ChatGPT App Reviews: User Sentiment and Feature Insights

Team 5 -
Shuting Zhang
Tanvi Dinesh Nandu
William Wang
Xiaoyi Wang
Xiaoyang Li

Table of Contents

Executive Summary

Objective: To analyze user reviews of the ChatGPT app from the Google Play Store uncovering insights into user sentiment, app performance across multiple app versions and feature feedback to evaluate the app's updates and user engagement. The key findings are:

- **Sentiment Trends:** User sentiment fluctuates across app updates, highlighting the need for targeted improvements to address recurring issues.
- **Feature Insights:** Performance, response quality, and user interface are frequently discussed in both positive and negative reviews.
- **Prompt Usability:** Users express varied satisfaction with the app's prompt handling, indicating opportunities for refining user interactions.

Impact: Provide actionable insights to improve user experience, guide app updates, and enhance user satisfaction by addressing pain points and optimizing key features.

About the Dataset

Source: Kaggle dataset - reviews from the Google Play Store for the ChatGPT app.

Dataset Features: user ratings, textual reviews, date of review, country, thumbs-up count, and app version. This dataset offers a diverse range of user feedback from different countries and app versions.

Length: 2,78,337 rows of reviews

Primary Use Cases

- **Sentiment Analysis:** Evaluate user sentiment to gauge satisfaction and identify improvement areas.
- **Natural Language Processing:** Apply NLP techniques for text classification, summarization, and keyword extraction.
- **Trend Analysis:** Monitor changes in user feedback over time and across different app versions.

Source Data Overview

This dataset consists of 270k+ daily-updated user reviews and ratings for the ChatGPT Android App.

reviewId id	userName reviewer	content review	score rating	thumbsUpCount likes	reviewCreatedV... app version	at date	appVersion app version
278337 unique values	**255900** unique values	good 5% nice 2% Other (265335) 94%	1 5	0 1193	[null] 9% 1.2024.268 6% Other (242288) 86%	2023-07-25 2024-11-28	[null] 9% 1.2024.268 6% Other (242288) 86%
42ad3464-5158-485f-a745-51058db2bdc3	Pavia Akter	that's amazing ,its help me ,and i am happy to used it	5	0	1.2024.324	2024-11-28 16:20:45	1.2024.324
d638be9c-18a1-42e6-b363-aae43c0ef53b	Yubraj Singh	it's really Good 👍	5	0	1.2024.324	2024-11-28 16:18:37	1.2024.324
273b8d07-a3db-4ac0-9166-9a043c8e3eb8	Hemanth 1219	very very very excellent 👏	4	0	1.2024.324	2024-11-28 16:17:55	1.2024.324
2b2244bb-1e0e-408e-bc8b-d5cddd8849de	RITESH Das	The best ai app	1	0	1.2024.324	2024-11-28 16:17:16	1.2024.324
e263b34a-545e-4ade-b423-c1ce8dd322a7	Aashiya Khan	it's very helpful	3	0	1.2024.324	2024-11-28 16:17:12	1.2024.324
f6161614-58c2-4397-9473-35703dc8d387	Moiz	it is good and helps me alot in tasks	4	0	1.2024.324	2024-11-28 16:17:06	1.2024.324
685a86cf-5588-408f-873c-9c951bc10f66	Sidhu moose wala	very good app	4	0	1.2024.324	2024-11-28 16:16:57	1.2024.324
c14eddfd-2ac2-48e3-a326-c754dfa2a396	Tanbir Ahmed saikat	nc apo	5	0	1.2024.324	2024-11-28 16:16:52	1.2024.324
c452c5ff-e151-4fe9-adb7-abadcffb6558	Chiku	one of my favorite friend	5	0	1.2024.310	2024-11-28 16:16:34	1.2024.310
aa91a3ea-24e3-44f9-80bf-ffc2c5142c5b	Khadiga Mohamed	جميل اوي اوي اوي	5	0	1.2024.310	2024-11-28 16:16:00	1.2024.310
2527d2c8-c4fe-4182-8b16-a77a17799340	Kenes Syiem	Thanks for the apps it was great to use and whatever I want to search it i found very helpful thank ...	5	0	1.2024.317	2024-11-28 16:15:51	1.2024.317

Exploratory Data Analysis (EDA) & Data Processing

Exploratory Data Analysis (EDA)

This step focuses on analyzing the **structure** of the dataset, **identifying and handling missing values**, understanding the distribution of **key features** (e.g., ratings, review length, thumbs-up counts)
Visualized trends such as ratings distribution, app version review counts, and sentiment over time.

Text Preprocessing

Converted all text to lowercase for uniformity. **Removed unnecessary punctuation** (e.g., . , -) while **retaining meaningful characters (e.g., emojis).**
Removed **special characters** and **excessive whitespace**.

E.g. Original: "I LOVE this app!!! 😍😍 It's super useful..."
After processing: "love app super useful 😍😍"

Tokenization, Stopword Removal, Lemmatization

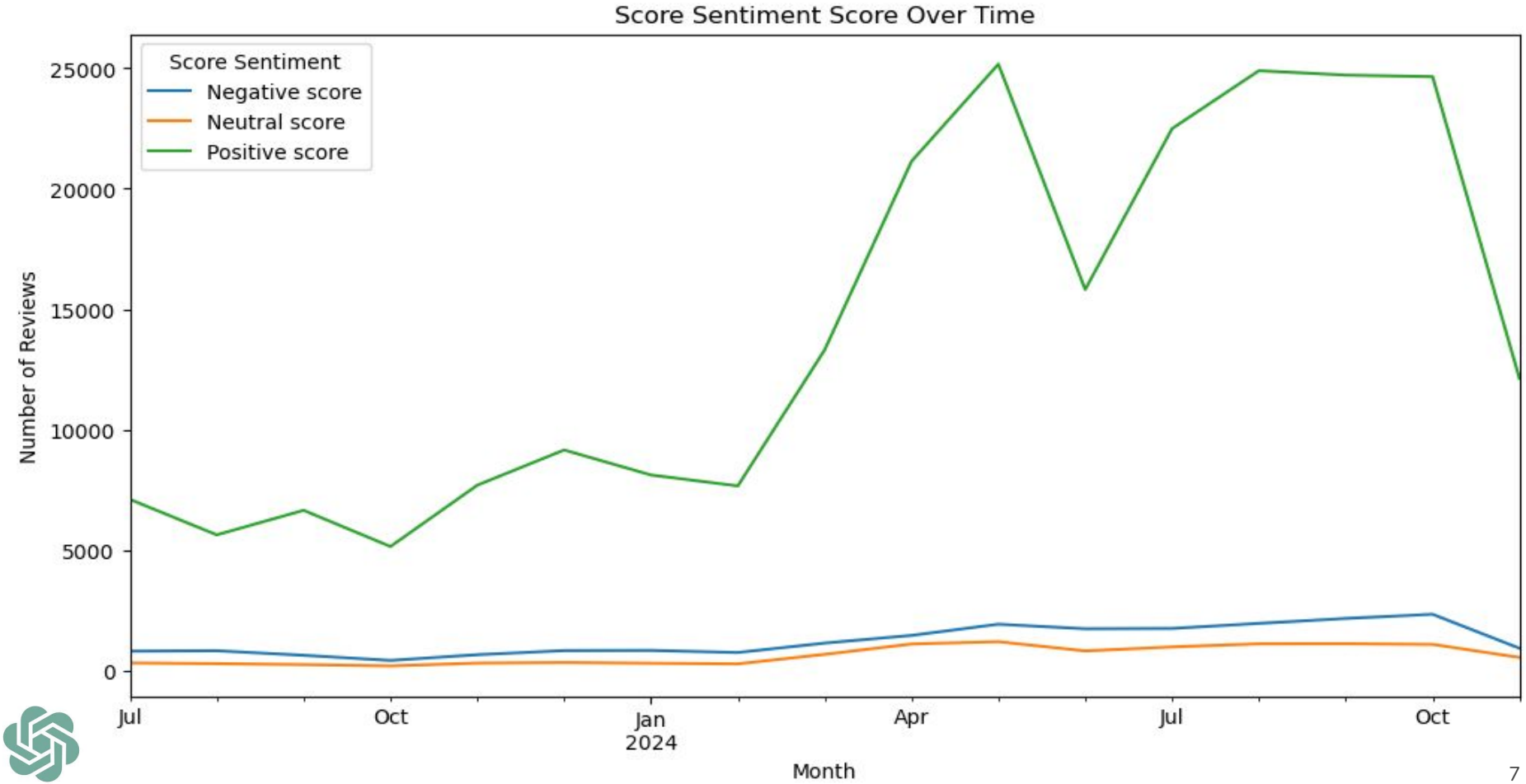
Split text into individual tokens (words or phrases)
We removed stop words but kept words like {'not', 'no', 'very', 'too', 'only', 'but'} that enhances meaning to a text.

Having a large df, we applied **lemmatization through a for loop instead of lambda function for faster runtime**.

(non-English) Language Filtering

Using pre-trained language detection model from fasttext, **the majority (~90%) of non-English text** was detected and dropped, ensuring focus on English content for better NLP analysis.

Addressed limitations:
Some transliterated text (e.g., **non-English languages** written in Roman script) may remain undetected


Score Sentiment Score Over Time
Score Sentiment
Negative score
Neutral score
Positive score
25000
20000
15000
10000
5000
0
Number of Reviews
Jul
Oct
Jan
2024
Apr
Jul
Oct
Month

Vader Sentiment Lib

Developed in 2014, **VADER (Valence Aware Dictionary and Sentiment Reasoner)** is a powerful lib particularly effective for analyzing sentiments in social media content, and allows to handle informal language, slang, emojis and emoticons

Strengths of the library:

- **Computationally economical** while maintaining **high accuracy**
- **Easy to understand** and use, even for little programming experience
- **Considers intensifiers, punctuation** and capitalization to gauge sentiment intensity
- **Applies rules for sentiment modifiers**, such as giving more weight to text after the word "but"

VADER Scoring System:

- Provides **aggregate compound score** (emoji, text, punctuation) ranging from -1 (very negative) to 1 (very positive)
  Capable of detecting both sentiment **polarity** and **intensity**

Compound Score Calculation:
The final compound score after applying the **'Sentiment Intensity Analyzer'**, is computed as

$$\frac{x}{\sqrt{x^2 + \alpha}}$$

Where:

- *x is the sum of adjusted valence scores for all text*
- *α is a normalization constant, set to 15*

Interpretation:

- Positive sentiment: compound score >= 0.05
- Neutral sentiment: -0.05 < compound score < 0.05
- Negative sentiment: compound score <= -0.05

Research Questions

1. Which specific features or aspects of the ChatGPT app (e.g., performance, user interface, response quality) are frequently mentioned in positive, neutral and negative reviews?

2. How has user's sentiment (positive, neutral and negative reviews) on specific features evolve across different app updates?

3. How does bigram analysis reveal about changes in user feedback across app versions and the challenges users face in prompt handling?

Research Question 1

Which specific features or aspects of the ChatGPT app (e.g., performance, user interface, response quality) are frequently mentioned in positive, neutral and negative reviews?

Step 1: Create a dictionary with features (performance, user interface, response quality, prompt handling, and updates) and associated keywords.

Step 2: Use sentiment analysis tools like Vader Sentiment to assign polarity scores (e.g., positive, negative, or neutral) to each review.

Step 3: Aggregate the sentiment scores for each feature to determine whether mentions are mostly positive or negative.

Results & Key Findings

Here's how we defined the five features that we want to analyze and the keywords to identify them

- **Performance**: Focus on aspects related to the app's speed, stability, reliability, and responsiveness.

  KEYWORDS{ `performance', 'accurate', 'faster', 'freeze', 'slow', 'crash', 'loading', 'delay}

- **User_Interface**: Provides an intuitive and pleasant environment for users to interact with the model, focusing on clarity, responsiveness, and usability. It simplifies complex backend processes and ensures that users can focus solely on their queries and responses.

  KEYWORDS {`ui', 'notification', 'alert', 'voice', 'dictate', 'recording', 'recognition'}

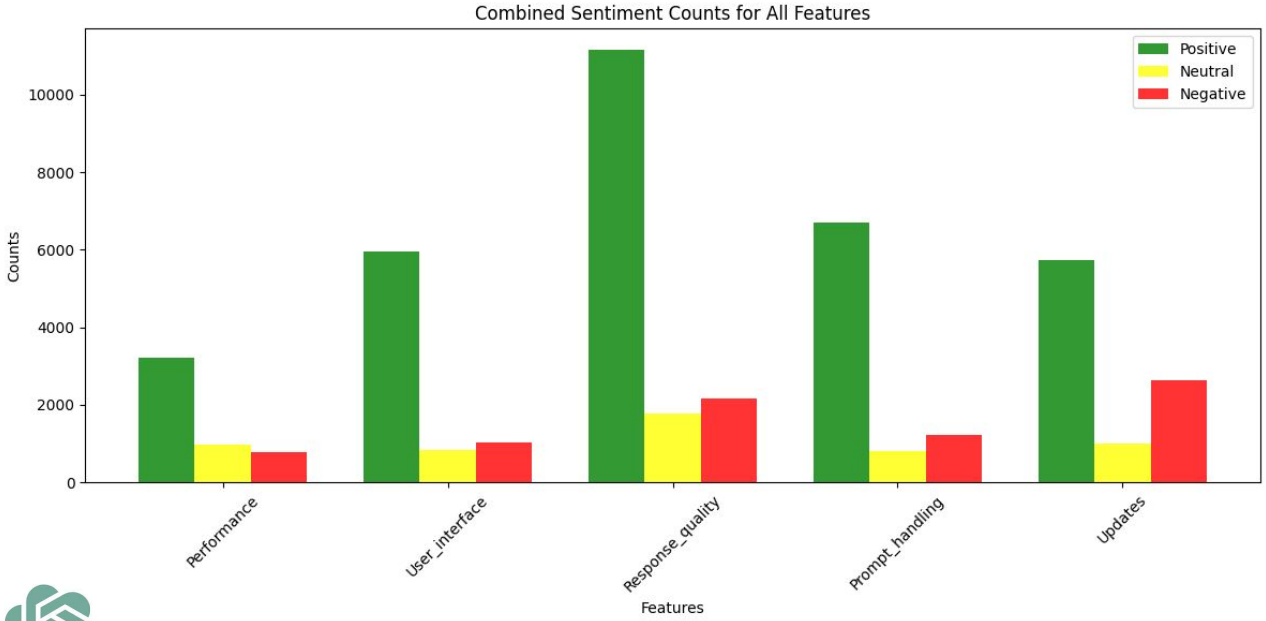
- **Response_Quality**: Evaluates how well the generated answers meet user expectations in terms of accuracy.
  KEYWORDS{`correct', 'response', 'language', 'accuracy', 'answer', 'output', 'insightful'}

- **Prompt_Handling**: Ensures effective interpretation, & generation of accurate responses based on user inputs.
  KEYWORDS {'prompt', 'question', 'input', 'command', 'generate', 'clarify', 'interpret'}

- **Updates**: Refers to the feedback, reactions, and behavior of users following the release of a new version update.
  KEYWORDS {'fix', 'update', 'updated', 'bug', 'error', 'improvement', 'version', 'modify'}

Results & Key Findings

Across all five features, response quality was the best performing, while users seem to be least satisfied with updates



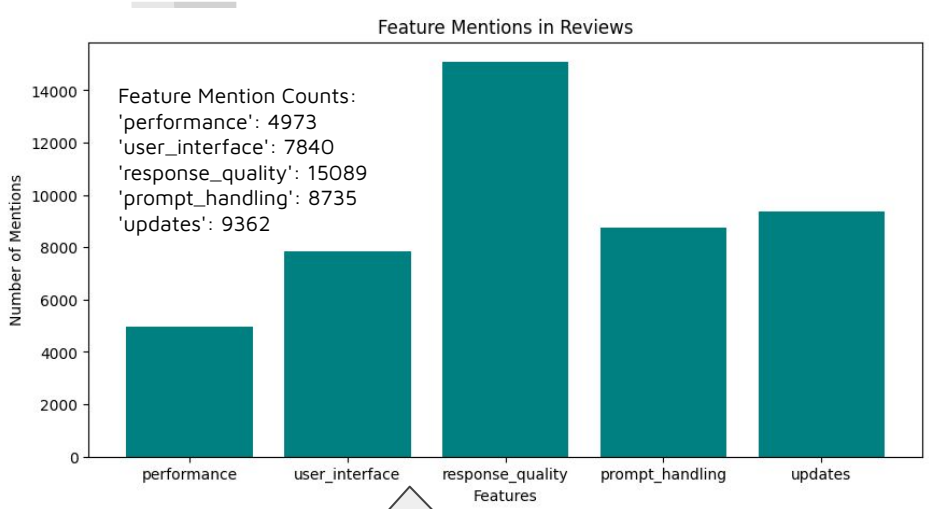
- Response_quality stands out with the highest positive sentiment count, suggesting a strong area of user satisfaction

- Both prompt handling and updates have a high count of positive feedback, though they also show noticeable negative sentiments, indicating areas where improvements might be expected

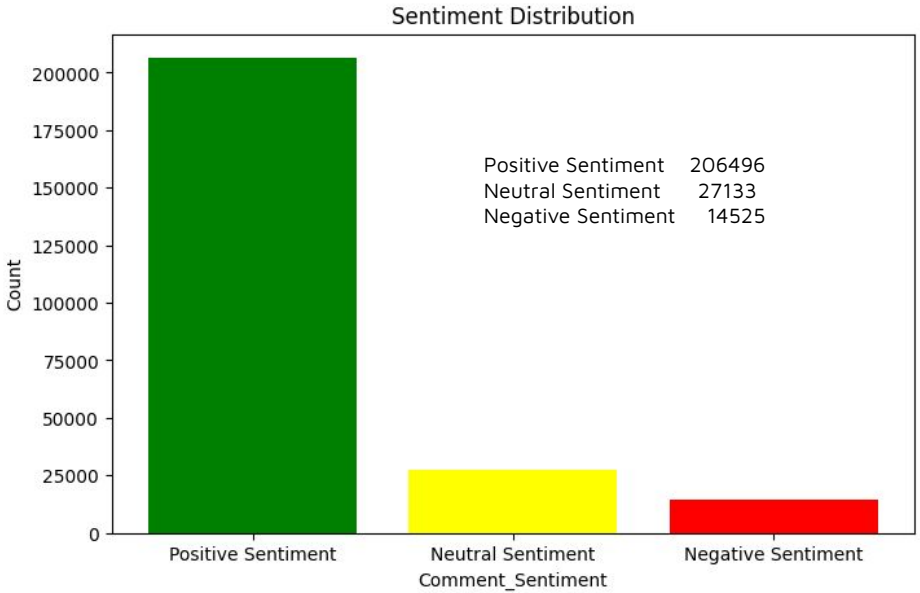
- User_interface has strong positive sentiment but comparatively more neutral and negative sentiments than response quality, signaling room for enhancement.

- Performance has the lowest positive sentiment count and the smallest overall feedback volume, suggesting it may be less impactful to users but also an area for improvement.

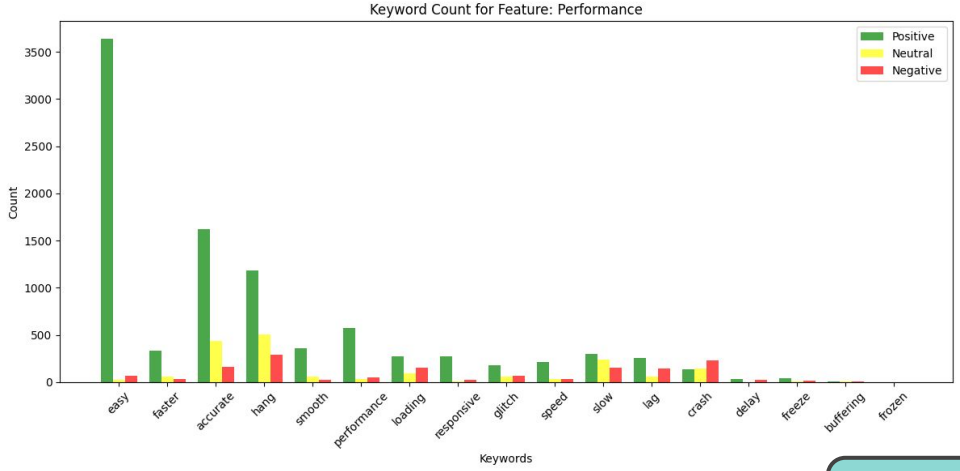
Results & Key Findings

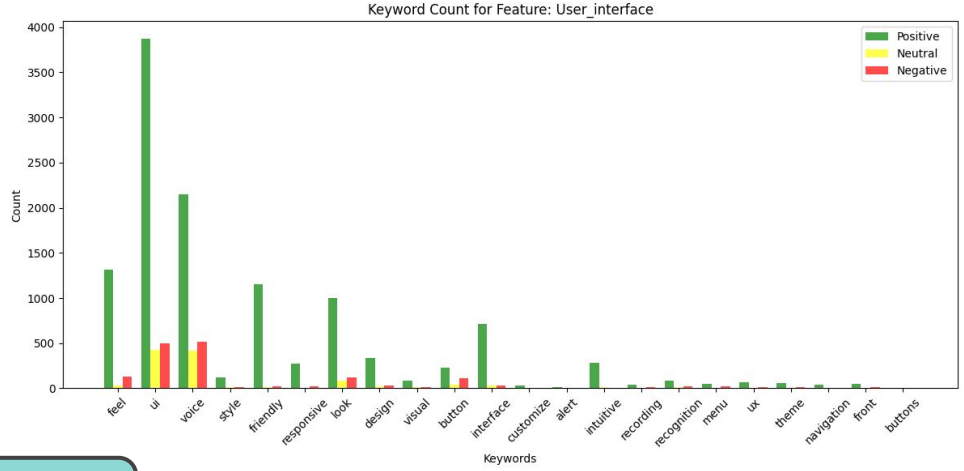


Over 15,000 responses mentioned “response quality”, both “updates” and “prompt handling” are mentioned over 8,000 times. Performance is the least discussed feature with less than 5,000 mentions.

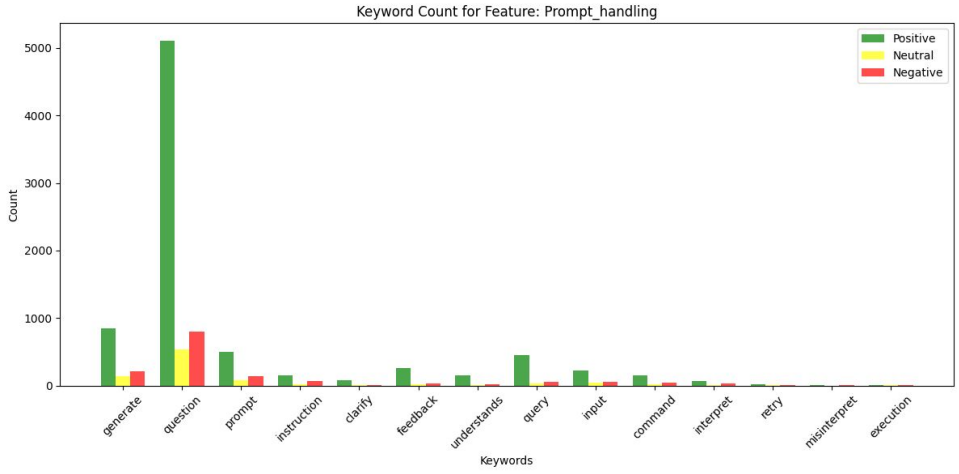


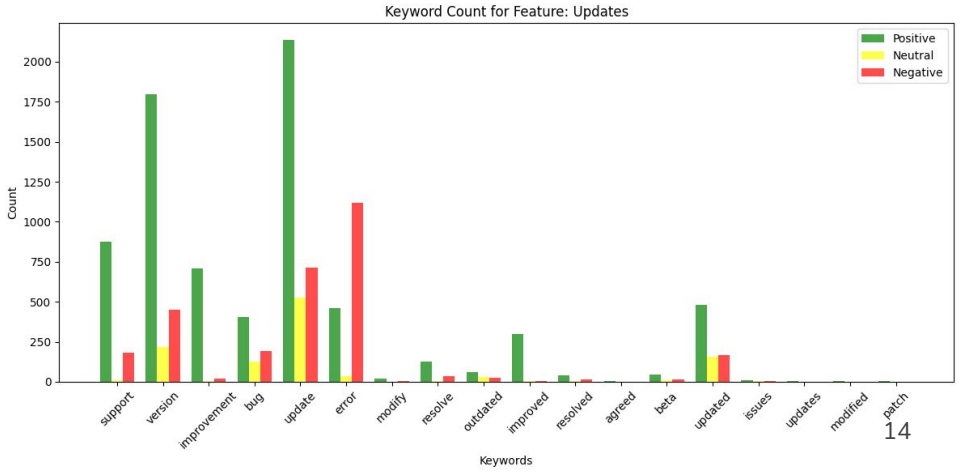
Overall, 83.2% of the sentiment distribution are positive, 10.9% of the sentiment distribution are neutral, 5.9% of the sentiment distribution are negative.



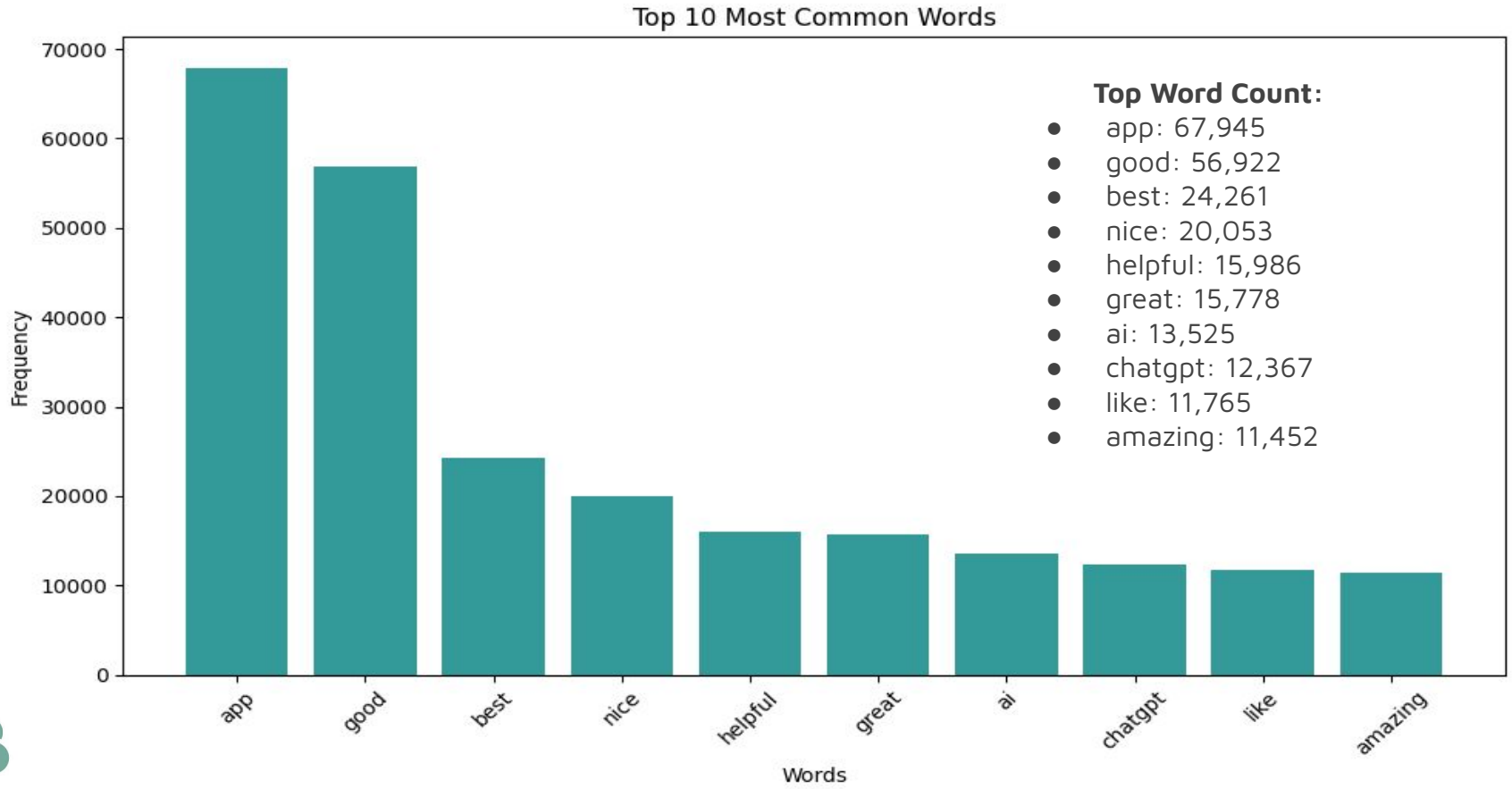


Results & Key Findings





Results & Key Findings



Research Question 2

How has user's sentiment (positive, neutral and negative reviews) on specific features evolved across different app updates?

Step 1: Evaluate user sentiment trends (Positive, Neutral, and Negative) across different app versions.

Step 2: Analyze the distribution and trends of user comments across functional categories: performance, user interface, response quality, prompt handling, and updates.

Step 3: Focus on how user sentiment for specific features evolves over time across app versions.

1. Evaluating user sentiment trends across different app versions

Method

- Using content_Sentiment column (categorized as Positive, Neutral, or Negative) to classify user sentiment.

- Group reviews by app version and sentiment to calculate the count of each sentiment category.

- Compute the **percentage of each sentiment** (positive, neutral, negative) for every version.

- Plot the sentiment proportions as trends over time.

Code Snippet

[15]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Check and clean the data
if 'appVersion' not in data.columns or 'Comment_Sentiment' not in data.columns:
    raise ValueError("Missing required columns: 'appVersion' or 'Comment_Sentiment'")

data = data.dropna(subset=['appVersion', 'Comment_Sentiment'])  # Remove Null Values

# Group by appVersion and Comment_Sentiment and count the number of each sentiment
sentiment_distribution = data.groupby(['appVersion', 'Comment_Sentiment']).size().unstack(fill_value=0)

# Make sure the column order is ['Positive Sentiment', 'Neutral Sentiment', 'Negative Sentiment']
sentiment_order = ['Positive Sentiment', 'Neutral Sentiment', 'Negative Sentiment']
for col in sentiment_order:
    if col not in sentiment_distribution.columns:
        sentiment_distribution[col] = 0  # If a column is missing, fill it with 0
sentiment_distribution = sentiment_distribution[sentiment_order]

# Check if the group result is empty
if sentiment_distribution.empty:
    raise ValueError("The sentiment distribution is empty. Please check your data.")

# Calculating Percentages
sentiment_distribution_percentage = sentiment_distribution.div(sentiment_distribution.sum(axis=1), axis=0) * 100
```

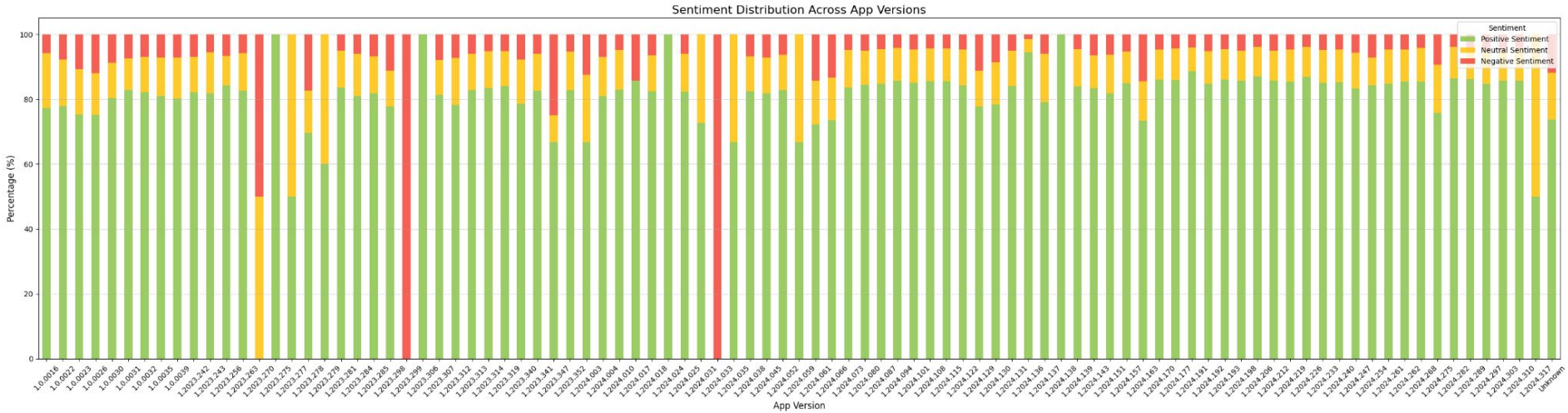
Results & Key Findings

1. Evaluating user sentiment trends across different app versions.

Positive Sentiment (Green): Shows a general upward trend across app updates, indicating improved user satisfaction as issues were resolved and new features were introduced.

Negative Sentiment (Red): Dominates in earlier versions but gradually declines, reflecting effective bug fixes and problem resolution in later updates.

Neutral Sentiment (Yellow): Fluctuates, possibly due to users adapting to experimental features or unclear benefits of updates.



*ATTENTION: the abnormal high percentage sometimes can't reflect the result since they might caused by the small review number.

2. Analyzing the distribution of comment trends across features

Code Snippet

Methodology:

- Use Keywords we defined earlier to categorize the reviews.

- Perform keyword matching on reviews to tag them with functional categories.

- Group reviews by app version and calculate **the count** of mentions for each functional feature.

- Compute **the percentage** of mentions for each feature and visualize the trends.

```python
from tabulate import tabulate

# calculate
categories = ["performance", "user_interface", "response_quality", "prompt_handling", "updates"]

for category, words in keywords.items():
    data[category] = data['cleaned_tokens'].apply(lambda x: match_keywords(x, words))

data['category_flag'] = data[categories].idxmax(axis=1)

version_stats = data.groupby('appVersion').agg(
    **{f'{category}_count': (category, 'sum') for category in categories},  # Number of matches per question type
    total_reviews=('cleaned_tokens', lambda x: x.notnull().sum())  # Only valid comments are counted
).reset_index()

for category in categories:
    version_stats[f'{category}_percentage'] = (
        version_stats[f'{category}_count'] / version_stats['total_reviews']
    ) * 100

version_stats = version_stats.reset_index()

formatted_table = tabulate(
    version_stats.head(10),
    headers="keys",
    tablefmt="pretty"
)

print(formatted_table)
```

✓ 1.9s Python

	index	appVersion	performance_count	user_interface_count	response_quality_count	prompt_handling_count	updates
0	0	1.0.0016	255	275	216	112	21
1	1	1.0.0022	164	204	154	101	14
2	2	1.0.0023	348	478	404	245	34
3	3	1.0.0026	77	89	99	53	93
4	4	1.0.0030	30	47	38	21	56
5	5	1.0.0031	53	69	79	48	42
6	6	1.0.0032	97	120	177	105	18
7	7	1.0.0035	97	102	156	92	90
8	8	1.0.0039	191	196	290	139	16
9	9	1.2023.242	57	72	149	73	83

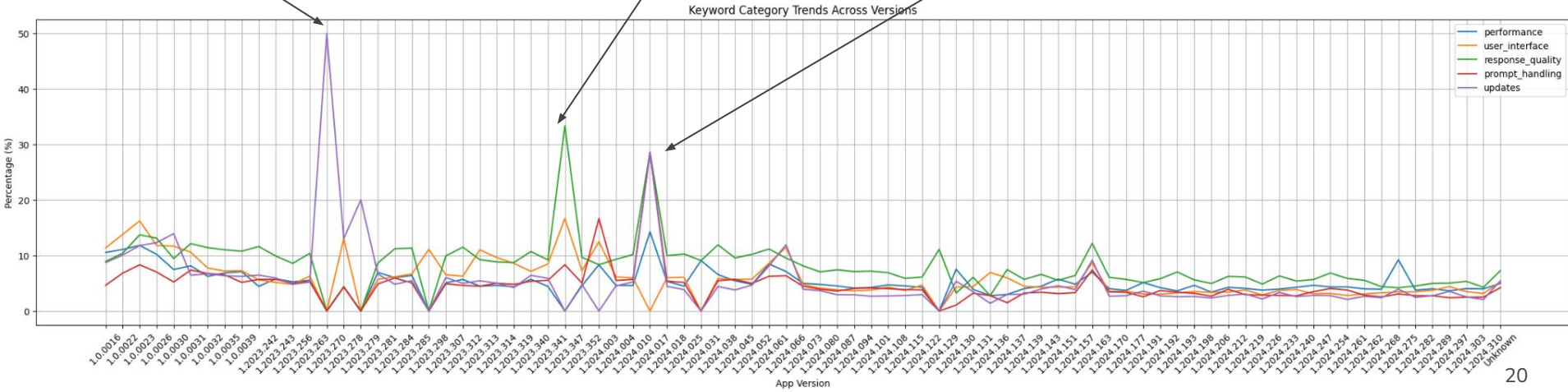
2. Analyzing the distribution of comment trends across features

Results & Key Findings

Around version 1.2023.256, the percentage of users' evaluations on response quality increased significantly.

Around version 1.2023.347. User comments focus on the keyword prompt_handling.

Around version 1.2024.017, response quality and prompt handling reached a small peak together, reflecting the concerns of users about these issues in GPT4.



3. Analyzing sentiment trends for specific features

Method

Group reviews by app version and feature category to calculate sentiment counts.

Compute the **percentage of each sentiment category for specific features.**

Plot sentiment trends to identify shifts over time.

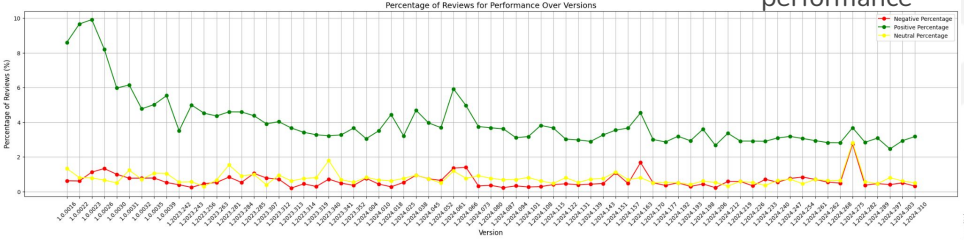
```python
import pandas as pd
import matplotlib.pyplot as plt

function_types = ['performance', 'user_interface', 'response_quality', 'prompt_handling', 'updates']

# Filter out "Unknown" versions
valid_version_data = data[data['appVersion'] != 'Unknown']

# Make sure version number ordering is not affected (keep it as a string)
valid_version_data['appVersion'] = valid_version_data['appVersion'].astype(str)

# Plot the number of positive, negative, and neutral reviews for each classification feature
for func in function_types:
    # Response to negative comments
    negative_counts = valid_version_data[valid_version_data['Comment_Sentiment'] == 'Negative Sentiment'] \
        .groupby('appVersion')[func].sum().reset_index(name='Negative_Count')

    # For positive reviews
    positive_counts = valid_version_data[valid_version_data['Comment_Sentiment'] == 'Positive Sentiment'] \
        .groupby('appVersion')[func].sum().reset_index(name='Positive_Count')

    # For neutral comments
    neutral_counts = valid_version_data[valid_version_data['Comment_Sentiment'] == 'Neutral Sentiment'] \
        .groupby('appVersion')[func].sum().reset_index(name='Neutral_Count')

    # Merge positive, negative, and neutral review data
    counts = pd.merge(negative_counts, positive_counts, on='appVersion', how='outer').fillna(0)
    counts = pd.merge(counts, neutral_counts, on='appVersion', how='outer').fillna(0)
    counts = counts.sort_values(by='appVersion')

    # Draw a line chart
    plt.figure(figsize=(30, 6))
    plt.plot(counts['appVersion'], counts['Negative_Count'], marker='o', label='Negative Reviews', color='red')
    plt.plot(counts['appVersion'], counts['Positive_Count'], marker='o', label='Positive Reviews', color='green')
    plt.plot(counts['appVersion'], counts['Neutral_Count'], marker='o', label='Neutral Reviews', color='yellow')
    plt.title(f'Trend of Reviews for {func.capitalize()} Over Versions', fontsize=14)
    plt.xlabel('Version', fontsize=12)
    plt.ylabel('Number of Reviews', fontsize=12)
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.legend()
    plt.show()
```

✓ 1.5s Python

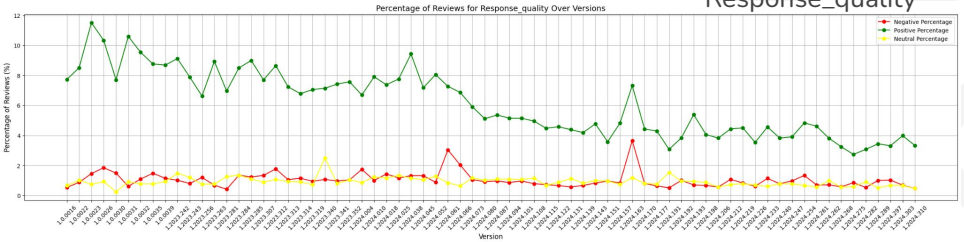
3. Analyzing sentiment trends for specific features

Results & Key Findings

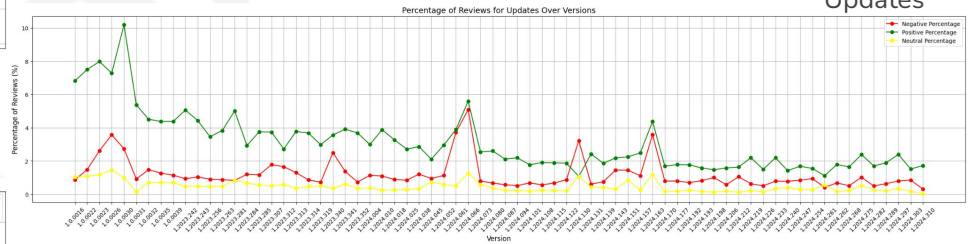
performance



Response_quality



Updates



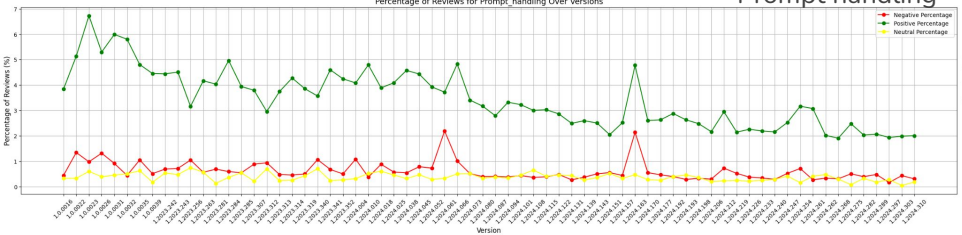
Performance: Maintains high proportions, highlighting its importance to users.

Response Quality: Consistently dominates user feedback, reflecting its critical role in the user experience.

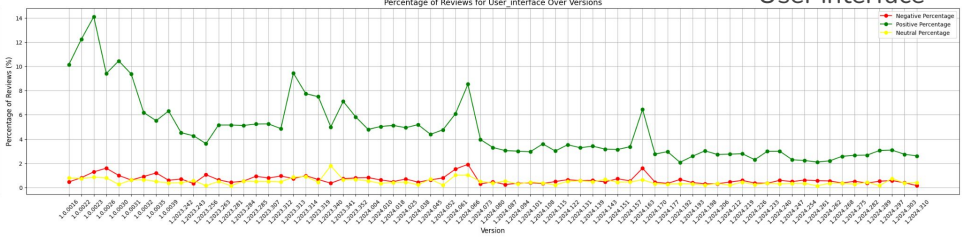
Updates: In early versions (GPT-1), user expectations for updates were high, but in later versions (GPT-3 and GPT-4), update-related comments significantly decreased. This indicates that users view the application as mature and no longer expect frequent updates.

Analyze sentiment trends for specific features

Prompt handling



User interface



Why Are Trends for All Five Features Similar?

Data Distribution Patterns:

Focus Shift: As the app evolved, user attention likely shifted from basic issues (e.g., performance, interface) to more advanced functionalities like coding or writing assistance. This dilution may reduce the prominence of positive feedback for foundational features.

Sentiment Classification Bias:

Users tend to express neutral sentiments more frequently unless they feel extremely satisfied or dissatisfied. This creates an illusion of stability in Neutral and Negative sentiments while Positive sentiment visibly declines.

Expectation-Reality Gap:

As the app improves, user expectations increase. If updates fail to fully meet heightened expectations, positive feedback may drop, even if the app objectively performs better.

Research Question 3

How does bigram analysis reveal about changes in user feedback across app versions and the challenges users face in prompt handling?

Step1: Introduce the bigram analysis methodology.

Step2: Compare high-frequency bigrams across app versions.

Step3: Summarize sentiment distribution in prompt handling.

Step4: Identify challenges using high-frequency bigrams and keywords.

Methodological Overview of Bigram Analysis

- Utilized **Python's nltk library** to extract keywords by tokenizing text, removing stopwords, and ranking word frequencies.
- Used **Python's CountVectorizer** to extract bigrams (two consecutive words) from user comments.
- Rank keywords and bigrams by frequency to provide insights into recurring themes and patterns in user feedback.

```
nltk.download('stopwords')

def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()
    return text

target_version = '1.2024.066'

before_comments = data[data['appVersion'] < target_version]
before_comments['cleaned_comment'] = before_comments['comment'].apply(clean_text)

after_comments = data[data['appVersion'] > target_version]
after_comments['cleaned_comment'] = after_comments['comment'].apply(clean_text)

def analyze_keywords_and_bigrams(comments):
    stop_words = set(stopwords.words('english'))
    tokens = comments['cleaned_comment'].dropna().apply(lambda x: [word for word in x.split() if word not in stop_words])
    token_list = list(chain.from_iterable(tokens))
    token_counts = Counter(token_list).most_common(20)

    vectorizer = CountVectorizer(ngram_range=(2, 2), max_features=20, stop_words='english')
    ngram_matrix = vectorizer.fit_transform(comments['cleaned_comment'].dropna())
    ngram_features = vectorizer.get_feature_names_out()
    ngram_counts = Counter(dict(zip(ngram_features, ngram_matrix.toarray().sum(axis=0)))).most_common(20)

    return token_counts, ngram_counts

before_token_counts, before_bigrams = analyze_keywords_and_bigrams(before_comments)
after_token_counts, after_bigrams = analyze_keywords_and_bigrams(after_comments)
```

High frequency bigrams before and after version 1.2024.066

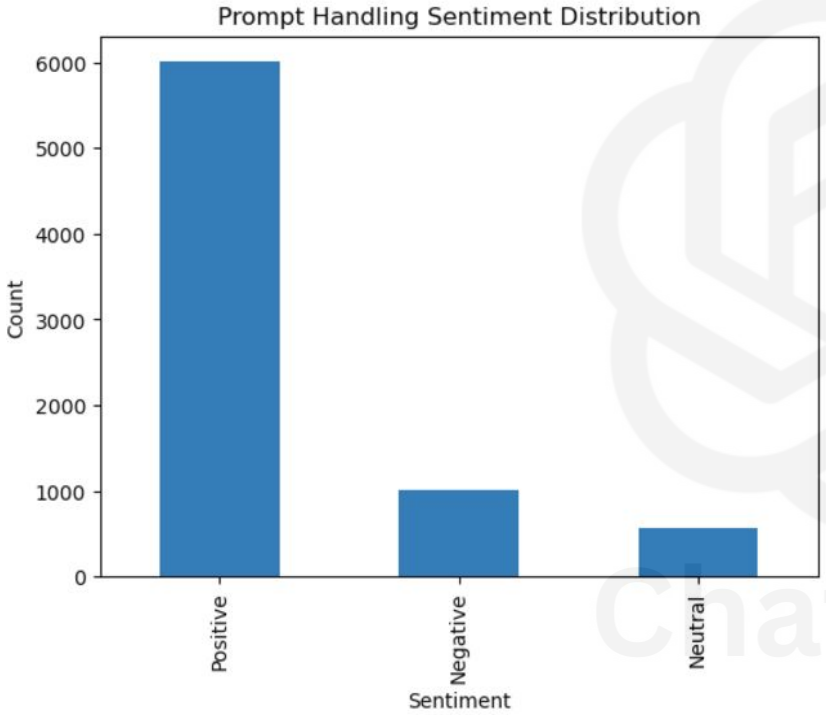
Bigram	Frequency	Bigram	Frequency
good app	1883	good app	6270
best app	1553	best app	5191
chat gpt	1164	nice app	3786
great app	1104	chat gpt	3102
nice app	958	great app	2915
love app	676	best ai	1871
best ai	671	love app	1780
amazing app	550	amazing app	1451
really good	455	useful app	1392
easy use	402	helpful app	1237
ai app	387	app good	1129
app good	374	really good	1114
useful app	364	easy use	1060
really helpful	330	really helpful	990
helpful app	317	ai app	973
app really	308	app helpful	946
voice chat	288	excellent app	900
web version	271	like app	822
excellent app	264	app useful	810
use app	260	app really	720

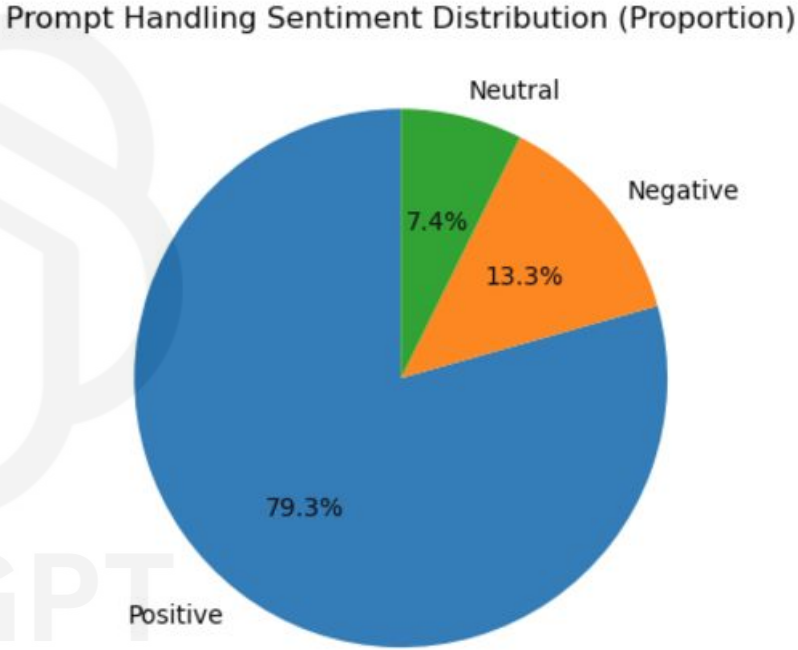
High frequency bigrams of negative comments before and after version 1.2024.066

Bigram	Frequency	Bigram	Frequency
chat gpt	103	chat gpt	285
phone number	68	try later	177
web version	65	wrong answer	159
good app	53	gives wrong	155
doesnt work	51	wrong answers	143
dont know	50	bad app	133
went wrong	50	dont know	131
wrong answers	48	good app	126
try later	46	doesnt work	125
wrong answer	46	worst app	122
gives wrong	44	stopped working	119
worst app	39	wrong information	116
use app	38	went wrong	103
app good	37	use app	98
error message	35	network error	94
error occurred	34	error occurred	93
voice chat	34	app doesnt	75
bad app	32	app good	72
wont let	32	keeps saying	71
stopped working	31		

Overall Feedback: Positive phrases like "**good app**" and "**best app**" remain prevalent, indicating general user satisfaction.
Negative Feedback: Issues like "wrong answer" and "stopped working" post-update suggest challenges in accuracy and system stability.

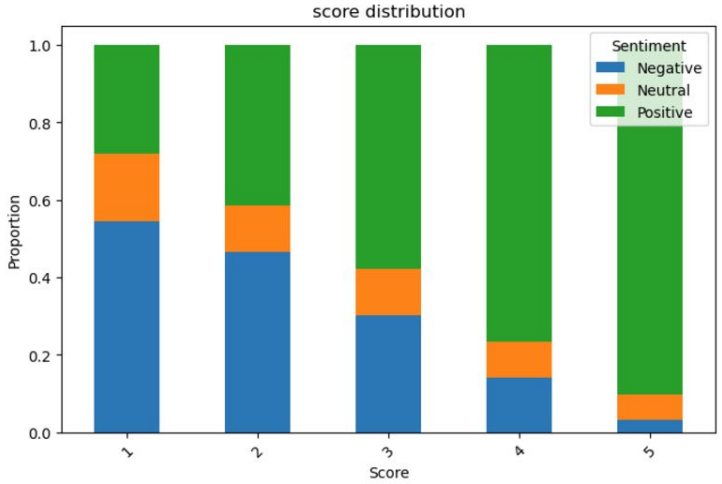
Sentiment Distribution in Prompt Handling Feedback





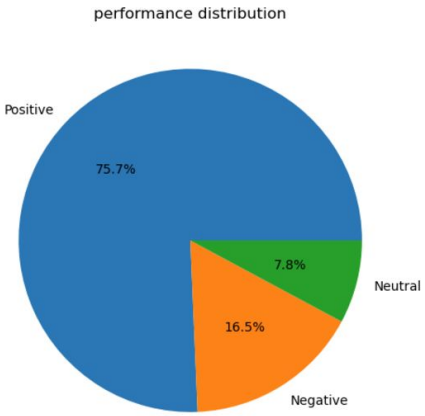
- Positive feedback dominates at 79.3%, indicating overall user satisfaction with prompt handling.
- Negative feedback at 13.3% highlights areas that need further exploration to understand user challenges better

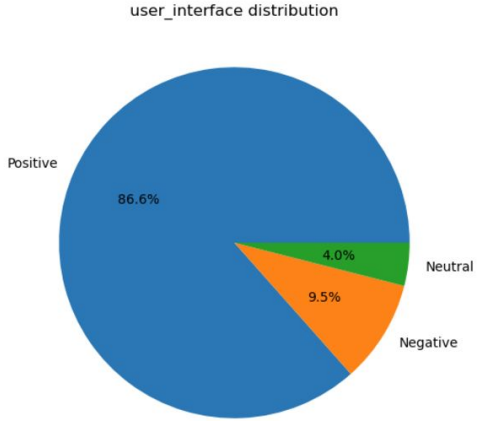
Sentiment Distribution in Prompt Handling Across Various Features

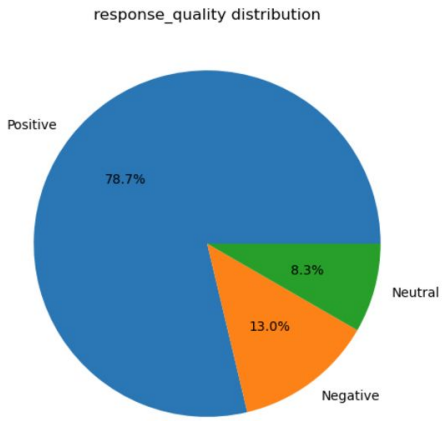


Higher user ratings (4-5) correlate with predominantly positive feedback, while lower ratings (1-2) exhibit increased negative sentiment.

Sentiment analysis across performance, user interface, and response quality shows positive feedback dominance, but negative sentiments point to specific areas requiring further attention.







Prompt Handling Issues: User Interface-Related Negative Feedback

- Focused on analyzing frequent keywords and bigrams in user interface-related negative comments.
- Identified meaningful bigrams such as *"**mandarin written**"* and *"**input field**"*, indicating specific challenges users face.
- Results help pinpoint actionable areas to enhance user interaction with prompts.

Keyword	Frequency	Bigram	Frequency
app	31	amazing app	3
question	22	answer questions	3
questions	19	app does	3
answer	19	app doesnt	2
chatgpt	17	app quite	2
ai	17	asked question	3
even	16	chat gpt	3
answers	13	chatgpt app	4
asked	13	false information	3
ask	12	input field	3
input	12	instead answering	2
use	11	mandarin written	2
good	11	paid version	2
doesnt	11	recent conversations	2
prompt	11	text input	3
one	9	view recent	2
issue	9	wont answer	2
chat	9	written sentences	3
quite	9		
responses	9		

Keyword and Bigram Frequencies for User Interface-Related Negative Comments

Prompt Handling Issues: Response Quality-Related Negative Feedback

- Analyzed response quality-related negative comments to uncover users' main frustrations.
- Key bigrams include ***"wrong answer"*** **and** ***"simple question,"*** highlighting gaps in AI response accuracy and handling basic prompts.

Keyword	Frequency	Bigram	Frequency
answer	367	answer question	56
question	279	answer questions	46
app	262	answer wrong	11
questions	247	answering questions	16
answers	168	answers questions	25
wrong	122	ask question	39
give	117	asked question	20
ask	113	chat gpt	22
chatgpt	91	correct answer	17
asked	82	doesnt answer	17
ai	81	gives wrong	14
even	80	good app	13
cant	77	question answer	25
dont	73	question asked	14
like	72	right answer	11
doesnt	72	simple question	10
good	67	waste time	10
time	64	worst app	11
response	61	wrong answer	32
gives	60	wrong answers	24

Keyword and Bigram Frequencies for Response Quality-Related Negative Comments

Prompt Handling Issues: Performance-Related Negative Feedback

- Explored negative comments focusing on performance issues like speed and system reliability.
- Key bigrams such as *"**fast answer**"* and *"**keeps crashing**"* reflect concerns over system performance and stability.

Keyword	Frequency	Bigram	Frequency
app	46	answer hindi	4
question	24	answer questions	4
slow	20	data hai	3
answer	18	dinner scene	2
questions	17	doesnt work	3
like	14	fast answer	2
prompt	14	fastboot mode	3
doesnt	11	faster running	2
ai	11	important note	3
prompts	11	keeps crashing	3
use	11	kind slow	2
version	11	koi bhi	2
even	10	lagging times	2
chatgpt	9	message wrong	2
wrong	9	paid version	3
cant	9	running mile	2
responses	9	simple question	3
time	9	way better	2
dont	8	web version	3
giving	7	wrong answer	4

Keyword and Bigram Frequencies for Performance-Related Negative Comments

Conclusion

Summary of Key Findings

Research Question 1 -

Users frequently praise the ChatGPT app for its accurate responses, smooth performance, and user-friendly interface, while occasional criticisms focus on issues like lag, crashes, and inconsistencies in response accuracy during specific contexts.

Research Question 2 -

- Positive sentiment generally increased in earlier versions but began declining as expectations rose and attention shifted to advanced functionalities.
- Response quality and performance remain core user concerns, showing the highest levels of feedback.
- Future improvements should focus on advanced features while maintaining reliability in foundational functionalities.

Research Question 3 -

With regards to prompt handling, sentiment distribution analysis showed positive feedback, but also highlighted specific challenges, such as feedback about slow response times, system instability, input challenges, and inaccurate answers

To better address our research question, we utilized these techniques

Techniques Utilized From Class:

- Data Cleaning (handling missing values, duplicates)
- Tokenization and Lemmatization
- Text Sentiment Analysis (using VADER or similar tools)

New Techniques and Skills Gained:

- Advanced Text Cleaning (handling emojis, slangs)
- Language Filtering (FastText for detecting non-English text)
- Sentiment Analysis for Emojis and Slangs
- Keyword Matching for Feature Identification
- Text Labeling and Finding Relevant Comments
- Extracting High-Frequency Keywords and Bigrams

Overall, the project journey created many learning opportunities and expanded our horizon on the capabilities of existing tools and libraries for sentiment analysis and

Limitations

- Not all non-English text were removed as some non-English languages were alphabetical

- Despite keeping negation and conjunction stop words, VADER SentimentIntensityAnalyzer sometimes cannot detect the accurate meaning of single text or emoji tokens, e.g.

tokens	cleaned_tokens	Text_Sentiment	Emoji_Sentiment	Overall_Sentiment	Comment_Sentiment
[wow, what, a, unique, creation, kudos, guys 🔥👍🔥]	[wow, unique, creation, kudos, guys🔥👍🔥]	{'neg': 0.0, 'neu': 0.098, 'pos': 0.902, 'compound': 0.8481}	{'neg': 0.615, 'neu': 0.385, 'pos': 0.0, 'compound': -0.5859}	{'neg': 0.267, 'neu': 0.222, 'pos': 0.511, 'compound': 0.6597}	Positive Sentiment
[mind, blowing]	[mind, blowing]	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}	{'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}	Neutral Sentiment

- Custom researched and created list of keywords is a crude method of text labelling can be an inaccurate representation of “mentioning of a feature” and therefore we considered bigrams

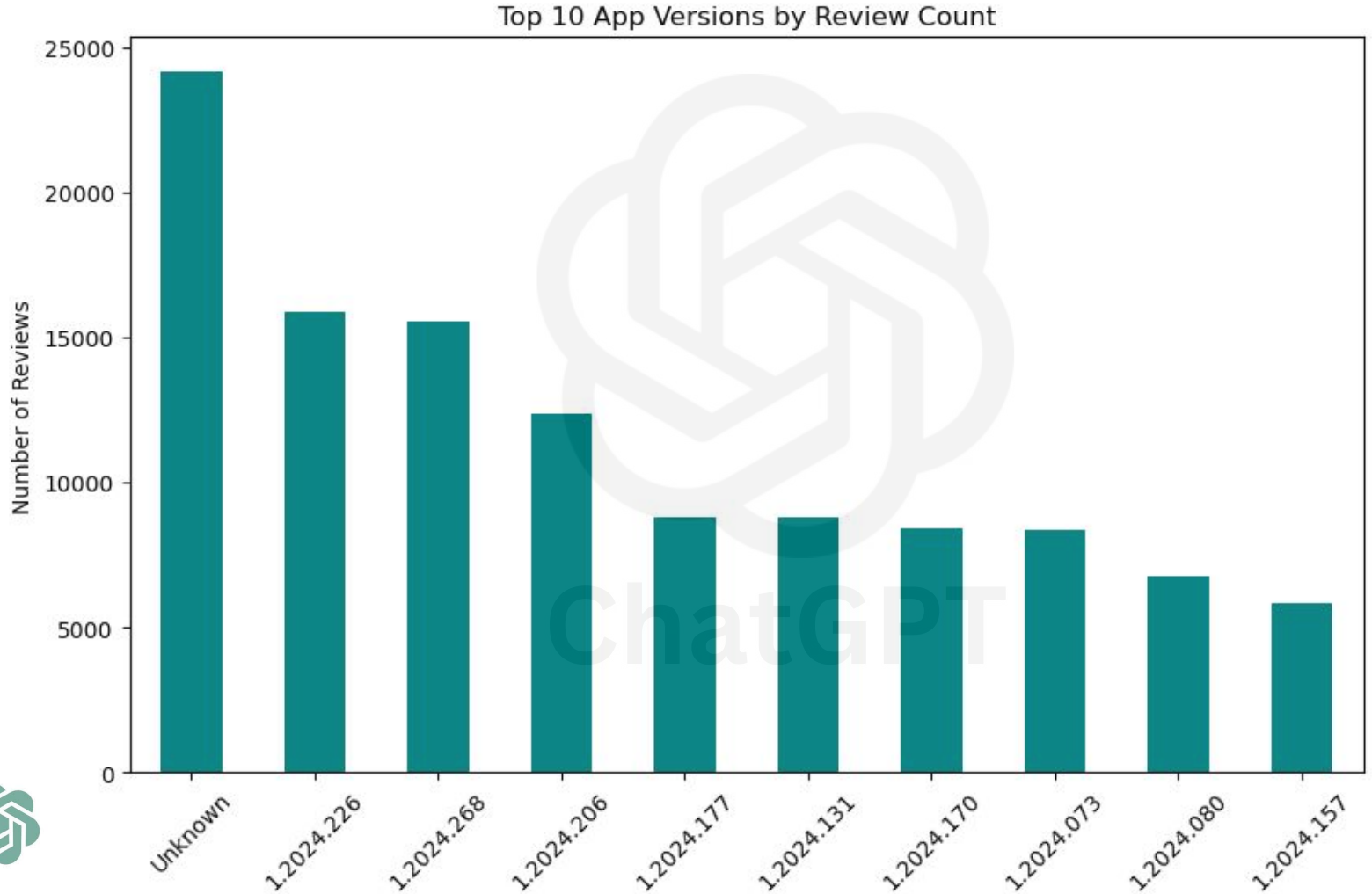
- Top keywords and bigrams appearances are usually not as meaningful (e.g. goog app, best app, great app, nice app)

Thank you!

Appendices
& Additional Research

Cleaned and Processed Data Dictionary

- **reviewId -** Unique Review ID
- **userName -** Username of the reviewer
- **comment -** The textual content of the review
- **score -** star rating that the reviewer posted (1 = lowest, 5 = highest)
- **date_time -** date and time of the review posted
- **appVersion** - The version of the app being reviewed
- **date** - date of the review
- **time** - timestamp of the review
- **Score_sentiment** - Rating >= 4: Positive; Rating == 3: Neutral; Rating <= 2: Negative
- **tokens** - Words in their original form without any lemmatization or stemming
- **Cleaned_tokens** - undergone additional preprocessing for analysis
- **Text Sentiment** - pos, neu, neg and standardized compound score derived from Sentiment Analyzer for text only
- **Emoji Sentiment** - pos, neu, neg and standardized compound score derived from Sentiment Analyzer for emoji only
- **Comment Sentiment** - pos, neu, neg and standardized compound score derived from Sentiment Analyzer for the overall comment


Top 10 App Versions by Review Count
Number of Reviews
25000
20000
15000
10000
5000
0
Unknown
1.2024.226
1.2024.268
1.2024.206
1.2024.177
1.2024.131
1.2024.170
1.2024.073
1.2024.080
1.2024.157

VADER handles emojis, emoticons, slangs and gen-z vocab quite well

- Slangs

```
print(sentiment_analyzer_scores("Today SUX!"))
print(sentiment_analyzer_scores("Today only kinda sux! But I'll get by, lol"))#outputToday SUX!------------------------------ {'neg': 0.779, 'neu': 0.221, 'pos': 0.0, 'compound': -0.5461}Today only kinda sux! But I'll get by, lol {'neg': 0.127, 'neu': 0.556, 'pos': 0.317, 'compound': 0.5249}
```

- Emoticons

```
print(sentiment_analyzer_scores("Make sure you :) or :D today!"))Make sure you :) or :D today!----------- {'neg': 0.0, 'neu': 0.294, 'pos': 0.706, 'compound': 0.8633}
```

- Emojis

```
print(sentiment_analyzer_scores('I am 😄 today'))
print(sentiment_analyzer_scores('😊'))
print(sentiment_analyzer_scores('😳'))
print(sentiment_analyzer_scores('😕'))#OutputI am 😄 today---------- {'neg': 0.0, 'neu': 0.476, 'pos': 0.524, 'compound': 0.6705}😊---------- {'neg': 0.0, 'neu': 0.333, 'pos': 0.667, 'compound': 0.7184}😳---------- {'neg': 0.275, 'neu': 0.268, 'pos': 0.456, 'compound': 0.3291}😕---------- {'neg': 0.706, 'neu': 0.294, 'pos': 0.0, 'compound': -0.34}💘---------- {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

```
#Baseline sentence
sentiment_analyzer_scores('The food here is good')
```

```
The food here is good---------- {'neg': 0.0, 'neu': 0.58, 'pos': 0.42,
'compound': 0.4404}
```

```
#Punctuation
print(sentiment_analyzer_scores('The food here is good!'))
print(sentiment_analyzer_scores('The food here is good!!'))
print(sentiment_analyzer_scores('The food here is good!!!'))
```

```
The food here is good!---------- {'neg': 0.0, 'neu': 0.556, 'pos': 0.44
4, 'compound': 0.4926}
None
The food here is good!!---------- {'neg': 0.0, 'neu': 0.534, 'pos': 0.46
6, 'compound': 0.5399}
None
The food here is good!!!---------- {'neg': 0.0, 'neu': 0.514, 'pos': 0.48
6, 'compound': 0.5826}
None
```

Punctuation: The use of an exclamation mark**(!)**, increases the magnitude of the intensity without modifying the semantic orientation.

```
#Baseline sentence
sentiment_analyzer_scores('The food here is great!')
```

```
The food here is great!---------- {'neg': 0.0, 'neu': 0.477, 'pos': 0.52
3, 'compound': 0.6588}
```

```
#Capitalisation
sentiment_analyzer_scores('The food here is GREAT!')
```

```
The food here is GREAT!---------- {'neg': 0.0, 'neu': 0.438, 'pos': 0.56
2, 'compound': 0.729}
```

Degree modifiers: Also called intensifiers, they impact the sentiment intensity by either increasing or decreasing the intensity.

VADER pays attention to punctuations, captilizations, degree modifiers, negations and conjunctions

```
#Baseline sentence
sentiment_analyzer_scores('The food here is great!')
```

```
The food here is great!---------- {'neg': 0.0, 'neu': 0.477, 'pos': 0.52
3, 'compound': 0.6588}
```

```
#Capitalisation
sentiment_analyzer_scores('The food here is GREAT!')
```

```
The food here is GREAT!---------- {'neg': 0.0, 'neu': 0.438, 'pos': 0.56
2, 'compound': 0.729}
```

Capitalization: Using **upper case** letters to emphasize a sentiment-relevant word in the presence of other non-capitalized words, increases the magnitude of the sentiment intensity.

```
#Conjunctions
sentiment_analyzer_scores('The food here is great, but the service is horrible')
```

```
The food here is great, but the service is horrible {'neg': 0.31, 'neu': 0.523,
'pos': 0.167, 'compound': -0.4939}
```

Conjunctions: Use of conjunctions like “but” signals a shift in sentiment polarity, with the sentiment of the text following the conjunction being dominant.

Word Cloud for Positive Sentiment

Word Cloud for Negative Sentiment

Word Cloud for Neutral Sentiment

High frequency bigrams before and after version 1.2023.313

Bigram	Frequency
good app	816
best app	693
great app	555
chat gpt	529
nice app	433
best ai	337
love app	326
amazing app	252
easy use	246
ai app	209
web version	207
android app	197
really good	182
app good	173
language model	169
useful app	169
open ai	161
chatgpt app	156
chatgpt android	154
app really	153

Bigram	Frequency
good app	7275
best app	6010
nice app	4287
chat gpt	3706
great app	3435
best ai	2192
love app	2100
amazing app	1732
useful app	1570
helpful app	1413
really good	1374
app good	1329
easy use	1205
really helpful	1160
ai app	1137
app helpful	1078
excellent app	1018
like app	961
app useful	901
app really	864

High frequency bigrams of negative comments before and after version 1.2023.313

Bigram	Frequency
phone number	62
chat gpt	50
web version	45
doesnt work	30
went wrong	30
good app	24
use app	22
worst app	22
microsoft account	19
error message	19
gives wrong	19
app doesnt	18
app good	18
wrong answer	18
wrong information	18
wont let	17
wrong answers	17
chat history	16

Bigram	Frequency
chat gpt	336
try later	223
wrong answer	186
gives wrong	178
wrong answers	174
dont know	160
bad app	154
good app	153
doesnt work	149
stopped working	141
worst app	140
wrong information	128
error occurred	123
went wrong	121
use app	116
network error	102
dont like	95
app good	90
voice chat	89
app doesnt	87

Thank you!