NEW GOVERNMENT:

REDDITOR'S VIEW TOWARDS ANWAR IBRAHIM AS MALAYSIA'S 10TH PRIME MINISTER.

GROUP

NOCTUA

LECTURER NAME: PROF MADYA MOHD SHAHIZAN OTHMAN

OUR TEAM

NAME	MATRIC NO.
AFIF HAZMIE ARSYAD BIN AGUS	A20EC0176
AHMAD AIMAN HAFIZI BIN MUHAMMAD	A20EC0177
LUQMAN ARIFF BIN NOOR AZHAR	A20EC0202
MADINA SURAYA BINTI ZHARIN	A20EC0203
MUHAMMAD IMRAN HAKIMI BIN MOHD SHUKRI	A20EC0213

BACKGROUND OF STUDY

SOCIAL MEDIA

PLATFORM

Social media is a digital platform where people can connect, share content, and interact with each other

REDDIT

COMMUNITY

Reddit is a popular online community where users can join topic-specific forums to engage in discussions and share content

PROBLEM STATEMENT

1 PUBLIC PERCEPTION

Analyzing their opinions, attitudes, and beliefs about Anwar Ibrahim and understanding how they perceive his capabilities and suitability for the role of Prime Minister.

2 DECISION MAKING

Understanding how the opinions expressed by Redditors can shape public sentiment and potentially impact the decision-making process surrounding Anwar Ibrahim's political career and potential candidacy for the position of Prime Minister.

OBJECTIVE

OBJECTIVE 01

To determine whether the public has positive or negative sentiment on our Prime Minister.

OBJECTIVE 02

To train a model that can successfully identify a positive or negative comment on Reddit.

OBJECTIVE 03

To make a prediction on whether or not Anwar Ibrahim can sustain his job as a Prime Minister.

CHALLENGES & LIMITATIONS

SMALL AMOUNT OF DATA

In Reddit, it is a challenge to gain a large sets of data considering the base user is not as high as other platforms such as Twitter and Facebook.

HIGH NEUTRALITY

The nature of Reddit is like a forum. Most data are from comments which sometimes not targeted directly to the subject or topic.

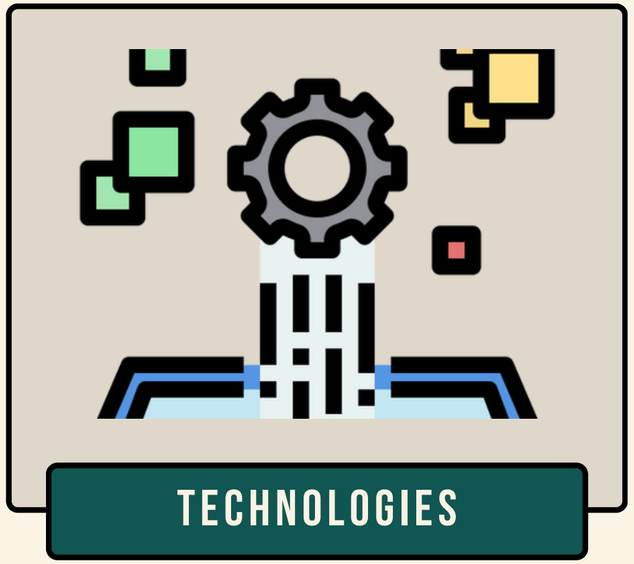
SARCASM

Reddit users use sarcastic texts most of the times. It is difficult to determine the sentiments based on the sarcastic context given.

SCOPE

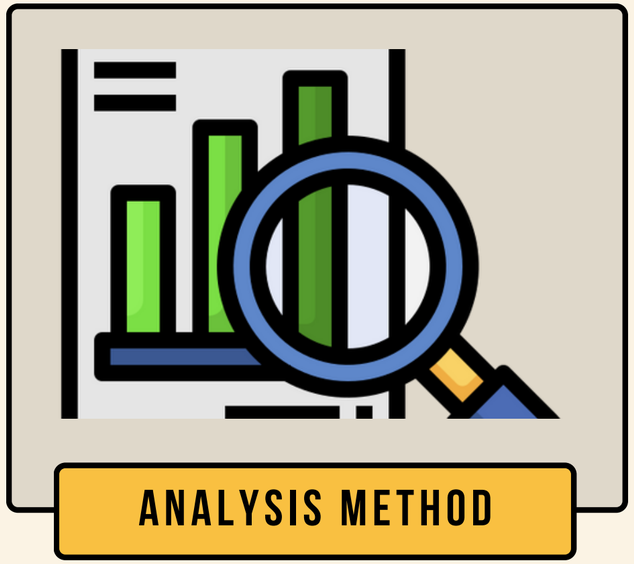
TARGET AUDIENCE

All reddit users who resides in Malaysia



TECHNOLOGIES

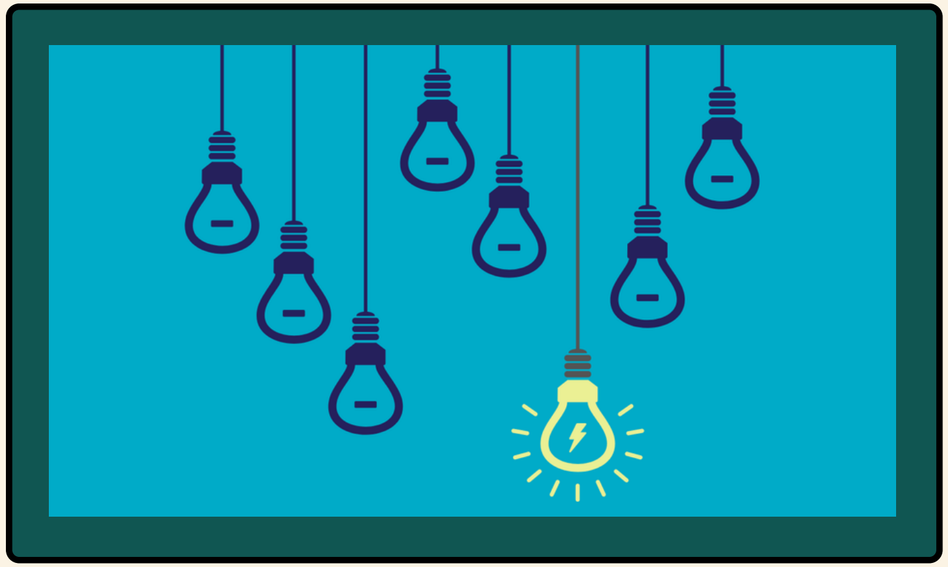
Implementation of API, PRAW library and utilization of Natural Language Processing (NLP) techniques



ANALYSIS METHOD

Desciptive statistics, topic modeling and sentiment analysis

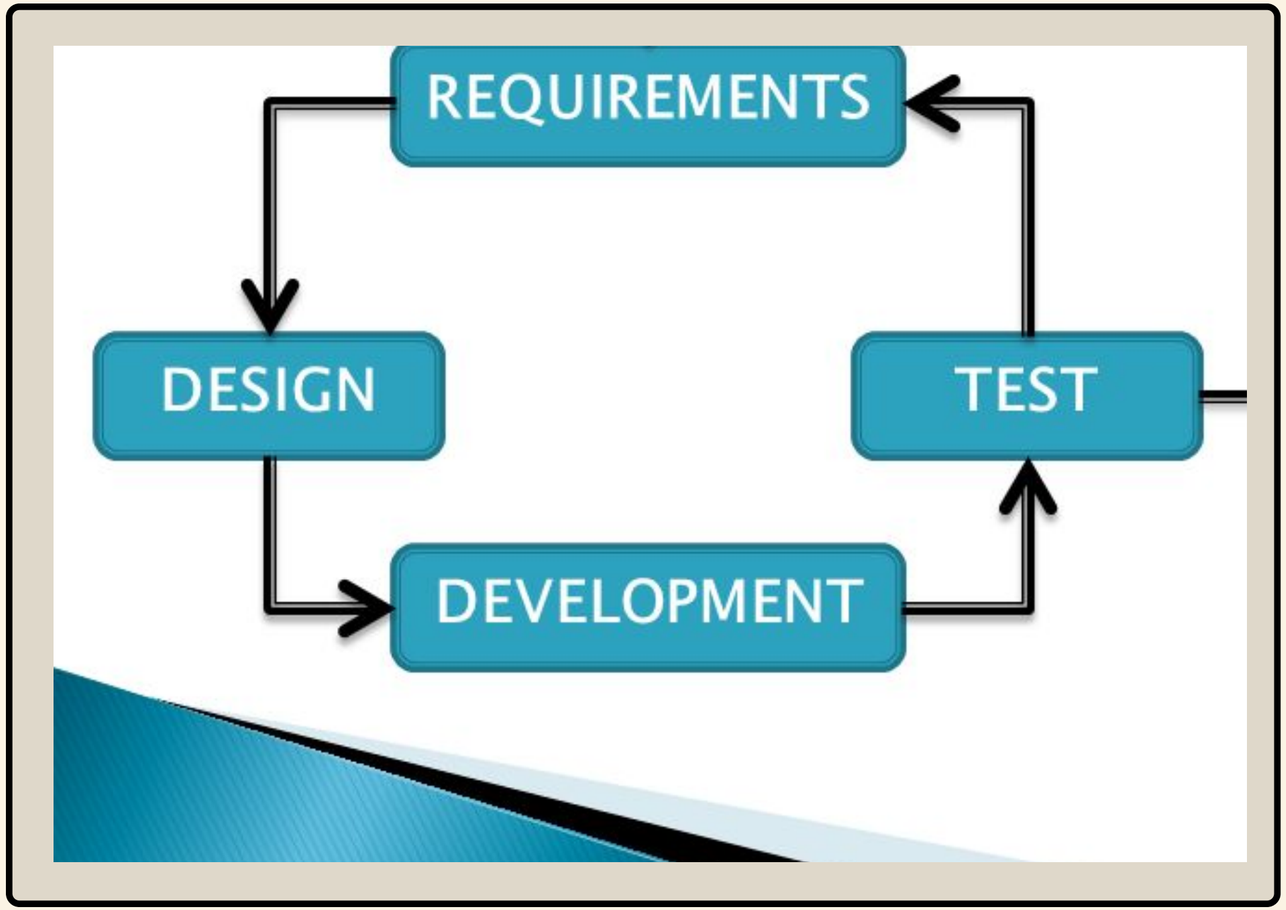
PROPOSED SOLUTION



Develop a visualization-based system that provides users with real-time insights and trends on discussions related to Anwar Ibrahim on Reddit

Allowing users to stay informed about the latest happenings, opinions, and sentiments surrounding his political career and Prime Minister prospects.

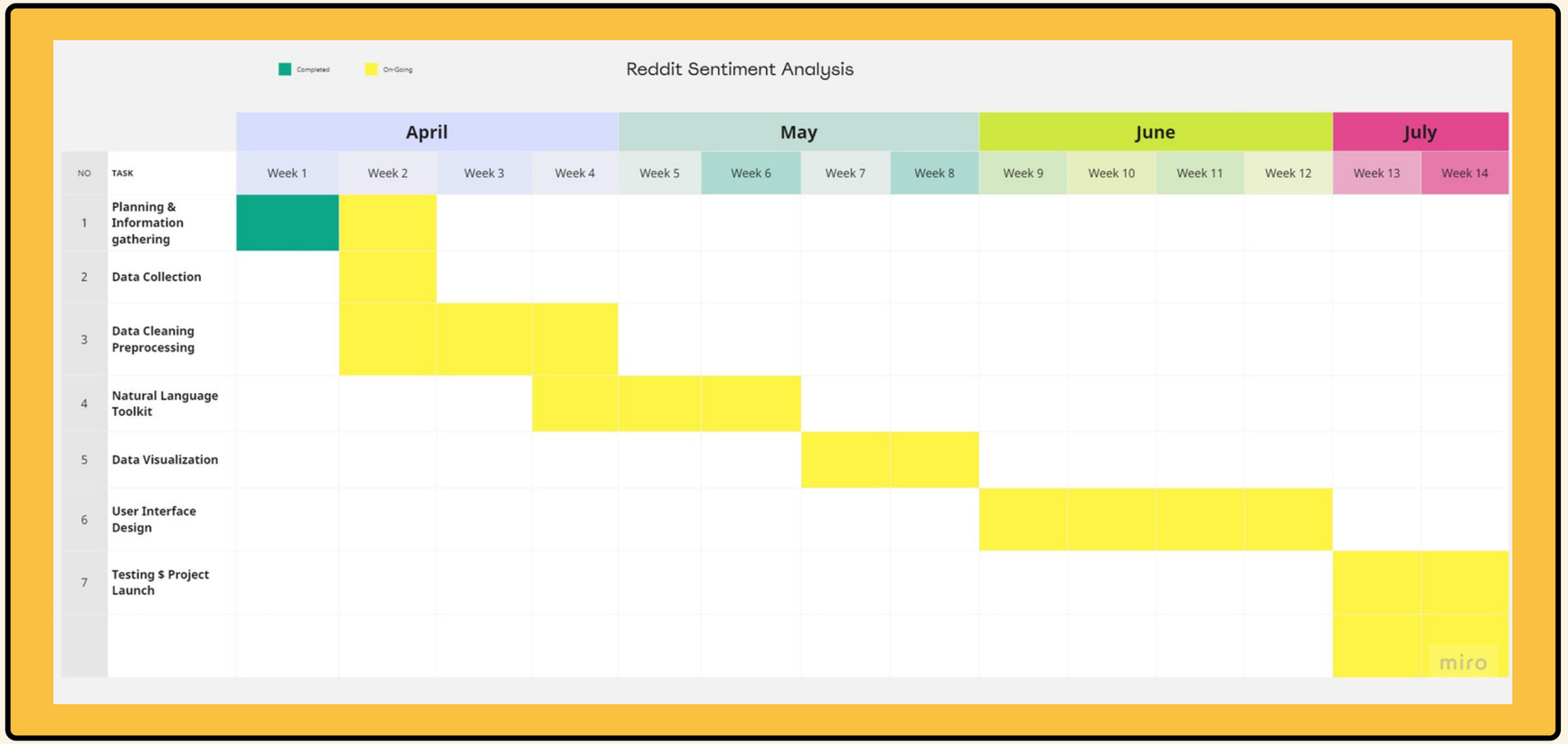
METHODOLOGY



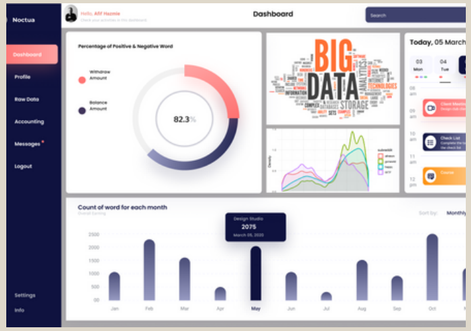
The waterfall methodology is a linear and sequential approach to software development, consisting of five distinct phases: requirements, analysis, design, implementation, and testing.

REQUIREMENT PHASE

The Gantt Chart provides a visual representation of the project timeline and helps ensure that all necessary activities are accounted for and completed in a timely manner.



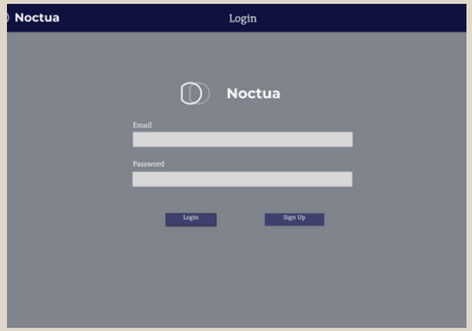
REQUIREMENT PHASE



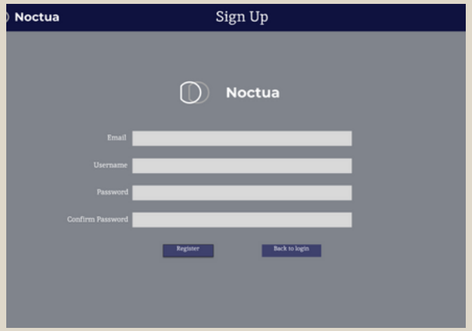
DASHBOARD

DATA TABLE

EDIT PROFILE



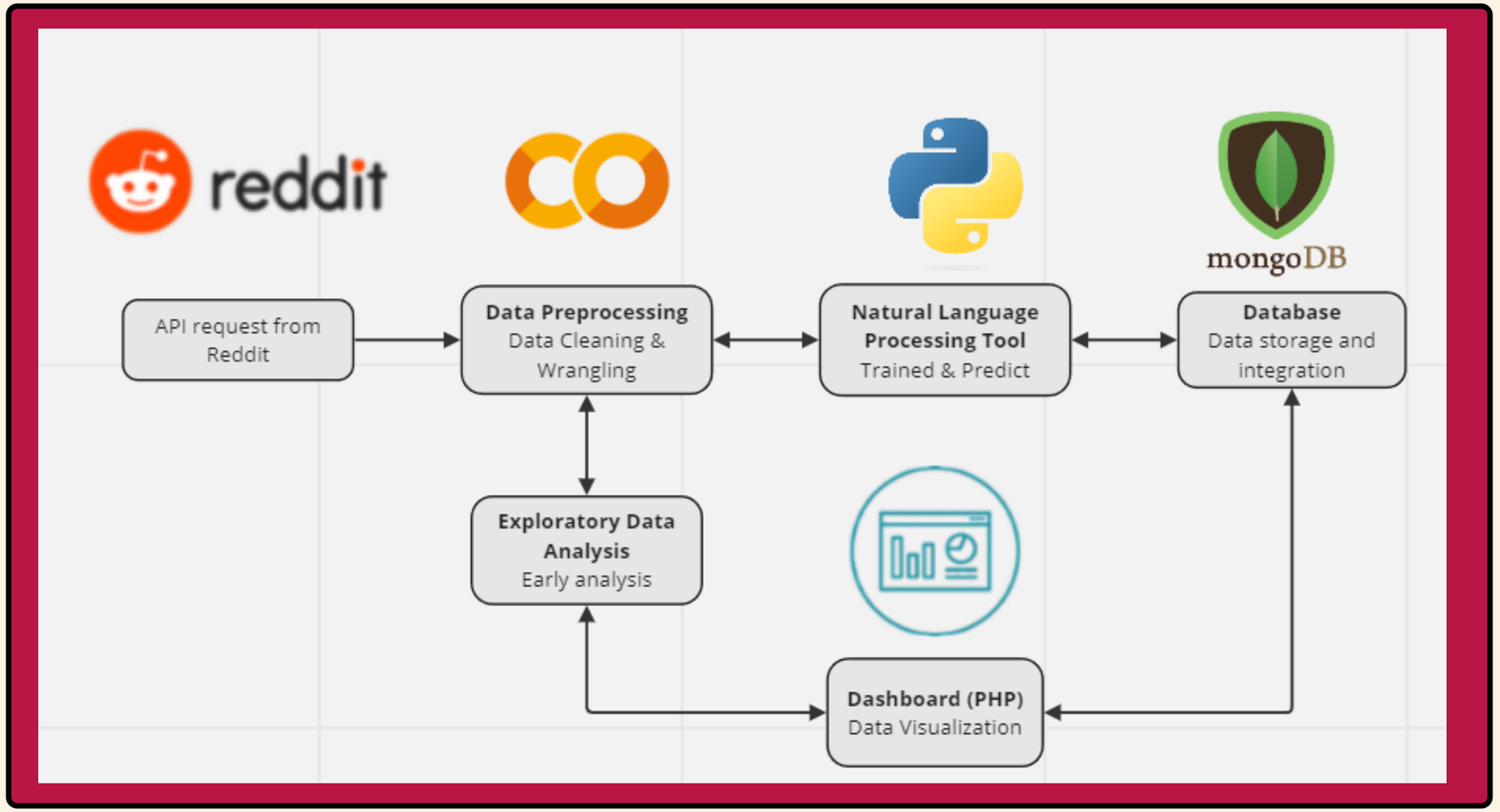
LOGIN



SIGN UP

DEVELOPMENT PHASE

During the implementation phase, data is collected from Reddit using the PRAW library, cleaned to remove irrelevant information, analyzed using descriptive statistics, and explored with machine learning algorithms to predict sentiment towards Anwar Ibrahim. The results are then visualized using data visualization tools for clear presentation.



BACKEND DEVELOPMENT

```python
import praw
import pandas as pd
import datetime as dt

reddit = praw.Reddit(
    client_id="0X1ATbYQOxzI_T4YExOPCA",
    client_secret="XAKrWmKM2kwTjpjJQ8ffrC0C8oJljQ",
    user_agent="SentimentAnalysis",
    check_for_async=False
)

subreddits = ['malaysia', 'Bolehland', 'MalaysiaPolitics', 'worldnews', 'economicCollapse']
queries = ['Anwar Ibrahim']
start_date = dt.datetime(2022, 6, 1)  # Specify the start date of the desired date range
end_date = dt.datetime(2023, 6, 1)  # Specify the end date of the desired date range

def get_date(created):
    return dt.datetime.fromtimestamp(int(created))

combined_data = []

for query in queries:
    for subreddit_name in subreddits:
        subreddit = reddit.subreddit(subreddit_name)

        for submission in subreddit.search(query, sort="top", time_filter='all', limit=300):
            submission_date = dt.datetime.fromtimestamp(int(submission.created_utc))
            if start_date <= submission_date <= end_date and submission.score >= 10:
                combined_data.append({
                    "type": "post",
                    "title": submission.title,
                    "score": submission.score,
                    "id": submission.id,
                    "comms_num": submission.num_comments,
                    "created": submission.created_utc,
                    "body": submission.selftext
                })

                submission.comments.replace_more(limit=1)
                for comment in submission.comments.list():
                    if comment.score >= 5:  # Add a filter for comment score
                        combined_data.append({
                            "type": "comment",
                            "title": "",
                            "score": comment.score,
                            "id": comment.id,
                            "comms_num": "",
                            "created": comment.created_utc,
                            "body": comment.body
                        })

combined_df = pd.DataFrame(combined_data)
combined_df["timestamp"] = combined_df["created"].apply(get_date)
combined_df.to_csv("combined_data.csv", index=False)
```

DATA COLLECTION

Data Scraping: We utilized the PRAW (Python Reddit API Wrapper) library to scrape relevant data from multiple subreddits, including "malaysia," "Bolehland," "MalaysiaPolitics," "worldnews," and "economicCollapse." We retrieved posts and comments containing the query "Anwar Ibrahim" within a specified date range and with certain score thresholds.

	A	B	C	D	E	F	G	H
1	type	title	score	id	comms_n	created	body	timestamp
2	post	Anwar Ibr	3415	z3bgy9	631	1.67E+09		24/11/2022 13:34
3	comment		594	ixkz4c4		1.67E+09	someone	24/11/2022 13:43
4	comment		337	ixkyspj		1.67E+09	No sign of	24/11/2022 13:39
5	comment		249	ixkyg7r		1.67E+09	Welp	24/11/2022 13:36
6	comment		812	ixkycnx		1.67E+09	He finally	24/11/2022 13:35
7	comment		511	ixkz0ue		1.67E+09	PRU 14: A	24/11/2022 13:42
8	comment		140	ixkz6fy		1.67E+09	I hope	24/11/2022 13:43
9	comment		448	ixkyc86		1.67E+09	HOLY SHIT	24/11/2022 13:35
10	comment		89	ixkzm49		1.67E+09	I haven't	24/11/2022 13:48
11	comment		384	ixkypo1		1.67E+09	The real	24/11/2022 13:38

DATA PROCESSING

Data Preprocessing: We performed several preprocessing steps on the collected data to clean and prepare it for analysis. This included removing URLs, emojis, special characters, and numbers. We converted the text to lowercase, tokenized it, removed stopwords, and applied lemmatization to normalize the words. The preprocessed text was then saved for further analysis.

A	B	C	D
preprocessed_text	body	timestamp	sentiment
someone pls check mahat	someone pls check on Ma	24/11/2022 13:43	neutral
sign unity government pn	No sign of unity governm	24/11/2022 13:39	neutral
welp finally epic opening e	Welp finally, the epic	24/11/2022 13:36	neutral
finally got number	He finally got the number	24/11/2022 13:35	neutral
pru new hope langkah she	PRU 14: A New Hope	24/11/2022 13:42	neutral
hope outward looking ma	I hope this is a more	24/11/2022 13:43	neutral
holy shit done man finally	HOLY SHIT, HE'S DONE IT	24/11/2022 13:35	positive
felt feeling since ph ge beg	I haven't felt this feeling	24/11/2022 13:48	positive
real battle begin provide s	The real battle begin	24/11/2022 13:38	neutral

```python
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Load the combined data from the CSV file
combined_df = pd.read_csv("combined_data.csv")

# Remove null and empty data
combined_df.dropna(subset=["body"], inplace=True)
combined_df = combined_df[combined_df["body"].str.strip().astype(bool)]

# Download Indonesian stopwords
nltk.download('stopwords')
indonesian_stopwords = set(stopwords.words('indonesian'))

# Define a function for data cleaning and preprocessing
def preprocess_text(text):
    # Remove URLs
    text = re.sub(r"http\S+|www\S+|https\S+", "", text)
    text = re.sub(r"(?i)\b(gif|giphy)\S+", "", text)  # Remove

    # Remove emojis
    text = re.sub(r"\\x\S+|\\u\S+|\\U\S+|\\n", "", text)

    # Remove special characters and numbers, excluding "pas"
    text = re.sub(r"[^a-zA-Z\s]|(?<!\w)pas(?!\w)", " ", text)

    # Convert to lowercase
    text = text.lower()

    # Check if the text is empty after preprocessing
    if not text.strip():
        return None

    # Tokenize the text
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words("english")) | indonesian_stopwords
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize the words
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Join the tokens back into a single string
    preprocessed_text = " ".join(tokens)

    return preprocessed_text

# Apply the preprocessing function to the "body" column
combined_df["preprocessed_text"] = combined_df["body"].apply(preprocess_text)

# Select relevant columns
columns_to_keep = ["preprocessed_text", "body", "timestamp"]
combined_df = combined_df[columns_to_keep]

# Remove rows with null values in the "preprocessed_text" column
combined_df.dropna(subset=["preprocessed_text"], inplace=True)

# Save the preprocessed data to a new CSV file
combined_df.to_csv("preprocessed_data5.csv", index=False)

print("Data preprocessing complete!")
```

DATA SPLITTING

```python
from sklearn.model_selection import train_test_split

# Load the preprocessed data from the CSV file
preprocessed_df = pd.read_csv("preprocessed_data4.csv")

# Split the data into training set and test set
train_df, test_df = train_test_split(preprocessed_df, test_size=0.2, random_state=42)

# Save the training set and test set to separate CSV files
train_df.to_csv("train_data.csv", index=False)
test_df.to_csv("test_data.csv", index=False)

print("Data splitting complete!")
```

Data Splitting: We split the preprocessed data into training and test sets using the 'train_test_split' function from scikit-learn. This allowed you to have separate data for training the sentiment analysis model and evaluating its performance.

TEXT REPRESENTATION

```python
from sklearn.feature_extraction.text import CountVectorizer

# Load the training data from the CSV file
train_df = pd.read_csv("train_data.csv")

# Drop rows with missing values in the preprocessed text
train_df.dropna(subset=["preprocessed_text"], inplace=True)

# Extract the preprocessed text and corresponding labels
X_train = train_df["preprocessed_text"].values
y_train = train_df["sentiment"].values

# Initialize the CountVectorizer
vectorizer = CountVectorizer()

# Fit and transform the training data
X_train_vectorized = vectorizer.fit_transform(X_train)

# Print the vocabulary size
print("Vocabulary Size:", len(vectorizer.vocabulary_))

# Print the first few feature names
print("Feature Names:", vectorizer.get_feature_names_out()[:10])

# Print the vectorized training data
print("Vectorized Training Data:\n", X_train_vectorized.toarray())
```

Feature Extraction: We utilized the CountVectorizer from scikit-learn to convert the preprocessed text into a numerical representation suitable for training a machine learning model. This process created a vocabulary of unique words and generated feature vectors representing the occurrence of those words in the text samples.

```
Vocabulary Size: 8094
Feature Names: ['aah' 'abah' 'abahcow' 'abandon' 'abandoned' 'abandoning' 'abang'
 'abbreviation' 'abducted' 'abduction']
Vectorized Training Data:
 [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

```python
# Load the test data from the CSV file
test_df = pd.read_csv("test_data.csv")

# Drop rows with missing values in the preprocessed text
test_df.dropna(subset=["preprocessed_text"], inplace=True

# Extract the preprocessed text from the test data
X_test = test_df["preprocessed_text"].values

# Transform the test data using the trained vectorizer
X_test_vectorized = vectorizer.transform(X_test)

# Print the vocabulary size
print("Vocabulary Size:", len(vectorizer.vocabulary_))

# Print the first few feature names
print("Feature Names:", vectorizer.get_feature_names_out(

# Print the vectorized test data
print("Vectorized Test Data:\n", X_test_vectorized.toarra
```

MODEL TRAINING

Model Training: We employed logistic regression as the classification model for sentiment analysis. The logistic regression model was trained using the vectorized training data obtained from the CountVectorizer. This involved fitting the model to the training data and optimizing the model's parameters.

```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
import pandas as pd

# Load the training data from the CSV file
train_df = pd.read_csv("train_data.csv")

# Drop rows with missing values in the preprocessed text
train_df.dropna(subset=["preprocessed_text"], inplace=True)

# Extract the preprocessed text and corresponding labels
X_train = train_df["preprocessed_text"].values
y_train = train_df["sentiment"].values

# Initialize the CountVectorizer
vectorizer = CountVectorizer()

# Fit and transform the training data
X_train_vectorized = vectorizer.fit_transform(X_train)

# Load the test data from the CSV file
test_df = pd.read_csv("test_data.csv")

# Drop rows with missing values in the preprocessed text
test_df.dropna(subset=["preprocessed_text"], inplace=True)

# Extract the preprocessed text from the test data
X_test = test_df["preprocessed_text"].values

# Transform the test data using the trained vectorizer
X_test_vectorized = vectorizer.transform(X_test)
```

MODEL EVALUATION

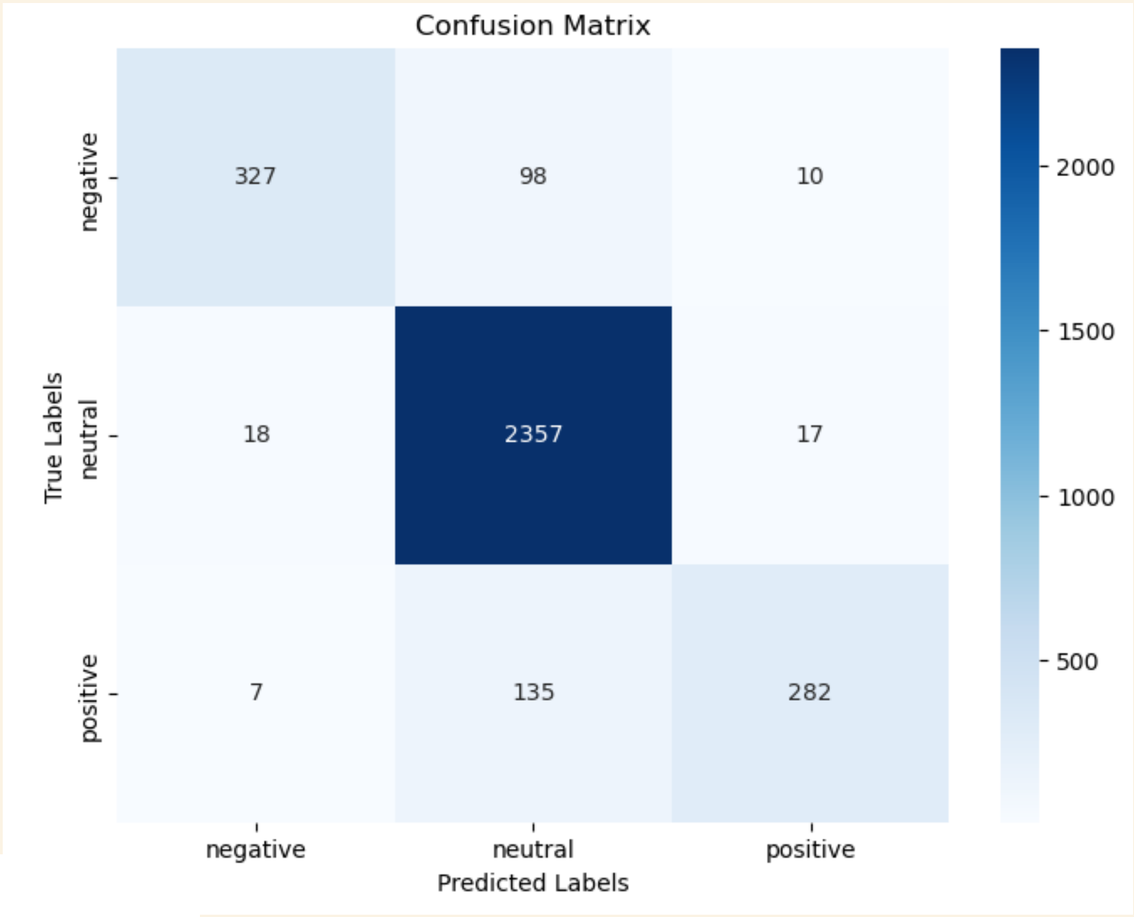
Model Evaluation: **We evaluated the trained sentiment analysis model using the test data. The accuracy score was calculated to measure the model's overall performance. Additionally, a classification report was generated, providing more detailed metrics such as precision, recall, and F1-score for individual sentiment classes.**

```
Test Accuracy: 0.7158218125960062
Classification Report:
              precision    recall  f1-score   support

    negative       0.33      0.15      0.20        82
     neutral       0.77      0.93      0.84       473
    positive       0.35      0.15      0.21        96

    accuracy                           0.72       651
   macro avg       0.48      0.41      0.42       651
weighted avg       0.65      0.72      0.67       651
```

```python
# Initialize the LogisticRegression model
model = LogisticRegression(max_iter=1000)

# Train the model
model.fit(X_train_vectorized, y_train)

# Predict the sentiment labels for the test data
y_pred = model.predict(X_test_vectorized)

# Evaluate the model's performance on the test data
accuracy = accuracy_score(test_df["sentiment"], y_pred)
print("Test Accuracy:", accuracy)

# Generate the classification report
report = classification_report(test_df["sentiment"], y_pred)
print("Classification Report:")
print(report)
```

MODEL EVALUATION

```python
from sklearn.metrics import confusion_matrix

# Load the test data from the CSV file
test_df = pd.read_csv("new_test_data.csv")

# Drop rows with missing values in the preprocessed text
test_df.dropna(subset=["preprocessed_text"], inplace=True)

# Extract the preprocessed text from the test data
X_test = test_df["preprocessed_text"].values

# Transform the test data using the trained vectorizer
X_test_vectorized = vectorizer.transform(X_test)

# Make predictions on the test data using the trained model
y_pred = model.predict(X_test_vectorized)

# Generate the confusion matrix
cm = confusion_matrix(test_df["sentiment"], y_pred)

# Create a heatmap of the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d", xticklabels=model.classes_, yticklabels=model.classes_)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()

# Evaluate the model's performance on the test data
accuracy = accuracy_score(test_df["sentiment"], y_pred)
print("Test Accuracy:", accuracy)

# Generate the classification report for the test data
report = classification_report(test_df["sentiment"], y_pred)
print("Classification Report:")
print(report)
```



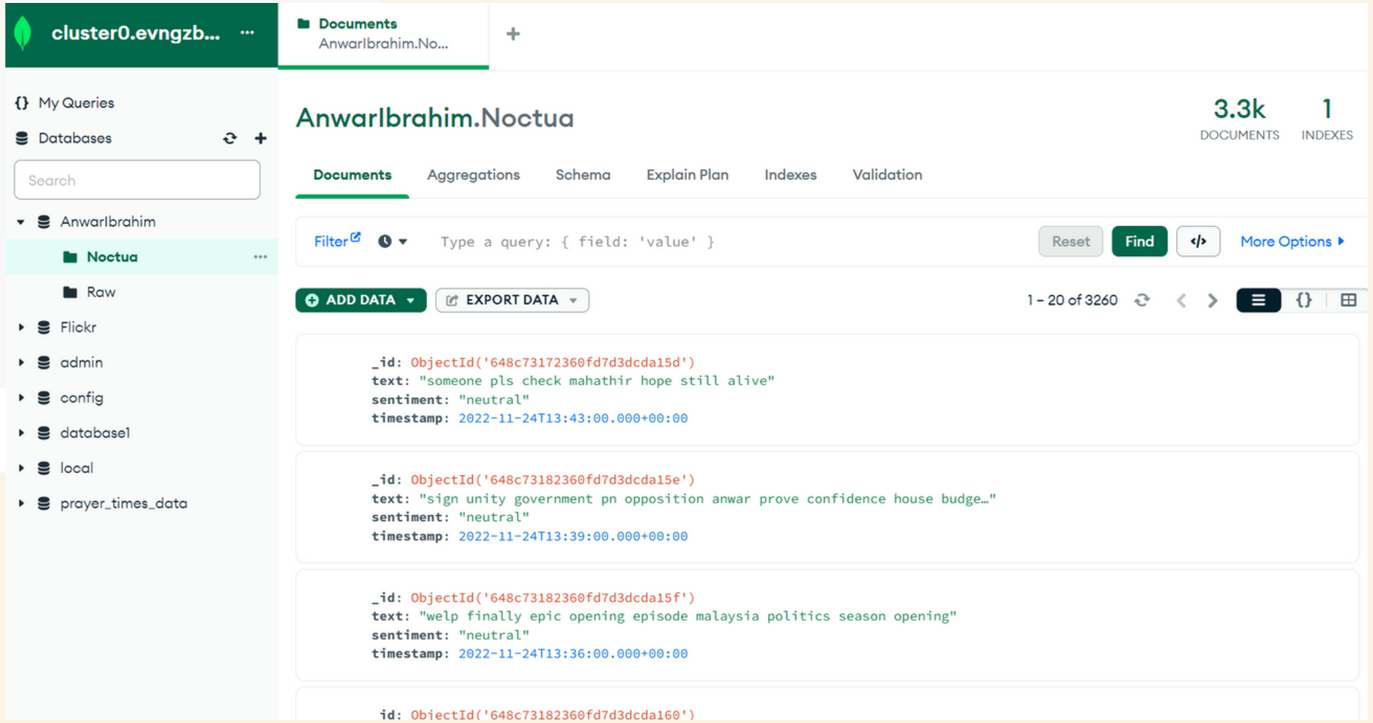

```
Test Accuracy: 0.9123346662565365
Classification Report:
              precision    recall  f1-score   support

    negative       0.93      0.75      0.83       435
     neutral       0.91      0.99      0.95      2392
    positive       0.91      0.67      0.77       424

    accuracy                           0.91      3251
   macro avg       0.92      0.80      0.85      3251
weighted avg       0.91      0.91      0.91      3251
```

MONGODB INTEGRATION

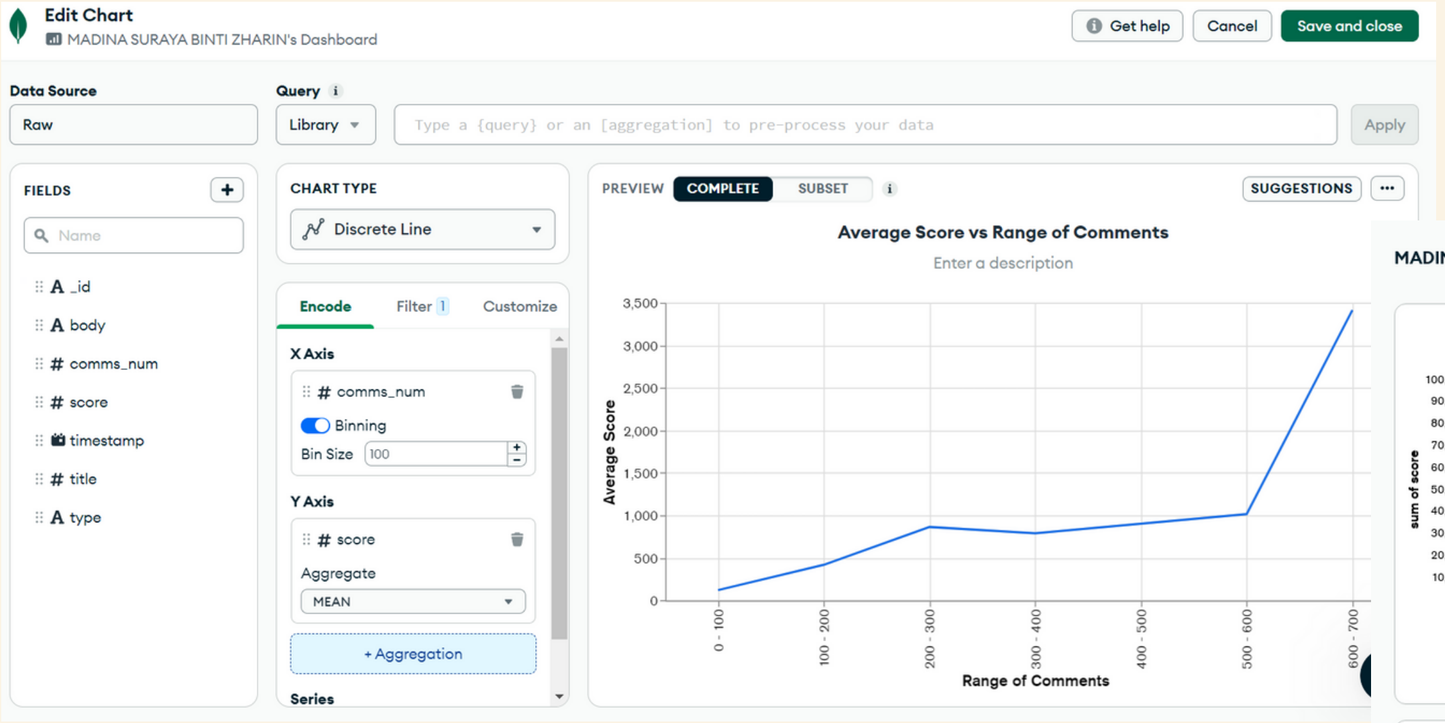
```
[ ] !pip install pymongo

    Requirement already satisfied: pymongo in c:\programdata\anaconda3\lib\site-packages (4.3.3)
    Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in c:\programdata\anaconda3\lib\site-packages (from pymongo) (2.3.0)
```

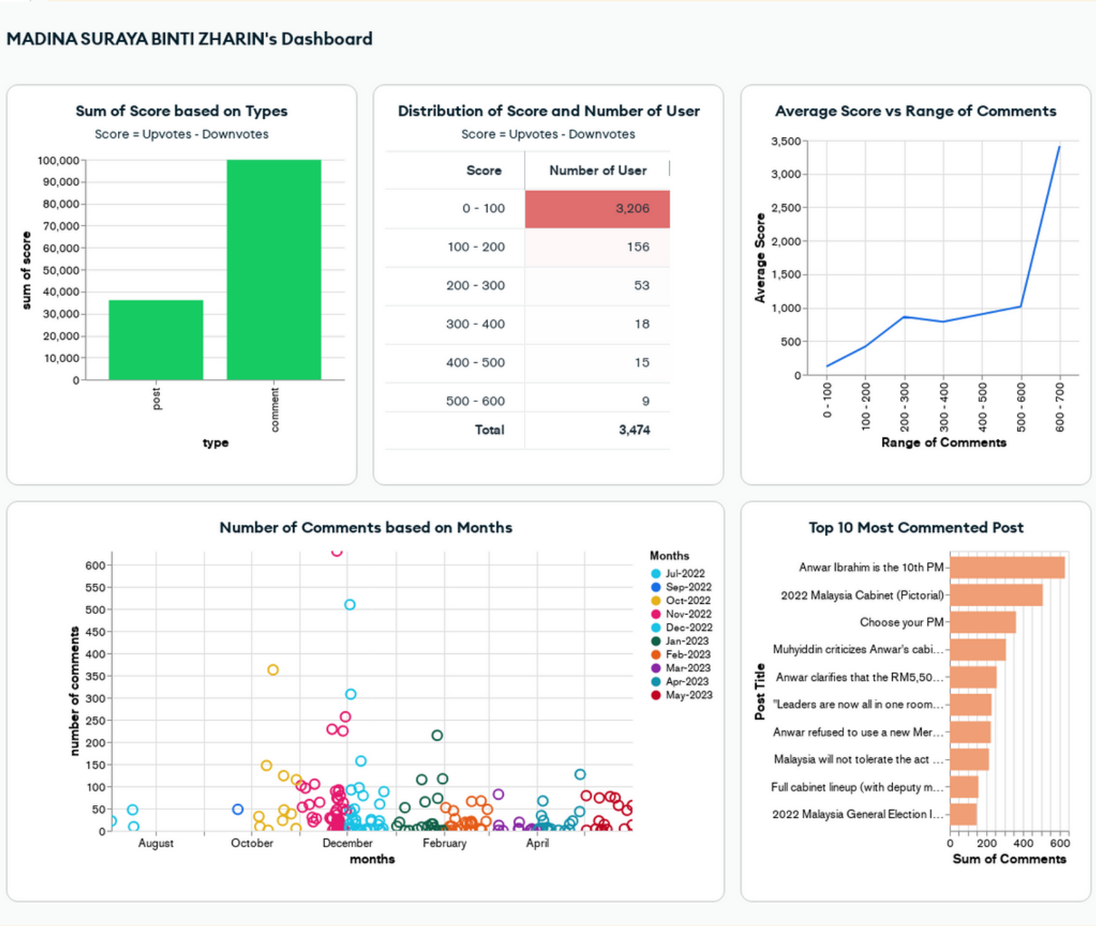
```
[ ] import pymongo
    import pandas as pd

    preprocessed_df = pd.read_csv("preprocessed_data4.csv")

    # Connect to MongoDB
    client = pymongo.MongoClient("mongodb+srv://muhdimranh:123@sentimentanalysis.5esk2hq.mongodb.net/")
    database = client["AnwarIbrahim"]
    collection = database["Noctua"]

    # Iterate over each row in the DataFrame and insert the data into the collection
    for index, row in preprocessed_df.iterrows():
        document = {
            "text": row["preprocessed_text"],
            "sentiment": row["sentiment"],
            "timestamp": pd.to_datetime(row["timestamp"])
        }
        collection.insert_one(document)

    # Close the MongoDB connection
    client.close()

    print('Added successfully!')

    Added successfully!
```

Data Storage: **We store data into MongoDB Compass by creating database and its collection. Data in MongoDB will be stored in JSON format.**



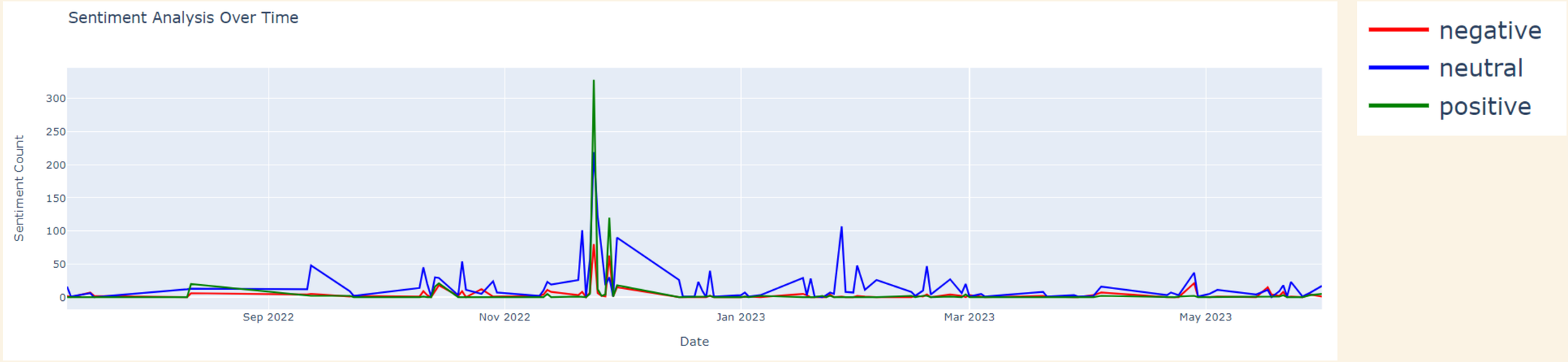
MONGODB INTEGRATION





Utilize MongoDB features: We utilize MongoDB Charts to visualize in-depth insights of the data collected.

ANALYSIS FINDINGS



From the chart above, we found a massive influx of positive comments towards DSAI during 24 Nov 2022 which is the day he was sworn in as the 10th PM. It gathered more than 300 positive comments that day alone which is an increase of almost 300 times compared to the same time of the previous month. However, we also noticed that there was also a surge of negative comments on the day of the swear in. This could imply that, of course there were more negative comments during the day of the swear in, but it is completely overshadowed by the massive influx of positive comments he received that day. Throughout our findings, from July 2022 until March 2023, excluding the day of the swear in, DSAI received mostly neutral comments, which could imply that people simply tolerate him rather than just liking or hating him.

ANALYSIS FINDINGS

Sentiment	Example Texts	
Neutral	Positive	Negative
ıthir hope still alive	holy shit done man finally got dream job	found anal bead dna sprinkled top
n opposition anwar prove	felt feeling since ph ge beginning lot also know long last short celebrate alhamdullilah	see go high hope high disappointment like last time
episode malaysia politics	wait legit dreaming go malaysia	found anal bead dna sprinkled top
	tear eye madman actually	see go high hope high disappointment like last time
ıeraton empire strike back	alhamdulillah syabas dsai rest political party agreeing form first malaysian unity government	good point always large amount people like wait year clean mess accumulated year old literally useles

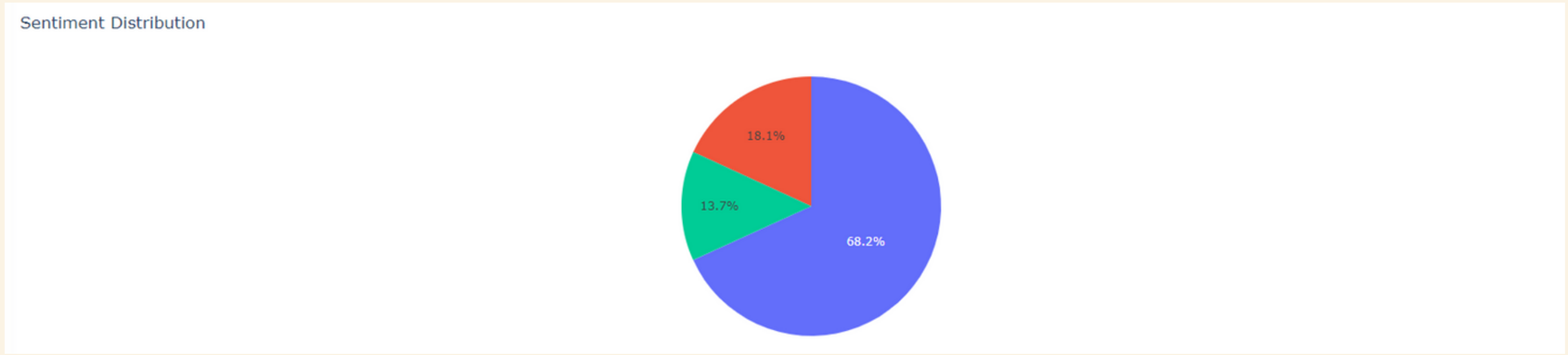
Here we have an example of texts along with the sentiments.

Neutral - "Sheraton empire strikes back". This comment is considered neutral because without any context it would simply make no sense. There aren't any praises nor criticism in the comment.

Positive - "Alhamdulillah syabas dsai rest political party agreeing form first malaysian unity government". This comment is considered positive as it contains the word "Alhamdulillah" which is a sign of gratitude along with "syabas" which is congratulations in Malay.

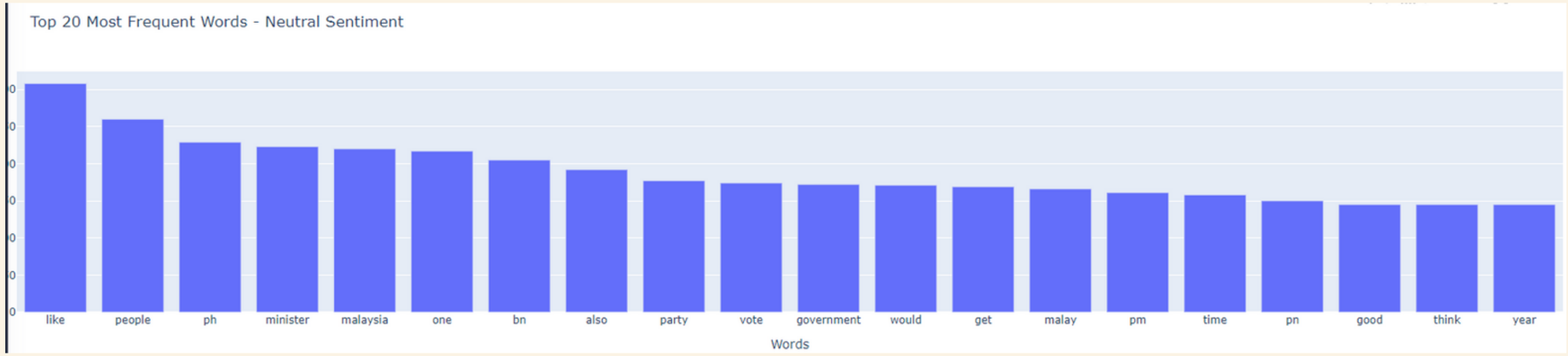
Negative - "Found anal bead DNA sprinkled top". This comment is considered negative because it contains sexual words such as "anal" which could be related to DSAI scandal back in the late 90s

ANALYSIS FINDINGS



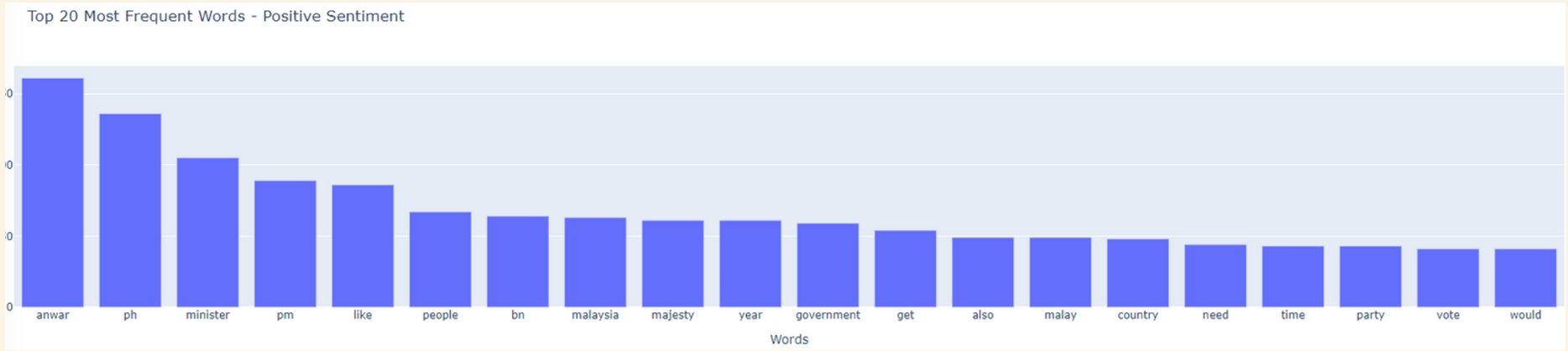
This is the overall sentiment distribution where we put together all the sentiments along with its frequency into a single pie chart. We can see there are almost 70% of neutral comments regarding DSAI followed by 18.1% of positive comments and the rest being negative comments. Here we can verify the assumption we made earlier, most of the comments he received were neutral comments, which means the public tolerates him.

ANALYSIS FINDINGS



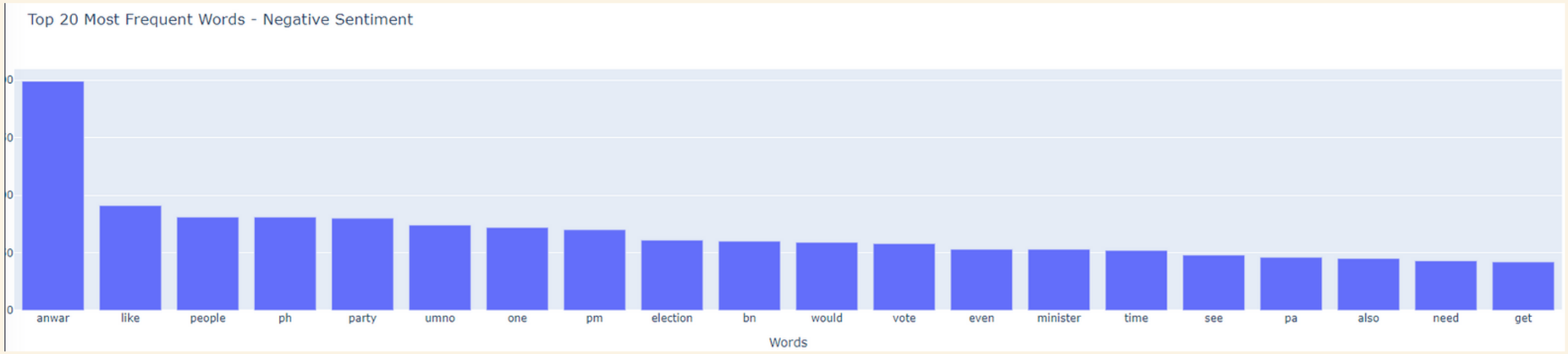
Let’s take a closer look at the 20 most frequent words used in a neutral sentiment. The word ‘like’ is most used in a neutral sentiment followed by people in second place. The words that are commonly found in a neutral sentiment doesn’t have any words that praises nor criticize him in any way.

ANALYSIS FINDINGS



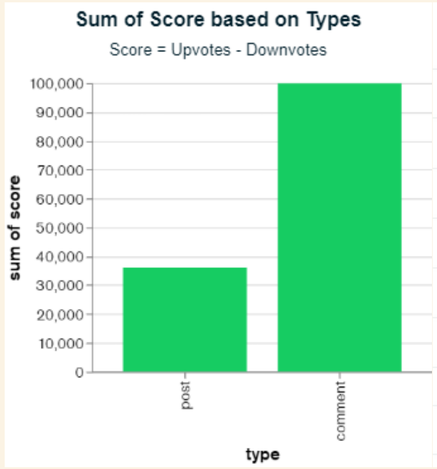
Moving over to the recurrent words found in positive sentiments. The first place goes to anwar because a positive comment would most likely be a praise and often times the praises would go to the man on top. A word we notice that appears in this chart is majesty which is not present in the neutral sentiment chart. This could imply that because DSAI received the approval of His Majesty, it helped sway the opinion of the public as they would often mention both His Majesty and praises towards DSAI.

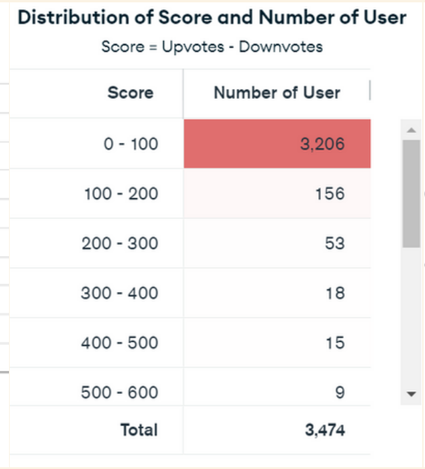
ANALYSIS FINDINGS

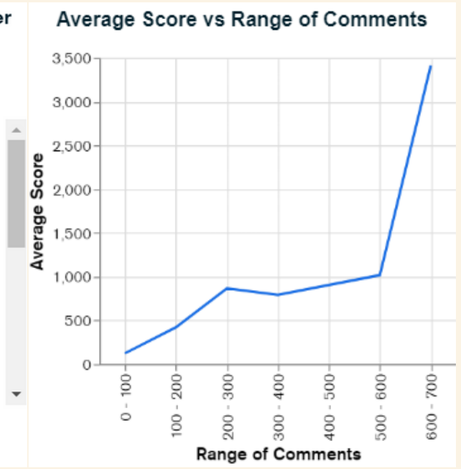


Last but not least, is the negative sentiment chart. Starting with the most used word is anwar. Similarly to the positive sentiment, any praise or criticism will often find its way to the man on top. A word that we notice that appears in the negative sentiment chart is "umno". Umno has a history of power abuse during their time in Putrajaya. Now they are back in office, people assume the same thing would happen all over again. This would then result in the comments that relate to Umno would be negative.

ADDITIONAL FINDINGS







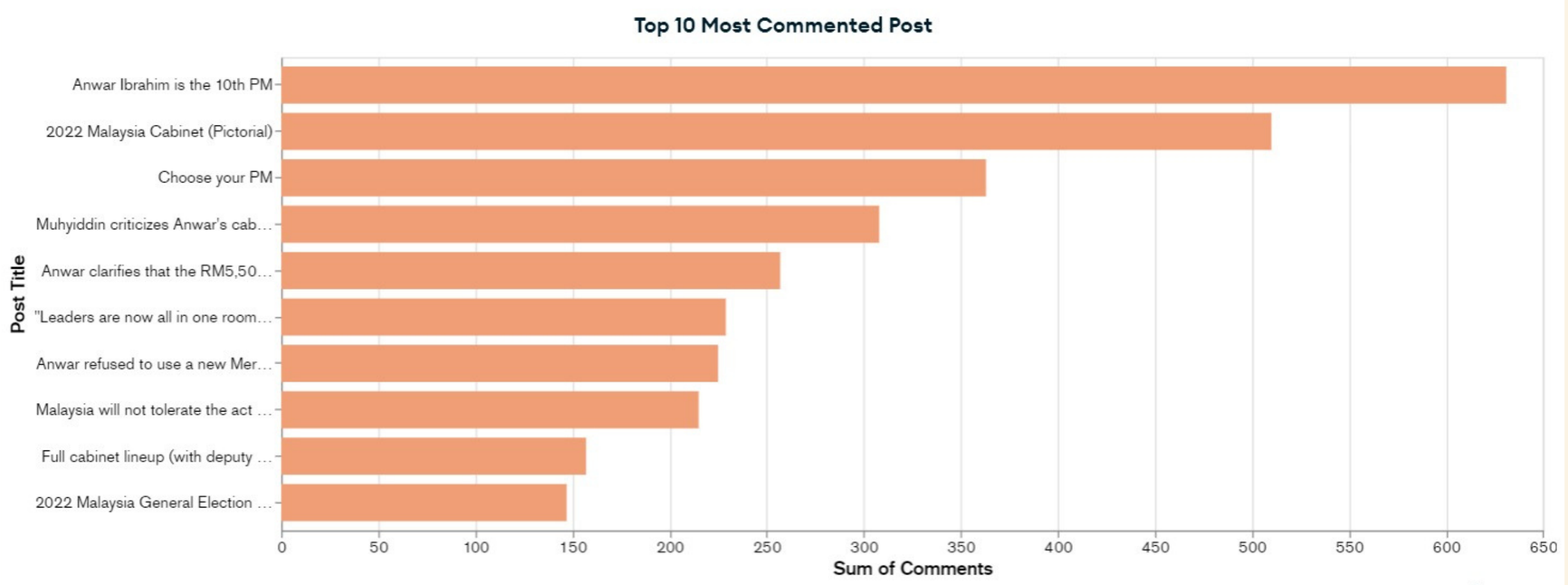
Left - we try to find the distribution of score and number of user. The score is calculated by subtracting upvotes and downvotes. We found that the majority of users: 3206, lies between 0-100 in terms of score, followed by 156 with the score of 100-200.

Mid - the relationship between average score for a post against the range of comments can be seen in the chart. A post with more average score will have a higher range of comments as seen in the gradual increase in the chart

Right - we try to find the sum of score based on types. As seen in the chart, comments have almost 60,000 in more sum of score compared to posts. This could be due to the fact there would be 39x more comments than there are posts.

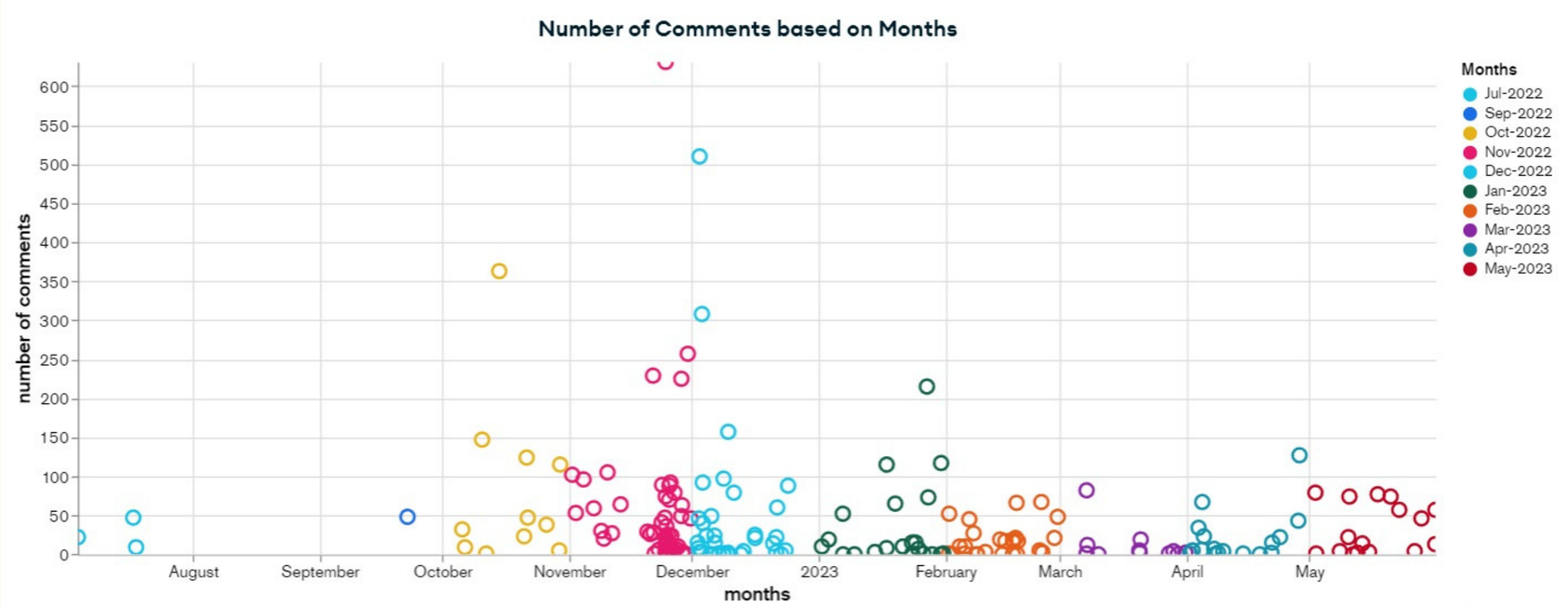
Summary - from the three charts, most of the users would have a score of 0-100 while the average score for a post would have a steady increase in the range of comments with respect to an increase of average score. By looking at the third chart, we can see that the bulk of the score accumulated originates from comments instead of posts.

ADDITIONAL FINDINGS



In this chart, we gathered 10 posts with the most comments. The post "Anwar Ibrahim is the 10th PM" takes first place with more than 600 comments in a single post. This post alone has 15 times the average comments in a single Reddit post. The post becomes quite a hot topic for people to share their opinions regarding the issue which results in a massive amount of comments.

ADDITIONAL FINDINGS



From the chart above, we can see that number of comments becomes much more significant after October. As the GE draws nearer, the number of comments towards DSAI increases over time. The number of comments skyrocketed in mid-to-late November which was during the time DSAI was sworn in as the PM. After the event, the number of comments reduces but still keeps a significant number until the end of our findings.