Milestone Report

Dacia Martinez Diaz

2024-12-06

Introduction

The goal of this task is to explore the dataset, clean the data, and build a preliminary predictive model. The data is sourced from three text files: blogs, news articles, and Twitter posts. In this report, I will provide an overview of the steps taken to process the data, highlight key findings, and outline the next steps toward building a prediction algorithm and a Shiny app.

Step 0: Importing Data

The first step was importing the datasets. I used the `fread()` function from the `data.table` package to read in the three files:

- **Blogs**
- **News**
- **Twitter**

```r
library(data.table)

blogs <- fread("~/Coursera_Specialization/Coursera-SwiftKey/en_US/en_US.blogs.txt", sep = "\n", header =

news <- fread("~/Coursera_Specialization/Coursera-SwiftKey/en_US/en_US.news.txt", sep = "\n", header = 

twitter <- fread("~/Coursera_Specialization/Coursera-SwiftKey/en_US/en_US.twitter.txt", sep = "\n", hea
```

Each dataset was read as a table, with a single column of text data (one line per observation). The encoding was set to "UTF-8" to ensure correct character representation. Here's an example of the first few rows of the twitter dataset:

V1
How are you? Btw thanks for the RT. You gonna be in DC anytime soon? Love to see you. Been way, way too long.
When you meet someone special… you'll know. Your heart will beat more rapidly and you'll smile for no reason.
they've decided its more fun if I don't.
So Tired D; Played Lazer Tag & Ran A LOT D; Ughh Going To Sleep Like In 5 Minutes ;)
Words from a complete stranger! Made my birthday even better :)

Step 1: Exploratory Data Analysis (EDA)

1. Basic Statistics

The first step in the exploratory data analysis (EDA) was to compute basic statistics, such as word counts and sentence counts, for each dataset. This helped us understand the scale of the data.

```
## # A tibble: 3 x 3
##   source  num_sentences num_words
##   <chr>           <int>     <int>
## 1 blogs          615491  25533567
## 2 news           987096  33585252
## 3 twitter       2360148  30373583
```

2. Data Cleaning

A basic preprocessing was performed to prepare the text for analysis. This included the following steps:

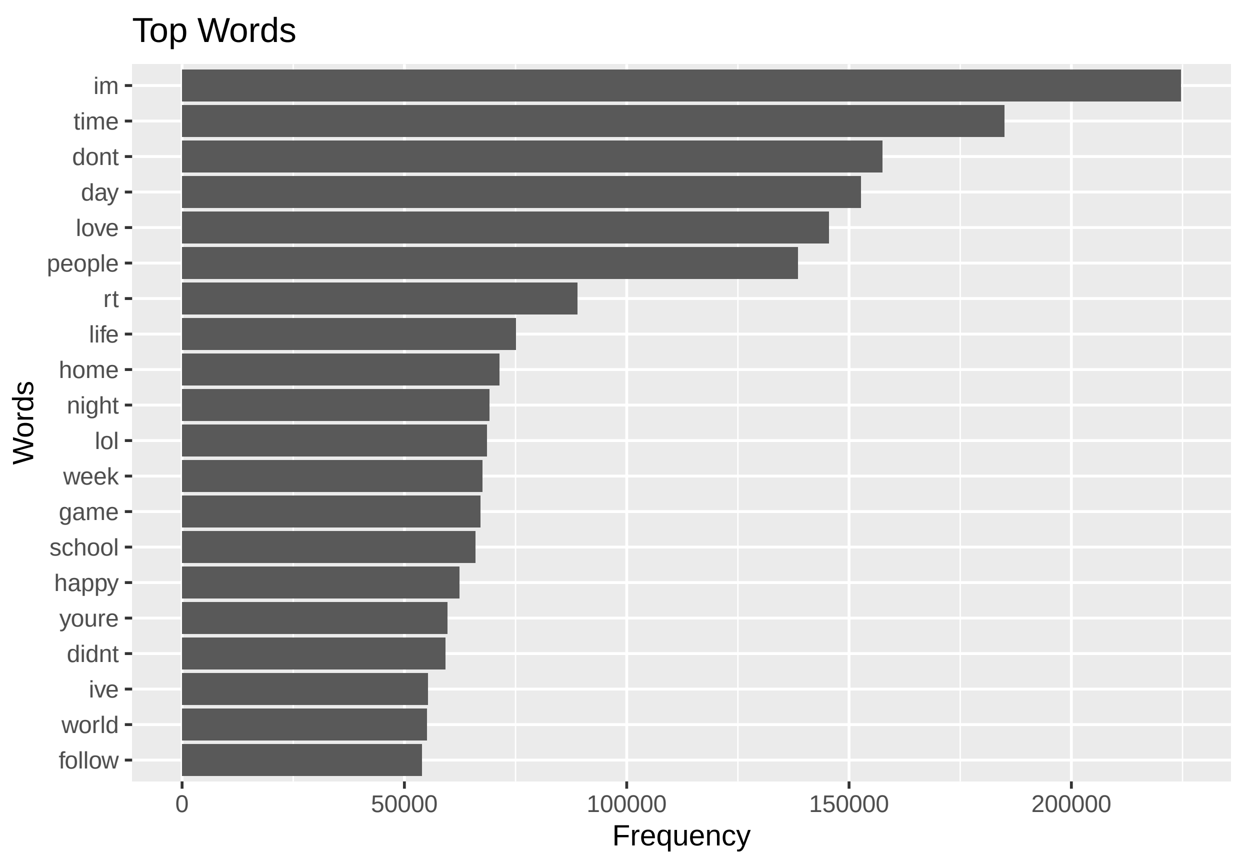
- Lowercase all text for consistency.
- Remove punctuation, numbers, and special characters to focus on words.
- Remove empty lines and extra white space.

```
# Basic preprocessing
all_data$text <- all_data %>%
  filter(!is.na(text)) %>%   # Remove rows where 'text' is NA
  pull(text) %>%             # Extract the 'text' column as a vector
  str_to_lower() %>%         # Convert to lowercase
  str_replace_all("[^[:alpha:][:space:]]", "") %>%  # Remove non-alphabetic characters
  str_squish()               # Remove extra whitespace

# Remove rows with NA
all_data <- all_data %>% filter(!is.na(text) )
```

3. Word Frequency Analysis

The text was tokenized into individual words to do some work frecuency analysis. The next figure displays the most frequent words excluding common stopwords such as “the” and “and”. This gives us insight into the most frequently used terms in the blogs, news, and Twitter posts and the key themes in the dataset.



Step 2: Build N-Gram Models

N-grams of varying lengths (unigrams, bigrams, trigrams, and fourgrams) were constructed and probabilities were calculated for each N-gram to enable prediction.

```
## [1] "Unigrams"
```

```
##      word       n       prob
##    <char>   <int>      <num>
## 1:    the 4129064 0.04732919
## 2:     to 2395818 0.02746195
## 3:      a 2077308 0.02381104
## 4:    and 2040353 0.02338745
## 5:     of 1710879 0.01961087
## 6:     in 1440932 0.01651661
```

```
## [1] "Bigrams"
```

```
##        w1     w2     n       prob
##    <char> <char> <int>      <num>
## 1:     of    the 36360 0.21418978
## 2:     in    the 35788 0.25202107
## 3:     to    the 18694 0.07831752
## 4:    for    the 18092 0.18598244
## 5:     on    the 16995 0.23981543
## 6:     to     be 13913 0.05828777

## [1] "Trigrams"

##         w1     w2     w3     n      prob
##     <char> <char> <char> <int>     <num>
## 1:    one     of    the  2919 0.4646609
## 2:      a    lot     of  2625 0.6578947
## 3: thanks    for    the  2353 0.5429165
## 4:     to     be      a  1655 0.1208912
## 5:  going     to     be  1570 0.2213450
## 6:      i   want     to  1329 0.5562997

## [1] "Fourgrams"

##         w1     w2     w3     w4     n      prob
##     <char> <char> <char> <char> <int>     <num>
## 1:    the    end     of    the   619 0.5149750
## 2: thanks    for    the follow   610 0.2616903
## 3:     at    the    end     of   564 0.8506787
## 4:    for    the  first   time   543 0.7520776
## 5:    the   rest     of    the   529 0.5794085
## 6:     at    the   same   time   411 0.8422131
```

Step 3: Build prediction model

Using the calculated N-gram probabilities, a prediction function was created to suggest the next word based on input text. The function implements a backoff strategy, leveraging N-grams of descending lengths when a match isn't found at higher levels.

Here is an example of the implementation of the prediction function

```
# Test prediction
input_text = "You're the reason why I smile everyday. Can you follow me please? It would mean the"

predict_next_word(input_text, fourgram_probs, trigram_probs, bigram_probs, unigram_probs, lambda = c(0.
```

```
## [1] "world" "first" "same"
```

Results and Next Steps

Results

- Successfully loaded and processed datasets (blogs, news, Twitter).
- Constructed and analyzed N-grams (unigrams, bigrams, trigrams, and fourgrams).
- Developed a prediction function using N-gram probabilities.

Next Steps

- Evaluate and optimize the model's performance for real-time text prediction.
- Integrate the prediction model into a Shiny app.

Conclusion

This exploratory analysis demonstrates that the data has been successfully loaded and cleaned. The next step is to implement a real-time prediction algorithm that can make use of the processed text data. The Shiny app will allow users to interact with the model and see real-time results.