

不眠之夜youtube评论数据分析

视频链接：<https://www.youtube.com/watch?v=U7W8QR9fsFw>
<https://www.youtube.com/watch?v=U7W8QR9fsFw>).

不眠之夜Honkai: Star Rail上传于 2024-01-26 20:00:2312,984,325 截至2024年4月6日下午一点左右评论数：8,615，赞：365,675

由于梯子问题，使用<https://zhuanlan.zhihu.com/p/591075110>
<https://zhuanlan.zhihu.com/p/591075110>的方法，但在这一天的尝试里一直握手不成功，
SSL1123ERROR

只能放弃，借助其他工具，只爬下来5000条数据，并做以下分析报告，旨在通过分析不眠之夜的评论数据，来得知不眠之夜预告片的效果如何，预告片的存活时长，玩家的评论时间分布情况，玩家对于不眠之夜视频的情感分析，并进行情感分层，分析正面情感，负面情感和中立情感玩家的关注点

进而给下次的预告片制作提供数据支撑，并对玩家的喜好与活跃情况做剖析

本文行文结构如下：

1. 评论频数相关分析
 - 1.1 评论频数时间趋势图 --以日期分组
 - 1.2 评论频数时间趋势图 --以小时分组
2. 点赞top5评论分析
3. 词频词云分析
4. 情感分析
 - 4.1 整体情感得分分析
 - 4.2 情感分层词频词云分析
5. 结论与建议 5.1 结论
- 5.2 建议

1. 评论频数相关分析

```
In [1]: 1 import pandas as pd  
2 import matplotlib.pyplot as plt
```

```
In [2]: 1 data_wn = pd.read_excel('D:/self/GS/DA/whitenight.xlsx', index_col=0)  
2 # 只保留需要的特征  
3 data = data_wn[['authorDisplayName', 'textDisplay', 'likeCount', 'publishedAt']]  
4 df = data.head(1500)
```

D:\applications\anaconda\lib\site-packages\openpyxl\styles\stylesheet.py:226: UseWarning: Workbook contains no default style, apply openpyxl's default
warn("Workbook contains no default style, apply openpyxl's default")

In [3]: 1 data.head(10)

Out[3]:

index	authorDisplayName	textDisplay	likeCount	publishedAt
1	@Suckmyfatdickngers	Dokkan is better, Mid + dokkan is better	0	2024-04-06T02:25:54Z
2	@Natsu	Me after 2.0: Firefly ;_; Me after 2.1: Fir...	2	2024-04-05T18:51:14Z
3	@ecolg5139	過完2.1之後對這首歌是真的上癮了	2	2024-04-05T16:16:01Z
4	@Zukie-	so that's why.. 😭 <a href="https://www.yout...	1	2024-04-05T15:59:10Z
5	@Zukie-	This song literally hits so DIFFRENT after 2.1 💀	3	2024-04-05T15:52:04Z
6	@sammir7275	<a href="https://www.youtube.com/watch?v=U7W8Q..."	0	2024-04-05T15:41:44Z
7	@mr_alhimik2	text pls)))	0	2024-04-05T15:22:58Z
8	@user-rk4yu8rl1m	我映像感動。 顏良、我惚。君惚？	0	2024-04-05T13:52:14Z
9	@KuniKaezushi	(spoiler 2.1) <a href...	2	2024-04-05T13:15:01Z
10	@fravalle8543	<a href="https://www.youtube.com/watch?v=U7W8Q..."	3	2024-04-05T13:05:30Z

1.1 评论频数时间趋势图 --以日期分组

In [4]:

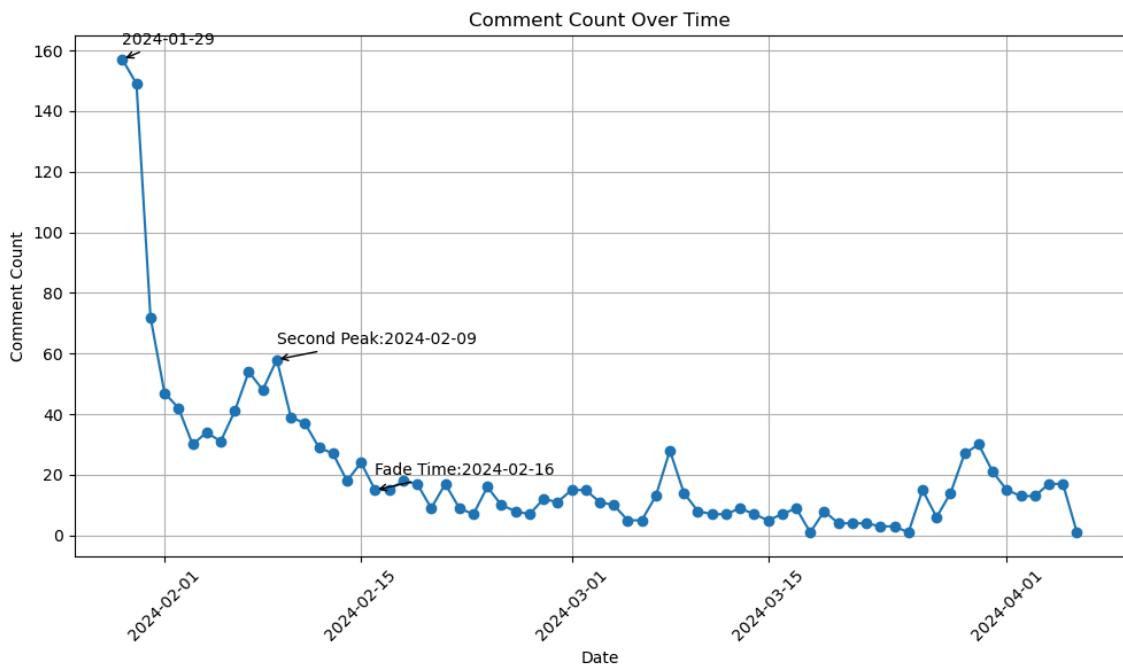
```
1 # 评论趋势图
2 # 将 publishedAt 列转换为日期时间类型
3 df['publishedAt'] = pd.to_datetime(df['publishedAt'])
4
5 # 根据 publishedAt 列排序
6 df = df.sort_values(by='publishedAt')
7
8 # 统计每天的评论数
9 daily_comments = df.groupby(df['publishedAt'].dt.date).size()
10 unique_dates = pd.unique(df['publishedAt'].dt.date).tolist()
11
12 # 绘制时间趋势图
13 plt.figure(figsize=(10, 6))
14 plt.plot(daily_comments.index, daily_comments.values, marker='o', linestyle='-' )
15 plt.title('Comment Count Over Time')
16 plt.xlabel('Date')
17 plt.ylabel('Comment Count')
18 plt.xticks(rotation=45)
19 plt.grid(True)
20 # 添加最早时间线的标签
21 earliest_date = df['publishedAt'].min().strftime('%Y-%m-%d')
22 plt.annotate('{}'.format(earliest_date), xy=(daily_comments.index[0], daily_comments.values[0]),
23               xytext=(daily_comments.index[0], daily_comments.values[0] + 5),
24               arrowprops=dict(facecolor='black', arrowstyle='->'),
25               fontsize=10, horizontalalignment='left')
26
27 # 添加第二个峰点
28 second_date = unique_dates[11]
29 plt.annotate('Second Peak: {}'.format(second_date), xy=(daily_comments.index[11],
30               xytext=(daily_comments.index[11], daily_comments.values[11]+5),
31               arrowprops=dict(facecolor='black', arrowstyle='->'),
32               fontsize=10, horizontalalignment='left')
33
34 # 添加完全褪去热度时间点
35 fade_date = unique_dates[18]
36 plt.annotate('Fade Time: {}'.format(fade_date), xy=(daily_comments.index[18],
37               xytext=(daily_comments.index[18], daily_comments.values[18]+5),
38               arrowprops=dict(facecolor='red', arrowstyle='->'),
39               fontsize=10, horizontalalignment='left')
40
41 plt.tight_layout()
42 plt.show()
```

C:\Users\CIFAR\AppData\Local\Temp\ipykernel_18016\4170389899.py:3: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df['publishedAt'] = pd.to_datetime(df['publishedAt'])
```



视频热度呈现波浪式衰减，在发布视频的1月29号时达到峰值，在三天后的2月1日到达第一个谷点，在一周之后的2月8号左右来到第二个小峰点，再一周之后的2月15日每日评论低于20，认为标志进入视频热度谷底，说明不眠之夜的热度持续时间大致为14天。得到如下结论：

- 热度最高的为前两天，然后迎来第一个谷点，
- 一周后会再有小热度出现，
- 两周后正是视频褪去大部分热度，到达视频生存极限时长

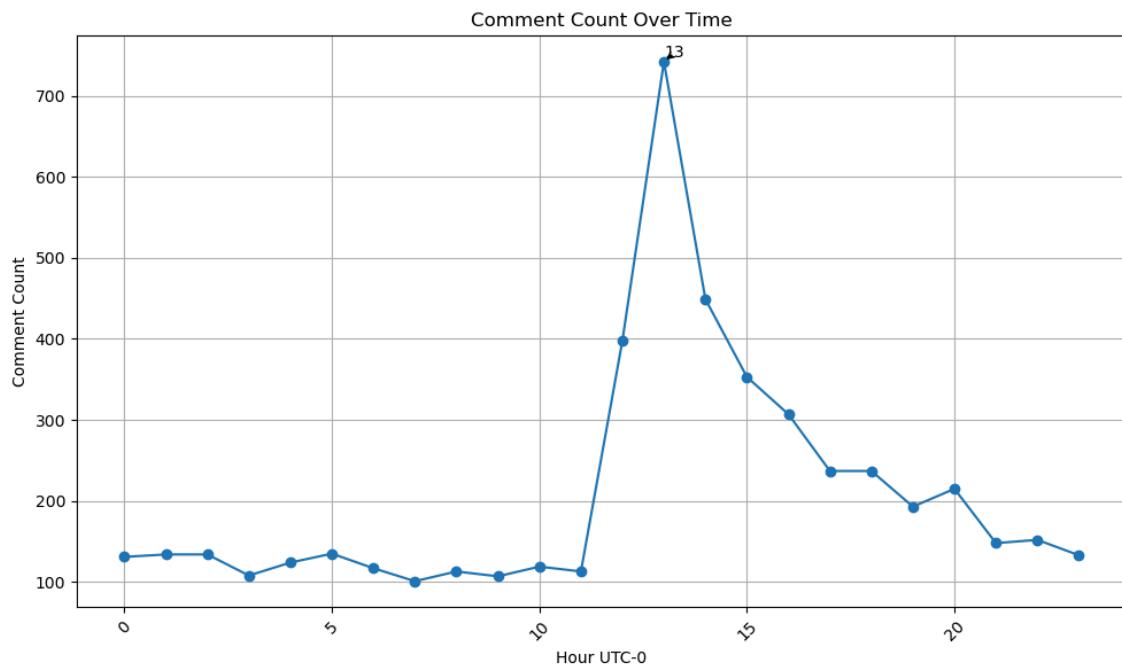
1.2 评论频数时间趋势图 --以小时分组

In [5]:

```

1 # 评论趋势图
2 # 将 publishedAt 列转换为日期时间类型
3 df = data.copy()
4 df['publishedAt'] = pd.to_datetime(df['publishedAt'])
5
6 # 根据 publishedAt 列排序
7 df = df.sort_values(by='publishedAt')
8
9 # 统计每天的评论数
10 daily_comments = df.groupby(df['publishedAt'].dt.hour).size()
11 unique_dates = pd.unique(df['publishedAt'].dt.hour).tolist()
12
13 # 绘制时间趋势图
14 plt.figure(figsize=(10, 6))
15 plt.plot(daily_comments.index, daily_comments.values, marker='o', linestyle='-' )
16 plt.title('Comment Count Over Time')
17 plt.xlabel('Hour UTC-0')
18 plt.ylabel('Comment Count')
19 plt.xticks(rotation=45)
20 plt.grid(True)
21 # 添加最早时间线的标签
22 plt.annotate('13', xy=(daily_comments.index[13], daily_comments.values[13]),
23               xytext=(daily_comments.index[13], daily_comments.values[13] + 5),
24               arrowprops=dict(facecolor='black', arrowstyle='->'),
25               fontsize=10, horizontalalignment='left')
26
27
28 plt.tight_layout()
29 plt.show()

```



从小时评论的角度出发，可以发现在UTC+0的时区下，12时开始评论显著更多，在13时时达到峰值，后续逐渐衰减，换算到中国UTC+8时区，日本UTC+9两个时区，峰值时间为21时和22时，可见这个时间段，玩家空闲的时间较多，更乐于浏览相关媒体资讯，并留下评论。

而在UTC+0的0-11时的评论则比较平稳，属于较少的时间段，这个时间换算到中国UTC+8时区，日本UTC+9两个时区【两个主要玩家来源】，为早上8-19时，早上9时-20时，为工作时间/上班时间段，可认为是由此原因导致，视频评论的热度较少。

2. 点赞数最高的前五评论分析

In [6]:

```
1 # 找到具有最高 likeCount 的前五个数据
2 top_five = df.nlargest(5, 'likeCount')
3 # print(top_five[['textDisplay','likeCount']])
4 # print(top_five.values)
```



点赞数最高的评论:

'Stella in her Trailer: Fighting for her life in a Nightmare Fever Dream Land.

Caelus in his Trailer: Cha Cha Real Smooth.'

为调侃男女主角在同一个版本的预告片却境遇完全不同的情况，女主在战斗，而男主却在快乐恰恰。

侧面说明，最受欢迎的还是调侃类话题，而且预告片的制作者所埋的小心思，完全能够被玩家所发现，并变成话题传播开来。

点赞数二、三的评论:

1.6: pure fiction. 2.0: peak fiction.

Before 2.0: 😊 After 2.0 trailblaze mission: 💀💀

都是在说剧情相关的，1.6版本前的剧情是欢快的，而2.0之后就开始刀人了。

侧面说明，在星铁游戏中，玩家对于剧情的反应是非常敏感的，说明了游戏剧情在玩家心理的重要性。

点赞数四、五的评论

m so glad hsr has such a creative team

m not saying they should pull a KDA and regularly release banger MVs and become a semi music label but they definitely should absolutely do that

都是在赞叹官方的创造力，完全说明了什么叫做官方下场做MAD的含金量，一个游戏公司却同时在技能树上点满了音乐和视频。

不仅仅侧面能够说明，很多面都能说明，这次不眠之夜的mv的出圈，音乐质量的不可置疑，还请来了游戏音乐专业户张杰，不眠之夜的策划显然是非常的成功的，也获得了玩家多方面的认可。

3. 词频与词云分析

In [7]:

```
1 import pandas as pd
2 from wordcloud import WordCloud
3 import jieba
4 import matplotlib.pyplot as plt
```

In [8]:

```
1 # 定义要排除的连词列表
2 stopwords = [
3     'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your',
4     'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'her',
5     'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'thems',
6     'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is',
7     'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',
8     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as',
9     'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'i',
10    'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up',
11    'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'ond',
12    'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few',
13    'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same',
14    'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'r',
15    'This', 'THIS', 'br', 'href', 'com', 'https', 'www', 'youtube', '...', 'IS',
16    '&', '&#', 'U7W8QR9fsFw', 'watch'
17 ]
18
19 def freqAwordcloud(df):
20     # 将 textDisplay 中的文本进行分词，并过滤掉停用词
21     text = ' '.join(df['textDisplay'])
22     words = jieba.lcut(text)
23
24     # 统计词频
25     word_freq = {}
26     for word in words:
27         if len(word) > 1 and word not in stopwords: # 只统计长度大于1且不在停用词中的词
28             word_freq[word] = word_freq.get(word, 0) + 1
29
30     # 转换为 DataFrame
31     word_freq_df = pd.DataFrame(list(word_freq.items()), columns=['词语', '词频'])
32
33     # 按词频降序排列
34     word_freq_df = word_freq_df.sort_values(by='词频', ascending=False)
35
36     # 显示结果
37     print(word_freq_df.head(20))
38
39
40     # 创建词云对象
41     wordcloud = WordCloud(width=800, height=400, background_color='white').gen
42
43
44     # 显示词云
45     plt.figure(figsize=(10, 5))
46     plt.imshow(wordcloud, interpolation='bilinear')
47     plt.axis('off')
48     plt.show()
49 freqAwordcloud(df)
```

Building prefix dict from the default dictionary ...

Loading model from cache C:\Users\CIFAR\AppData\Local\Temp\jieba.cache

Loading model cost 0.421 seconds.

Prefix dict has been built successfully.

		词语	词频
16		39	660
11		song	212
280		like	165
311		love	150
108		Caelus	137
334		HSR	129
85		trailer	122
65		Genshin	119
76		good	110
140		Firefly	108
55		genshin	107
24		game	106
57		never	102
261		firefly	102
309		2.0	95
107	Aventurine		95
78		music	94
133		one	89
514		video	84
56		could	83



从上述的词频统计中可知，其中song, like, love, Caelus, trailer, good, Firefly, Aventurine, 2.0, music, video的含义较为明显，大致可以按顺序分为以下几类：

- song, trailer, music, video在表达对于视频和音乐的看法
 - like, love, good在表达玩家对于视频的态度正向态度居多
 - Caelus, Firefly, Aventurine分别为男主，流萤，砂金的角色名
 - 2.0为版本信息，说明玩家对于新版本的关注程度

可知，玩家观众对于不眠之夜的关注点主要有三个角度，
一是媒体的感知，例如音乐，视频；
二是对于这次预告片的态度：喜欢；
三是视频出现的角色，关注性降序排序为男主穹，流萤，砂金。

4. 情感分析

具体做法为，将每一评论输入TextBlob进行判断情感分数，生成sentiment_score

4.1 整体情感得分分析

In [9]:

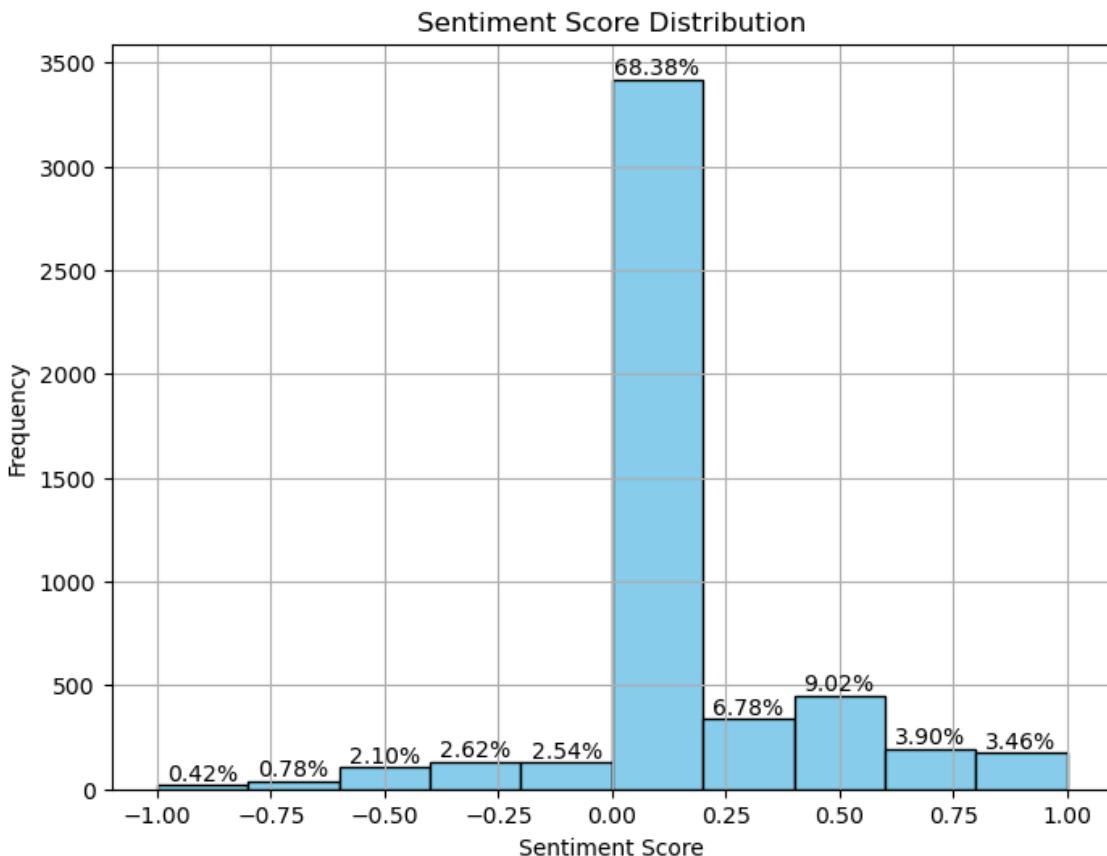
```
1 from textblob import TextBlob
2
3 def analyze_sentiment(text):
4     # 创建 TextBlob 对象
5     blob = TextBlob(text)
6
7     # 获取情感分数
8     sentiment_score = blob.sentiment.polarity
9
10    # 返回情感分数
11    return sentiment_score
12
13 # 对 textDisplay 列应用情感分析函数，并生成 sentiment_score 列
14 df['sentiment_score'] = df['textDisplay'].apply(analyze_sentiment)
```

对情感得分进行描述统计，查看其分布情况

In [10]:

```
1 import matplotlib.pyplot as plt
2 print(df['sentiment_score'].describe())
3
4 # 画出分布图
5 plt.figure(figsize=(8, 6))
6 hist, bins, _ = plt.hist(df['sentiment_score'], bins=10, color='skyblue', edgecolor='black')
7 plt.title('Sentiment Score Distribution') # 设置标题
8 plt.xlabel('Sentiment Score') # 设置 x 轴标签
9 plt.ylabel('Frequency') # 设置 y 轴标签
10 plt.grid(True) # 添加网格线
11
12 # 计算每个箱子的占比
13 bin_width = bins[1] - bins[0]
14 total_samples = len(df['sentiment_score'])
15 for i in range(len(hist)):
16     bin_center = bins[i] + bin_width / 2
17     percentage = hist[i] / total_samples * 100
18     plt.text(bin_center, hist[i], f'{percentage:.2f}%', ha='center', va='bottom')
19
20 plt.show()
```

```
count      5000.000000
mean       0.105445
std        0.289332
min       -1.000000
25%        0.000000
50%        0.000000
75%        0.200000
max        1.000000
Name: sentiment_score, dtype: float64
```



从分布图，可以大致看出，整体对于不眠之夜视频的情感为正向情感。

在正面情绪的评论中，情绪向的词频：love, good, like, amazing, cool占据多数，可以说明，玩家对于不眠之夜视频的态度主要表现是惊喜。

在其余词汇中，Caelus和music出现较为频繁，分别为男主和此次媒体的出彩，说明在正面态度的玩家群体中，不眠之夜视频的两个主要的话题为主角与音乐媒体，结合点赞top评论可以联想到，是主角的欢乐恰恰和高质量的音乐媒体的功劳。

```
In [12]: 1 print(' ##### negative_text word frequency and word cloud #####')
2 freqAwordcloud(negative_texts)
```

negative_text word frequency and word cloud		
	词语	词频
8		39 105
7	game	48
21	hard	34
261	like	25
28	Firefly	22
10	song	20
125	firefly	19
65	bad	18
190	Acheron	18
211	crying	18
212	HSR	18
162	trailer	17
531	que	16
322	Sam	14
26	Genshin	14
278	Caelus	14
223	going	14
156	Aventurine	14
241	Why	13
78	video	13



同理，在负面情绪中，可以看出，情绪向的词汇为：hard, crying, why，这些词汇表面负面情绪的评论大多都在说明一种悲伤的情绪，这与大多bad类词汇不同。

其次，结合其他词汇，特别是firefly ($22+19=41$) 的频率次数居高，再结合video, why, crying，可以理解为，玩家是在为流萤的剧情而哭泣，属于剧情方面的评论，可认为不属于对视频本身的负面情绪吐槽评论。

```
In [13]: 1 print('##### neutral_text word frequency and word cloud #####')
          2 freqAwordcloud(neutral_texts)
```

##### neutral_text word frequency and word cloud #####			
		词语	词频
39		39	243
103		song	105
148		Genshin	78
24		never	77
129		like	74
130		Caelus	69
22		genshin	66
23		could	61
29		11UJg	58
28	UCkszU2WH9gy1mb0dV		58
73		Firefly	58
67		one	56
262		que	55
269		need	55
203		HSR	55
252		de	54
316		22	54
224		2.0	54
296		GENSHIN	54
314		82	52



中性评论的词频词云表显示，玩家的话题集中在song, Caelus, Firefly。再次表面了不眠之夜视频的三个重要元素，音乐，主角与流萤。

5. 结论与建议

5.1 结论

- (1) 不眠之夜视频的最高热度持续天数为三天，之后进入第一个谷点，在视频发布一周时会进入第二个小峰点，视频两周后视频热度基本褪去，视频存活期大概为两周。
 - (2) 玩家评论的时间峰点为UTC+0时区的13时左右，对应玩家分布最广的中国和日本时区分别为晚上9点和晚上10点，视频的沉静期（评论数较少且稳定）为UTC+0时区的0-11时，对应中国和日本时区为8-19，9-20时，与工作/学习时间吻合。说明玩家活跃区间为UTC+0时区的13时左右，对应中国的20时，日本21时。

(3) 最受欢迎的还是调侃类话题，预告片的制作者所埋的小心思，能够被玩家所发现，并变成话题传播开来。其次是剧情和版本，最后是音乐媒体。

(4) 玩家观众对于不眠之夜的关注点主要有三个角度，

一是媒体的感知，例如音乐，视频；

二是对于这次预告片的态度：喜欢；

三是视频出现的角色，关注性降序排序为男主穹，流萤，砂金。

(5) 评论整体对于不眠之夜视频的情感为正向情感。若不考虑0-0.25区间，其中，负面情感占比：8.46%，而正面情感占比为：23.16%

其中，正面情感评论中对于不眠之夜视频的态度主要表现是惊喜，两个主要的话题为主角的欢乐恰恰和高质量的音乐媒体；

负面情感评论中多数为对流萤剧情的悲伤，而不来自于对视频本身的吐槽；

中立评论中发现不眠之夜视频的三个重要元素分别为音乐，主角与流萤

5.2 建议

(1) 视频的热度持续时间大约为3天，如果有相关的宣传活动可以在这个区间快结束时趁热打铁进行宣发。

(2) 玩家的活跃时间为UTC+0时区的13时，若有相关视频或预热宣传可以选择在这个时间点宣发。

(3) 可以适当埋一些有趣的梗进去，玩家完全有能力发现，并形成话题，在社区形成二次传播，增加话题热度。

(4) 玩家最在意的三个元素为，媒体质量，剧情，出现角色，可以着重从这三个角度出发设计后续宣传片或物料。