Modeling and Prediction Algorithm

Ashwin Sai Murali Neelakandan

2025-01-29

Setting up the Environment

Due to large size of the data, I have downloaded the data from the above link, unzipped it and stored it locally in the folder Coursera-SwiftKey to save time downloading and unzipping the data.

```
# Loading the necessary packages

library(knitr)
library(quanteda)
library(tm)
library(dplyr)
library(stringi)
library(stringr)
library(readr)
library(tokenizers)
library(tau)
```

Document Setup

Reading the Blogs, News, and Twitter Files

Sampling the Corpus Data

Due to large sizes of the data files, to improve algorithm efficiency and reduce computation time, I will be taking a random sampling of from the three files

```
# Setting the seed
set.seed(14928)

# Sampling the data from each file

sam_blog <- sample(blogfile, size = 1000, replace = FALSE)
sam_twitter <- sample(twitterfile, size = 1000, replace = FALSE)
sam_news <- sample(newsfile, size = 1000, replace = FALSE)

# Combining the samples

final_sample <- c(sam_blog,sam_twitter,sam_blog)
```

Creating a corpora file

A corpora file is created with the sample data from the three files. The corpora file is saved to the disk so that it can be used later for the shiny app.

```
EN_corpus <- VCorpus(VectorSource(final_sample))
t_S <- content_transformer(function(x, pattern)
  gsub(pattern, " ", x)
  )
```

Cleaning the Corpus Data

The sample data is cleaned with the following transformations:
- Remove Punctuation
- Remove Numbers
- Remove White spaces
- Remove stopwords
- Converting the entire text to lower case

I have downloaded the Profanities in English file from Kaggle to my disk. Loading it to removing profanity from corpus file

```
p <- read.csv("~/Coursera/Data Science Capstone/Project Data/Word_Prediction/profanity_en.csv")

p_words <- p$text
path <- paste(getwd(), "~/Coursera/Data Science Capstone/Project Data/Word_Prediction/profanity.txt", se
writeLines(p$text, "~/Coursera/Data Science Capstone/Project Data/Word_Prediction/profanity.txt")

profanity <- readLines("~/Coursera/Data Science Capstone/Project Data/Word_Prediction/profanity.txt", wa
profanity <- iconv(profanity, "latin1", "ASCII", sub = "")
```

```
# Defining a function to clean the corpus data and to build a ngram model

create_ngram_model <- function(corpus, n = 3) {
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, stripWhitespace)

  ngram_model <- list()  # Initialize empty model

  for (i in seq_along(corpus)) {
    text <- content(corpus[[i]])
    words <- unlist(strsplit(text, " "))
    words <- words[words != ""]  # Remove empty strings

    if (length(words) < n) next

    for (j in 1:(length(words) - n)) {
  prefix <- paste(words[j:(j + n - 1)], collapse = " ")
  next_word <- words[j + n]

  print(paste("Adding:", prefix, "->", next_word))  # Debug print
```

```
  if (!prefix %in% names(ngram_model)) {
    ngram_model[[prefix]] <- list()
  }

  if (next_word %in% names(ngram_model[[prefix]])) {
    ngram_model[[prefix]][[next_word]] <- ngram_model[[prefix]][[next_word]] + 1
  } else {
    ngram_model[[prefix]][[next_word]] <- 1
  }
}



  # Convert counts to probabilities
  for (prefix in names(ngram_model)) {
  total_count <- sum(unlist(ngram_model[[prefix]]))  # Convert list to numeric vector

  print(paste("Processing prefix:", prefix, "| Total count:", total_count))  # Debug print

  if (total_count > 0) {  # Avoid division by zero
    ngram_model[[prefix]] <- lapply(ngram_model[[prefix]], function(x) x / total_count)
  } else {
    print(paste("Warning: Empty prefix:", prefix))  # Debug warning
  }
  }
  }
}
```

Creating the N gram model and Saving it to file

```
corpus <- VCorpus(VectorSource(final_sample))  # Example corpus
ngram_model <- create_ngram_model(corpus, n = 3)
save(ngram_model, file = "ngram_model.RData")
```