1. Importing Libraries

Code:

library(readxl)

library(tidyverse)

library(tidytext)

library(ggplot2)

* **library(readxl):** Loads the readxl package to read Excel files into R. It provides the read\_excel() function to read .xlsx files.
* **library(tidyverse):** Loads the tidyverse, which is a collection of R packages designed for data manipulation, analysis, and visualization (such as dplyr, ggplot2, and others).
* **library(tidytext):** Loads the tidytext package, which is specifically designed for text mining in a tidy format. It helps in text tokenization, removing stopwords, and performing sentiment analysis.
* **library(ggplot2):** Loads the ggplot2 package, a part of tidyverse used for data visualization. It allows you to create complex plots easily using the grammar of graphics.

1. Reading Data from an Excel File

Code:

data\_frame <- read\_excel("D:/AIUB/Thesis/GIT/Dataset/Processed Data.xlsx", sheet = "Dhaka Regency")

View(data\_frame)

* **data\_frame <- read\_excel(...):** This reads the Excel file located at "D:/AIUB/Thesis/GIT/Dataset/Processed Data.xlsx" and selects the sheet named "Dhaka Regency". The data is stored in the variable data\_frame.
* **View(data\_frame):** This opens a spreadsheet-like viewer window in RStudio to show the contents of the data\_frame. This is useful for checking what the dataset looks like before further processing.

1. Tokenization (Breaking Text into Words)

Code:

tidy\_data <- data\_frame %>%

unnest\_tokens(word, Text)

View(tidy\_data)

* **unnest\_tokens(word, Text):** This function from the tidytext package tokenizes (breaks) the text into individual words. Here:
* **word:** The name of the new column that will store the individual words.
* **Text:** The column in the data\_frame that contains the text you want to tokenize.
* The result is a new data frame tidy\_data, where each row corresponds to a single word from the original text data. The unnest\_tokens() function helps convert text data into a tidy format, with each word as a separate row.
* **View(tidy\_data):** Again, this opens a spreadsheet-like viewer window in RStudio to check the tokenized data.

1. Getting a Sentiment Lexicon

Code:

bing\_lexicon <- get\_sentiments("bing")

* **get\_sentiments("bing"):** This function retrieves the Bing Liu sentiment lexicon. The Bing lexicon classifies words into two categories: positive or negative sentiment. It is used for sentiment analysis and contains words commonly associated with positive or negative emotions.

1. Performing Sentiment Analysis

Code:

sentiment\_scores <- tidy\_data %>%

inner\_join(bing\_lexicon, by = "word")

* **inner\_join(bing\_lexicon, by = "word"):** This performs an inner join between the tokenized words (tidy\_data) and the Bing sentiment lexicon (bing\_lexicon) based on the column word. An inner join keeps only the words that are present in both the tokenized dataset and the Bing lexicon.
* This step maps each word in the tidy\_data to its corresponding sentiment (positive or negative) from the Bing lexicon and stores the result in sentiment\_scores.

1. Summarizing the Sentiment Data

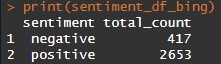
Code:

sentiment\_summary\_bing <- sentiment\_scores %>%

group\_by(sentiment) %>%

summarize(total\_count = n())

* **group\_by(sentiment):** This groups the data by the sentiment column (which contains either "positive" or "negative").
* **summarize(total\_count = n()):** This creates a summary that calculates the total number of occurrences for each sentiment by counting the number of words associated with either "positive" or "negative" sentiment. The result is stored in sentiment\_summary\_bing, which contains the total number of positive and negative words.



1. Converting the Summary to a Data Frame

Code:

sentiment\_df\_bing <- as.data.frame(sentiment\_summary\_bing)

print(sentiment\_df\_bing)

* **as.data.frame(sentiment\_summary\_bing):** Converts the summary of sentiments into a standard R data frame format and stores it in sentiment\_df\_bing.
* **print(sentiment\_df\_bing):** This prints the sentiment\_df\_bing to the console to see the summarized results, which will display the count of positive and negative words.

1. Visualizing the Sentiment Analysis Results

Code:

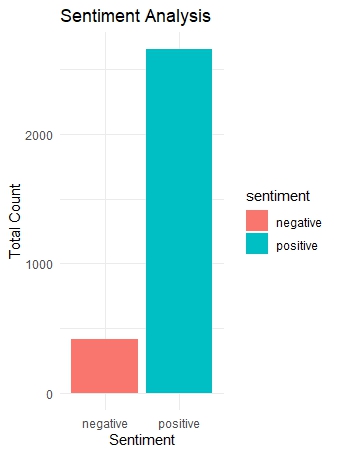
ggplot(sentiment\_summary\_bing, aes(x = sentiment, y = total\_count, fill = sentiment)) +

geom\_bar(stat = "identity") +

labs(title = "Sentiment Analysis", x = "Sentiment", y = "Total Count") +

theme\_minimal()

* **ggplot(sentiment\_summary\_bing, aes(x = sentiment, y = total\_count, fill = sentiment)):** This initializes a ggplot object to create a bar chart. The arguments in aes() specify:
* **x = sentiment:** The sentiment (positive or negative) will be on the x-axis.
* **y = total\_count:** The total count of positive and negative words will be on the y-axis.
* **fill = sentiment:** Each bar in the bar chart will be filled with a color representing the sentiment (positive or negative).
* **geom\_bar(stat = "identity"):** Adds the bars to the bar chart, where the height of each bar is determined by the actual values (not counts, which is what stat = "identity" does).
* **labs(...):** Adds labels to the plot:
* **title = "Sentiment Analysis":** The title of the plot.
* **x = "Sentiment":** The x-axis label.
* **y = "Total Count":** The y-axis label.
* **theme\_minimal():** Applies a clean, minimal theme to the plot.



* The result is a bar plot that visually displays the number of positive and negative words in the dataset, giving insight into the overall sentiment of the text.

Summary:

1. **Data Loading:** Reads the Excel file containing hotel reviews.
2. **Tokenization:** Splits the text into individual words.
3. **Sentiment Analysis:** Uses the Bing sentiment lexicon to classify each word as positive or negative.
4. **Summary:** Groups the sentiment results to count the total number of positive and negative words.
5. **Visualization:** Creates a bar plot to visualize the sentiment analysis results.