```
! pip install vaderSentiment
```

Show hidden output

```
! pip install qdrant_client
```

Show hidden output

SCRAPING THE DATA

YOUTUBE

```
from googleapiclient.discovery import build
import time

API_KEY = "YOUTUBE_API_KEY"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"

def get_video_comments(video_id, N=5):
    youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION, developerKey=API_KEY)
    comments = []
    try:
        request = youtube.commentThreads().list(
            part="snippet",
            videoId=video_id,
            maxResults=N,
            order="relevance"
        )
        response = request.execute()

        for item in response.get('items', []):
            comment = item['snippet']['topLevelComment']['snippet']['textDisplay']
            comments.append(comment)

    except Exception as e:
        print(f"Comments disabled or error for video {video_id}: {e}")

    return comments


def scrape_youtube(query="smart fan", N=100):
    youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION, developerKey=API_KEY)
    results = []
    next_page_token = None

    while len(results) < N:
        request = youtube.search().list(
            q=query,
            part="snippet",
            type="video",
            maxResults=min(50, N - len(results)),
            pageToken=next_page_token
        )
        response = request.execute()

        for item in response.get('items', []):
            video_id = item['id']['videoId']
            title = item['snippet']['title']
            description = item['snippet'].get('description', '')
            url = f"https://www.youtube.com/watch?v={video_id}"

            # Get video statistics
            try:
                stats_request = youtube.videos().list(
                    part="statistics",
                    id=video_id
                )
                stats_response = stats_request.execute()
                stats = stats_response['items'][0]['statistics']
                likes = int(stats.get('likeCount', 0))
                views = int(stats.get('viewCount', 0))
            except Exception as e:
                print(f"Could not get statistics for video {video_id}: {e}")
                likes, views = 0, 0

            # Get top comments
            comments = get_video_comments(video_id)

            results.append({
                "ID": video_id,
                "Title": title,
                "Description": description,
                "URL": url,
                "Comments": comments,
                "Likes": likes,
                "Views": views
            })

        next_page_token = response.get('nextPageToken')
        if not next_page_token:
            break

        time.sleep(1)  # Avoid hitting quota limits

    return results



smart_fan_videos = scrape_youtube(query="smart fan", N=200)
print(f"Scraped {len(smart_fan_videos)} videos.")
```

Show hidden output

```
import pandas as pd
data = pd.DataFrame(smart_fan_videos)
data
```

	ID	Title	Description	URL	Comments	Likes	Views
0	Ycj6LWtT_Nw	AHAWILL ceiling fan with light Remote Control ...	Within 5 seconds after power on, press the "PA...	https://www.youtube.com/watch?v=Ycj6LWtT_Nw	[My box came with absolutely zero directions, ...	11	5017
1	PG05pZ6S8w0	Make Your Ceiling Fans Smart: The Ultimate Guide!	Discover the best way to make YOUR ceiling fan...	https://www.youtube.com/watch?v=PG05pZ6S8w0	[Thank you to Best Buy for sponsoring this vid...	1163	76800
2	2Y4kaHRWjUE	5 Best Smart Ceiling Fans 2025 | Best Ceiling ...	5 Best Smart Ceiling Fans 2025 | Best Ceiling ...	https://www.youtube.com/watch?v=2Y4kaHRWjUE	[5 Best Smart Ceiling Fans 2025 | Best Ceiling...	81	13327
3	MbnPOZj9GI0	2024&#39;s Best Value Ceiling Fan! - DREO CLF7...	The New King Of Smart Ceiling Fans! - DREO CF7...	https://www.youtube.com/watch?v=MbnPOZj9GI0	[DREO CF712S Smart Ceiling Fan: <a href="https...	2614	47125
4	VOrnBVxSDvQ	Smart Ceiling Fan 🫨 #fansound #fan #shortvideo...	Smart Ceiling Fan #fansound #fan #shortvideo #...	https://www.youtube.com/watch?v=VOrnBVxSDvQ	[Good 👍 🖤🖤🖤🖤🖤, Nice, Nice fan, Fake, Everyth...	8888	2458146
...	...	...	...	...	...	...	...
195	vzcIdyXZhvk	Installation of Ottomate Smart Fan	For more details check out: https://www.ottoma...	https://www.youtube.com/watch?v=vzcIdyXZhvk	[Nice to see many companies getting into Smart...	404	55874
196	Qrm_X6eNcVU	Haiku L Series Smart Fan Review: &quot;Alexa, ...	Haiku L Series smart fan review: https://smart...	https://www.youtube.com/watch?v=Qrm_X6eNcVU	[Do you have a separate link for that cool Hai...	383	60498
197	Ip4AoI6OeOs	Carro smart ceiling fan	Best Smart Ceiling fan! Here	https://www.youtube.com/watch?	[Nice video 👍 my friend. Interesting	60	12715

BRANDS

```
brand_list = [
    'Xiaomi',
    'Atomberg',
    'LG',
    'TORRAS',
    'Polycab',
    'Kühl',
    'Dreo',
    'Hunter',
    'Tuya',
    'Duux',
    'Wahoo',
    'Smartwares',
    'Crompton',
    'Bajaj',
    'Havells',
    'ZyXEL',
    'JAZAVA',
    'ELEHINSER',
    'YOUKAIN',
    'Globe',
    'Ondaum',
    'AHAWILL',
    'Govee',
    'Orient',
    'Rise Mode',
    'Usha',
    'Havells'
]
```

Collecting BrandCounts

I have Collected number of brands which is mentioned in each videos title,description and comment and i have also considered repetations and redundancy so i have also done cleaning

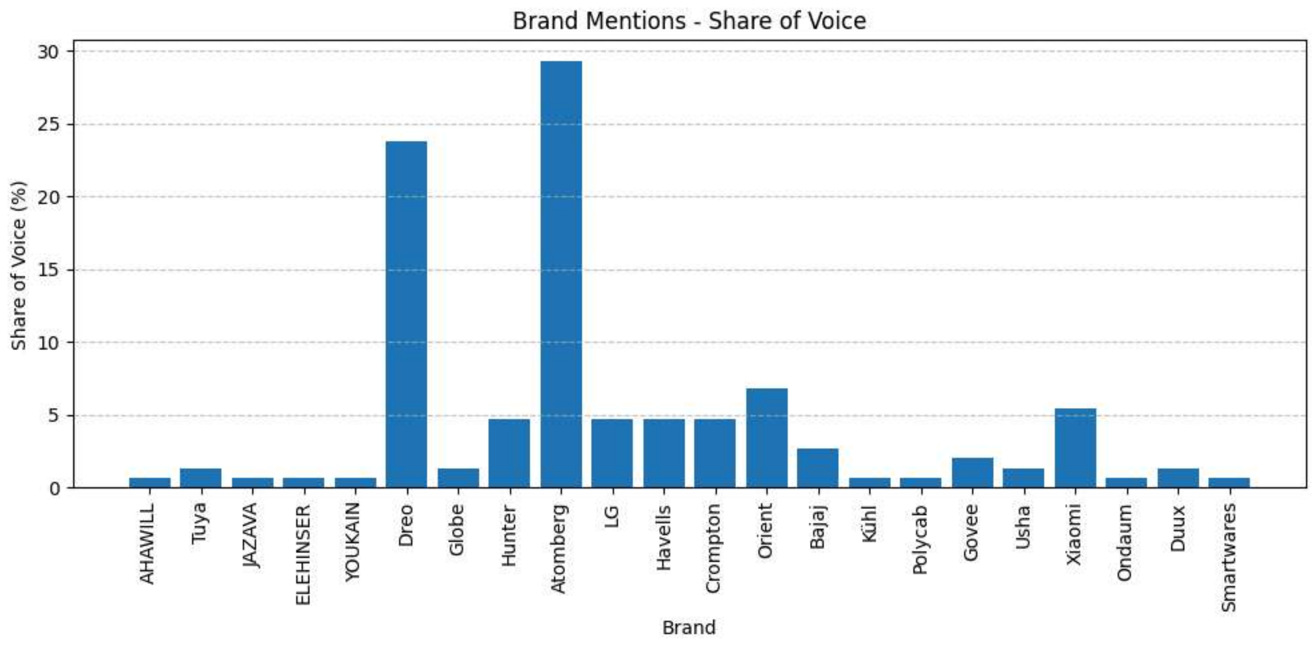
```
from collections import defaultdict

# Initialize brand mention counts
brand_counts = defaultdict(int)

for idx, row in data.iterrows():
    # Get title and description (as lowercase strings)
    title = str(row.get('Title', '')).lower()
    description = str(row.get('Description', '')).lower()

    # Get comments (ensure it's a list of strings)
    comments = row.get('Comments', [])
    if not isinstance(comments, list):
        comments = []

    # De-duplicate and clean up comments
    cleaned_comments = set(str(c).lower().strip() for c in comments if isinstance(c, str))

    # Combine all text sources into one list
    all_texts = [title, description] + list(cleaned_comments)

    # Track which brands were found in this post
    brands_found_in_post = set()

    for brand in brand_list:
        brand_lower = brand.lower()
        for text in all_texts:
            if brand_lower in text:
                brands_found_in_post.add(brand)
                break  # Stop after the first match in any field

    # Update global brand counts (once per video post)
    for brand in brands_found_in_post:
        brand_counts[brand] += 1

# Optional: Sort and print nicely
sorted_counts = dict(sorted(brand_counts.items(), key=lambda x: x[1], reverse=True))
print(sorted_counts)
```

```
{'Atomberg': 43, 'Dreo': 35, 'Orient': 10, 'Xiaomi': 8, 'Hunter': 7, 'LG': 7, 'Havells': 7, 'Crompton': 7, 'Bajaj': 4, 'Govee': 3, 'Tuya
```

```
total = sum(brand_counts.values())
print(f"Total mentions: {total}")
Atomberg_mentions = brand_counts.get('Atomberg', 0)
print(f"Atomberg mentions: {Atomberg_mentions}")
print(f"Atomberg SoV : {Atomberg_mentions / total * 100:.2f}%")
```

```
Total mentions: 147
Atomberg mentions: 43
Atomberg SoV : 29.25%
```

```
import matplotlib.pyplot as plt

# Convert to lists
brands = list(brand_counts.keys())
mentions = list(brand_counts.values())
sov = [m / total * 100 for m in mentions]  # SoV as percentage

# Plot
plt.figure(figsize=(10, 5))
plt.bar(brands, sov)
plt.xlabel('Brand')
plt.ylabel('Share of Voice (%)')
plt.title('Brand Mentions - Share of Voice')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Engagement analysis

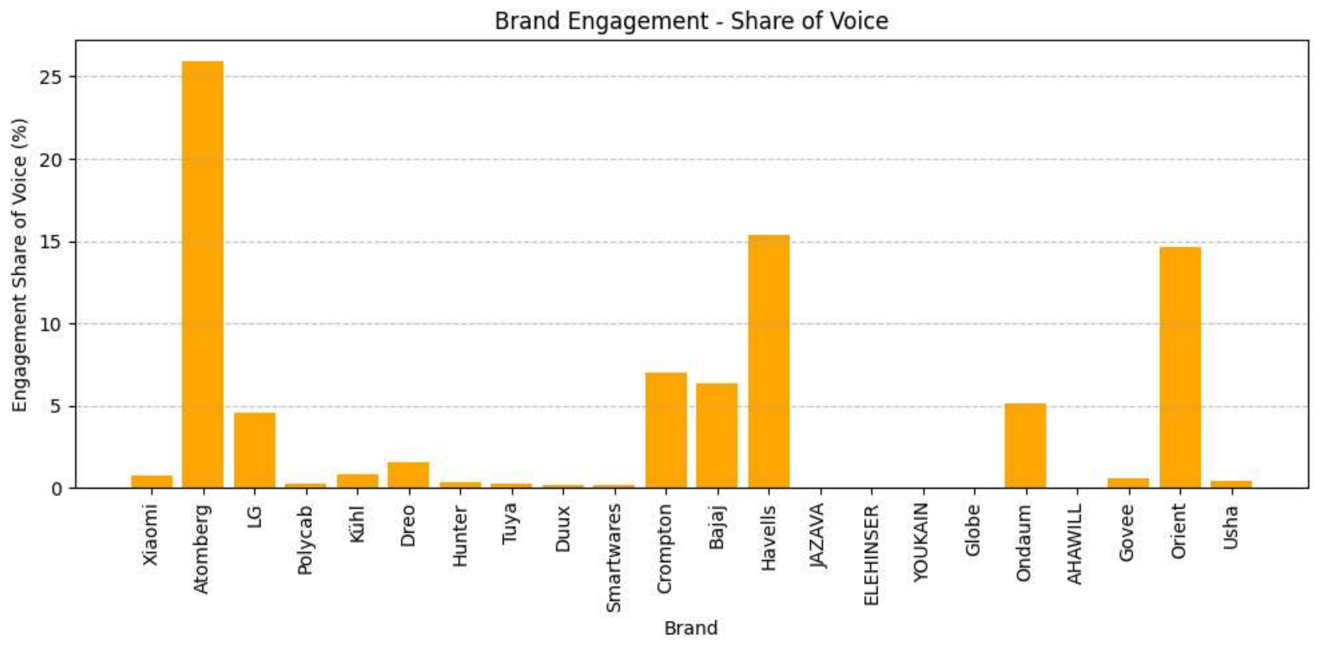
```
brand_likes = defaultdict(int)
brand_views = defaultdict(int)

for idx, row in data.iterrows():
    title = str(row.get('Title', '')).lower()
    description = str(row.get('Description', '')).lower()
    comments = row.get('Comments', [])
    if not isinstance(comments, list):
        comments = []
    cleaned_comments = set(str(c).lower().strip() for c in comments if isinstance(c, str))
    all_texts = [title, description] + list(cleaned_comments)

    brands_found_in_post = set()
    for brand in brand_list:
        brand_lower = brand.lower()
        if any(brand_lower in text for text in all_texts):
            brands_found_in_post.add(brand)

    # Add engagement data for brands mentioned in this video
    for brand in brands_found_in_post:
        brand_likes[brand] += int(row.get("Likes", 0))
        brand_views[brand] += int(row.get("Views", 0))
```

decided that a "like" is four times more valuable than a "view"

```
# Calculate Engagement-weighted SoV
total_engagement = sum(4*brand_likes[b] + brand_views[b] for b in brand_list)
engagement_sov = {
    brand: ((4*brand_likes[brand] + brand_views[brand]) / total_engagement) * 100
    for brand in brand_list if (4*brand_likes[brand] + brand_views[brand]) > 0
}

print("Brand Likes:", dict(brand_likes))
print("Brand Views:", dict(brand_views))
print("Engagement-weighted SoV:", engagement_sov)
print(f"Atomberg Engagement SoV: {engagement_sov.get('Atomberg', 0):.2f}%")
```

```
Brand Likes: {'AHAWILL': 11, 'Tuya': 2697, 'JAZAVA': 81, 'ELEHINSER': 81, 'YOUKAIN': 81, 'Dreo': 14068, 'Globe': 445, 'Hunter': 3094, 'A
Brand Views: {'AHAWILL': 5017, 'Tuya': 176822, 'JAZAVA': 13327, 'ELEHINSER': 13327, 'YOUKAIN': 13327, 'Dreo': 1003070, 'Globe': 32681, '
Engagement-weighted SoV: {'Xiaomi': 0.7513531049607737, 'Atomberg': 25.90032490578013, 'LG': 4.545821353355844, 'Polycab': 0.27092172443
Atomberg Engagement SoV: 25.90%
```

```
import matplotlib.pyplot as plt

# Convert engagement_sov dictionary to lists
brands = list(engagement_sov.keys())
sov_values = list(engagement_sov.values())

# Plot Engagement Share of Voice
plt.figure(figsize=(10, 5))
plt.bar(brands, sov_values, color='orange')
plt.xlabel('Brand')
plt.ylabel('Engagement Share of Voice (%)')
plt.title('Brand Engagement - Share of Voice')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



COLLECTING POSTIVE FEEDBACKS

USING VADER SENTIMENT ANALYSER TO COLLECT POSITIVE COMMENTS

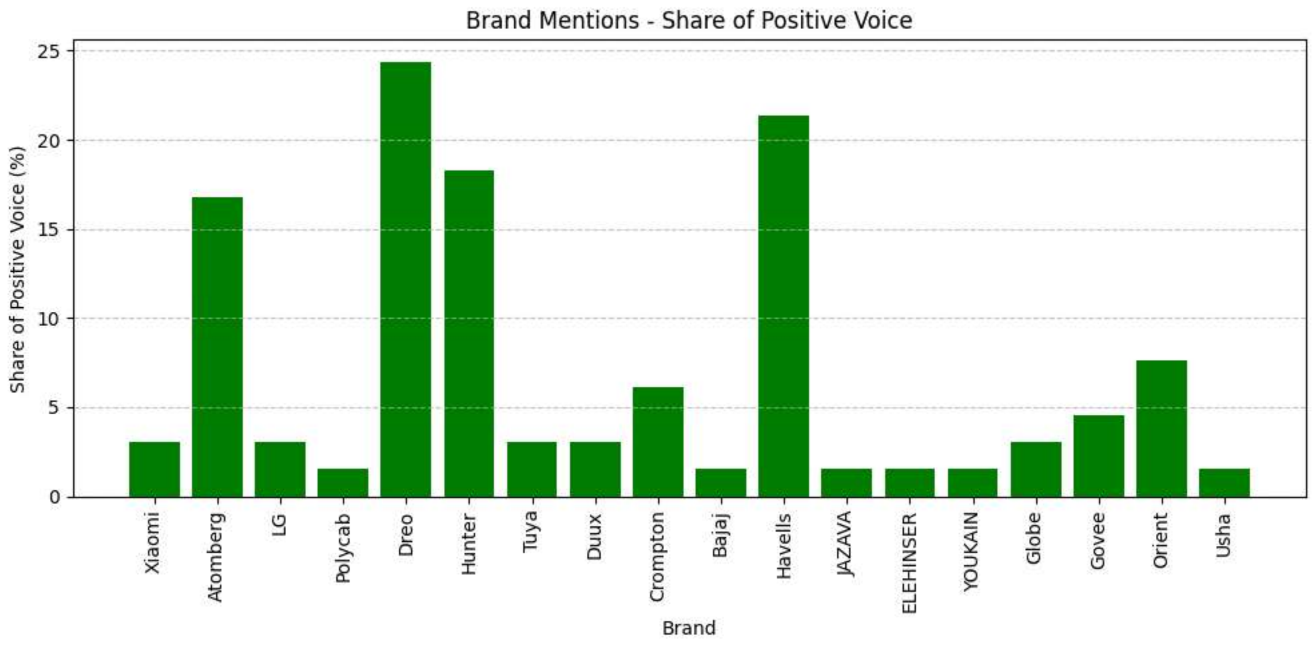
```
from collections import defaultdict
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()
positive_mentions = defaultdict(int)
negative_mentions = defaultdict(int)

for comment_list in data['Comments']:
    # Skip NaNs or non-list entries
    if not isinstance(comment_list, list):
        continue

    for comment in comment_list:
        if not isinstance(comment, str):
            continue

        text = comment.lower()
        sentiment = analyzer.polarity_scores(text)

        if sentiment['compound'] >= 0.05:
            for brand in brand_list:
                if brand.lower() in text:
                    positive_mentions[brand] += 1

        elif sentiment['compound'] < -0.05:
            for brand in brand_list:
                if brand.lower() in text:
                    negative_mentions[brand] += 1

# To view as sorted dictionary:
print("Positive Mentions:", dict(sorted(positive_mentions.items(), key=lambda x: x[1], reverse=True)))
print("Negative Mentions:", dict(sorted(negative_mentions.items(), key=lambda x: x[1], reverse=True)))
```

```
Positive Mentions: {'Dreo': 16, 'Havells': 14, 'Hunter': 12, 'Atomberg': 11, 'Orient': 5, 'Crompton': 4, 'Govee': 3, 'Tuya': 2, 'Globe':
Negative Mentions: {'Atomberg': 4, 'Hunter': 3, 'Crompton': 2, 'Usha': 1, 'Bajaj': 1, 'LG': 1}
```

```
total_positive = sum(positive_mentions.values())
positive_sov = {
    brand: (positive_mentions[brand] / total_positive) * 100
    for brand in brand_list if positive_mentions[brand] > 0
}


print(f"Atomberg SoPV: {positive_sov.get('Atomberg', 0):.2f}%")
```

```
Atomberg SoPV: 13.58%
```

```
import matplotlib.pyplot as plt

# Convert to lists
brands = list(positive_sov.keys())
positive_mentions = list(positive_sov.values())
sov = [m / total_positive * 100 for m in positive_mentions]

# Plot
plt.figure(figsize=(10, 5))
plt.bar(brands, sov, color="green")
plt.xlabel('Brand')
plt.ylabel('Share of Positive Voice (%)')
plt.title('Brand Mentions - Share of Positive Voice')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

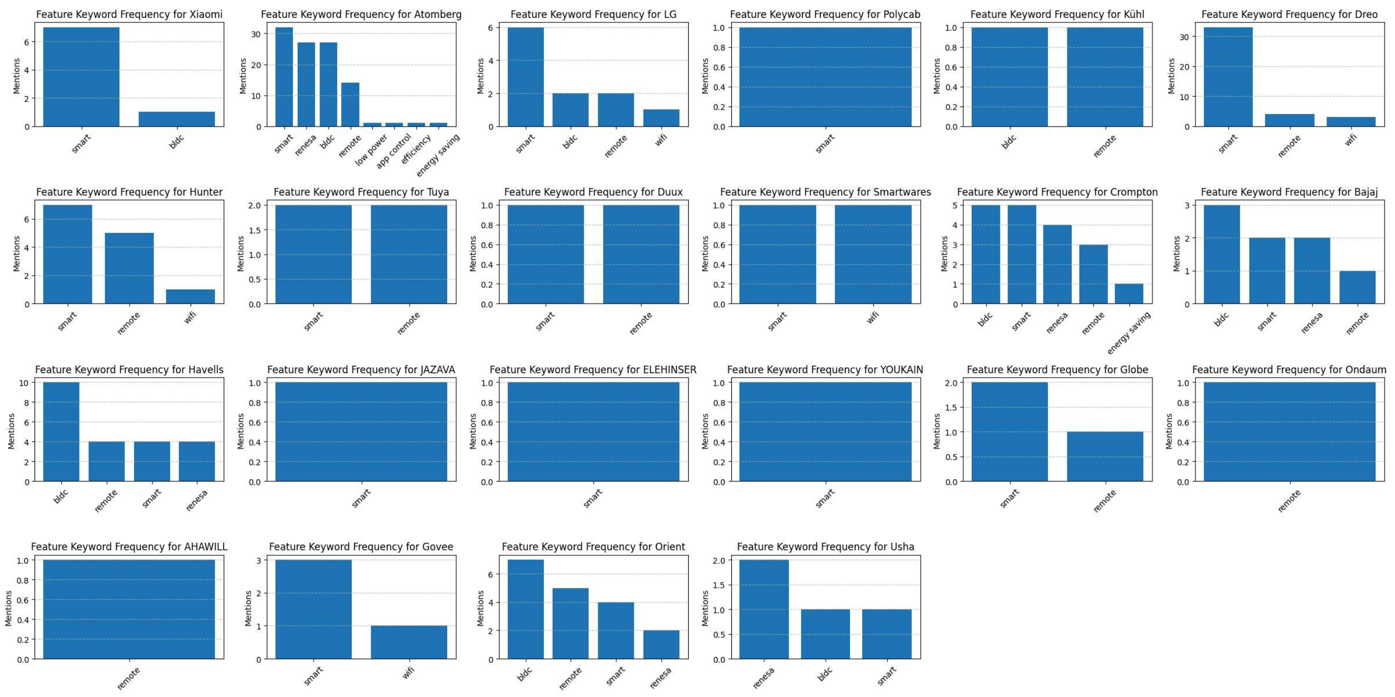


Drawing insights from multiple similar keywords

```
from collections import Counter
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string

# Ensure stopwords are available
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
# Define the feature keywords you care about (lowercase for matching)
feature_keywords = [
    "bldc", "smart", "remote", "energy saving", "low power",
    "wifi", "app control", "noiseless", "renesa", "efficiency"
]

# Initialize dictionary to count per brand
brand_feature_counts = {brand: defaultdict(int) for brand in brand_list}

# Iterate through dataset
for idx, row in data.iterrows():
    # Combine relevant text fields
    title = str(row.get('Title', '')).lower()
    description = str(row.get('Description', '')).lower()
    comments = row.get('Comments', [])
    if not isinstance(comments, list):
        comments = []
    comments_text = " ".join([str(c).lower() for c in comments if isinstance(c, str)])

    combined_text = " ".join([title, description, comments_text])

    # Check each keyword
    for keyword in feature_keywords:
        if keyword in combined_text:
            # Increment for all brands mentioned in the post
            for brand in brand_list:
                if brand.lower() in combined_text:
                    brand_feature_counts[brand][keyword] += 1


# Plot keyword frequency for each brand
# Determine the number of brands with keyword matches
brands_with_features = [brand for brand, features in brand_feature_counts.items() if sum(features.values()) > 0]
num_brands = len(brands_with_features)

# Calculate the number of rows and columns for subplots
# Aim for 5-6 plots per row
cols = min(len(feature_keywords), 6)  # Max 6 columns
rows = (num_brands + cols - 1) // cols # Calculate rows needed

fig, axes = plt.subplots(rows, cols, figsize=(cols * 4, rows * 3))
axes = axes.flatten() # Flatten the 2D array of axes for easy iteration

for i, brand in enumerate(brands_with_features):
    features = brand_feature_counts[brand]
    sorted_features = dict(sorted(features.items(), key=lambda x: x[1], reverse=True))

    axes[i].bar(sorted_features.keys(), sorted_features.values())
    axes[i].set_title(f"Feature Keyword Frequency for {brand}")
    axes[i].set_ylabel("Mentions")
    axes[i].tick_params(axis='x', rotation=45)
    axes[i].grid(axis='y', linestyle='--', alpha=0.7)


# Hide any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



SEMANTIC ANALYSIS

```
all_comments = []
all_video_titles = []
all_video_urls = []

for idx, row in data.iterrows():
    comments = row.get('Comments', [])
    title = str(row.get('Title', ''))
    video_id = row.get('Video_ID', '')  # Assuming you have this column
    video_url = f"https://youtube.com/watch?v={video_id}"

    if isinstance(comments, list):
        for c in comments:
            if isinstance(c, str):
                all_comments.append(c)
                all_video_titles.append(title)
                all_video_urls.append(video_url)
```

```
len(all_comments), len(all_video_titles), len(all_video_urls)
```

```
(705, 705, 705)
```

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-mpnet-base-v2")
comment_embeddings = model.encode(all_comments, show_progress_bar=True)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%                                349/349 [00:00<00:00, 10.9kB/s]
config_sentence_transformers.json: 100%           116/116 [00:00<00:00, 6.83kB/s]
README.md:       10.4k/? [00:00<00:00, 485kB/s]
sentence_bert_config.json: 100%                   53.0/53.0 [00:00<00:00, 2.59kB/s]
config.json: 100%                                 571/571 [00:00<00:00, 39.3kB/s]
model.safetensors: 100%                           438M/438M [00:04<00:00, 135MB/s]
tokenizer_config.json: 100%                       363/363 [00:00<00:00, 18.0kB/s]
vocab.txt:       232k/? [00:00<00:00, 7.05MB/s]
tokenizer.json:       466k/? [00:00<00:00, 18.1MB/s]
special_tokens_map.json: 100%                     239/239 [00:00<00:00, 18.2kB/s]
config.json: 100%                                 190/190 [00:00<00:00, 10.1kB/s]
Batches: 100%                                     23/23 [02:48<00:00,  1.42s/it]
```

```python
from qdrant_client import QdrantClient
from qdrant_client.http.models import VectorParams, Distance, PointStruct

client = QdrantClient(
    url="https://23a37241-1707-4f1a-8f5e-47c00502551d.us-west-1-0.aws.cloud.qdrant.io:6333",
    api_key="QDRANT_API_KEY"
)

client.recreate_collection(  # this deletes any existing collection with same name
    collection_name="SmartFan",
    vectors_config=VectorParams(size= comment_embeddings.shape[1], distance=Distance.COSINE),
)

points = [
    PointStruct(
        id=i,
        vector=comment_embeddings[i],
        payload={
            "text": all_comments[i],
            "video_title": all_video_titles[i],
            "video_url": all_video_urls[i]
        }
    )
    for i in range(len(all_comments))
]

client.upsert(
    collection_name="SmartFan",
    points=points
)
```

```
/tmp/ipython-input-1074503806.py:9: DeprecationWarning: `recreate_collection` method is deprecated and will be removed in the future. Us
  client.recreate_collection(  # this deletes any existing collection with same name
UpdateResult(operation_id=0, status=<UpdateStatus.COMPLETED: 'completed'>)
```

```python
query = "Best BLDC Fan"
query_vec = model.encode(query)

hits = client.search(
    collection_name="SmartFan",
    query_vector=query_vec,
    limit=10  # Top 3 relevant results
)

for hit in hits:
    print("🎯 Score:", hit.score)
    print("📌 Title:", hit.payload["video_title"]
    print("💬 comments:", hit.payload["text"])
    print("🔗 URL:", hit.payload["video_url"])
    print("---\n")
```

```
🎯 Score: 0.72418267
📌 Title: Orient BLDC remote celling fan amazing speed #orient #remote #fans #amazing #speed
💬 comments: Best BLDC fan under 3000...
🔗 URL: https://youtube.com/watch?v=
---

🎯 Score: 0.6499137
📌 Title: Smart BLDC Ceiling Fan Designed in India Atomberg Ikano 1200mm Remote contr | Upto 65% Energy Saving
💬 comments: I heard the bldc fans are stopped working after some days.Can you please suggest the best BLDC fan without any complaints
🔗 URL: https://youtube.com/watch?v=
---

🎯 Score: 0.59092164
📌 Title: Atomberg Rensa Plus BLDC Fan #gadgetshorts  #shorts
💬 comments: Atormberg fan bldc fan <br>🖤🖤🖤🖤🖤🖤🖤🖤<br>🖤🖤🖤🖤🖤🖤🖤🖤<br>I will Buy Atormberg fan
🔗 URL: https://youtube.com/watch?v=
---

🎯 Score: 0.56239116
📌 Title: [NEW] Best Premium Fan | Best Silent Ceiling Fan | with Best Smart Fan | Best BLDC Fan 2025
💬 comments:  😎 Important Videos👇🏻<br>▶ Ceiling Fan Playlist😎: <a href="https://bit.ly/EtaFan">https://bit.ly/EtaFan</a><br>OR <br
🔗 URL: https://youtube.com/watch?v=
---

🎯 Score: 0.5514368
📌 Title: #atomberg Studio Smart+ 1200mm BLDC Ceiling Fan with IoT &amp; Remote #shorts #shortsfeed #ceilingfan
💬 comments: Yeh bldc fan to normal fan se jyada sound karta hai
🔗 URL: https://youtube.com/watch?v=
---

🎯 Score: 0.53304094
```