Hacker News Data Analysis

May 27, 2024

1 Uncovering Insights from Hacker News Posts: A Data-Driven Analysis

Hacker News, a popular online community started by Y Combinator, has become a hub for technology and startup enthusiasts to share and discuss a wide range of topics. With a focus on user-submitted content, known as "posts," Hacker News provides a platform for individuals to share stories, ask questions, and showcase their projects. Posts that gain traction within the community can attract significant attention, drawing hundreds of thousands of visitors. In this project, the study aims to delve into a dataset of Hacker News posts to uncover insights and trends within the community. The analysis will focus on two specific types of posts: "Ask HN" and "Show HN." "Ask HN" posts allow users to pose questions to the community, seeking advice, opinions, or expertise on various subjects. On the other hand, "Show HN" posts provide an opportunity for individuals to showcase their projects, products, or interesting findings. The study seeks to answer the following questions:

1.0.1 Do "Ask HN" or "Show HN" posts receive a higher average number of comments?

1.0.2 Does the timing of a post's creation influence the number of comments it receives?

Through a data-driven approach, the study will analyze the dataset, preprocess the data, and employ statistical techniques to uncover patterns and insights. By understanding the engagement dynamics within the Hacker News community, the study aims to shed light on the types of content that resonate with users and the optimal timing for maximizing visibility and interaction.

1.0.3 Import data and preliminary inspection

The dataset is loaded from a CSV file named 'hacker_news.csv' using Python's built-in csv module. The hn variable is created as a list of lists, where each inner list represents a row from the CSV file. The first five rows of the dataset are printed for preliminary inspection.

[40]:
```python
import csv

with open('hacker_news.csv', 'r') as f:
    reader = csv.reader(f)
    hn = list(reader)

print(hn[:5])
```

```
[['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_at'],
['12224879', 'Interactive Dynamic Video',
'http://www.interactivedynamicvideo.com/', '386', '52', 'ne0phyte', '8/4/2016
11:52'], ['10975351', 'How to Use Open Source and Shut the Fuck Up at the Same
Time', 'http://hueniverse.com/2016/01/26/how-to-use-open-source-and-shut-the-
fuck-up-at-the-same-time/', '39', '10', 'josep2', '1/26/2016 19:30'],
['11964716', "Florida DJs May Face Felony for April Fools' Water Joke",
'http://www.thewire.com/entertainment/2013/04/florida-djs-april-fools-water-
joke/63798/', '2', '1', 'vezycash', '6/23/2016 22:20'], ['11919867', 'Technology
ventures: From Idea to Enterprise', 'https://www.amazon.com/Technology-Ventures-
Enterprise-Thomas-Byers/dp/0073523429', '3', '1', 'hswarna', '6/17/2016 0:01']]
```

1.0.4 Identify headers and inspect first five rows

The first row of the dataset is extracted and stored in the headers variable, representing the column names. The hn list is then updated by removing the header row. The first five rows of the updated hn list are printed to inspect the actual data.

[2]:
```
headers = hn[:1]
headers
```

[2]:
```
[['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_at']]
```

[41]:
```
hn = hn[1:]

# Display the first five rows of hn
hn[:5]
```

[41]:
```
[['12224879',
  'Interactive Dynamic Video',
  'http://www.interactivedynamicvideo.com/',
  '386',
  '52',
  'ne0phyte',
  '8/4/2016 11:52'],
 ['10975351',
  'How to Use Open Source and Shut the Fuck Up at the Same Time',
  'http://hueniverse.com/2016/01/26/how-to-use-open-source-and-shut-the-fuck-up-
at-the-same-time/',
  '39',
  '10',
  'josep2',
  '1/26/2016 19:30'],
 ['11964716',
  "Florida DJs May Face Felony for April Fools' Water Joke",
  'http://www.thewire.com/entertainment/2013/04/florida-djs-april-fools-water-
joke/63798/',
  '2',
  '1',
  'vezycash',
  '6/23/2016 22:20'],
 ['11919867',
  'Technology ventures: From Idea to Enterprise',
  'https://www.amazon.com/Technology-Ventures-Enterprise-Thomas-
Byers/dp/0073523429',
  '3',
  '1',
  'hswarna',
  '6/17/2016 0:01'],
 ['10301696',
  'Note by Note: The Making of Steinway L1037 (2007)',
  'http://www.nytimes.com/2007/11/07/movies/07stein.html?_r=0',
  '8',
  '2',
  'walterbell',
  '9/30/2015 4:12']]
```

1.0.5 Categorizing Posts

The posts are categorized into three lists based on their titles:

1. ***ask_posts:*** Posts with titles starting with "ask hn" (case-insensitive)
2. ***show_posts:*** Posts with titles starting with "show hn" (case-insensitive)
3. ***other_posts:*** Posts that do not fall into either the "ask_posts" or "show_posts" categories

The number of posts in each category is printed, along with the first five titles from the ask_posts and show_posts lists.

[4]:
```python
ask_posts = []
show_posts = []
other_posts = []

for row in hn:
    title = row[1]
    if title.lower().startswith('ask hn'):
        ask_posts.append(title)
    elif title.lower().startswith('show hn'):
        show_posts.append(title)
    else:
        other_posts.append(title)

print('Ask posts:', len(ask_posts))
print('Show posts:', len(show_posts))
print('Other posts:', len(other_posts))
```

```
Ask posts: 1744
Show posts: 1162
```

```
Other posts: 17194
```

[5]:
```
ask_posts[:5]
```

[5]:
```
['Ask HN: How to improve my personal website?',
 'Ask HN: Am I the only one outraged by Twitter shutting down share counts?',
 'Ask HN: Aby recent changes to CSS that broke mobile?',
 'Ask HN: Looking for Employee #3 How do I do it?',
 'Ask HN: Someone offered to buy my browser extension from me. What now?']
```

[6]:
```
show_posts[:5]
```

[6]:
```
['Show HN: Wio Link  ESP8266 Based Web of Things Hardware Development Platform',
 'Show HN: Something pointless I made',
 'Show HN: Shanhu.io, a programming playground powered by e8vm',
 'Show HN: Webscope  Easy way for web developers to communicate with Clients',
 'Show HN: GeoScreenshot  Easily test Geo-IP based web pages']
```

1.0.6 Calculating Average Comments for Ask HN and Show HN Posts

To calculate the average number of comments for Ask HN and Show HN posts:

1. The total number of comments for Ask HN posts is calculated by iterating over the dataset, filtering posts with titles starting with "ask hn," and summing the comment counts.
2. The average number of comments for Ask HN posts is calculated by dividing the total comments by the number of Ask HN posts and rounding the result.
3. The process is repeated for Show HN posts.

The average number of comments for Ask HN and Show HN posts is printed.

[7]:
```
total_ask_comments = 0

for row in hn:
    title = row[1]
    if title.lower().startswith('ask hn'):
        comment = int(row[4])
        total_ask_comments += comment

avg_ask_comments = total_ask_comments / len(ask_posts)
avg_ask_comments = round(avg_ask_comments)
print("Average amount of comments for Ask HN: ", avg_ask_comments)
```

```
Average amount of comments for Ask HN:  14
```

[8]:
```
total_show_comments = 0

for row in hn:
    title = row[1]
    if title.lower().startswith('show hn'):
        comment = int(row[4])
        total_show_comments += comment

avg_show_comments = total_show_comments / len(show_posts)
avg_show_comments = round(avg_show_comments)
print("Average amount of comments for Show HN: ", avg_show_comments)
```

```
Average amount of comments for Show HN:  10
```

1.0.7 Analyzing Post Timing and Comment Engagement

To analyze the relationship between post timing and comment engagement:

1. The datetime module is imported for handling date and time operations.
2. The dataset is iterated over, and the creation timestamp and comment count of each post are extracted and stored as tuples in the result_list.
3. Two dictionaries, counts_by_hour and comments_by_hour, are created to store the count of posts and the total number of comments for each hour of the day.
4. The result_list is iterated over, and for each row:
    - The creation timestamp is converted to a datetime object.
    - The hour is extracted from the datetime object.
    - The count and total comments for the corresponding hour are updated in the respective dictionaries.

[9]:
```
import datetime as dt
```

[25]:
```
result_list = []

for row in hn:
    created_at = row[6]
    comments = int(row[4])
    result_list.append((created_at, comments))  # Store created_at and comments␣
 ↪as a tuple

counts_by_hour = {}
comments_by_hour = {}

for row in result_list:
    datetime_row = row[0]
    hours_date = dt.datetime.strptime(datetime_row, '%m/%d/%Y %H:%M')  # Create␣
 ↪datetime object
    hour_only = hours_date.strftime('%H')  # Get the hour from the datetime␣
 ↪object

    if hour_only not in counts_by_hour:
        counts_by_hour[hour_only] = 1
        comments_by_hour[hour_only] = row[1]
    else:
        counts_by_hour[hour_only] += 1
        comments_by_hour[hour_only] += row[1]
print(comments_by_hour)
```

```
{'11': 20664, '19': 27894, '22': 18684, '00': 17478, '04': 11537, '09': 15274,
'16': 30857, '18': 31587, '14': 33545, '10': 16818, '12': 25351, '13': 30562,
'20': 23414, '03': 11626, '17': 34784, '01': 12465, '23': 17582, '08': 14062,
'02': 13762, '21': 22652, '15': 35809, '06': 9253, '07': 12576, '05': 10290}
```

1.0.8 Calculate average number of comments per post for posts created during each hour of the day

The average number of comments per post for each hour is calculated by dividing the total comments by the count of posts for each hour. The results are stored in the avg_by_hour list.

[34]:
```
avg_by_hour = []

for key in counts_by_hour:
    avg_comments = comments_by_hour[key]/counts_by_hour[key]
    avg_by_hour.append([key, avg_comments])

print(avg_by_hour)
```

```
[['11', 27.118110236220474], ['19', 24.361572052401748], ['22',
21.353142857142856], ['00', 25.076040172166426], ['04', 21.891840607210625],
['09', 25.080459770114942], ['16', 23.69969278033794], ['18',
25.188995215311003], ['14', 29.14422241529105], ['10', 24.516034985422742],
['12', 27.465872156013003], ['13', 27.733212341197824], ['20',
22.27783063748811], ['03', 23.82377049180328], ['17', 25.53891336270191], ['01',
21.198979591836736], ['23', 22.59897172236504], ['08', 24.32871972318339],
['02', 26.015122873345934], ['21', 21.992233009708738], ['15',
29.01863857374392], ['06', 19.771367521367523], ['07', 24.755905511811022],
['05', 22.71523178807947]]
```

1.0.9 Sorting values

The avg_by_hour list is transformed to swap the positions of the hour and average comments, creating the swap_avg_by_hour list.

The swap_avg_by_hour list is sorted in descending order based on the average comments. The top five hours with the highest average comments are printed, with the hour formatted as ‘HH:MM’.

[33]:
```
swap_avg_by_hour = []

for row in avg_by_hour:
    first_position = row[0]
    second_position = row[1]
    swap_avg_by_hour.append([second_position, first_position])
print(swap_avg_by_hour)
```

```
[[27.118110236220474, '11'], [24.361572052401748, '19'], [21.353142857142856,
'22'], [25.076040172166426, '00'], [21.891840607210625, '04'],
[25.080459770114942, '09'], [23.69969278033794, '16'], [25.188995215311003,
'18'], [29.14422241529105, '14'], [24.516034985422742, '10'],
[27.465872156013003, '12'], [27.733212341197824, '13'], [22.27783063748811,
'20'], [23.82377049180328, '03'], [25.53891336270191, '17'],
[21.198979591836736, '01'], [22.59897172236504, '23'], [24.32871972318339,
'08'], [26.015122873345934, '02'], [21.992233009708738, '21'],
[29.01863857374392, '15'], [19.771367521367523, '06'], [24.755905511811022,
'07'], [22.71523178807947, '05']]
```

[35]:
```
sorted_swap = sorted(swap_avg_by_hour, reverse=True)
```

[38]:
```
for avg, hour in sorted_swap[:5]:
    hour_formatted = dt.datetime.strptime(hour, '%H').strftime('%H:%M')
    print("{}: {:.2f} average comments per post".format(hour_formatted, avg))
```

```
14:00: 29.14 average comments per post
15:00: 29.02 average comments per post
13:00: 27.73 average comments per post
12:00: 27.47 average comments per post
11:00: 27.12 average comments per post
```

1.1 Optimal Posting Hours for Maximizing Comment Engagement on Hacker News

The study analyzed the Hacker News dataset to determine the hours during which creating a post would yield the highest likelihood of receiving a large number of comments. By examining the average number of comments per post for each hour of the day, the study identified the following top five time slots:

- 14:00 Eastern Time (13:00 Central Time): 29.14 average comments per post
- 15:00 Eastern Time (14:00 Central Time): 29.02 average comments per post
- 13:00 Eastern Time (12:00 Central Time): 27.73 average comments per post
- 12:00 Eastern Time (11:00 Central Time): 27.47 average comments per post
- 11:00 Eastern Time (10:00 Central Time): 27.12 average comments per post

The study found that posting during typical weekday working hours in the United States, particularly around lunchtime and early afternoon, tends to generate the highest levels of engagement and discussion within the Hacker News community. This finding suggests that a significant portion of the Hacker News user base is likely located in the US and is most active on the platform during these hours.

In contrast, posts created during late night or early morning hours in US time zones consistently received fewer comments on average. The study hypothesizes that this trend can be attributed to the reduced activity of users during non-working hours and the potential differences in the user base's geographic distribution.

The implications of these findings are particularly relevant for individuals or organizations aiming to maximize the visibility and engagement of their posts on Hacker News. By strategically timing their post submissions to coincide with the identified optimal hours, they can increase the likelihood of attracting a larger number of comments and fostering meaningful discussions around their content.

However, the study acknowledges that while posting during these high-engagement hours can increase the probability of receiving more comments, it does not guarantee the quality or nature of the interactions. Other factors, such as the post's title, content, and relevance to the Hacker News community's interests, also play crucial roles in determining the level and quality of engagement.

In conclusion, the study's analysis of the Hacker News dataset reveals that creating posts during the late morning to early afternoon hours in the Central or Eastern US time zones offers the highest potential for garnering a large number of comments. This insight can be valuable for individuals and organizations looking to optimize their posting strategies and maximize the impact of their content on the Hacker News platform.

1.2 Limitations and Future Directions

While the study provides valuable insights into the engagement dynamics of Hacker News posts based on timing and post type, it is essential to acknowledge the limitations of the analysis and explore potential avenues for future research.

Limited Scope: The study focuses solely on the relationship between post timing, post type (Ask HN and Show HN), and comment engagement. It does not consider other factors that may influence engagement, such as the quality of the post content, the reputation of the author, or the specific topic of the post. Future studies could incorporate additional variables to gain a more comprehensive understanding of the factors driving engagement on Hacker News.

Temporal Limitations: The analysis is based on a specific dataset covering a limited time period. The engagement patterns and optimal posting hours identified in the study may not necessarily remain constant over time. As the Hacker News community evolves and user behavior changes, it would be valuable to conduct longitudinal studies to examine how engagement dynamics shift and adapt to new trends and preferences.

Geographic Bias: The study assumes that a significant portion of the Hacker News user base is located in the United States, based on the observed engagement patterns during US working hours. However, this assumption may not accurately reflect the global nature of the Hacker News community. Future research could explore the geographic distribution of users and investigate potential differences in engagement patterns across different time zones and regions.

Qualitative Analysis: The study primarily relies on quantitative metrics, such as the average number of comments per post, to gauge engagement. However, the quality and nature of the comments are equally important in understanding the depth and value of the interactions. Future studies could employ qualitative analysis techniques, such as sentiment analysis or content analysis, to gain insights into the quality and tenor of the discussions generated by different types of posts.

Causal Relationships: While the study identifies correlations between post timing, post type, and comment engagement, it does not establish causal relationships. Further research using experimental designs or advanced statistical techniques could help determine the causal impact of timing and post type on engagement, controlling for other potential confounding variables.

Generalizability: The findings of this study are specific to the Hacker News platform and its unique community dynamics. The engagement patterns and optimal posting strategies identified may not directly translate to other online communities or social media platforms. Future research could explore similar analyses across different platforms to compare and contrast engagement dynamics and identify platform-specific best practices.

By acknowledging these limitations and pursuing future research directions, we can deepen our understanding of the complex factors influencing engagement on Hacker News and other online communities. Such insights can help content creators, marketers, and platform managers develop more effective strategies for fostering meaningful interactions and driving valuable discussions within their respective communities.

1.3 Conclusion

This study provides valuable insights into the engagement dynamics of Hacker News posts, specifically focusing on the impact of post timing and post type (Ask HN and Show HN) on comment engagement. By analyzing a dataset of Hacker News posts, the study identifies the optimal posting hours for maximizing comment engagement, with posts created during the late morning to early afternoon hours in the Central or Eastern US time zones receiving the highest average number of comments. Additionally, the study reveals that Ask HN posts tend to receive slightly more comments on average compared to Show HN posts, suggesting that the Hacker News community is more inclined to engage with posts seeking advice, opinions, or discussions.

However, it is crucial to recognize the limitations of the study, such as the limited scope, temporal limitations, geographic bias, and the absence of qualitative analysis. Future research can build upon these findings by incorporating additional variables, conducting longitudinal studies, exploring geographic differences, employing qualitative analysis techniques, investigating causal relationships, and examining the generalizability of the findings across different online communities. Despite these limitations, the insights gained from this study can be valuable for individuals and organizations looking to optimize their posting strategies on Hacker News.

By leveraging the identified optimal posting hours and considering the engagement dynamics of different post types, content creators can increase the visibility and impact of their posts, fostering meaningful discussions and interactions within the Hacker News community. As we continue to explore the complexities of online engagement and community dynamics, studies like this contribute to our understanding of the factors that shape user behavior and participation. By refining our strategies and adapting to the evolving preferences of online communities, we can create more engaging and valuable experiences for users across various platforms.