iOS Update Sentiment Analysis in Social Media

Rebecca Bronfeld

Nicholas Nehemia

DS 5110: Intro to Data Management & Processing

Introduction

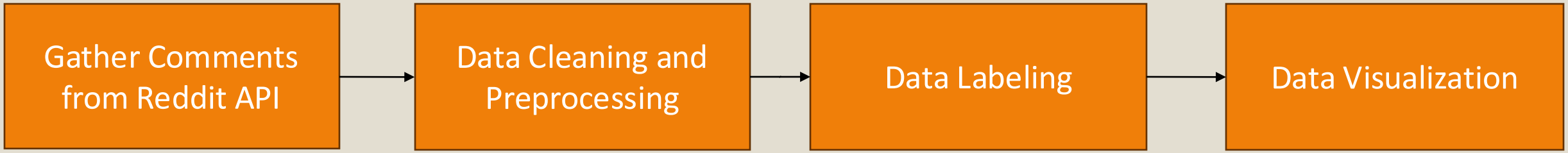
Objectives & Goals 

- Gather initial commentary and reactions from Reddit of previous iOS updates
- Create a sentiment analysis & visualize our findings

Scope 

- This required gathering, cleaning, analyzing, and visualizing the data received.
- We will act as Data Engineer and Data Analyst here, as we focus more on the data gathering and visualization.

Methodology - Overall Process



Methodology – Tools/Packages Used

NLTK
Collections
Re
itertools

Methodology – Extraction and Labeling

1. Hit the API

```python
def scrape_reddit(urlCSVpath):

# Define the post URL or ID
    urlcsv=pd.read_csv(urlCSVpath) # get URL to scrape
    urlList=urlcsv['urlList'].to_list() #convert to list
    all_comments_data = []
    for val in urlList:
        post_url = val
        submission = reddit.submission(url=post_url)
        # Fetch top-level comments with their scores and store in a list
        submission.comments.replace_more(limit=0)  # Removes "More Comments" placeholders
        for top_level_comment in submission.comments:
            all_comments_data.append({
                "subreddit": submission.subreddit.display_name,
                "post_title": submission.title,
                "post_url": val,
                "comment_id": top_level_comment.id,
                "parent_id": submission.id,  # Replies to the post itself
                "comment": top_level_comment.body,
                "comment_score": top_level_comment.score,
                "comment_timestamp": datetime.fromtimestamp(top_level_comment.created_utc),
                "is_top_level": True  # Flag for top-level comment
            })

                # Check if there's a first reply and add it
            for first_reply in top_level_comment.replies:
                all_comments_data.append({
                    "subreddit": submission.subreddit.display_name,
                    "post_title": submission.title,
                    "post_url": val,
                    "comment_id": first_reply.id,
                    "parent_id": top_level_comment.id,  # Points to the top-level comment ID
                    "comment": first_reply.body,
                    "comment_score": first_reply.score,
                    "comment_timestamp": datetime.fromtimestamp(first_reply.created_utc),
                    "is_top_level": False  # Flag for first reply
                })

    # Convert to DataFrame
    df_comments = pd.DataFrame(all_comments_data)
    return(df_comments)
```

2. Remove Emoji

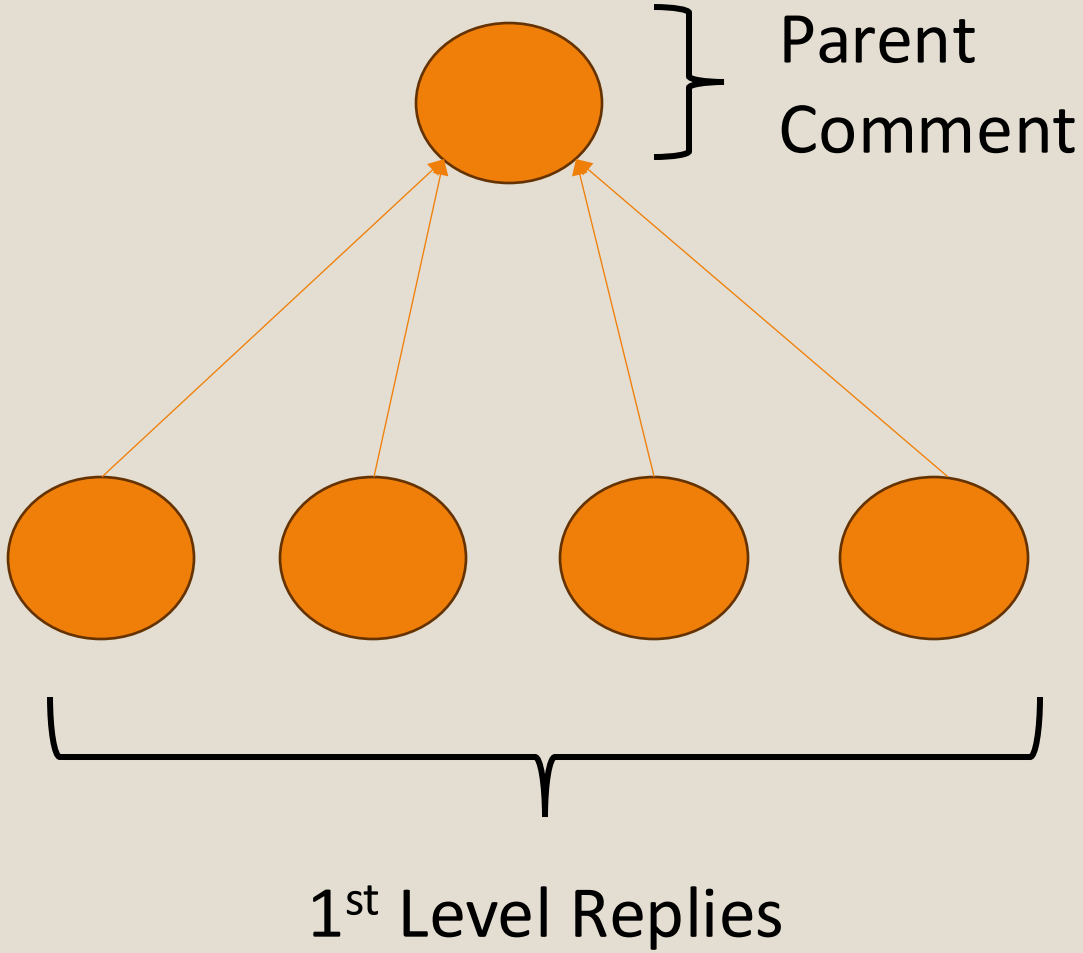
```python
import re

# Function to remove emojis from text
def remove_emojis(text):
    emoji_pattern = re.compile(
        "["
        "\U0001F600-\U0001F64F"  # emoticons
        "\U0001F300-\U0001F5FF"  # symbols & pictographs
        "\U0001F680-\U0001F6FF"  # transport & map symbols
        "\U0001F1E0-\U0001F1FF"  # flags (iOS)
        "\U00002700-\U000027BF"  # other miscellaneous symbols
        "\U0001F900-\U0001F9FF"  # supplemental symbols and pictographs
        "\U00002600-\U000026FF"  # other miscellaneous symbols
        "\U00002B50-\U00002B55"  # additional symbols
        "]+", flags=re.UNICODE
    )
    return emoji_pattern.sub(r'', text)
```

3. HuggingFace Pipeline

```python
classifier=pipeline("sentiment-analysis")

def classfyComment(comment):
    sentiment=classifier(comment)[0]
    return sentiment['label'],sentiment['score']

df_commentsAll[['classification_result','classification_score']]=df_commentsAll['comment'] \
    .apply(lambda x :pd.Series(classfyComment(x)))
df_commentsAll
```

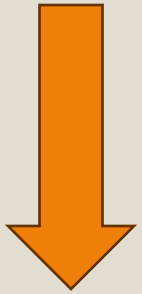
Methodology – Data Transformation

```python
structured_data = []

# Loop through each top-level comment and its replies
for idx, comment in df_commentsAll.iterrows():
    if comment["is_top_level"]:  # Process top-level comments
        structured_data.append({
            "subreddit": comment["subreddit"],
            "post_title": comment["post_title"],
            "post_url": comment["post_url"],
            "parent_comment_id": comment["comment_id"],  # Top-level comment's ID
            "parent_comment": comment["comment"],
            "parent_comment_score": comment["comment_score"],
            "parent_comment_sentiment": comment["classification_result"],
            "parent_adjusted_classification_score" : comment["adjusted_classification_score"],
            "parent_comment_timestamp": comment["comment_timestamp"],
            "reply_comment_id": None,  # No reply for this row, as it's the top-level comment
            "reply_comment": None,
            "reply_comment_score": None,
            "reply_comment_sentiment": None,
            "reply_comment_timestamp": None,
            "reply_adjusted_classification_score" : None,
            "IOS":comment["IOS"]
        })

        # Add each reply to the top-level comment as a new row
        replies = df_commentsAll[(df_commentsAll["parent_id"] == comment["comment_id"]) & (df_commentsAll["is_top_level"] == False)]
        for _, reply in replies.iterrows():
            structured_data.append({
                "subreddit": comment["subreddit"],
                "post_title": comment["post_title"],
                "post_url": comment["post_url"],
                "parent_comment_id": comment["comment_id"],  # ID of the top-level comment
                "parent_comment": comment["comment"],
                "parent_comment_score": comment["comment_score"],
                "parent_comment_sentiment": comment["classification_result"],
                "parent_adjusted_classification_score" : comment["adjusted_classification_score"],
                "parent_comment_timestamp": comment["comment_timestamp"],
                "reply_comment_id": reply["comment_id"],  # ID of the reply
                "reply_comment": reply["comment"],
                "reply_comment_score": reply["comment_score"],
                "reply_comment_sentiment": reply["classification_result"],
                "reply_comment_timestamp": reply["comment_timestamp"],
                "reply_adjusted_classification_score" : reply["adjusted_classification_score"],
                "IOS":reply['IOS']
            })

# Create a DataFrame from structured data
df_structured_comments = pd.DataFrame(structured_data)

# Display the final structured DataFrame
df_structured_comments
```



Methodology – Extraction and Labeling Cont

subreddit	comment_id	parent_id	comment	comment_score	is_top_level	IOS	classification_result
ios	k15zjgg	16m12vy	The tweak they made to Haptic Touch where you can adjust the rea	199	TRUE	17	POSITIVE
ios	k16lc7h	k15zjgg	Omg this is incredible	41	FALSE	17	POSITIVE



subreddit	parent_comment_id	parent_comment	parent_com	parent_comment	reply_comm	reply_comment	reply_comment_score	reply_comment_sentiment	IOS
ios	k15zjgg	The tweak they made to Haptic To	199	POSITIVE					17
ios	k15zjgg	The tweak they made to Haptic To	199	POSITIVE	k16lc7h	Omg this is incredible	41	POSITIVE	17

Methodology – Word Cloud Generation

1. Clean base data frame

```
pattern = r'[^\w\s]'
df["comment_cleaned"]=df["comment"].str.lower()
df["comment_cleaned"] = df["comment_cleaned"].apply(lambda x: re.sub(pattern, '', x) if(pd.notnull(x)) else x )
df["unigrams"] = df["comment_cleaned"].apply(lambda x: re.sub(pattern, '', x))
df["unigrams"]=df["unigrams"].apply(tweet_tokenizer.tokenize)
```

2. Create wordcount grouped by iOS and Sentiment

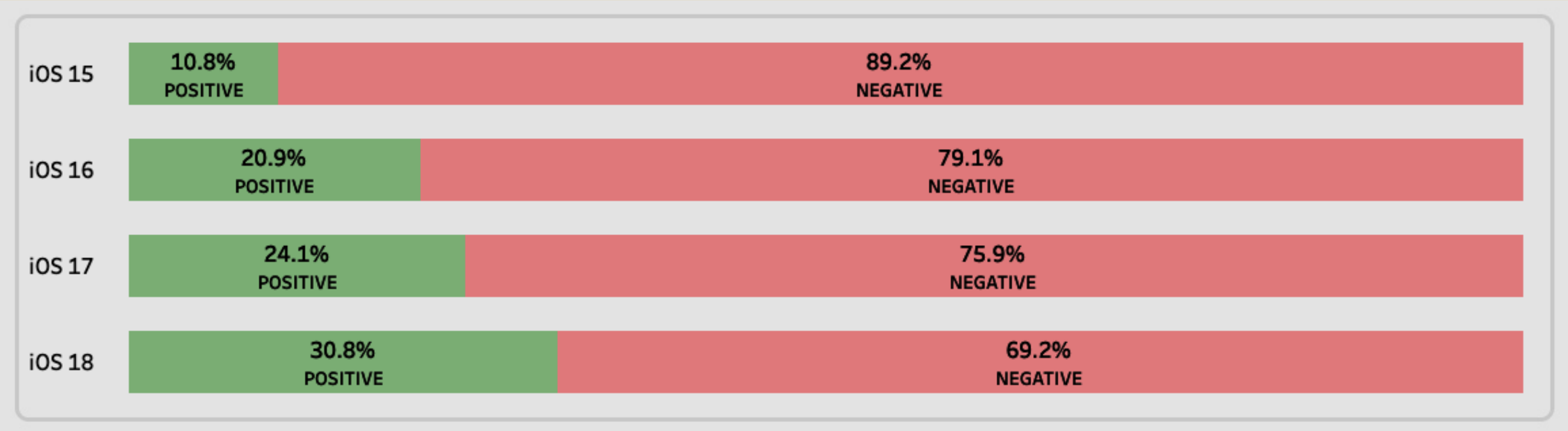
```
def cleanCount(df,ios,sentiment):
    stop = stopwords.words('english')
    cleaned_stopwords=df[(df["IOS"]==ios )& (df["classification_result"]==sentiment)]["unigrams"].apply(lambda x: [item for item in x if item not in stop])
    all_words_nsw = list(itertools.chain(*cleaned_stopwords)) #This will generate all tokenized words into a single list #
    counts_nsw = collections.Counter(all_words_nsw) # This will create a count of all tokenized words
    final_counts_nsw=counts_nsw.most_common() # this will sort on most common words
    dfCounts=pd.DataFrame(final_counts_nsw,columns=["words","count"])
    dfCounts["ios"]=ios
    dfCounts["Sentiment"]=sentiment
    return dfCounts
```

Methodology – Tableau Visualization

- Tableau Dashboard consists of 3 graphs/charts:
  - Horizontal Bar Chart: grouped by iOS Update & Sentiment
  - Word Cloud
  - Sentiment Line Graph at the comment-level
- Dashboard consists of 3 dynamic filters:
  - iOS Upgrade
  - Sentiment
  - Word in Comments

- Demo: iOS Update Sentiment Analysis

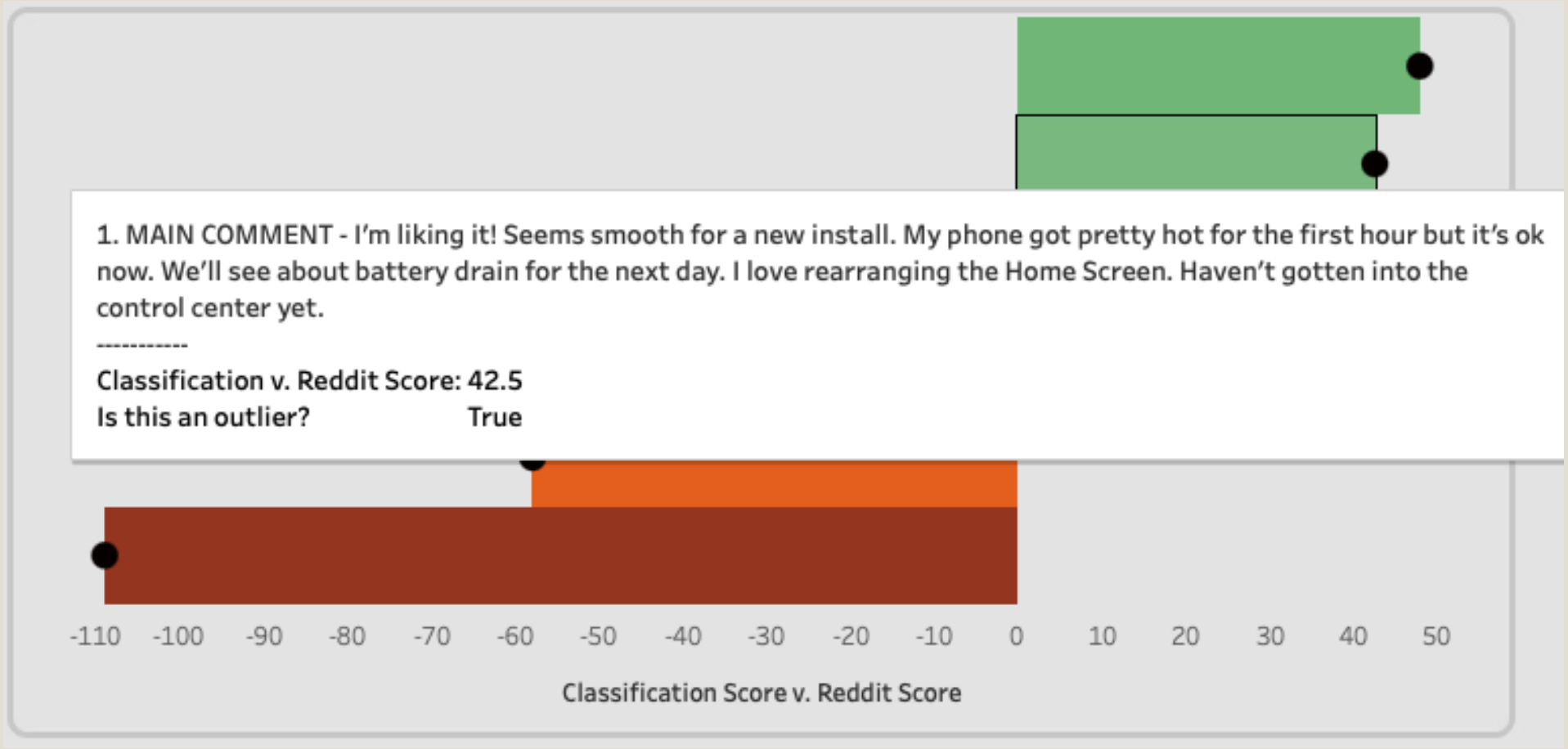
Analysis & Results



Analysis & Results

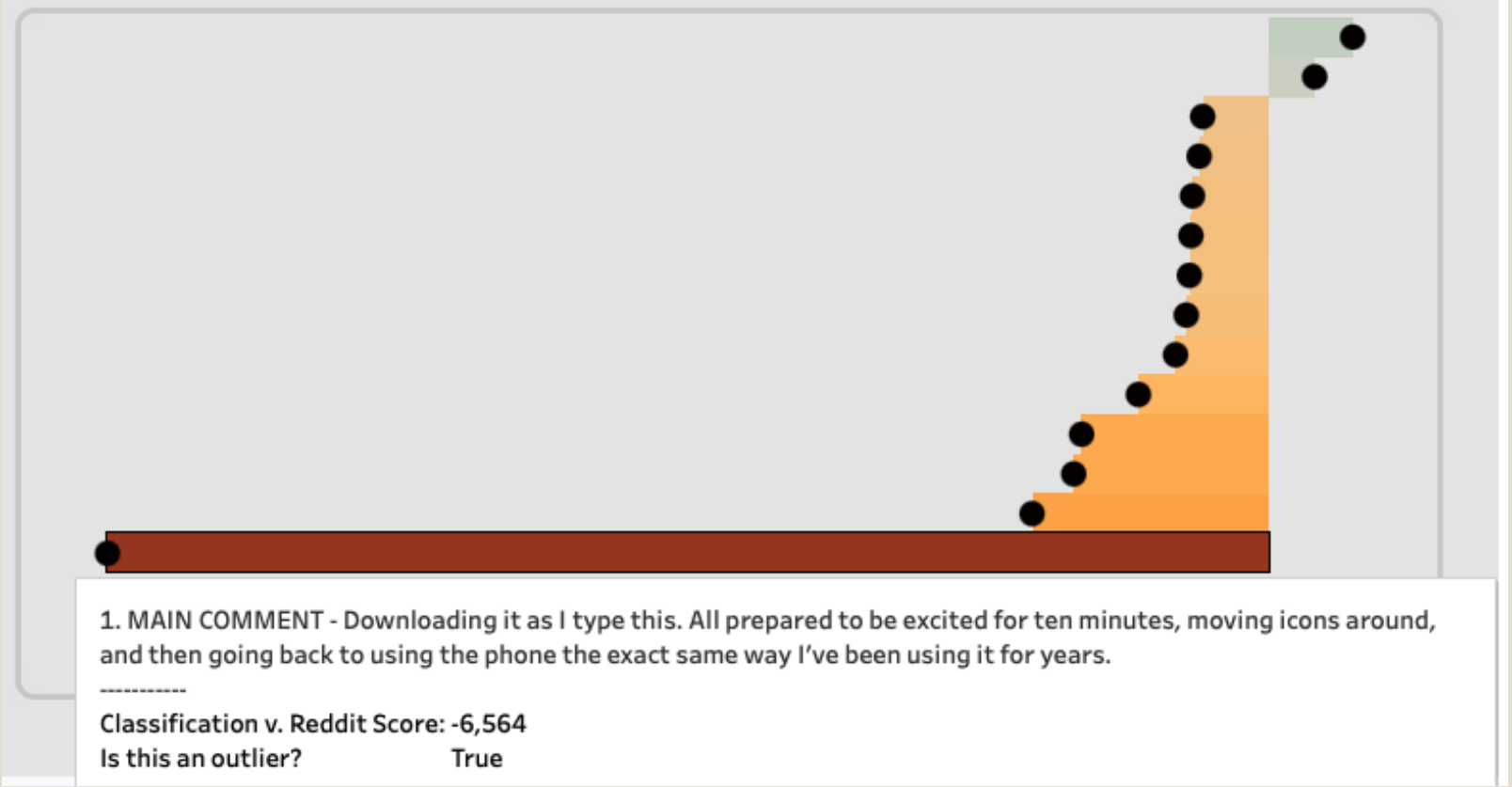
Analysis & Results

Filters: iOS 18 | "Control"



Discussion

- Our findings implicate a consistent negative initial reaction to iOS updates
- The comment with the strongest reaction displays initial negativity, but overall indifference:



Limitations

- Model does not detect sarcasm
- Does not detect questions which could skew our positive sentiments
- Application does not search all subreddits, but rather looks at posts given to it
- Sampling Bias: Users may only vocalize their opinion when they have something to complain about, potentially skewing data

Conclusion

- The majority of sentiments was negative towards all iOS updates
- Out of all updates, the worst sentiment came from iOS 15
- Overall commentary suggest users complain about similar issues with each update: Battery life impact, lock screen changes, control center

Future Recommendations

- Enhance our process: replacing one-time batch process to a data stream, allowing new comments to flow in and see real-time sentiment

- Model enhancement: allowing for accurate neutral sentiments

- For future iOS updates: while net-new features are important, having a focus on improving features that provide consistent negative feedback could be beneficial for continuation of customer loyalty.

References

- https://saturncloud.io/blog/how-to-remove-special-characters-in-pandas-dataframe/#:~:text=Use%20Regex%20Substitution%3A&text=sub()%20function%20from%20the,)%2C%20effectively%20removing%20special%20characters
- https://www.geeksforgeeks.org/scraping-reddit-using-python/
- https://praw.readthedocs.io/en/stable/
- https://www.earthdatascience.org/courses/use-data-open-source-python/intro-to-apis/calculate-tweet-word-frequencies-in-python/
- https://public.tableau.com/app/profile/ken.flerlage/viz/TextAnalysisStarterKit/00ChartMenu

Q&A