# Alterism – Analyzing image description usage in mastodon.social

December 17, 2024

## 1 Abstract

The present report aims at analysing the use of alt text (image description) in social media posts published on the Fediverse instance mastodon.social [4], on the basis of tthe dataset mastodon.social alt text use by client app [5] by Stefan Bohacek [6], published on Kaggle under the MIT license [7].

This project was conceived and carried out by the group "Alterism" [3], formed by Cristal Rivera [1] and Tommaso Marmo [2], in the context of the Introduction to Data Science [8] course of the Artificial Intelligence and Sustainable Societies master [9].

Our research focused on understanding image description usage, analyzing it in detail using data science techniquies, and comparing it with literature concerning related topics.

## 2 Project Domain

### 2.1 Definitions

- **Alt-text**: also known as "image description", it is a critical accessibility feature, providing written descriptions for visual content (Voykinska et al., 2016). It is important for several reasons, among which the following are the three main ones.
  - Fundamental for users who are blind or visually impaired.
  - Central to the concept of Universal Design (Garofolo et al., 2022).
  - Adds context and meaning to images.
- **Clients**: the interfaces through which users access social media platforms, and their design could heavily influence the process individuals go through to publish a post, includeing the addition of alt text. While it is out of the scope of this project to inquiry the nature of any specific client, we will observe trends, recurrencies, and relevant information provided by this dataset.
- **Fediverse**: the colloquial term to describe a collection of decentralized and interoperable social network platforms that are federated using the ActivityPub standard (W3C, 2018).
- **Mastodon**: a social networking software implementing ActivityPub, the most common in the Fediverse.
- **instance**: also knowns as "node" or "server", it is one independent and self-governed endpoint inside the Fediverse.
- **Mastodon.social**: one of the most popular Fediverse instances, the one under analysis in this project.

## 2.2 Contextual information

### 2.2.1 Why accessibility?

Although often praised in the public opinion, accessibility stays a critical yet thorny and overlooked issue, especially in the context of social media (Brady & Bigham, 2014). As our team is particularly concerned with the topic, we chose to develop our project at the intersection of data science and accessibility in social media.

It is one of our chief intentions to highlight how visually disabled social media users are often being discriminated by abled users, even if unwillingly, because of the absence of image description. Our analysis will expose that the addition of image descriptions to social media posts is a virtuous but rare practice, instead of the norm.

### 2.2.2 Why the Fediverse?

We opted to base our analysis on a Fediverse instance because of a combination of three reasons. Firstly, because the current policies of the most popular social networks make it extremely difficult and costly to access and analyze data accurately (Graham, 2023). Secondly, because decentralized social networks are gaining popularity and adoption, and they pride themselves on being more ethical and human-centric (Zulli et al., 2020). Lastly, but most importantly, because Tommaso is a passionate long-term Fediverse user and instance administrator, and Cristal is very curious about the characteristics and potentiality of this network.

### 2.2.3 Why focusing on clients?

Clients are interfaces through which users access social media platforms. The diversity of clients is of paramount importance in Mastodon, given its open and interoperable nature. This allows any client to interact with the software, provided that it adheres to the application programming interface (API) standards.

As they are quintessential to the fruition of any content on Mastodon, clients significantly **influence the user experience**, including the process of adding alt text.

While investigating the specifics of client design is outside the scope of this analysis, we aim to uncover trends, recurrencies, and relevant insights starting from the information available in the dataset. Furthermore, by pinpointing the most popular clients and their features, we are going to observe how clients design and options could incentivize users to write image descriptions.

### 2.2.4 Mastodon users' information and growth

According to Dixon (2023), federated Mastodon servers had over ten million registered users collectively, as of March 2023. Mastodon, which shares similar micro-blogging features with respect to Twitter, gained roughly 500 thousand users within ten days of Elon Musk's Twitter takeover on October 27th, 2022 (Bin Zia et al., 2023).

Since Mastodon is a decentralized software, it is harder to track its users with respect to centralized networks. Nevertheless, community projects similar to the-federation.inforef++ attempt to keep track of users' statistics in real time.

# 3 Project scope and objectives

The main research question of this project is to **analyse alt text usage in Mastodon.social in relation to the clients used to publish posts on the platform**. In particular, we will develop our analysis to address the following:

1. Global alt text usage within the dataset.
2. Pinpointing the most popular clients being used.
3. Variation of alt text usage across different clients.
4. Observing possible trends or patterns linking client usage and alt text usage.
5. Assessing general inclination to add image descriptions.
6. Reflecting on how image descriptions contribute to a more inclusive and welcoming online environment.

# 4 Methods

To analyze the usage of alt text in Mastodon posts across different clients, the following methods and algorithms will be applied:

## 4.1 Exploratory Data Analysis (EDA)

EDA will be essential to explore the data and understand its structure.

### 4.1.1 Descriptive Statistics

Key statistical measures such as mean, median, standard deviation, and percentiles will be calculated to summarize and understand the distribution of posts in different clients. These metrics will help identify central tendencies and variability in the data, providing insights into patterns and anomalies.

### 4.1.2 Visualizations

- **Bar charts** and **boxplots** will be used to examine the distribution of variables and detect outliers.
- **Correlation analysis**: relationships between variables will be explored using both Pearson and Spearman correlation coefficients. For example, correlations between posts (the number of statuses posted) and alt text usage percentage (percentage of posts with image descriptions) will help assess how variables might influence each other.
- **Scatter Plots**: relationships between variables such as posts and alt text usage percentage, providing a graphical representation of trends.

These steps align with Tukey's (1977) focus on EDA as a fundamental process to understand data before applying advanced techniques.

### 4.1.3 Data Preprocessing and Feature Engineering

**Feature transformation**: certain features will be transformed using logarithmic transformations. Logarithmic scale allows the visualization of the broad range of post counts. Transformations will help improve model performance by addressing skewness or heteroscedasticity in the data (Field, 2013).

# 5 Dataset structure and description

- The source of the dataset comes from Bohacek (2024), retrieved on Kaggle: mastodon.social alt text use by client app.
- All the posts under analysis come *exclusively* from the Fediverse instance mastodon.social.

[1]:
```
# Importing libraries

# Data wrangling and analysis
import pandas as pd
from scipy.stats import pearsonr, spearmanr

# Visualization
import matplotlib.pyplot as plt
import squarify as sq
import seaborn as sns

# Appendix
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

## 5.1 Dataset import and preview

Import and print dataset assigning the variables `data` and `original_data`, so that the original dataset can be preserved, while we operate on the `data` variable.

[2]:
```
data = original_data = pd.read_csv('./fediverse-client-alt-text-data-2024-05-13.
↪csv')
data
```

[2]:
```
                     client  status_count  descriptions_all_count  \
0                       Web          8272                    1438
1                    dlvr.it          5806                       1
2       Mastodon for Android          1894                     270
3                   unknown          1428                     366
4          AboveMaidstoneBot          1339                       0
..                       …             …                       …
261               socialbot             1                       1
262   PhonocasterMusicShare             1                       0
263                openvibe             1                       0
264                 iflaapp             1                       0
265  Today's Dérive app task             1                       1

     descriptions_all_percent  descriptions_some_count  \
0                   17.383946                       7
1                    0.017224                       0
2                   14.255544                       3
```

```
3                    25.630252                          1
4                     0.000000                          0
..                          …                        …
261                 100.000000                          0
262                   0.000000                          0
263                   0.000000                          0
264                   0.000000                          0
265                 100.000000                          0

     descriptions_some_percent  descriptions_none_count  \
0                     0.084623                    6827
1                     0.000000                    5805
2                     0.158395                    1621
3                     0.070028                    1061
4                     0.000000                    1339
..                         …                       …
261                   0.000000                       0
262                   0.000000                       1
263                   0.000000                       1
264                   0.000000                       1
265                   0.000000                       0

     descriptions_none_percent
0                    82.531431
1                    99.982776
2                    85.586061
3                    74.299720
4                   100.000000
..                         …
261                   0.000000
262                 100.000000
263                 100.000000
264                 100.000000
265                   0.000000

[266 rows x 8 columns]
```

[3]:
```
data_rows, data_columns = original_data.shape # Tuple with number of rows and␣
 ↪number of columns
data_size = original_data.size # Integer that is the result of n of columns * n␣
 ↪of rows

print(f'''
Number of rows:                    {data_rows}
Number of columns:                 {data_columns}
Data size (rows×columns):          {data_size}
''')
```

```
Number of rows:                 266
Number of columns:              8
Data size (rows×columns):       2128
```

The dataset we are analyzing contains 266 rows, 8 columns and its dimension is of 2128.

### 5.2 Dataset dictionary

All columns correspond to relevant data, and to achieve this we will be going through each column, one by one, to understand its meaning and rename it in more explicatory name.

[4]:
```
data.rename(columns={
        'status_count': 'posts',
        'descriptions_all_count': 'atxt_yes',
        'descriptions_all_percent': 'atxt_yes_pct',
        'descriptions_some_count': 'atxt_some',
        'descriptions_some_percent': 'atxt_some_pct',
        'descriptions_none_count': 'atxt_no',
        'descriptions_none_percent': 'atxt_no_pct'
}, inplace=True)
```

| variable | meaning |
|---|---|
| `client` | Name of the client the posts come from. |
| `posts` | Total number of posts containing images for that client. |
| `atxt_yes` | Number of posts containing alt text in *all* images. |
| `atxt_yes_pct` | The percentage of `atxt_yes` in relation to `posts`. |
| `atxt_some` | Number of posts containing alt text in *some* images. |
| `atxt_some_pct` | The percentage of `atxt_some` in relation to `posts`. |
| `atxt_no` | Number of posts containing *no* alt text in *any* images. |
| `atxt_no_pct` | The percentage of `atxt_no` in relation to `posts`. |

For the sake of simplicity, columns' labels have been renamed using shorter variables. Furthermore, since we have now *modified* the dataset (by renaming column indexes), we are starting to use the `data` variable, preserving the `original_data` one as unchanged.

# 6 Data analysis and results

In this section, we undertake a comprehensive exploration of the dataset to uncover patterns, trends, and insights that can inform our understanding of the data and guide subsequent analysis of the mastodon data set. The primary focus is on performing Exploratory Data Analysis (EDA) to assess the dataset's analyis to identify relationships between variables, and detect potential anomalies. By applying statistical techniques and visualizations, we aim to analyze both individual variables and their interactions, uncovering meaningful clusters, trends, and dependencies. This stage also involves identifying feature importance, analyzing variability, and validating insights within the broader context of alt text usage on mastodon contributing to a deeper understanding of its role in promoting accessibility.

[5]:
```
## Utilities

# Set theme colors
blue = '#355766'
light_blue = '#8EB7CA'
lighter_blue = '#D0E7F1'

# Creating a figure numbering generator
figure_count = 0
def figure_caption_generator():
        global figure_count
        figure_count += 1
        caption = plt.text(
                1,
                1,
                f'Figure {figure_count}',
                ha = 'right',
                va = 'top',
                transform = plt.gca().transAxes,
                backgroundcolor = 'black',
                color = 'white'
        )
        return caption
```

## 6.1 Data exploration

All columns represent relevant data. To ensure clarity and usability, we will review each column individually, analyze its meaning, and assign it a more descriptive and explanatory name.

The following code provides a summary of each column in the DataFrame, including: - Data type - Number of missing values - Number of unique values - Descriptive statistics for numerical columns using pandas' `describe()`

[6]:
```
# Data type and count of missing values
summary = pd.DataFrame({
        'Data Type': original_data.dtypes,
        'Missing values?': any(original_data.isnull().sum())
```

```
})

# Descriptive statistics for numerical columns
numerical_summary = original_data.describe()
summary = summary.merge(numerical_summary.T, left_index=True, right_index=True,␣
↪how='left')
summary
```

[6]:

| | Data Type | Missing values? | count | mean | std | min | \ |
|---|---|---|---|---|---|---|---|
| client | object | False | NaN | NaN | NaN | NaN | |
| posts | int64 | False | 266.0 | 112.240602 | 643.625189 | 1.0 | |
| atxt_yes | int64 | False | 266.0 | 21.473684 | 105.282555 | 0.0 | |
| atxt_yes_pct | float64 | False | 266.0 | 36.290411 | 45.079671 | 0.0 | |
| atxt_some | int64 | False | 266.0 | 0.093985 | 0.622382 | 0.0 | |
| atxt_some_pct | float64 | False | 266.0 | 0.060461 | 0.527118 | 0.0 | |
| atxt_no | int64 | False | 266.0 | 90.672932 | 568.477040 | 0.0 | |
| atxt_no_pct | float64 | False | 266.0 | 63.649128 | 45.113843 | 0.0 | |

| | 25% | 50% | 75% | max |
|---|---|---|---|---|
| client | NaN | NaN | NaN | NaN |
| posts | 2.0 | 8.0 | 39.75 | 8272.000000 |
| atxt_yes | 0.0 | 0.0 | 4.00 | 1438.000000 |
| atxt_yes_pct | 0.0 | 0.0 | 100.00 | 100.000000 |
| atxt_some | 0.0 | 0.0 | 0.00 | 7.000000 |
| atxt_some_pct | 0.0 | 0.0 | 0.00 | 7.142857 |
| atxt_no | 0.0 | 2.0 | 20.00 | 6827.000000 |
| atxt_no_pct | 0.0 | 100.0 | 100.00 | 100.000000 |

Our dataset comprises eight columns, of which only the `client` column has the data type of an object (string), while the other columns contain numeric values. Upon summarizing the data, we can confirm that there are no missing values across any of the eight columns.

Key observations at first glance include:

- **`atxt_yes_pct`**: The mean value is 36%, indicating that, on average, 36% of clients' posts contain alt text.
- **`atxt_some_pct`**: The mean value is 0.06%, showing a negligible percentage of clients with some alt text in their posts.
- **`atxt_no_pct`**: The mean value is 64%, revealing that, on average, 64% of clients' posts lack alt text.

**Techinical Note**: Although `client` is a series of strings with no missing values, pandas assigns it the `object` data type by default, as it opts for the least strict type. For consistency and memory optimization, we will later explicitly instruct pandas to treat all `client` entries as strings.

[7]:
```
print(f'Are there any duplicated clients? {any(original_data['client'].
↪duplicated())}')
```

```
Are there any duplicated clients? False
```

There are no duplicated clients in our dataset, it means that there are 266 unique clients we will be analyzing.

## 6.2 Refine the dataset for analysis

We will be starting by sorting the data based on the amount of posts showing the most popular clients on top.

[8]:
```
# Sort clients by status_count
data = data.sort_values('posts',ascending=False)
```

[9]:
```
print(f'Client data type: {data["client"].dtype}')
```

```
Client data type: object
```

**Technical note**: Even though it is a serie of strings with no missing values, pandas returns `object` as the data type for `client`, because it assigns the less strict type possible, by default. At a later moment, we will arbitrarily instruct pandas to treat all `client`s as strings, both for consistency and for memory optimization purposes.

[10]:
```
# Consider all client names as strings (they were objects)
data['client'] = data['client'].astype('string')
print(f'Client data type: {data["client"].dtype}')
```

```
Client data type: string
```

To minimize precision errors and enhance clarity, we deem it important to round percentage values to a single decimal digit.

[11]:
```
data['atxt_yes_pct'] = data['atxt_yes_pct'].round(1)
data['atxt_some_pct'] = data['atxt_some_pct'].round(1)
data['atxt_no_pct'] = data['atxt_no_pct'].round(1)
```

### 6.2.1 Exclude posts from unknown client(s)

Based on our initial observation, we recognize that the third most used client is labelled as "unknown". Since our project stongly focuses on the impact of specific clients on the use of alt text, **we arbitrarily choose to drop (exclude) from the dataset posts that come from an undefined client**.

[12]:
```
data.drop(data[data['client'] == 'unknown'].index, inplace=True)
```

### 6.2.2 Count total clients

Since there are no duplicates, we can safely assume that the length of the dataset is equal to the number of clients.

[13]:
```
clients = len(data)
clients
```

[13]: 265

This means that we will be working with 265 clients, one less than the original 266, since we excluded the unknown ones.

### 6.2.3 Checking values

[14]:
```
# The dataset already includes percentages. Let's calculate and verify them.
posts_tot = data['posts'].sum()
atxt_yes_tot = data['atxt_yes'].sum()
atxt_some_tot = data['atxt_some'].sum()
atxt_no_tot = data['atxt_no'].sum()

atxt_yes_tot_pct = round(atxt_yes_tot / posts_tot * 100, 1)
atxt_some_tot_pct = round(atxt_some_tot / posts_tot * 100, 1)
atxt_no_tot_pct = round(atxt_no_tot / posts_tot * 100, 1)

if posts_tot == atxt_yes_tot + atxt_some_tot + atxt_no_tot:
        print('Posts count is matching!')
else:
        print('Something is wrong in the dataset. Please revise the number of␣
 ↪posts!')
```

```
Posts count is matching!
```

[15]:
```
# The dataset already includes percentages. Let's calculate and verify them.
if data['atxt_yes_pct'].equals(round(data['atxt_yes']/data['posts']*100, 1))␣
 ↪and data['atxt_no_pct'].equals(round(data['atxt_no']/data['posts']*100, 1)):
        print('The percentages in the dataset are accurate!')
```

```
The percentages in the dataset are accurate!
```

This verification process confirms that the values in the columns originally provided in the dataset are accurate and suitable for analysis without requiring the creation of additional columns. This step was essential because calculated columns might contain errors or be subject to alternative interpretations, which could impact the reliability of the analysis.

## 6.3 Global analysis

In this section, we will begin analyzing the overall use of alt text across all clients, independent of specific client information. This will provide a broader understanding of the general trends and patterns in the use of alt text within the dataset.

[16]:
```
print(f'''
Total number of posts under analysis: {posts_tot}
        of which {atxt_yes_tot} ({atxt_yes_tot_pct}%) contain an alt-text for␣
 ↪all images,
        {atxt_some_tot} ({atxt_some_tot_pct}%) for some,
        and {atxt_no_tot} ({atxt_no_tot_pct}%) have no alt-text at all.
''')
```

```
Total number of posts under analysis: 28428
        of which 5346 (18.8%) contain an alt-text for all images,
        24 (0.1%) for some,
        and 23058 (81.1%) have no alt-text at all.
```

The usage of alt text for some images only is very negligible (0.1%), therefore we are arbitrarily deciding to **drop (exclude) this column from the dataset**.

[17]:
```
# Subtracting from the total of posts the ones that only have alt text for some␣
 ↪images only
data['posts'] = data['posts'] - data['atxt_some']

# Remove columns related to some alt text
data.drop(columns=['atxt_some', 'atxt_some_pct'], inplace=True)
```

[18]:
```
plt.figure(figsize=(15, 5), dpi=300)

pie_labels = ['Alt-text for all images', 'No alt-text']
pie_colors = [lighter_blue, light_blue]

plt.pie(
        [atxt_yes_tot, atxt_no_tot],
        labels = pie_labels,
        colors = pie_colors,
        autopct = '%1.1f%%'
)

plt.title('Alt text usage in posts', fontweight='bold')
figure_caption_generator()
plt.show()
```

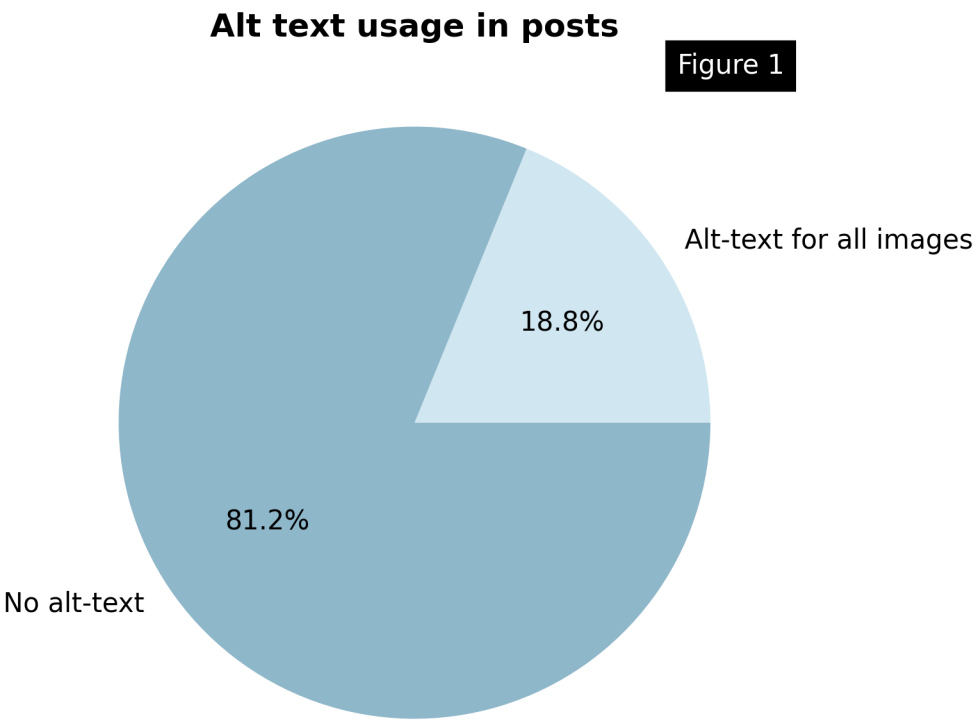



Figure 1 shows the global use of alt text (image description) in the set of posts under analysis. We notice that more than  of them (81.2%) *do not* contain any image description. Only 18.8% of them do.

This data supports the thesis sustained in Gleason et al. (2019), proving how users are not incentivized to provide descriptions to images on social media platforms.

Nevertheless, comparing the present dataset with the 1 million Tweets sample analysed by Gleason et al., we can observe that mastodon.social users tend to use alt text relevantly more than Twitter users, as only 0.1% of tweets containing pictures were found to have image descriptions.

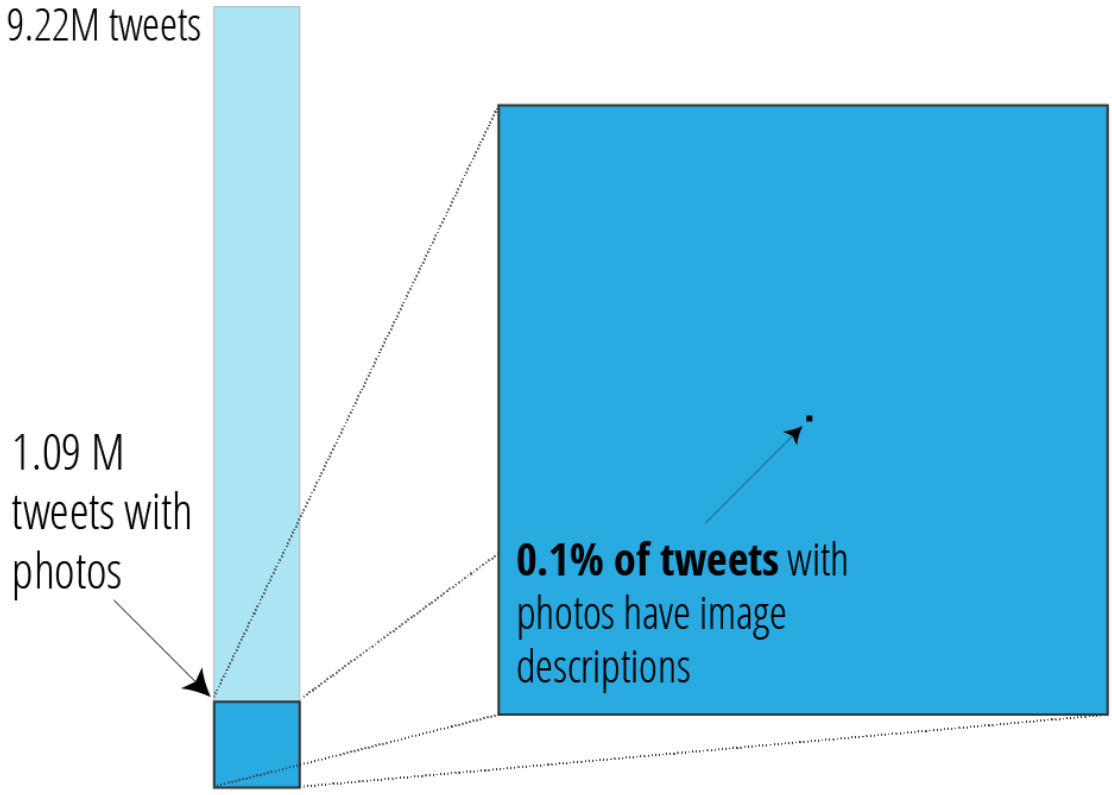



*Image 1*

It must be noted that this dataset and the tweet sample were gathered five years apart. Because of the current API usage conditions enforced by X (formerly Twitter) it is very hard to gather more recent data (Graham, 2023).

### 6.3.1 Analyzing client data

To visualize the distribution of posts across clients, we will plot the data to review potential outliers or skewed distributions. This will help us understand how the total number of posts is spread across each client and identify any patterns or irregularities in the data.

[19]:
```
plt.figure(figsize=(10, 5), dpi=300)
sns.boxenplot(
        x = data['posts'],
        color = lighter_blue
)
plt.xscale('log')
plt.title('Distribution of Posts Across Clients (Log Scale)', fontsize=14,␣
 ↪fontweight='bold')
plt.xlabel('Log of Posts Across Clients')
plt.yticks([])
plt.tight_layout()
figure_caption_generator()
plt.show()
```

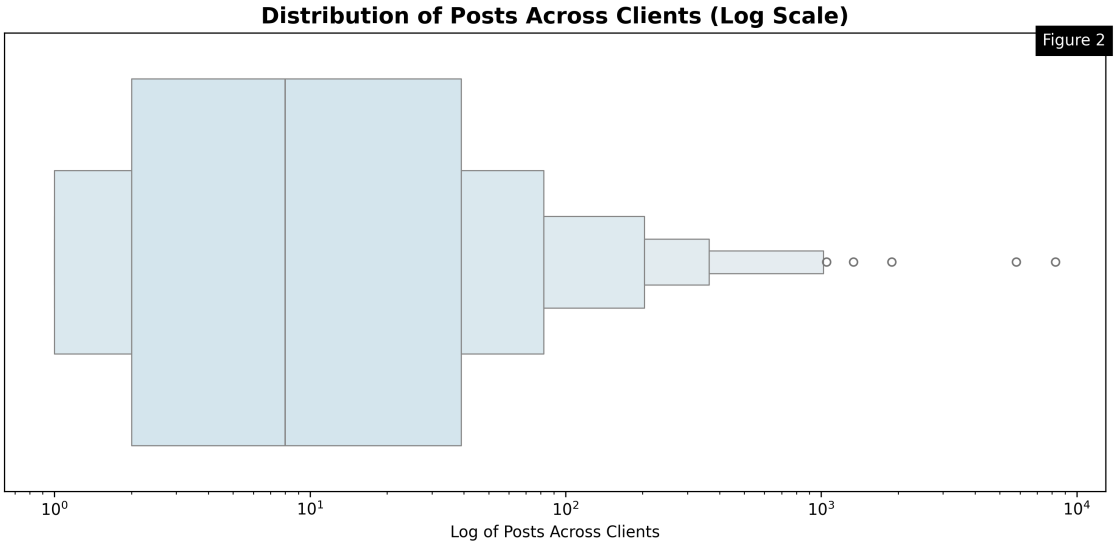



Figure 2 shows that: the majority of clients have a relatively low post count (within the first tapering layers). A few clients have exceptionally high post counts, as indicated by outliers. The distribution is **skewed**, meaning a small number of clients contribute disproportionately to the total post count. Furthermore, we can observe **high variablity**, since the logarithmic scale and tapering layers suggest substantial variability in the dataset, with a small number of clients driving much higher activity. The long right tail, highlights that a small number of clients have significantly higher post counts (as shown by the outliers).

[20]:
```
data['posts'].describe()
```

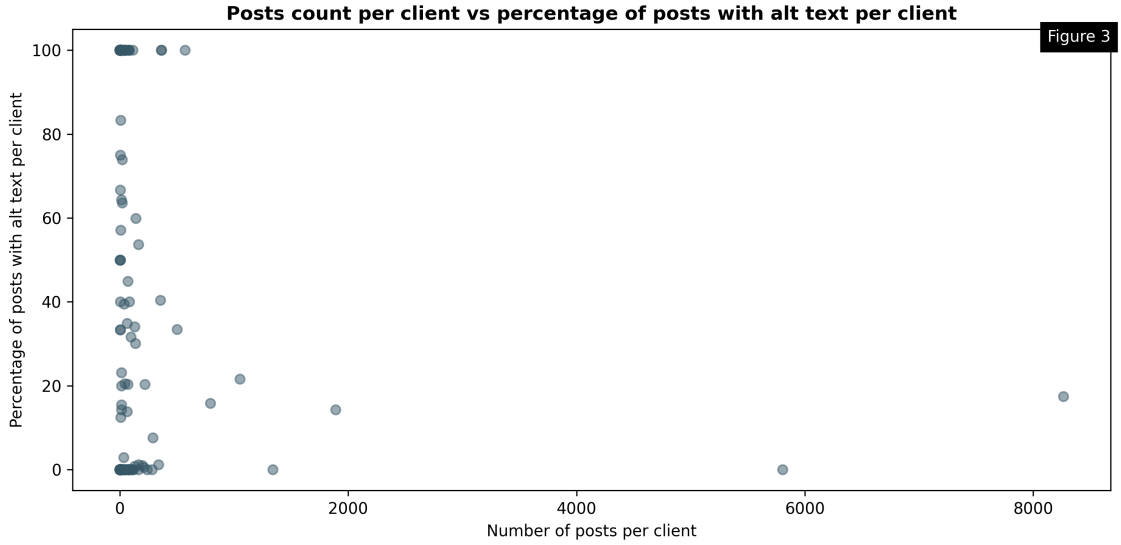
[20]:
```
count     265.000000
mean      107.184906
std       639.324886
min         1.000000
25%         2.000000
50%         8.000000
75%        39.000000
max      8265.000000
Name: posts, dtype: float64
```

Based on figure 2 and the information about the posts, we can observe that the median value is 8 posts per client, signifying that half of the clients published less than nine posts.

Additionally, there is a significant variability in the number of posts across clients, with the range varying from a minimum of 1 post to a maximum of 8,265 posts.

We will be creating a scatter plot to review the correlation between the post counts and the alt text usage. As we saw previously in the boxplot, there is a high variability of the number of posts. However, we can to review if there is any relationship beween those two variables to have a better understanding on how to proceed with the analysis.

[21]:
```
plt.figure(figsize=(10, 5), dpi=300)
plt.scatter(
        x = 'posts',
        y = 'atxt_yes_pct',
        data = data,
        color = '#365766',
        alpha = 0.5
)
plt.title('Posts count per client vs percentage of posts with alt text per client', fontweight='bold')
plt.xlabel('Number of posts per client')
plt.ylabel('Percentage of posts with alt text per client')
plt.tight_layout()
figure_caption_generator()
plt.show()
```




In figure 3 it appears that both variables do not follow a linear relationship. Most of the data points are concentrated in a low range of number of posts and spread along the percentage of posts with all alt text across clients. However, there are other data points that are more separated from the others, which indicates that the clients with a higher number of posts but a lower percentage of posts with alt text.

### 6.3.2 Correlation coefficients

To prove the absence of a correlation numerically, we will be calculating the Pearson and the Spearman correlation coefficients.

[22]:
```
# Calculate correlation coefficients
pcc, pearson_pct = pearsonr(data['posts'], data['atxt_yes_pct'])
```

```
scc, spearman_pct = spearmanr(data['posts'], data['atxt_yes_pct'])

print(f'Pearson correlation coefficient: {round(pcc,2)}')
print(f'Spearman correlation coefficient: {round(scc,2)}')
```

```
Pearson correlation coefficient: -0.07
Spearman correlation coefficient: 0.02
```

The purpose of Pearson Correlation measures the strength and direction of a linear relationship between two variables, assuming the data follows a linear pattern (Pearson, 1895). As the value of -0.07 is very close to zero, we can assess that there is no linear correlation between the number of posts and the percentage of posts with alt text.

In addition, we calculated the Spearman Correlation, used to measure monotonic relationships rather than just linear relationships (Spearman, 1904). A monotonic relationship implies that the other tends to increase or decrease as one variable increases. A Spearman correlation coefficient of 0.02 indicates an extremely weak positive monotonic relationship between post count and percentage of posts with all alt text.

We calculated both coefficients to obtain a comprehensive view of the relationship between the variables. Since both correlations are close to zero we can confirm there is **no strong relationship** between the number of posts and the alt text usage percentage.

These findings indicate that the frequency of posting does not significantly impact how likely a client is to use alt text. This insight may lead to exploring other factors, like client type, to understand alt text usage better.

A weak correlation can justify focusing on top clients (those with higher post counts = popular clients) rather than clients with very few posts.

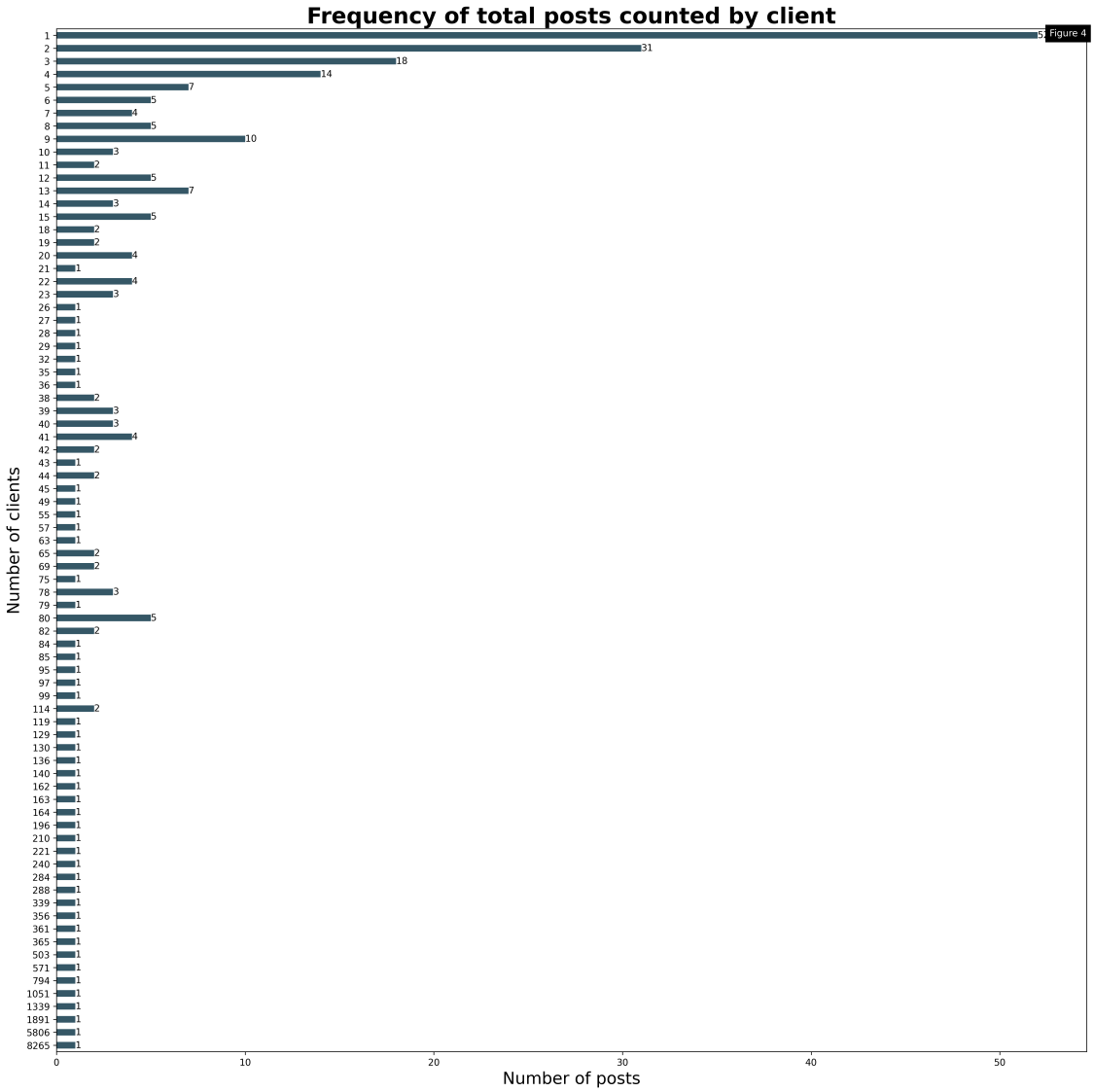
*Justification:*

1. Clients with very few posts (e.g., 1–4) contribute limited data, which might not provide reliable insights into alt text usage patterns. Focusing on top clients ensures you're analyzing data that carries more statistical weight.
2. Top clients (with higher activity levels) likely have a greater impact on the overall trend of alt text usage. These clients might represent larger user bases or more influential behaviors. Data Sparsity:
3. Clients with minimal posts may lead to skewed or noisy results, as their behavior could be random or inconsistent.
4. Insights from top clients are often more actionable. For example, improving alt text accessibility in popular clients can lead to broader platform-wide impacts.

To obtain a clear and precise understanding of the distributions of posts per client, the following bar plot will be counting the frequency of total posts per client.

[23]:
```
posts_count = data['posts'].value_counts().sort_index(ascending=False)

plt.figure(figsize=(20, 20), dpi=300)
bars = posts_count.plot(
        kind = 'barh',
```

```
        color = blue
)

for idx, count in enumerate(posts_count):
        plt.text(
                count,
                idx,
                str(count),
                ha = 'left',
                va = 'center',
        )

# Add labels and title
plt.xlabel('Number of posts', fontsize='18')
plt.ylabel('Number of clients', fontsize='18')
plt.title('Frequency of total posts counted by client', fontsize='24',␣
 ↪fontweight='bold')
figure_caption_generator()
plt.show()
```




The frequency of total post per client displayed in figure 4 highlights a relevant amount of clients (52 out of 265) that published only one post. Furthermore, we can observe that the majority of clients that contains between 1 and 4 posts, and that very few clients contain more than 1000 posts.

In the following sections, we will be diving deeper in calculating the distribution of posts per client, and assessing potential correlations with alt text usage.

### 6.3.3 Find the mode of posts per client

As the graph above shows, most of the clients have one post. Therefore, **the most frequent total number of posts clients have**, the mode, should be equal to one.

[24]:

```
mode = data['posts'].mode()[0] # Using [0] to get only the first mode, as this␣
  ↪function can return multiple values.
print(f'Mode: {mode}')
```

```
Mode: 1
```

### 6.4 Grouping by client popularity

The results of the observations on the most and least popular clients brought us to divide the analysis in two groups, and explore the use of alt text separately.

At this point, we will be determining the most and least popular clients depending on the number of posts published by each of them.

[25]:
```
# Determining arbitrary post numbers thresholds, based on previous graphs
top = 1000
worse = 4
```

[26]:
```
data_clients_least = data[data['posts'] <= worse]
posts_clients_least = data_clients_least['posts'].sum()
clients_least = len(data_clients_least)
posts_clients_least_pct = posts_clients_least/posts_tot*100

print(f'''
Total of clients: {clients}.        Clients containing less than {worse} posts:␣
  ↪{clients_least}.
Total of posts: {posts_tot}.        Total of posts from the {clients_least}␣
  ↪less popular clients: {posts_clients_least}.
Around {round(posts_clients_least_pct,1)}% of the posts comes from the␣
  ↪{round(clients_least/clients*100)}% least popular clients.
''')
```

```
Total of clients: 265.  Clients containing less than 4 posts: 115.
Total of posts: 28428.  Total of posts from the 115 less popular clients: 224.
Around 0.8% of the posts comes from the 43% least popular clients.
```

[27]:
```
data_clients_top = data[data['posts'] >= top]
posts_clients_top = data_clients_top['posts'].sum()
clients_top = len(data_clients_top)
posts_clients_top_pct = posts_clients_top/posts_tot*100

print(f'''
Total of clients: {clients}.        Clients containing more than {top} posts:␣
  ↪{clients_top}.
Total of posts: {posts_tot}.        Total of posts from top {clients_top}␣
  ↪clients: {posts_clients_top}.
```

```
Around {round(posts_clients_top_pct)}% of the posts come from the top␣
  ↪{round(clients_top/clients*100)}% of the clients.
''')
```

```
Total of clients: 265.  Clients containing more than 1000 posts: 5.
Total of posts: 28428.  Total of posts from top 5 clients: 18352.
Around 65% of the posts come from the top 2% of the clients.
```

Figures 2, 3, and 4 show that indeed there is a very small number of clients as the source of the majority (65%) of the posts in the dataset. They are displayed as outliers, nevertheless, considering that the relevance of our analysis is based on the total amount of posts, we will be focusing on the top five clients in particular.

### 6.5 Client-specific alt text analysis

So far, we focused our analysis at a global level. Now, we will be diving deeper by investigating the alt text usage in relation to clients, and looking for possible patterns.

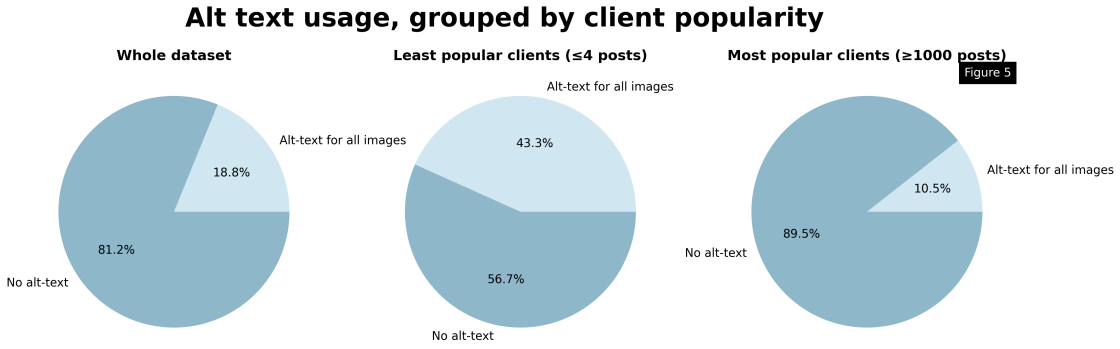
[28]:
```
print(f'There are {len(data[data['atxt_yes'] == 0])} clients whose posts do not␣
  ↪have any alt text.')
```

```
There are 138 clients whose posts do not have any alt text.
```

[29]:
```
clients_least_atxt_yes_tot = data_clients_least['atxt_yes'].sum()
clients_least_atxt_no_tot = data_clients_least['atxt_no'].sum()
clients_top_atxt_yes_tot = data_clients_top['atxt_yes'].sum()
clients_top_atxt_no_tot = data_clients_top['atxt_no'].sum()
```

[30]:
```
plt.figure(figsize=(15,5), dpi=300)

plt.subplot(1,3,1)
plt.pie(
        [atxt_yes_tot, atxt_no_tot],
        labels = pie_labels,
        colors = pie_colors,
        autopct = '%1.1f%%'
)
plt.title('Whole dataset', fontweight='bold')

plt.subplot(1,3,2)
plt.pie(
        [clients_least_atxt_yes_tot, clients_least_atxt_no_tot],
        labels = pie_labels,
        colors = pie_colors,
        autopct = '%1.1f%%'
)
plt.title(f'Least popular clients ( {worse} posts)', fontweight='bold')
```

```
plt.subplot(1,3,3)
plt.pie(
        [clients_top_atxt_yes_tot, clients_top_atxt_no_tot],
        labels = pie_labels,
        colors = pie_colors,
        autopct = '%1.1f%%'
)
plt.title(f'Most popular clients ( {top} posts)', fontweight='bold')

plt.suptitle('Alt text usage, grouped by client popularity', fontweight='bold',␣
 ↪fontsize=22)
#plt.legend()
figure_caption_generator()
plt.show()
```




In figure 5 we can observe the proportions of alt text usage from the whole data set (first pie chart), least popular clients (second pie chart), and the most popular clients (third pie chart).

In all cases, the majority of posts do not contain any alt text. In the most popular clients alt text usage is below the average of the dataset, and image descriptions are more used in the least popular clients, but this does not particularly influence the average because the number of posts coming from the latter is less than 1% of all the posts accounted for by this dataset.

### 6.5.1 Alt text usage in top 5 clients

In the dataset, only five clients generated more than 1000 posts, accounting for more than 60% of the total posts. Hence, we will be investingating the specific posts more in detail.

[31]:
```
plt.figure(figsize=(10, 5), dpi=300)

# Sort the data based on the percentage of posts with alt text␣
 ↪(descriptions_all_percent) in descending order
#data_clients_top = data_clients_top.sort_values(by='posts', ascending=True)
```

```
plt.barh(
        data_clients_top['client'],
        data_clients_top['atxt_yes'],
        color = blue,
        label = 'With Alt Text'
)
plt.barh(
        data_clients_top['client'],
        data_clients_top['atxt_no'],
        left = data_clients_top['atxt_yes'],
        color = light_blue,
        label = 'Without Alt Text'
)

plt.xlabel('Number of posts')
plt.ylabel('Clients')
plt.title('Number of posts with and without alt text in top 5␣
 ↪clients',fontweight='bold')
plt.legend()
figure_caption_generator()
plt.show()
```

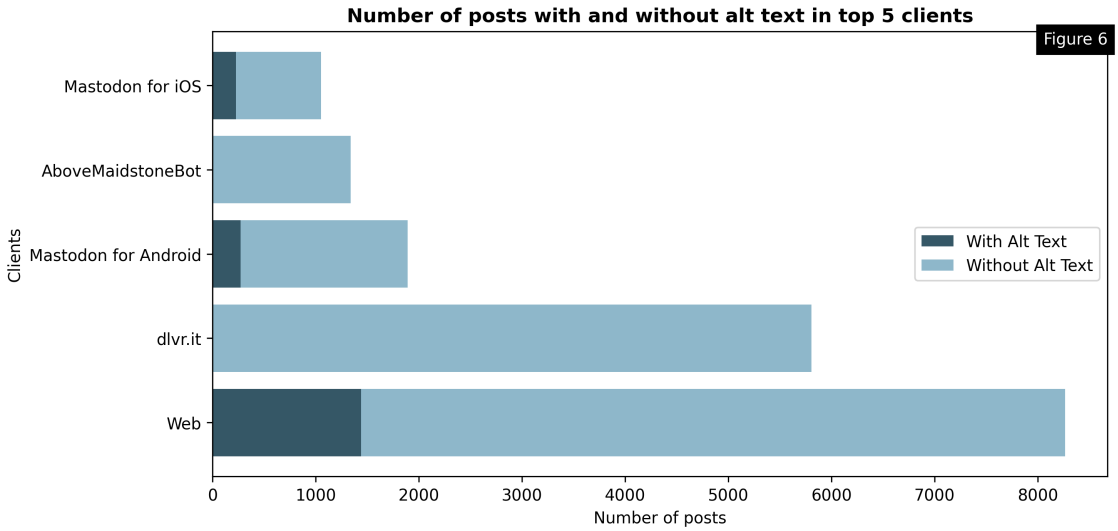



Figure 6 illustrates the first exploratory visualization focusing on the number of posts from the top 5 most popular Mastodon clients: “Web”, “dlvr.it”, “Mastodon for Android”, “AboveMaidstoneBot”, “Mastodon for iOS”. The data reveals that while these clients lead in the total number of posts, this does not necessarily imply a higher likelihood of alt text usage in their images. This discrepancy highlights the need to further investigate how each client supports or encourages the inclusion of alt text, potentially influencing the accessibility of content shared on their platforms. We will be addressing this in a dedicated section, at a later point.

[32]:
```
data_clients_top_sortpct = data_clients_top.sort_values(by='atxt_yes_pct')

plt.figure(figsize=(10, 5), dpi=300)
data_clients_top_barchart = plt.barh(
        data_clients_top_sortpct['client'],
        data_clients_top_sortpct['atxt_yes_pct'],
        label = 'Percentage of alt text use by client',
        color = blue
)
plt.xlim(0, 100)
plt.axvline(
        x = atxt_yes_tot_pct,
        color = light_blue,
        linestyle = ':',
        label = f'{atxt_yes_tot_pct}%, Average alt text use'
)
#plt.text(atxt_yes_tot_pct + 2, len(data_clients_top) - 1, f'Average value␣
 ↪{atxt_yes_tot_pct}%', color='red', verticalalignment='center',␣
 ↪horizontalalignment='left', fontsize=10)

for bar in data_clients_top_barchart:
        vertical_position = bar.get_y() + bar.get_height() / 2
        plt.text(
                2,
                vertical_position,
                f'{bar.get_width()}%',
                color = '#FFF',
                ha = 'left',
                va = 'center'
)

plt.xlabel('Number of posts')
plt.ylabel('Clients')
plt.title('Alt text usage percentage in top 5 clients', fontweight='bold')
plt.legend()
figure_caption_generator()
plt.show()
```

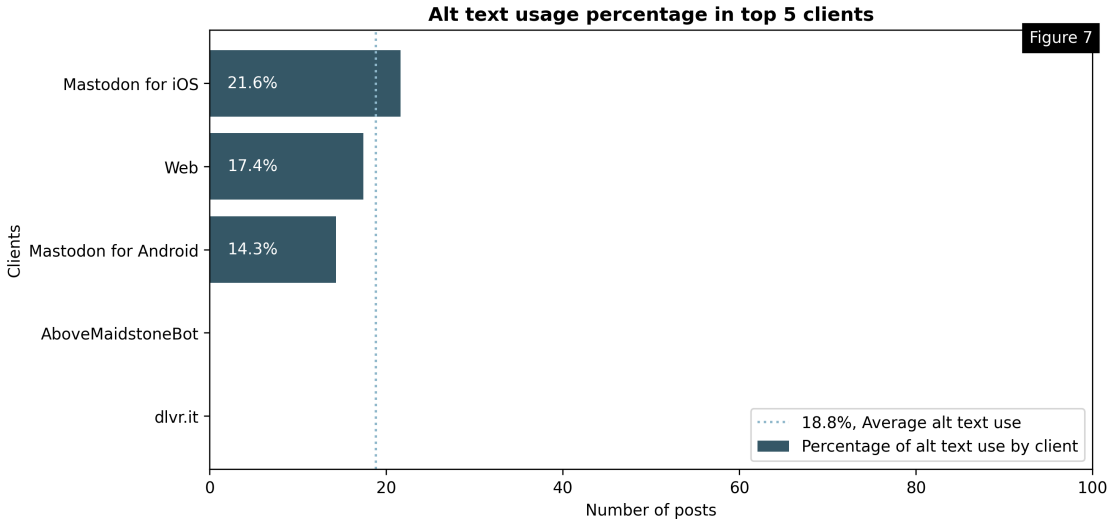
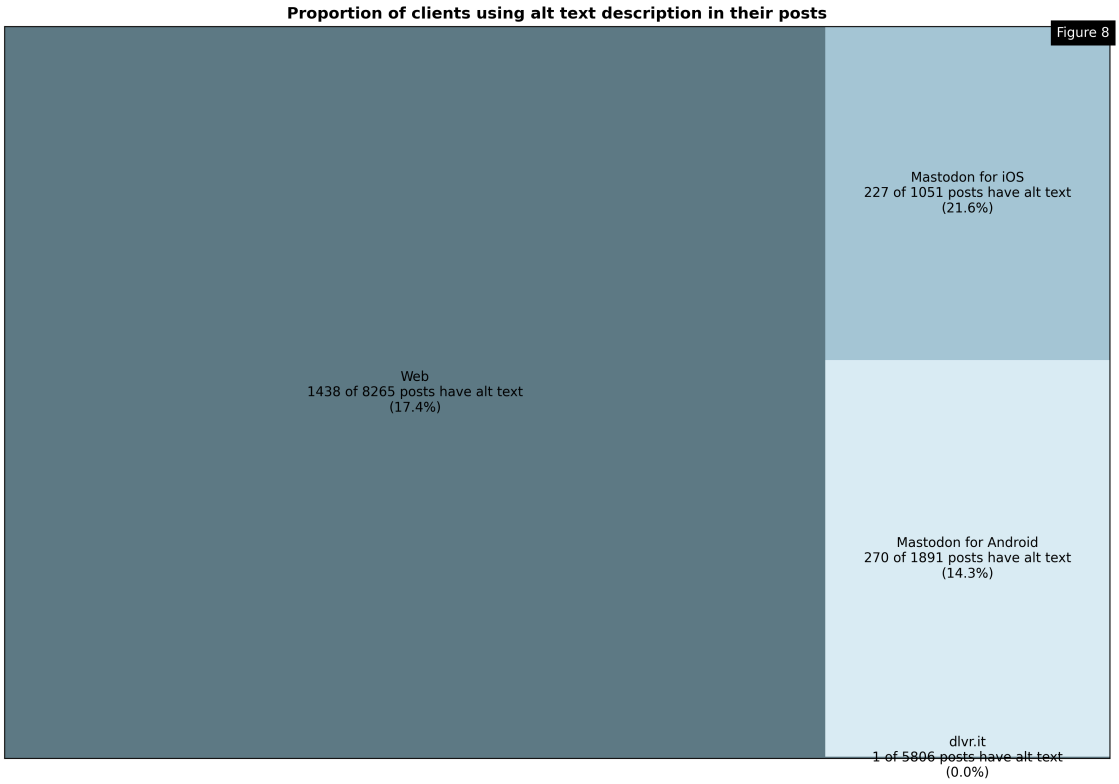



Figure 7 shows that, the among the most popular clients, "Mastodon for iOS" is the one that contains the greatest percentage of posts with image descriptions, and the only one above the average percentage of alt text usage in the dataset. "Web" and "Mastodon for Android" are the source for a minor but still relevant percentage of posts with alt text, while "AboveMaidstoneBot" and "dlvr.it" seem to be having no posts with image description.

[33]:
```
data_clients_top = data_clients_top[data_clients_top['atxt_yes'] > 0]

data_clients_top['label'] = data_clients_top.apply(
        lambda row: f"{row['client']}\n{row['atxt_yes']} of {row['posts']}␣
 ↪posts have alt text\n({row['atxt_yes_pct']:.1f}%)", axis=1
)

# Generate a color palette using matplotlib's colormaps
#colors = plt.get_cmap('tab20')(range(len(data_clients_top)))

plt.figure(figsize=(15, 10), dpi=300)
sq.plot(
        sizes = data_clients_top['atxt_yes'],
        label = data_clients_top['label'],
        color = [blue,blue,lighter_blue,light_blue],
        alpha = 0.8
)

plt.title('Proportion of clients using alt text description in their posts',␣
 ↪fontweight='bold')
plt.xticks([])
plt.yticks([])
figure_caption_generator()
```

```
plt.show()
```




Considering the total number of posts instead of the percentage relative to a single client, figure 8 shows that the Web client is still the source for the most posts (1438) with alt text in the dataset.

## 6.6 Exploring alt text features of the top 5 clients

### 6.6.1 Web

Following what is brought to light by figure 6, we will now focus more in detail on the characteristics of the most popular clients for what concerns image description.

The most popular client is "Web", the Web interface integrated in Mastodon's source code [10] by default. It is what users access to by accessing the URL of their instance. Being accessible via browser, instances administrators can change the user interface to encourage alt text usage, for example by adding CSS snippets [11]. The administrators of mastodon.social do not seem to have adopted any practice to encourage alt text in addition to the default Web interface settings.

It is possible to enable a warning notification before posting an image without any description from the Web interface, but only in Mastodon Glitch Edition, an experimental fork of Mastodon, with more features (image 2).

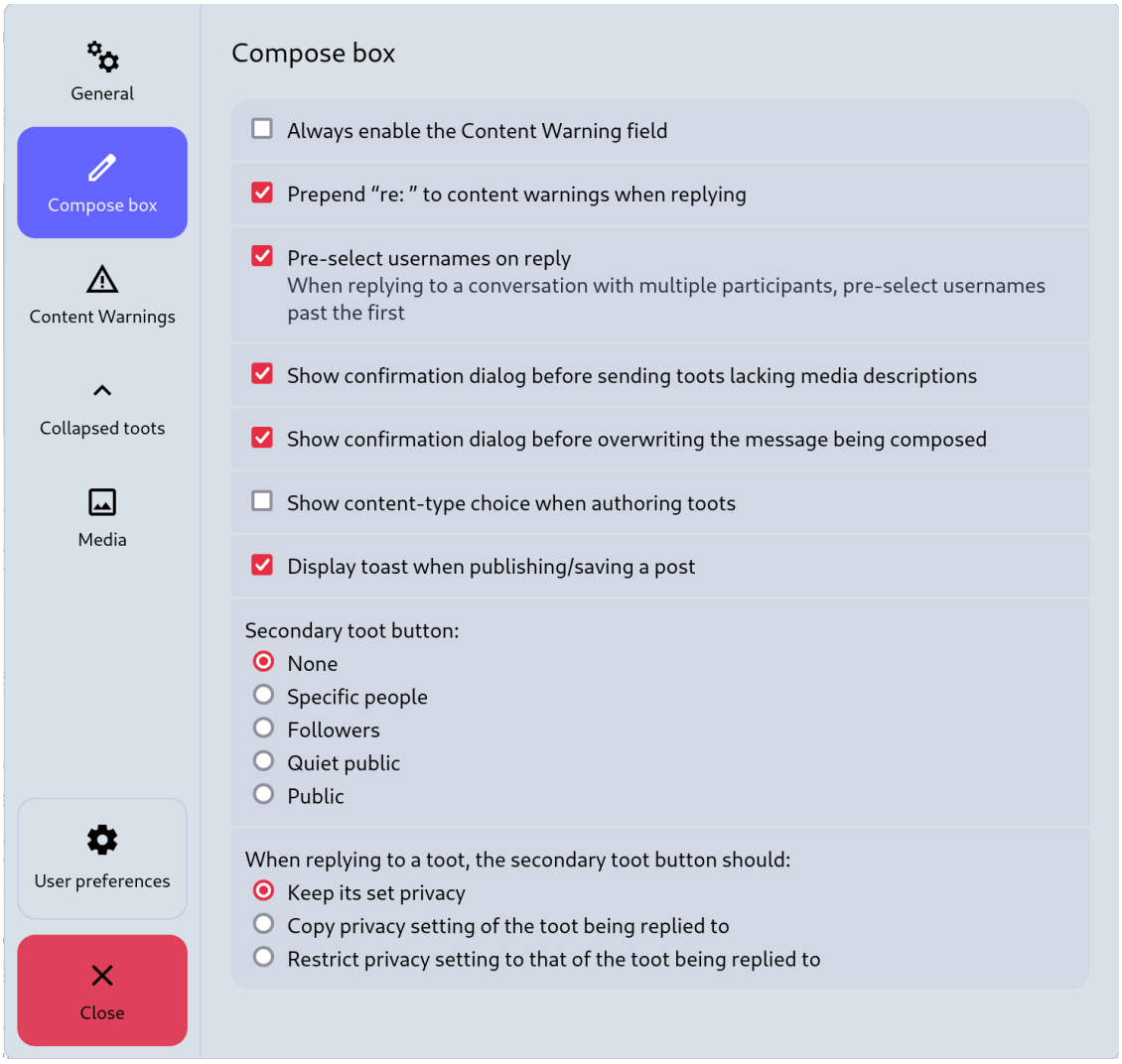



*Image 2*

Furthermore, in Mastodon Glitch Edition [12] is possible to enable a warning notification before sharing a post with no image descreption even when published by someone else (image 3).

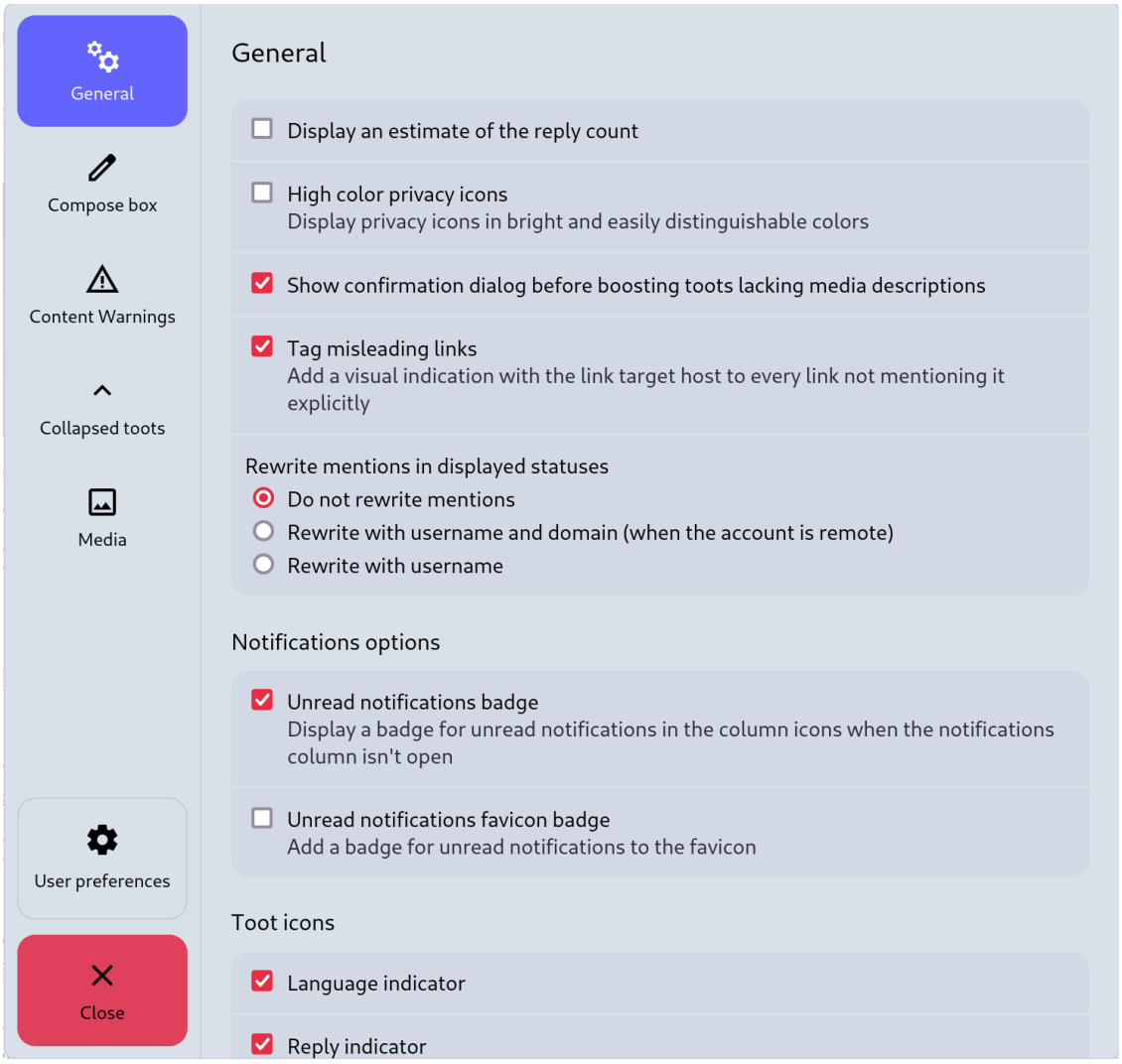



*Image 3*

### 6.6.2 Official Mastodon apps

The third and fifth most popular clients are the official Mastodon applications for Android [13] and for iOS [14], respecively. In these cases, too, the default interface does not particularly incentivize the use of alt text by default. Nevertheless, from the applications settings it is possible to enable a notification warning before posting an image without description (opt-in) (images 4 and 5).

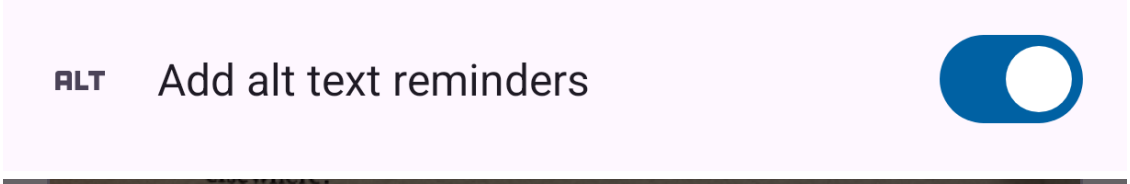



*Image 4*

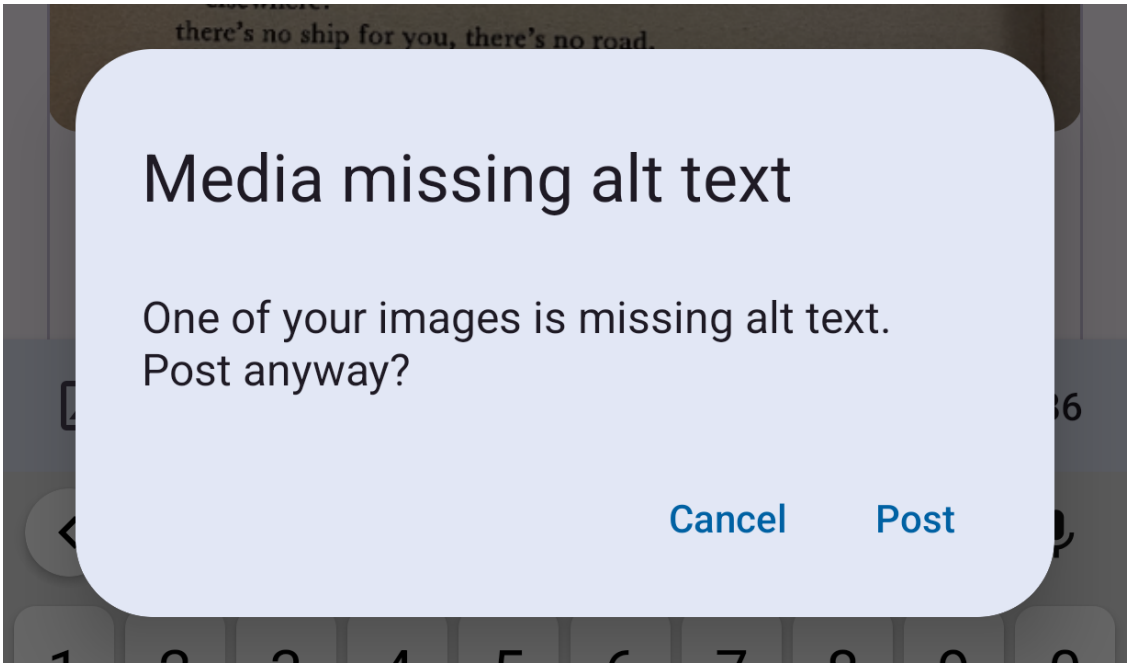



*Image 5*

### 6.6.3 Bots

Lastly, AboveMaidstoneBot and dlvr.it show very peculiar characteristics, strongly suggesting they are both automated software. We were unable to gather any information about the former, and we can only assume that it is probably a bot automating the publishing of some kind of content. The latter, instead, has an informative website [15] where it appears that it is a software built purposefully to enable cross-posting (the practice of publishing on one social media platform and automatically publishing the same content on another).

There are two possibilities explaining the lack of image descriptions: either the software does not support the inclusion of alt text, or images originally posted on other platforms almost never have an image description. The second option is the most likely, as only one post out of 5806 has alt text, meaning that the software supported alt text posting at least once.

## 6.7 Note on Analytical Methods

**Why Linear Regression, Logistic Regression, and KNN were not considered**.

Several analytical approaches were initially considered to understand patterns in the dataset, including linear regression, logistic regression, and k-nearest neighbors (KNN). However, these methods were not considered for the following reasons:

- Linear Regression: The correlation analysis between the percentage of posts with alt text and the total status count revealed a very weak relationship (Pearson correlation: -0.06, Spearman correlation: 0.03). This indicates a lack of linear association, a prerequisite for

linear regression. Applying this method would not yield significant insights or could lead to overfitting (not being able to generalize) or underfitting (trying to build a linear model in nonlinear data).

- Logistic Regression: Logistic regression is typically used when the target variable is binary or categorical. In this project, 90% of the data was continuous metrics, such as the percentage of posts with alt text and post count. Also, our goal was mostly to analyze the alt text usage instead of predicting binary outcomes like whether a client uses alt text or not.
- K-Nearest Neighbors (KNN): KNN requires a well-defined target variable for classification or regression, which our dataset lacked, as it did not include a categorical variable suitable for classification. One option would have been creating a variable, but it would not have been accurate due to insufficient significant features to support its construction. Instead, we opted to explore relationships between posts and the percentage of alt text usage or to group clients exhibiting similar behaviors through unsupervised clustering(shown in Future Development).

# 7 Conclusion and future development

## 7.1 Summary

In conclusion, the results of our analysis proved to be positive, as alt text usage in the dataset revealed to be strongly greater than in Twitter, but at the same time they are showing a lot of room for improvement for image descriptions in absolute terms, because only a minority of posts in the dataset had them. Furthermore, we found a strong inequality in the distribution of posts among clients, something we were not expecting, that forced us to adapt our approach and the flow of our analysis accordingly.

Concerning the environment of the Fediverse, even though they supposedly are more sensitive to ethical issues, this analysis showed without any doubt that the use of image descriptions is still too limited.

## 7.2 Challenges

While our initial expectations for the project included applying more advanced technical models, the reality of working with real-world data often brings unforeseen challenges that must first be addressed. During our analysis, we encountered limitations such as the lack of significant features for certain models and the need for extensive data cleaning and restructuring. These challenges highlighted an essential truth: data analysis is often less about immediately implementing complex models and more about solving foundational issues to ensure data quality and reliability.

Furthermore, our findings align with broader statistics, which indicate that much of the data on accessibility is either incomplete or underutilized. This underscores the pressing need for improved data collection practices and the prioritization of accessibility in digital content creation. By addressing these initial hurdles, future analyses can leverage more robust data to apply sophisticated techniques and generate actionable insights, ultimately contributing to a more inclusive digital landscape.

## 7.3 Future development

We believe that there is a great room for expansion and deepening of this analysis, not merely in terms of development, but chiefly in relation to the expansion of the dataset, in particular:

- by updating the dataset to a more recent date
- by including more posts in the dataset
- by gathering information from other servers outside of mastodon.social
- by cross-referencing alt text use among different servers, not only among different clients

In addition, we deem it would be very interesting, also using skills and tools learned from other courses, to investigate the qualities and the design on the most popular clients, pinpointing best practices to incentivize users to add image descriptions.

One of the future developments we explored, despite having no prior experience, was K-means clustering. We aimed to gain an initial understanding of this method, as we considered it a suitable technique for identifying grouping patterns based on status counts and the percentage of alt text usage in our dataset. This unsupervised learning approach enabled us to segment clients without relying on predefined labels or assumptions about relationships in the data.

As showed in the appendix (below), a key discovery from this analysis was the confirmation of clusters formed based on the number of posts and the percentage of alt text usage. For example, Web and dlvr.it were the only clients in Cluster 2, aligning with their status as top 5 clients in our project, characterized by high post volumes but relatively low alt text usage. In contrast, Mastodon for iOS, Mastodon for Android, and Mastodon.bot were grouped in Cluster 0.

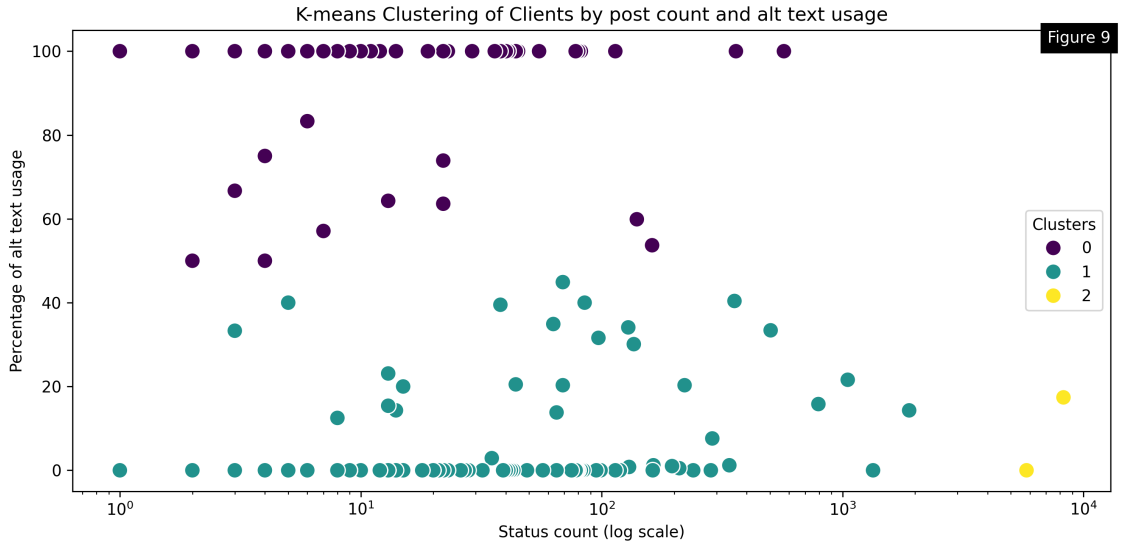
This approach provides a foundation for future analysis in our master's degree projects and contributes to making social media platforms more accessible. By uncovering meaningful insights, we aim to improve data quality and develop more robust analyses using this technique and others in the future (code details can be found in the Appendix).

### 7.4 Appendix

Figure 9 is the result of an experimental exploratory analysis of applying K-means clustering with scikit.

[34]:
```
# Select features for clustering
X = data[['posts', 'atxt_yes_pct']].copy()

# Standardize the data for better clustering
X_scaled = StandardScaler().fit_transform(X)

# Apply K-means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
data['cluster'] = kmeans.fit_predict(X_scaled)

#print(data[['client', 'posts', 'atxt_yes_pct', 'cluster']])

plt.figure(figsize=(10, 5), dpi=300)
sns.scatterplot(
        x = data['posts'],
        y = data['atxt_yes_pct'],
        hue = data['cluster'],
        palette = 'viridis',
        s = 100
```

```
)

plt.xscale('log')
plt.xlabel('Status count (log scale)')
plt.ylabel('Percentage of alt text usage')
plt.title('K-means Clustering of Clients by post count and alt text usage')
plt.legend(title='Clusters')
plt.tight_layout()
figure_caption_generator()
plt.show()
```




[35]:
```
data = data[data['cluster']==2]
data
```

[35]:

| | client | posts | atxt_yes | atxt_yes_pct | atxt_no | atxt_no_pct | cluster |
|---|---|---|---|---|---|---|---|
| 0 | Web | 8265 | 1438 | 17.4 | 6827 | 82.5 | 2 |
| 1 | dlvr.it | 5806 | 1 | 0.0 | 5805 | 100.0 | 2 |

[36]:
```
sil_score = silhouette_score(X_scaled, kmeans.labels_)
print(f'Silhouette Score: {sil_score:.2f}')
```

```
Silhouette Score: 0.85
```

## 8 References


**Dataset**: Bohacek, S. (2024). Mastodon.social alt text use by client app. Kaggle. https://www.kaggle.com/datasets/fourtonfish/mastodon-social-alt-text-use-by-client-app

- Bin Zia, H., He, J., Raman, A., Castro, I., Sastry, N. & Tyson, G.

(2023). Flocking to Mastodon: Tracking the Great Twitter Migration. https://doi.org/10.48550/arXiv.2302.14294
- Brady E. & Bigham, J.P. (2014). How companies engage customers around accessibility on social media. In Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility (ASSETS '14). Association for Computing Machinery, New York, NY, USA, 51–58. https://doi.org/10.1145/2661334.2661355
- Dixon, S. J. (2023). Mastodon: Number of registered users 2022-2023. Statista. https://www.statista.com/statistics/1376022/global-registered-mastodon-users/
- Field, A. (2005). Discovering statistics using IBM SPSS statistics. SAGE Publications.
- Garofolo, I., Bencini, G., & Arenghi, A. (2022). In Preface. Transforming our World through Universal Design for Human Development. Proceedings of the Sixth International Conference on Universal Design (UD2022). IOS Press. https://www.iospress.com/catalog/books/transforming-our-world-through-universal-design-for-human-development
- Gleason, C., Carrington, P., Cassidy, C., Ringel Morris, M., Kitani, K. M., & Bigham, J.P. (2019). "It's almost like they're trying to hide it": How User-Provided Image Descriptions Have Failed to Make Twitter Accessible. 13 May 2019. WWW '19: The World Wide Web Conference, 549-559. https://doi.org/10.1145/3308558.3313605
- Graham, F. (2023). Daily briefing: What the end of Twitter's free API means for research. Nature (London), 2023-02. https://doi.org/10.1038/d41586-023-00480-9
- Lemmer-Webber, C., Tallon, J., Shepherd, E., Guy, A., & Prodromou, E. (2018). ActivityPub. W3C Recommendation 23 January 2018. https://www.w3.org/TR/2018/REC-activitypub-20180123/
- Pearson, K. (1895). Note on Regression and Inheritance in the Case of Two Parents. Proceedings of the Royal Society of London, 240–242. https://www.jstor.org/stable/115794
- Spearman, C. (1904). The Proof and Measurement of Association between Two Things. The American Journal of Psychology, 15(1), 72–101. https://doi.org/10.2307/1412159
- Tukey, J. W. (1977). Exploratory data analysis. Addison-Wesley. https://doi.org/10.1007/978-0-387-32833-1_136
- Voykinska, V. Azenkot, S., Wu, S., & Leshed, G. (2016). How Blind People Interact with Visual Content on Social Networking Services. CSCW '16: Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing, 1584–1595. https://doi.org/10.1145/2818048.2820013
- Zulli, D., Liu, M., & Gehl, R. (2020). Rethinking the "social" in "social media": Insights into topology, abstraction, and scale on the Mastodon social network. New Media & Society, 22(7), 1188–1205. https://doi.org/10.1177/1461444820912533

[1]: https://linkedin.com/in/cristal-rivera-picado/
[2]: https://tommi.space/
[3]: https://github.com/alterism
[4]: https://mastodon.social/about
[5]: https://www.kaggle.com/datasets/fourtonfish/mastodon-social-alt-text-use-by-client-app
[6]: https://stefanbohacek.com/
[7]: https://www.mit.edu/~amini/LICENSE.md
[8]: https://ois2.tlu.ee/tluois/subject/ULP6613-23265)
[9]: https://aissprogram.eu
[10]: https://github.com/mastodon/mastodon/
[11]: https://multiline.co/mment/2022/12/alt-text-mastodon-css/

[12]: https://github.com/glitch-soc/mastodon
[13]: https://github.com/mastodon/mastodon-android
[14]: https://github.com/mastodon/mastodon-ios
[15]: https://dlvr.it