R Operations Code

Frequency, Document Frequency, and Term Frequency-Inverse Document Frequency (TFIDF) for Unweighted and Weighted Unigrams, Bigrams, and Trigrams

```
library(quanteda)
library(readtext)
VARIABLE_READ = readtext("FILE.csv", text_field="COLUMN_HEADER", header = TRUE, encoding = "UTF-8")
<!--
TO WEIGHT CORPUS
VARIABLE_EXPANDED = VARIABLE_READ[rep(seq.int(1,nrow(VARIABLE_READ)), VARIABLE_READ$COLUMN_HEADER_OF_HOW_MANY_TIMES), ]
-->
VARIABLE_CORPUS = corpus(VARIABLE_READ)
VARIABLE_DFM = dfm(VARIABLE_CORPUS, remove = stopwords("english"), stem = TRUE, ngrams = NUMBER_OF_NGRAMS, remove_punct = TRUE, remove_numbers = TRUE)
textstat_frequency(VARIABLE_DFM)
VARIABLE_TFIDF = dfm_tfidf(VARIABLE_DFM)
topfeatures(VARIABLE_TFIDF, HOW_MANY_TO_SHOW)
```

Plotted Frequency Co-occurrence Matrix (FCM) for Unweighted and Weighted Unigrams

```
VARIABLE_FCM = fcm(VARIABLE_DFM, context = c("document"), count = c("frequency"), tri = TRUE)
feat <- names(topfeatures(VARIABLE_FCM, 50))
VARIABLE_FCM_SELECT <- fcm_select(VARIABLE_FCM, pattern = feat, selection = "keep")
textplot_network(VARIABLE_FCM_SELECT)
```

Unweighted Bigram Network Graph

```
library(dplyr)
library(tidyr)
library(tidytext)
library(ggplot2)
library(igraph)
library(ggraph)
library(readtext)
VARIABLE_READ = readtext("./FILE.csv", text_field="text", docid_field = "doc_id", header = TRUE, encoding = "UTF-8")
VARIABLE_BIGRAMS <- VARIABLE_READ %>%
        unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
        separate(bigram, c("word1", "word2"), sep = " ") %>%
        filter(!word1 %in% stop_words$word,
                !word2 %in% stop_words$word) %>%
        count(word1, word2, sort = TRUE)
```

```
BIGRAM_GRAPH <- VARIABLE_BIGRAMS %>%
	filter(n > 1) %>%
	graph_from_data_frame()
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
ggraph(BIGRAM_GRAPH, layout = "fr") +
	geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
		arrow = a, end_cap = circle(.07, 'inches')) +
	geom_node_point(color = "lightblue", size = 5) +
	geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
	theme_void()
```

Weighted Bigram Network Graph

```
library(dplyr)
library(tidyr)
library(tidytext)
library(ggplot2)
library(igraph)
library(ggraph)
library(readtext)
VARIABLE_READ = readtext("./FILE.csv", text_field="text", docid_field = "doc_id", header =
TRUE, encoding = "UTF-8")
VARIABLE_EXPANDED = VARIABLE_READ[rep(seq.int(1,nrow(VARIABLE_READ)),
VARIABLE_READ$AVAILABLE.HITS), ]
VARIABLE_BIGRAMS <- VARIABLE_EXPANDED %>%
	unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
	separate(bigram, c("word1", "word2"), sep = " ") %>%
	filter(!word1 %in% stop_words$word,
		!word2 %in% stop_words$word) %>%
	count(word1, word2, sort = TRUE)
BIGRAM_GRAPH <- VARIABLE_BIGRAMS %>%
	filter(n > 1) %>%
	graph_from_data_frame()
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
ggraph(BIGRAM_GRAPH, layout = "fr") +
	geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
		arrow = a, end_cap = circle(.07, 'inches')) +
	geom_node_point(color = "lightblue", size = 5) +
	geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
	theme_void()
```

Unweighted Trigram Network Graph

```
library(dplyr)
library(tidyr)
library(tidytext)
```

```
library(ggplot2)
library(igraph)
library(ggraph)
library(readtext)
VARIABLE_READ = readtext("./readabl.csv", text_field="text", docid_field = "doc_id", header =
TRUE, encoding = "UTF-8")
VARIABLE_TRIGRAMS <- VARIABLE_READ %>%
        unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
        separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
        filter(!word1 %in% stop_words$word,
                !word2 %in% stop_words$word,
                !word3 %in% stop_words$word) %>%
        count(word1, word2, word3, sort = TRUE)
TRIGRAM_GRAPH <- VARIABLE_TRIGRAMS %>%
        filter(n > 1) %>%
        graph_from_data_frame()
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
ggraph(TRIGRAM_GRAPH, layout = "fr") +
        geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                arrow = a, end_cap = circle(.07, 'inches')) +
        geom_node_point(color = "lightblue", size = 5) +
        geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
        theme_void()
```

Unweighted Trigram Network Graph

```
library(dplyr)
library(tidyr)
library(tidytext)
library(ggplot2)
library(igraph)
library(ggraph)
library(readtext)
VARIABLE_READ = readtext("./readabl.csv", text_field="text", docid_field = "doc_id", header =
TRUE, encoding = "UTF-8")
VARIABLE_EXPANDED = VARIABLE_READ[rep(seq.int(1,nrow(VARIABLE_READ)),
VARIABLE_READ$AVAILABLE.HITS), ]
VARIABLE_TRIGRAMS <- VARIABLE_EXPANDED %>%
        unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
        separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
        filter(!word1 %in% stop_words$word,
                !word2 %in% stop_words$word,
                !word3 %in% stop_words$word) %>%
        count(word1, word2, word3, sort = TRUE)
TRIGRAM_GRAPH <- VARIABLE_TRIGRAMS %>%
```

```
        filter(n > 1) %>%
        graph_from_data_frame()
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
ggraph(TRIGRAM_GRAPH, layout = "fr") +
        geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                arrow = a, end_cap = circle(.07, 'inches')) +
        geom_node_point(color = "lightblue", size = 5) +
        geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
        theme_void()
```

Elbow Method for K-Means Clustering Performed on Four Random Samples of 50% of the Data

```
library(readtext)
library(quanteda)
VARIABLE_READ = readtext("./RANDOM_50%_SAMPLE_1-4.csv", text_field="text",
docid_field = "doc_id", header = TRUE, encoding = "UTF-8")
VARIABLE_CORPUS = corpus(VARIABLE_READ)
VARIABLE_DFM = dfm(VARIABLE_CORPUS, remove = stopwords("english"), stem = TRUE,
ngrams = 1, remove_punct = TRUE, remove_numbers = FALSE)
wss = 2:30
d = textstat_dist(VARIABLE_DFM)
for (i in 2:30) wss[i] = sum(kmeans(d,centers=i,nstart=1)$withinss)
plot(2:30, wss[2:30], type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")
```

STM searchK

```
library(stm)
set.seed(161)
FINDK <- searchK(documents = VARIABLE$documents, vocab = VARIABLE$vocab, K = 2:30,
init.type = "Spectral", data = VARIABLE$meta)
plot(FINDK)
```

Hierarchial Cluster Dendrogram

```
VARIABLE_DIST = textstat_dist(dfm_weight(VARIABLE_DFM, scheme = "prop"))
VARIABLE_CLUSTER = hclust(as.dist(VARIABLE_DIST), method = "X")
VARIABLE_CLUSTER$labels = docnames(VARIABLE_DFM)
plot(VARIABLE_CLUSTER, xlab = "xlab", sub = "sub", main = "Euclidean Distance on
Normalized Token Frequency", labels = FALSE)
rect.hclust(VARIABLE_CLUSTER, k = X, border="red")
VARIABLE_CLUSTER_OUTPUT = cutree(VARIABLE_CLUSTER, k = X)
```

Spherical K-Means Clusters

```
library(skmeans)
set.seed(161)
```

```
VARIABLE_CLUSTER = skmeans(VARIABLE_DFM, k = number_of_clusters, method =
'METHOD', control = list(verbose = TRUE, nruns=10))
```

K-Means Clusters

```
set.seed(161)
kmeans(VARIABLE, k, iter.max = 10, nstart = 1, algorithm = c("METHOD"), trace=FALSE)
```

LDA Topic Models

```
library(topicmodels)
VARIABLE_LDA = LDA(VARIABLE_READ, k = NUMBER_OF_TOPICS, method = "METHOD")
terms(VARIABLE_LDA, 10)
topics(VARIABLE_LDA)
VARIABLE_LDA@gamma
```

Visualize LDA with LDAvis

```
library(readtext)
library(quanteda)
library(topicmodels)
library(LDAvis)
VARIABLE_LDA = LDA(VARIABLE_DFM, k = 20, method = "Gibbs")
phi <- posterior(VARIABLE_LDA)$terms %>% as.matrix
theta <- posterior(VARIABLE_LDA)$topics %>% as.matrix
vocab <- colnames(phi)
doc_length <- ntoken(VARIABLE_DFM[rownames(VARIABLE_DFM)])
temp_frequency <- as.matrix(VARIABLE_DFM)
freq_matrix <- data.frame(ST = colnames(temp_frequency), Freq = colSums(temp_frequency))
rm(temp_frequency)
json_lda <- LDAvis::createJSON(phi = phi, theta = theta, vocab = vocab, doc.length =
doc_length, term.frequency = freq_matrix$Freq)
serVis(json_lda, out.dir = "OUTPUT", open.browser = TRUE)
```

Build STM (from DFM)

```
library(stm)
VARIABLE_STM = stm(VARIABLE_READ, K = NUMBER_OF_TOPICS, init.type =
c("METHOD"), seed = 161)
labelTopics(VARIABLE_STM_MODEL, n = 10)
apply(VARIABLE_STM_MODEL$theta, MARGIN=1, FUN=which.max)
```

Compensation Tiers' Keyness Measures

```
VARIABLE_READ = readtext("./FILE.csv", text_field="text", docid_field = "doc_id", header =
TRUE, encoding = "UTF-8")
VARIABLE_DFM = dfm(VARIABLE_READ$text, remove = stopwords("english"), stem = TRUE,
ngrams = 1, remove_punct = TRUE, remove_numbers = FALSE, groups =
VARIABLE_READ$compensation)
```

```
GROUP1_KEYNESS = textstat_keyness(VARIABLE_DFM, target = "GROUPX", measure =
c("chi2"))
textplot_keyness(GROUPX_KEYNESS)
```

Compensation Tiers' TFIDF Scores

```
cluster1 = subset(VARIABLE_READ, COMP.LEVEL == "top_20")
cluster2 = subset(VARIABLE_READ, COMP.LEVEL == "middle_60")
cluster3 = subset(VARIABLE_READ, COMP.LEVEL == "bottom_20")
cluster4 = subset(VARIABLE_READ, COMP.LEVEL == "penny")
cluster5 = subset(VARIABLE_READ, COMP.LEVEL == "zero")
corpus1 = corpus(cluster1)
corpus2 = corpus(cluster2)
corpus3 = corpus(cluster3)
corpus4 = corpus(cluster4)
corpus5 = corpus(cluster5)
dfm1 = dfm(corpus1, remove = stopwords("english"), stem = TRUE, ngrams =
NUMBER_OF_NGRAMS, remove_punct = TRUE, remove_numbers = FALSE)
dfm2 = dfm(corpus2, remove = stopwords("english"), stem = TRUE, ngrams =
NUMBER_OF_NGRAMS, remove_punct = TRUE, remove_numbers = FALSE)
dfm3 = dfm(corpus3, remove = stopwords("english"), stem = TRUE, ngrams =
NUMBER_OF_NGRAMS, remove_punct = TRUE, remove_numbers = FALSE)
dfm4 = dfm(corpus4, remove = stopwords("english"), stem = TRUE, ngrams =
NUMBER_OF_NGRAMS, remove_punct = TRUE, remove_numbers = FALSE)
dfm5 = dfm(corpus5, remove = stopwords("english"), stem = TRUE, ngrams =
NUMBER_OF_NGRAMS, remove_punct = TRUE, remove_numbers = FALSE)
tfidf1 = dfm_tfidf(dfm1)
tfidf2 = dfm_tfidf(dfm2)
tfidf3 = dfm_tfidf(dfm3)
tfidf4 = dfm_tfidf(dfm4)
tfidf5 = dfm_tfidf(dfm5)
tfidf_weighted_words1 = topfeatures(tfidf1, 50000)
tfidf_weighted_words2 = topfeatures(tfidf2, 50000)
tfidf_weighted_words3 = topfeatures(tfidf3, 50000)
tfidf_weighted_words4 = topfeatures(tfidf4, 50000)
tfidf_weighted_words5 = topfeatures(tfidf5, 50000)
write.csv(tfidf_weighted_words1, file = "top_20_trigram_expanded_tfidf.csv")
write.csv(tfidf_weighted_words2, file = "middle_60_trigram_expanded_tfidf.csv")
write.csv(tfidf_weighted_words3, file = "bottom_20_trigram_expanded_tfidf.csv")
write.csv(tfidf_weighted_words4, file = "penny_trigram_expanded_tfidf.csv")
write.csv(tfidf_weighted_words5, file = "zero_trigram_expanded_tfidf.csv")
```

Unweighted Similarity Measures

```
unweighted_simil_select = textstat_simil(unweighted_dfm, unweighted_dfm[, c("readable",
"transcribe", "transcribed", "transcript", "transcripts", "extract", "extraction", "describe",
```

"describes", "description", "descriptions", "summary", "summaries", "summarize", "summarisation", "rewrite")], method = "cosine", margin = "features")

Weighted Similarity Measures

```
weighted_simil_select = textstat_simil(weighted_dfm, weighted_dfm[, c("readable", "transcribe", "transcribed", "transcript", "transcripts", "extract", "extraction", "describe", "describes", "description", "descriptions", "summary", "summaries", "summarize", "summarisation", "rewrite")], method = "cosine", margin = "features")
```

Unweighted TidyText Word Vector Synonyms

```
library(tidyverse)
library(tidytext)
library(widyr)
library(readtext)
VARIABLE_READ = readtext("./FILE.csv", text_field="text", docid_field = "doc_id", header = TRUE, encoding = "UTF-8")
VARIABLE_CLEANED = VARIABLE_READ %>% mutate(postID = row_number())
slide_windows <- function(tbl, doc_var, window_size) {
  each_total <- tbl %>%
    group_by(!!doc_var) %>%
    mutate(doc_total = n(),
           each_total = pmin(doc_total, window_size, na.rm = TRUE)) %>%
    pull(each_total)
  rle_each <- rle(each_total)
  counts <- rle_each[["lengths"]]
  counts[rle_each$values != window_size] <- 1
  id_counts <- rep(rle_each$values, counts)
  window_id <- rep(seq_along(id_counts), id_counts)
  indexer <- (seq_along(rle_each[["values"]]) - 1) %>%
    map2(rle_each[["values"]] - 1,
         ~ seq.int(.x, .x + .y)) %>%
    map2(counts, ~ rep(.x, .y)) %>%
    flatten_int() +
    window_id
  tbl[indexer, ] %>%
    bind_cols(data_frame(window_id)) %>%
    group_by(window_id) %>%
    filter(n_distinct(!!doc_var) == 1) %>%
    ungroup
}
tidy_pmi <- VARIABLE_CLEANED %>%
  unnest_tokens(word, text) %>%
        anti_join(stop_words, by = c("word" = "word")) %>%
  add_count(word) %>%
```

```
   filter(n >= 1) %>%
   select(-n) %>%
   slide_windows(quo(postID), 8) %>%
   pairwise_pmi(word, window_id)
tidy_word_vectors <- tidy_pmi %>%
   widely_svd(item1, item2, pmi, nv = 1)
nearest_synonyms <- function(df, token) {
   df %>%
      widely(~ . %*% (.[token, ]), sort = TRUE)(item1, dimension, value) %>%
      select(-item2)
}
output = tidy_word_vectors %>%
   nearest_synonyms("TERM")
```

Weighted TidyText Word Vector Synonyms

```
library(tidyverse)
library(tidytext)
library(widyr)
library(readtext)
VARIABLE_READ = readtext("./FILE.csv", text_field="text", docid_field = "doc_id", header =
TRUE, encoding = "UTF-8")
VARIABLE_EXPANDED = VARIABLE_READ[rep(seq.int(1,nrow(VARIABLE_READ)),
VARIABLE_READ$AVAILABLE.HITS), ]
VARIABLE_CLEANED = VARIABLE_EXPANDED %>% mutate(postID = row_number())
slide_windows <- function(tbl, doc_var, window_size) {
   each_total <- tbl %>%
      group_by(!!doc_var) %>%
      mutate(doc_total = n(),
            each_total = pmin(doc_total, window_size, na.rm = TRUE)) %>%
      pull(each_total)
   rle_each <- rle(each_total)
   counts <- rle_each[["lengths"]]
   counts[rle_each$values != window_size] <- 1
   id_counts <- rep(rle_each$values, counts)
   window_id <- rep(seq_along(id_counts), id_counts)
   indexer <- (seq_along(rle_each[["values"]]) - 1) %>%
      map2(rle_each[["values"]] - 1,
          ~ seq.int(.x, .x + .y)) %>%
      map2(counts, ~ rep(.x, .y)) %>%
      flatten_int() +
      window_id
   tbl[indexer, ] %>%
      bind_cols(data_frame(window_id)) %>%
      group_by(window_id) %>%
```

```
        filter(n_distinct(!!doc_var) == 1) %>%
        ungroup
}
tidy_pmi <- VARIABLE_CLEANED %>%
    unnest_tokens(word, text) %>%
          anti_join(stop_words, by = c("word" = "word")) %>%
    add_count(word) %>%
    filter(n >= 1) %>%
    select(-n) %>%
    slide_windows(quo(postID), 8) %>%
    pairwise_pmi(word, window_id)
tidy_word_vectors <- tidy_pmi %>%
    widely_svd(item1, item2, pmi, nv = 1)
nearest_synonyms <- function(df, token) {
    df %>%
        widely(~ . %*% (.[token, ]), sort = TRUE)(item1, dimension, value) %>%
        select(-item2)
}
output = tidy_word_vectors %>%
    nearest_synonyms("TERM")
```