ORCmyPDF with Bash and Tidytextmining with Rstudio

By: Markus Oliver Nielsen, 201907052
Characters w. Learning Journal: 24707

Abstract.

When working with digitized data you sometimes run into problems, when you want to do a textmining analysis of the data because the data you are working with have not been optical character recognized (ORC). There is a number of ways to have pdf-documents optical character recognized. Most of these are explained in depth in a number of places on the internet. But how do you go from having a number of pdf-files without OCR, to have a finished textmining analysis? Combining the methods from ORCmyPDF and tidytextmining – textmining with R – I show how to get from A to Z in this process, by using data from a CIA archive.

Keywords:

CIA and Doctor Zhivago; OCRmyPDF; Tidytextmining;

A - Free text

1 Introduction / Goal & 2 Problems and Background

When working with my "kildenært emne"-exam I ran into problems when I had to work with 98 pdf-documents from a CIA archive. These documents were not Optical Caracter Recognized (OCR), which made it hard to search for specific topics in this rather large corpus of documents. From the subjects covered in this class I developed a curiosity towards textmining and the potentials it promises. When doing textmining you have to get you pdf-documents OCR'ed in order for the program to be able to read the text. This made me interested in getting the 98 pdf-documents character recognized, to give me a greater range of possibilities when working with the data hidden in the texts. This project gives others a guide to how one can go from pdf without character recognition to a preliminary textmining analysis.

3 Software Framework

This project was performed on my Macbook air (Retina, 13-inch, 2019), which runs OS Big Sur version 11.0.1. The processor is 1,6 GHz Dual-Core Intel Core i5, it has 8 GB ram, and the graphics card is an Intel UHD Graphics 617 1536 MB. To do OCR on the pdf-documents I needed to install bash (v3.2) and OCRmyPDF (v.11.3.3). To install the OCRmyPDF package I needed to install Homebrew (v2.5.11) a package manager for bash. To be able to do the textmining for this project I needed to install R (v4.0.3) and RStudio (v1.3.1093). For my textmining project I needed the following packages: Tidytext (v.0.2.6), tidyverse, (v.1.3.0), Readtext (v.0.80), textdata (v.0.4.1), Lubridate (v.1.7.9.2).

4 Data Acquisition and Processing

I used data from CIA's internet archive describing CIA's role in the publication of the book Doctor Zhivago by the Russian author Boris Pasternak[16]. I published this data in my GitHub repository[17], According to CIA's website, the data available on their website is part of the public domain and I can therefore publish these on my GitHub repository[18].

The OCR was performed with the OCRmyPDF software in my terminal with bash. I followed the guide from OCRmyPDF's official website[19]. To perform the OCR in the most effective manner I wrote a bash script for a forloop to perform the OCR on all pdf-documents in a given folder.[20]

For the text mining in R I used a number of packages which all had a specific purpose for the analysis. For ease of use and because of the well documented nature of the tidy packages I chose do use *tidyverse[21] (v.1.3.0)* and *tidytext[22] (v.0.2.6)* for my text mining. I followed the textmining-guide that's available online for tidytext[23]. For the import of the text from the pdf-files to R I used *readtext[24] (v.0.80)*. To download the sentiment lexicon for my sentiment analysis I used the *textdata[25]*-package *(v.0.4.1)*. To convert the "date" into date-format I used *lubridate[26] (v.1.7.9.2)*. With this package I used the mutate function to convert the "date"-datapoint into date-format so that I could sort by month and filter for year 1958. I realize after completing this project that lubridate is a part of the tidyverse package, so I did not have to load this package separately.

[16] "Doctor Zhivago", CIA, accessed January 1, 2021, https://web.archive.org/web/20201212131158/https://www.cia.gov/library/readingroom/collection/doctor-zhivago
[17] "CIA_sources_original", GitHub, accessed December 30, 2020, https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/tree/main/CIA_sources_orginal
[18] "Site Policies", CIA, accessed December 30, 2020, https://www.cia.gov/site-policies/
[19]"OCRmyPDF documentation", OCRmyPDF, accessed December 30, 2020, https://ocrmypdf.readthedocs.io/en/latest/index.html
[20]"CIA_script.bsh.txt", GitHub, accessed December 30, 2020, https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/blob/main/CIA_doctorzhivago_textmining_and_OCR/CIA_script.bsh.txt
[21] "tidyverse", Cran, accessed January 2, 2021,*https://cran.r-project.org/web/packages/tidyverse/index.html*
[22] "tidytext", GitHub, accessed December 30, 2020, *https://github.com/juliasilge/tidytext*
[23]"Welcome to text mining with R", Text mining with R, accessed December 30, 2020, https://www.tidytextmining.com/
[24]"Package ´readtext`", Cran, accessed December 30, 2020, *https://cran.r-project.org/web/packages/readtext/readtext.pdf, accessed by 30/12/2020*
[25] "textdata", Cran, accessed December 30, 2020,*https://cran.r-project.org/web/packages/textdata/readme/README.html*
[26]"Package ´lubridate`", Cran, accessed December 30, 2020, *https://cran.r-project.org/web/packages/lubridate/lubridate.pdf*

5 Implementation and Empirical Results

My forloop script was written to be used in the terminal with bash and OCRmyPDF. To use this script, you need to download OCRmyPDF in advance. The goal for the forloop was to perform the OCRmyPDF task on all pdf-files in a given folder and place these in a separate folder. In my case I wanted the files to end up in a folder called "CIA_sources_output_ocr". The script only works if you are located in the folder where the pdf-files are located.

I wrote the echo command into the script to get some visual input after engaging the script. By applying the echo command to the script, the terminal prints the names of the files it processes. Underneath here I show the working part of the script. The script includes a short explanation of the script as well, disengaged by #. I included the explanation to make it easier for my future self and others to understand the script. This is to comply with the FAIR principles[27].

——————————— script start ———————————

```
for pdf in *.pdf
           do
                      echo "$pdf"
                      ocrmypdf "$pdf" ../CIA_sources_output_ocr/$pdf
done
```

——————————— script end ———————————

With the files OCR'ed I went on to include metadata about the files into the name of the files. This was done manually. I don't see any way of automating this process. The files with OCR and metadata in the name are available on my GitHub repository[28].

I present here a part of my Rmarkdown script. I explain the different steps as I go through the script. It is accessible in its full length from my GitHub repository[29]. I chose to show the part where I load the data and analyses it with term frequency, inverse document frequency and tf-ifd. I chose to present this part of the Rmarkdown script because I found this bit to be the most relevant for my dataset. I left out the part where I generate a column plot and a word cloud with the most common words, and where I make a sentiment analysis and visualize it with a comparison cloud. Whilst this is very much a relevant part of my Rmarkdown, it is a long script and I had to compromise to stay within the boundaries of this assignment.

---

[27] Mark D. Wilkinson et al., "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data* 3, no. 1 (2016).
[28] "CIA_sources_OCR_wMetadata" GitHub, accessed January 4, 2021, https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/tree/main/CIA_sources_OCR_wMetadata
[29] "CIA_test.Rmd", GitHub, accessed December 30, 2020, https://github.com/Digital-Methods-HASS/au_615046_niel-sen_markus/blob/main/CIA_doctorzhivago_textmining_and_OCR/CIA_test.Rmd

——————————— script start ———————————

~~~
# Loading R-packages
To make this project possible I need to load a couple of R-packages. These in-
clude:
```{r message=FALSE}
library(readtext)
library(tidyverse)
library(tidytext)
library(textdata)
library(lubridate)
~~~
With the packages loaded I can proceed with my work

# Loading data
I made a folder om my computer wherein I placed the data that I want to work
with. I place these in a folder called data
I want to load the data that I put in a directory called data.
I ask the readtext function to make a dataset called CIA with the data from the
data directory.

```{r}
CIA <- as_tibble(readtext("./data/*.pdf",
                docvarsfrom = "filenames",
                docvarnames = c("date", "from", "to","subject"),
                dvsep = "_",
                encoding = "UTF-8"))
```

I have put metadata about the documents in the filename. It's structured like
this '"date"_"from"_"to"_"subject".pdf'. An example of this is "1957-12-
12_cia_cia_pasternak.pdf". In the courpus I used 4 different values for the
"from" section: cia, sr, sr2, na. I used 5 different values for the "to" sec-
tion: cia, sr, sr2, poor, nr. I used 5 different values for the "subject" sec-
tion: sr, feltrinelli, mouton, pasternak, na.

Explanation of the values:
  from/to:
    cia = Central Intellegence Agency
    sr = Soviet Russia Division/AEDinasaure
    sr2 = Soviet Russia Division 2
    poor = Henry Poor, Counselor of law
    na = data no available

  Subject:
    sr = regarding AEDinasaure
    feltrinelli = regarding Feltrinelli and copyrights for "Doctor Zhivago", by
Boris Pasternak
    mouton = Regarding the 1958 mouton edition of "Doctor Zhivago"
    pasternak = Regarding "Doctor Zhivago" or Boris Pasternak unspecified
    na = Data not available

I tell the readtext function that I want det files sorted by the filename and
that I separated the different data by a underscore "_" in the filenames, with
the "dvsep =" function.
I use the UTF-8 encoding to make sure that the program will understand all the
characters. Without this encoding that program might not understand special
characters
```
```

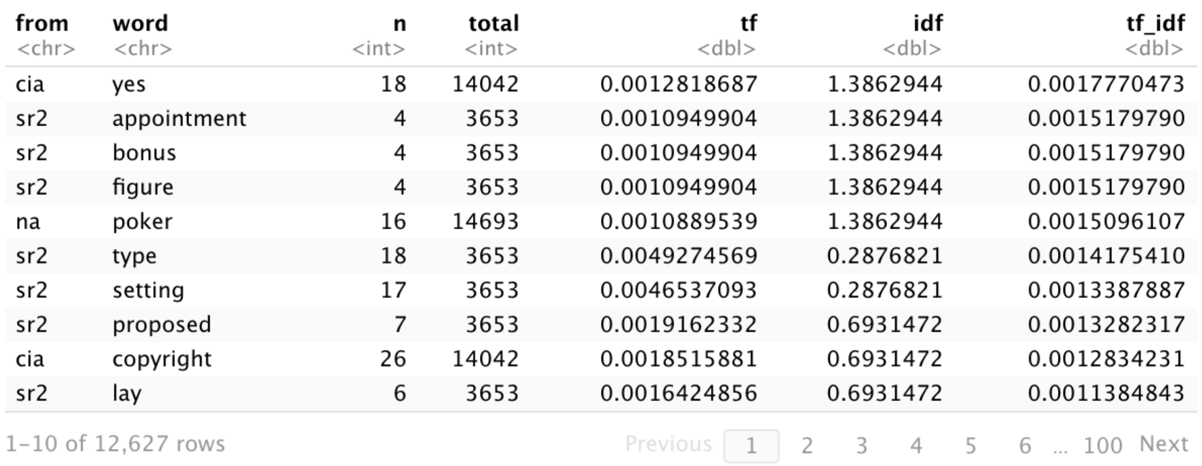
~~~
# Analysis

To begin my analysis, I need to make sure that my data is in a tidydata-format.
For this I use the unnest_tokens-function. This function makes my dataset into a
tidydata-format which breaks the text into single words and removes capital let-
ters, among other things.
I want to remove all numbers from my text since I don't need them for my analy-
sis. I do this with the mutate-function, by using the stringr-package replace
all function. This allows me to remove all numbers with a regex function.
I remove stopwords by using the stopwordlist from tidytext.

```{r}
CIA_clean <- CIA %>%
  mutate(text =
           str_replace_all(text,
                           pattern =
                            "\\d", "")) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(doc_id, word, sort = TRUE)

CIA_clean
```

## Term Frequency and Inverse Document Frequency

(...)[30]

In order to calculate the total number of words I need to use a tibble on which
there hasn't been used a stopwordlist. Therefore, I use the CIA data and use the
same functions as I did on the CIA_clean, but without that stopwordlist. I call
this CIA_ftw (ftw = for total words).

I calculate the total number of words and by assigning the sum of "n" catego-
rized by "from" (sender) to "total".

```{r}
CIA_ftw <- CIA %>%
  mutate(text =
           str_replace_all(text,
                           pattern =
                            "\\d", "")) %>%
  unnest_tokens(word, text) %>%
  count(from, word, sort=TRUE) %>%
  ungroup()


total_words <- CIA_ftw %>%
  group_by(from) %>%
  summarize(total=sum(n))

```

I then want to use the tf-idf function on the dataset.
~~~

[30] In the Script I explain what term frequency and inverse document frequency is. Because I'm a little short on space I cut this part out.

First, I need to combine my CIA_ftw with the total number of words that I calculated earlier. I do this by using the left_join function. I call this tibble CIA_tw (tw = total words)

I can now use the function from the tidytext package called bind_tf_idf. This makes all the necessary calculations, and prints a tibble with tf, idf and tf_idf.

~~~
```{r}
CIA_tw <- left_join(CIA_ftw, total_words)

CIA_tw %>%
  bind_tf_idf(word, from, n) %>%
  arrange(desc(tf_idf))
```
~~~



from <chr>	word <chr>	n <int>	total <int>	tf <dbl>	idf <dbl>	tf_idf <dbl>
cia	yes	18	14042	0.0012818687	1.3862944	0.0017770473
sr2	appointment	4	3653	0.0010949904	1.3862944	0.0015179790
sr2	bonus	4	3653	0.0010949904	1.3862944	0.0015179790
sr2	figure	4	3653	0.0010949904	1.3862944	0.0015179790
na	poker	16	14693	0.0010889539	1.3862944	0.0015096107
sr2	type	18	3653	0.0049274569	0.2876821	0.0014175410
sr2	setting	17	3653	0.0046537093	0.2876821	0.0013387887
sr2	proposed	7	3653	0.0019162332	0.6931472	0.0013282317
cia	copyright	26	14042	0.0018515881	0.6931472	0.0012834231
sr2	lay	6	3653	0.0016424856	0.6931472	0.0011384843

1–10 of 12,627 rows

With this function I can gather a number of informations. But in order to make the most use of it I need to filter the data to narrow my search.

### Feltrinelli
I can for example choose to filter by subject and look at what words are least common in the documents concerning Feltrinelli.

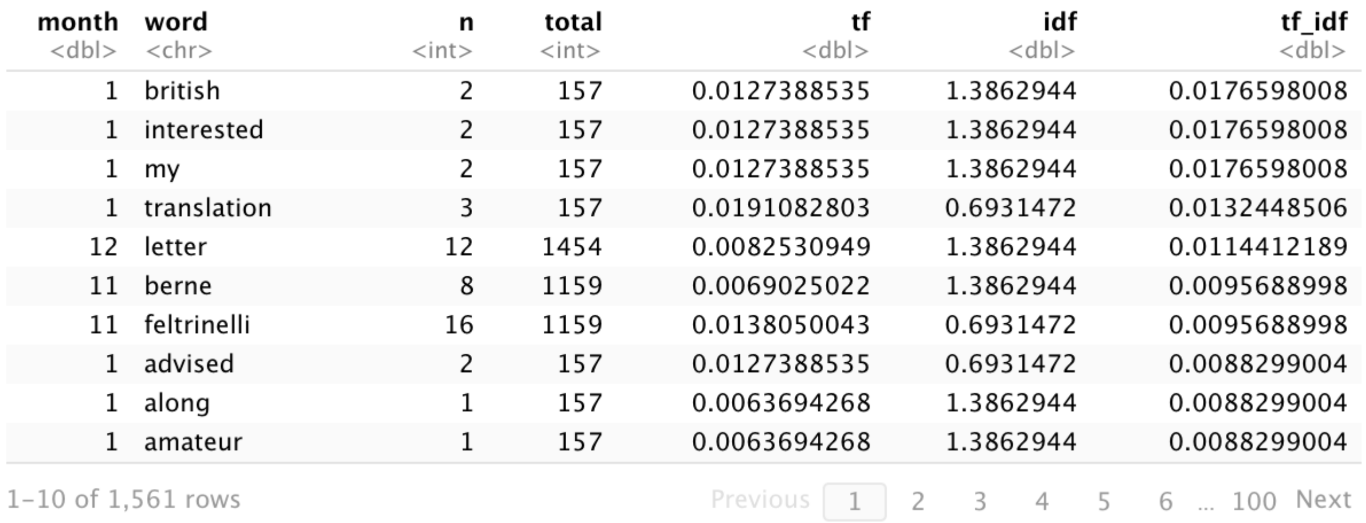
First, I filter all the documents that have the subject "feltrinelli"

~~~
```{r}
CIA %>%
  filter(subject == "feltrinelli") -> CIA_feltrinelli
```
~~~

Then I clean the data of numbers and filter to show only documents from 1958.

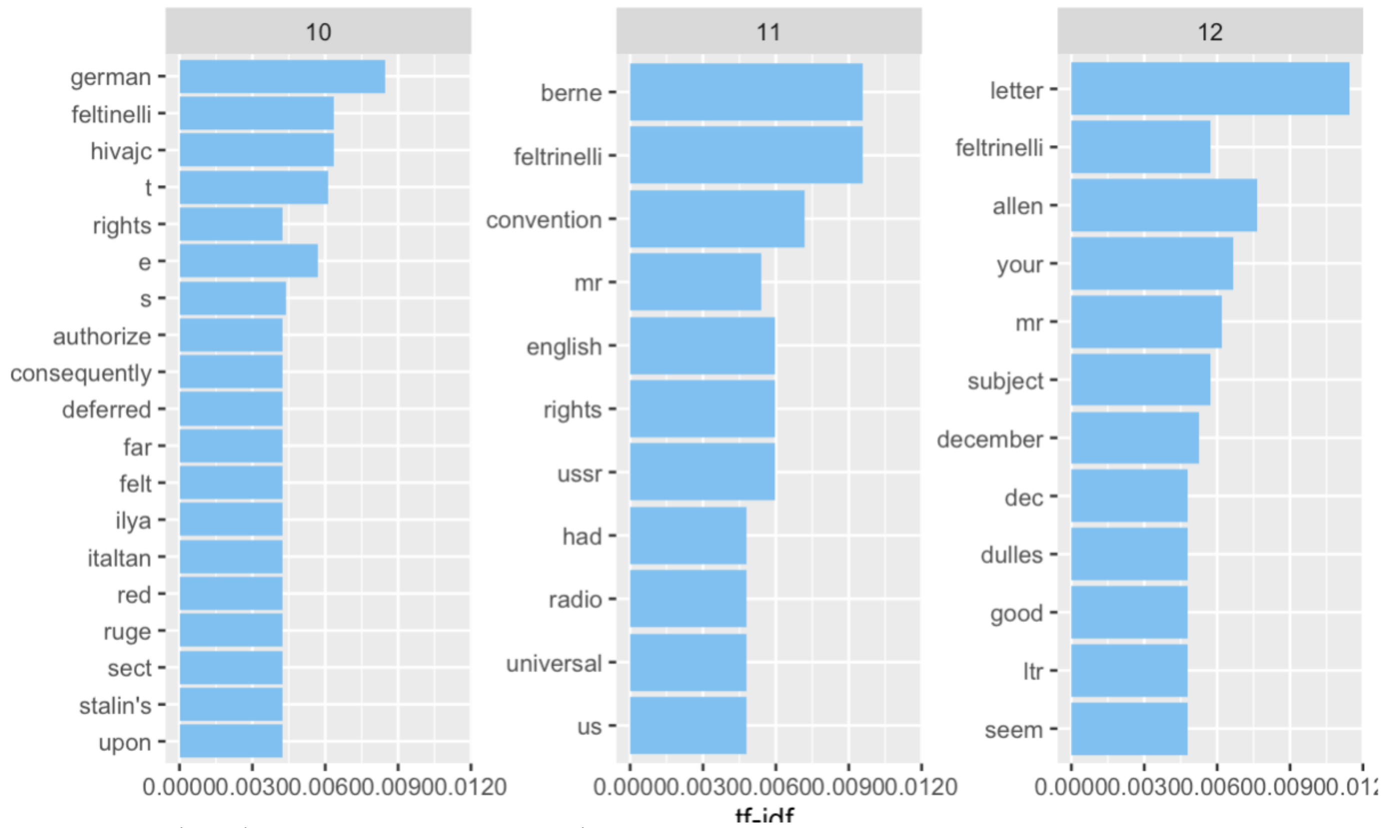
~~~
```{r}
CIA_feltrinelli %>%
   mutate(text =
           str_replace_all(text,
                           pattern =
                            "\\d", "")) %>%
  mutate(month = month(date)) %>%
  mutate(year = year(date)) %>%
  filter(year == "1958")-> CIA_feltrinelli
```
~~~

With the filtered data I can use the tf_idf function to show the least common words in the texts concerning "feltrinelli", sorted by month.

~~~
```{r}
total_words_feltrinelli <- CIA_feltrinelli %>%
  unnest_tokens(word, text) %>%
  count(month, word) %>%
  group_by(month) %>%
  summarize(total=sum(n))


CIA_feltrinelli_tw <- CIA_feltrinelli %>%
   unnest_tokens(word, text) %>%
  count(month, word) %>%
  left_join(total_words_feltrinelli, "month")

CIA_feltrinelli_tw %>%
  bind_tf_idf(word, month, n) %>%
  arrange(desc(tf_idf))
~~~
```



| month <dbl> | word <chr> | n <int> | total <int> | tf <dbl> | idf <dbl> | tf_idf <dbl> |
|---|---|---|---|---|---|---|
| 1 | british | 2 | 157 | 0.0127388535 | 1.3862944 | 0.0176598008 |
| 1 | interested | 2 | 157 | 0.0127388535 | 1.3862944 | 0.0176598008 |
| 1 | my | 2 | 157 | 0.0127388535 | 1.3862944 | 0.0176598008 |
| 1 | translation | 3 | 157 | 0.0191082803 | 0.6931472 | 0.0132448506 |
| 12 | letter | 12 | 1454 | 0.0082530949 | 1.3862944 | 0.0114412189 |
| 11 | berne | 8 | 1159 | 0.0069025022 | 1.3862944 | 0.0095688998 |
| 11 | feltrinelli | 16 | 1159 | 0.0138050043 | 0.6931472 | 0.0095688998 |
| 1 | advised | 2 | 157 | 0.0127388535 | 0.6931472 | 0.0088299004 |
| 1 | along | 1 | 157 | 0.0063694268 | 1.3862944 | 0.0088299004 |
| 1 | amateur | 1 | 157 | 0.0063694268 | 1.3862944 | 0.0088299004 |

1–10 of 1,561 rows Previous 1 2 3 4 5 6 ... 100 Next

```
Visualization

I can visualize the data that I made with the feltrinelli subject from 1958
sorted by months.

I filter out the month of January, since it's a very small datapoint and there-
fore skews my plot.
```{r}
CIA_feltrinelli_tw %>%
  bind_tf_idf(word, month, n) %>%
  arrange(desc(tf_idf)) %>%
  filter(month != 1) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(month) %>%
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(word, tf_idf)) +
  geom_col(show.legend = FALSE, fill = "skyblue2") +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~month, ncol = 3, scales = "free_y") +
  scale_y_continuous(labels = scales::comma_format(accuracy = 0.0001)) +
  coord_flip()
```
```
```



```
I can do this with any number of subjects, dates, etc..
```

———————————— script end ————————————

This script can be found in html-format in my GitHub repository[31].

6 Critical evaluation

Working with this project proved to be rather complicated due to the data I chose to work with. The pdf-files that I got off CIA's website were bad scans from machine written documents with handwritings on them. This made it very hard for the OCR software to recognize the text. This made my options for working with the data somewhat limited. I wasn't able to do the same extensive analysis and comparison as e.g. that of Julia Silge in *Text Mining with R*[32]. To do this I would have had to spend a very long time correcting and setting up the text before I started the text mining. The CIA archive from which I got the data for the analysis was by all means a bad digital archive. To prove my point, I would like to point out the fact that on January 4th the website was updated making the data inaccessible for about 3 days. The URL that I used during all the time working on this project now returns a 404-fail message. The website I used to collect the data for this project is now only

[31]"CIA_test.html", GitHub, accessed December 30, 2020, https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/blob/main/CIA_doctorzhivago_textmining_and_OCR/CIA_test.html
[32]"The tidy text format", Text Mining with R, accessed December 30, 2020, https://www.tidytextmining.com/tidytext.html

accessible via WaybackMachine. I will therefore advise anyone trying to reproduce my work to collect the data off my GitHub repository[33]. The data is available from the website again, but under a different site-page[34].

I think the digital tools available to me were easy to use once you understood the syntax. The OCRmyPDF tool was by far the simplest of them all. The trouble I encountered when working with it was due to bash forloop troubles, not the tool itself. The R language was made accessible by the Rstudio desktop program. This made it easy to understand and work with R.

In working with this project, I learned to understand the logic behind these digital tools. It is usually possible to find the answer to your problem in the manual or in a chat forum. This makes it easy to carry out a project like this, if you take the time to read the manual. I wish I knew this before going into the project, because that would have saved me a lot of time just trying out different gibberish commands that did not work.

7 Conclusions

By Working on this project, I learned how to OCR pdf-documents and to use the text data from the OCR to perform a textmining analysis. I created a forloop script for bash, that repeats OCRmyPDF for any number of pdf-files in a given folder. I showed the difference between term frequency and inverse document frequency and how to combine the two – tf_idf. I performed a sentiment analysis on the text and visualized this by a comparison cloud. This comes to show that there is a number of tools that can be used when performing text mining and that a pdf-file without text data does not represent a boundary for this work. Even with little computing power and with little digital knowhow you can do extensive text mining on large bodies of text.

Learning Journal

I present here 3 pages of my Learning Journal. I included the journal of week 46 and 47, because these were the weeks where I learned the skills necessary to complete my exam project. I also included the weekly exercise of week 47, because I think it is relevant to see my learning process through the exercise.

[33] GitHub, "CIA_sources_original"
[34] "Doctor Zhivago", CIA, accessed January 8, 2021, https://www.cia.gov/readingroom/search/site/zhivago

W46 - research design and an introduction to *shell* and *git*

09.11.2020

08:10

In preparation for the lecture today I'm reading up on the texts and data carpentry lecture for this week. First, I took a look at the text that's linked in the syllabus, http://cristal.inria.fr/~weis/info/commandline.html.

It didn't take me long to lose all will to finish the text. Not because it wasn't exciting stuff. I love this kind of stories, but the length and the format made me impatient, and I quit reading it. I still skimmed down through the page to see if there was anything of specific value to me. It seemed like a well written text in a format that didn't make it justice. So, I had to google the author, Neal Stephenson. It turns out that he's a well renown author who writes both fiction and nonfiction. That's why it seemed so well written; this also adds to my confusion about why it's presented in such a weird format.

10:15

I'm playing around with the terminal exercises in data carpentry. Again, this is a completely new language. I'm very excited about the capabilities of the terminal. I think that this would be at very useful tool to use when managing documents among other things. I'm a little afraid that I do something that can't be undone, so it's with some degree of fear that I go into this exercise.

I ran into problems when trying to open the help option. I typed in $ ls --help and got an error message. I immediately panicked and googled the problem. That didn't help much so I went to slack to ask the teachers. It didn't take me long to realize that all I had to do was scroll a little down in the exercise to see that all I had to do was to use another command $ man ls. Next time I shouldn't panic so quickly.

11.11.2020

07:18

I'm slowly getting familiar with the terminal, but it takes such a long time to finish the DC (data carpentry) tutorial and even longer to learn how to use the terminal. I think that this skill would be useful for my final exam project, so I'm even more focused on learning this skill.

I'm still a little afraid of doing something wrong since I read that you easily can delete something for good. Since there is no trash folder in terminal there is no easy way of retrieving that deleted data.

11:20

Used a lot of time trying to follow the DC tutorial on shell. It's quite complicated and I'm not sure I fully understand its properties. This might come if I work on it some more. Especially the loops and $ grep function I'm having a hard time wrapping my head around.

I'm now getting into the git DC tutorial. Hopefully I won't have to use as much time on this subject, otherwise I'm going to run into time problems.

12.11.2020

12:00

Hands-on session. The blended learning format doesn't work if you are the one who's following on the computer. I don't know how I'm supposed to learn this when I have no possibilities to get the help of the teacher. I don't know if I should attend the lessons when they are online. It doesn't work for me.

W47 - R for social scientists

19.11.2020

08:30
I follow the DC tutorial to learn R. I want to go through the basics before class at 11 in order to know what's going on.

10:00
I ran the basic R functions that we go through in the DC tutorial. It seems like the R language is very similar to the bash language that we did last week. There are some differences in syntax, but the functions seem to be the same. I like how it's a complete package in Rstudio.

12:00
In class I quickly realize that that stuff I did with the tutorial is the same as what we'll do in class, so I start to research my exam project. I want to do OCR on 98 .pdf documents. I have heard that you can do this in your shell with a package called OCRmyPDF. This seems like the program for me, I just need to figure out how to download and use the program.

13:30
I found that in order to download OCRmyPDF you have got to install the downloader Homebrew. From Homebrews website I downloaded homebrew. With homebrew installed I could download OCRmyPDF in my terminal. I just followed the guide on https://ocrmypdf.readthedocs.io/en/latest/. This was all new for me, but manageable as long as you did every step the guide tells you to.
Petra told me about a package for R which supposedly does the same as OCRmyPDF, it's called Tesseract. I will try this out before I try OCRmyPDF, since it would be nice to stay within the R workspace for my entire project.

14:30
It seems that the tesseract package works, but it doesn't do what I want. It produces a separate .txt file and converts my PDF file to .png. I want the .pdf files to stay .pdf but with the OCR text within the file. I think I need to figure out OCRmyPDF to have this function.

17:30
With OCRmyPDF installed I can check if it is working. I did some test runs with just one file to learn the syntax of the OCRmyPDF function. With this working I needed to figure out how to run a for-loop to repeat det command on all of my 98 documents. Since I didn't really understand for-loops last week and I wasn't able to ask my questions in class, I read the

tutorial on for-loops once more and wrote out my best guess. Then I can ask the teachers at hacky hours tomorrow.

20.11.2020

14:30

For hacky hours I worked with Petra to make my for-loop work. Now it worked. I made a bash script for my for-loop which I find very cool.

Weekly exercise:

1) Use R to figure out how many elements in the vector below are greater than 2. (You need to filter out the NAs first)

rooms <- c(1, 2, 1, 3, 1, NA, 3, 1, 3, 2, 1, NA, 1, 8, 3, 1, 4, NA, 1, 3, 1, 2, 1, 7, 1, NA)

```
#In order to work with the vector I need to tell it to Rstudio
rooms <- c(1, 2, 1, 3, 1, NA, 3, 1, 3, 2, 1, NA, 1, 8, 3, 1, 4, NA, 1, 3, 1, 2, 1, 7, 1, NA)
#I extract the NA's, and name this list of vectors "rooms_clean"
rooms_clean <- rooms[!is.na(rooms)]
#I want to know how many elements in the vector is greater than 2
length (rooms_clean[rooms_clean>2])
#This gives me 8. So 8 of the elements in the vector is greater than 2
```

2) What is the average number of rooms (result of running mean() function) in the above 'rooms' vector? Again, best remove the NAs first.

#To find the average number of rooms I use the mean-function. This function *calculates by taking the sum of the values and dividing with the number of values in a data series* = average. I use the same list of vectors that I created before "rooms_clean".

```
mean(rooms_clean)
#This gives me 2.318182
#I want it to be a whole number, since you can't have 2.3 rooms in a house. For this I use the round function
round(mean(rooms_clean))
#this tells me that the average number of rooms is 2
```

3) What type of data is in the 'rooms' vector? What function helps you determine the answer?

```
#I think the function I want to use is class()
class(rooms)
#this tells me that the vectors in "rooms_clean" is numeric.
```

*Acknowledgements

I would like to say thank you to Max Odsbjerg Pedersen for his technical help with text mining in R. I would also like to say thank you to Petra Hermankova for helping me figure out how to make my forloop work.

References

CIA. “Doctor Zhivago.” Accessed January 1, 2021. https://web.archive.org/web/20201212131158/https://www.cia.gov/library/readingroom/collection/doctor-zhivago

OCRmyPDF. “OCRmyPDF documentation.” Accessed December 30, 2020. https://ocrmypdf.readthedocs.io/en/latest/

Odsbjerg Pedersen, Max. Det Kgl. Bibliotek. “Aarhus Byråds forhandlingsprotokoller 1930-1940 term frequency(tf) – Inverse Document Frequency(idf).” Accessed December 30, 2020. https://kulturarvscluster.kb.dk/arkiver/109

Silge, Julia. ”Text Mining, the Tidy Way.” Filmed April 21, 2017, at Work-Bench, New York. https://www.youtube.com/watch?v=0poJP8WQxew

Software Carpentry. “The Unix Shell, Loops.” Accessed January 1, 2021. http://swcarpentry.github.io/shell-novice/05-loop/index.html

Text mining with R. ”The tidy text format.” Accessed December 30, 2020. https://www.tidytextmining.com/tidytext.html

Wilkinson, Mark D., Michel Dumontier, Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, *et al.* "The Fair Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3, no. 1 (2016)

B - Required Metadata

Table 1 – Software metadata

Nr	**Software metadata description**	***Please fill in this column***
S1	Current software version	*R v4.0.3, RStudio v1.3.1093, bash v3.2, OCRmyPDF v.11.3.3, Homebrew v2.5.11, Tidytext v.0.2.6, tidyverse, v.1.3.0, Readtext v.0.80, textdata v.0.4.1, Lubridate v.1.7.9.2.*
S2	Permanent link to executables of this version (your Github repo URL)	*https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus*
S3	Legal Software License	*This software is to be used under the license of creative commons 4.0*
S4	Computing platform / Operating System	*OS Big Sur version 11.0.1, Macbook Air (Retina, 13-inch, 2019), 1,6 GHz Dual-Core Intel Core i5, 8 GB ram, Intel UHD Graphics 617 1536 MB*
S5	Installation requirements & dependencies for software not used in class	*Homebrew, Tidytext, Readtext, Lubridate, Textdata, OCRmyPDF*
S6	If available Link to software documentation for special software	*Homebrew (https://brew.sh/), Tidytext (https://github.com/juliasilge/tidytext), Readtext (https://cran.r-project.org/web/packages/readtext/readtext.pdf), Lubridate (https://cran.r-project.org/web/packages/lubridate/lubridate.pdf), Textdata (https://cran.r-project.org/web/packages/textdata/readme/README.html), tidyverse (https://cran.r-project.org/web/packages/tidyverse/index.html), OCRmyPDF (https://ocrmypdf.readthedocs.io/en/latest/)*
S6	Support email for questions	*201907052@post.au.dk*

Table 2 – Data metadata

Nr	**Metadata description**	***Please fill in this column***
D1	CIA_sources_original, The original data from the CIA archive.	*https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/tree/main/CIA_sources_orginal*
D2	CIA_sources_OCR_wMetadata, The data from the CIA archive after OCR and metadata in document-name	*https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/tree/main/CIA_sources_OCR_wMetadata*
D3	CIA_script.bsh.txt, forloop script for Bash.	*https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/blob/main/CIA_doctorzhivago_textmining_and_OCR/CIA_script.bsh.txt*
D4	CIA_test.Rmd, Rmarkdown of the text mining analysis in .Rmd-format	*https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/blob/main/CIA_doctorzhivago_textmining_and_OCR/CIA_test.Rmd*
D5	CIA_test.html, Rmarkdown of the text mining analysis in html-format	*https://github.com/Digital-Methods-HASS/au_615046_nielsen_markus/blob/main/CIA_doctorzhivago_textmining_and_OCR/CIA_test.html*