Assignment2

Sagnik Chakravarty

Loading the data

```
library(tm)
library(ggplot2)
library(word2vec)
library(uwot)
library(glmnet)
library(text2vec)
cos.sim <- function(a,b)
{

    return( sum(a*b)/sqrt(sum(a^2)*sum(b^2)) )
}
library(readr)
library(readxl)
library(dplyr)
data <- read_csv("thread_cleaned.csv")
data_sample <-  read_excel("~/Desktop/UMD_College_Work/Semester 2/SURV622/Assignment/SURV6
                      sheet = "thread_cleaned_sample.csv",
                      col_types = c("skip", "numeric", "numeric",
                                    "text", "text", "text",
                                    "numeric", "text", "text",
                                    "text", "text", "text"))
head(data)
```

```
# A tibble: 6 x 9
  date_utc    timestamp title       text  subreddit comments url   cleaned_title
  <date>          <dbl> <chr>       <chr> <chr>        <dbl> <chr> <chr>
1 2025-03-02 1740954795 "Zelensky ~ <NA>  europe           0 http~ zelensky tel~
2 2025-03-01 1740845366 "Tens of t~ <NA>  europe           6 http~ tens thousan~
```

```
3 2025-02-27 1740677287 "Trump rif~ <NA>  europe           1 http~ trump rift d~
4 2025-02-26 1740587428 "The U.S. ~ <NA>  europe           0 http~ u.s ukraine ~
5 2025-02-26 1740575235 "What can ~ <NA>  europe           2 http~ starmer trum~
6 2025-02-26 1740574460 "European ~ <NA>  europe           0 http~ european pra~
# i 1 more variable: cleaned_text <chr>
```

```
head(data_sample)
```

```
# A tibble: 6 x 11
  date_utc  timestamp title         text  subreddit comments url   cleaned_title
     <dbl>      <dbl> <chr>         <chr> <chr>        <dbl> <chr> <chr>
1    45717 1740845366 "Tens of tho~ NA    europe           6 http~ tens thousan~
2    45714 1740551729 "This is Pol~ NA    europe           0 http~ poland film ~
3    45713 1740498870 "Rally in Sr~ NA    europe           3 http~ rally srpska~
4    45713 1740476755 "The grants ~ NA    europe           6 http~ grants loans~
5    45717 1740853687 "\u001cYou c~ NA    europe          34 http~ insult count~
6    45714 1740579121 "Freedomhous~ NA    europe          75 http~ freedomhouse~
# i 3 more variables: cleaned_text <chr>, Sentiment <chr>,
#   `Sentiment Full` <chr>
```

```
dim(data)
```

```
[1] 4423    9
```

```
dim(data_sample)
```

```
[1] 400  11
```

Changing Text and Title to unigram

```
library(word2vec)
train_word2vec_model <- function(title_column,
                                 text_column,
                                 type = "cbow",
                                 dim = 100,
                                 window = 5,
```

```r
                                  iter = 10,
                                  min_count = 1,
                                  return_model = TRUE) {

  # Step 0: Replace NA with empty strings
  title_column[is.na(title_column)] <- ""
  text_column[is.na(text_column)] <- ""

  # Step 1: Combine title and text
  combined_text <- paste(title_column, text_column, sep = " ")

  # Step 2: Clean the text
  clean_text <- tolower(combined_text)
  clean_text <- gsub("[^a-z\\s]", "", clean_text)
  clean_text <- gsub("\\s+", " ", clean_text)

  # Step 3: Tokenize
  tokens <- word_tokenizer(clean_text)

  # Step 4: Prepare training data
  cleaned_sentences <- sapply(tokens, paste, collapse = " ")

  # Step 5: Train Word2Vec model
  model <- word2vec(
    x = cleaned_sentences,
    type = type,
    dim = dim,
    window = window,
    iter = iter,
    min_count = min_count
  )

  # Step 6: Return model or embedding matrix
  if (return_model) {
    return(model)
  } else {
    return(as.matrix(model))
  }
}
```

```r
model_word_2vec <- train_word2vec_model(data_sample$title,
                                        data_sample$text)
```

```r
library(text2vec)
library(word2vec)
library(e1071)         # for SVM
```

```
Warning: package 'e1071' was built under R version 4.3.3
```

```r
library(rpart)         # for decision tree
```

```
Warning: package 'rpart' was built under R version 4.3.3
```

```r
library(caret)         # for evaluation
```

```
Warning: package 'caret' was built under R version 4.3.3

Loading required package: lattice

Warning: package 'lattice' was built under R version 4.3.3
```

```r
library(dplyr)

# Step 1: Reuse your cleaned text generator
generate_cleaned_sentences <- function(title_column, text_column) {
  title_column[is.na(title_column)] <- ""
  text_column[is.na(text_column)] <- ""
  combined <- paste(title_column, text_column, sep = " ")
  cleaned <- tolower(combined)
  cleaned <- gsub("[^a-z\\s]", "", cleaned)
  cleaned <- gsub("\\s+", " ", cleaned)
  tokens <- text2vec::word_tokenizer(cleaned)
  sentences <- sapply(tokens, paste, collapse = " ")
  return(sentences)
}
```

```r
# Step 2: Get cleaned sentences
sentences <- generate_cleaned_sentences(data_sample$title,
                                        data_sample$text)

# Step 3: Get Word2Vec embeddings for each document
get_sentence_embedding <- function(model, sentence) {
  words <- unlist(strsplit(sentence, " "))
  embeddings <- predict(model, newdata = words, type = "embedding")
  colMeans(embeddings, na.rm = TRUE)
}

# Apply to all sentences
X <- t(sapply(sentences, function(s) get_sentence_embedding(model_word_2vec, s)))

# Step 4: Prepare target variable
y <- as.factor(data_sample$`Sentiment Full`)

# Optional: Remove rows with NA in embedding (caused by unseen words)
valid_rows <- complete.cases(X)
X <- X[valid_rows, ]
y <- y[valid_rows]

# Step 5: Train/test split
set.seed(123)
train_idx <- sample(seq_len(nrow(X)), size = 0.8 * nrow(X))
X_train <- X[train_idx, ]
X_test  <- X[-train_idx, ]
y_train <- y[train_idx]
y_test  <- y[-train_idx]

# Step 6a: Train SVM
svm_model <- svm(X_train, y_train, kernel = "linear")
svm_pred  <- predict(svm_model, X_test)

# Step 6b: Train Decision Tree
tree_model <- rpart(y_train ~ ., data = data.frame(X_train, y_train))
tree_pred  <- predict(tree_model, newdata = data.frame(X_test), type = "class")

# Step 7: Evaluate both models
svm_cm <- confusionMatrix(svm_pred, y_test)
tree_cm <- confusionMatrix(tree_pred, y_test)
```

```r
cat("  SVM Performance:\n")
```

```
  SVM Performance:
```

```r
print(svm_cm)
```

```
Confusion Matrix and Statistics

            Reference
Prediction   Favor Irrelevant Neutral Oppose
  Favor          0          5       1      3
  Irrelevant     2         12       3     11
  Neutral        0          9       1      2
  Oppose         0          9       2      2

Overall Statistics

               Accuracy : 0.2419
                 95% CI : (0.1422, 0.3674)
    No Information Rate : 0.5645
    P-Value [Acc > NIR] : 1.0000

                  Kappa : -0.1527

 Mcnemar's Test P-Value : 0.2046

Statistics by Class:

                     Class: Favor Class: Irrelevant Class: Neutral
Sensitivity               0.00000            0.3429        0.14286
Specificity               0.85000            0.4074        0.80000
Pos Pred Value            0.00000            0.4286        0.08333
Neg Pred Value            0.96226            0.3235        0.88000
Prevalence                0.03226            0.5645        0.11290
Detection Rate            0.00000            0.1935        0.01613
Detection Prevalence      0.14516            0.4516        0.19355
Balanced Accuracy         0.42500            0.3751        0.47143
                     Class: Oppose
Sensitivity                0.11111
Specificity                0.75000
```

```
Pos Pred Value            0.15385
Neg Pred Value            0.67347
Prevalence                0.29032
Detection Rate            0.03226
Detection Prevalence      0.20968
Balanced Accuracy         0.43056
```

```
cat("\n  Decision Tree Performance:\n")
```

```

  Decision Tree Performance:
```

```
print(tree_cm)
```

```
Confusion Matrix and Statistics

            Reference
Prediction   Favor Irrelevant Neutral Oppose
  Favor          0          5       1      3
  Irrelevant     2         19       5     12
  Neutral        0          2       0      0
  Oppose         0          9       1      3

Overall Statistics

               Accuracy : 0.3548
                 95% CI : (0.2374, 0.4866)
    No Information Rate : 0.5645
    P-Value [Acc > NIR] : 0.9997

                  Kappa : -0.1032

 Mcnemar's Test P-Value : 0.2381

Statistics by Class:

                     Class: Favor Class: Irrelevant Class: Neutral
Sensitivity               0.00000            0.5429        0.00000
Specificity               0.85000            0.2963        0.96364
Pos Pred Value            0.00000            0.5000        0.00000
```

```
Neg Pred Value              0.96226              0.3333         0.88333
Prevalence                  0.03226              0.5645         0.11290
Detection Rate              0.00000              0.3065         0.00000
Detection Prevalence        0.14516              0.6129         0.03226
Balanced Accuracy           0.42500              0.4196         0.48182
                       Class: Oppose
Sensitivity                  0.16667
Specificity                  0.77273
Pos Pred Value               0.23077
Neg Pred Value               0.69388
Prevalence                   0.29032
Detection Rate               0.04839
Detection Prevalence         0.20968
Balanced Accuracy            0.46970
```

```
model_word_2vec_full <- train_word2vec_model(data$title,
                                             data$text)
# Combine title and text (handle NA properly)
sentences <- mapply(function(t, txt) {
  paste0(ifelse(is.na(t), "", t), " ", ifelse(is.na(txt), "", txt))
}, data$title, data$text)

# Ensure all sentences are character
sentences <- as.character(sentences)

# Clean text (optional: include your clean_text() function here if used earlier)

# Generate embeddings safely, skipping bad ones
get_valid_embedding <- function(sentence) {
  if (is.na(sentence) || !is.character(sentence) || sentence == "") return(NULL)
  tryCatch({
    emb <- get_sentence_embedding(model_word_2vec_full, sentence)
    if (is.null(emb) || !is.numeric(emb)) return(NULL)
    return(emb)
  }, error = function(e) {
    return(NULL)
  })
}

# Apply safely
embeddings <- lapply(sentences, get_valid_embedding)
```

```r
# Filter out NULLs and convert to matrix
valid_embeddings <- Filter(Negate(is.null), embeddings)
X <- do.call(rbind, valid_embeddings)

colnames(X) <- colnames(X_train)
```

```r
data$Sentiment_ML_predicted <- predict(tree_model, newdata = data.frame(X), type = 'class'
```

```r
write_csv(data, file = 'ML_Predicted_Sentiment.csv')
```