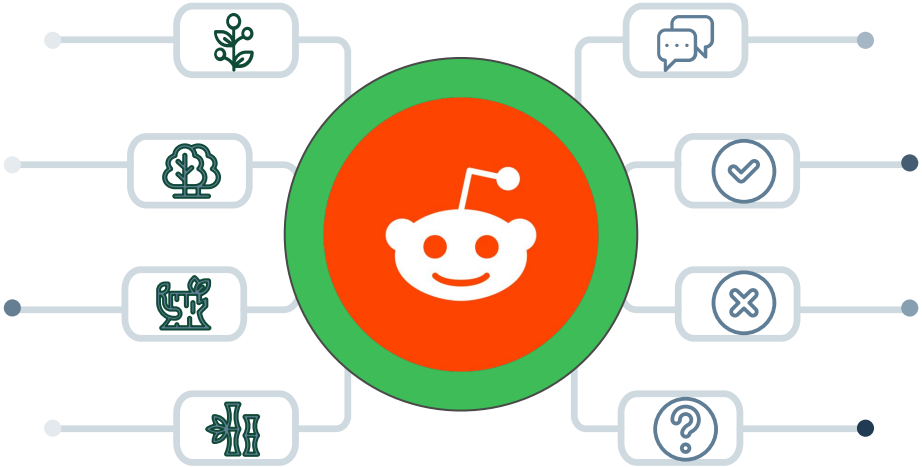
Table of Contents



Problem Overview

Determine the sentiment of Reddit comments regarding environmental policies on three different subreddits using esg_scores and the averages of the sentiment compound scores

Datasets

Environment

Article comments from environment subreddit

Climate

Article comments from climate subreddit

Sustainability

Article comments from sustainability subreddit

We used a combination of two separate datasets we found on Kaggle:

1. "🌱Greenwashing on Reddit🌍"
   - Greenwashing Articles and Reddit's Posts and Comments up to 03/2025
2. "S&P 500 Firms ESG Sustainability Reports Dataset" (preview below)

```
# Read snp_df
import matplotlib.pyplot as plt
import pandas as pd
snp_df = pd.read_csv('/work/snp_500.csv')
snp_df
```

Visualize

	Unnamed: 0 int64 0 - 865	filename object ASX_BSX_2... 0.1% ASX_BSX_2... 0.1% 864 others 99.8%	ticker object ABT 0.6% AAPL 0.5% 261 others 99%	year int64 2014 - 2023	e_score float64 0.01 - 24.84	s_score float64 1.28 - 26.87	g_score float64 3.16 - 16.47	total_score float64 6.99 - 46.51
0	0	ASX_BSX_2020.pdf	BSX	2020	3.16	18	11.83	32.98
1	1	ASX_BSX_2022.pdf	BSX	2022	2.83	12.86	10.32	26.02
2	2	ASX_EXR_2022.pdf	EXR	2022	3.81	4.28	5.86	13.94
3	3	LSE_ADM_2019.pdf	ADM	2019	16.38	14.2	5.9	36.36
4	4	LSE_ADM_2020.p...	ADM	2020	15.89	13.51	5.38	34.78
5	5	LSE_ADM_2021.pdf	ADM	2021	15.91	13.49	4.85	34.25
6	6	LSE_BBY_2020.pdf	BBY	2020	2.11	5.2	4.6	11.92
7	7	NASDAQ_AAPL_2...	AAPL	2019	0.55	13.58	10.47	24.32
8	8	NASDAQ_AAPL_2...	AAPL	2021	0.18	7.69	8.86	16.72
9	9	NASDAQ_AAPL_2...	AAPL	2022	0.65	6.86	8.95	16.45

Cleaning Datasets

Removing stop words

a. Stop words csv
b. Adding custom stop words

stopwords.txt

```
x
y
your
yours
yourself
yourselves
you
yond
yonder
yon
ye
yet
z
zillion
j
u
umpteen
usually
us
username
uponed
upons
uponing
```

```python
# snp_df Word Cloud

from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

# Environment Subreddit Articles Comments Word Cloud

comments_df = pd.read_csv('/work/reddit_folders/environmentSubreddit/Comments/environmentArticlesComments.cs
comments = ' '.join(comments_df['Text'].astype(str))

#stopwords
stopwords_df = pd.read_csv('/work/stopwords.txt', header = None)
custom_stopwords = STOPWORDS.union(set(stopwords_df[0].astype(str).str.lower()))
commentscloud = WordCloud(
    width=800,
    height=400,
    background_color='white',
    colormap='viridis',
    max_words=100,
    stopwords=custom_stopwords
).generate(comments)
plt.figure(figsize=(15, 8))
plt.imshow(commentscloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Environment Subreddit Articles Comments', fontsize=20)
plt.tight_layout()
plt.show()
```

Visualizations

a. Word Clouds☁️

b. Word Frequencies📊

c. Scatter Plots on relationship between ESG scores📈

Word Clouds

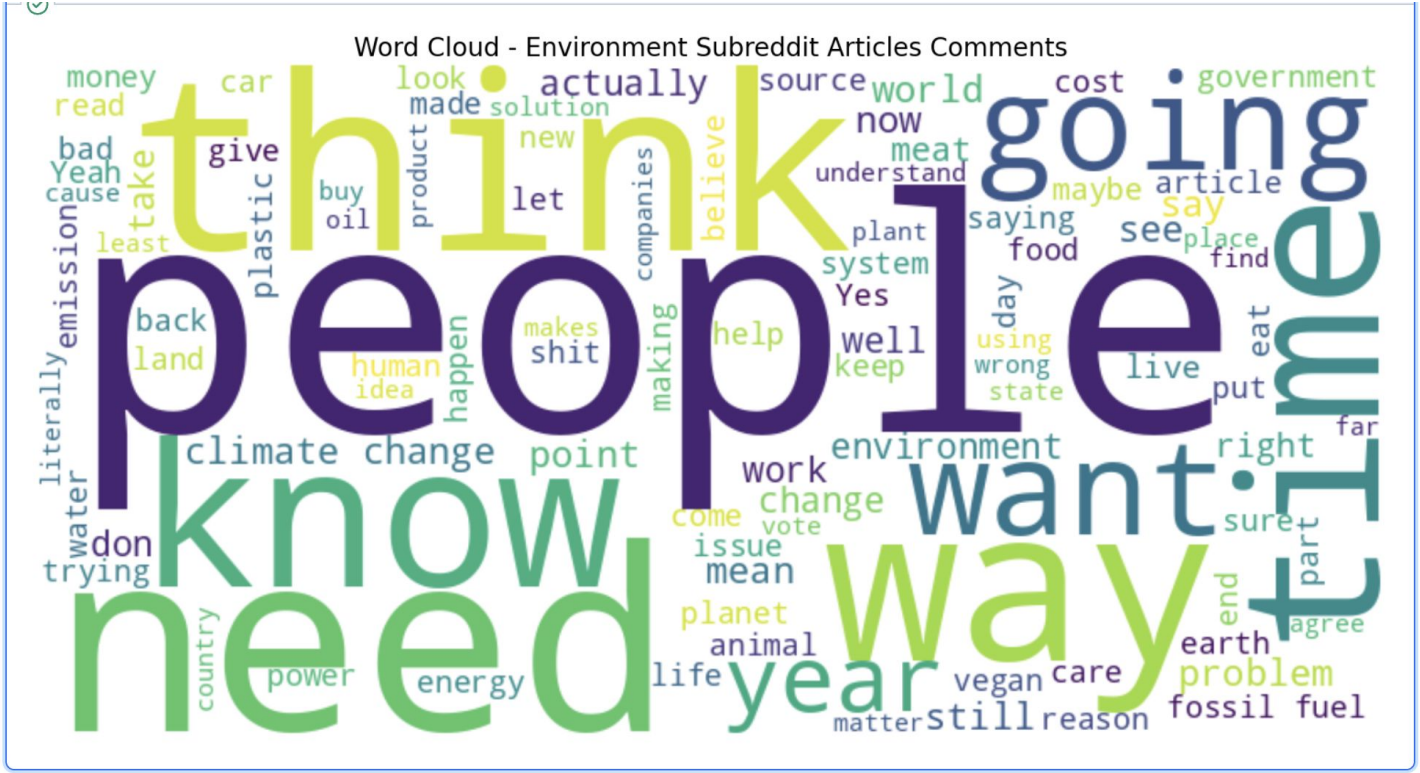
a. Environment Subreddit Comment Word Cloud
b. Climate Subreddit Comment Word Cloud
c. Sustainability Subreddit Comment Word Cloud
d. S&P 500 ESG Corporate Reports Word Cloud

Goal: Visually compare how the public talks about sustainability versus how companies officially describe them.

Environment Subreddit Comment Word Cloud

```
# snp_df Word Cloud

from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

# Environment Subreddit Articles Comments Word Cloud

comments_df = pd.read_csv('/work/reddit_folders/environmentSubreddit/Comments/environmentArticlesComments.csv')
comments = ' '.join(comments_df['Text'].astype(str))

#stopwords
stopwords_df = pd.read_csv('/work/stopwords.txt', header = None)
custom_stopwords = STOPWORDS.union(set(stopwords_df[0].astype(str).str.lower()))
commentscloud = WordCloud(
    width=800,
    height=400,
    background_color='white',
    colormap='viridis',
    max_words=100,
    stopwords=custom_stopwords
).generate(comments)
plt.figure(figsize=(15, 8))
plt.imshow(commentscloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Environment Subreddit Articles Comments', fontsize=20)
plt.tight_layout()
plt.show()
```

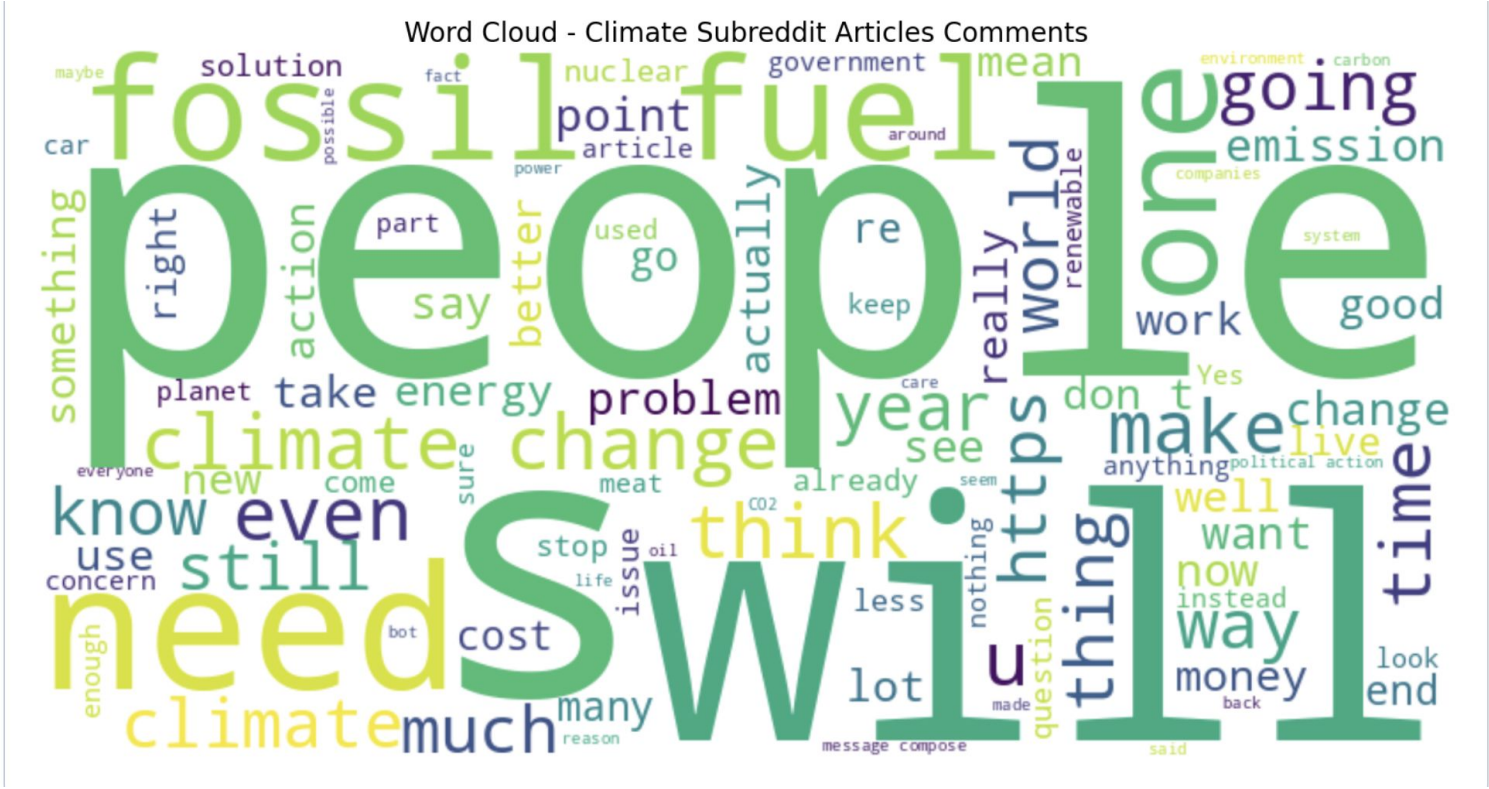
Environment Subreddit Comment Word Cloud



Climate Subreddit Comment Word Cloud

```
# Climate Subreddit Articles Comments Word Cloud

climateComments_df = pd.read_csv('/work/reddit_folders/climateSubreddit/Comments/climateArticlesComments.csv')
climateComments = ' '.join(climateComments_df['Text'].astype(str))
climateCommentsCloud = WordCloud(width=800, height=400,
                                 background_color='white',
                                 colormap='viridis',
                                 max_words=100).generate(climateComments)
plt.figure(figsize=(15, 8))
plt.imshow(climateCommentsCloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Climate Subreddit Articles Comments', fontsize=20)
plt.tight_layout()
plt.show()
```

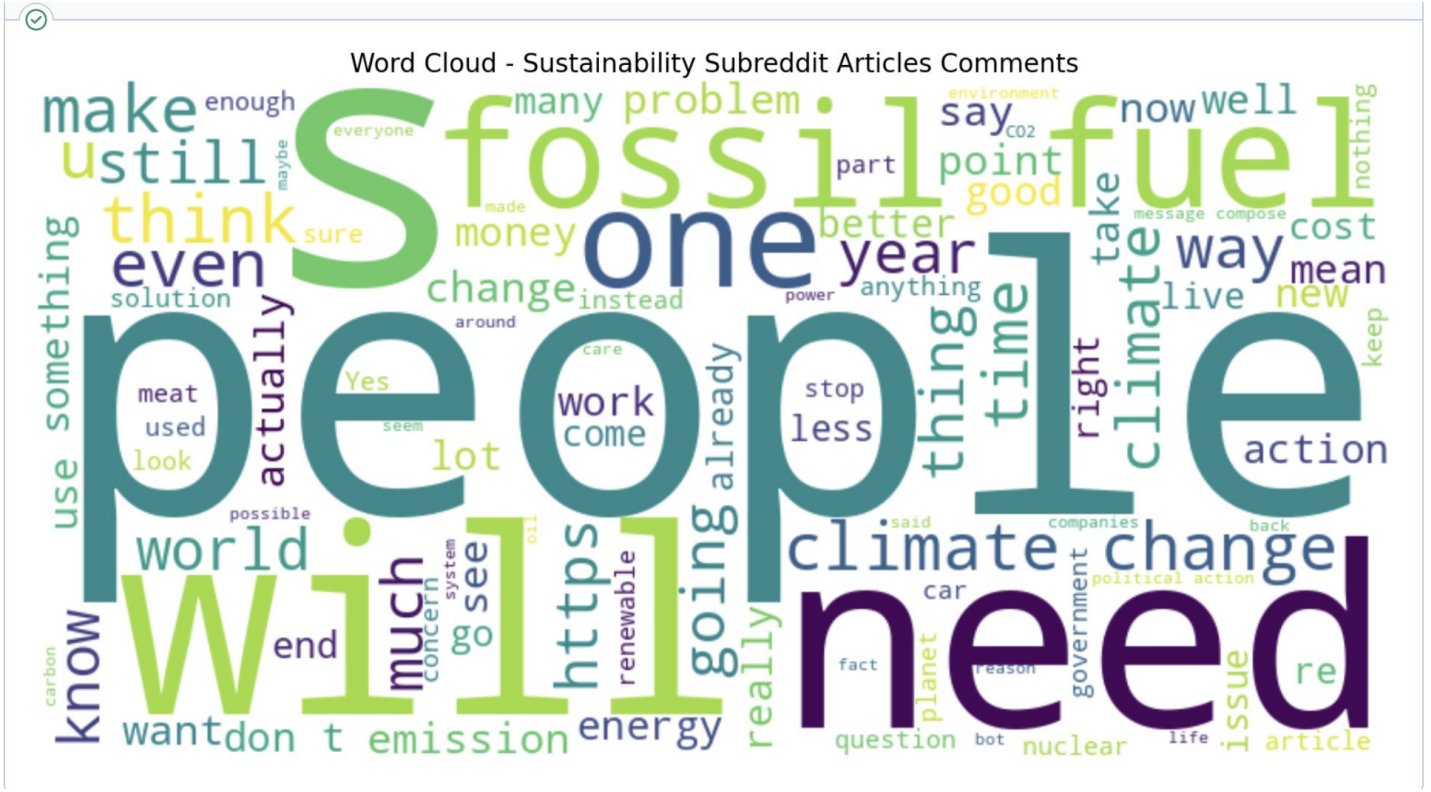
Climate Subreddit Comment Word Cloud



Sustainability Subreddit Comment Word Cloud

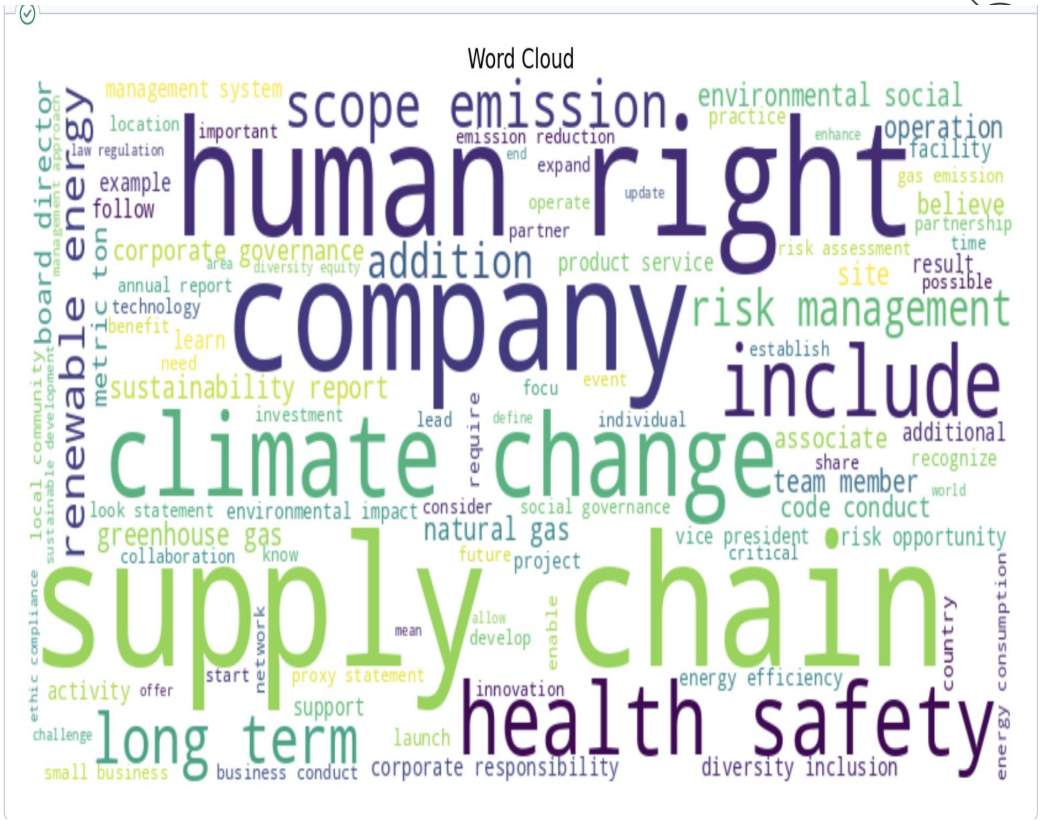
```
# Sustainability Subreddit Articles Comments Word Cloud

sustainabilityComments_df = pd.read_csv('/work/reddit_folders/sustainabilitySubreddit/Comments/sustainabilityArticlesCo
sustainabilityComments = ' '.join(climateComments_df['Text'].astype(str))
sustainabilityCommentsCloud = WordCloud(width=800, height=400,
                                        background_color='white',
                                        colormap='viridis',
                                        max_words=100).generate(sustainabilityComments)
plt.figure(figsize=(15, 8))
plt.imshow(sustainabilityCommentsCloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Sustainability Subreddit Articles Comments', fontsize=20)
plt.tight_layout()
plt.show()
```

Sustainability Subreddit Comment Word Cloud



S&P Corporate Reports Word Cloud

```
1
2  # Combine all text from the column into one string
3  text = ' '.join(snp_df['preprocessed_content'].astype(str))
4
5  # Create the word cloud
6  wordcloud = WordCloud(stopwords=custom_stopwords,
7                        width=800, height=400,
8                        background_color='white',
9                        colormap='viridis',
10                       max_words=100).generate(text)
11 # Display the word cloud
12 plt.figure(figsize=(15, 8))
13 plt.imshow(wordcloud, interpolation='bilinear')
14 plt.axis('off')
15 plt.title('Word Cloud', fontsize=20)
16 plt.tight_layout()
17 plt.show()
```



Word Frequency – Process

- Extracted all article text from Climate subreddit using custom stopwords to filter out noise

```
1  climate_change = pd.read_csv('reddit_folders/climatechangeSubreddit/Articles/climatechangeArticles.csv')
2  climate_change.head()
```

	Article_Id int64	Article_Url object	Post_id object	Article_Title object	Article_Content o...
0	1	https://www.eur...	e74s78	'Do no harm': Nuc...	Euractiv +For indi...
1	2	https://insidecli...	e83dht	Lawyers Challeng...	Lawyers Challeng...
2	3	https://www.cbc....	ehe1pu	New industry dev...	When search sug...
3	4	https://medium.c...	gh21y2	Microsoft's 'Carb...	Microsoft's 'Carb...
4	5	https://2030.buil...	jvsc81	5 Ways to Avoid ...	Greenwashing: W...

5 rows, 5 cols 10 / page Page 1 of 1 Form

```
1  climate_change["Article_Content"][0]
```

```
"Euractiv +For individualsEuractiv ProFor corporationsLooking to access paid articles across multiple policy topics?Int
```

Word Frequency – Process

```
STOPWORDS = {
    'the','a','an','and','or','to','of','in','is','it','for','on','that','this','with','as',
    'are','was','be','by','from','at','if','but','so','not','you','i','we','they','he','she'
}

word_counts = {}

for article in climate_change["Article_Content"]:
    words = re.findall(r"[a-z]+", article.lower())
    #words = article.lower().split()  # lowercase + split on whitespace
    for word in words:
        if word in STOPWORDS:
            continue
        word_counts[word] = word_counts.get(word, 0) + 1

# This sorts our dictionary based off the corresponding frequency, and turns it into a list.
word_counts_show = sorted(word_counts.items(), key=lambda item: item[1], reverse = True)
word_counts_show[:10]  # Display the top 20 most frequent words
```

```
[('s', 589),
 ('climate', 399),
 ('emissions', 335),
 ('have', 282),
 ('more', 269),
 ('carbon', 266),
 ('co', 258),
 ('has', 227),
 ('energy', 224),
 ('can', 209)]
```

- Counted word frequencies by
  - Extract alphabetical tokens
  - Remove stopwords
  - Aggregate their occurrence across all articles

Word Frequency – Process

```python
def wordDictionary(df, columnName):
    STOPWORDS = {'the','a','an','and','or','to','of','in','is','it','for','on','that','this','with','as',
    'are','was','be','by','from','at','if','but','so','not','you','i','we','they','he','she'}
    word_counts = {}
    for article in df[columnName]:
        words = re.findall(r"[a-z]+", article.lower())
        #words = article.lower().split()  # lowercase + split on whitespace
        for word in words:
            if word in STOPWORDS:
                continue
            word_counts[word] = word_counts.get(word, 0) + 1
    return word_counts
```

- Extract tokens, remove stopwords, and compute word frequencies

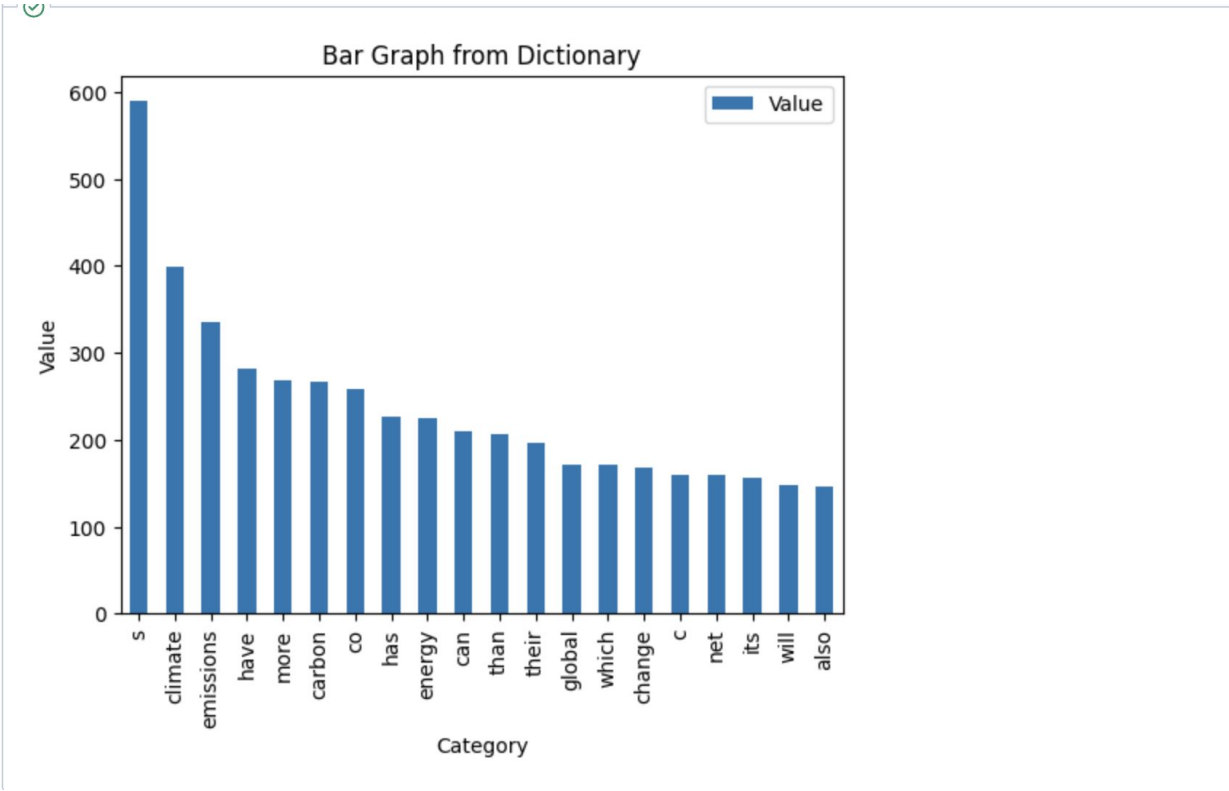
```python
df = pd.DataFrame.from_dict(word_counts, orient='index', columns=['Value'])
df = df.sort_values(by=['Value'], ascending=False)
df.head(5)
df = df[:20]

```

- Convert the frequency into a dataframe

```python
df.plot(kind='bar')
plt.title('Bar Graph from Dictionary')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```

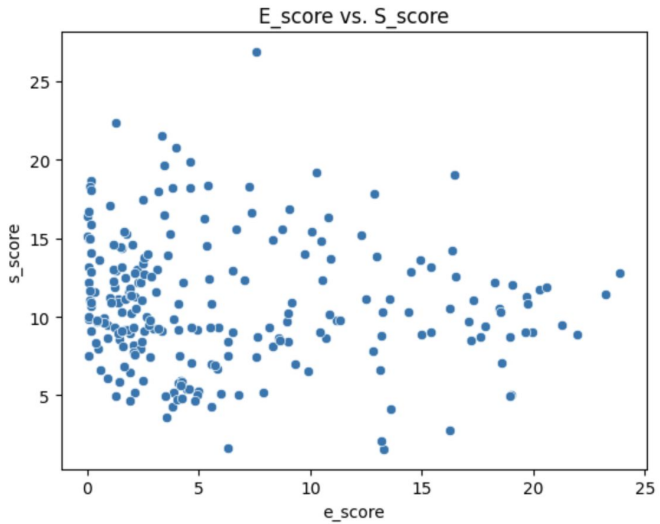
- Visualize the top word frequency with a bar chart

Word Frequency – Result

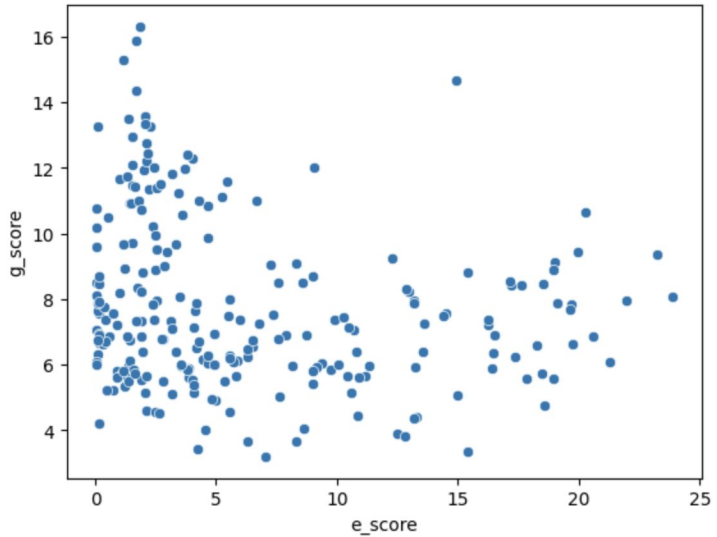


Scatter Plots – Relationship between ESG Scores

```python
import seaborn as sns

sns.scatterplot(x="e_score", y="s_score", data=snp_grouped)
plt.title("E_score vs. S_score")
plt.show()
```

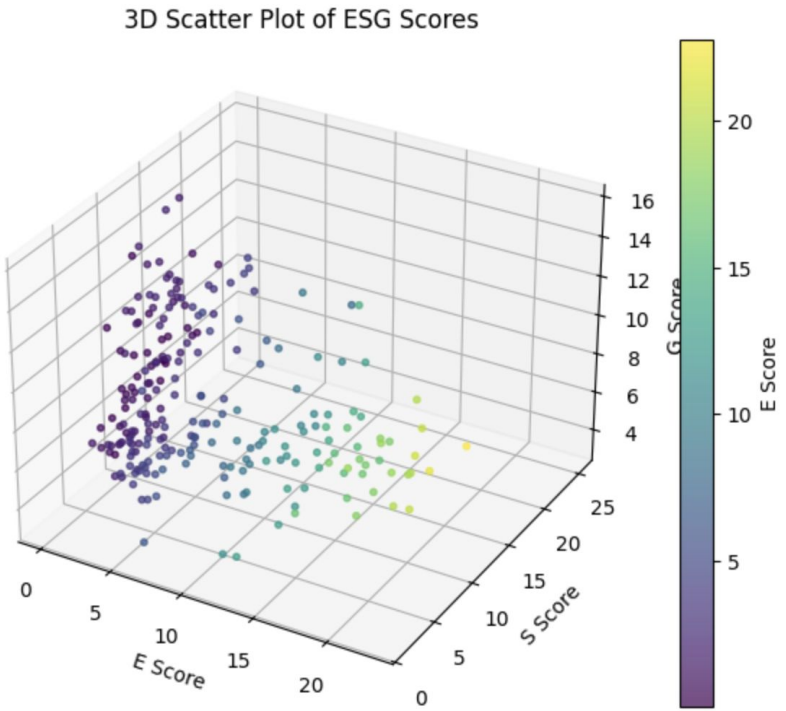


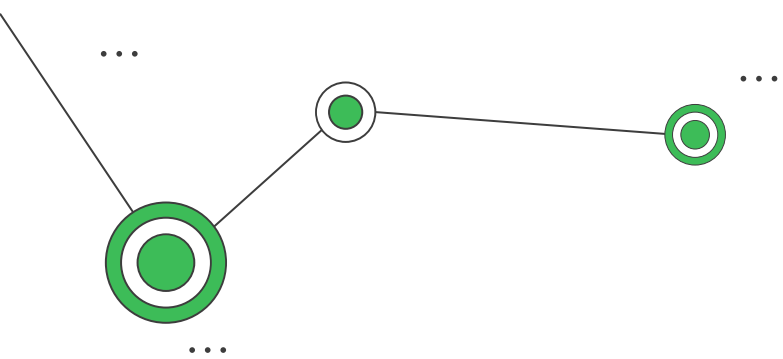

```python
import matplotlib.pyplot as plt
import seaborn as sns

sns.scatterplot(x="e_score", y="g_score", data=snp_grouped)
plt.show()
```



3D Scatter Plot

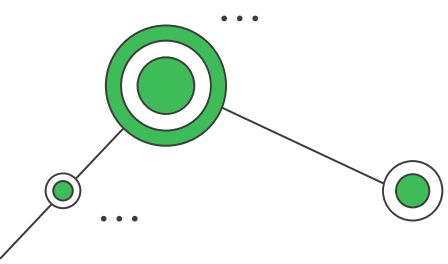
```python
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

sc = ax.scatter(
    avg['e_score'],
    avg['s_score'],
    avg['g_score'],
    c=avg['e_score'],
    cmap='viridis',
    s = 10, alpha = 0.7
)
ax.set_xlabel('E Score')
ax.set_ylabel('S Score')
ax.set_zlabel('G Score')
ax.set_title('3D Scatter Plot of ESG Scores')
fig.colorbar(sc, ax=ax, label = 'E Score')
plt.show()
```

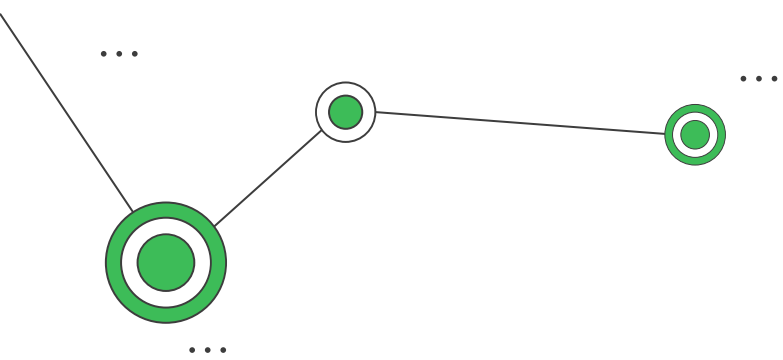




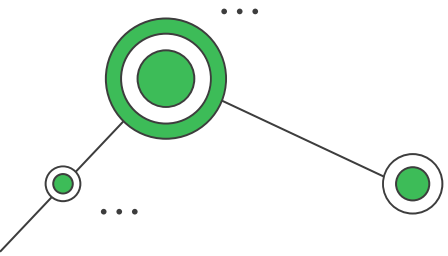
Categorizing Reddit comments into sectors🍰

- To gain structured insights
- To learn what industries people care about the most

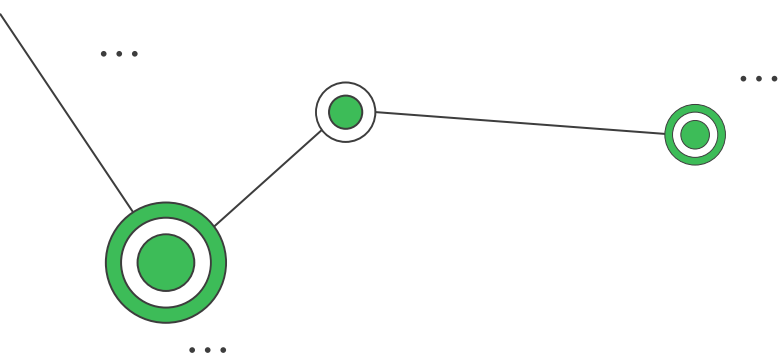




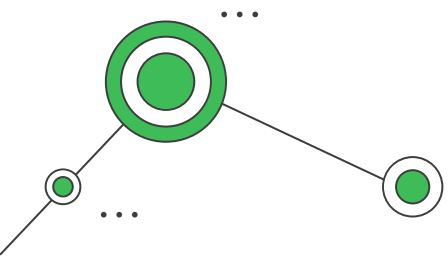
Creating lookup tables linking sector keywords to S&P 500 sectors



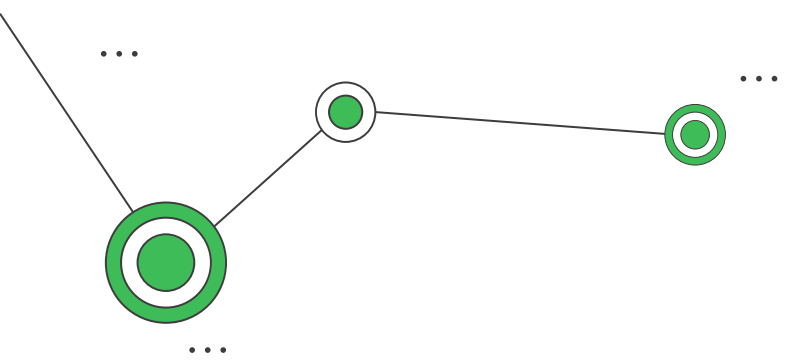
```python
def extract_sector_info(sp500_df):
    """
    Build lookups for sector keywords and map companies to sectors.
    Similar to extract_company_info but for sectors.

    Returns:
    - sector_keywords: dict mapping keywords to sectors
    - sector_lookup: dict mapping tickers to sectors (for reference)
    """
    # Define comprehensive sector-specific keywords
    sector_keyword_map = {
        'Technology': [
            'tech', 'software', 'cloud', 'ai', 'artificial intelligence',
            'semiconductor', 'chip', 'chips', 'data center', 'cybersecurity', 'saas',
            'hardware', 'computing', 'digital', 'platform', 'app', 'apps',
            'mobile', 'internet', 'web', 'coding', 'programming'
        ],
        'Health Care': [
            'pharma', 'pharmaceutical', 'biotech', 'healthcare', 'health care',
            'medical', 'drug', 'drugs', 'medicine', 'hospital', 'clinical',
            'vaccine', 'vaccines', 'therapy', 'patient', 'device', 'diagnostic',
            'treatment', 'doctor', 'healthcare provider'
        ],
        'Financials': [
            'bank', 'banking', 'financial', 'finance', 'insurance', 'credit',
            'loan', 'loans', 'mortgage', 'investment', 'asset management',
            'wealth', 'broker', 'brokerage', 'lending', 'fintech',
            'payment', 'payments', 'credit card'
        ],
        'Consumer Discretionary': [
            'retail', 'ecommerce', 'e-commerce', 'consumer', 'shopping', 'store',
            'restaurant', 'restaurants', 'auto', 'automotive', 'car', 'cars',
            'hotel', 'hotels', 'travel', 'entertainment', 'luxury', 'apparel',
            'clothing', 'fashion', 'furniture', 'home improvement'
        ],
        'Communication Services': [
            'telecom', 'telecommunications', 'media', 'streaming', 'content',
            'advertising', 'social media', 'network', 'broadcast', 'cable',
            'wireless', '5g', 'internet service', 'video streaming'
        ],
        'Industrials': [
            'industrial', 'manufacturing', 'aerospace', 'defense',
            'construction', 'machinery', 'transportation', 'logistics',
            'shipping', 'railroad', 'airline', 'airlines', 'freight'
        ],
```



Creating sector lookup dictionaries

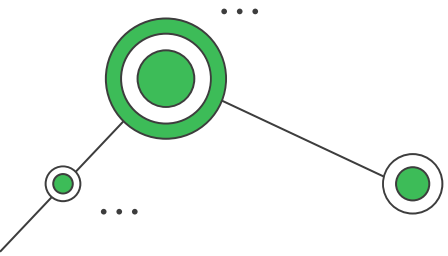


```
    ],
    'Consumer Staples': [
        'food', 'beverage', 'grocery', 'household', 'staples',
        'packaged goods', 'tobacco', 'personal care', 'consumer goods',
        'snacks', 'drinks'
    ],
    'Energy': [
        'oil', 'gas', 'energy', 'renewable', 'solar', 'wind',
        'petroleum', 'drilling', 'refining', 'electric utility',
        'fossil fuel', 'clean energy', 'crude'
    ],
    'Utilities': [
        'utility', 'utilities', 'electric', 'electricity', 'power',
        'water', 'gas utility', 'power grid'
    ],
    'Real Estate': [
        'real estate', 'reit', 'reits', 'property', 'properties',
        'commercial real estate', 'residential', 'mortgage reit',
        'landlord', 'housing'
    ],
    'Materials': [
        'materials', 'mining', 'chemical', 'chemicals', 'metals',
        'steel', 'commodity', 'commodities', 'packaging', 'paper',
        'construction materials', 'aluminum', 'copper'
    ]
}

# Create reverse lookup: keyword -> sector
sector_keywords = {}
for sector, keywords in sector_keyword_map.items():
    for keyword in keywords:
        sector_keywords[keyword.lower()] = sector

# Create ticker -> sector lookup (same as before)
sector_lookup = {}
for index, row in sp500_df.iterrows():
    ticker = str(row['ticker']).upper()
    sector = row['Sector'] if ('Sector' in row and pd.notna(row['Sector'])) else 'Unknown'
    sector_lookup[ticker] = sector

return sector_keywords, sector_lookup
```



Identifying sector mentions and extracting context

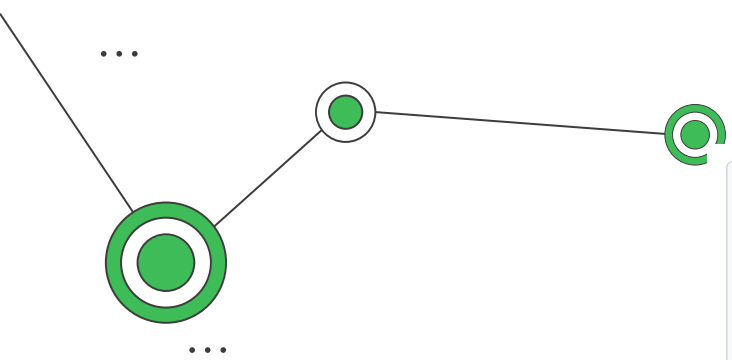
Return mentions (at the end)



```python
def find_sector_mentions_with_context(text, sector_keywords):
    """
    Find sector mentions and extract surrounding context.
    Similar to find_company_mentions_with_context but for sectors.

    Returns list of dicts with sector and its context window
    """
    if pd.isna(text):
        return []

    text = str(text)
    doc = nlp(text)
    mentions = []

    # Track which sectors we've already found to avoid duplicates
    found_sectors = {}

    # Search for each keyword in the text
    for keyword, sector in sector_keywords.items():
        pattern = r'\b' + re.escape(keyword) + r'\b'
        for match in re.finditer(pattern, text, re.IGNORECASE):
            # Skip if we already found this sector
            if sector in found_sectors:
                continue

            # Find the sentence containing this match
            for sent in doc.sents:
                if sent.start_char <= match.start() <= sent.end_char:
                    # Get surrounding sentences for better context
                    sent_index = list(doc.sents).index(sent)
                    all_sents = list(doc.sents)

                    context_start = max(0, sent_index - 1)
                    context_end = min(len(all_sents), sent_index + 2)
                    context = ' '.join([s.text for s in all_sents[context_start:context_end]])

                    mentions.append({
                        'sector': sector,
                        'mention_text': match.group(),
                        'sentence': sent.text,
                        'context': context,
                        'char_start': match.start(),
                        'char_end': match.end()
                    })

                    found_sectors[sector] = True
                    break
```

```python
def group_comments_by_sector(comments_df, sp500_df, keep_single_sector_only=False):
    """
    Main function to group Reddit comments by sector.
    Similar to group_comments_by_company but for sectors.

    Parameters:
    - comments_df: DataFrame with Reddit comments
    - sp500_df: DataFrame with S&P 500 company data
    - keep_single_sector_only: If True, only keeps comments mentioning exactly 1 sector

    Returns:
    - enriched_comments: DataFrame with sector info added to each comment
    - sector_groups: Dictionary with sectors as keys, lists of comments as values
    """
    # Extract sector information
    sector_keywords, sector_lookup = extract_sector_info(sp500_df)

    # Extract sector mentions with context
    comments_df['sector_mentions'] = comments_df['Text'].apply(
        lambda x: find_sector_mentions_with_context(x, sector_keywords)
    )

    # Expand to one row per sector mention
    expanded_rows = []

    for index, row in comments_df.iterrows():
        mentions = row['sector_mentions']

        if keep_single_sector_only and len(mentions) != 1:
            continue  # Skip comments with 0 or multiple sectors

        if len(mentions) == 0:
            continue  # Skip comments with no sector mentions

        # Create a row for each sector mentioned
        for mention in mentions:
            new_row = row.copy()
            new_row['mentioned_sector'] = mention['sector']
            new_row['sector_mention_text'] = mention['mention_text']
            new_row['sector_context'] = mention['context']
            new_row['sector_sentence'] = mention['sentence']
            new_row['is_multi_sector'] = len(mentions) > 1
            expanded_rows.append(new_row)

    expanded_df = pd.DataFrame(expanded_rows)

    # Create grouped datasets by sector
    sector_groups = defaultdict(list)
    for index, row in expanded_df.iterrows():
        sector = row['mentioned_sector']
        sector_groups[sector].append({
            'comment_id': row.get('Parent_id', index),
            'text': row['Text'],
            'author': row.get('Author', ''),
            'score': row.get('Score', 0),
            'timestamp': row.get('Timestamp', ''),
            'sector': row['mentioned_sector'],
            'sector_context': row['sector_context'],
            'is_multi_sector': row['is_multi_sector']
        })

    return expanded_df, sector_groups


def create_sector_summary(sector_groups):
    """
    Create a summary of comments by sector.
    Similar to create_company_summary but for sectors.
    """
    summary = []
    for sector, comments in sector_groups.items():
        summary.append({
            'Sector': sector,
            'Comment_Count': len(comments),
            'Single_Sector_Comments': sum(1 for comment in comments if not comment['is_multi_sector']),
            'Multi_Sector_Comments': sum(1 for comment in comments if comment['is_multi_sector']),
            'Total_Score': sum(comment['score'] for comment in comments),
            'Avg_Score': sum(comment['score'] for comment in comments) / len(comments) if comments else 0,
            'Unique_Authors': len(set(comment['author'] for comment in comments))
        })

    return pd.DataFrame(summary).sort_values('Comment_Count', ascending=False)
```

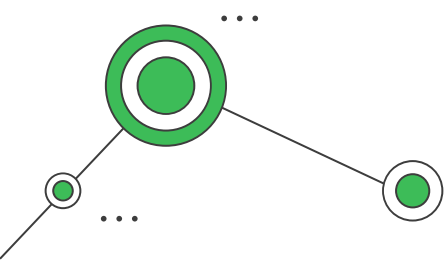
Grouping comments by sector and create sector-level summaries



Applying sector grouping to all Reddit datasets

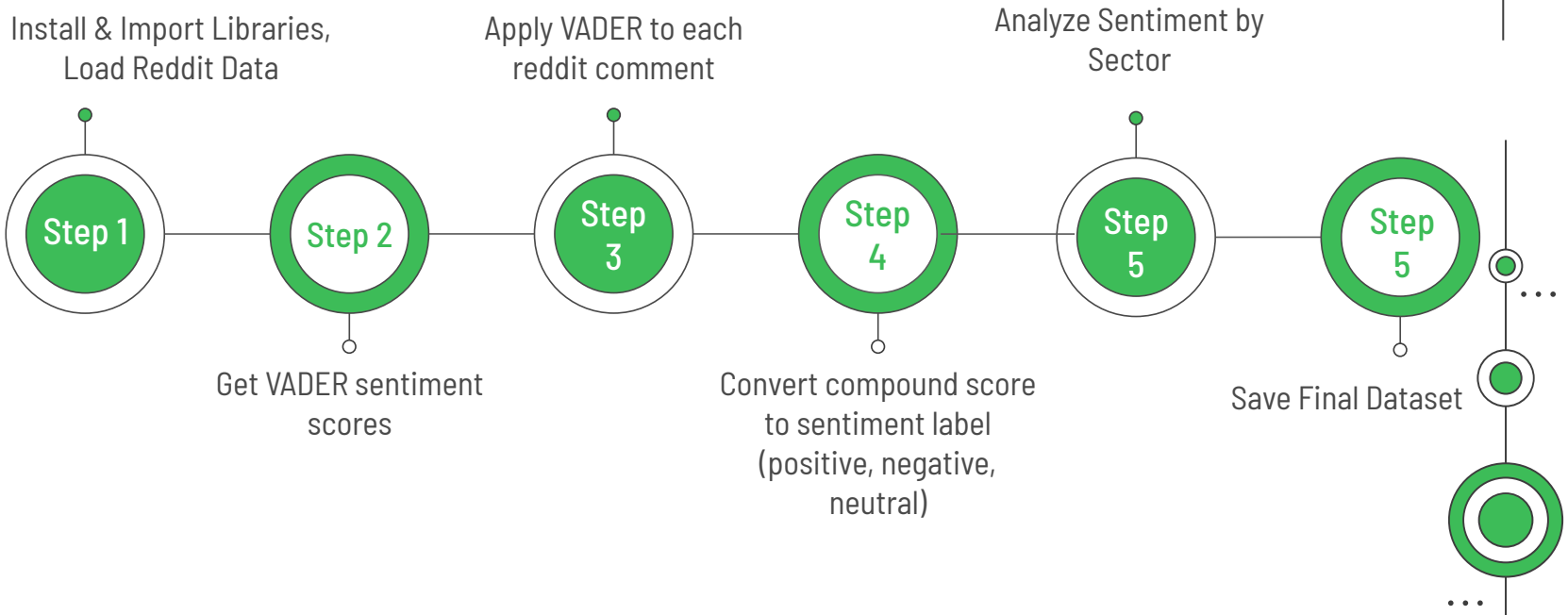
```
#TODO call the function here on all of the dataframes we have, try to call it on the regular dataframe (so not the sam
#if it takes way too long then call the function on the sampled version
#make sure to store these new tables into different variables we can use later on
sp500_df = pd.read_csv("/work/snp_500.csv")
envCommentsDF_grouped = group_comments_by_sector(
    envCommentsDF, sp500_df, keep_single_sector_only=False
)[0]
envCommentsDF_grouped.to_csv("envCommentsDF_grouped.csv", index=False)

```

```
#TODO call the function here on all of the dataframes we have, try to call it on the regular dataframe (so not the sam
#if it takes way too long then call the function on the sampled version
#make sure to store these new tables into different variables we can use later on
sp500_df = pd.read_csv("/work/snp_500.csv")
climateCommentsDF_grouped = group_comments_by_sector(
    climateCommentsDF, sp500_df, keep_single_sector_only=False
)[0]
climateCommentsDF_grouped.to_csv("climateCommentsDF_grouped.csv", index=False)
```



```
#TODO call the function here on all of the dataframes we have, try to call it on the regular dataframe (so not the sam
#if it takes way too long then call the function on the sampled version
#make sure to store these new tables into different variables we can use later on
sp500_df = pd.read_csv("/work/snp_500.csv")
sustainabilityCommentsDF_grouped = group_comments_by_sector(
    sustainabilityCommentsDF_sampled, sp500_df, keep_single_sector_only=False
)[0]
sustainabilityCommentsDF_grouped.to_csv("sustainabilityCommentsDF_grouped.csv", index=False)
sustainabilityCommentsDF_grouped
```

Six Step Sentiment Analysis Process (VADER)



1. Install and Import Libraries + Load Reddit Data

```
!pip install nltk pandas

import pandas as pd
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

# Download the VADER lexicon (run once)
nltk.download("vader_lexicon")

```

Outputs...

```
Requirement already satisfied: nltk in /root/venv/lib/python3.10/site-packages (3.7)
Requirement already satisfied: pandas in /root/venv/lib/python3.10/site-packages (2.1.4)
Requirement already satisfied: tqdm in /root/venv/lib/python3.10/site-packages (from nltk) (4.67.1)
Requirement already satisfied: joblib in /root/venv/lib/python3.10/site-packages (from nltk) (1.5.2)
Requirement already satisfied: regex>=2021.8.3 in /root/venv/lib/python3.10/site-packages (from nltk) (2025.11.3)
Requirement already satisfied: click in /root/venv/lib/python3.10/site-packages (from nltk) (8.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /root/venv/lib/python3.10/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /root/venv/lib/python3.10/site-packages (from pandas) (2025.2)
Requirement already satisfied: numpy<2,>=1.22.4 in /root/venv/lib/python3.10/site-packages (from pandas) (1.25.2)
Requirement already satisfied: tzdata>=2022.1 in /root/venv/lib/python3.10/site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /root/venv/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

[notice] A new release of pip is available: 23.0.1 -> 25.3
[notice] To update, run: pip install --upgrade pip
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...

True
```

```
# Load in Reddit dataframe
climateCommentsDF_grouped = pd.read_csv('/work/climateCommentsDF_grouped.csv')
climateCommentsDF_grouped.head()
```

Visualize

	Unnamed: 0 int64	Type object	Comment_id obj...	Parent_id object	Post_id object	Author object	Timestamp object
0	66	Comment	d4ivv4y	t3_4p7lz9	4p7lz9	Lighting	22/06/2016 2:31
1	66	Comment	d4ivv4y	t3_4p7lz9	4p7lz9	Lighting	22/06/2016 2:31
2	73	Comment	d4j01od	t3_4p7lz9	4p7lz9	mulgs	22/06/2016 4:31
3	74	Comment	d4j0h2w	t3_4p7lz9	4p7lz9	Tommy27	22/06/2016 4:46
4	76	Comment	d4j3z1f	t3_4p7lz9	4p7lz9	Eco-nome	22/06/2016 7:13

5 rows, 17 cols 10 / page Page 1 of 1 Format

```
# Install VADER libraries

# Download lexicon once
nltk.download("vader_lexicon")

# Initialize VADER
sia = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

2. Get VADER sentiment scores

```python
# Function that returns all VADER scores
def apply_vader(text):
    if pd.isna(text):
        text = ""
    text = str(text)
    scores = sia.polarity_scores(text)
    return pd.Series({
        "Negative Score": scores["neg"],
        "Neutral Score": scores["neu"],
        "Positive Score": scores["pos"],
        "Compound Score": scores["compound"]
    })

```

3. Apply VADER to each reddit comment

```
# Add vader sentiment analysis
climateCommentsDF_grouped[["Negative Score", "Neutral Score", "Positive Score", "Compound Score"]] = (
    climateCommentsDF_grouped["Text"].apply(apply_vader)
)
```

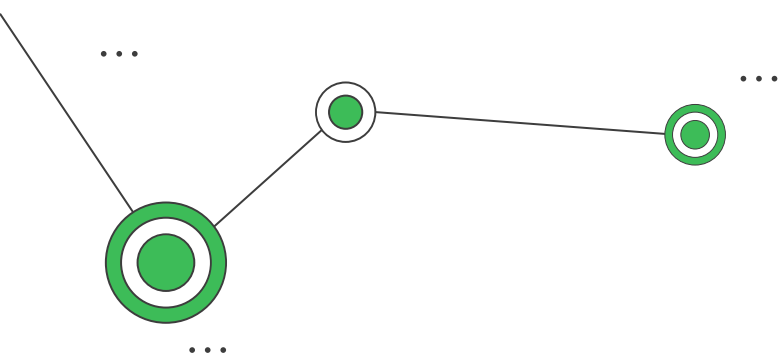
4. Convert compound score to sentiment label (positive, negative, neutral)

```
# convert compound score → discrete sentiment
def label_sentiment(compound):
    if compound >= 0.05:
        return "positive"
    elif compound <= -0.05:
        return "negative"
    else:
        return "neutral"

climateCommentsDF_grouped["vader_sentiment"] = climateCommentsDF_grouped["Compound Score"].apply(label_sentiment)
```

5. Analyze Sentiment by Sector

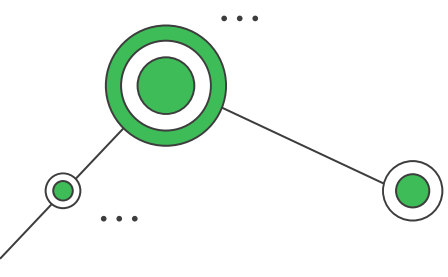
```
sector_sentiment_climate = (
    climateCommentsDF_grouped
    .groupby("mentioned_sector")["Compound Score"]
    .mean()
    .sort_values()
)

sector_sentiment_climate
```

```
mentioned_sector
Real Estate               0.029970
Industrials               0.137623
Consumer Discretionary    0.158428
Consumer Staples          0.183451
Communication Services    0.211088
Utilities                 0.236732
Materials                 0.267280
Technology                0.272308
Financials                0.313255
Energy                    0.328844
Health Care               0.510546
Name: Compound Score, dtype: float64
```



6. Save Final Dataset!

```
climateCommentsDF_grouped.to_csv("climateCommentsDF_grouped_with_sentiment.csv", index=False)
```



Repeat with other two subreddits...

Environment (envCommentsDF_grouped):

```
envCommentsDF_grouped = pd.read_csv('/work/envCommentsDF_grouped.csv')
#envCommentsDF_grouped.head()
envCommentsDF_grouped[["Negative Score", "Neutral Score", "Positive Score", "Compound Score"]] = (
    envCommentsDF_grouped["Text"].apply(apply_vader)
)
envCommentsDF_grouped["vader_sentiment"] = envCommentsDF_grouped["Compound Score"].apply(label_sentiment)

sector_sentiment_env = (
    envCommentsDF_grouped
    .groupby("mentioned_sector")["Compound Score"]
    .mean()
    .sort_values()
)

envCommentsDF_grouped.to_csv("envCommentsDF_grouped_with_sentiment.csv", index=False)
```

```
sector_sentiment_env
```

```
mentioned_sector
Industrials                0.113069
Consumer Discretionary     0.169247
Consumer Staples           0.195384
Real Estate                0.202945
Communication Services     0.232625
Health Care                0.251750
Utilities                  0.282159
Energy                     0.292724
Materials                  0.306743
Technology                 0.391694
Financials                 0.413507
Name: Compound Score, dtype: float64
```

Sustainability (sustainabilityCommentsDF_grouped)

```
sustainabilityCommentsDF_grouped = pd.read_csv('/work/sustainabilityCommentsDF_grouped.csv')
sustainabilityCommentsDF_grouped[["Negative Score", "Neutral Score", "Positive Score", "Compound Score"]] = (
    sustainabilityCommentsDF_grouped["Text"].apply(apply_vader)
)
sustainabilityCommentsDF_grouped["vader_sentiment"] = sustainabilityCommentsDF_grouped["Compound Score"].apply(label_s

sector_sentiment_sustainability = (
    sustainabilityCommentsDF_grouped
    .groupby("mentioned_sector")["Compound Score"]
    .mean()
    .sort_values()
)
sector_sentiment_sustainability
sustainabilityCommentsDF_grouped.to_csv("sustainabilityCommentsDF_grouped_with_sentiment.csv", index=False)
```

```
sector_sentiment_sustainability
```

```
mentioned_sector
Financials                 0.296590
Technology                 0.316769
Consumer Staples           0.334345
Energy                     0.381421
Materials                  0.386750
Utilities                  0.394625
Consumer Discretionary     0.412874
Industrials                0.498292
Communication Services     0.655223
Health Care                0.700940
Real Estate                0.711938
Name: Compound Score, dtype: float64
```

Results

Cross-dataset Analysis

Most of the sentiments are positive.
Sustainability has the highest positivity.

Challenges

Data Cleaning

Company-level matching is PROBLEMATIC:
1. Company names
2. Running time is fairly long

Insight 1. In the Climate dataset, sentiment is positive overall, but varies sharply across sectors

```
1  sector_sentiment_climate = (
2      climateCommentsDF_grouped
3      .groupby("mentioned_sector")["Compound Score"]
4      .mean()
5      .sort_values()
6  )
7
8  sector_sentiment_climate
```

```
mentioned_sector
Real Estate               0.029970
Industrials               0.137623
Consumer Discretionary    0.158428
Consumer Staples          0.183451
Communication Services    0.211088
Utilities                 0.236732
Materials                 0.267280
Technology                0.272308
Financials                0.313255
Energy                    0.328844
Health Care               0.510546
Name: Compound Score, dtype: float64
```

Insight 2. In the Environment dataset, Technology and Financials remain positive, while Industrials are viewed less favorably

```
1  sector_sentiment_env
```

mentioned_sector	
Industrials	0.113069
Consumer Discretionary	0.169247
Consumer Staples	0.195384
Real Estate	0.202945
Communication Services	0.232625
Health Care	0.251750
Utilities	0.282159
Energy	0.292724
Materials	0.306743
Technology	0.391694
Financials	0.413507

Name: Compound Score, dtype: float64

Insight 3. In the Sustainability dataset, sentiment becomes overwhelmingly positive

```
1  sector_sentiment_sustainability
```

mentioned_sector	
Financials	0.296590
Technology	0.316769
Consumer Staples	0.334345
Energy	0.381421
Materials	0.386750
Utilities	0.394625
Consumer Discretionary	0.412874
Industrials	0.498292
Communication Services	0.655223
Health Care	0.700940
Real Estate	0.711938

Name: Compound Score, dtype: float64

Insight 4. Recurring Trends & Interpretation

a. Technology, Materials, and Utilities consistently score positively across climate, environment, and sustainability topics.
→ progress and innovation.

b. Industrials and Real Estate repeatedly appear on the lower end for climate/environment conversations.
→ responsible for emissions or environmental risk.

Thanks for Listening!