08:27:10 +0800 2022	好幸福

4.84E+15	Sun Nov 20 08:07:09 +0800 2022	@Deepper_throat_ 真快乐哦呜呜
4.84E+15	Sun Nov 20 07:35:50 +0800 2022	幸福的福宝！真心对待福宝的好爷爷
4.84E+15	Sun Nov 20 06:28:14 +0800 2022	福宝好开心，好幸福！保育员爷爷真好