# Technical Report: Final Project DS 5110: Introduction to Data Management and Processing

Team Members: Nicholas Nehemia, Rebecca Bronfeld
Khoury College of Computer Sciences
Data Science Program
nehemia.n@northeastern.edu, bronfeld.r@northeastern.edu

December 10, 2024

## Contents

# 1 Introduction

In today's age, many individuals express the opinions of current events using various social media platforms. In our final project assignment, we will utilize social media, specifically Reddit, to gather comments from posts discussing iOS 15, 16, 17 and 18 upgrades, and conduct a sentiment analysis based on the comments and replies found.

# 2 Literature Review

# 3 Methodology

The Reddit API and PRAW Package allowed us to obtain top-level comments and their first reply, then gather that data. The transformers package allowed us to pass the data and begin labeling and classification.
Once data collection and preprocessing was complete, we had outputted two .CSV files for analysis: one related to the comment data, and the second related to the word count from the comments, grouped by iOS upgrades. The two .CSV files were then uploaded to a Tableau Desktop workbook. A horizontal bar-chart comparison, a word cloud, and a sentiment line graph was created, to better understand the types of comments that are within the negative/positive sentiments, which words were found most frequently for each iOS update, and the sentiment surrounding them.

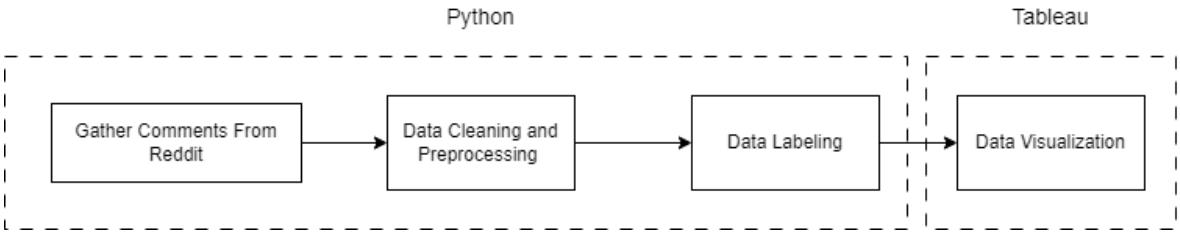


Figure 1: Process Flow of Project

## 3.1 Data Collection

Explain how the data was collected, including the sources and tools used.
In order to collect the data, we will manually find the reddit post we would like to extract comments from , and we will add to a .CSV stored in our system. From there , the application will read into the CSV and extract the data for all links in the CSV, and obtain all top level comments and the first level reply. (see App. A listing 1)
Data was collected using Reddit's API feature. We located posts relating to iOS 15-18 within Reddit, and applied their URLs to the API. Once we had the comments available, we then began processing and cleaning the comments.

## 3.2 Data Preprocessing

Our dataset contains 2 level of comments , the main comment and the 1st level replies to the 1st level comment. We only take the first reply higher level replies might go off topic.

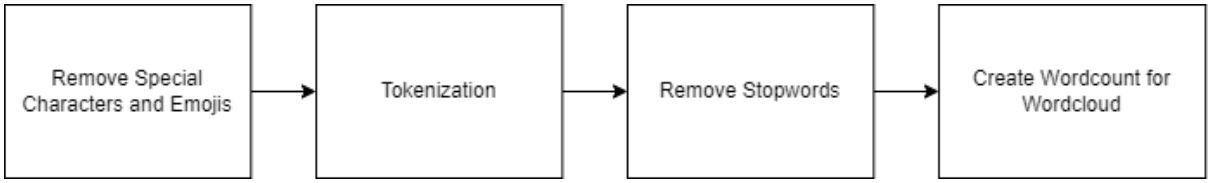


Figure 2: Data Cleaning and preprocessing for analysis

The notebook script will then find all parent comments ,and join the child comments , so we can have the parent and child comments in one row for easier identification and visualization purposes. (See App. A listing 3)
To generate the wordcloud , we also need to have a wordcount table. First , we will clean the data by removing the stopwords and emojis, and doing tokenization. We will then do a word count grouped by iOS version and classification label. (See App. A listing 4)

### 3.3 Analysis Techniques

In this project , we use the transformers package to label the sentiment of all the comments. Using the HuggingFace Transformers pipeline will simplfy this process as we can just pass through the comment, and it will run through a premade pipeline and return the sentiment. (See App. A listing 3) Although we do not need specific preprocessing for the labeling of data, we still need it for the word count and analysis, as the stopwords will appear on the wordcloud and give no insight.

We will also need to perform transformations on the Data Received, as the data structure of the comments is similar to a tree, where we have to link the child comment to the parent comment in order to gain insight in the visualization more easily. To accommodate this , we have transformed the data to have both the child comment and parent comment in one row through some transformation.

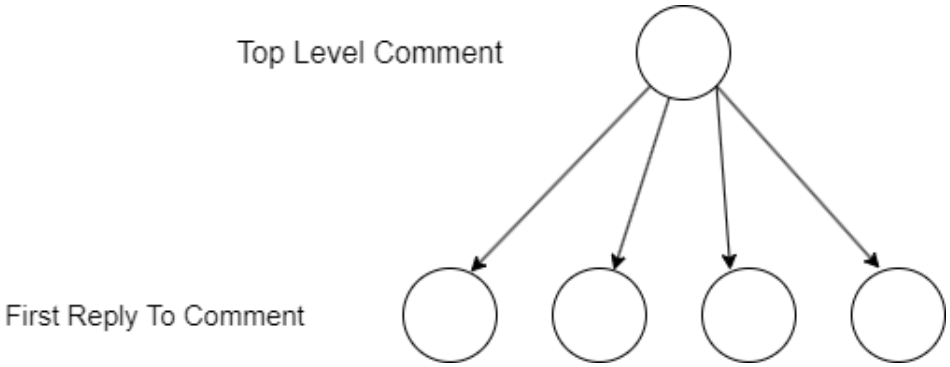


Figure 3: Reddit Comment Data Structure

## 4 Results

Overall, each iOS update has a majority of negative sentiment, across all comments. More than 60 percent of sentiment was negative per update (see Figures 4-7).
By analyzing the word cloud by each update, we could see the most frequently referenced

enhancements included notifications, the control center, and lock screen (see Figures 4-7). Once we had the two .CSV files structured for visualization, we imported the files to a Tableau Workbook and began analysis. Figures 8-10 display the backend of creating the charts used for analysis.

# 5 Discussion

Our results from the Tableau dashboard indicate a need for the business to consider fixing current enhancements as a priority, over net-new features. If a user is not satisfied with their experience, customer loyalty will diminish, and therefore there will be less users to appreciate future updates.

# 6 Conclusion

A key finding from our sentiment analysis is that users displayed negative sentiment for major iOS features, including the battery life, notification bar, and control center. Out of all the updates, the worst sentiment came from iOS 15, with only 10 percent sentiment being positive.
There were a few limitations from end-to-end process. First, we had removed emojis from final analysis, as there was a limitation in Tableau for visualizing these character types. This limited our sentiment to only be shown as text.
The model we had used also does not detect sarcasm, or have consistent accuracy with questions. This could skew the data towards positive sentiment.
We also had limited data in our analysis, as it was only a sample of reviews users had towards this updates. If we had a larger sample size, our sentiment could be viewed differently.
Finally, sampling bias could have occurred, as we were only reviewing iOS reactions from a social media site. Users tend to be vocal online when they have a topic to complain about, therefore skewing the sentiment towards negative.

# 7 References

# References

Remove symbols : saturncloud.io
Source for how to scrape using PRAW : GeeksForGeeks
PRAW Documentation : praw.readthedocs.io
Source for obtaining word counts : earthdatascience.org
Text Analysis Tableau Reference : Tableau Public
Remove Emojis : stackOverflow

# A Appendix A: Code

```
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)
l
```

Listing 1: getting all top level comments and first level replies.

```
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
```

Listing 2: Function to generate the sentiment using HuggingFace

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

Listing 3: Generate data structure for visualization

```
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))
    counts_nsw = collections.Counter(all_words_nsw)
    final_counts_nsw=counts_nsw.most_common()
    dfCounts=pd.DataFrame(final_counts_nsw,columns=["words","count"])
    dfCounts["ios"]=ios
    dfCounts["Sentiment"]=sentiment
    return dfCounts
```

Listing 4: Function to Remove stop words and generate word count

# B Appendix B: Tableau Analysis

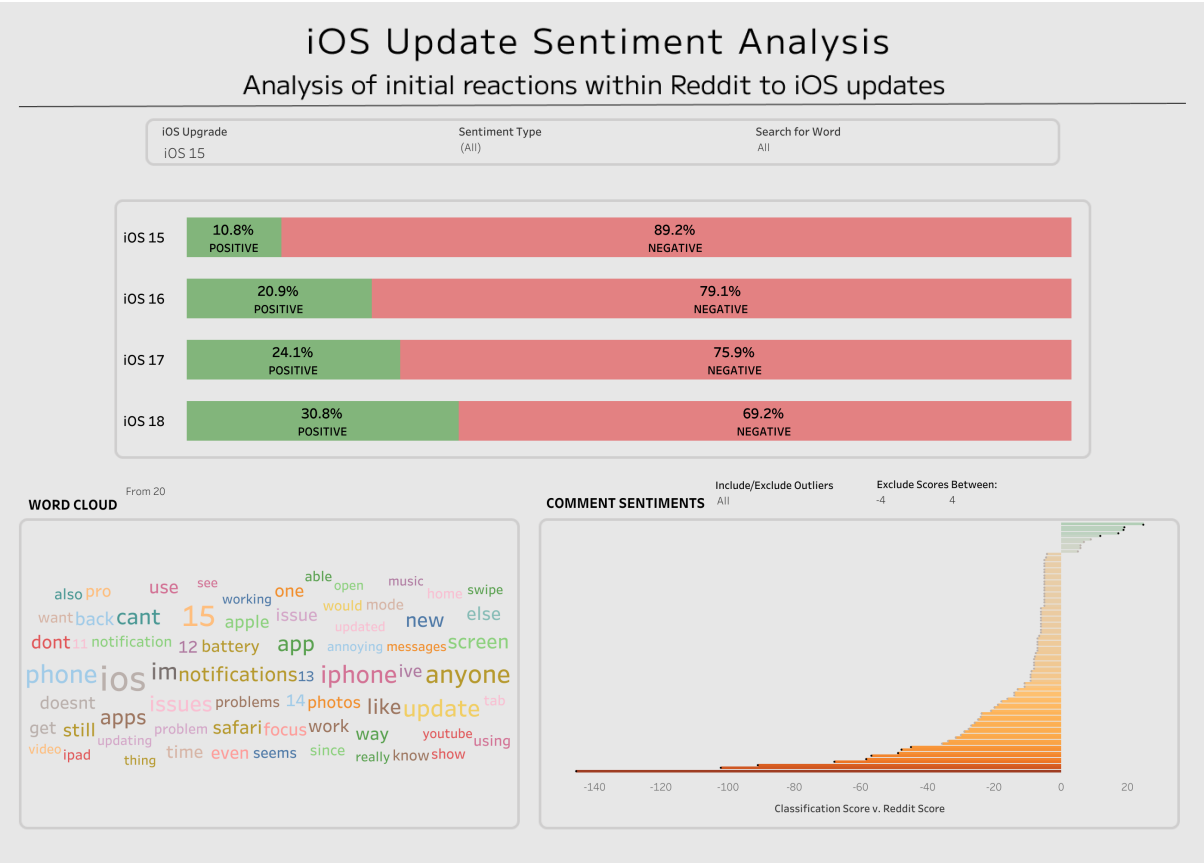


Figure 4: iOS Dashboard Filtered to iOS 15

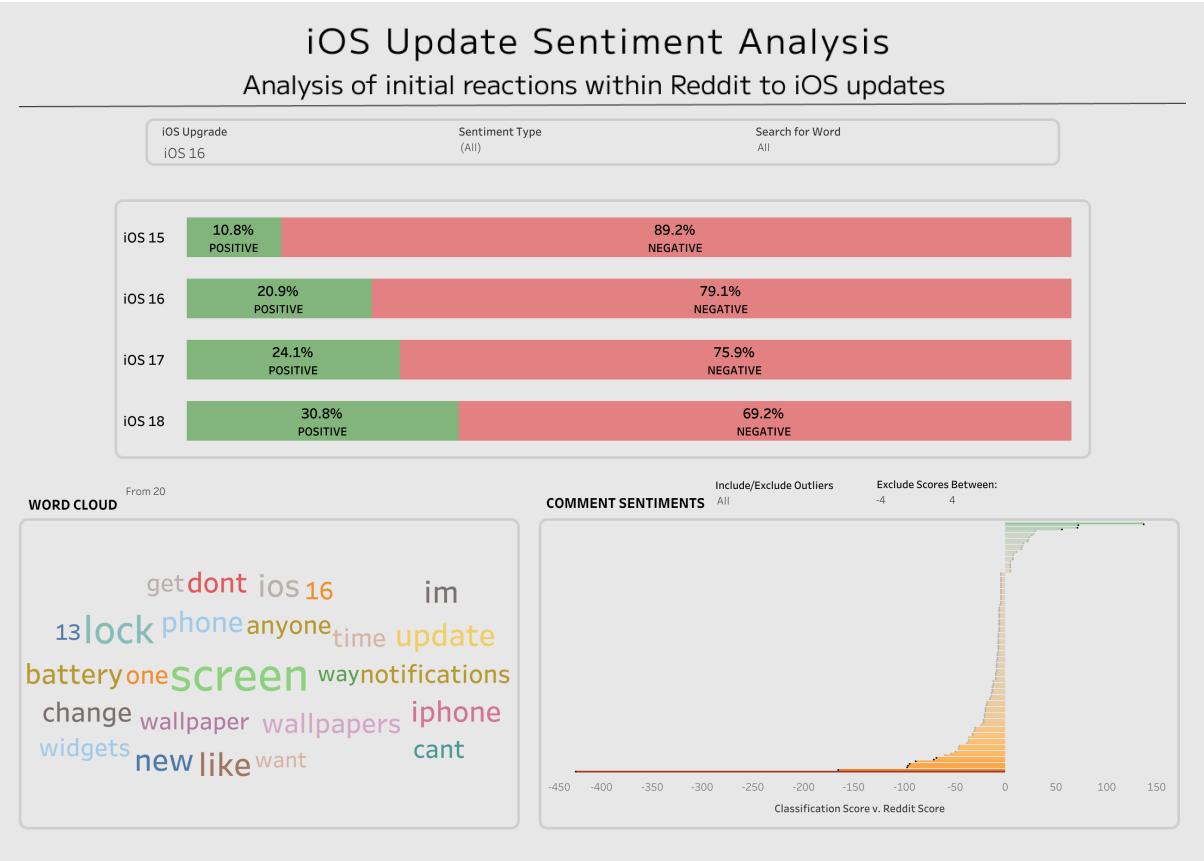


Figure 5: iOS Dashboard Filtered to iOS 16

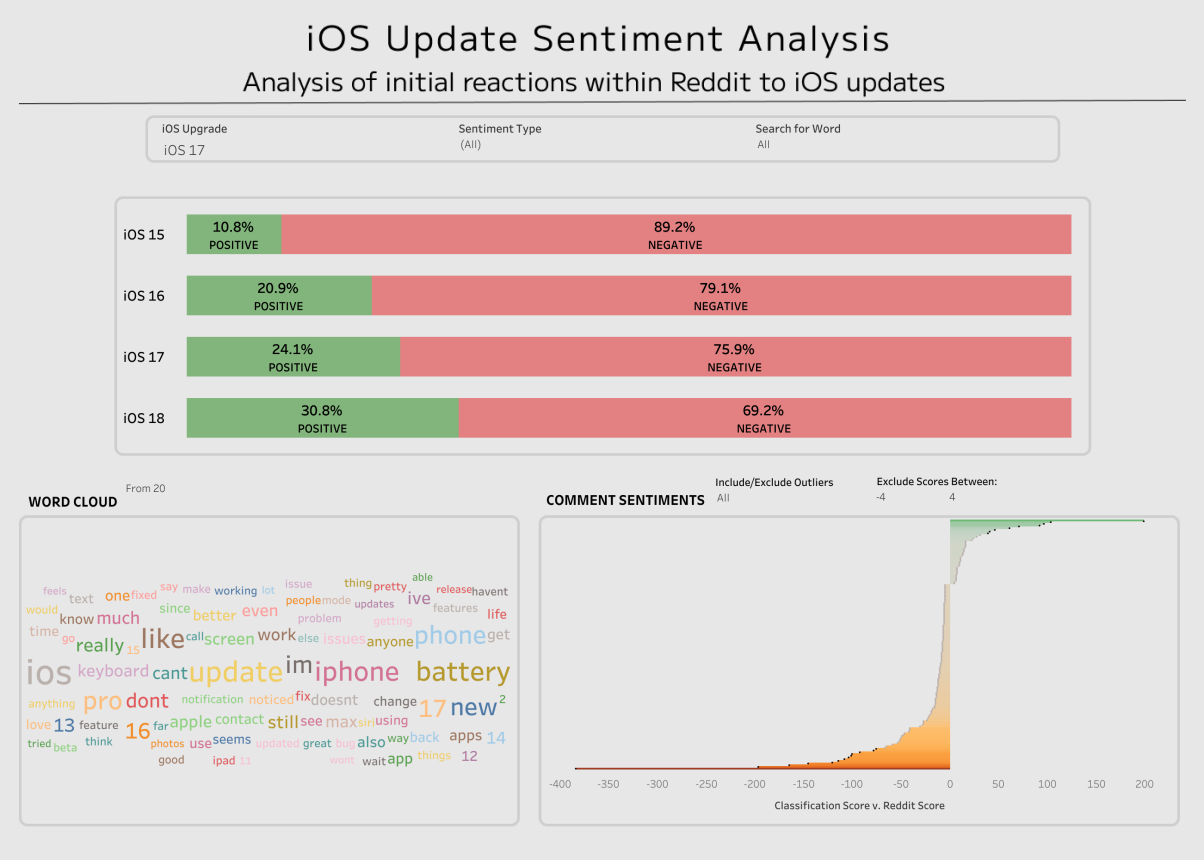


Figure 6: iOS Dashboard Filtered to iOS 17

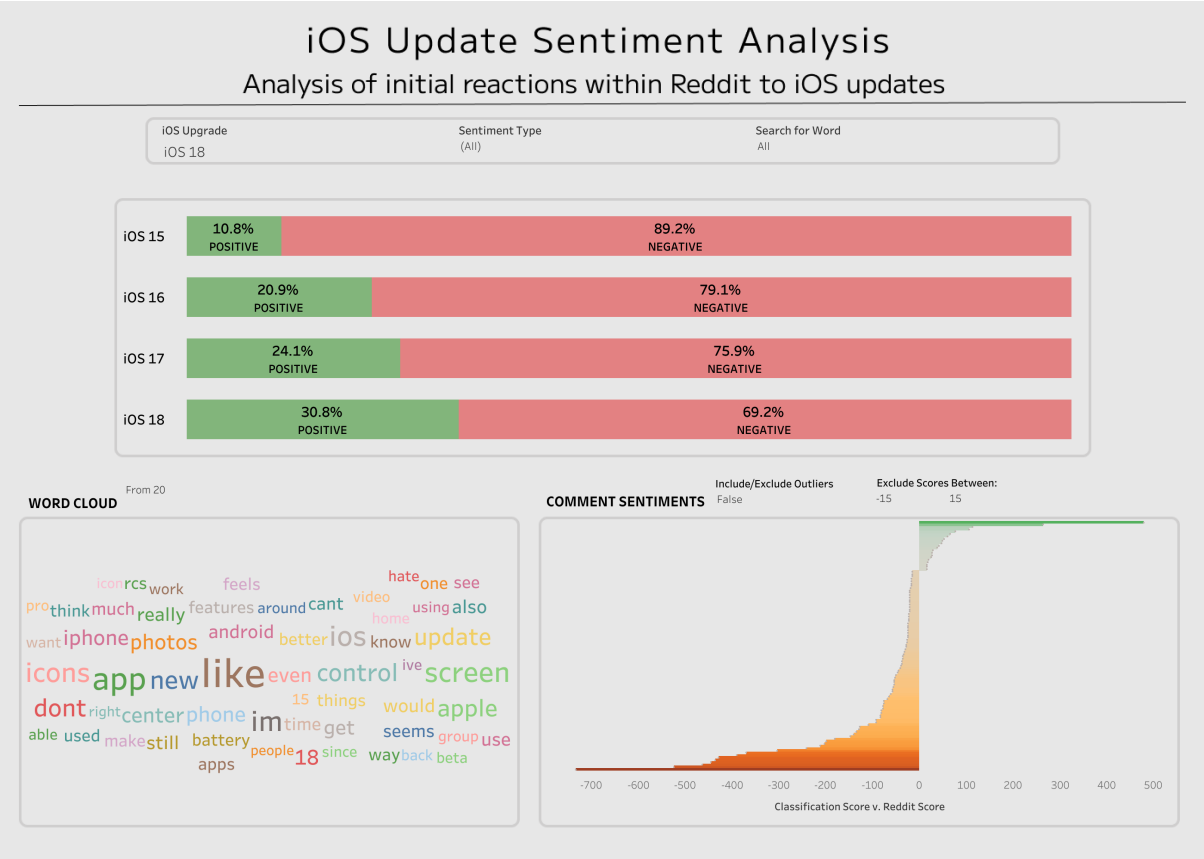


Figure 7: iOS Dashboard Filtered to iOS 18

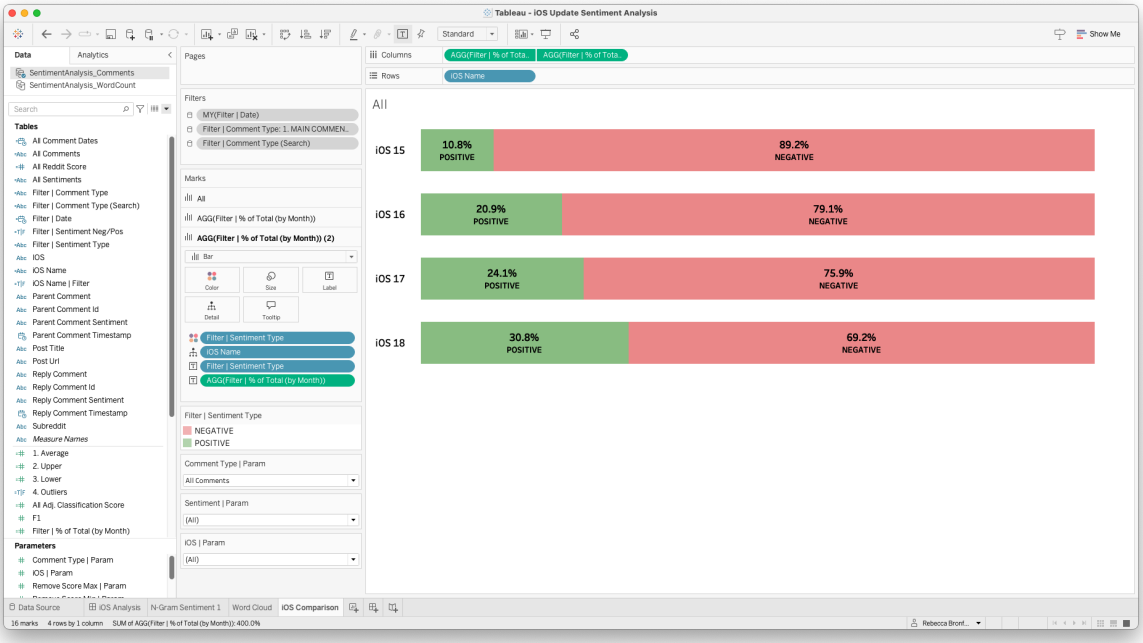


Figure 8: Tableau Backend - Creating the Comparison Bar Chart

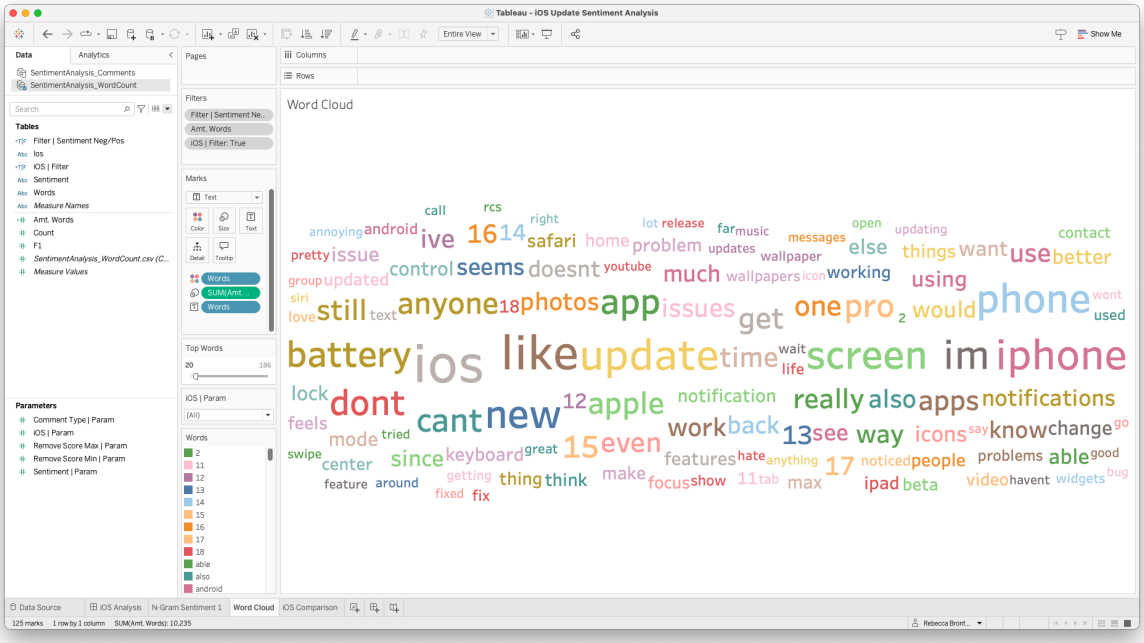


Figure 9: Tableau Backend - Creating the Word Cloud

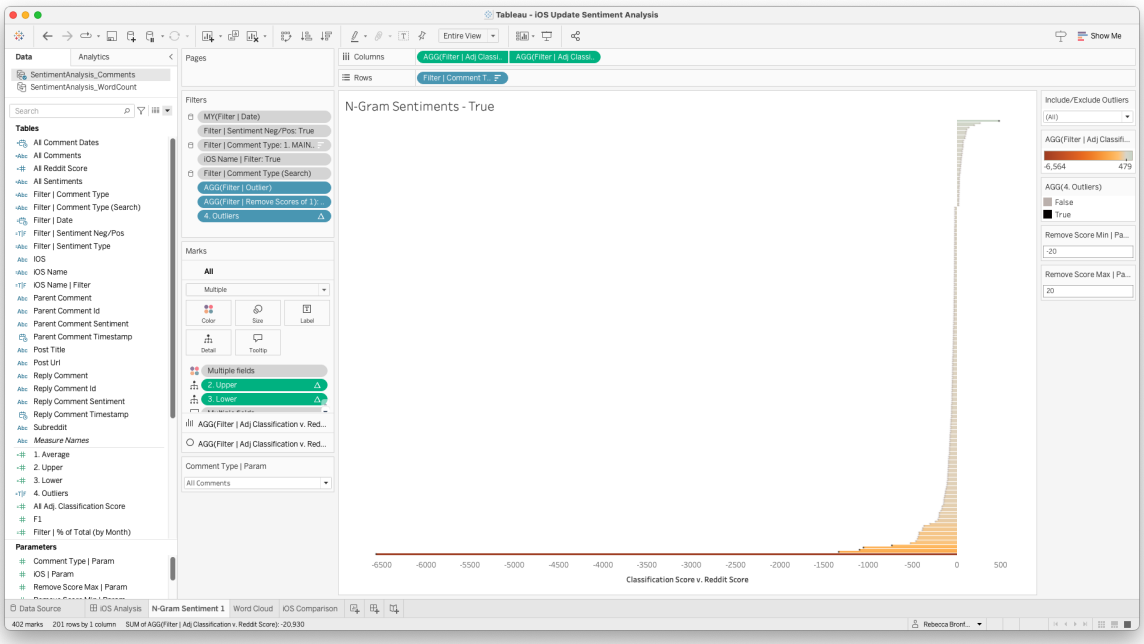


Figure 10: Tableau Backend - Creating the Comment-Level Sentiment graph