layout

Rokesh

2024-03-26

```r
library(tidyverse)
library(stringi)

word_count <- read.csv("Data/unigram_freq1.csv")

# Convert the words to lowercase to avoid case sensitivity
word_count$word <- tolower(word_count$word)

# Create a vector of possible pairs of alphabets
pair_space <- combn(letters, 2, FUN = paste, collapse = "")
pair_space <- c(pair_space, stri_reverse(pair_space))
pair_space <- c(pair_space, paste0(letters, letters))

# Function to count occurrences of each alphabet
count_pattern <- function(word_vec, pattern_vec) {
    pattern_counts <- sapply(pattern_vec, function(pattern_vec) {
        sum(str_count(word_vec, (pattern_vec)))
    })
    return(pattern_counts)
}

# Extract all words from the data frame
all_words <- word_count$word

# Count occurrences of each alphabet
alphabet_count = count_pattern(all_words, letters)

# Count adjacent alphabet pairs
pair_count <- count_pattern(all_words, pair_space)

# Create data frames for the two tables
alphabet_df <- data.frame(letter = names(alphabet_count), count = alphabet_count)
pair_df <- data.frame(pairs = names(pair_count), count = pair_count)

# Rearranging the tables based on the count column
alphabet_df <- alphabet_df |>
    arrange(desc(count))
pair_df <- pair_df |>
    arrange(desc(count))

# Planning the layout layout_2: method analogous to tree structure Step 1:
# Splitting the pairs
```

```r
pair_split_df <- pair_df |>
    mutate(first = substr(pairs, 1, 1), second = substr(pairs, 2, 2)) |>
    relocate(c(first, second), .before = count)

# Step 2: Using loop to set the order Taking 'e' as the first among the rank
# due to its count
rank <- character(length = 26)

for (i in seq_along(letters)) {
    if (i == 1) {
        rank[i] <- "e"
    } else {
        rank[i] <- pair_split_df |>
            filter(first == rank[i - 1]) |>
            filter(!second %in% rank) |>
            slice_max(count) |>
            pull(second)
    }
}
names(rank) <- rep(c("l", "r"), length.out = length(rank))

rank_df <- data.frame(letters = rank, position = names(rank))

rank_df <- rank_df |>
    arrange((position))

# Mapping based on the count
l <- list()  # initialising list to store the key cluster
bound_upper <- 1
bound_lower <- 0
cluster_vector <- c(1, 1, 1, 2, 2, 2, 4, 4)

# excluding the top nine alphabets to use for the home row
alph_nine <- alphabet_df$letter[1:9]
alph_nine_removed <- alphabet_df$letter[!alphabet_df$letter %in% alph_nine]

# building the tree (linear)
for (i in seq_along(cluster_vector)) {
    l[[i]] <- character(cluster_vector[i])
    for (j in seq_len(cluster_vector[i])) {
        if (!j%%2 == 0) {
            l[[i]][j] <- alph_nine_removed[17 - bound_lower]
            bound_lower <- bound_lower + 1
        } else {
            l[[i]][j] <- alph_nine_removed[bound_upper]
            bound_upper <- bound_upper + 1
        }
    }
}

# Mapping the key cluster to the existing layout
names(l[[1]]) <- "p"
names(l[[2]]) <- c("o")
names(l[[3]]) <- c("i")
names(l[[4]]) <- c("c", "e")
names(l[[5]]) <- c("x", "w")
names(l[[6]]) <- c("z", "q")
names(l[[7]]) <- c("b", "r", "t", "v")
names(l[[8]]) <- c("u", "y", "m", "n")

unlisted_l <- unlist(l)
existing_layout <- list(toprow = c("q", "w", "e", "r", "t", "y", "u", "i", "o", "p"),
    homerow = c(unlist(str_split("asdfghjkl", ""))), bottomrow = c("z", "x", "c",
        "v", "b", "n", "m"))
layout_2 <- list(toprow = character(length = 10), homerow = character(length = 9),
    bottomrow = character(length = 7))

for (i in seq_along(existing_layout)) {
    if (i == 2) {
        layout_2[[2]] <- unlist(str_split("tnosleair", ""))

    } else {
        for (j in seq_along(existing_layout[[i]])) {
            names(layout_2[[i]][j]) <- existing_layout[[i]][j]
            layout_2[[i]][j] <- unlisted_l[existing_layout[[i]][j]]
        }
    }
}

# splitting the alphabets into l and r
layout_2_l <- unlist(str_split("hkxjqeairgb", ""))
layout_2_r <- unlist(str_split("mdcuytnoslwvzpf", ""))

# layout_1 by populating clusters cluster calculation no of clusters : 2 (left
# and right)
parent_sum <- sum(alphabet_count)/2

# populating right-hand cluster
cluster_1 <- character(length = 11)
distance_1 <- integer(length = 11)
for (i in seq_len(11)) {
    if (i == 1) {
        cluster_1[1] <- names(alphabet_count[alphabet_count == max(alphabet_count)])
        distance_1[1] <- max(alphabet_count)
    } else {
        current_average <- (parent_sum - sum(distance_1))/(11 - (i - 1))
        current_alphabet_count <- alphabet_count[!names(alphabet_count) %in% cluster_1]
        diff <- current_alphabet_count - current_average
        abs_diff <- abs(diff)
        names(abs_diff) <- names(current_alphabet_count)
        distance_1[i] <- alphabet_count[names(abs_diff[abs_diff == min(abs_diff)])]
        cluster_1[i] <- names(abs_diff[abs_diff == min(abs_diff)])
    }
}
alphabet_count <- alphabet_count[order(alphabet_count, decreasing = TRUE)]
```

```r
# populating left-hand cluster
cluster_2 <- names(alphabet_count[(!names(alphabet_count) %in% cluster_1)])
cluster_1_df <- data.frame(cluster_members = cluster_1, count = distance_1)
cluster_2_df <- alphabet_df |>
    filter(letter %in% cluster_2)
cluster_1_df <- cluster_1_df |>
    arrange(desc(count))

layout_1 <- list()
layout_1[[1]] <- unlist(str_split("qzwbycmphg", ""))
layout_1[[2]] <- unlist(str_split("sriaolent", ""))
layout_1[[3]] <- unlist(str_split("jxvkfdu", ""))

# testing starts function to remove non-alphabetic characters
remove_non_alphabetic <- function(text) {
    # use regular expression to keep only alphabetic characters
    cleaned_text <- gsub("[^a-zA-Z]", "", text)
    # convert to lowercase
    cleaned_text <- tolower(cleaned_text)
    return(cleaned_text)
}

# other functions to calculate left and right hand counts
hand_count <- function(testfile_alphabet_count, cluster) {
    return(sum(testfile_alphabet_count[names(testfile_alphabet_count) %in% cluster]))
}

diff_hand <- function(left_count, right_count) {
    return(abs(left_count - right_count))
}

# function to populate pairs from clusters
concatenate_elements <- function(x, y) {
    paste(x, y, sep = "")
}

# Test_1 Begins reading file contents
file_content <- readLines("Data/mobydick.txt")

# apply the function to each line of the file
cleaned_content_test_1 <- sapply(file_content, remove_non_alphabetic)

# Combine the lines into a single string
cleaned_text_test_1 <- paste(cleaned_content_test_1, collapse = "")

# counts of different alphabets in the test file no. 1
test_1_alphabet_count <- count_pattern(cleaned_text_test_1, letters)
# layout_1
left_count_layout_1_test_1 <- hand_count(test_1_alphabet_count, layout_2_l)
right_count_layout_1_test_1 <- hand_count(test_1_alphabet_count, layout_2_r)
diff_layout_1_test_1 <- diff_hand(left_count_layout_1_test_1, right_count_layout_1_test_1)

pairs_layout_1 <- c(c(outer(layout_2_l, layout_2_r, concatenate_elements)), c(outer(layout_2_r,
    layout_2_l, concatenate_elements)))
pair_count_layout_1_test_1 <- sum(count_pattern(cleaned_text_test_1, pairs_layout_1))

# layout_2
left_count_layout_2_test_1 <- hand_count(test_1_alphabet_count, cluster_2)
right_count_layout_2_test_1 <- hand_count(test_1_alphabet_count, cluster_1)
diff_layout_2_test_1 <- diff_hand(left_count_layout_2_test_1, right_count_layout_2_test_1)

pairs_layout_2 <- c(c(outer(cluster_1, cluster_2, concatenate_elements)), c(outer(cluster_2,
    cluster_1, concatenate_elements)))
pair_count_layout_2_test_1 <- sum(count_pattern(cleaned_text_test_1, pairs_layout_2))

# Existing layout
cluster_l_existing <- unlist(str_split("qwertasdfgzxcvb", ""))
cluster_r_existing <- unlist(str_split("yuiophjklnm", ""))
left_count_existing_layout <- sum(test_1_alphabet_count[names(test_1_alphabet_count) %in%
    cluster_l_existing])
right_count_existing_layout <- sum(test_1_alphabet_count[names(test_1_alphabet_count) %in%
    cluster_r_existing])
diff_existing_layout_test_1 <- abs(left_count_existing_layout - right_count_existing_layout)
existing_layout_pairs <- c(c(outer(cluster_l_existing, cluster_r_existing, concatenate_elements)),
    c(outer(cluster_r_existing, cluster_l_existing, concatenate_elements)))
pair_count_existing_test_1 <- sum(count_pattern(cleaned_text_test_1, existing_layout_pairs))

# Test_2 Begins reading file contents
file_content <- readLines("Data/adventures_of_tom_sawyer.txt")

# apply the function to each line of the file
cleaned_content_test_2 <- sapply(file_content, remove_non_alphabetic)

# Combine the lines into a single string
cleaned_text_test_2 <- paste(cleaned_content_test_2, collapse = "")

# counts of different alphabets in the test file no. 1
test_2_alphabet_count <- count_pattern(cleaned_text_test_2, letters)
# layout_1
left_count_layout_1_test_2 <- hand_count(test_2_alphabet_count, layout_2_l)
right_count_layout_1_test_2 <- hand_count(test_2_alphabet_count, layout_2_r)
diff_layout_1_test_2 <- diff_hand(left_count_layout_1_test_2, right_count_layout_1_test_2)

pairs_layout_1 <- c(c(outer(layout_2_l, layout_2_r, concatenate_elements)), c(outer(layout_2_r,
    layout_2_l, concatenate_elements)))
pair_count_layout_1_test_2 <- sum(count_pattern(cleaned_text_test_2, pairs_layout_1))

# layout_2
left_count_layout_2_test_2 <- hand_count(test_2_alphabet_count, cluster_2)
right_count_layout_2_test_2 <- hand_count(test_2_alphabet_count, cluster_1)
diff_layout_2_test_2 <- diff_hand(left_count_layout_2_test_2, right_count_layout_2_test_2)

pairs_layout_2 <- c(c(outer(cluster_1, cluster_2, concatenate_elements)), c(outer(cluster_2,
    cluster_1, concatenate_elements)))
pair_count_layout_2_test_2 <- sum(count_pattern(cleaned_text_test_2, pairs_layout_2))
```

```r
# Existing layout
cluster_l_existing <- unlist(str_split("qwertasdfgzxcvb", ""))
cluster_r_existing <- unlist(str_split("yuiophjklnm", ""))
left_count_existing_layout <- sum(test_2_alphabet_count[names(test_2_alphabet_count) %in%
    cluster_l_existing])
right_count_existing_layout <- sum(test_2_alphabet_count[names(test_2_alphabet_count) %in%
    cluster_r_existing])
diff_existing_layout_test_2 <- abs(left_count_existing_layout - right_count_existing_layout)
existing_layout_pairs <- c(c(outer(cluster_l_existing, cluster_r_existing, concatenate_elements)),
    c(outer(cluster_r_existing, cluster_l_existing, concatenate_elements)))
pair_count_existing_test_2 <- sum(count_pattern(cleaned_text_test_2, existing_layout_pairs))

avg_diff_layout_1 <- ((diff_layout_1_test_1 - diff_existing_layout_test_1)/diff_existing_layout_test_1
    (diff_layout_1_test_2 - diff_existing_layout_test_2)/diff_existing_layout_test_2)/2 *
    100
avg_diff_layout_2 <- ((diff_layout_2_test_1 - diff_existing_layout_test_1)/diff_existing_layout_test_1
    (diff_layout_2_test_2 - diff_existing_layout_test_2)/diff_existing_layout_test_2)/2 *
    100
avg_pair_count_layout_1 <- ((pair_count_layout_1_test_1 - pair_count_existing_test_1)/pair_count_existi
    (pair_count_layout_1_test_2 - pair_count_existing_test_2)/pair_count_existing_test_2)/2 *
    100
avg_pair_count_layout_2 <- ((pair_count_layout_2_test_1 - pair_count_existing_test_1)/pair_count_existi
    (pair_count_layout_2_test_2 - pair_count_existing_test_2)/pair_count_existing_test_2)/2 *
    100

print("layout_1")
```

```
## [1] "layout_1"
```

```r
for (i in (layout_1)) print(i)
```

```
##  [1] "q" "z" "w" "b" "y" "c" "m" "p" "h" "g"
## [1] "s" "r" "i" "a" "o" "l" "e" "n" "t"
## [1] "j" "x" "v" "k" "f" "d" "u"
```

```r
print("layout_2")
```

```
## [1] "layout_2"
```

```r
for (i in layout_2) print(i)
```

```
##  [1] "m" "d" "c" "u" "y" "h" "k" "x" "j" "q"
## [1] "t" "n" "o" "s" "l" "e" "a" "i" "r"
## [1] "w" "v" "z" "p" "f" "g" "b"
```

```r
avg_diff_layout_1
```

```
## [1] -24.32371
```

```r
avg_diff_layout_2
```

```
## [1] -30.82528
```

```r
avg_pair_count_layout_1
```

```
## [1] 1.320107
```

```r
avg_pair_count_layout_2
```

```
## [1] -0.8712978
```