**Introduction to Corpus Linguistics**

WiSe 2018-2019

by Dr. Natalia Levshina ©

**Session 5. 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 UD corpus, parsed with the help of udpipe;

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 a corpus compiled from online Wikipedia articles, which is freely available from the Leipzig Corpora Collection. One can also download news and miscellaneous websites in about 200 languages. The URL is as follows:

<http://wortschatz.uni-leipzig.de/de/download>

The corpus has already been parsed by myself (in order to save time!). Please save it locally first.

The corpus is available from GitHub. Its name is eng\_wiki\_ud.txt. Only 8 columns are available. This is what the first lines look like. The first line contains the column names. Do you remember what they mean?

token\_id token lemma upos xpos feats head\_token\_id dep\_rel

1 10 10 NUM CD NumType=Card 2 nummod

2 Years year NOUN NNS Number=Plur 6 nsubj

3 of of ADP IN NA 5 case

4 Time time NOUN NN Number=Sing 5 compound

5 Team team NOUN NN Number=Sing 2 nmod

6 presented present VERB VBD Mood=Ind|Tense=Past|VerbForm=Fin 0 root

7 a a DET DT Definite=Ind|PronType=Art 10 det

8 round round NOUN NN Number=Sing 10 compound

**R practice**

1. Open the corpus in R.

wiki <- read.table(file = file.choose(), header = T, sep = "\t", quote = "")

dim(wiki)

#[1] 243131 8

head(wiki)

… #The same as in the corpus file

This means we have more than 243K tokens – words, numbers and punctuation marks. We won’t make a distinction between those for the time being. As you’ll see that won’t affect the results. See previous files for information how to remove punctuation marks.

2. Extract all lemmas.

wiki\_lemmas <- wiki$lemma

length(wiki\_lemmas)

#[1] 243131

3. Compute all frequencies of lemmas.

We’ll use the familiar function table() to compute the frequencies. For technical reasons, we need to transform the numeric vector with frequencies into a data frame.

wiki\_lemmas\_freq <- table(wiki\_lemmas)

sort(wiki\_lemmas\_freq, decreasing = TRUE)[1:20]

#wiki\_lemmas

#the , . be of and to a in " he as - have for that ( ) with by

#15130 10910 10347 7913 7190 5855 5355 5308 5149 2531 1999 1828 1805 1787 1785 1712 1681 1663 1482 1447

wiki\_lemmas\_freq\_df <- data.frame(lemma = names(wiki\_lemmas\_freq),

lemma\_freq= as.numeric(wiki\_lemmas\_freq))

head(wiki\_lemmas\_freq\_df)

#lemma lemma\_freq

#1 ' 369

#2 '' 1

#3 '. 1

#4 '06 1

#5 '60 1

#6 '74 1

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

4. Create pairwise combinations of words by putting together pairs of lemmas.

For example, the first lemma will go together with the second lemma, the second one with the third one, and so on. We can create two lists (factors) of lemmas: the left neighbour and the right neighbour. The last lemma doesn’t have another word on the right, so we exclude it.

wiki\_lemmas\_right <- wiki$lemma[2:243131]

length(wiki\_lemmas\_right)

#[1] 243130

wiki\_lemmas1 <- wiki\_lemmas[-243131]

wiki\_coll <- data.frame(left = wiki\_lemmas1, right = wiki\_lemmas\_right)

head(wiki\_coll)

#left right together

#1 10 year

#2 year of

#3 of time

#4 time team

#5 team present

#6 present a

Now we can create the combination of the words – candidates for collocations:

wiki\_coll$together <- as.factor(paste(wiki\_coll$left, wiki\_coll$right))

head(wiki\_coll)

#left right together

#1 10 year 10 year

#2 year of year of

#3 of time of time

#4 time team time team

#5 team present team present

#6 present a present a

5. Compute the frequencies of the combinations of words.

As usual, we use the function table(). As in Step 3, we need to create a data frame from the frequency vector.

wiki\_together\_freq <- table(wiki\_coll$together)

sort(wiki\_together\_freq, decreasing = TRUE)[1:10]

#of the . the in the , and , the to the . in it be and the ) .

#1966 1696 1284 1162 866 803 630 500 472 465

wiki\_together\_freq\_df <- data.frame(coll = names(wiki\_together\_freq),

freq\_coll = as.numeric(wiki\_together\_freq))

Not surprisingly, the function words and punctuation marks are the most frequent combinations. They are not interesting for us, however. What we need are those collocations that are more frequent than expected. For this purpose, we need to compute collocational strength, using the frequencies a, b, c and d.

6. Put together all frequencies: collocations 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. Note that this will make the list almost twice shorter.

wiki\_coll\_unique <- unique(wiki\_coll)

#[1] 134905 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:

wiki\_coll\_all <- merge(wiki\_coll\_unique, wiki\_lemmas\_freq\_df, by.x = "left", by.y = "lemma")

head(wiki\_coll\_all)

#left right together lemma\_freq

#1 ' bilingual ' bilingual 369

#2 ' Harbor ' Harbor 369

#3 ' average ' average 369

#4 ' training ' training 369

#5 ' Silent ' Silent 369

#6 ' attention ' attention 369

colnames(wiki\_coll\_all)[4] <- "freq\_left"

Next, we add the frequency of the right neighbour. Note that we use the same list of frequencies in wiki\_lemmas\_freq\_df!

wiki\_coll\_all <- merge(wiki\_coll\_all, wiki\_lemmas\_freq\_df, by.x = "right",

by.y = "lemma")

head(wiki\_coll\_all)

#right left together freq\_left lemma\_freq

#1 ' Tamagotchis Tamagotchis ' 2 369

#2 ' 'romance 'romance ' 1 369

#3 ' wholesaler wholesaler ' 2 369

#4 ' word word ' 117 369

#5 ' cold cold ' 12 369

#6 ' Boas Boas ' 4 369

colnames(wiki\_coll\_all)[5] <- "freq\_right"

Finally, we add the frequencies of the combinations.

wiki\_coll\_all <- merge(wiki\_coll\_all, wiki\_together\_freq\_df, by.x = "together", by.y = "coll")

head(wiki\_coll\_all)

#together right left freq\_left freq\_right freq\_coll

#1 '' " " '' 1 2531 1

#2 ' - - ' 369 1805 1

#3 ' " " ' 369 2531 6

#4 ' & & ' 369 52 1

#5 ' ( ( ' 369 1681 2

#6 ' ) ) ' 369 1663 4

7. Compute the frequencies b, c and d.

Now we have the frequency a (freq\_coll) 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 - freq\_coll. The frequency c is the frequency of the right neighbour with all other words. It can be computed as follows: freq\_c = freq\_right – freq\_coll. The frequency d can be computed as the frequency of all tokens (number of rows in the corpus) minus the frequencies a, b and c.

nrow(wiki)

#[1] 243131

wiki\_coll\_all$freq\_b <- wiki\_coll\_all$freq\_left - wiki\_coll\_all$freq\_coll

wiki\_coll\_all$freq\_c <- wiki\_coll\_all$freq\_right - wiki\_coll\_all$freq\_coll

wiki\_coll\_all$freq\_d <- nrow(wiki) - wiki\_coll\_all$freq\_coll - wiki\_coll\_all$freq\_b - wiki\_coll\_all$freq\_c

head(wiki\_coll\_all)

#together right left freq\_left freq\_right freq\_coll freq\_b freq\_c freq\_d

#1 '' " " '' 1 2531 1 0 2530 240600

#2 ' - - ' 369 1805 1 368 1804 240958

#3 ' " " ' 369 2531 6 363 2525 240237

#4 ' & & ' 369 52 1 368 51 242711

#5 ' ( ( ' 369 1681 2 367 1679 241083

#6 ' ) ) ' 369 1663 4 365 1659 241103

8. 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))

wiki\_coll\_all$LOR <- log((wiki\_coll\_all$freq\_coll + 0.5)\*(wiki\_coll\_all$freq\_d + 0.5)/((wiki\_coll\_all$freq\_b + 0.5)\*(wiki\_coll\_all$freq\_c + 0.5)))

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

wiki\_coll\_short <- wiki\_coll\_all[wiki\_coll\_all$freq\_coll > 5,]

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

together LOR

75154 Los Angeles 16.138986

103277 San Francisco 13.326204

51128 et al. 12.714954

96278 Prime Minister 12.095894

111992 Supreme Court 11.753265

127636 United States 11.656189

19947 Air Force 11.005428

82405 New Zealand 10.989902

82403 New York 10.696120

63224 High School 10.545776

127635 United Nations 9.641755

78640 Middle East 9.494288

13324 19th century 9.417907

32713 blood vessel 9.162544

44409 de la 9.038874

12269 16th century 9.034846

127633 United Kingdom 8.918725

133881 World War 8.898453

59877 Great Britain 8.676736

13721 20th century 8.650618

**Exercise**

Inspect the collocations with LOR greater than 5. What classes of collocations can you identify? What are your conclusions about the nature of Wikipedia, the primary interests of its creators?

#Tip:

wiki\_coll\_short[wiki\_coll\_short$LOR > 5, c(1, 10)]

**Exercise**

a) Use the corpus of Trump’s tweets and find the top 20 strongest collocations.

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

It may be useful to know how to extract the top 20 strongest collocates of a word. Let us take the word **language** and find its left collocates:

language <- wiki\_coll\_all[wiki\_coll\_all$right == "language",]

language[order(-language$LOR), c(1, 10)][1:20,]

together LOR

26218 aryan language 8.733046

41503 constructed language 8.733046

58013 fusional language 8.733046

67209 Incan language 8.733046

67970 inflected language 8.733046

71590 Khoisan language 8.733046

77066 Markup language 8.733046

77390 MATLAB language 8.733046

107282 Slavic language 8.733046

127847 unspoken language 8.733046

128160 Uralic language 8.733046

129161 Ventureño language 8.733046

129703 Visayan language 8.733046

32226 Biblical language 7.634430

51622 everyday language 7.634430

58853 Germanic language 7.634430

67411 inclusive language 7.634430

93361 philippine language 7.634430

96962 programming language 7.400271

70587 JavaScript language 7.123600

**Exercise**

Choose another word and explore its left and right collocates. Present the results of your collocation analysis.

**Part II. Different measures of collocational strength**

Check how well you remember the following concepts:

* What are the frequencies a, b, c and d?
* What are the marginal frequencies?
* What is the expected frequency?
* What is LOR?

Let us also compute the expected frequencies for every cell in the co-occurrence table: a\_exp, b\_exp, c\_exp, d\_exp. For example, expected frequency a is computed as follows:

**Formula for the expected frequency a**

**(frequency of the combination of words)**

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

In R:

wiki\_coll\_all$a\_exp <- (wiki\_coll\_all$freq\_coll + wiki\_coll\_all$freq\_b)\*(wiki\_coll\_all$freq\_coll + wiki\_coll\_all$freq\_c)/ 243131

The expected frequency of b is b\_exp:

**Formula for the expected frequency b**

**(frequency of the left neighbour in all other contexts)**

b\_exp = (b + a)\*(b + d)/N

In R:

wiki\_coll\_all$b\_exp <- (wiki\_coll\_all$freq\_b + wiki\_coll\_all$freq\_coll)\*(wiki\_coll\_all$freq\_b + wiki\_coll\_all$freq\_d)/ 243131

**Exercise**

What are the formulas for the expected frequencies of c and d?

Compute c\_exp and d\_exp in R using the same logic.

Using the observed frequencies a, b, c and d, and the corresponding expected frequencies, we can compute several popular measures of collocation strength.

**Formulas**

t-score: *t* = (a – a\_exp)/

Pointwise Mutual Information: PMI = log2 (a /a\_exp)

Log-likelihood: *L* = 2\*(a \* log(a / a\_exp) + b \* log(b / b\_exp) + c \* log(c / c\_exp) + d\*log(d / d\_exp))

In R:

wiki\_coll\_all$t\_score <- (wiki\_coll\_all$freq\_coll - wiki\_coll\_all$a\_exp)/sqrt(wiki\_coll\_all$freq\_coll)

wiki\_coll\_all$PMI <- log2(wiki\_coll\_all$freq\_coll/wiki\_coll\_all$a\_exp)

wiki\_coll\_all$LogLik <- 2\*( wiki\_coll\_all$freq\_coll \* log(wiki\_coll\_all$freq\_coll / wiki\_coll\_all$a\_exp) + wiki\_coll\_all$freq\_b \* log(wiki\_coll\_all$freq\_b / wiki\_coll\_all$b\_exp) + wiki\_coll\_all$freq\_c \* log(wiki\_coll\_all$freq\_c / wiki\_coll\_all$c\_exp) + wiki\_coll\_all$freq\_d\*log(wiki\_coll\_all$freq\_d / wiki\_coll\_all$d\_exp))

Let us have a look at the left collocates of the word **party**.

party <- wiki\_coll\_all[wiki\_coll\_all$right == "party",]

party[, 1]

[1] - party ) party , party 1927 party 45 party

[6] all party as party book party coerced party commercial party

[11] convention party declare party Democratic party Democrats party he party

[16] interested party Islamist party line party loyal party main party

[21] majority party numerous party one party opposition party outside party

[26] own party parliamentary party political party populist party progressive party

[31] Republican party responsible party search party that party the party

[36] third party this party two party when party

134905 Levels: '' " ' - ' " ' & ' ( ' ) ' , ' . ' : ' 29 ' a ' a' ' aa ' ability ' Ac ' action ' ad ' affair ... Zytek )

party[order(-party$LOR), c(1, 10)]

together LOR

39425 coerced party 9.2440652

94960 populist party 9.2440652

45149 Democratic party 8.6705604

69553 Islamist party 8.1454488

100858 Republican party 7.6346191

75406 loyal party 7.2981427

45158 Democrats party 7.0468242

89135 opposition party 6.9359306

68754 interested party 6.8461494

91880 parliamentary party 6.5359862

#…

party[order(-party$t\_score), c(1, 15)]

together t\_score

94643 political party 2.9924074

118480 the party 2.8954206

1830 - party 1.7364487

121791 third party 1.7210918

45149 Democratic party 1.4135941

89135 opposition party 1.4121486

126803 two party 1.3318233

62270 he party 1.0014363

39425 coerced party 0.9997080

94960 populist party 0.9997080

69553 Islamist party 0.9994160

party[order(-party$PMI), c(1, 16)]

together PMI

39425 coerced party 11.7416272

94960 populist party 11.7416272

45149 Democratic party 11.1566647

69553 Islamist party 10.7416272

100858 Republican party 10.1566647

75406 loyal party 9.7416272

45158 Democrats party 9.4196991

89135 opposition party 9.4196991

68754 interested party 9.1566647

91880 parliamentary party 8.7416272

#...

party[order(-party$LogLik), c(1, 17)]

together LogLik

94643 political party 91.9335549

45149 Democratic party 28.7930440

121791 third party 24.6812854

89135 opposition party 22.6080544

118480 the party 20.1674150

69553 Islamist party 13.5194858

100858 Republican party 12.4735654

75406 loyal party 11.7945452

45158 Democrats party 11.2897780

68754 interested party 10.8876436

91880 parliamentary party 10.2672074

#...

**Exercise**

1. Compare the collocation strength measures of the right collocates of **party**. What are your conclusions?

2. Choose another word and report the results of your collocational analysis.

Now we can compare the top 20 collocations, produced by LOR, t-score, PMI and Log-Likelihood. Again, it is useful to focus only on high-frequency collocations:

wiki\_coll\_short <- wiki\_coll\_all[wiki\_coll\_all$freq\_coll > 5,]

What are the differences?

#1. Log Odds Ratio

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

together LOR

75154 Los Angeles 16.138986

103277 San Francisco 13.326204

51128 et al. 12.714954

96278 Prime Minister 12.095894

111992 Supreme Court 11.753265

127636 United States 11.656189

19947 Air Force 11.005428

82405 New Zealand 10.989902

82403 New York 10.696120

63224 High School 10.545776

127635 United Nations 9.641755

78640 Middle East 9.494288

13324 19th century 9.417907

32713 blood vessel 9.162544

44409 de la 9.038874

12269 16th century 9.034846

127633 United Kingdom 8.918725

133881 World War 8.898453

59877 Great Britain 8.676736

13721 20th century 8.650618

#2. t-scores

wiki\_coll\_short[order(-wiki\_coll\_short$t\_score), c(1, 15)][1:20,]

together t\_score

87083 of the 34.24856

67038 in the 26.89086

5283 , and 26.38071

10417 . the 25.54744

69728 it be 20.63881

4585 ) . 18.28185

5508 , but 17.86404

88341 on the 17.21139

26251 as a 17.01606

124786 to the 16.57744

9183 . in 16.36955

10432 . this 16.15488

120697 there be 15.65301

9248 . it 14.97682

61131 have be 14.96429

47253 do not 14.83745

4584 ) , 14.75188

7861 , which 14.59631

30322 be not 14.48536

64269 however , 14.17595

#3. Pointwise Mutual Information

wiki\_coll\_short[order(-wiki\_coll\_short$PMI), c(1, 16)][1:20,]

together PMI

75154 Los Angeles 14.56945

51128 et al. 13.98448

111992 Supreme Court 13.76209

19947 Air Force 13.60597

96278 Prime Minister 13.50594

103277 San Francisco 13.24752

63224 High School 12.90143

78640 Middle East 12.02984

32713 blood vessel 12.02513

59877 Great Britain 11.40666

44409 de la 11.35222

108801 Soviet Union 11.34705

82405 New Zealand 11.05848

82403 New York 10.93718

130220 War ii 10.91409

127636 United States 10.90387

133881 World War 10.86186

127635 United Nations 10.84698

46777 disk drive 10.69591

12269 16th century 10.63556

#4. Log-likelihood ratio

wiki\_coll\_short[order(-wiki\_coll\_short$LogLik), c(1, 17)][1:20,]

together LogLik

87083 of the 3327.3460

47253 do not 2028.9032

67038 in the 1912.2678

5283 , and 1891.5080

69728 it be 1887.2280

5508 , but 1585.3223

120697 there be 1438.7485

10417 . the 1382.7637

127636 United States 1374.7309

111454 such as 1325.4397

64269 however , 1211.4128

4585 ) . 1085.1613

26251 as a 1049.5171

10432 . this 959.7032

88341 on the 884.5911

82403 New York 874.0357

116762 the first 871.7758

7861 , which 832.6319

8415 . citation 806.5536

35898 can be 794.2756

**Exercise**

a) Use the corpus of Trump’s tweets and compare the top 20 strongest collocations using the four measures. Find all top collocations with one word and report the results.

b) Use the corpus from Pottermore and compare the top 20 strongest collocations using the four measures. Find all top collocations with one word and report the results.