Amazon Review Analysis Code - Airpods and Alternative

Chae Yeon Ryu

1 Code Appendix

```
####### All libraries used #######

library(cmu.textstat)
library(tidyverse)
library(quanteda)
library(nnet)
library(quanteda.textstats)
library(udpipe)
library(future.apply)
library(nFactors)
library(spacyr)
library(pseudobibeR)
library(ggplot2)
library(ggpubr)
library(scales)
library(ggraph)
library(tm)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
library(syuzhet)
library(tidytext)
library(dplyr)
library(stringr)
library(reshape2)
library(janeaustenr)
library(tidygraph)
library(RTextTools)
library(e1071)



####### read files #######

au_neg_filelist <- list.files("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/review/au_negative
    full.names = T)
au_pos_filelist <- list.files("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/review/au_positive
    full.names = T)
count_neg_filelist <- list.files("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/review/count_neg
    full.names = T)
```

```
count_pos_filelist <- list.files("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/review/count_pos
    full.names = T)
meta <- read.csv("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/final_metaa.csv",
    header = TRUE)
aut <- c(au_neg_filelist, au_pos_filelist)


####### tokens original without tag #######

au_neg_tokens <- au_neg_filelist %>%
    readtext::readtext() %>%
    corpus() %>%
    tokens(what = "fastestword", remove_punct = TRUE, remove_separators = TRUE)

au_pos_tokens <- au_pos_filelist %>%
    readtext::readtext() %>%
    corpus() %>%
    tokens(what = "fastestword", remove_punct = TRUE, remove_separators = TRUE)

count_neg_tokens <- count_neg_filelist %>%
    readtext::readtext() %>%
    corpus() %>%
    tokens(what = "fastestword", remove_punct = TRUE, remove_separators = TRUE)

count_pos_tokens <- count_pos_filelist %>%
    readtext::readtext() %>%
    corpus() %>%
    tokens(what = "fastestword", remove_punct = TRUE, remove_separators = TRUE)


sw <- stopwords("english")
# words that are unnecessary for analysis because they are commonly
# used by all reviews
unwords = c("and", "for", "have", "not", "the", "this", "which", "will",
    "would", "that", "are", "may", "its", "were", "than", "those", "between",
    "has", "who", "there", "has", "their", "from", "with", "ought", "must",
    "could", "every", "of", "on", "by", "upon", "to", "in", "while", "within",
    "among", "might", "can")

au_pos_tokens <- tokens_tolower(tokens_remove(au_pos_tokens, c(sw, unwords)))
au_neg_tokens <- tokens_tolower(tokens_remove(au_neg_tokens, c(sw, unwords)))
count_pos_tokens <- tokens_tolower(tokens_remove(count_pos_tokens, c(sw,
    unwords)))
count_neg_tokens <- tokens_tolower(tokens_remove(count_neg_tokens, c(sw,
    unwords)))



####### turn into data frame for explanatory data analysis #######

meta$customer_id = as.factor(meta$customer_id)

au_neg_mts <- au_neg_tokens %>%
```

```
    dfm() %>%
    as.data.frame() %>%
    mutate(doc_id = str_remove_all(doc_id, ".rtf")) %>%
    remove_rownames %>%
    column_to_rownames(var = "doc_id")

au_neg_mts$tcount <- rowSums(au_neg_mts)
au_neg_mts$customer_id <- rownames(au_neg_mts)
au_neg_mts <- au_neg_mts[c("customer_id", "tcount")]
au_neg_mts <- au_neg_mts %>%
    mutate(customer_id = as.factor(customer_id)) %>%
    left_join(select(meta, customer_id, star, year, authentic), by = "customer_id")

au_pos_mts <- au_pos_tokens %>%
    dfm() %>%
    as.data.frame() %>%
    mutate(doc_id = str_remove_all(doc_id, ".rtf")) %>%
    remove_rownames %>%
    column_to_rownames(var = "doc_id")

au_pos_mts$tcount <- rowSums(au_pos_mts)
au_pos_mts$customer_id <- rownames(au_pos_mts)
au_pos_mts <- au_pos_mts[c("customer_id", "tcount")]
au_pos_mts <- au_pos_mts %>%
    mutate(customer_id = as.factor(customer_id)) %>%
    left_join(select(meta, customer_id, star, year, authentic), by = "customer_id")

count_neg_mts <- count_neg_tokens %>%
    dfm() %>%
    as.data.frame() %>%
    mutate(doc_id = str_remove_all(doc_id, ".rtf")) %>%
    remove_rownames %>%
    column_to_rownames(var = "doc_id")

count_neg_mts$tcount <- rowSums(count_neg_mts)
count_neg_mts$customer_id <- rownames(count_neg_mts)
count_neg_mts <- count_neg_mts[c("customer_id", "tcount")]
count_neg_mts <- count_neg_mts %>%
    mutate(customer_id = as.factor(customer_id)) %>%
    left_join(select(meta, customer_id, star, year, authentic), by = "customer_id")


count_pos_mts <- count_pos_tokens %>%
    dfm() %>%
    as.data.frame() %>%
    mutate(doc_id = str_remove_all(doc_id, ".rtf")) %>%
    remove_rownames %>%
    column_to_rownames(var = "doc_id")

count_pos_mts$tcount <- rowSums(count_pos_mts)
count_pos_mts$customer_id <- rownames(count_pos_mts)
count_pos_mts <- count_pos_mts[c("customer_id", "tcount")]
count_pos_mts <- count_pos_mts %>%
```

```
    mutate(customer_id = as.factor(customer_id)) %>%
    left_join(select(meta, customer_id, star, year, authentic), by = "customer_id")

all_mts = rbind(au_pos_mts, au_neg_mts, count_pos_mts, count_neg_mts)


####### code for Figure 1 #######

all_mts = rbind(au_pos_mts, au_neg_mts, count_pos_mts, count_neg_mts)
a = spread(all_mts, star, tcount)
all_mts_long <- melt(a[3:8], id = colnames(a)[3])
all_mts_long$product_type = ifelse(all_mts_long$authentic == "Y", "App",
    "Alt")
ggplot(all_mts_long, aes(x = variable, y = value, fill = product_type)) +
    geom_boxplot() + labs(x = "Review Rating", y = "Token Counts", legend.title = "product type",
    legend.text = c("Alt", "App"))


####### code for Table 1 #######

table1 = matrix(c(sum(au_pos_mts$tcount), sum(au_neg_mts$tcount), sum(count_pos_mts$tcount),
    sum(count_neg_mts$tcount), c(nrow(au_pos_mts), nrow(au_neg_mts), nrow(count_pos_mts),
        nrow(count_neg_mts))), ncol = 4, nrow = 2, byrow = TRUE)
colnames(table1) <- c("App_good", "App_bad", "Alt_good", "Alt_bad")
rownames(table1) <- c("Words Counts (tokens)", "Number of Reviews")

knitr::kable(table1, "simple", caption = "Review Subcorpora on AirPods and Alternative Earbuds")




####### code for Chi-square test #######

test1 = chisq.test(all_mts$tcount, all_mts$authentic)
test1 = test1$p.value  # not significant

test2 = chisq.test(all_mts$tcount, all_mts$star)
test2 = test2$p.value  # not significant


chi_table = matrix(c(test1, test2), nrow = 2, ncol = 1, byrow = TRUE)
rownames(chi_table) <- c("Authentic vs. Length of Reviews", "Review Score vs. Length of Reviews")
colnames(chi_table) <- c("p-value")




####### code for Table 2 #######

au_pos_wc <- frequency_table(au_pos_tokens)
au_neg_wc <- frequency_table(au_neg_tokens)
count_pos_wc <- frequency_table(count_pos_tokens)
```

```
count_neg_wc <- frequency_table(count_neg_tokens)


table2 = matrix(c(head(au_pos_wc[[1]], 10), head(au_neg_wc[[1]], 10), head(count_pos_wc[[1]],
    10), head(count_neg_wc[[1]], 10)), ncol = 4, nrow = 10, byrow = TRUE)
colnames(table2) <- c("App_good", "App_bad", "Alt_good", "Alt_bad")

knitr::kable(table2, "simple", caption = "Most Frequent Tokens (descending order)")




####### code for Sentiment Analysis - Figure 2 #######

em_au_pos <- get_nrc_sentiment(readtext::readtext(au_pos_filelist)[[2]])
em_au_neg <- get_nrc_sentiment(readtext::readtext(au_neg_filelist)[[2]])
em_count_pos <- get_nrc_sentiment(readtext::readtext(count_pos_filelist)[[2]])
em_count_neg <- get_nrc_sentiment(readtext::readtext(count_neg_filelist)[[2]])

a <- get_nrc_sentiment(c(readtext::readtext(au_pos_filelist)[[2]], readtext::readtext(au_neg_filelist)[
b <- get_nrc_sentiment(c(readtext::readtext(count_pos_filelist)[[2]], readtext::readtext(count_neg_file


em_au_pos <- get_nrc_sentiment(readtext::readtext(au_pos_filelist)[[2]])
em_au_neg <- get_nrc_sentiment(readtext::readtext(au_neg_filelist)[[2]])
em_count_pos <- get_nrc_sentiment(readtext::readtext(count_pos_filelist)[[2]])
em_count_neg <- get_nrc_sentiment(readtext::readtext(count_neg_filelist)[[2]])

a <- get_nrc_sentiment(c(readtext::readtext(au_pos_filelist)[[2]], readtext::readtext(au_neg_filelist)[
b <- get_nrc_sentiment(c(readtext::readtext(count_pos_filelist)[[2]], readtext::readtext(count_neg_file


au_pos_td <- data.frame(t(em_au_pos))
au_pos_td_new <- data.frame(rowSums(au_pos_td[1:50]))
names(au_pos_td_new)[1] <- "count"
au_pos_td_new <- cbind(sentiment = rownames(au_pos_td_new), au_pos_td_new)
rownames(au_pos_td_new) <- NULL
au_pos_td_new2 <- au_pos_td_new[1:10, ]

aupp <- quickplot(sentiment, data = au_pos_td_new2, weight = count, geom = "bar",
    fill = sentiment, ylab = "count") + theme(axis.text.x = element_blank()) +
    ggtitle("App_good")

au_neg_td <- data.frame(t(em_au_neg))
au_neg_td_new <- data.frame(rowSums(au_neg_td[1:50]))
names(au_neg_td_new)[1] <- "count"
au_neg_td_new <- cbind(sentiment = rownames(au_neg_td_new), au_neg_td_new)
rownames(au_neg_td_new) <- NULL
au_neg_td_new2 <- au_neg_td_new[1:10, ]

aunp <- quickplot(sentiment, data = au_neg_td_new2, weight = count, geom = "bar",
    fill = sentiment, ylab = "count") + theme(axis.text.x = element_blank()) +
    ggtitle("App_bad")
```

```
count_pos_td <- data.frame(t(em_count_pos))
count_pos_td_new <- data.frame(rowSums(count_pos_td[1:50]))
names(count_pos_td_new)[1] <- "count"
count_pos_td_new <- cbind(sentiment = rownames(count_pos_td_new), count_pos_td_new)
rownames(count_pos_td_new) <- NULL
count_pos_td_new2 <- count_pos_td_new[1:10, ]

cpp <- quickplot(sentiment, data = count_pos_td_new2, weight = count, geom = "bar",
    fill = sentiment, ylab = "count") + theme(axis.text.x = element_blank()) +
    ggtitle("Alt_good")

count_neg_td <- data.frame(t(em_count_neg))
count_neg_td_new <- data.frame(rowSums(count_neg_td[1:50]))
names(count_neg_td_new)[1] <- "count"
count_neg_td_new <- cbind(sentiment = rownames(count_neg_td_new), count_neg_td_new)
rownames(count_neg_td_new) <- NULL
count_neg_td_new2 <- count_neg_td_new[1:10, ]

cnp <- quickplot(sentiment, data = count_neg_td_new2, weight = count, geom = "bar",
    fill = sentiment, ylab = "count") + theme(axis.text.x = element_blank()) +
    ggtitle("Alt_bad")


allSentiment = ggarrange(aupp, aunp, cpp, cnp, ncol = 2, nrow = 2, widths = c(4,
    4), common.legend = TRUE, legend = "right")
allSentiment
# annotate_figure(allSentiment, top = text_grob('Diagram 1: Review
# Sentiment', face = 'bold', size = 14))




####### code for Sentiment Analysis - Figure 3 #######

dfap <- data_frame(customer_id = readtext::readtext(au_pos_filelist)[[1]],
    text = readtext::readtext(au_pos_filelist)[[2]])
dfap <- dfap %>%
    unnest_tokens(id, text)
colnames(dfap)[2] <- "word"

dfan <- data_frame(customer_id = readtext::readtext(au_neg_filelist)[[1]],
    text = readtext::readtext(au_neg_filelist)[[2]])
dfan <- dfan %>%
    unnest_tokens(id, text)
colnames(dfan)[2] <- "word"

dfcp <- data_frame(customer_id = readtext::readtext(count_pos_filelist)[[1]],
    text = readtext::readtext(count_pos_filelist)[[2]])
dfcp <- dfcp %>%
    unnest_tokens(id, text)
colnames(dfcp)[2] <- "word"
```

```
dfcn <- data_frame(customer_id = readtext::readtext(count_neg_filelist)[[1]],
    text = readtext::readtext(count_neg_filelist)[[2]])
dfcn <- dfcn %>%
    unnest_tokens(id, text)
colnames(dfcn)[2] <- "word"



ap_word_counts <- dfap %>%
    inner_join(get_sentiments("bing"), by = "word")
ap_word_counts <- ap_word_counts %>%
    count(word, sentiment, sort = TRUE)

an_word_counts <- dfan %>%
    inner_join(get_sentiments("bing"), by = "word")
an_word_counts <- an_word_counts %>%
    count(word, sentiment, sort = TRUE)

cp_word_counts <- dfcp %>%
    inner_join(get_sentiments("bing"), by = "word")
cp_word_counts <- cp_word_counts %>%
    count(word, sentiment, sort = TRUE)

cn_word_counts <- dfcn %>%
    inner_join(get_sentiments("bing"), by = "word")
cn_word_counts <- cn_word_counts %>%
    count(word, sentiment, sort = TRUE)



apwc <- ap_word_counts %>%
    group_by(sentiment) %>%
    top_n(10) %>%
    ggplot(aes(reorder(word, n), n, fill = sentiment)) + geom_bar(alpha = 0.8,
    stat = "identity", show.legend = FALSE) + facet_wrap(~sentiment, scales = "free_y") +
    labs(y = "Contribution to sentiment", x = NULL) + coord_flip()

anwc <- an_word_counts[an_word_counts$sentiment == "negative", ] %>%
    group_by(sentiment) %>%
    top_n(10) %>%
    ggplot(aes(reorder(word, n), n, fill = sentiment)) + geom_bar(alpha = 0.8,
    stat = "identity", show.legend = FALSE) + facet_wrap(~sentiment, scales = "free_y") +
    labs(y = "App_bad", x = NULL) + coord_flip()

cpwc <- cp_word_counts %>%
    group_by(sentiment) %>%
    top_n(10) %>%
    ggplot(aes(reorder(word, n), n, fill = sentiment)) + geom_bar(alpha = 0.8,
    stat = "identity", show.legend = FALSE) + facet_wrap(~sentiment, scales = "free_y") +
    labs(y = "Contribution to sentiment", x = NULL) + coord_flip()

cnwc <- cn_word_counts[cn_word_counts$sentiment == "negative", ] %>%
    group_by(sentiment) %>%
```

```
    top_n(10) %>%
    ggplot(aes(reorder(word, n), n, fill = sentiment)) + geom_bar(alpha = 0.8,
    stat = "identity", show.legend = FALSE) + facet_wrap(~sentiment, scales = "free_y") +
    labs(y = "Alt_bad", x = NULL) + coord_flip()

ggarrange(anwc, cnwc, ncol = 2, nrow = 1, widths = c(3, 3), common.legend = TRUE,
    legend = "right")


####### code for ngram Analysis Experiment #######

au_grams <- tokens_skipgrams(au_neg_tokens, n = 4:5, skip = 0, concatenator = " ")
count_grams <- tokens_skipgrams(count_neg_tokens, n = 4:5, skip = 0, concatenator = " ")

au_grams <- tokens_select(au_grams, "battery", selection = "keep", valuetype = "regex",
    case_insensitive = T)
count_grams <- tokens_select(count_grams, "battery", selection = "keep",
    valuetype = "regex", case_insensitive = T)
# Filter(Negate(is.null), au_grams) Filter(Negate(is.null),
# count_grams)
a = as.vector(unlist(au_grams))
b = as.vector(unlist(count_grams))

au_pos_bigram = au_pos_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

au_pos_bigram_sep <- au_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ")

au_pos_bigram_united <- au_pos_bigram_sep %>%
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

# au_pos_bigram_united %>% count(bigram, sort = TRUE)

au_pos_bigram <- au_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(!first %in% stop_words$word) %>%
    filter(!second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

au_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(first == "battery" | second == "battery") %>%
    count(battery = str_c(first, second, sep = " "), sort = TRUE)


au_neg_bigram = au_neg_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

au_neg_bigram_sep <- au_neg_bigram %>%
```

```
    separate(bigram, into = c("first", "second"), sep = " ")

au_neg_bigram_united <- au_neg_bigram_sep %>%
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

# au_neg_bigram_united %>% count(bigram, sort = TRUE)

au_neg_bigram <- au_neg_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(!first %in% stop_words$word) %>%
    filter(!second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

au_neg_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(first == "battery" | second == "battery") %>%
    count(battery = str_c(first, second, sep = " "), sort = TRUE)


count_pos_bigram = count_pos_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

count_pos_bigram_sep <- count_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ")

count_pos_bigram_united <- count_pos_bigram_sep %>%
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

# count_pos_bigram_united %>% count(bigram, sort = TRUE)

count_pos_bigram <- count_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(!first %in% stop_words$word) %>%
    filter(!second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

count_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(first == "battery" | second == "battery") %>%
    count(battery = str_c(first, second, sep = " "), sort = TRUE)


count_neg_bigram = count_neg_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

count_neg_bigram_sep <- count_neg_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ")

count_neg_bigram_united <- count_neg_bigram_sep %>%
```

```
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

# au_neg_bigram_united %>% count(bigram, sort = TRUE)

count_neg_bigram <- count_neg_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(!first %in% stop_words$word) %>%
    filter(!second %in% stop_words$word) %>%
    unite(bigram, c(first, second), sep = " ")

count_neg_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ") %>%
    filter(first == "battery" | second == "battery") %>%
    count(battery = str_c(first, second, sep = " "), sort = TRUE)



####### code for Bigram Analysis #######

au_pos_bigram = au_pos_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

au_pos_bigram_sep <- au_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ")

au_neg_bigram = au_neg_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

au_neg_bigram_sep <- au_neg_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ")

count_pos_bigram = count_pos_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

count_pos_bigram_sep <- count_pos_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ")

count_neg_bigram = count_neg_filelist %>%
    readtext::readtext() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

count_neg_bigram_sep <- count_neg_bigram %>%
    separate(bigram, into = c("first", "second"), sep = " ")




count_pos_bigram_counts <- count_pos_bigram_sep %>%
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
```

```
    count(first, second, sort = TRUE)

count_pos_bigram_graph <- count_pos_bigram_counts %>%
    filter(n > 1) %>%
    as_tbl_graph()

arrow <- grid::arrow(type = "closed", length = unit(0.07, "inches"))

cpbg <- ggraph(count_pos_bigram_graph, layout = "fr") + geom_edge_link(aes(alpha = n),
    show.legend = F, arrow = arrow, end_cap = circle(0.02, "inches")) +
    geom_node_point(color = "lightblue", size = 2) + geom_node_text(aes(label = name),
    vjust = 0.1, hjust = 0.1, size = 2.5) + ggtitle("Alt_good bigrams")

######

count_neg_bigram_counts <- count_neg_bigram_sep %>%
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
    count(first, second, sort = TRUE)

count_neg_bigram_graph <- count_neg_bigram_counts %>%
    filter(n > 1) %>%
    as_tbl_graph()

cnbg <- ggraph(count_neg_bigram_graph, layout = "fr") + geom_edge_link(aes(alpha = n),
    show.legend = F, arrow = arrow, end_cap = circle(0.02, "inches")) +
    geom_node_point(color = "pink", size = 2) + geom_node_text(aes(label = name),
    vjust = 0.1, hjust = 0.1, size = 2.5) + ggtitle("Alt_bad bigrams")

#########


au_pos_bigram_counts <- au_pos_bigram_sep %>%
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
    count(first, second, sort = TRUE)

au_pos_bigram_graph <- au_pos_bigram_counts %>%
    filter(n > 2) %>%
    as_tbl_graph()

apbg <- ggraph(au_pos_bigram_graph, layout = "fr") + geom_edge_link(aes(alpha = n),
    show.legend = F, arrow = arrow, end_cap = circle(0.02, "inches")) +
    geom_node_point(color = "lightblue", size = 2) + geom_node_text(aes(label = name),
    vjust = 0.1, hjust = 0.1, size = 2.5) + ggtitle("App_good bigrams")


#######

au_neg_bigram_counts <- au_neg_bigram_sep %>%
    filter(!first %in% stop_words$word, !second %in% stop_words$word) %>%
    count(first, second, sort = TRUE)

au_neg_bigram_graph <- au_neg_bigram_counts %>%
    filter(n > 1) %>%
```

```
    as_tbl_graph()

anbg <- ggraph(au_neg_bigram_graph, layout = "fr") + geom_edge_link(aes(alpha = n),
    show.legend = F, arrow = arrow, end_cap = circle(0.02, "inches")) +
    geom_node_point(color = "pink", size = 2) + geom_node_text(aes(label = name),
    vjust = 0.1, hjust = 0.1, size = 2.5) + ggtitle("App_bad bigrams")

ggarrange(apbg, anbg, cpbg, cnbg, ncol = 2, nrow = 2, common.legend = TRUE,
    legend = "right")




####### code for Collocational Analysis #######

cppc <- collocates_by_MI(count_pos_tokens, "price") %>%
    filter(MI_1 >= 5)
cpac <- collocates_by_MI(count_pos_tokens, "airpods") %>%
    filter(MI_1 >= 5)
# cnac <- collocates_by_MI(count_neg_tokens, 'airpods') %>%
# filter(MI_1 >= 3) cpqc <- collocates_by_MI(count_pos_tokens,
# 'quality') %>% filter(MI_1 >= 3)

netcp1 <- col_network(cppc, cpac)
# netcp2 <- col_network(cpac) netcp3 <- col_network(cnac)


ggraph(netcp1, weight = link_weight, layout = "stress") + geom_edge_link(color = "gray80",
    alpha = 0.75) + geom_node_point(aes(alpha = node_weight, size = 1,
    color = n_intersects)) + geom_node_text(aes(label = label), repel = T,
    size = 2) + scale_alpha(range = c(0.2, 0.9)) + theme_graph() + theme(legend.position = "none")


####### code for ML Modeling #######

files_list <- list.files("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/review/all",
    full.names = T)
reviews <- files_list %>%
    readtext::readtext()

tst_meta <- read.csv("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/test/test_meta.csv",
    header = TRUE)

all_meta <- meta %>%
    dplyr::select(customer_id, star, authentic)
all_meta$review <- ifelse(all_meta$star >= 4, "good", "bad")
all_meta <- all_meta %>%
    dplyr::select(customer_id, authentic, review)

all_meta$customer_id = as.character(all_meta$customer_id)
all_meta$authentic = as.character(all_meta$authentic)
all_meta$review = as.character(all_meta$review)
```

```
test_meta <- tst_meta %>%
    dplyr::select(customer_id, star, review)

test_meta$customer_id = as.character(test_meta$customer_id)
test_meta$star = as.character(test_meta$star)
test_meta$review = as.character(test_meta$review)

reviews$doc_id <- str_remove_all(reviews$doc_id, ".rtf")



reviews <- reviews %>%
    left_join(all_meta, by = c(doc_id = "customer_id")) %>%
    as_tibble()

test_files_list <- list.files("/Users/chae/Desktop/CMU/FALL 2021/36468/final project/test/reviews",
    full.names = T)

test <- test_files_list %>%
    readtext::readtext()

test$doc_id <- str_remove_all(test$doc_id, ".rtf")

test <- test %>%
    left_join(test_meta, by = c(doc_id = "customer_id")) %>%
    as_tibble()


train_matrix = create_matrix(reviews$text, language = "english", removeStopwords = FALSE,
    removeNumbers = TRUE, stemWords = FALSE)
test_matrix = create_matrix(test$text, language = "english", removeStopwords = FALSE,
    removeNumbers = TRUE, stemWords = FALSE)


train_mat = as.matrix(train_matrix)
test_mat = as.matrix(test_matrix)
classifier = naiveBayes(train_mat, as.factor(reviews$review))


predicted = predict(classifier, test_mat)


f = ifelse(c(reviews$review, test$review) == "good", 2, 1)
total_matrix = rbind(reviews[, c(1, 2, 4)], test[, c(1, 2, 4)])
total_matrix = create_matrix(total_matrix$text, language = "english", removeStopwords = FALSE,
    removeNumbers = TRUE, stemWords = FALSE)
container = create_container(total_matrix, as.numeric(f), trainSize = 1:199,
    testSize = 200:219, virgin = FALSE)


set.seed(100)
models = train_models(container, algorithms = c("BAGGING", "BOOSTING",
    "RF", "SVM", "TREE"))
```

```
results = classify_models(container, models)


result1 <- recall_accuracy(test$review, predicted)
result2 <- recall_accuracy(as.numeric(as.factor(test$review)), results[,
    "BAGGING_LABEL"])
result3 <- recall_accuracy(as.numeric(as.factor(test$review)), results[,
    "LOGITBOOST_LABEL"])
result4 <- recall_accuracy(as.numeric(as.factor(test$review)), results[,
    "FORESTS_LABEL"])
result5 <- recall_accuracy(as.numeric(as.factor(test$review)), results[,
    "SVM_LABEL"])
result6 <- recall_accuracy(as.numeric(as.factor(test$review)), results[,
    "TREE_LABEL"])




t1 = matrix(c(result1, result2, result3, result4, result5, result6), ncol = 6,
    byrow = TRUE)
colnames(t1) <- c("Bayes", "Bagging", "Boosting", "RF", "SVM", "Tree")

# Table 3
knitr::kable(t1, "simple", caption = "Predictions Accuracy")



Actual <- matrix(test$review, nrow = 20, ncol = 1)
Bayes <- matrix(ifelse(predicted == "good", "good", "bad"), nrow = 20,
    ncol = 1)
Bagging <- matrix(ifelse(results[, "BAGGING_LABEL"] == 2, "good", "bad"),
    nrow = 20, ncol = 1)
Boosting <- matrix(ifelse(results[, "LOGITBOOST_LABEL"] == 2, "good", "bad"),
    nrow = 20, ncol = 1)
RF <- matrix(ifelse(results[, "FORESTS_LABEL"] == 2, "good", "bad"), nrow = 20,
    ncol = 1)
SVM <- matrix(ifelse(results[, "SVM_LABEL"] == 2, "good", "bad"), nrow = 20,
    ncol = 1)
Tree <- matrix(ifelse(results[, "TREE_LABEL"] == 2, "good", "bad"), nrow = 20,
    ncol = 1)



t2 = cbind(Actual, Bayes, Bagging, Boosting, RF, SVM, Tree)
colnames(t2) <- c("Actual", "Bayes", "Bagging", "Boosting", "RF", "SVM",
    "Tree")


# Table 4
knitr::kable(t2, "simple", caption = "Predictions Comparison")
```