Chapter 8: Analyzing the Language of Climate Change in the United States Congressional Record

The output is a data frame with the top words for each keyword ("climate", "woman", "environmentalist", and "government")

```
library(tmha.data)
```

we will take just the `content` column

```
library(text2vec)
library(tidyverse)
library(kableExtra)

data("congress_daily_climate_change_2000_2020")
```

Create new variables that filter the `congress_daily_climate_change_2000_2020` data for the decade and take just the `content` column.

```
congress_daily_climate_change_2000 <- congress_daily_climate_change_2000_2020 %>%
  filter(decade == "2000")

climate_change_text_2000 <- congress_daily_climate_change_2000$content

congress_daily_climate_change_2010 <- congress_daily_climate_change_2000_2020 %>%
  filter(decade == "2010")

climate_change_text_2010 <- congress_daily_climate_change_2000$content

congress_daily_climate_change_2020 <- congress_daily_climate_change_2000_2020 %>%
  filter(decade == "2020")

climate_change_text_2020 <- congress_daily_climate_change_2020$content
```

The parameters control the behavior of the GloVe word embedding model. The rank (50) sets the dimensionality of the learned word vectors — this is the number of numeric values used to represent a word. Higher dimensionality allows the model to capture more nuance, subtle meaning, and fine-grained semantic distinctions between words, while smaller dimensionality produces simpler models that run faster but encode less detail. The window (5) determines how many neighboring words on each side are considered when learning meaning, based on the idea that words appearing near each other tend to be related. The term_count_min (1) filters out words that appear less than a minimum number of times in the corpus, preventing extremely rare tokens or typos from influencing the model. The x_max (10) parameter downweights extremely frequent filler words so they do not dominate the training process. The n_iter (30) controls how many passes the model makes over the co-occurrence data — more iterations usually improve results but take longer. Finally, the seed (42) ensures reproducibility so the same settings and data produce the same model when rerun.

We do not go into detail about some of these concepts – ENTER – here.

```r
find_word_embeddings <- function(data,
                                 rank=50,
                                 window=5L,
                                 term_count_min=1,
                                 x_max=10,
                                 n_iter=30,
                                 seed=42) {

  tokens <- word_tokenizer(str_to_lower(data))
  it <- itoken(tokens, ids = seq_along(tokens), progressbar = FALSE)

  vocab <- create_vocabulary(it) %>%
    prune_vocabulary(term_count_min = term_count_min)

  vectorizer <- vocab_vectorizer(vocab)
  tcm <- create_tcm(it, vectorizer, skip_grams_window = window)

  set.seed(seed)
  glove <- GlobalVectors$new(rank = rank, x_max = x_max)
  wv_main <- glove$fit_transform(tcm, n_iter = n_iter)
  wv_context <- glove$components
  word_embeddings <- wv_main + t(wv_context) }
```

While the model is training, GloVe prints progress updates so you can see how training is advancing. The progress updates include the current timestamp, which training epoch the model is on, and the current loss value. An epoch means one full pass over the entire dataset — so if the console says epoch 10, the model has now processed the corpus 10 times. The loss value tells you how well the model is learning at that point — lower loss generally means the model is improving. So during training, you will see output messages formatted like `INFO [<timestamp>] epoch <k>, loss <value>` appear in the console to show the model's progress step-by-step.

```r
word_embeddings_2000 <- find_word_embeddings(climate_change_text_2000)
word_embeddings_2010 <- find_word_embeddings(climate_change_text_2010)
word_embeddings_2020 <- find_word_embeddings(climate_change_text_2020)
```

Finding Words with the Greatest Association

We can now write a function

```r
get_similar <- function(keyword, word_embeddings, top_n = top_n_default) {

  kw_vec <- word_embeddings[keyword, , drop = FALSE]
  cos_sim <- sim2(word_embeddings, kw_vec, method = "cosine", norm = "l2")[, 1]

  tibble(word = names(cos_sim), similarity = unname(cos_sim)) %>%
    filter(word != keyword) %>%
    arrange(desc(similarity)) %>%
    slice_head(n = top_n) }
```

```r
most_similar_climate <- get_similar("climate", word_embeddings_2000, 10)
most_similar_emissions <- get_similar("emissions", word_embeddings_2000, 10)
most_similar_coal <- get_similar("coal", word_embeddings_2000, 20)
```

Table 8.1: Words Most Related to "Climate": 2000s Congressional Records

Word	Cosine similarity
change	0.985
global	0.895
warming	0.847
address	0.736
impacts	0.719
addressing	0.705
issue	0.696
problem	0.689
international	0.667
this	0.663

```
tbl_2000 <- most_similar_climate %>%
  mutate(similarity = round(similarity, 3)) %>%
  kable(format = "latex",
    caption = 'Words Most Related to "Climate": 2000s Congressional Records',
        col.names = c("Word", "Cosine similarity"),
        align = c("l", "r"),
    booktabs = TRUE) %>%
  kable_styling(full_width = FALSE)

tbl_2000
```

```
tbl_2010 <- most_similar_emissions %>%
  mutate(similarity = round(similarity, 3)) %>%
  kable(format = "latex", caption = 'Words Most Related to "Emissions": 2010s Congressional Records',
    col.names = c("Word", "Cosine similarity"),
    align = c("l", "r"),
    booktabs = TRUE) %>%
  kable_styling(full_width = FALSE)

tbl_2010
```

```
table_2020 <- most_similar_coal %>%
  mutate(similarity = round(similarity, 3)) %>%
  kable(format = "latex",
        caption = 'Words Most Related to "Coal": 2020s Congressional Records',
        col.names = c("Word", "Cosine similarity"),
        align = c("l", "r"),
        booktabs = TRUE) %>%
  kable_styling(full_width = FALSE,
                latex_options = c("hold_position"))

table_2020
```

Table 8.2: Words Most Related to "Emissions": 2010s Congressional Records

Word	Cosine similarity
greenhouse	0.911
carbon	0.876
dioxide	0.850
reduce	0.833
gases	0.805
gas	0.775
reductions	0.773
reduction	0.751
reducing	0.744
co2	0.744

Table 8.3: Words Most Related to "Coal": 2020s Congressional Records

Word	Cosine similarity
plants	0.831
power	0.804
using	0.745
generation	0.737
burning	0.725
nuclear	0.717
fired	0.715
sources	0.710
use	0.692
produce	0.685
powerplants	0.683
generating	0.663
fossil	0.657
fuels	0.654
electricity	0.645
natural	0.644
technologies	0.639
plant	0.639
producing	0.635
wind	0.632

```r
library(ggrepel)

# take the top N words returned by most_similar_coal
words <- most_similar_coal$word

# include the anchor term too
words <- c("coal", words)

# subset embedding matrix to just those words
emb_small <- word_embeddings_2000[words, ]
```

```r
# reduce high dimensions to 2D PCA for visualization purposes
pca <- prcomp(emb_small, center = TRUE, scale. = TRUE)
coords <- as_tibble(pca$x[,1:2]) %>%
  mutate(word = words)

ggplot(coords, aes(PC1, PC2, label = word)) +
  geom_point(size = 3) +
  geom_text_repel() +
  theme_minimal() +
  ggtitle("Words close to COAL in embedding space (2D PCA)")
```

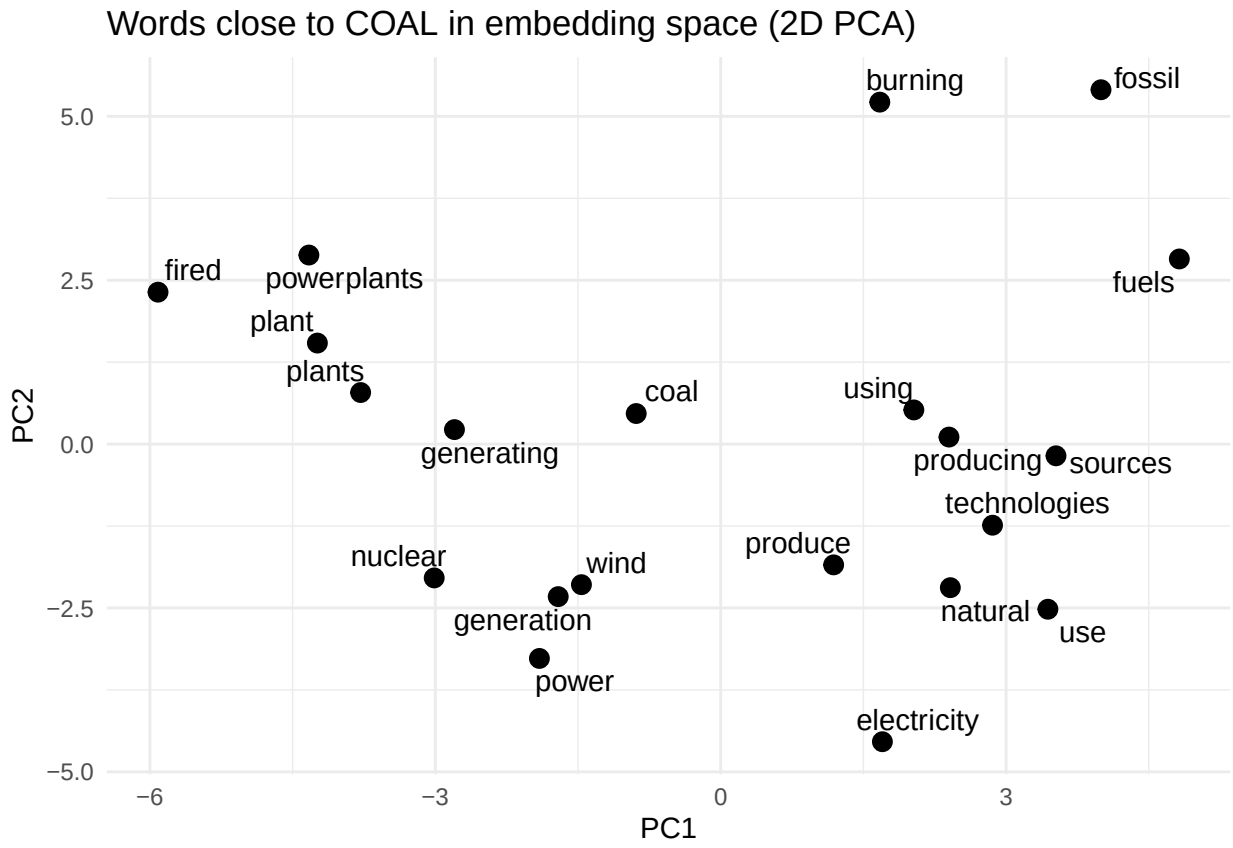

Figure 1: Two-dimensional projection of word embeddings showing the terms most semantically similar to coal in the Climate Change Congressional corpus.

Vector Subtraction

Vector subtraction in word embeddings asks: what part of the meaning belongs to word1 but not word2? Because embeddings place words inside a multidimensional geometric "meaning space," where similar words live closer together, subtracting one vector from another isolates the semantic direction unique to the first term. Vector subtraction lets us reason about conceptual differences directly in mathematical space — extracting the portion of meaning that distinguishes one concept from another.

```r
vector_subtraction <- function(word1, word2, embeddings, top_n = 10) {

  # keep 1×d matrices so sim2() gets matrices on both sides
  a <- embeddings[word1, , drop = FALSE]
  b <- embeddings[word2, , drop = FALSE]
  target <- a - b

  sims <- sim2(embeddings, target, method = "cosine", norm = "l2")[, 1]

  tibble(word = names(sims), similarity = unname(sims)) %>%
    filter(!word %in% c(word1, word2)) %>%
    arrange(desc(similarity)) %>%
    slice_head(n = top_n) }
```

```r
vector_subtraction("renewable", "fossil", word_embeddings_2000)
```

```
## # A tibble: 10 x 2
##    word          similarity
##    <chr>              <dbl>
##  1 portfolio          0.604
##  2 creation           0.561
##  3 80,000             0.531
##  4 establishing       0.521
##  5 opportunities      0.516
##  6 investment         0.493
##  7 provides           0.490
##  8 restoration        0.485
##  9 incentives         0.480
## 10 unprepared         0.477
```

If you get an error like `Error in embeddings[word2, , drop = FALSE] : subscript out of bounds` then that means the word was not in the dataset.

```r
vector_subtraction("renewable", "fossil", word_embeddings_2010)
```

```
## # A tibble: 10 x 2
##    word          similarity
##    <chr>              <dbl>
##  1 portfolio          0.573
##  2 incentives         0.547
##  3 investment         0.539
##  4 creation           0.527
##  5 opportunities      0.513
##  6 provides           0.493
##  7 development        0.485
##  8 2008               0.479
##  9 program            0.467
## 10 includes           0.465
```

Vector Addition

```
vector_addition <- function(word1, word2, embeddings, top_n = 10) {
  a <- embeddings[word1, , drop = FALSE]
  b <- embeddings[word2, , drop = FALSE]

  target <- a + b

  sims <- sim2(embeddings, target, method = "cosine", norm = "l2")[,1]

  tibble(word = names(sims), similarity = unname(sims)) %>%
    filter(!word %in% c(word1, word2)) %>%
    arrange(desc(similarity)) %>%
    slice_head(n = top_n) }
```

If one of the words does not exist in the corpus, we will see the error `Error in FUN(left, right) : non-numeric argument to binary operator`.

```
vector_addition("man", "woman", congress_daily_climate_change_2000)
```

```
## Error in FUN(left, right): non-numeric argument to binary operator
```

Vectors Over Time

```
congress_daily_climate_change_enter <- congress_daily_climate_change_2000_2020 %>%
  mutate(date = ymd(date),
         year = year(date),
         period_5 = paste0(floor(year/5)*5, "-", floor(year/5)*5 + 4))

congress_daily_climate_change_enter <- congress_daily_climate_change_enter %>%
  mutate(period_5 = ifelse(year == 2020, "2020", period_5))
```

```
# Make 5-year periods (2000-2004, 2005-2009, 2010-2014, 2015-2019, 2020)
congress_5 <- congress_daily_climate_change_2000_2020 %>%
  mutate(date = ymd(date),
         year = year(date),
         period_5 = paste0(floor(year/5)*5, "-", floor(year/5)*5 + 4),
         period_5 = ifelse(year == 2020, "2020", period_5) )

# Split text by 5-year period
texts_by_period <- congress_5 %>%
  filter(!is.na(content), str_squish(content) != "") %>%
  split(.$period_5) %>%
  map(~ .x$content)

# Build embeddings per 5-year period
embeddings_by_period <- texts_by_period %>%
  imap(~ {if (length(.x) == 0) return(NULL)
    find_word_embeddings(.x)}) %>%
  discard(is.null)
```

```r
# Similarity trajectory over 5-year periods
similarity_trajectory <- function(word_a, word_b, embs = embeddings_by_period) {

  # order the x-axis by the numeric start year of each label
  order_levels <- names(embs) %>%
    as_tibble() %>%
    rename(lbl = value) %>%
    mutate(start = as.integer(str_replace(lbl, "-.*$", ""))) %>%
    arrange(start) %>%
    pull(lbl)

  tibble(period = factor(names(embs), levels = order_levels)) %>%
    mutate(similarity = map_dbl(as.character(period), \(p) {
      W <- embs[[p]]
      if (is.null(W))
        return(NA_real_)
      if (!all(c(word_a, word_b) %in% rownames(W)))
        return(NA_real_)
      as.numeric(sim2(W[word_a, , drop = FALSE],
                      W[word_b, , drop = FALSE],
                      method = "cosine", norm = "l2"))}))}

df_sim <- similarity_trajectory("coal", "emissions")

ggplot(df_sim, aes(period, similarity, group = 1)) +
  geom_line(linewidth = 1) +
  geom_point(size = 3) +
  theme_minimal() +
  labs(title = "Cosine similarity across 5-year periods",
       subtitle = "coal  ~  emissions",
       x = "5-year period",
       y = "Cosine similarity")
```

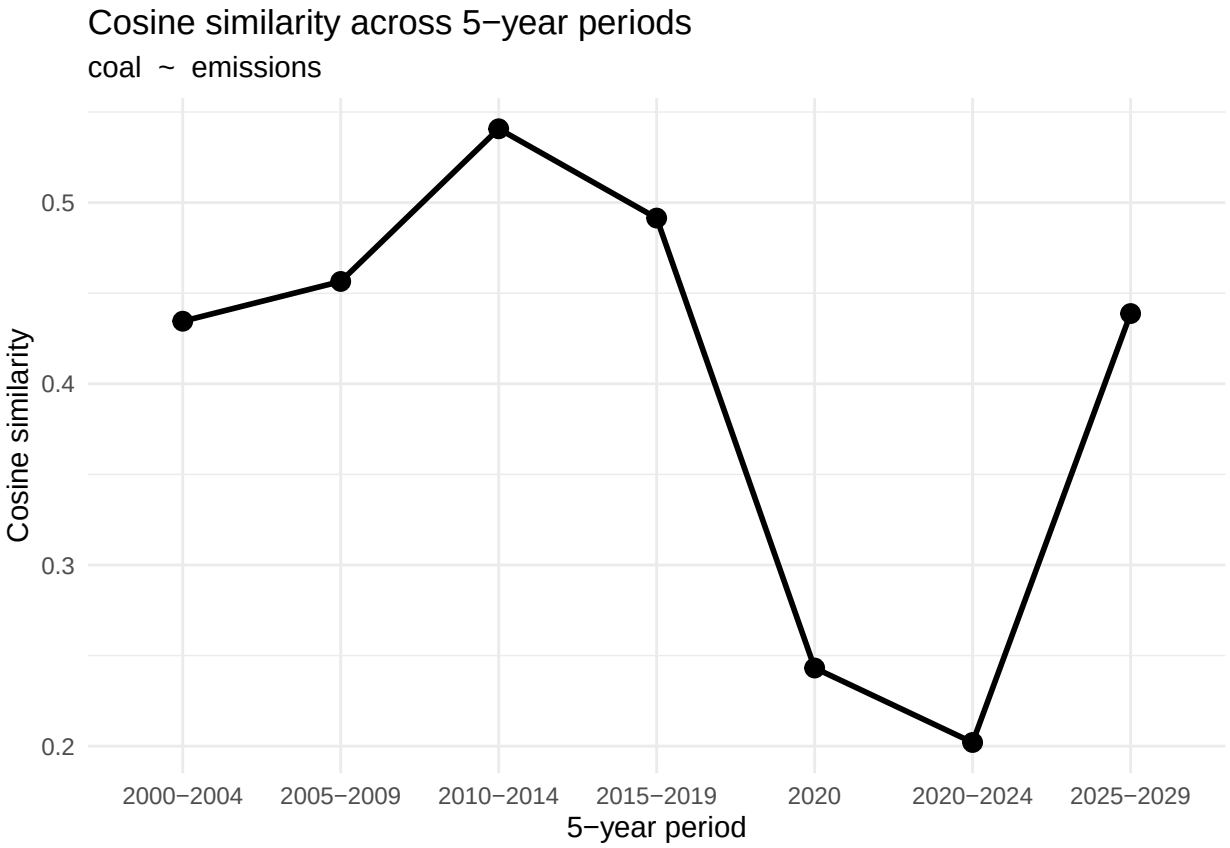

Figure 2: Cosine similarity between coal and emissions across five-year periods, showing how their semantic relationship shifts over time in the U.S. Congressional records mentioning climate change.