**Introduction to Corpus Linguistics**

SoSe 2019

by Dr. Natalia Levshina ©

**Collocations**

The goals of our today’s session is to learn

a) what collocations are;

b) how to find frequencies of combinations of two words in a corpus

c) how to compute their collocation scores

Exercise

Provide the missing word:

Who \_\_\_\_\_ the beans?

\_\_\_\_\_ breads \_\_\_\_\_\_\_.

We’ve decided to withdraw from the project for the \_\_\_\_\_\_\_\_ future.

The wrong \_\_\_\_\_ of the stick.

Sorry I’m late. I’ll make \_\_\_\_ the time this evening.

He was in trouble, but the support of his fans pulled him \_\_\_\_.

His talk was an \_\_\_\_\_\_\_ disaster!

Collocation = a sequence of words that co-occur more often than one could predict due to chance alone (i.e. based on the frequencies of the individual words).

Compare:

- *of* + *the*: both of and the are highly frequent, so it is not surprising than their combination is frequent, too.

- *foreseeable future*

*future* is a common word with 22,378 instances in the British National Corpus (Data: Taylor (2012: 107);

*foreseeable* is a rare word with 427 instances;

*foreseeable* + *future* occurs 294 times: two thirds of the total frequency of foreseeable!

For computing different measures of collocation strength, we need the following for frequencies for each pair of words:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Word 2** | **Not Word 2** | **Marginal row frequencies** |
| **Word 1** | a | b | a + b (the total frequency of Word 1) |
| **Not Word 1** | c | d | c + d (the total frequency of all words other than Word 1) |
| **Marginal column frequencies** | a + c  (the total frequency of Word 2) | b + d (the total frequency of all words other than Word 2) | Total size of corpus: N = a + b + c + d |

The frequency d can be computed as the total size of the corpus minus a, b and c.

Exercises

1. Compute the frequencies a, b, c and d and all marginal frequencies for *foreseeable* + *future*. The size of the British National Corpus is 100M words.

2. The formula for **expected frequency** is as follows:

E = (a + b)\*(a + c)/N = (F1\*F2)/N

where F1 is the frequency of the first word, F2 is the frequency of the second word, and N is the total corpus size.

Compute the **expected frequency** of *foreseeable* + *future* and compare it with the **observed frequency**!

Where can collocations be important?

* lexicography
* translation studies
* language teaching
* language for specific purposes (e.g. legal or scientific terminology)
* psycholinguistics and child language acquisition (pre-fabricated chunks)
* distributional models of semantics

**Data**

We’ll use the same data from trump\_big.txt as for the extraction of n-grams.

**R practice**

First, we repeat the same steps as in the previous study of n-grams. If you have saved the object *trump\_tokens*, skip steps 1 to 4.

1. Open the corpus in R. We’ll treat each new tweet as a separate unit:

**trump <- scan(file = file.choose(), what = "character", sep = "\n", quote = "", encoding = "UTF-8")**

**head(trump)**

[1] "RT @GOP: “Tomorrow the voters of this state will cast their ballots in one of the most important Senate elections of your lives—of all of…"

[2] "RT @GOP: “Only with a strong Senate GOP majority can we defend your tax cuts defend your Second Amendment protect your Medicare and Socia…"

[3] "RT @GOP: “I need the great people of Mississippi to send a message to… the radical Democrats by electing @cindyhydesmith.” -@realDonaldTrum…"

[output omitted]

2. Remove all units that are retweets and include the string “RT @”:

First, get the indices of those tweets.

**rt <- grep("RT @", trump)**

**head(rt)**

[1] 1 2 3 25 26 41

Next, remove all tweets with those indices:

**trump <- trump[-rt]**

**head(trump)**

[1] "....starts today election is on December 4th. @VoteBradRaff is tough on Crime and Borders Loves our Military and Vets. He will be great for jobs!"

[output omitted]

Now you see that the first retweets are gone.

3. Extract all tokens

First, we should split each of the tweets into tokens. The result is a list of 34927 elements (tweets), each with nested tokens. We will merge all of them with the help of function unlist(). As a result, we have 634080 tokens.

**trump\_tokens <- strsplit(trump, split = " ")**

**length(trump\_tokens)**

[1] 34927

**head(trump\_tokens)**

[[1]]

[1] "....starts" "today" "election" "is" "on" "December"

[7] "4th." "@VoteBradRaff" "is" "tough" "on" "Crime"

[output omitted]

**trump\_tokens <- unlist(trump\_tokens)**

**head(trump\_tokens)**

[1] "....starts" "today" "election" "is" "on" "December"

**length(trump\_tokens)**

[1] 634080

4. Clean up the data

As usual, we remove all punctuation, turn the capital letters into lowercase and remove all empty strings.

**trump\_tokens <- tolower(trump\_tokens)**

**trump\_tokens <- gsub("[[:punct:]]", "", trump\_tokens)**

**trump\_tokens <- trump\_tokens[nchar(trump\_tokens) > 0]**

**head(trump\_tokens)**

[1] "starts" "today" "election" "is" "on" "december"

5. Create a data frame with the left neighbours, right neighbours and the bigrams:

**final <- length(trump\_tokens)**

**grams1 <- trump\_tokens[-final]**

**grams2 <- trump\_tokens[-1]**

**bigrams <- paste(grams1, grams2, sep = " ")**

**#NEW!!!**

**trump\_df <- data.frame(grams1, grams2, bigrams)**

**head(trump\_df)**

# grams1 grams2 bigrams

#1 starts today starts today

#2 today election today election

#3 election is election is

#4 is on is on

#5 on december on december

#6 december 4th december 4th

6. Now we need to compute the frequencies of each 1-gram and each 2-gram.

**unigrams\_freq <- table(trump\_tokens)**

**bigrams\_freq <- table(bigrams)**

7. For technical purposes, we need to turn these frequency vectors into data frames with the words/bigrams as one column and their frequencies as the second column:

**unigram\_freq\_df <- data.frame(unigram = names(unigrams\_freq),**

**unigram\_freq = as.numeric(unigrams\_freq))**

**head(unigram\_freq\_df)**

# unigram unigram\_freq

#1 – 410

#2 –“what 1

#3 –barnesandnoblecom 1

#4 –but 1

#5 –cuomo 1

#6 –donald 3

**bigram\_freq\_df <- data.frame(bigram = names(bigrams\_freq),**

**bigram\_freq = as.numeric(bigrams\_freq))**

**head(bigram\_freq\_df)**

bigram bigram\_freq

1 – ‘midas 1

2 – “donald 2

3 – “jeb 1

4 – “trump 1

5 – 1 1

6 – 10th 1

You can see that there’s a lot of noise. We’ll ignore it for the time being.

8. Put together all frequencies: bigrams and individual lemmas.

First, we get rid of repeated lines in the data frame with words and their combinations because we already have all frequency information. We only need unique combinations now.

**trump\_unique <- unique(trump\_df)**

**dim(trump\_unique)**

[1] 274175 3

Next, we need to put all frequencies together. For each combination, we need to get the frequency of the left neighbour, the frequency of the right neighbour, and the frequency of the combination.

First, we add the frequency of the left neighbour:

**trump\_all <- merge(trump\_unique, unigram\_freq\_df, by.x = "grams1", by.y = "unigram")**

**head(trump\_all)**

grams1 grams2 bigrams unigram\_freq

1 – obamacare – obamacare 410

2 – owner – owner 410

3 – of – of 410

4 – heated – heated 410

5 – time – time 410

6 – lao – lao 410

**colnames(trump\_all)[4] <- "freq\_left"**

Next, we add the frequency of the right neighbour, using the same frequency lists of unigrams.

**trump\_all <- merge(trump\_all, unigram\_freq\_df, by.x = "grams2", by.y = "unigram")**

**head(trump\_all)**

grams2 grams1 bigrams freq\_left unigram\_freq

1 – protecting” protecting” – 1 410

2 – persistent” persistent” – 2 410

3 – 6th 6th – 9 410

4 – benefit” benefit” – 1 410

5 – forces forces – 26 410

6 – quit” quit” – 2 410

**colnames(trump\_all)[5] <- "freq\_right"**

Finally, we add the frequencies of the combinations (bigrams).

**trump\_all <- merge(trump\_all, bigram\_freq\_df, by.x = "bigrams", by.y = "bigram")**

**head(trump\_all)**

bigrams grams2 grams1 freq\_left freq\_right bigram\_freq

1 – ‘midas ‘midas – 410 1 1

2 – “donald “donald – 410 130 2

3 – “jeb “jeb – 410 2 1

4 – “trump “trump – 410 111 1

5 – 1 1 – 410 276 1

6 – 10th 10th – 410 7 1

9. Compute the frequencies b, c and d.

Now we have the frequency a (bigram\_freq) from the table. But we need the frequencies b, c and d. The frequency b (the frequency of the left neighbour with all other words) can be obtained as freq\_b = freq\_left – bigram\_freq. The frequency c is the frequency of the right neighbour with all other words. It can be computed as follows: freq\_c = freq\_right – bigram\_freq. The frequency d can be computed as the frequency of all tokens (number of tokens in the corpus) minus the frequencies a, b and c.

**length(trump\_tokens)**

#[1] 618812

**trump\_all$freq\_b <- trump\_all$freq\_left - trump\_all$bigram\_freq**

**trump\_all$freq\_c <- trump\_all$freq\_right - trump\_all$bigram\_freq**

**trump\_all$freq\_d <- length(trump\_tokens) - trump\_all$bigram\_freq - trump\_all$freq\_b - trump\_all$freq\_c**

**head(trump\_all)**

#[…]

10. Compute collocational strength (log Odds Ratios).

Log Odds Ratios

LOR = log (a\*d/(b\*c))

In order to avoid division by 0 (if b or c is equal to 0), we add a small amount to each frequency:

Adjusted LOR = log ((a + 0.5)\*(d + 0.5)/((b + 0.5)\*(c + 0.5))

**trump\_all$LOR <- log((trump\_all$bigram\_freq + 0.5)\*(trump\_all$freq\_d + 0.5)/((trump\_all$freq\_b + 0.5)\*(trump\_all$freq\_c + 0.5)))**

It may be useful to inspect only frequent collocations (e.g. more frequent than 5):

**trump\_short <- trump\_all[trump\_all$bigram\_freq > 5,]**

**trump\_short[order(-trump\_short$LOR), ][1:20, c(1,10)]**

bigrams LOR

65896 des moines 17.69224

264527 witch hunt 17.22530

32555 baton rouge 17.16418

188796 puerto rico 16.81365

256006 waldo emerson 15.87452

55407 conde nast 15.76329

203439 saudi arabia 15.66439

193 ‘all star’ 15.63813

75101 eminent domain 15.63813

170756 olivia culpo 15.49503

30720 babe ruth 15.36369

38389 bin laden 15.34434

177520 palos verdes 15.30165

179193 pearl harbor 15.12730

143059 los angeles 15.12729

264255 winston churchill 15.02718

92923 gina haspel 14.98420

27358 arnold palmer 14.68260

64823 del este 14.66468

198622 rex tillerson 14.66468

**Exercise**

Inspect the collocations with LOR greater than 5. What classes of collocations can you identify? What are your conclusions about the topics of Trumps’ tweets?

**Exercise**

Use the corpus from Pottermore and find the top 20 strongest collocations.