**Introduction to Corpus Linguistics**

WiSe 2018-2019

by Dr. Natalia Levshina ©

**Session 2. Frequency Lists**

The first goal of this session is to learn to make frequency lists and identify the frequencies of any word, part of speech or syntactic dependency in a UD corpus.

The second goal is to learn how to make your own corpus (in any popular language), open the file(s) in R and parse them using the *udpipe* package.

**1. How to compute the frequency of a word in a UD corpus**

First, we’ll use the code from the previous session in order to open the UD corpus file in R and obtain the tokens.

**ud <- scan(file = file.choose(), what = "character", sep = "\n", comment.char = "#", encoding = "UTF-8")**

**head(ud)**

[1] "1\tFrom\tfrom\tADP\tIN\t\_\t3\tcase\t3:case\t\_"

[2] "2\tthe\tthe\tDET\tDT\tDefinite=Def|PronType=Art\t3\tdet\t3:det\t\_"

...

Next, we need to split the lines into the UD slots by tab characters.

**ud\_list <- strsplit(ud, "\t")**

**head(ud\_list)**

[[1]]

[1] "1" "From" "from" "ADP" "IN" "\_" "3" "case" "3:case" "\_"

…

The next step is to eliminate the punctuation marks.

ud\_list\_punct <- sapply(ud\_list, function (x) x[4] == "PUNCT")

ud\_list\_punct[1:10]

[1] FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE

Exclude these elements from the list of tokens:

tokens <- sapply(ud\_list, function (x) x[2])

tokens[1:10]

[1] "From" "the" "AP" "comes" "this" "story" ":"

[8] "President" "Bush" "on"

tokens\_nopunct <- tokens[!ud\_list\_punct]

tokens\_nopunct[1:10]

[1] "From" "the" "AP" "comes" "this" "story" "President"

[8] "Bush" "on" "Tuesday"

Now we need to normalize the forms, turning the upper-case letters into lower-case letters:

tokens\_lower <- tolower(tokens\_nopunct)

tokens\_lower[1:10]

[1] "from" "the" "ap" "comes" "this" "story" "president"

[8] "bush" "on" "tuesday"

This was the old material. Our goal now is to compute the frequencies of each token.

tokens\_freq <- table(tokens\_lower)

head(tokens\_freq)

tokens\_lower

' 'd 'll 'm 're 's

21 4 12 25 12 120

Let us sort the tokens now by frequencies, from the most frequent to the least frequent. We can examine the top 20 words.

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

tokens\_lower

the to and a i of in is you for it that have on are with this

981 561 559 504 447 388 365 333 324 241 235 201 176 168 155 147 139

they 's was

133 120 120

We can also look at the words in any range, e.g. from ranks 150 to 170:

sort(tokens\_freq, decreasing = TRUE)[150:170]

tokens\_lower

file hope information price recommend then

17 17 17 17 17 17

those year around ca call email

17 17 16 16 16 16

looking miramar thank too two many

16 16 16 16 16 15

phone pizza think

15 15 15

**Exercise**

Investigate different frequency ranges. Find at least **two** differences between the most frequent words and the words with low frequencies.

How to get the frequency of a word you are interested in? Let us take the word *language*:

tokens\_freq[names(tokens\_freq) == "language"]

language

3

**Exercise**

Compare the frequencies of the wordforms *war* and *peace*. Which of them is more frequent?

How to get all words with a certain frequency, e.g. 20?

tokens\_freq[tokens\_freq == 20]

tokens\_lower

after called come much name over

20 20 20 20 20 20

How to get all words with the frequency greater than 100?

> tokens\_freq[tokens\_freq > 100]

tokens\_lower

's a and are at be do for have i in is it my

120 504 559 155 103 112 108 241 176 447 365 333 235 108

not of on that the they this to was with you

104 388 168 201 981 133 139 561 120 147 324

How to get all words with the frequency greater than 50 but less than 100?

> tokens\_freq[tokens\_freq < 100&tokens\_freq > 50]

tokens\_lower

about all an as been best but by can

64 80 54 91 54 52 98 65 84

from get good great has if just know like

75 62 73 86 79 95 51 54 68

me n't or service so there very we what

84 89 89 51 53 86 65 84 79

will would your

95 55 70

**Exercise**

Using the same approach, get the frequencies of different parts of speech in the corpus. Note: you do not need to perform normalization or get rid of punctuation marks! You need only two lines of R code.

**2. How to parse your own text file and analyze the frequencies**

First, install two packages from CRAN: udpipe and readr. For this purpose, you can use the Packages tab > Install.

Next, load the packages:

library(udpipe)

library(readr)

The next step is to open the text file we want to analyze. Let’s take the lyrics of Ed Sheeran’s song Perfect.

perfect <- read\_file(file = file.choose())

perfect

[1] "I found a love for me.\r\nDarling, just dive right in\r\nAnd follow my lead.\r\nWell, I found a girl beautiful and sweet. \r\nI never knew you were the someone waiting for me\r\n'Cause we were just kids when we fell in love,\r\nNot knowing what it was.\r\nI will not give you up this time.\r\nBut darling, just kiss me slow, your heart is all I own.\r\nAnd in your eyes you're holding mine.\r\nBaby, I'm dancing in the dark with you between my arms,\r\nBarefoot on the grass

…

Next, we need to download and run the UD parser:

ud\_eng <- udpipe\_download\_model(language = "english")

Downloading udpipe model from https://raw.githubusercontent.com/jwijffels/udpipe.models.ud.2.0/master/inst/udpipe-ud-2.0-170801/english-ud-2.0-170801.udpipe to C:/Users/natalevs/Documents/R/Teaching/Jena/english-ud-2.0-170801.udpipe

trying URL 'https://raw.githubusercontent.com/jwijffels/udpipe.models.ud.2.0/master/inst/udpipe-ud-2.0-170801/english-ud-2.0-170801.udpipe'

Content type 'application/octet-stream' length 16741882 bytes (16.0 MB)

downloaded 16.0 MB

ud\_eng <- udpipe\_load\_model(ud\_eng)

Now we can parse the lyrics:

perfect\_ud <- udpipe(perfect, object = ud\_eng)

head(perfect\_ud)

…

colnames(perfect\_ud)

[1] "doc\_id" "paragraph\_id" "sentence\_id" "sentence"

[5] "start" "end" "term\_id" "token\_id"

[9] "token" "lemma" "upos" "xpos"

[13] "feats" "head\_token\_id" "dep\_rel" "deps"

[17] "misc"

The object perfect\_ud is a data frame. It is like an Excel table with rows and columns. The columns can be of any kind and contain numbers or text. You can see that there are more columns in the data than in the original UD corpus that we have been using. This is not a problem. Let us make a frequency list of lemmas. First, we need to get a vector with all lemmas. This is very easy. We can take the

perfect\_lemmas <- perfect\_ud$lemma

head(perfect\_lemmas)

[1] "I" "find" "a" "love" "for" "I"

Now you know how to compute and sort the frequencies:

perfect\_freq <- table(perfect\_lemmas)

sort(perfect\_freq, decreasing = TRUE)

perfect\_lemmas

, I . you be in

34 27 24 17 13 11

my we the a look to

10 9 7 6 6 6

just know love and find not

5 5 5 4 4 4

perfect this will arm baby Barefoot

4 4 4 3 3 3

between but carry dance dark favorite

3 3 3 3 3 3

grass listen on see she song

3 3 3 3 3 3

tonight when with all beautiful darl

3 3 3 2 2 2

Darling deserve do eye for girl

2 2 2 2 2 2

have hold it kid own share

2 2 2 2 2 2

so than that time well what

2 2 2 2 2 2

' against alright angel anyone breath

1 1 1 1 1 1

cause child darling dive dream dress

1 1 1 1 1 1

faith fall fight follow future give

1 1 1 1 1 1

hand hear heart home hope kiss

1 1 1 1 1 1

lead man meet mess mine more

1 1 1 1 1 1

never now odd of person right

1 1 1 1 1 1

say secret slow someday someone still

1 1 1 1 1 1

stronger sweet underneath up wait whisper

1 1 1 1 1 1

woman

1

**Exercise**

Take the lyrics of any other artist and make a frequency list of lemmas. Note: you might need to add the punctuation marks in order to separate the sentences!