Topic 7: Word Embeddings

This week's Rmd file here: https://github.com/MaRo406/EDS_231-text-sentiment/blob/main/topic_7.Rmd

Today we are using climbing incident data from this repo: https://github.com/ecaroom/climbing-accidents. Some analysis (in Excel) on the data was written up into a Rock and Ice magazine article.

But I've constructed our data set (link below) by pulling a few key variables including the full text of each incident report.

```
incidents_df<-read_csv("https://raw.githubusercontent.com/MaRo406/EDS_231-text-sentiment/825b159b6da4c7
```

```
## Rows: 2770 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): ID, Accident Title, Text
## dbl (1): Publication Year
##
## 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.
```

First, let's calculate the unigram probabilities, how often we see each word in this corpus.

```
unigram_probs <- incidents_df %>%
    unnest_tokens(word, Text) %>%
    anti_join(stop_words, by = 'word') %>%
    count(word, sort = TRUE) %>%
    mutate(p = n / sum(n))
unigram_probs
```

```
## # A tibble: 25,205 x 3
##    word         n       p
##    <chr>    <int>   <dbl>
##  1 rope      5129 0.00922
##  2 feet      5101 0.00917
##  3 climbing  4755 0.00855
##  4 route     4357 0.00783
##  5 climbers  3611 0.00649
##  6 climb     3209 0.00577
##  7 fall      3168 0.00569
##  8 climber   2964 0.00533
##  9 rescue    2928 0.00526
## 10 source    2867 0.00515
## # ... with 25,195 more rows
```

Next, we need to know how often we find each word near each other word – the skipgram probabilities. This is where we use the sliding window.

```r
skipgrams <- incidents_df %>%
    unnest_tokens(ngram, Text, token = "ngrams", n = 5) %>%
    mutate(ngramID = row_number()) %>%
    tidyr::unite(skipgramID, ID, ngramID) %>%
    unnest_tokens(word, ngram) %>%
    anti_join(stop_words, by = 'word')

skipgrams
```

```
## # A tibble: 2,737,146 x 4
##    skipgramID `Accident Title`                            `Publication Y~` word
##    <chr>      <chr>                                                  <dbl> <chr>
##  1 1_1        Failure of Rappel Setup (Protection Pulled~             1990 colo~
##  2 1_1        Failure of Rappel Setup (Protection Pulled~             1990 rocky
##  3 1_1        Failure of Rappel Setup (Protection Pulled~             1990 moun~
##  4 1_1        Failure of Rappel Setup (Protection Pulled~             1990 nati~
##  5 1_1        Failure of Rappel Setup (Protection Pulled~             1990 park
##  6 1_2        Failure of Rappel Setup (Protection Pulled~             1990 rocky
##  7 1_2        Failure of Rappel Setup (Protection Pulled~             1990 moun~
##  8 1_2        Failure of Rappel Setup (Protection Pulled~             1990 nati~
##  9 1_2        Failure of Rappel Setup (Protection Pulled~             1990 park
## 10 1_3        Failure of Rappel Setup (Protection Pulled~             1990 moun~
## # ... with 2,737,136 more rows
```

```r
#calculate probabilities
skipgram_probs <- skipgrams %>%
    pairwise_count(word, skipgramID, diag = TRUE, sort = TRUE) %>%
    mutate(p = n / sum(n))
```

```
## Warning: `distinct_()` was deprecated in dplyr 0.7.0.
## Please use `distinct()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

Having all the skipgram windows lets us calculate how often words together occur within a window, relative to their total occurrences in the data. We do this using the point-wise mutual information (PMI). It's the logarithm of the probability of finding two words together, normalized for the probability of finding each of the words alone. PMI tells us which words occur together more often than expected based on how often they occurred on their own.

```r
#normalize probabilities
normalized_prob <- skipgram_probs %>%
    filter(n > 20) %>%
    rename(word1 = item1, word2 = item2) %>%
    left_join(unigram_probs %>%
                  select(word1 = word, p1 = p),
              by = "word1") %>%
    left_join(unigram_probs %>%
                  select(word2 = word, p2 = p),
              by = "word2") %>%
    mutate(p_together = p / p1 / p2)
```

```r
#Which words are most associated with "rope"?
normalized_prob %>%
    filter(word1 == "rope") %>%
    arrange(-p_together)
```

```
## # A tibble: 295 x 7
##    word1 word2        n          p      p1        p2 p_together
##    <chr> <chr>    <dbl>      <dbl>   <dbl>     <dbl>      <dbl>
##  1 rope  rope     25494 0.00340    0.00922 0.00922         40.0
##  2 rope  lengths    101 0.0000135  0.00922 0.0000575       25.4
##  3 rope  skinny      24 0.00000320 0.00922 0.0000144       24.2
##  4 rope  drag       211 0.0000281  0.00922 0.000138        22.1
##  5 rope  taut        98 0.0000131  0.00922 0.0000701       20.2
##  6 rope  coiled      60 0.00000800 0.00922 0.0000431       20.1
##  7 rope  thicker     21 0.00000280 0.00922 0.0000162       18.8
##  8 rope  trailing    68 0.00000907 0.00922 0.0000539       18.3
##  9 rope  fed         48 0.00000640 0.00922 0.0000413       16.8
## 10 rope  70m         31 0.00000414 0.00922 0.0000270       16.6
## # ... with 285 more rows
```

Now we convert to a matrix so we can use matrix factorization and reduce the dimensionality of the data.

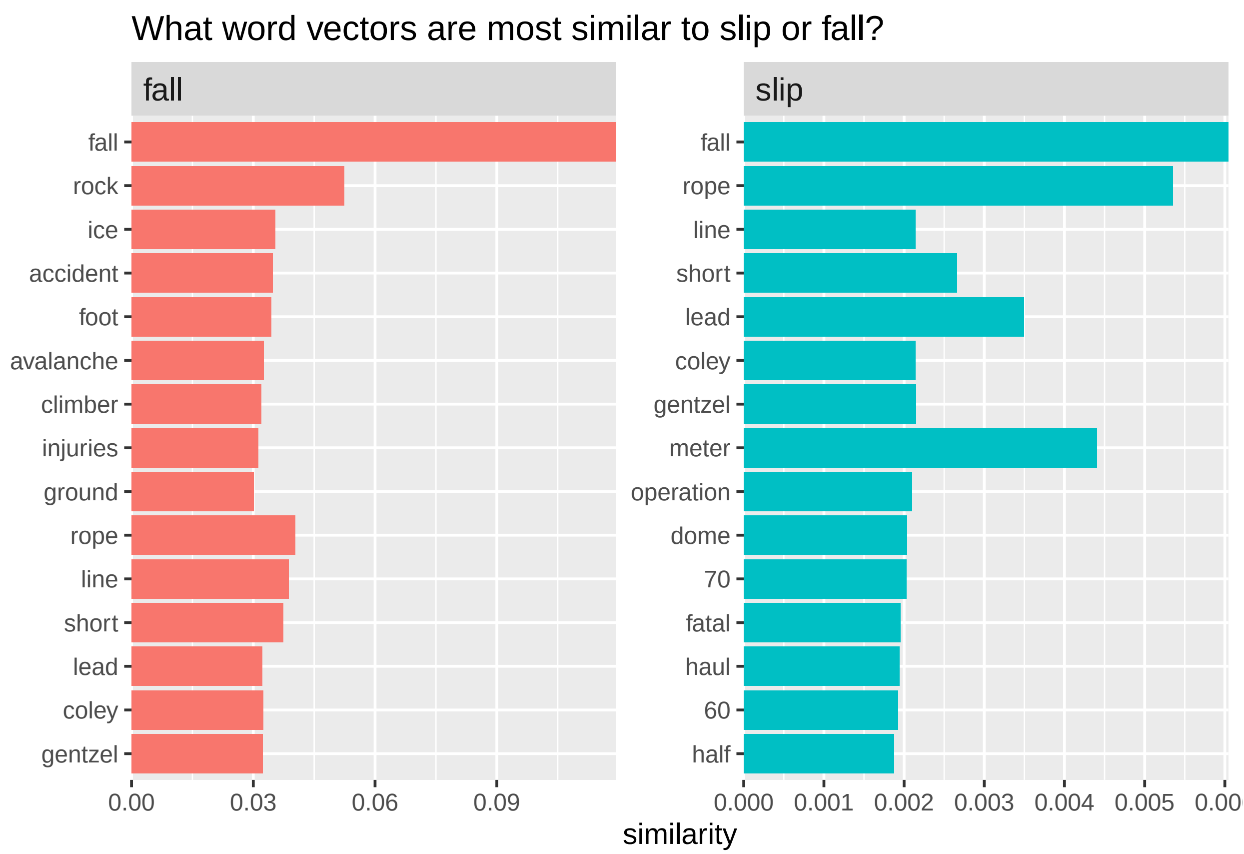
```r
pmi_matrix <- normalized_prob %>%
    mutate(pmi = log10(p_together)) %>%
    cast_sparse(word1, word2, pmi)

#remove missing data
pmi_matrix@x[is.na(pmi_matrix@x)] <- 0
#run SVD using irlba() which is good for sparse matrices
pmi_svd <- irlba(pmi_matrix, 100, maxit = 500) #Reducing to 100 dimensions
#next we output the word vectors:
word_vectors <- pmi_svd$u
rownames(word_vectors) <- rownames(pmi_matrix)
```

```r
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))
}
```

```r
fall <- search_synonyms(word_vectors,word_vectors["fall",])
slip <- search_synonyms(word_vectors,word_vectors["slip",])
```

```r
slip %>%
    mutate(selected = "slip") %>%
    bind_rows(fall %>%
```

```
                  mutate(selected = "fall")) %>%
    group_by(selected) %>%
    top_n(15, similarity) %>%
    ungroup %>%
    mutate(token = reorder(token, similarity)) %>%
    ggplot(aes(token, similarity, fill = selected)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~selected, scales = "free") +
    coord_flip() +
    theme(strip.text=element_text(hjust=0, size=12)) +
    scale_y_continuous(expand = c(0,0)) +
    labs(x = NULL, title = "What word vectors are most similar to slip or fall?")
```




```
snow_danger <- word_vectors["snow",] + word_vectors["danger",]
search_synonyms(word_vectors, snow_danger)
```

```
## # A tibble: 9,104 x 2
##    token      similarity
##    <chr>           <dbl>
##  1 snow           0.396
##  2 avalanche      0.131
##  3 conditions     0.0918
##  4 soft           0.0806
##  5 wet            0.0783
##  6 ice            0.0769
```

```
##   7 icy             0.0735
##   8 slope           0.0703
##   9 fresh           0.0604
##  10 blindness       0.0596
## # ... with 9,094 more rows
```

```
no_snow_danger <- word_vectors["danger",] - word_vectors["snow",]
search_synonyms(word_vectors, no_snow_danger)
```

```
## # A tibble: 9,104 x 2
##    token     similarity
##    <chr>          <dbl>
##  1 avalanche     0.0882
##  2 danger        0.0547
##  3 rockfall      0.0540
##  4 gulch         0.0534
##  5 class         0.0507
##  6 hazard        0.0403
##  7 hazards       0.0394
##  8 occurred      0.0376
##  9 potential     0.0373
## 10 mph           0.0361
## # ... with 9,094 more rows
```

Assignment

Download a set of pretrained vectors, GloVe, and explore them.

Grab data here:

```
# read in data
library(data.table)
```

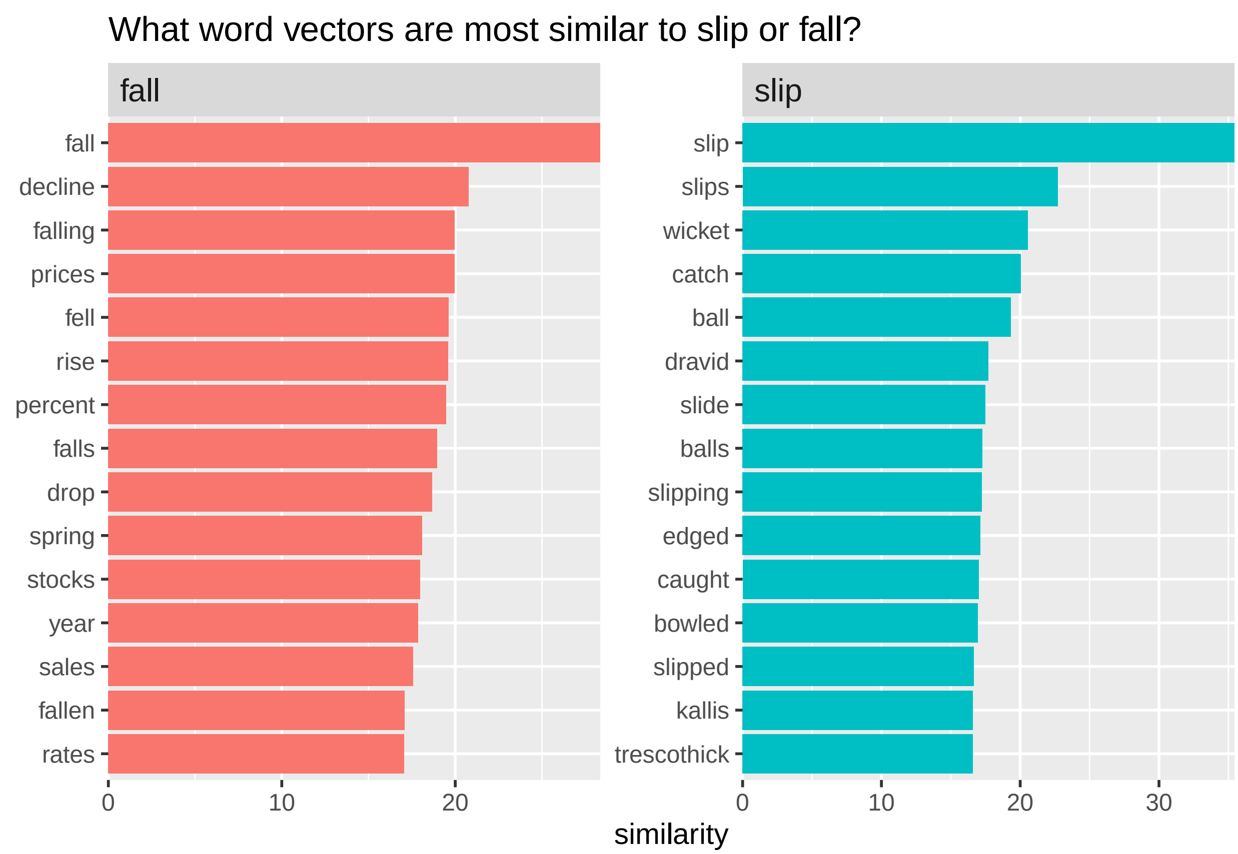
```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
glove <- fread('glove.6B.300d.txt', header = FALSE) %>%
  remove_rownames() %>%
  column_to_rownames(var = "V1")

glove <- as.matrix(glove)
```

1. Recreate the analyses in the last three chunks (find-synonyms, plot-synonyms, word-math) with the GloVe embeddings. How are they different from the embeddings created from the climbing accident data? Why do you think they are different?

```
# find synonyms

fall_2 <- search_synonyms(glove, glove["fall",])
slip_2 <- search_synonyms(glove, glove["slip",])
```

```
# plot synonyms

slip_2 %>%
    mutate(selected = "slip") %>%
    bind_rows(fall_2 %>%
                  mutate(selected = "fall")) %>%
    group_by(selected) %>%
    top_n(15, similarity) %>%
    ungroup %>%
    mutate(token = reorder(token, similarity)) %>%
    ggplot(aes(token, similarity, fill = selected)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~selected, scales = "free") +
    coord_flip() +
    theme(strip.text=element_text(hjust=0, size=12)) +
    scale_y_continuous(expand = c(0,0)) +
    labs(x = NULL, title = "What word vectors are most similar to slip or fall?")
```



```r
# word math

snow_danger <- glove["snow",] + glove["danger",]
search_synonyms(glove, snow_danger)
```

```
## # A tibble: 400,000 x 2
##    token      similarity
##    <chr>           <dbl>
##  1 snow             57.6
##  2 rain             40.6
##  3 danger           40.5
##  4 snowfall         34.8
##  5 weather          34.4
##  6 winds            34.0
##  7 rains            34.0
##  8 fog              33.6
##  9 landslides       33.3
## 10 threat           33.0
## # ... with 399,990 more rows
```

```r
no_snow_danger <- word_vectors["danger",] - word_vectors["snow",]
search_synonyms(word_vectors, no_snow_danger)
```

```
## # A tibble: 9,104 x 2
##    token     similarity
##    <chr>          <dbl>
##  1 avalanche     0.0882
##  2 danger        0.0547
##  3 rockfall      0.0540
##  4 gulch         0.0534
##  5 class         0.0507
##  6 hazard        0.0403
##  7 hazards       0.0394
##  8 occurred      0.0376
##  9 potential     0.0373
## 10 mph           0.0361
## # ... with 9,094 more rows
```

The GloVe embeddings differ from the climbing data embeddings in that the words related to fall are much more generic or related to economics in some way (e.g., decline, drop, price, stocks), as opposed to being related in a climbing context (e.g., rock, ice avalanche, climber). The synonyms for slip from the GloVe dataset were very interesting beacuse many of them were related to the game of cricket (e.g., wicket, bowled, dravid, etc.). On the other hand, in the climbing context we saw synonyms of fall related to climbing (line, rope, etc.). Since the climbing words came from climbing incident reports it makes much more sense that these synonyms would be much more concentrated within the climbing topic specifically. Another thing to point out is that the similarity values and much larger for the GloVe words (ranging to above 30) as compared to the climbing words (all below 1). The word math results were more similar, both containing words related to cold, extreme weather and hazardous conditions.

2. Run the classic word math equation, “king” - “man” = ?

```r
king_man <- glove["king",] - glove["man",]
search_synonyms(glove, king_man)
```

```
## # A tibble: 400,000 x 2
##    token       similarity
##    <chr>            <dbl>
##  1 king              35.3
##  2 kalākaua          26.8
##  3 adulyadej         26.3
##  4 bhumibol          25.9
##  5 ehrenkrantz       25.5
##  6 gyanendra         25.2
##  7 birendra          25.2
##  8 sigismund         25.1
##  9 letsie            24.7
## 10 mswati            24.0
## # ... with 399,990 more rows
```

3. Think of three new word math equations. They can involve any words you'd like, whatever catches your interest.

```r
ultimate_frisbee <- glove["frisbee",] + glove["ultimate",]
search_synonyms(glove, ultimate_frisbee)
```

```
## # A tibble: 400,000 x 2
##    token       similarity
##    <chr>            <dbl>
##  1 frisbee           54.7
##  2 ultimate          46.2
##  3 volleyball        27.9
##  4 wrestling         27.0
##  5 golf              26.7
##  6 ifbb              26.6
##  7 softball          25.4
##  8 immortality       24.0
##  9 thrill            23.3
## 10 kickball          23.2
## # ... with 399,990 more rows
```

```r
#search_synonyms(glove, glove["frisbee",])
```

I play ultimate frisbee so I wanted to see if any terms pertaining to this sport would come up. Although it is an up-and-coming sport, this word math equation very much showed how it is still an obscure sport in many ways. The results that came up were more related to other sports from volleyball to kickball, etc. and didn't have to do with ultimate frisbee at all. There were also generic sports-related terms like thrill and fitness, but words that are very related to ultimate frisbee, like disc, were much farther down on the list.

```r
university_coffee <- glove["university",] + glove["coffee",]
search_synonyms(glove, university_coffee)
```

```
## # A tibble: 400,000 x 2
##    token        similarity
##    <chr>             <dbl>
##  1 university         62.1
##  2 coffee             55.3
##  3 college            43.2
##  4 professor          42.9
##  5 harvard            39.0
##  6 faculty            38.2
##  7 graduate           37.6
##  8 universities       37.2
##  9 institute          36.7
## 10 school             36.6
## # ... with 399,990 more rows
```

I thought this word math equation would be interesting because college students drink a lot of coffee. Many college-related words came up but so did things like tea and studies and student.

```r
data_science <- glove["data",] + glove["science",]
search_synonyms(glove, data_science)
```

```
## # A tibble: 400,000 x 2
##    token       similarity
##    <chr>            <dbl>
##  1 data              65.7
##  2 science           61.9
##  3 research          53.9
##  4 scientific        52.1
##  5 technology        50.0
##  6 computer          49.5
##  7 sciences          48.9
##  8 information       48.7
##  9 physics           45.2
## 10 studies           45.0
## # ... with 399,990 more rows
```

Most of the words that came up for this word math equation didn't surprise me since they are all pretty related to tech and computers more broadly.