Assignment 5

Colleen McCamy

2023-05-18

```r
# reading in the data
nuclear_df <- read_csv("/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-s
```

Train Your Own Embeddings

```
## Rows: 100 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Article
## dbl (1): ID
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# calculating the unigram probabilities
unigram_probs <- nuclear_df |>
  unnest_tokens(word, Article) |>
  anti_join(stop_words, by = 'word') |>
  count(word, sort = 1) |> # creates an n column
  mutate(p = n / sum(n)) # probability of word
```

```r
# Define a custom function to remove numbers
remove_numbers <- function(text) {
  gsub("\\b\\d+\\b", "", text)
}

skipgrams <- nuclear_df |>
  mutate(Article = remove_numbers(Article)) |>
  unnest_tokens(ngram, Article,
                token = "ngrams", # new mode for tokens
                n = 5) |>
  mutate(ngramID = row_number()) |>
  tidyr::unite(skipgramID, ID, ngramID) |> # paste strings together in columns
  unnest_tokens(word, ngram) |> # unnest five word sequences to word level
  anti_join(stop_words, by = "word")
```

```r
#calculating probabilities with a pairwise count
skipgram_probs <- skipgrams |>
  pairwise_count(word, skipgramID, diag = T, sort = T) |>
  mutate(p = n/sum(n))
```

```
# normalizing and filtering the top to reduce the data if it is occurring less than 20 times (since they
normalized_prob <- skipgram_probs |>
  filter(n > 20 ) |>
  rename(word1 = item1,
         word2 = item2) |>
  left_join(unigram_probs |>
              select(word1 = word,
                     p1 = p),
            by = "word1") |>  # joining probabilities
   left_join(unigram_probs |>
              select(word2 = word,
                     p2 = p),
            by = "word2") |>
  mutate(p_together = p/p1/p2)

normalized_prob[5000:5010,]
```

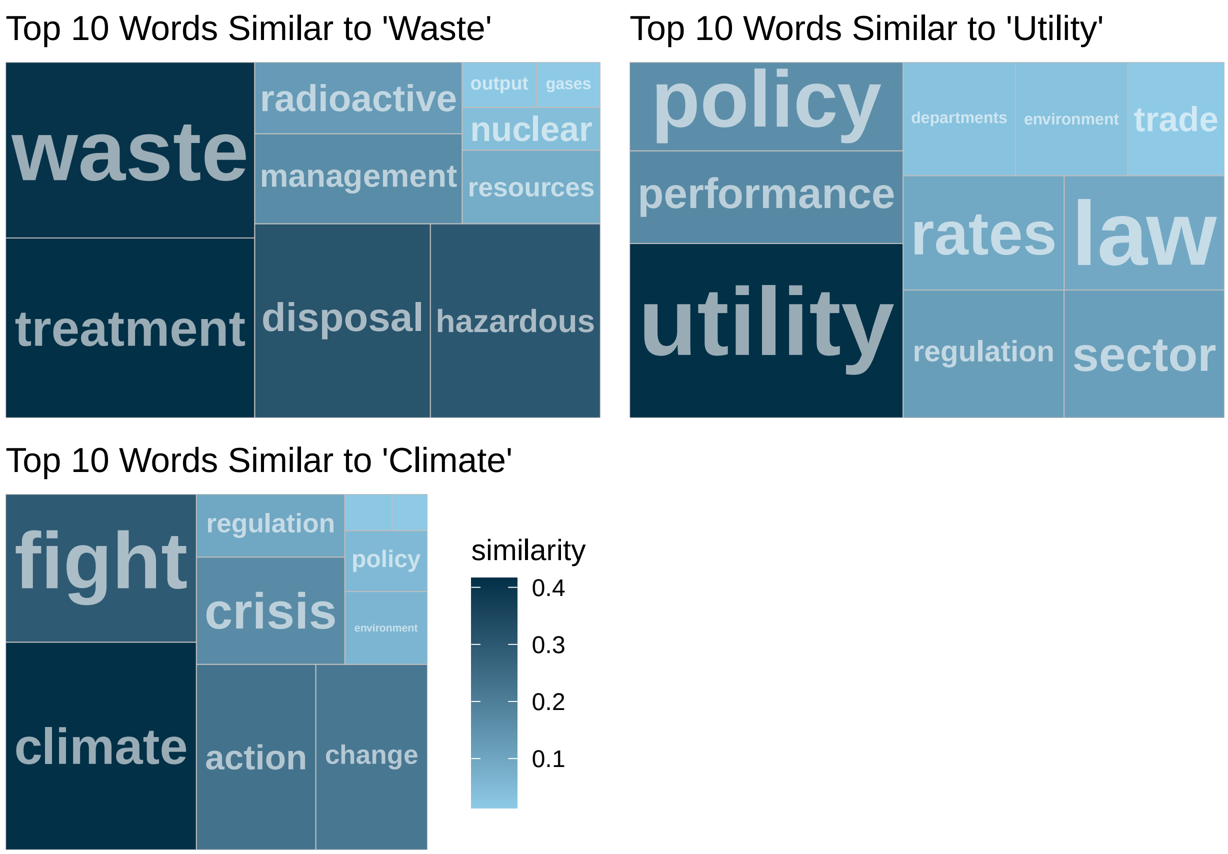
```
# calculating word similarities by location in the n-dimension space
pmi_matrix <- normalized_prob |>
  mutate(pmi = log10(p_together)) |> # log of probability to normalized
  cast_sparse(word1, word2, pmi) # converting from tidyformat to a sparse matrix
```

```
# setting NAs to zero
pmi_matrix@x[is.na(pmi_matrix@x)] <- 0

# creating one matrix with ID values and two orthogonal matrices
pmi_svd <- irlba::irlba(pmi_matrix,
                        100,
                        maxit = 500 # decides how to adjust estimates of singular values
                        )

# pulling the 'u' matrix with singular values from the orthogonal vectors to create the space to calcula
word_vectors <- pmi_svd$u

# combining the word paired with the probabilities
rownames(word_vectors) <- rownames(pmi_matrix)
```

PART 2: Calculate and plot the 10 most semantically similar words for three key words

```
# creating a search synonyms function
search_synonyms <- function(word_vectors, selected_vector) {
dat <- word_vectors %*% selected_vector

similarities <- dat |>
        tibble(token = rownames(dat), similarity = dat[,1])
similarities |>
       arrange(-similarity) |>
        select(c(2,3))
}

# calculating similar synonyms
utility <- search_synonyms(word_vectors, word_vectors["utility", ]) |> head(n = 10)
climate <- search_synonyms(word_vectors, word_vectors["climate", ]) |> head(n = 10)
```

```r
waste <- search_synonyms(word_vectors, word_vectors["waste", ]) |> head(n = 10)

# Creating treemaps
waste_plot <- ggplot(waste,
                     aes(area = similarity,
                         fill = similarity,
                         label = token)) +
  geom_treemap() +
  geom_treemap_text(place = "centre",
                    grow = TRUE,
                    alpha = 0.6,
                    color = "white",
                    fontface = "bold") +
  scale_fill_gradient(high = "#023047", low = "#8ecae6") +
  theme_minimal() +
  labs(title = "Top 10 Words Similar to 'Waste'")+
  theme(legend.position = "none")

climate_plot <- ggplot(climate, aes(area = similarity, fill = similarity, label = token)) +
  geom_treemap() +
  geom_treemap_text(place = "centre",
                    grow = TRUE,
                    alpha = 0.6,
                    color = "white",
                    fontface = "bold") +
  scale_fill_gradient(high = "#023047", low = "#8ecae6") +
  theme_minimal() +
  labs(title = "Top 10 Words Similar to 'Climate'")


utility_plot <- ggplot(utility, aes(area = similarity, fill = similarity, label = token)) +
  geom_treemap() +
  geom_treemap_text(place = "centre",
                    grow = TRUE,
                    alpha = 0.6,
                    color = "white",
                    fontface = "bold") +
  scale_fill_gradient(high = "#023047", low = "#8ecae6") +
  theme_minimal() +
  labs(title = "Top 10 Words Similar to 'Utility'")+
  theme(legend.position = "none")


ggpubr::ggarrange(waste_plot, utility_plot, climate_plot, ncol = 2, nrow = 2)
```



PART 3: Assembling word math equations

```r
# waste with hazardous
waste_hazard <- word_vectors["waste", ] + word_vectors["hazardous", ]
search_synonyms(word_vectors, waste_hazard) |> head(n = 10) |> gt::gt()
```

token	similarity
treatment	0.51912731
waste	0.47285236
disposal	0.39370662
hazardous	0.37913576
radioactive	0.14944572
management	0.14519861
nuclear	0.08102873
resources	0.03749219
civil	0.02140820
medicine	0.02032920

```r
# waste without hazardous
waste_nohazard <- word_vectors["waste", ] - word_vectors["hazardous", ]
search_synonyms(word_vectors, waste_nohazard)  |> head(n = 10) |> gt::gt()
```

token	similarity
resources	0.08764769
management	0.08465243
waste	0.06640598
human	0.03447593
environment	0.03395193
radioactive	0.03318820
treatment	0.03283364
hazardous	0.02731063
disposal	0.02627796
output	0.02575198

```
# seeing government regulation
gov_reg <- word_vectors["regulation", ] + word_vectors["government", ]
search_synonyms(word_vectors, gov_reg)  |> head(n = 10) |> gt::gt()
```

token	similarity
government	0.2813685
policy	0.2533957
regulation	0.2523979
advisors	0.2493851
ministers	0.1830578
departments	0.1514234
public	0.1513646
administration	0.1309293
utility	0.1305547
wind	0.1241190

Pretrained Embeddings PART 4: Create a set of 100-dimensional GloVe word embeddings

```
#glove6b <- embedding_glove6b(dimensions = 100,
#                              options(timeout = 250))

#write.csv(glove6b, "/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-sent

glove6b <- read_csv("/Users/colleenmccamy/Documents/MEDS/classes/spring/eds-231-text-analysis/text-sent
```

```
## New names:
## Rows: 400000 Columns: 102
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): token dbl (101): ...1, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12,
## d13, d14...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
glove6b <- glove6b[ , -1]
```

```r
# Assuming your dataframe is called 'df'
selected_columns <- glove6b |>
  select(-token)  # Exclude the first column

matrix_data <- as.matrix(selected_columns)  # Exclude the first column and convert to a matrix

row_names <- glove6b$token  # Extract values from the first column

rownames(matrix_data) <- row_names
```

PART 5: Test the cannonical word math equation on the GloVe embeddings: "berlin" - "germany" + "france" = ?

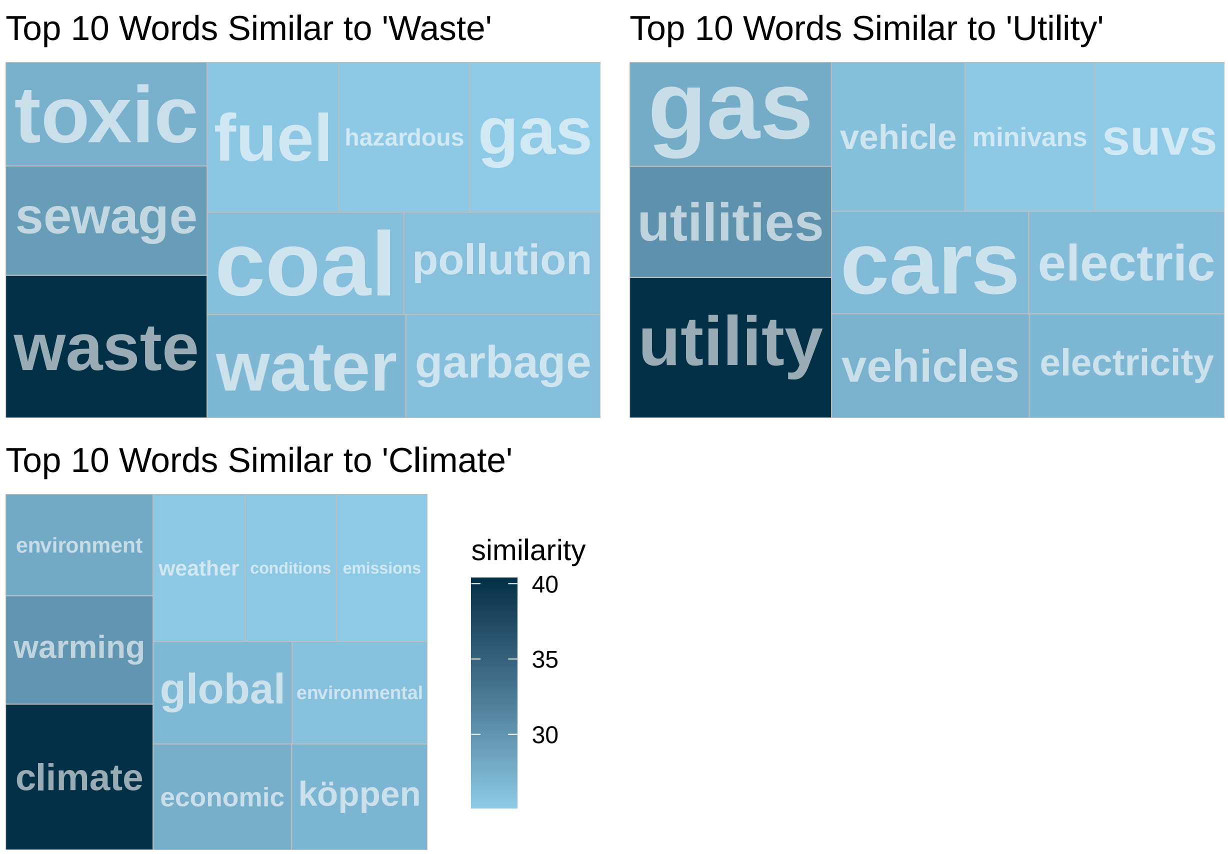
```r
# word math equation
countries <- matrix_data["berlin", ] - matrix_data["germany", ] + matrix_data["france", ]

search_synonyms(matrix_data, countries) |> head(n = 10)
```

```
## # A tibble: 10 x 2
##    token    similarity
##    <chr>         <dbl>
##  1 paris          34.4
##  2 france         31.5
##  3 french         28.0
##  4 de             28.0
##  5 le             26.6
##  6 london         25.4
##  7 la             24.3
##  8 brussels       23.3
##  9 berlin         23.3
## 10 lyon           22.2
```

PART 6: Recreate parts 2 and 3 above using the the GloVe embeddings in place of the ones you trained.

```r
# calculating similar synonyms
utility_glove <- search_synonyms(matrix_data,
                                 matrix_data["utility", ]) |>
  head(n = 10)

climate_glove <- search_synonyms(matrix_data,
                                 matrix_data["climate", ]) |>
  head(n = 10)

waste_glove <- search_synonyms(matrix_data,
                               matrix_data["waste", ]) |>
  head(n = 10)

# Creating treemaps
waste_plot_glove <- ggplot(waste_glove, aes(area = similarity, fill = similarity, label = token)) +
  geom_treemap() +
  geom_treemap_text(place = "centre",
                    grow = TRUE,
```

```
                    alpha = 0.6,
                    color = "white",
                    fontface = "bold") +
  scale_fill_gradient(high = "#023047", low = "#8ecae6") +
  theme_minimal() +
  labs(title = "Top 10 Words Similar to 'Waste'")+
  theme(legend.position = "none")

climate_plot_glove <- ggplot(climate_glove, aes(area = similarity, fill = similarity, label = token)) +
  geom_treemap() +
  geom_treemap_text(place = "centre",
                    grow = TRUE,
                    alpha = 0.6,
                    color = "white",
                    fontface = "bold") +
  scale_fill_gradient(high = "#023047", low = "#8ecae6") +
  theme_minimal() +
  labs(title = "Top 10 Words Similar to 'Climate'")


utility_plot_glove <- ggplot(utility_glove, aes(area = similarity, fill = similarity, label = token)) +
  geom_treemap() +
  geom_treemap_text(place = "centre",
                    grow = TRUE,
                    alpha = 0.6,
                    color = "white",
                    fontface = "bold") +
  scale_fill_gradient(high = "#023047", low = "#8ecae6") +
  theme_minimal() +
  labs(title = "Top 10 Words Similar to 'Utility'")+
  theme(legend.position = "none")


ggpubr::ggarrange(waste_plot_glove, utility_plot_glove, climate_plot_glove, ncol = 2, nrow = 2)
```



RESPONSE: How do they compare? These words are a lot more general and include additional topics. For instance when looking at waste it includes sewage as a top word when this doesn't apply as much in a nuclear context. It is also interesting to see utility as within the context as it appears to include phrases such as a utility vehicle. Overall, I think this helps to showcase the importance of understanding the corpus and context in which you are doing analyzing word sentiment as it will only be in the context of the data (or corpus).

What are the implications for applications of these embeddings? In looking at the Glove data, it is important to acknowledge the implications of doing this analysis on the nuclear corpus from the articles. This text analysis will only carry and maybe even amplify biases present within the corpus. From looking at some of the sources, it appears that the nuclear corpus has a lot of government and technical articles. The sentiment analysis will only look at similarities within the context of the articles. This could leave out public opinion, communities and voices under represented within the corpus of text.

```
# waste with hazardous
waste_hazard_glove <- matrix_data["waste", ] + matrix_data["hazardous", ]
search_synonyms(matrix_data, waste_hazard_glove) |> head(n = 10) |> gt::gt()
```

token	similarity
waste	59.03691
hazardous	57.83832
toxic	51.69843
radioactive	47.31472
pollution	47.18033
sewage	44.55662

chemicals	44.25145
gases	42.31807
polluted	42.20145
wastes	42.08571

```
# waste without hazardous
waste_nohazard_glove <- matrix_data["waste", ] - matrix_data["hazardous", ]
search_synonyms(matrix_data, waste_nohazard_glove)  |> head(n = 10) |> gt::gt()
```

token	similarity
billion	12.66372
pot	12.42347
chicken	11.39519
million	11.25154
waste	11.13298
bread	10.52063
coffers	10.50605
dollars	10.37823
cash	10.20472
power	10.16914

```
# seeing government regulation
gov_reg_glove <- matrix_data["regulation", ] + matrix_data["government", ]
search_synonyms(matrix_data, gov_reg_glove)  |> head(n = 10) |> gt::gt()
```

token	similarity
government	57.06907
federal	49.26666
regulation	46.92997
laws	46.59144
regulations	46.46560
state	46.14397
law	45.87510
security	45.32624
economic	45.19340
administration	44.86828