Review Summarization

Summarizing product reviews through Data scraping

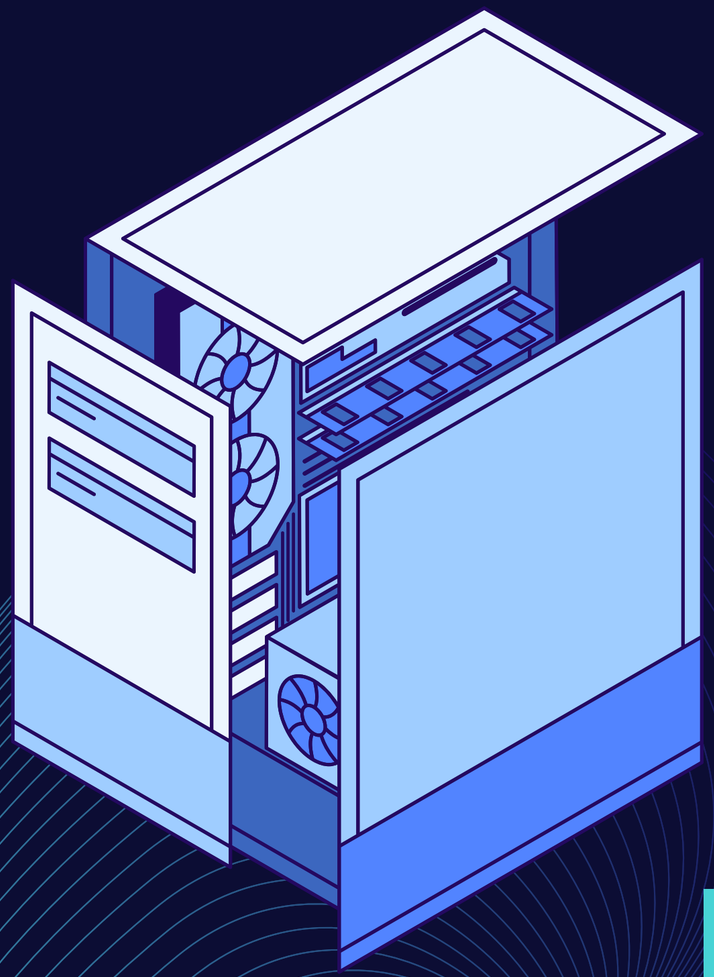
Mohana Misra

Harshita Singh Tanwar

Arya Gupta

What is it?

- Takes user input: brand/product name
- Web scraper scrapes data from Reddit
- Filters relevant reviews
- Performs sentiment analysis
- Summarization of reviews



Why is it needed?

- Tedious to process reviews manually
- Finding product or brand-specific customer reviews is time consuming
- Summarizing customer opinions across multiple sources is challenging
- Automates the scraping, analyzing, and summarizing reviews.

How does it work?

Data Collection

- Web scraper collects data from reddit based on user input using PRAW.
- Comments are fetched concurrently for faster performance

```python
# Function to scrape data from Reddit
def fetch_comments(submission, keyword):
    comments = []
    submission.comments.replace_more(limit=None)  # Fetch all comments at once
    for top_level_comment in submission.comments:
        if isinstance(top_level_comment, MoreComments):
            continue
        comments.append({
            'product_name': keyword,
            'review_content': top_level_comment.body,
            'rating': None,  # You can manually add ratings if needed
            'review_title': submission.title,
            'category': submission.subreddit.display_name  # Get the subreddit name
        })
    return comments

def scrape_reddit(keyword, subreddit='BuyItForLife', post_limit=100):
    posts = []

    # Search across the specified subreddit or all of Reddit
    submissions = reddit.subreddit(subreddit).search(keyword, limit=post_limit)

    # Use ThreadPoolExecutor for concurrent fetching of comments
    with ThreadPoolExecutor() as executor:
        # Map the fetch_comments function to each submission
        results = list(executor.map(lambda submission: fetch_comments(submission, keyword), s

    # Flatten the list of results
```

How does it work?

Preprocessing

- Collected data is processed converted to lowercase, followed by removal of stop words and punctuation

- Data structured into CSV format for easy processing.

```
    def regex_match(input_text, entity_list):
            # Create a pattern that allows for optional words between product terms
            pattern = re.sub(r"\s+", r".*", input_text.lower())  # "boat headphones" becomes "boat.*headphones"
            for entity in entity_list:
                if re.search(pattern, entity.lower()):
                    return entity
            return None
    matched_rows = []
    for idx, row in df.iterrows():
        product_name = row['productName']
        entity_list = [product_name]                # Use productName as the entity list
        # Apply regex matching
        match = regex_match(user_input, entity_list)
        if match:
            matched_rows.append(row)

    filtered_df = pd.DataFrame(matched_rows)        # Convert matched rows to a DataFrame
    if filtered_df.empty:
        print(f"No reviews found for {user_input} using regex matching.")
    else:
        print(f"Found {len(filtered_df)} reviews for {user_input} using regex matching.")
    return filtered_df
else:                                               # single word input recieved
    filtered_df = df[(df['productName'].str.contains(user_input, case=False, na=False)) |
      (df['productCategory'].str.contains(user_input, case=False, na=False))]
    if filtered_df.empty:
        print(f"No reviews found for {user_input}.")
    else:
        print(f"Found {len(filtered_df)} reviews for {user_input}.")
        return filtered_df
```

How does it work?

Sentiment Analysis and Summarization

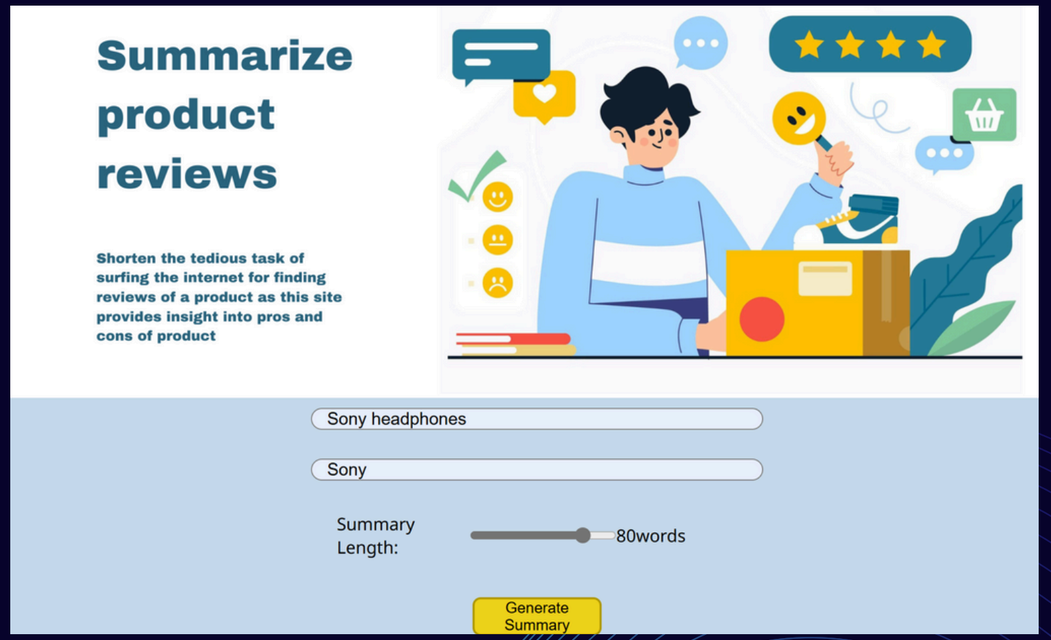
- Matches the input product or category using regex matching for single or multi-word input.
- TextBlob classifies reviews into Positive, Negative, or Neutral.
- Gemini API used for summarizing positive and negative reviews, with adjustable length.

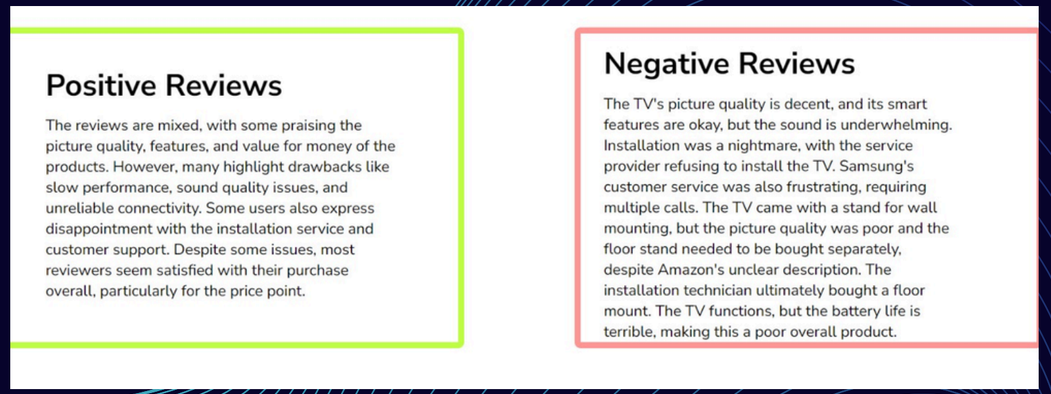
```python
# SENTIMENT ANALYSIS
def classify_sentiment(text):
        from textblob import TextBlob
        blob = TextBlob(text)
        polarity = blob.sentiment.polarity
        # Classify the sentiment
        if polarity > 0:
            return 'Positive'
        elif polarity < 0:
            return 'Negative'
        else:
            return 'Neutral'

# CORPUS SUMMARIZATION BASED ON INPUT LENGTH
def summarize_corpus(corpus_text, length):
    prompt = f"Summarize these reviews concisely in {length} words in a paragraph: {corpus_text}."
    response = model.generate_content(prompt)
    return response.text
```

User Interface

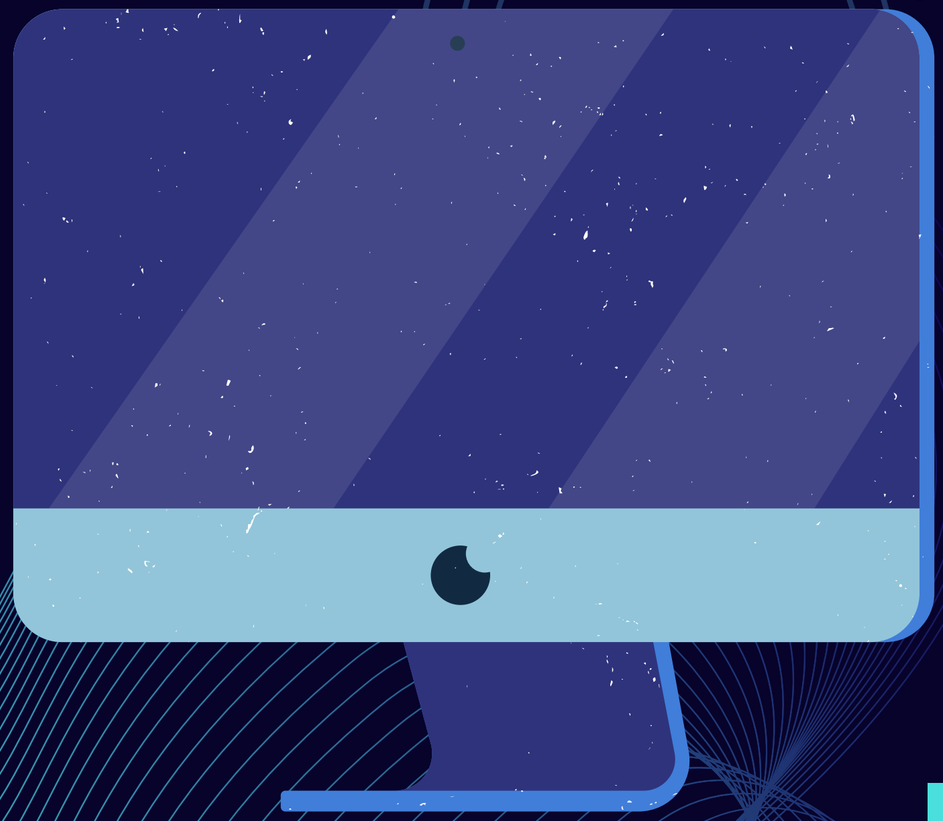
- User inputs: Product name/ Brand name (required) and subreddit name (optional).
- Sliding bar to adjust the lengths of output summaries.
- The positive and negative reviews are summarized





Future Work

- Implement keyword extraction to highlight top concerns.
- Optimize scraping speed
- Implement more advanced models for sentiment analysis



Thank You