NPFL142.C4DHI – tutorial #3
n-grams with *Migrant stories* and UDPipe on *Andersen* tales

Exercises with *Migrant stories* dataset.. 2
Exercise 1.1 – Loading the data set.. 2
Exercise 1.2 – Create a unigram frequency dictionary of women who left Afghanistan...............2
Exercise 1.3 – Create a bigram frequency dictionary of women who left Afghanistan................ 2
Exercise 1.4 – Visualize both frequency dictionaries using barplots...3
Exercises with *Andersen* dataset..5
Exercise 2.1 – Loading the data set.. 5
Exercise 2.2 – Process "Aunty" tale using UDPipe.. 6
Exercise 2.3 – Create a lemma frequency dictionary of "Aunty"...7
Exercise 2.4 – Compare the lengths of the language versions of the books..................................7

Exercises with *Migrant stories* dataset

Exercise 1.1 – Loading the data set

In RStudio create a blank R script

- Move to Files&Plots desktop
- In Files manager
    - move to the home directory
    - create a new directory New folder > `3`
    - move to the directory `3` and create a new directory New folder > `migrants`
    - move to the directory `3/migrants`
    - use More in the menu to run Set as working directory
    - use New Blank File in the menu to create a blank R Script and name it `migrants.t.R`. Then the script is open in the Code editor window (upper-left window) and you can add the `commands` listed below to the script.

We suppose using these packages

```
library(tidyverse)
library(tidytext)
library(gridExtra)
```

Load the *Migrant stories* dataset into your R environment.

```
dataset <- read_tsv("dataset <- read_tsv("../../1/migrants/migrants.tsv")
names(dataset)   # attributes
```

Exercise 1.2 – Create a unigram frequency dictionary of women who left Afghanistan

We will prioritize words that appear in the given stories more than 5 times to ensure a barplot is clear.

```
uni <- dataset %>%
  filter(gender == "female")  %>%          # females only
  filter(country_or == "Afghanistan") %>%  # Afghanistan as an origin country
  unnest_tokens(word, story)  %>%          # tokenize stories, i.e. extract unigrams
  anti_join(stop_words) %>%                # exclude stop words
  count(word, sort=TRUE)   %>%             # count word frequencies
  filter(n > 5) %>%                        # filter out the words used more than 5 times
  mutate(word = reorder(word, n))          # sort by n
```

Exercise 1.3 – Create a bigram frequency dictionary of women who left Afghanistan

For extracting bigrams we use the `unnest_tokens` function, which we used for tokenization. The difference is that the tokens will not be individual words but n-grams (`token = "ngrams"`), specifically bigrams (`n=2`).

```
bi.1 <- dataset %>%
  filter(gender == "female")  %>%                        # females only
  filter(country_or == "Afghanistan") %>%               # Afghanistan as an origin country
  unnest_tokens(bigram, story, token = "ngrams", n = 2) # extract bigrams
```

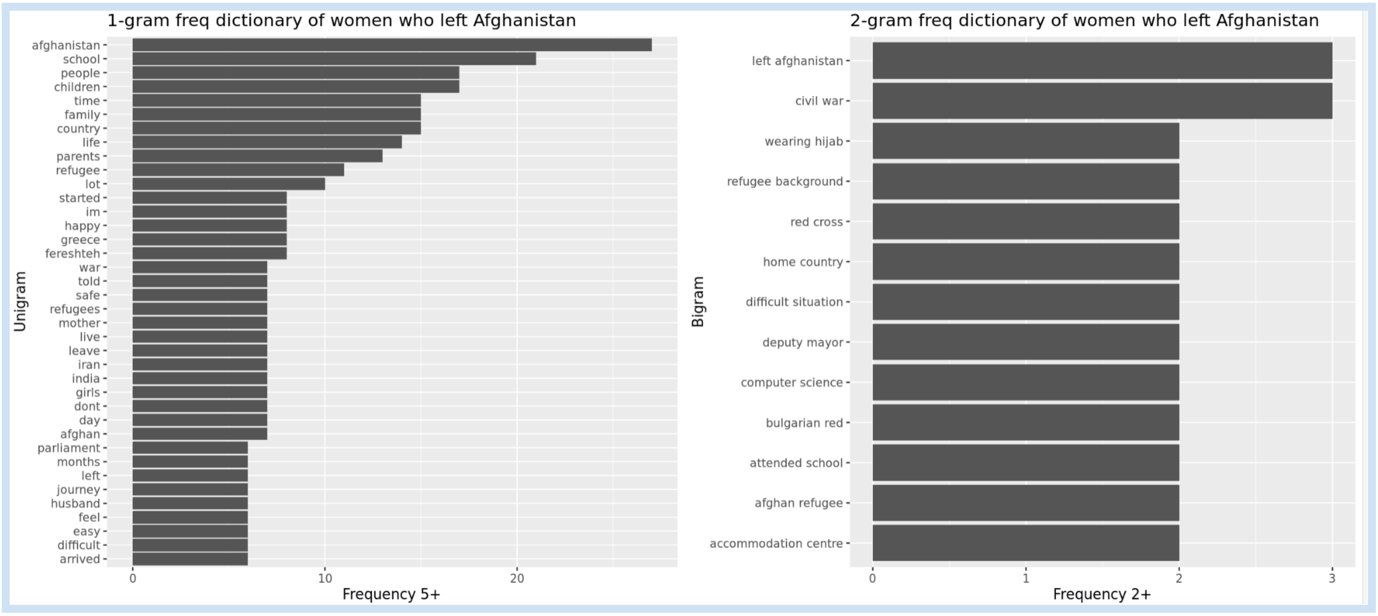
Bigrams of stop words are unimportant for understanding the content of the story. Therefore, we filter them out. We will prioritize bigrams that appear in the given stories more than once to ensure a barplot is clear.

```
bi <- bi.1 %>%
  separate(bigram, into = c("w1", "w2"), sep = " ")  %>% # split bigram into w1 and w2
  filter(!w1 %in% stop_words$word,           # filter out bigrams with stop words
         !w2 %in% stop_words$word) %>%
  unite(bigram, c(w1, w2), sep = " ")  %>%   # join w1 and w2 into bigram
  count(bigram, sort = TRUE) %>%             # count bigram frequencies
  filter(n > 1) %>%                          # filter out the bigrams used more than once
  mutate(bigram = reorder(bigram, n))        # sort by n
```

Exercise 1.4 – Visualize both frequency dictionaries using barplots

Draw barplots showing unigram and bigram frequencies and make them side by side (see `grid.arrange` from `gridExtra` package).

```
plot.uni <- ggplot(uni, aes(word, n)) +     # create a chart showing unigram frequencies
  geom_col() +
  xlab("Unigram") + ylab("Frequency 5+") + coord_flip() +
  ggtitle("1-gram freq dictionary of women who left Afghanistan")


plot.bi <- ggplot(bi, aes(bigram, n)) +     # create a chart showing bigram frequencies
  geom_col() +
  xlab("Bigram") + ylab("Frequency 2+") + coord_flip() +
  ggtitle("2-gram freq dictionary of women who left Afghanistan")

grid.arrange(plot.uni, plot.bi, ncol = 2)   # draw the charts side by side
```



1-gram freq dictionary of women who left Afghanistan
Unigram
afghanistan
school
people
children
time
family
country
life
parents
refugee
lot
started
im
happy
greece
fereshteh
war
told
safe
refugees
mother
live
leave
iran
india
girls
dont
day
afghan
parliament
months
left
journey
husband
feel
easy
difficult
arrived
0
10
20
Frequency 5+
2-gram freq dictionary of women who left Afghanistan
Bigram
left afghanistan
civil war
wearing hijab
refugee background
red cross
home country
difficult situation
deputy mayor
computer science
bulgarian red
attended school
afghan refugee
accommodation centre
0
1
2
3
Frequency 2+

Exercises with *Andersen* dataset

Exercise 2.1 – Loading the data set

In RStudio create a blank R script

- Move to Files&Plots desktop
- In Files manager
    - move to the directory `3` and create a new directory New folder > `andersen`
    - move to the directory `3/andersen`
    - use More in the menu to run Set as working directory
    - use New Blank File in the menu to create a blank R Script and name it `andersen.t.R`. Then the script is open in the Code editor window (upper-left window) and you can add the `commands` listed below to the script.

We suppose using these packages

```
library(tidyverse)
library(tidytext)
```

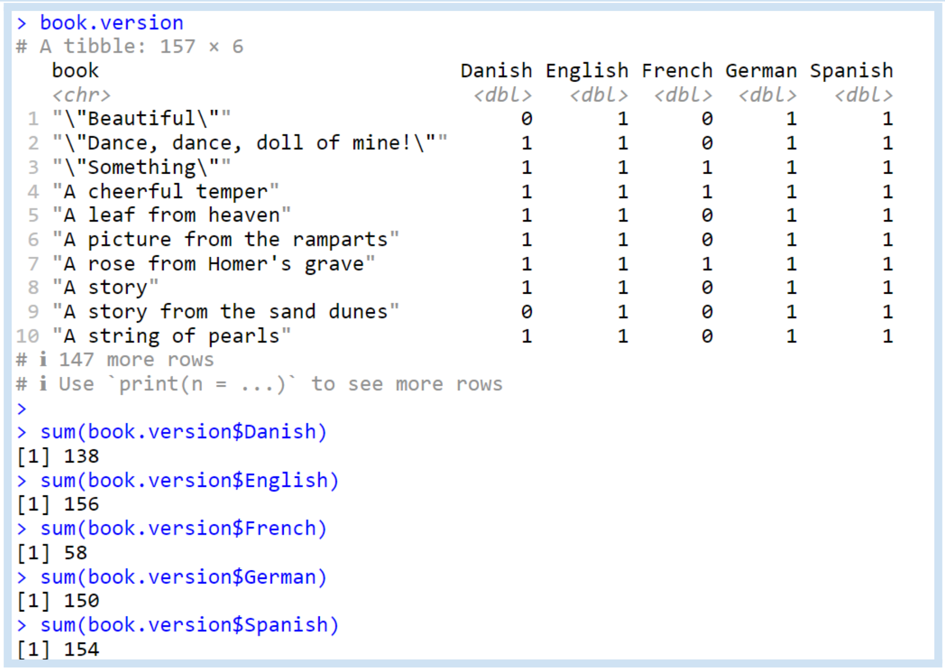
Fairy tales written by Hans Christian Andersen are available in the `hcandersenr` package.

```
library(hcandersenr)
```

It is a multilingual dataset. The books are formatted to be convenient for text analysis. The UTF-8 plain texts were sourced from http://www.andersenstories.com/ and are divided into elements of up to about 80 characters each (see the `text` attribute below).

For each fairy tale, we want to determine the languages in which it is available. `hca_fairytales` is a table with three attributes: `text`, `book` and `language`. We will focus on `book` and `text` only. We will remove duplicate rows and create a new binary attribute `present`, which will be set to 1 for each book-language pair. Then we will transform this table so that the number of rows corresponds to the number of books in the dataset, and the number of columns will be six: `book` (title), `Danish`, `English`, `French`, `German` and `Spanish`. The values in the cells will be either 0 or 1, depending on whether the respective book is available in the given language, ensuring that each row corresponds to a single book.

```
book.version <- hca_fairytales() %>%
  select(book, language) %>%            # select the book names and languages
  unique()  %>%                         # remove duplicate rows
  mutate(present = 1) %>%               # add attribute 'present'
  spread(language, present, fill = 0)

book.version

sum(book.version$Danish)
sum(book.version$English)
sum(book.version$French)
sum(book.version$German)
sum(book.version$Spanish)
```



```
> book.version
# A tibble: 157 × 6
   book                                Danish English French German Spanish
   <chr>                                <dbl>   <dbl>  <dbl>  <dbl>   <dbl>
 1 "\"Beautiful\""                          0       1      0      1       1
 2 "\"Dance, dance, doll of mine!\""        1       1      0      1       1
 3 "\"Something\""                          1       1      1      1       1
 4 "A cheerful temper"                      1       1      1      1       1
 5 "A leaf from heaven"                     1       1      0      1       1
 6 "A picture from the ramparts"            1       1      0      1       1
 7 "A rose from Homer's grave"              1       1      1      1       1
 8 "A story"                                1       1      0      1       1
 9 "A story from the sand dunes"            0       1      0      1       1
10 "A string of pearls"                     1       1      0      1       1
# ℹ 147 more rows
# ℹ Use `print(n = ...)` to see more rows
>
> sum(book.version$Danish)
[1] 138
> sum(book.version$English)
[1] 156
> sum(book.version$French)
[1] 58
> sum(book.version$German)
[1] 150
> sum(book.version$Spanish)
[1] 154
```

View metadata.

```
print(EK)
```

Exercise 2.2 – Process "Aunty" tale using UDPipe

Read the tale in English.

```
dataset <- hcandersen_en %>%
  filter(book == "Aunty")
```

UDPipe is a tool that provides tokenization, parts of speech tagging, lemmatization and syntactic parsing of raw text. It exists as a third-party R CRAN package `udpipe`.

```
library(udpipe)
```

Create a data frame because the **`udpipe`** function operates on a dataframe with columns **`doc_id`** and **`text`**. Therefore we are naming the attributes accordingly.

```
book.text <- data.frame(doc_id = dataset$book, text = dataset$text)
```

Read the English UDPipe model

```
ud_model <- udpipe_download_model(language = "english")
ud_model <- udpipe_load_model(ud_model$file_model)
```

Run UDPipe on **`book.text`**

```
book.text.ud <- udpipe(book.text, ud_model)
```

```
names(book.text.ud)
```

Exercise 2.3 – Create a lemma frequency dictionary of "Aunty"

```
book.text.ud %>%
  filter(!(lemma %in% stop_words$word)) %>%
  count(lemma, sort=TRUE) %>%
  head(n=10)                              # display 10 most frequent lemmas
```

Exercise 2.4 – Compare the lengths of the language versions of the books

We will focus on books that are available in the dataset in all five languages. First, we extract a list of such books. Subsequently, we process each of them using UDPipe and count the number of tokens.

```
book.lang.all <- book.version %>%
  filter (Danish == 1 & English == 1 & French == 1 & German == 1 & Spanish == 1) %>%
  pull(book)
```

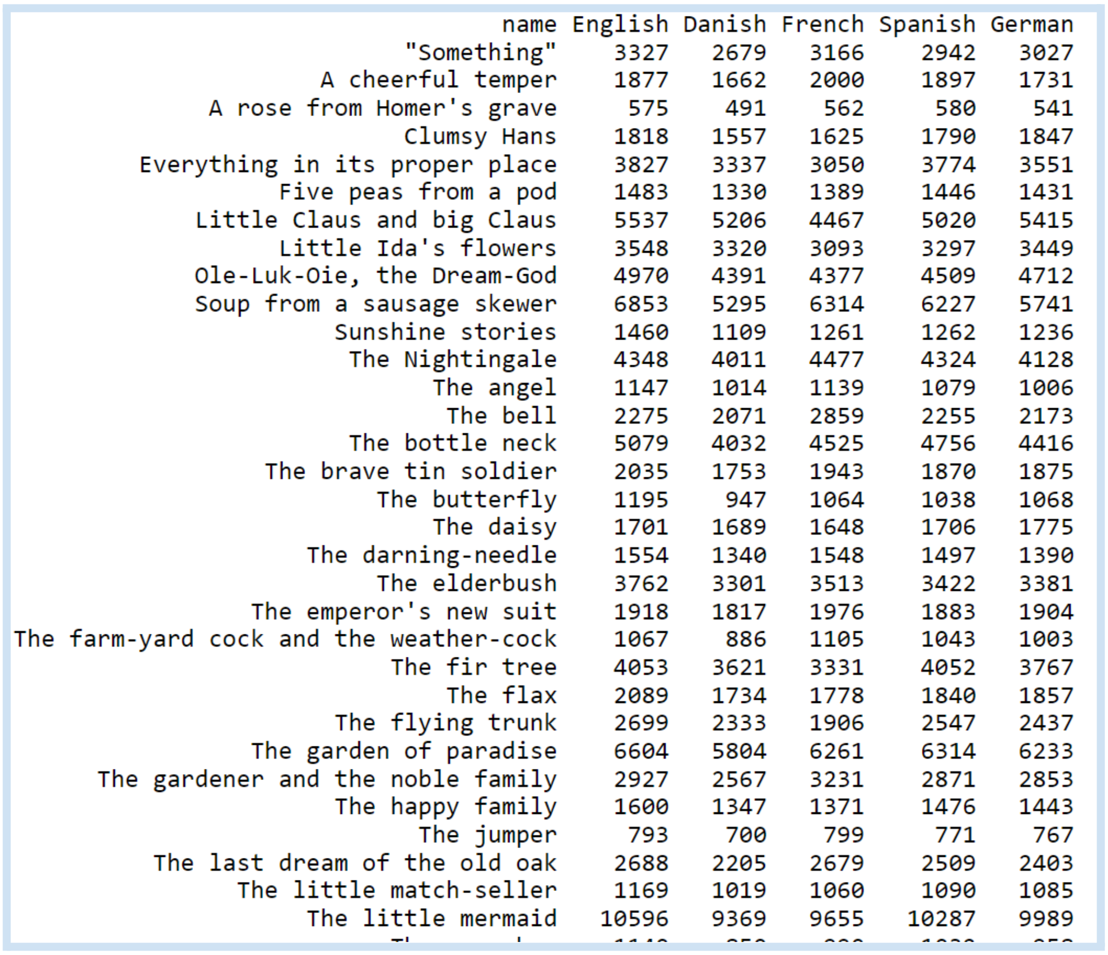
Extract the books and process them using UDPipe: we have a list of English book names (`book.lang.all`), but the `hcandersen_[da|es|fr|de]` collections contain book names in their respective languages. Therefore, we will use metadata (`EK`) to match English names with names in other languages (see `left_join` below). The `udpipe` function operates on a dataframe with columns `doc_id` and `text`. so in each collection, we are renaming the attributes accordingly (see `set_names` below).

```
# English
en.text.ud <- hcandersen_en %>%
  filter(book %in% book.lang.all) %>%
  set_names(c("text","doc_id")) %>%                  # rename the attributes, see ?udpipe
  udpipe(ud_model)

# Danish
ud_model <- udpipe_download_model(language = "danish")
ud_model <- udpipe_load_model(ud_model$file_model)
da.text.ud <- hcandersen_da %>%                      # it contains Danish names
  left_join(EK, by = c("book" = "name_da")) %>%
  filter(name_en %in% book.lang.all) %>%
  select(text, name_en)  %>%
  set_names(c("text","doc_id")) %>%
  udpipe(ud_model)

# French
ud_model <- udpipe_download_model(language = "french")
ud_model <- udpipe_load_model(ud_model$file_model)
fr.text.ud <- hcandersen_fr %>%                      # it contains French names
  left_join(EK, by = c("book" = "name_fr")) %>%
  filter(name_en %in% book.lang.all) %>%
  select(text, name_en)  %>%
  set_names(c("text","doc_id")) %>%
  udpipe(ud_model)
```

```
# Spanish
ud_model <- udpipe_download_model(language = "spanish")
ud_model <- udpipe_load_model(ud_model$file_model)
es.text.ud <- hcandersen_es %>%                      # it contains Spanish names
  left_join(EK, by = c("book" = "name_es")) %>%
  filter(name_en %in% book.lang.all) %>%
  select(text, name_en)  %>%
  set_names(c("text","doc_id")) %>%
  udpipe(ud_model)

# German
ud_model <- udpipe_download_model(language = "german")
ud_model <- udpipe_load_model(ud_model$file_model)
de.text.ud <- hcandersen_de %>%
  left_join(EK, by = c("book" = "name_de")) %>%      # it contains German names
  filter(name_en %in% book.lang.all)%>%
  select(text, name_en)  %>%
  set_names(c("text","doc_id")) %>%
  udpipe(ud_model)
```

Get the length of the books where the length is the number of tokens.

```
en.c <- en.text.ud %>%
    count(doc_id)
da.c <- da.text.ud %>%
  count(doc_id)
fr.c <- fr.text.ud %>%
  count(doc_id)
es.c <- es.text.ud %>%
  count(doc_id)
de.c <- de.text.ud %>%
  count(doc_id)
en.c %>%
  left_join(da.c, by = "doc_id") %>%
  left_join(fr.c, by = "doc_id") %>%
  left_join(es.c, by = "doc_id") %>%
  left_join(de.c, by = "doc_id") %>%
  set_names(c("name", "English", "Danish", "French", "Spanish", "German"))
```



```
                                 name English Danish French Spanish German
                          "Something"    3327   2679   3166    2942   3027
                     A cheerful temper    1877   1662   2000    1897   1731
               A rose from Homer's grave     575    491    562     580    541
                          Clumsy Hans    1818   1557   1625    1790   1847
           Everything in its proper place    3827   3337   3050    3774   3551
                     Five peas from a pod    1483   1330   1389    1446   1431
              Little Claus and big Claus    5537   5206   4467    5020   5415
                    Little Ida's flowers    3548   3320   3093    3297   3449
              Ole-Luk-Oie, the Dream-God    4970   4391   4377    4509   4712
              Soup from a sausage skewer    6853   5295   6314    6227   5741
                        Sunshine stories    1460   1109   1261    1262   1236
                         The Nightingale    4348   4011   4477    4324   4128
                               The angel    1147   1014   1139    1079   1006
                                The bell    2275   2071   2859    2255   2173
                         The bottle neck    5079   4032   4525    4756   4416
                    The brave tin soldier    2035   1753   1943    1870   1875
                           The butterfly    1195    947   1064    1038   1068
                               The daisy    1701   1689   1648    1706   1775
                     The darning-needle    1554   1340   1548    1497   1390
                           The elderbush    3762   3301   3513    3422   3381
                   The emperor's new suit    1918   1817   1976    1883   1904
The farm-yard cock and the weather-cock    1067    886   1105    1043   1003
                            The fir tree    4053   3621   3331    4052   3767
                                The flax    2089   1734   1778    1840   1857
                        The flying trunk    2699   2333   1906    2547   2437
                   The garden of paradise    6604   5804   6261    6314   6233
          The gardener and the noble family    2927   2567   3231    2871   2853
                        The happy family    1600   1347   1371    1476   1443
                              The jumper     793    700    799     771    767
            The last dream of the old oak    2688   2205   2679    2509   2403
                  The little match-seller    1169   1019   1060    1090   1085
                       The little mermaid   10596   9369   9655   10287   9989
```