Project Replication Dossier

Contributor: Jonathan Standing

2023-12-12

This RMarkdown File summarizes all code (except for those related to GPT) in our project. One can knit this file and reproduce the graphs, word clouds, and other outputs in our poster.

Part 1 - Folders and Workflow

- *introduce folders and the sequence (see README.md)*

Part 2 - Code Breakdown

Step 0 - Downloading Data from Info Tracer

- *key words*
- *start date & end date*

Step 1 - Data Pre-Prosessing

After downloading the datasets from Info Tracer, the following code cleans and combines 14 datasets for the five keywords. Only the code for "Israel" is shown in this RMarkdown file for demonstration purposes, but other keywords underwent the same process. After cleaning for individual keywords, the five keywords are combined into a single dataset.

Step 1.1 Load libraries

```
library(tidyverse) # data wrangling
library(textcat) # filtering language
```

Step 1.2 Create Functions

1. Create function for filtering rows for twitter, leaving only day (no year, month, and time) of the tweet:

```
filter_twitter <- function(df) {
  twitter_df <- df[df$platform == "twitter",
                   c("text", "created_at", "total_interactions_count")]
  twitter_df$created_at <- substr(twitter_df$created_at,9,10)
  return(twitter_df)
}
```

2. Clean text (remove symbols, links, usernames, etc)

```
clean = function(text) {
  text = iconv(text, "latin1", "ASCII", sub="") #change encoding
  text = gsub("(@)\\w+", "", text) #remove numbers, alphabets after "@" (username)
  text = gsub("(http|https)://.*", "", text) # remove links
  text = gsub("[ \t]{2,}", "", text) #remove two blank spaces and tab
  text = gsub("\\n"," ",text) # remove newline
  text = gsub("\\s+"," ",text) # remove blank spaces
  text = gsub("^\\s+|\\s+$","",text) # remove blank spaces
  text = gsub("&.*;","",text) # remove special symbols and html
  text = gsub("[^a-zA-Z0-9?!. ']","",text) # remove emojis
}
```

Step 1.3 Load data

```
file_names <- c("Israel_0607.csv",
                "Israel_0708.csv",
                "Israel_0809.csv",
                "Israel_0910.csv",
                "Israel_1011.csv",
                "Israel_1112.csv",
                "Israel_1213.csv",
                "Israel_1314.csv",
                "Israel_1415.csv",
                "Israel_1516.csv",
                "Israel_1617.csv",
                "Israel_1718.csv",
                "Israel_1819.csv",
                "Israel_1920.csv",
                "Israel_2021.csv",
```

```
                "Israel_2122.csv")

df_list <- list() # create empty list to store data frame

for (file in file_names) {
  file_path <- paste0("data/israel/", file)
  df <- read.csv(file_path)
  df_list[[file]] <- df
}

filtered_list <- lapply(df_list, filter_twitter)
```

Step 1.4 Combine datasets into one big dataset

```
israel_combined <- bind_rows(filtered_list)
```

Step 1.5 Arrange by date, remove Oct 06 and Oct 22

```
israel_combined <- israel_combined |>
  arrange(created_at) |>
  filter(!(created_at %in% c("06", "22")))
```

Step 1.6 Filter English language using the "textcat" package (include "scots" english)

```
israel_combined$language <- textcat(israel_combined$text)
israel_combined_eng <- israel_combined |>
  filter(language %in% c("english", "scots"))
```

Step 1.7 Delete replicated rows

```
israel_combined_clean <- israel_combined_eng |>
  mutate(clean(israel_combined_eng$text)) |>
  rename("cleaned_text" = "clean(israel_combined_eng$text)") |>
  select(cleaned_text, created_at, total_interactions_count) |>
  distinct(cleaned_text, .keep_all = TRUE)
```

Step 1.8 Count tweets per day and plot changes

```
israel_count <- israel_combined_clean |>
  distinct(cleaned_text, .keep_all = TRUE) |>
  group_by(created_at) |>
  count()
```

Step 1.9 Save combined dataset, remove unneeded dataframes

```
write.csv(israel_combined_clean,
          file = "output/israel_clean.csv",
          row.names = FALSE)
write.csv(israel_count,
          file = "output/israel_count.csv",
          row.names = FALSE)

rm(df, df_list, filtered_list, file, file_names, file_path)
```

Step 1.10 Combining five datasets

```
d_israel <- read.csv("output/israel_clean.csv")
d_conflict <- read.csv("output/conflict_clean.csv")
d_hamas <- read.csv("output/hamas_clean.csv")
d_gaza <- read.csv("output/gaza_clean.csv")
d_palestine <- read.csv("output/palestine_clean.csv")

d_combined <- bind_rows(d_israel, d_conflict, d_hamas, d_gaza, d_palestine)
d_combined <- d_combined |>
  distinct(cleaned_text, .keep_all = TRUE) |>
  arrange(created_at)
d_combined_count <- d_combined |>
  group_by(created_at) |>
  count()

write.csv(d_combined,
          file = "output/combined_clean.csv",
          row.names = FALSE)
write.csv(d_combined_count,
          file = "output/combined_count.csv",
          row.names = FALSE)
```

Note 1 - GPT Labeling

(GPT code not included in this dossier)

Step 2 - Labelled Data Cleaning

As the GPT labelled data only contain three columns: tweet ID, tweet, and sentiment, we need to match it to the original dataset and retrieve the date and interaction count information. We also create five binary keyword column for the five keywords.

Step 2.1 Assign tweet to original data. The tweets are arranged by date, and we assign 1000 to the first tweet, and +1 for each after.

```
d_all <- read.csv("output/combined_clean.csv")
d_all_id <- d_all |>
  mutate(id_of_tweet = seq(1000, length.out = nrow(d_all), by = 1))
write.csv(d_all_id, "gpt_labelled_data/all_data.csv")
```

Step 2.2 Load GPT labelled data

```
d_b1 <- readRDS("gpt_labelled_data/Batch_1.rds")
d_b2 <- readRDS("gpt_labelled_data/Batch_2.rds")
d_b3 <- readRDS("gpt_labelled_data/Batch_3.rds")
d_b4 <- readRDS("gpt_labelled_data/Batch_4.rds")
d_id <- read.csv("gpt_labelled_data/all_data.csv")
```

Step 2.3 Add "created_at" and "interaction_count" columns

```
d1 <- inner_join(d_b1, d_id, by = "id_of_tweet") |>
  select(-cleaned_text.y, -X)
d2 <- inner_join(d_b2, d_id, by = "id_of_tweet") |>
  select(-cleaned_text.y, -X)
d3 <- inner_join(d_b3, d_id, by = "id_of_tweet") |>
  select(-cleaned_text.y, -X)
d4 <- inner_join(d_b4, d_id, by = "id_of_tweet") |>
  select(-cleaned_text.y, -X)
d <- bind_rows(d1, d2, d3, d4)
```

Step 2.4 Remove error

```
d <- d |> filter(!grepl("1\\n1", sentiment))

write.csv(d1, "gpt_output/batch1.csv")
write.csv(d2, "gpt_output/batch2.csv")
write.csv(d3, "gpt_output/batch3.csv")
write.csv(d4, "gpt_output/batch4.csv")
write.csv(d, "gpt_output/batch_all.csv")
```

Step 3.5 Add keyword columns

Entire dataset:

```
d_id$cleaned_text <- tolower(d_id$cleaned_text)

d_keywords <- d_id |>
  mutate(cleaned_text = tolower(cleaned_text)) |>
  mutate(keyword_israel = ifelse(str_detect(cleaned_text, paste0("israel")), 1, 0)) |>
```

```
  mutate(keyword_gaza = ifelse(str_detect(cleaned_text, paste0("gaza")), 1, 0)) |>
  mutate(keyword_palestine = ifelse(str_detect(cleaned_text, paste0("palestine")), 1, 0)) |>
  mutate(keyword_hamas = ifelse(str_detect(cleaned_text, paste0("hamas")), 1, 0)) |>
  mutate(keyword_conflict = ifelse(str_detect(cleaned_text, paste0("conflict")), 1, 0))

write.csv(d_keywords, "gpt_output/all_keywords.csv")
```

Batches (the ones used in the final visualization):

```
d_batch <- read.csv("gpt_output/batch_all.csv")

d_b_keywords <- d_batch |>
  rename("cleaned_text" = "cleaned_text.x") |>
  mutate(cleaned_text = tolower(cleaned_text)) |>
  mutate(keyword_israel = ifelse(str_detect(cleaned_text, paste0("israel")), 1, 0)) |>
  mutate(keyword_gaza = ifelse(str_detect(cleaned_text, paste0("gaza")), 1, 0)) |>
  mutate(keyword_palestine = ifelse(str_detect(cleaned_text, paste0("palestine")), 1, 0)) |>
  mutate(keyword_hamas = ifelse(str_detect(cleaned_text, paste0("hamas")), 1, 0)) |>
  mutate(keyword_conflict = ifelse(str_detect(cleaned_text, paste0("conflict")), 1, 0))

write.csv(d_b_keywords, "gpt_output/batch_all_keywords.csv")
```

Step 3 - Data Visualisation

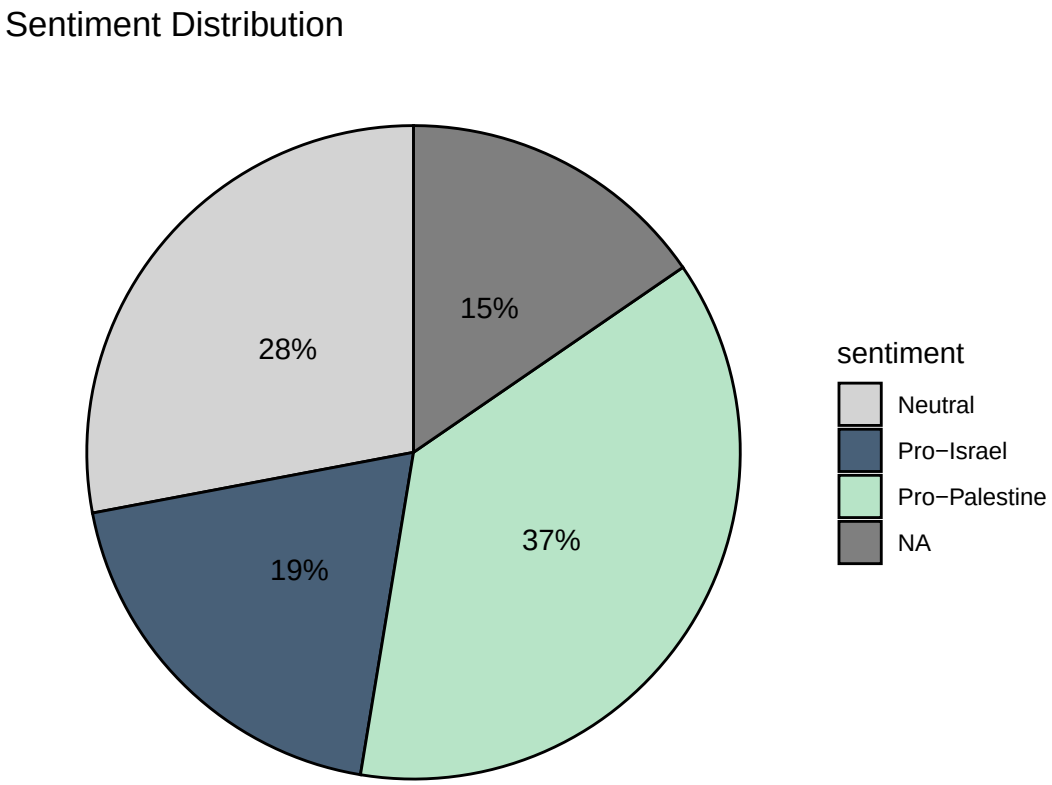
```
library(ggplot2)
library(dplyr)
library(ggpie)
library(cowplot)
```

Step 3.1 Read relevant dataset

```
d <- read.csv("gpt_output/batch_all.csv")
```

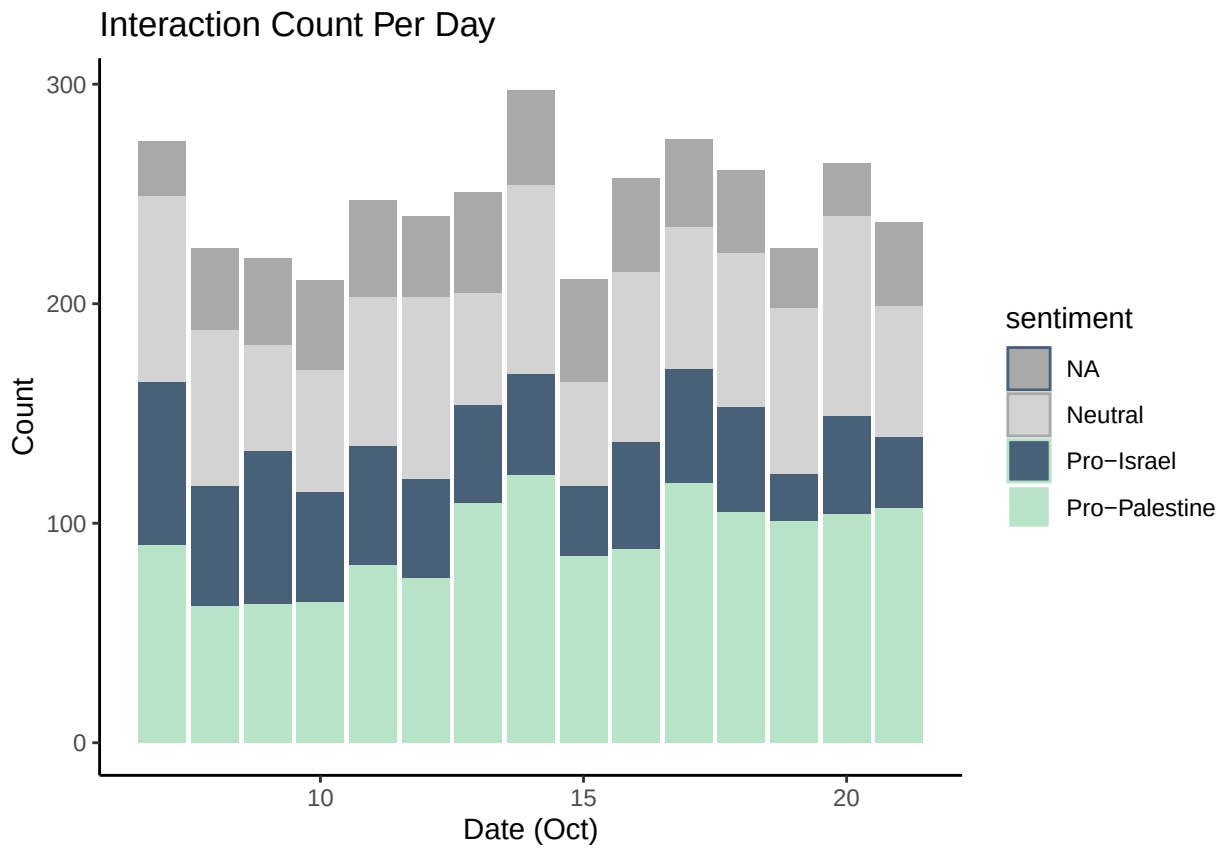
Step 3.2 Pie Chart

```
d_pie <- d |>
  mutate(sentiment = case_when(
    sentiment == 1 ~ "Pro-Israel",
    sentiment == 0 ~ "Neutral",
    sentiment == -1 ~ "Pro-Palestine",
    TRUE ~ as.character(sentiment)
  ))

ggpie(d_pie, sentiment) +
  scale_fill_manual(values=c("#D3D3D3", "#486078FF", "#B7e4C7")) +
  labs(title = "Sentiment Distribution")
```



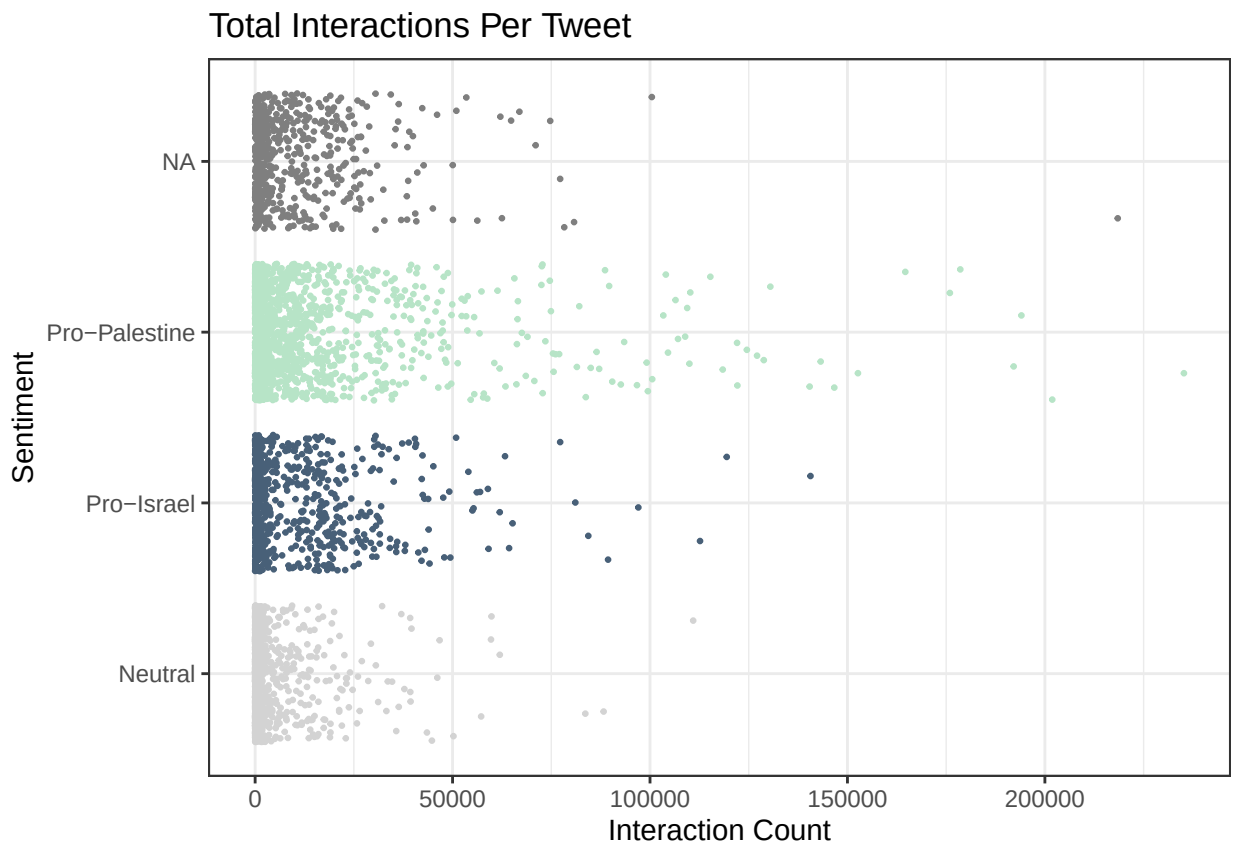
Step 3.3 Stacked bar chart

```
d_bar <- d |>
  mutate(sentiment = case_when(
    sentiment == 1 ~ "Pro-Israel",
    sentiment == 0 ~ "Neutral",
    sentiment == -1 ~ "Pro-Palestine",
    is.na(sentiment) ~ "NA",
    TRUE ~ as.character(sentiment)
  ))

d_bar |> mutate(sentiment = as.factor(sentiment)) |>
  ggplot(aes(x = created_at, fill = sentiment)) +
  geom_bar(position = "stack") +
  scale_fill_manual(values = c("Pro-Israel" = "#486078FF",
                               "Neutral" = "#D3D3D3",
                               "Pro-Palestine" = "#B7e4C7",
                               "NA" = "darkgray")) +
  theme_classic() +
  labs(title = "Interaction Count Per Day", x = "Date (Oct)", y = "Count") +
  guides(fill = guide_legend(override.aes = list(color = c("#486078FF",
                                                           "darkgray",
                                                           "#B7e4C7",
                                                           "white"))))
```



Step 3.4 Interaction count scatter plot

```
d_interaction <- d |>
  mutate(sentiment = case_when(
    sentiment == 1 ~ "Pro-Israel",
    sentiment == 0 ~ "Neutral",
    sentiment == -1 ~ "Pro-Palestine",
    TRUE ~ as.character(sentiment)
  ))

d_interaction <- d_interaction |>
  mutate(sentiment = as.factor(sentiment))

ggplot(d_interaction, aes(y = sentiment, x = total_interactions_count, color = sentiment)) +
  geom_jitter(size = 0.5) +
  scale_color_manual(values = c("Pro-Israel" = "#486078FF",
                                "Neutral" = "#D3D3D3",
                                "Pro-Palestine" = "#B7e4C7")) +
  theme_bw() +
  labs(title = "Total Interactions Per Tweet", x = "Interaction Count", y = "Sentiment") +
  theme(panel.background = element_rect(fill = "white")) +
  guides(color = FALSE)
```



Step 3.5 Keyword Time Series Graph - Grid

For the grid plot, we first create individual graphs for the entire dataset and each keyword, then plot them together.

We first create the corresponding subsets of data for each keyword.

```
d_keywords <- read.csv("gpt_output/batch_all_keywords.csv")

d_is <- d_keywords |> filter(keyword_israel == 1)
d_pa <- d_keywords |> filter(keyword_palestine == 1)
d_co <- d_keywords |> filter(keyword_conflict == 1)
d_ha <- d_keywords |> filter(keyword_hamas == 1)
d_ga <- d_keywords |> filter(keyword_gaza == 1)
```

Then, we create plot for each keyword and the entire datasets separately:

Entire dataset:

```
d_avg_sen <- d_keywords |>
  mutate(sentiment = as.numeric(sentiment)) |>
  filter(!is.na(sentiment)) |>
  group_by(created_at) |>
  summarise(avg_sen = mean(sentiment, na.rm = TRUE))
p <- d_avg_sen |>
  ggplot(aes(x = created_at, y = avg_sen)) +
  geom_point(color = "darkgray", size = 1.5) +
  geom_line(color = "#486078FF", size = 0.7) +
  geom_smooth(method = "gam", se = FALSE, linetype = "dashed", color = "darkred", size = 0.7) +
  labs(title = "All", x = "Date (Oct)", y = "Average Sentiment") +
  theme_classic()
```

Israel:

```
d_avg_sen_is <- d_is |>
  mutate(sentiment = as.numeric(sentiment)) |>
  filter(!is.na(sentiment)) |>
  group_by(created_at) |>
  summarise(avg_sen = mean(sentiment, na.rm = TRUE))
p_is <- d_avg_sen_is |>
  ggplot(aes(x = created_at, y = avg_sen)) +
  geom_point(color = "darkgray", size = 1.5) +
  geom_line(color = "#486078FF", size = 0.7) +
  geom_smooth(method = "gam", se = FALSE, linetype = "dashed", color = "darkred", size = 0.7) +
  labs(title = "Israel", x = "Date (Oct)", y = "Average Sentiment") +
  theme_classic()
```

Palestine:

```
d_avg_sen_pa <- d_pa |>
  mutate(sentiment = as.numeric(sentiment)) |>
  filter(!is.na(sentiment)) |>
  group_by(created_at) |>
  summarise(avg_sen = mean(sentiment, na.rm = TRUE))
```

```
p_pa <- d_avg_sen_pa |>
  ggplot(aes(x = created_at, y = avg_sen)) +
  geom_point(color = "darkgray", size = 1.5) +
  geom_line(color = "#486078FF", size = 0.7) +
  geom_smooth(method = "gam", se = FALSE, linetype = "dashed", color = "darkred", size = 0.7) +
  labs(title = "Palestine", x = "Date (Oct)", y = "Average Sentiment") +
  theme_classic()
```

Conflict:

```
d_avg_sen_co <- d_co |>
  mutate(sentiment = as.numeric(sentiment)) |>
  filter(!is.na(sentiment)) |>
  group_by(created_at) |>
  summarise(avg_sen = mean(sentiment, na.rm = TRUE))
p_co <- d_avg_sen_co |>
  ggplot(aes(x = created_at, y = avg_sen)) +
  geom_point(color = "darkgray", size = 1.5) +
  geom_line(color = "#486078FF", size = 0.7) +
  geom_smooth(method = "gam", se = FALSE, linetype = "dashed", color = "darkred", size = 0.7) +
  labs(title = "Conflict", x = "Date (Oct)", y = "Average Sentiment") +
  theme_classic()
```

Hamas:

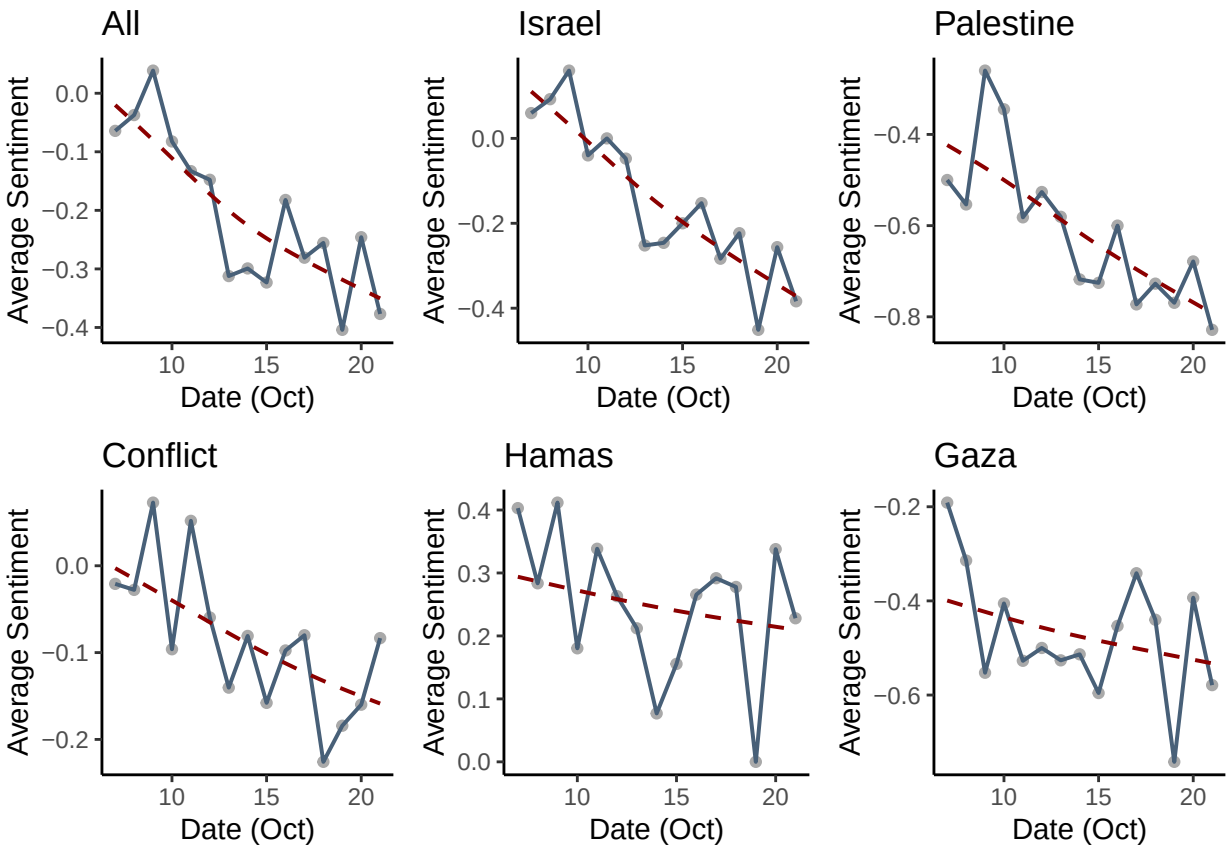
```
d_avg_sen_ha <- d_ha |>
  mutate(sentiment = as.numeric(sentiment)) |>
  filter(!is.na(sentiment)) |>
  group_by(created_at) |>
  summarise(avg_sen = mean(sentiment, na.rm = TRUE))
p_ha <- d_avg_sen_ha |>
  ggplot(aes(x = created_at, y = avg_sen)) +
  geom_point(color = "darkgray", size = 1.5) +
  geom_line(color = "#486078FF", size = 0.7) +
  geom_smooth(method = "gam", se = FALSE, linetype = "dashed", color = "darkred", size = 0.7) +
  labs(title = "Hamas", x = "Date (Oct)", y = "Average Sentiment") +
  theme_classic()
```

Gaza:

```
d_avg_sen_ga <- d_ga |>
  mutate(sentiment = as.numeric(sentiment)) |>
  filter(!is.na(sentiment)) |>
  group_by(created_at) |>
  summarise(avg_sen = mean(sentiment, na.rm = TRUE))
p_ga <- d_avg_sen_ga |>
  ggplot(aes(x = created_at, y = avg_sen)) +
  geom_point(color = "darkgray", size = 1.5) +
  geom_line(color = "#486078FF", size = 0.7) +
  geom_smooth(method = "gam", se = FALSE, linetype = "dashed", color = "darkred", size = 0.7) +
  labs(title = "Gaza", x = "Date (Oct)", y = "Average Sentiment") +
  theme_classic()
```

Then, we plot the graphs together using `plot_grid` from the `cowplot` package:

```
plot_grid(p, p_is, p_pa, p_co, p_ha, p_ga)
```

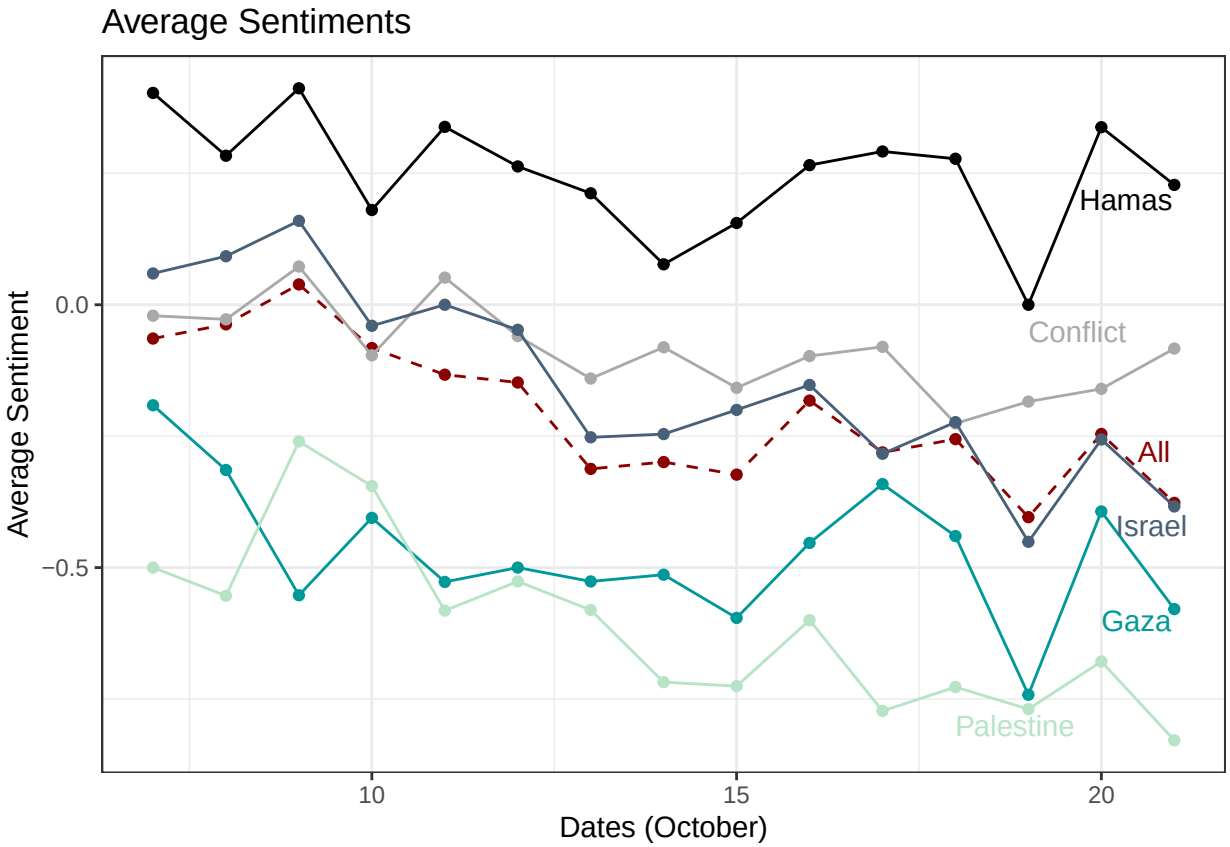



```
ggsave("gpt_output/Sentiment_Keyword_Grid.png", width = 8, height = 6)
```

Step 3.6 - Keyword Time Series Graph - Color

As the individual graphs are on the same scale, we plot a composite graph to show all of them on the same scale. We adjusted the colors to fit the theme of the poster.

```
ggplot() +
  geom_point(data = d_avg_sen, mapping = aes(x = created_at, y = avg_sen),
             color = "darkred", linetype = "dashed") +
  geom_line(data = d_avg_sen, mapping = aes(x = created_at, y = avg_sen),
            color = "darkred", linetype = "dashed") +
  geom_text(aes(x = 20.5, y = -0.28, label = "All"), hjust = 0,
            color = "darkred") +
  geom_point(data = d_avg_sen_co, mapping = aes(x = created_at, y = avg_sen),
             color = "darkgray") +
  geom_line(data = d_avg_sen_co, mapping = aes(x = created_at, y = avg_sen),
            color = "darkgray") +
  geom_text(aes(x = 19, y = -0.05, label = "Conflict"), hjust = 0,
            color = "darkgray") +
  geom_point(data = d_avg_sen_ga, mapping = aes(x = created_at, y = avg_sen),
             color = "#009999") +
  geom_line(data = d_avg_sen_ga, mapping = aes(x = created_at, y = avg_sen),
            color = "#009999") +
  geom_text(aes(x = 20, y = -0.6, label = "Gaza"), hjust = 0,
            color = "#009999") +
  geom_point(data = d_avg_sen_ha, mapping = aes(x = created_at, y = avg_sen),
             color = "black") +
  geom_line(data = d_avg_sen_ha, mapping = aes(x = created_at, y = avg_sen),
            color = "black") +
  geom_text(aes(x = 19.7, y = 0.2, label = "Hamas"), hjust = 0,
            color = "black") +
  geom_point(data = d_avg_sen_is, mapping = aes(x = created_at, y = avg_sen),
             color = "#486078FF") +
  geom_line(data = d_avg_sen_is, mapping = aes(x = created_at, y = avg_sen),
            color = "#486078FF") +
  geom_text(aes(x = 20.2, y = -0.42, label = "Israel"), hjust = 0,
            color = "#486078FF") +
  geom_point(data = d_avg_sen_pa, mapping = aes(x = created_at, y = avg_sen),
             color = "#B7e4C7") +
  geom_line(data = d_avg_sen_pa, mapping = aes(x = created_at, y = avg_sen),
            color = "#B7e4C7") +
  geom_text(aes(x = 18, y = -0.8, label = "Palestine"), hjust = 0,
            color = "#B7e4C7") +
  labs(title = "Average Sentiments", x = "Dates (October)", y = "Average Sentiment") +
  theme_bw()
```




```
ggsave("gpt_output/Sentiment_Keyword_Colors.png", width = 8, height = 6)
```

Step 4 - Text mining and word cloud generation

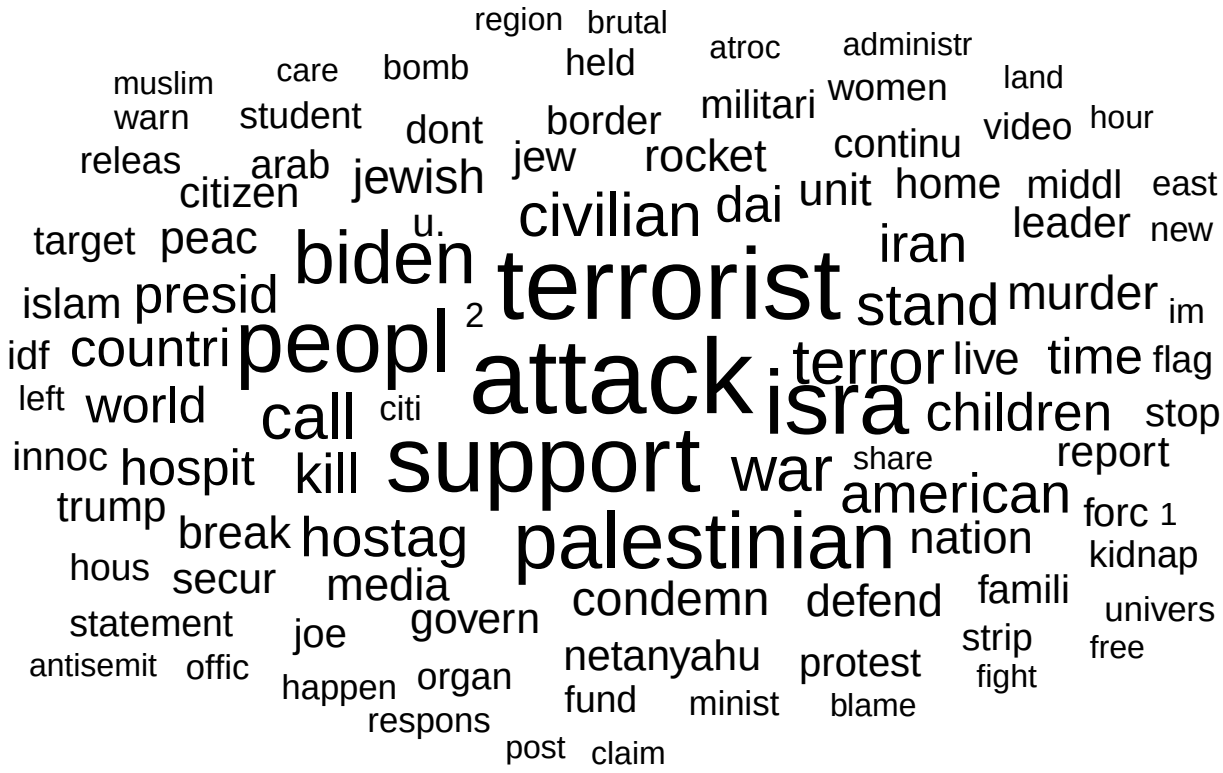
Finally, to figure out the most frequently used words for each sentiment, we generated word clouds for each of them.

Step 4.1 Load dataset and packages

```
library(tidytext)
library(SnowballC)
library(ggwordcloud)

data("stop_words")
```

Step 4.2 Word cloud for Pro-Israel (1)

```
d_is <- d |> filter(sentiment == 1)

d_token_is = d_is |>
  select(id_of_tweet, cleaned_text.x, sentiment) |>
  unnest_tokens(word, cleaned_text.x) |>
  anti_join(stop_words, by = "word") |>
  mutate(stem = wordStem(word))

word_freq_is = d_token_is |>
  count(stem, sort = TRUE) |>
  rename("word" = "stem")

word_freq_top_is = word_freq_is |>
  arrange(desc(n)) |>
  slice(1:200) |>
  filter(!(word %in% c("israel", "hama", "conflict", "gaza", "palestin")))

word_freq_top_is |>
  slice(1:100) |>
  ggplot(aes(label = word, size = n)) +
  scale_size_area(max_size = 14) +
  geom_text_wordcloud() +
  theme_minimal()
```



Step 4.3 Word cloud for Pro-Palestine (-1)

```
d_pa <- d |> filter(sentiment == -1)

d_token_pa = d_pa |>
  select(id_of_tweet, cleaned_text.x, sentiment) |>
  unnest_tokens(word, cleaned_text.x) |>
  anti_join(stop_words, by = "word") |>
  mutate(stem = wordStem(word))

word_freq_pa = d_token_pa |>
  count(stem, sort = TRUE) |>
  rename("word" = "stem")

word_freq_top_pa = word_freq_pa |>
  arrange(desc(n)) |>
  slice(1:200) |>
  filter(!(word %in% c("israel", "hama", "conflict", "gaza", "palestin")))

word_freq_top_pa |>
  slice(1:100) |>
  ggplot(aes(label = word, size = n)) +
  scale_size_area(max_size = 14) +
  geom_text_wordcloud() +
  theme_minimal()
```

Step 4.4 Word cloud for Neutral (0)

```
d_nu <- d |> filter(sentiment == 0)

d_token_nu = d_nu |>
  select(id_of_tweet, cleaned_text.x, sentiment) |>
  unnest_tokens(word, cleaned_text.x) |>
  anti_join(stop_words, by = "word") |>
  mutate(stem = wordStem(word))

word_freq_nu = d_token_nu |>
  count(stem, sort = TRUE) |>
  rename("word" = "stem")

word_freq_top_nu = word_freq_nu |>
  arrange(desc(n)) |>
  slice(1:200) |>
  filter(!(word %in% c("israel", "hama", "conflict", "gaza", "palestin")))

word_freq_top_nu |>
  slice(1:100) |>
  ggplot(aes(label = word, size = n)) +
  scale_size_area(max_size = 14) +
  geom_text_wordcloud() +
  theme_minimal()
```

Step 4.5 Word cloud for content filtered (NA)

```
d_fi <- d[is.na(d$sentiment), ]

d_token_fi = d_fi |>
  select(id_of_tweet, cleaned_text.x, sentiment) |>
  unnest_tokens(word, cleaned_text.x) |>
  anti_join(stop_words, by = "word") |>
  mutate(stem = wordStem(word))

word_freq_fi = d_token_fi |>
  count(stem, sort = TRUE) |>
  rename("word" = "stem")

word_freq_top_fi = word_freq_fi |>
  arrange(desc(n)) |>
  slice(1:200) |>
  filter(!(word %in% c("israel", "hama", "conflict", "gaza", "palestin")))

word_freq_top_fi |>
  slice(1:100) |>
  ggplot(aes(label = word, size = n)) +
  scale_size_area(max_size = 14) +
  geom_text_wordcloud() +
  theme_minimal()
```