1:W35: Regular expressions and spreadsheets

1. What regular expressions do you use to extract all the dates in this blurb: http://bit.ly/regexexercise2 and to put them into the following format YYYY-MM-DD?
   - Marking all numbers:
     - (\d+.)(\d+.*)(\d+)
   - Substituting numbers
     - $2$1$3
2. Write a regular expression to convert the stopwordlist (list of most frequent Danish words) from Voyant in http://bit.ly/regexexercise3 into a neat stopword list for R (which comprises "words" separated by commas, such as http://bit.ly/regexexercise4). Then take the stopwordlist from R http://bit.ly/regexexercise4 and convert it into a Voyant list (words on separate line without interpunction)
   - Removing all characters:
     - (\", ")|(\")
   - In substitution field add new line:
     - \n
3. In 250 words, answer the following question: "What are the basic principles for using spreadsheets for good data organisation?"

Keep data in a tidy format with variables in columns and observations in rows. Each field should contain only one value. Keep strings and numeric variables separated. Be consistent in naming variables, avoid special characters and spaces except for underscore (_). Name variables in a logical and comprehensible way.

2:W35: Open Refine

1. **Create a spreadsheet listing the names of Danish monarchs with their birth- and death-date and start and end year of reign. Make it *tidy*! They should be sortable by year of birth. Suitable source websites are [here]() and [here](), but you can also use another source, provided you reference it. (Group collaboration is expected and welcome. Remember to attach this spreadsheet to Brightspace submission)**
    a. Link to spreadsheet: https://docs.google.com/spreadsheets/d/18bc1U0Xa_XkwxHNv9C_OYTzzFO1oGQqC6_3kZ7mYYeA/edit?usp=sharing
2. **Does OpenRefine alter the raw data during sorting and filtering?**
    a. No, it only happens at the UI in OpenRefine.
3. **Fix the [interviews dataset]() in OpenRefine enough to answer this question: *"Which two months are reported as the most water-deprived/driest by the interviewed farmer households?"***
    a. Go into custom facet on column and
        i. value.replace("['","").replace("']","").replace("'","").replace(" ","")
        ii. And then split by: value.split(”;”)
        iii. Value.split(”;”)
    b. By then I get the result: November 41
4. **Real-Data-Challenge: What are the 10 most frequent occupations (erhverv) among unmarried men and women in [1801 Aarhus]()? (hint: some expert judgement interpretation is necessary, look at the [HISCO classification]() "Historical International Standard of Classification of Occupations" on [Dataverse]() if ambitious)**
    a. I could do this in OpenRefine by loading the data by the URL, apply text facet, and sort by count. The top 10 occupations are the listed as below. I am aware that we have a problem with naming, namely several of the variables are called by different names although the occupation is the same. This could be easily fixed in R with the replace() function or something similar.

b. I achieve this with the following command in R:

i. data.frame(sort(table(df$erhverv), decreasing = TRUE)[1:10])

Variable	**Frequency overall**
Bonde og Gaardbeboer	1780
Huusmand med Jord	660
bonde og gaardbeboer	231
bonde og gÃ¥rdbeboer	223
soldat ved 1. Jyske Inf. Reg.	136
Bonde og Gaard Beboer	102
National Soldat	99
Bonde og Gaardmand	98
Gaardbeboer	97
nyder Ophold af Gaarden	90

Variable	**Frequency for unmarried women**
Tienestepige	61
hospitalslem	21
Væverske	18
Lever af at spinde	17
Inderste	14
Tjener faderen	13
Lever af almisse	12
Spinder	12
Vanfør	10
almisselem	9

Variable	**Frequency for unmarried men**
National Soldat	96
Soldat ved 1. Jyske Inf. Reg.	94
nationalsoldat	61
Tienestesoldat	47

læredreng	42
Nationalsoldat	36
Bonde og Gaardbeboer	32
indludeeitTienestedræng	32
Væver	32
gårdskarl	30

Homework 3

WIBE

8/30/2022

Instructions: For this assignment, you need to answer a couple questions with code and then take a screenshot of your working environment.

1) Use R to figure out how many elements in the vector below are greater than 2 and then tell me what their sum (of the larger than 2 elements) is.

```
rooms <- c(1, 2, 4, 5, 1, 3, 1, NA, 3, 1, 3, 2, 1, NA, 1, 8, 3, 1, 4, NA, 1, 3, 1, 2, 1, 7, 1, 9, 3, NA)

# Counting how many elements are greater than two excluding the NA's
length(na.omit(rooms[rooms > 2]))
```

```
## [1] 12
```

```
# Counting the sum of numbers greater than two excluding the NA's
sum(na.omit(rooms[rooms > 2]))
```

```
## [1] 55
```

Answer:

- Length of elements greater than two = **12**
- Sum of elements greater than two = **55**

2) What type of data is in the 'rooms' vector?

'Class' function tells us the class of a vector:

```
class(rooms)
```

```
## [1] "numeric"
```

The class of rooms is **numerical**

3) Installing tidyverse, loading data using read_csv()

Save the screenshot as an image and put it in your AUID_lastname_firstname repository inside our Github organisation (github.com/Digital-Methods-HASS) or equivalent. Place here the URL leading to the screenshot in your repository.

```
library(tidyverse)
interviews <- read_csv("/Users/WIBE/Desktop/CogSci/Cultural data science/Introduction course/Day 2/data/
```

```
## Rows: 131 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): village, respondent_wall_type, memb_assoc, affect_conflicts, items...
## dbl  (6): key_ID, no_membrs, years_liv, rooms, liv_count, no_meals
## dttm (1): interview_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

4) Challenge: If you managed to create your own Danish king dataset, use it. If not, you the one attached to this assignment (it might need to be cleaned up a bit). Load the dataset into R as a tibble. Calculate the mean() and median() duration of rule over time and find the three monarchs ruling the longest. How many days did they rule (accounting for transition year?)

```
kings_and_queens <- read_csv(as_tibble("/Users/WIBE/Desktop/CogSci/Cultural data science/Introduction c
```

```
## Rows: 55 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Monarch, birth_date, death_date
## dbl (4): birth_year, death_year, start_reign, end_reign
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Calculating mean and median duration of rule over time

```
median((kings_and_queens$end_reign)-(kings_and_queens$start_reign), na.rm = T)
```

```
## [1] 16.5
```

```
mean((kings_and_queens$end_reign)-(kings_and_queens$start_reign), na.rm = T)
```

```
## [1] 19.33333
```

Mean reign length is: **47 years** Median reign length is: **43 years**

Finding the three monarchs ruling the longest

```
kings_and_queens %>%
  mutate(reign_length = end_reign-start_reign) %>%
  arrange(desc(reign_length)) %>%
  select(Monarch,reign_length) %>%
  head(3)
```

```
## # A tibble: 3 x 2
##   Monarch            reign_length
##   <chr>                     <dbl>
## 1 Christian 4.                 60
## 2 Erik 7. af Pommern           43
## 3 Christian 9.                 43
```

The three longest reigning kings were

1) Christian 4. ruling for 60 years
2) Erik 7. af Pommern ruling for 43 years
3) Christian 9. ruling for 43 years

Are we more civilized today?

Jørgen Højlund Wibe

31 December, 2022

Contents

This exercise is based on the dataset provided by OurWorldInData project based at the Oxford University.

The long-term trend in Homicides in Western Europe

Understanding how homicide rates have changed prior to the modern era requires the help of historians and archivists. Manuel Eisner, a criminology professor at the University of Cambridge, and his colleagues published the Historical Violence Database : a compilation of data on long-term trends in homicide rates, in addition to qualitative information such as the cause of death, perpetrator and victim. This database is limited to countries with relatively complete historical records on violence and crime – mainly Western Europe and the US.

Starting in the second half of the nineteenth century, these European regions have consistent police records of those accused of murder or manslaughter and annual counts of homicide victims. To go back further in time, reaching as far back as the thirteenth century, Eisner collected estimates (from historical records of coroner reports, court trials, and the police) of homicide rates made in over ninety publications by scholars.

Homicide rates – measured as the number of homicides per 100,000 individuals – up to 1990 are sourced from Eisner's (2003) publication and the Historical Violence Database.

Are homicide rates in Europe today lower or higher than in the past? Using the provided dataset, display and describe the long-run homicide rates for the five European regions: Italy, England, Germany, Netherlands and Scandinavia.

```
library(pacman)
pacman::p_load(tidyverse, gridExtra, cowplot)
```

Load the available data from ourworldindata.org

You should always interrogate the source of your data. Who compiled it, from where, what is missing, how representative the data are? Check the data/Metadata.txt to learn about the data provenance.

```
Western_Europe <- read_csv("data/homicide-rates-across-western-europe.csv")
```

Inspect the data

The generally data looks clean and ready for analysis. We should rename the Homicide columns due to the spaces and length of it

Description of column names

- *Entity:* Contains the name of the country
- *Year:* Contains the year from which the homicide rate revolves around
- *Homicide rate:* Measured as the number of homicides per 100,000 individuals

```
ls.str(Western_Europe)
```

```
## Code :  chr [1:206] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA ...
## Entity :  chr [1:206] "England" "England" "England" "England" "England" "England" ...
## Homicide rate in Europe over long-term (per 100,000) (homicides per 100,000 people) :  num [1:206] 23
## Year :  num [1:206] 1300 1550 1625 1675 1725 ...
```

- Use the `names()` function and assignment key to relabel this column to `homicides_per_100k`

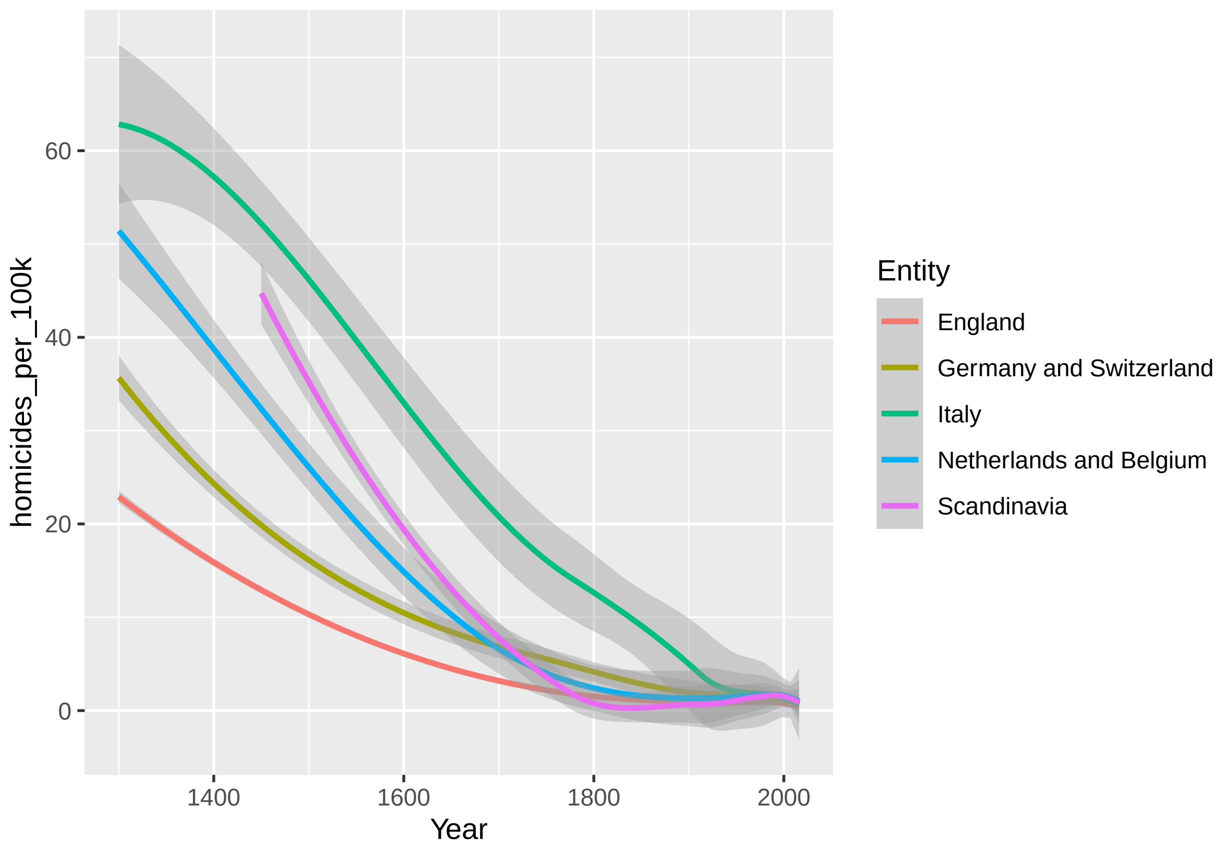
```
# Using the rename() function to rename homicide variable.
Western_Europe <- Western_Europe %>%
  rename(homicides_per_100k = `Homicide rate in Europe over long-term (per 100,000) (homicides per 100,0
```

Now, that you have looked at what the data looks like and what it represents, and streamlined it, let's see what big picture it contains.

Let's see what the long-term trend is in homicides

- use `ggplot()` function and remember the`+` at the end of the line
- chose a meaningful `geom_......()` for geometry (hint: points are not great)
- load `Year` on the `x` axis and `homicides_per_100k` column in y axis
- to color individual country entries consistently, assign the country column to the argument `color`.
- provide meaningful title and axis labels
- remember to change the `eval` flag so that the code chunk renders when knitted

```
pl1 <- ggplot(data = Western_Europe,
       aes(x = Year, y = homicides_per_100k, colour = Entity)) +
  geom_smooth()
print(pl1)
```



The plot shows a descending homicide rate over time for all countries. Italy has generally had a higher homicide rate than the other four countries but today the rates are fairly similar.

Uncouple the homicides of individual countries for easier view

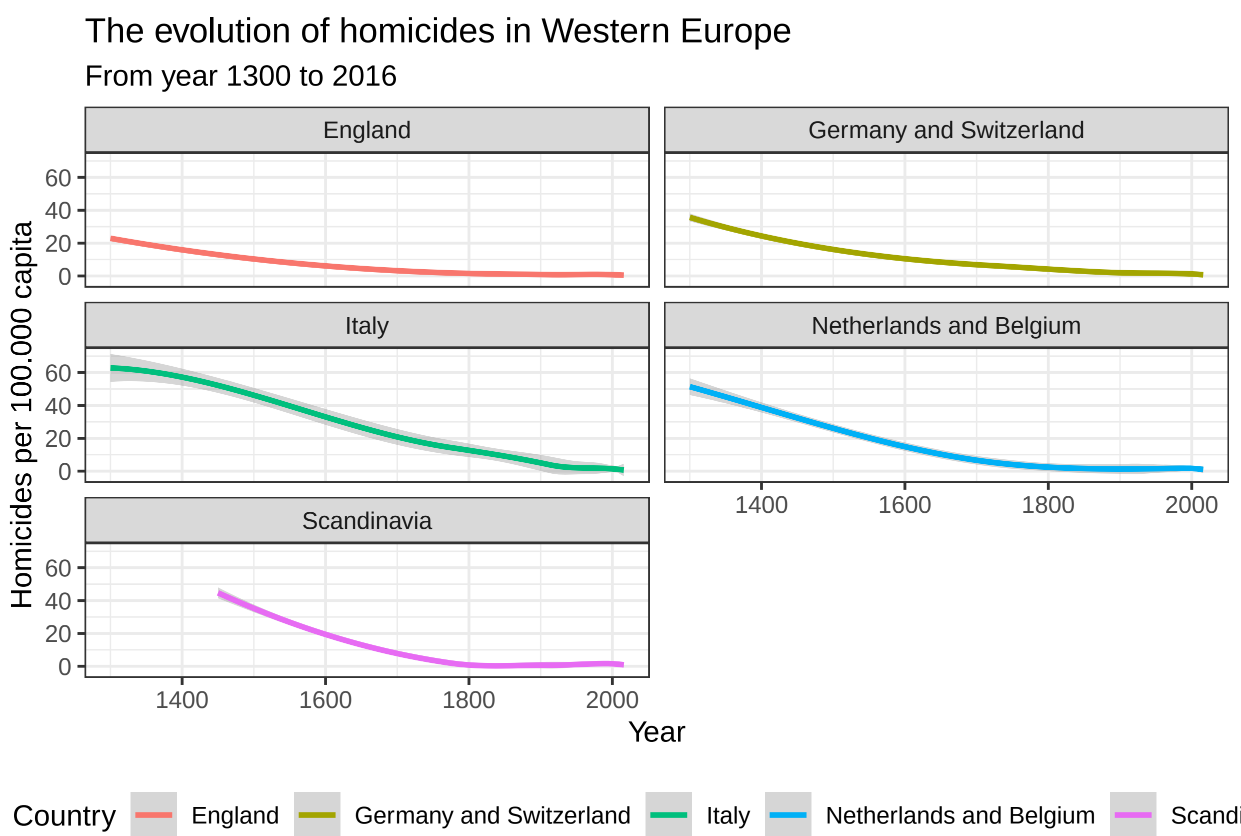
You can visualize each country's trend separately by adding an extra argument to the ggplot, the `facet_wrap()` and feeding it the country column. If in doubt, check your ggplot tutorial and your country column name for exact usage.

- reuse the ggplot from the chunk above
- insert `facet_wrap()` after the specification of geometry to split countries in separate charts
- change the facet "layout" to two columns and three rows so that the trends are easier to see in horizontal layout.

```
# Plotting the evolution of homicides through time
plt1 <- ggplot(data = Western_Europe,
       aes(x = Year, y = homicides_per_100k)) +
  geom_smooth(aes(colour = Entity))+
  facet_wrap(~ Entity, nrow = 3, ncol = 2)+
  labs(title = "The evolution of homicides in Western Europe",
       subtitle = "From year 1300 to 2016",
       x = "Year",
       y = "Homicides per 100.000 capita")+
  theme_bw()+
  theme(legend.position = "bottom")+ # changing legend position
```

```
  labs(colour = "Country") # title of legend

plt1
```



Compare the trends in homicide with the pattern of reign duration among Danish rulers through time.

- Load your Danish king dataset. Hopefully it is tidy and your years and duration of reign are all numeric.
- You need to have a consistent way of plotting the rulers' reign on the x axis, so I recommend you create a midyear column by calculating the middle of each monarch's rule (Hint: `midyear` = `endyear - (endyear-startyear)/2`)
- Start a ggplot plotting midyear on x axis and duration on y axis
- Try `geom_smooth()` for geometry
- Provide meaningful labels and a title
- How would you characterize the trend compared to the homicides above?

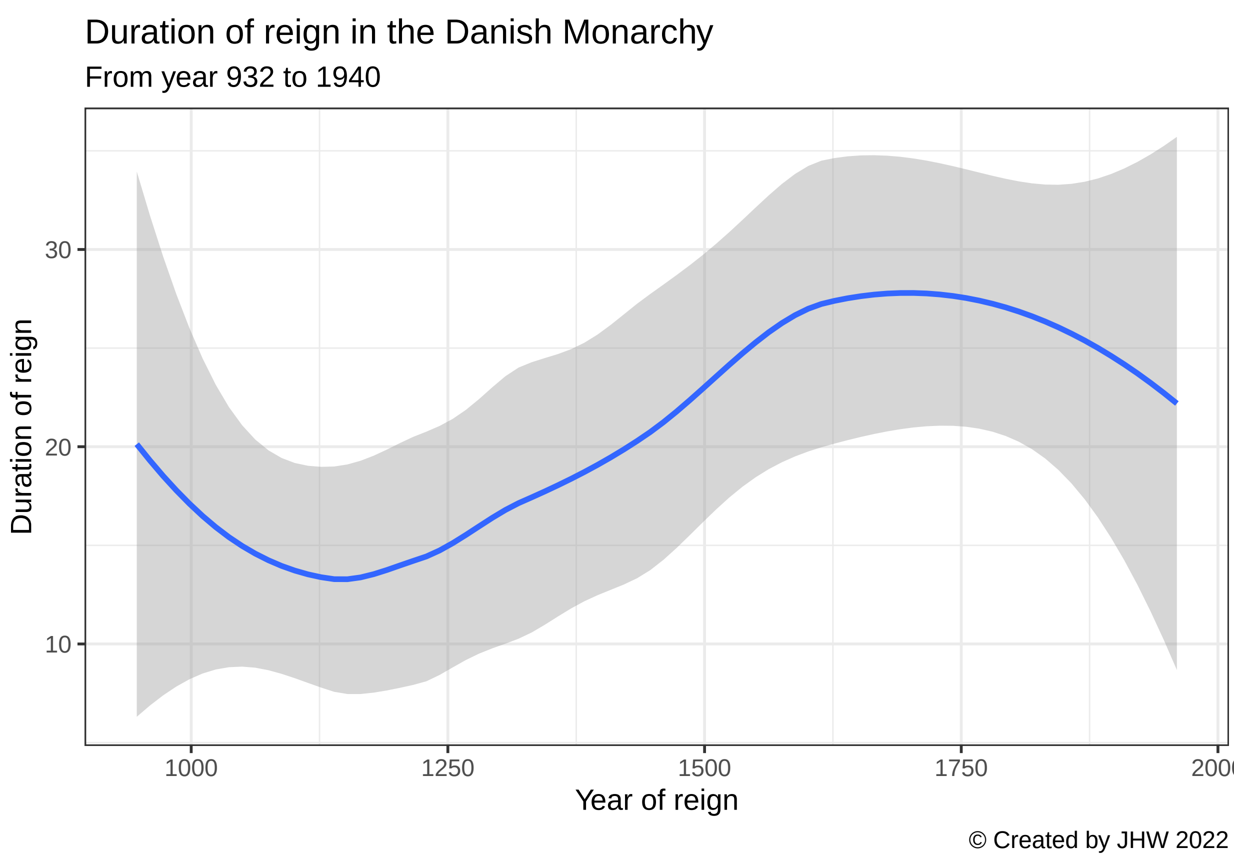
```
# Loading the monarchs dataset
dk_monarchs <- read_csv("data/Danish Monarchs - Sheet1.csv")

# creating reign_length and middle year of range
dk_monarchs <- dk_monarchs %>%
  mutate(reign_length = end_reign-start_reign) %>% #creating reign_length variable
  mutate(midyear = round(end_reign - (reign_length/2)), digits = 0) # finding middle of the regents rei
```

```r
# Plotting monarchs reign length through time
plt2 <- ggplot(dk_monarchs,
               aes(x = midyear, y = reign_length))+
  geom_smooth()+
  labs(title = "Duration of reign in the Danish Monarchy",
       subtitle = "From year 932 to 1940",
       x = "Year of reign",
       y = "Duration of reign",
       caption = "© Created by JHW 2022")+
  theme_bw()

plt2
```

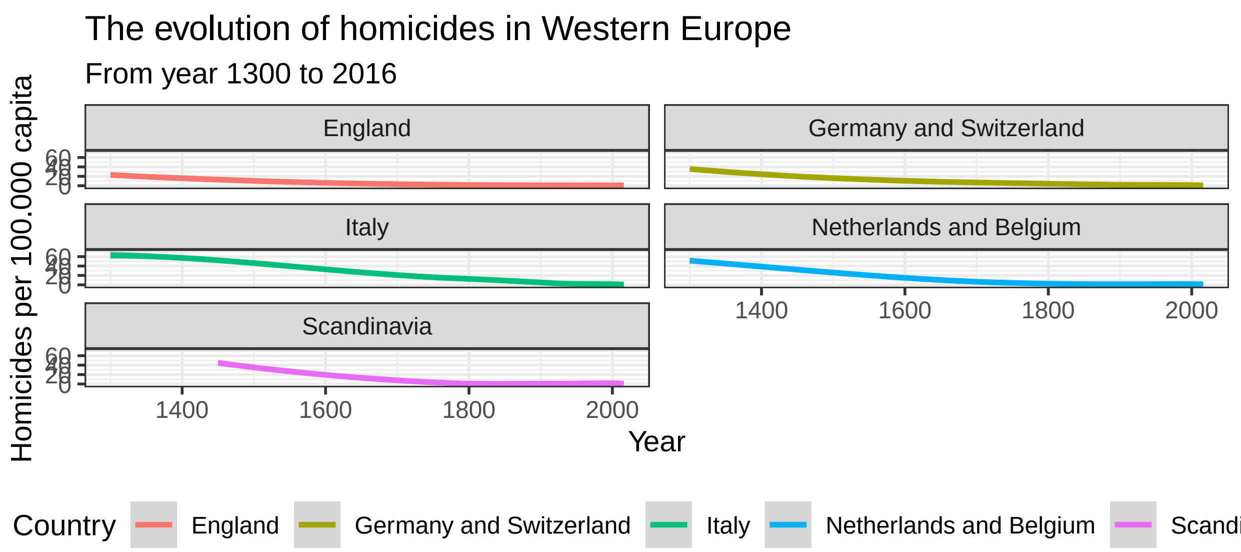
```
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
```

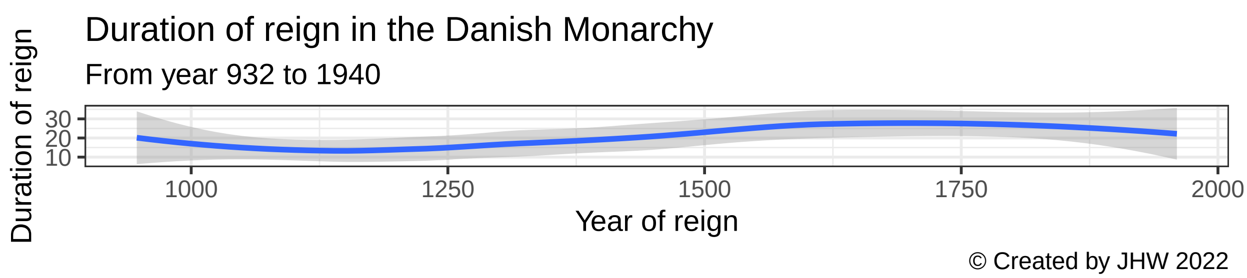



```r
plot_grid(plt1, plt2, align = 'v', nrow = 2, rel_heights = c(2/3,1/3))
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).

## Warning: Graphs cannot be vertically aligned unless the axis parameter is set.
## Placing graphs unaligned.
```





From the plot it is obvious that there is a direct causal link between the decline of homicide rates in Europe and an increase of reign duration among Danish monarchs. That means that we should do our best to keep queen Margrethe the 2nd alive for as long as possible. For her sake and for the people of Europe's sake. Long live the queen.

#Final tasks:

1) Plot: In the faceted plot above, move the legend from the current position on the side to below the facets, and label it "Country" instead of "Entity".

See plot number 1.

2) Rmarkdown:

- edit the author of the document, and convert 'Final Tasks' into heading #2 (like the other headings)
- add a `floating table of contents` to your Rmarkdown document,
- provide informative `chunk-names` and edit flags in your R chunks, and
- automatically generate a `timestamp` to show when the document was last updated. (Hint: check the Rmarkdown episode in our Data Carpentry tutorial)

3) Question: In <250 words articulate your answer on the basis of the data visualisations to the following question: are we more civilized today?

Based on the limited data we have access to through this assignment we are better off today than in earlier days. Less people are deliberately killed and it seems as if the monarchy is more stabilized with the monarchs sitting for longer periods.

5:W35: Managing Files on Steroids with Shell

1) Identify the names and format of the 3 biggest files. Can you come up with a command to generate a numerically ordered list of 3 biggest files? (hint: consider using wc to gauge image size)

Commands to find 3 largest file:
 1. wc -c * > filesizes.txt
 2. sort -n filesizes.txt | tail -n 4
-n orders files numerically

Result:
```
 14713856 9237_Overview_W.RW2
 14733312 9247_Overview_SW.RW2
 14761472 9240_Overview_S.RW2
```

2) Some of the image files are empty, a sign of corruption. Can you find the empty photo files (0 kb size) , count them, and generate a list of their filenames to make their later replacement easier?

Commands:
 1. find . -size 0 > corrupted_files.txt
 2. wc -l corrupted_files.txt

The number of corrupted files are 74 and the files are placed in the .txt file corrupted_files.txt

3) Optional/Advanced: Imagine you have a directory goodphotos/ (same password as above) with original non-zero-length files sitting at the same level as the current directory. How would you write a loop to replace the zero length files?

Homework 6

Cultural Datascience 2022

Jørgen Højlund Wibe

Loading data

```
library(pacman)
pacman::p_load(gapminder, tidyverse)
df <- gapminder
```

Task 1

Task description

Define a defensive function that calculates the Gross Domestic Product of a nation from the data available in the gapminder dataset. You can use the population and GDPpercapita columns for it.

Using that function, calculate the GDP of Denmark in the following years: 1967, 1977, 1987, 1997, 2007, and 2017.

Solution

```
# Defining function
find_gdp <- function(df, year=NULL, country=NULL) {
  if (!is.numeric(year)) {
    warning(stop("Year must be of type numeric"))
  }
  if (!is.character(country)) {
    warning(stop("Country must be of type character"))
  }
  if(!is.null(year)) { # if the 'year' variable is not empty...
    df <- df[df$year %in% year, ] # add the specified year to df
  }

  if (!is.null(country)) { # same as above
    df <- df[df$country %in% country, ] # -//-
  }

  gdp <- df$pop * df$gdpPercap # calculating gdp: multiplies population with GDP per capita

  new_df <- cbind(df, gdp=gdp) # combining results into one dataframe
  return(new_df)
}

# Running function, specifying values
find_gdp(df, year = c(1967, 1977, 1987, 1997, 2007, 2017), country = "Denmark" )
```

```
##   country continent year lifeExp     pop gdpPercap          gdp
## 1 Denmark    Europe 1967  72.960 4838800  15937.21  77116977700
## 2 Denmark    Europe 1977  74.690 5088419  20422.90 103920280028
## 3 Denmark    Europe 1987  74.800 5127024  25116.18 128771236166
## 4 Denmark    Europe 1997  76.110 5283663  29804.35 157476118456
## 5 Denmark    Europe 2007  78.332 5468120  35278.42 192906627081
```

Task 2

Task description

Write a script that loops over each country in the gapminder dataset, tests whether the country starts with a 'B' , and prints out whether the life expectancy is smaller than 50, between 50 and 70, or greater than 70. (Hint: remember the grepl function, and review the Control Flow tutorial (https://swcarpentry.github.io/r-novice-gapminder/07-control-flow/index.html))

Solution

```
countries <- unique(df$country)

for (i in countries) {
  life_expectancy <- df %>%
    filter(country == i) %>%
    pull(lifeExp)

  if ((mean(life_expectancy) < 50) && (i %in% grep("^B", countries, value = T))) {
    print(paste0(i, " starts with 'B' and has a average life expectancy less than 50!"))
  }

  else if (between(mean(life_expectancy), 50, 70) && (i %in% grep("^B", countries, value = T))) {
    print(paste0(i, " starts with 'B' and has an average life expectancy between 50 and 70"))
  }

  else if (mean(life_expectancy) > 70 && (i %in% grep("^B", countries, value = T))) {
    print(paste0(i, " starts with 'B' and has an average life expectancy more than 70!"))
  }
}
```

```
## [1] "Bahrain starts with 'B' and has an average life expectancy between 50 and 70"
## [1] "Bangladesh starts with 'B' and has a average life expectancy less than 50!"
## [1] "Belgium starts with 'B' and has an average life expectancy more than 70!"
## [1] "Benin starts with 'B' and has a average life expectancy less than 50!"
## [1] "Bolivia starts with 'B' and has an average life expectancy between 50 and 70"
## [1] "Bosnia and Herzegovina starts with 'B' and has an average life expectancy between 50 and 70"
## [1] "Botswana starts with 'B' and has an average life expectancy between 50 and 70"
## [1] "Brazil starts with 'B' and has an average life expectancy between 50 and 70"
## [1] "Bulgaria starts with 'B' and has an average life expectancy between 50 and 70"
## [1] "Burkina Faso starts with 'B' and has a average life expectancy less than 50!"
## [1] "Burundi starts with 'B' and has a average life expectancy less than 50!"
```

GrundtviGPT-3: Where AI Meets Culture

Exploring how GPT-3 can be used to generate Danish lyrics of cultural nature by fine-tuning it on the Danish songbook: *Højskolesangbogen*

Jørgen Højlund Wibe (201807750)
Niels Krogsgaard (202008114)

Supervisor: Adéla Sobotkova
Exam Paper · Cultural Data Science
School of Communication and Culture, University of Aarhus
January 12th 2023

GitHub Repository: `https://github.com/jorgenhw/GrundtviGPT-3`

Characters in total: 19132

Abstract

This paper is an exploratory investigation of the capabilities of state-of-the-art multilingual language models to generate Danish song lyrics of cultural nature. We fine-tune OpenAI's GPT-3 on a Danish songbook named *Højskolesangbogen* which contains songs of cultural importance to Denmark and made it generate songs of similar content based on prompts comparable to the original data.

We found that GPT-3 is able to pick up the tone and structure from the songs although rhyming and semantic coherence was not always perfect. However, when faced with English songs, rhyming and coherence were nearly perfect. One explanation for this could be that GPT-3 is pre-trained on a massive amount of English data and much less for Danish meaning that it generally performs better in English.

For further research, we suggest using participants to assess both the quality of the generated song lyrics, but also the distinguishability between the original and AI-generated versions. By validating this method for producing song lyrics it can then be used as a tool for future song creation.

Overall, our research demonstrates the potential of using fine-tuned language models like GPT-3 for generating culturally relevant lyrics in different languages. This has important implications for the music industry and for preserving cultural traditions through the creation of new, relevant content.

Keywords: *NLP, machine learning, text generation, transformer model, GPT-3, Danish*

Contents

1 Introduction (NK)

Music accompanied by song lyrics has existed for centuries. The oldest document with both text and melody is the 'Seikilos Epitaph' that can be dated back to 200-100 BC (Hemingway, 2001), however music accompanied by lyrics has most likely existed long before that. Through time, creating songs has been used to entertain, tell tales, proclaim feelings, describe the indescribable, and pass on knowledge to the generations to come. The task of creating songs is a creative task that has not changed its workflow through the many centuries it has existed. To create a song, a person or group of people need to come up with a matching text and melody. However, modern technology can now automate the creative process of generating content that was previously exclusive to humans such as paintings, melodies, poems, fairy tales, and song lyrics. Tools such as DALL-E and GPT-3 have made it possible for normal people with little experience to generate original art in the form of pictures and text. This indicates the beginning of a new era for working creatively, where content can either be generated solely by or in collaboration with an AI tool. It also sparks the debate of how art generated by an AI should be understood, and whether we can even talk about it in the same way as art generated by humans.

Automatically generating lyrics has the potential of making song generation much simpler. The problem for low-resource languages like Danish is that the best-performing language models like GPT-3 are primarily trained on English text. Despite this, it has shown surprisingly good performances in Danish as well (Lin et al., 2022). However, much research still needs to be done on low-resource languages such as Danish. Therefore, we intend to investigate how well a fine-tuned version of GPT-3 can generate song lyrics in Danish. We intend to use a corpus of songs that are homogenous in structure and language to create the best possible conditions for GPT-3. Therefore we use the Danish Folk Highschool Songbook (hereafter *Højskolesangbogen*). It is a collection of songs that are central to the Danish culture of singing together with some songs dating back to 16th-century (Faurholt, 2021). The book today consists of a broad range of songs that have been found culturally relevant for Danish people of all ages.

2 Problems and Background (JW)

2.1 Large Language Models

Natural Language Processing is the field of computer science that has to do with text, speech and language in general. It is used to automate tasks such as translation, summarisation, topic extraction, or text generation (Gudivada & Arbabifard, 2018). The field consists of a large diversity of techniques ranging from analysing simple word frequencies to training complex artificial neural networks for a large number of tasks. This paper will make use of one of the best-performing large language models that make use of methods originating in artificial neural networks (Lek & Park, 2008). These networks are capable of training on a large corpus of text and learning patterns in the data (Lek & Park, 2008).

In recent years we have seen rapid improvements in the area of NLP. Methods such as deep learning and the transformer architecture have greatly improved the performance of large language models (Vaswani et al., 2017). Since GPT-3 is a transformer model, we will in the following section briefly explain the main idea behind this architecture.

The transformer is a type of neural network that consists of an encoder and a decoder. The encoder part can be understood as the model creating a vector representation of an input. The decoder part can be understood as the model using the vector representation as input to create a sequence of words as output (Cho, van Merrienboer, Bahdanau, & Bengio, 2014). If the task was a translation of an English sentence to Danish, then the input sequence of the encoder part would be an English sentence, while the output of the decoder part would be a Danish sentence. This is illustrated in Figure 1.

Another central concept in the transformer network is self-attention. Conceptually speaking, it is a way for the model to incorporate, how the words in a sentence relate to each other. For example, in the sentence "My dog cannot lift the book because it is too heavy". Self-attention allows the model to solve the task of associating "it" with "My dog", which is not a trivial task for a computer. Self-attention can be generalised to a large number of situations in different sentences, and in a highly influential paper by Vaswani et al. (2017) they show that this concept of self-attention solves the task of connecting words meaningfully together to produce impressive results. Explaining transformer models and their inner workings in detail is outside the scope of this paper, but we refer to Vaswani et al. (2017) and Alammar (2018) for a more thorough understanding.

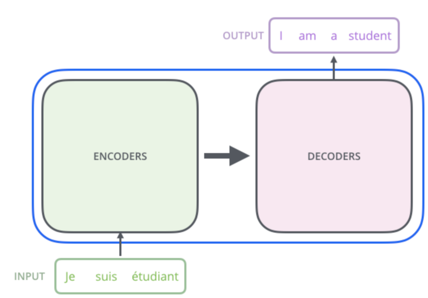


Figure 1: Visualisation of the encoder and decoder process of a transformer model. Source: (Alammar, 2018).

2.2 Generative Pre-trained Transformer 3 (GPT-3)

GPT-3 is developed by Brown et al. (2020) in collaboration with OpenAI. When released in 2020 it was by far the largest model with 175 billion parameters - more than 10 times the amount of parameters for the second largest model as can be seen in Figure 2. It achieves an incredibly high performance right out of the box, i.e. without any fine-tuning. This performance only accelerates with fine-tuning. Compared to its predecessor GPT-2, it is especially good at creative tasks such as generating song lyrics and telling stories.

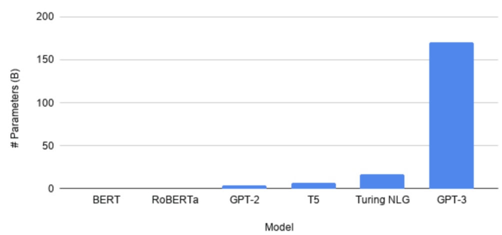


Figure 2: Histogram of the number of parameters in billions for the best-performing language model architectures in 2020. Source: (Ammu, Bhaskar, 2020).

3 Software Framework (NK)

The code for the project was written on a Macbook Pro from 2013 running on MacOS BigSur (version 11.6.8). To scrape data, preprocess data and fine-tune GPT-3, two programming languages were used: Python and R (R Core Team, 2022; Van Rossum & Drake, 2009). Each language provides different software packages enabling different analysis. RStudio (RStudio Team, 2020) and Google Colaboratory (Colab) (Bisong, 2019) were used as integrated development environments (IDE). Colab is a virtual notebook environment that executes code on virtual machines (Bisong, 2019).

For R, the following packages were used: Tidyverse (Wickham et al., 2019), Rvest (Wickham, 2022a), Stringr (Wickham, 2022b), and magrittr (Bache & Wickham, 2022).

For Python, the following libraries were used: Pandas (McKinney, 2010), numpy (Harris et al., 2020) and openai (Brockman et al., 2016).

The code used to produce the results in the present paper is presented as a notebook using the r-markdown format (Xie, 2022). Also, all code can be easily accessed via the GitHub repository:

`https://github.com/jorgenhw/GrundtviGPT-3`

The work in this paper can be easily reproduced for further research or quality checking by following the guidelines outlined in the `readme.md` file present in the GitHub repository.

4 Data Acquisition and Processing(JW)

The primary dataset used for the analysis was a comma-separated values file (.csv) containing 80% of the songs in *Højskolesangbogen*. In the following section, the acquisition and preprocessing of this data are outlined.

4.1 Data Acquisition

The songs were acquired through scraping directly from `https://hojskolesangbogen.dk/`. To scrape the songs manually, we used the R-package RVest (Wickham, 2022a) for data extraction with RStudio as IDE (RStudio Team, 2020). Concretely, Rvest was set up to extract all HTML links on the page containing links to the respective songs. Some songs had music videos or audio files available instead of lyrics. These songs were not included in the dataset used in this paper. For more information on the scraping process, we refer to our GitHub repository `https://github.com/jorgenhw/GrundtviGPT-3`

4.2 Pre-processing

All songs were pre-processed in Python (Van Rossum & Drake, 2009) in Google Collaboratory (Bisong, 2019). To prepare the data for fine-tuning a GPT-3 model, we moved the first verse of each song to a new column in the dataset which we named 'prompt'. We removed ten random songs from the dataset to use these as prompts for lyrics generation after having fine-tuned GPT-3. One empty row was removed and nine duplicate rows were excluded. Again, we refer to our GitHub repository for detailed information on pre-processing.

5 Fine-tuning GPT-3 for lyrics generation(NK)

GPT-3 was fine-tuned with 396 pre-processed lyrics using OpenAI's API in Python (Brockman et al., 2016). All of the lyrics were formatted to a JSONL in accordance with the specific data formatting guidelines provided by the OpenAI API documentation (Brockman et al., 2016).

The objective of fine-tuning GPT-3 is to enable the generation of lyrics similar in structure and semantics to the ones found in Højskolesangbogen. Therefore we the first verses of each song was given to the model as prompts where the corresponding completion was the rest of the song.

Before beginning the fine-tuning, the data were inspected using OpenAI's built-in function for ensuring that the data was pre-processed correctly. The function correctly identified 8 duplicate columns which were removed resulting in 396 songs left. The GPT-3 model Davinci was selected due to its superior performance compared to other models in the GPT-3 family (ChrisHMSFT, 2022). Specifically, text-davinci-002 was used as this was the best available

GPT-3 model at the time of writing (ChrisHMSFT, 2022).

The hyperparameters, which control how the language model learns from the training data, were specified as follows:

- Number of epochs: 4
- Batch size: 4
- Learning rate: 0.1
- Weight loss: 0.1

As finding the optimal hyperparameters is a complex job we resort to the default hyperparameters from the documentation (Brockman et al., 2016). This means that the fine-tuning was performed with a batch size of 4, a learning rate of 0.1 and a weight loss of 0.1. The final hyperparameter, the number of epochs, is optimally found according to the training loss during the fine-tuning. However, due to the high costs of using GPT-3 but more importantly previous work showing promising results with just a few epochs, we decided to terminate the training after four epochs. One reason why GPT-3 may only need a few epochs for fine-tuning is because it is pre-trained on a massive dataset, which means it has already learned a wide range of general language patterns and structures.

5.1 Inferencing with GPT-3

After fine-tuning, the model was applied to generate lyrics in a similar style and structure as those in Højskolesangbogen. Similar to the training phase, the prompts came from 10 actual songs extracted from the original song corpus. The prompts were the first verses of each of these songs (see appendix A.1). These prompts are used to generate 10 completions in the form of two subsequent verses for each song. Sanity checks were performed on each song to verify that the lyrics are similar to real lyrics in semantics and structure but otherwise no modifications were done to the lyrics.

6 Results(JW)

In the following, we will only present the results most relevant to our discussion, however, the full text from all 10 generated songs can be seen in Appendix A.1. Figure 3 shows two word clouds visualizing the differences between the original ten songs used as prompts for GPT-3 and the ones GPT-3 generated.

Det er hvidt herude
1. Det er hvidt herude,
kyndelmisse slår sin knude
overmåde hvas og hård,
hvidt forneden, hvidt foroven,
pudret tykt står træ i skoven

som udi min abildgård.

Generated continuation
2. Det er hvidt herude,
hvide huer over hoveder,
hvide hænder over nødder,
hvide vogne på de vogne,
hvide marker og hvide veje,
hvide skyer i den blåe.

3. Det er hvidt herude,
hvide vadestens-fodtrin står,
hvide træer i den skov,
hvide kornmarker til jorden,
hvide sæler i det hav,
hvide soler i den blåe.

Vem kan segla förutan vind
1. Vem kan segla förutan vind?
Vem kan ro utan åror?
Vem kan skiljas från vännen sin
utan att fälla tårar?

Generated continuation
2. Vem kan lita på löftesord,
om intet sinne därtill?
Vem kan njuta av frid och ro,
om intet hjärta är stilla?

3. Vem kan finna nåd utan synd,
när hjärtat är vårt fängelse?
Vem kan lämna sitt hus och sin gård
utan att drabbas av katastrof?

Vær velkommen Herrens år
1. Vær velkommen, Herrens år,
og velkommen herhid!
Julenat, da vor Herre blev fød,
da tændte sig lyset i mørkets skød.
Velkommen, nytår, og velkommen her!

Generated continuation
2. Vær velkommen, nu tiden er kort,
vi skal glædes og græde og tordne løs.
Vær velkommen, mens tiden er god,
nu åbner sig Gud, hvad vi bør tage imod.
Velkommen nytår og velkommen her

3. Vær velkommen, nytårsnat,
da hjertet blev mildt og varmt!
Da klangen rørtes fra englehær,
da rosenrøde viste sig der.
Velkommen, nat, da engle kom!

Jens Vejmand
1. Hvem sidder dér bag skærmen
med klude om sin hånd,
med læderlap for øje
tog om sin sko et bånd?
Det er såmænd Jens vejmand,
der af sin sure nød
med hamren må forvandlede hårde sten til brød.

Generated continuation
2. Hvem sidder dér bag skærmen
med sorte strimler på,
med gylden knap for øjet
sin kappe, hård og blå
Det er såmænd den store,
høje, sorte dame,
som på det herlige kort
har arbejdet på.

3. Hvem sidder dér bag skærmen
og presser næven i,
og slår den i klaveret,
så spillerne forsvinder?
Det er såmænd det onde,
som jeg inde bær,
og som jeg kan ikke trække
ud af min krop igen.

Amazing grace
1. Amazing grace how sweet the sound
That saved a wretch like me!
I once was lost, but now I'm found,
Was blind but now I see.

Generated continuation
2. The Lord has promised good to me,
His word my hope secures;

Figure 3: Figure showing wordcloud from A) the following two verses of the original ten songs for which the first verse was used as prompt for GPT-3 and B) a wordcloud of the two verses generated by GPT-3

He will my shield and portion be
As long as life endures.
3. Oh, when this flesh and heart shall fail,
And mortal life shall cease,
I shall possess, within the veil,
A life of joy and peace.

6.1 Evaluation of the quality

For some songs, the finetuned version of GPT-3 performs really well. Let us start by looking at some of the Danish songs. Firstly, the generated songs all resemble the structure of the verse given as prompts. It also seems to catch some of the central terms and repeat them at appropriate places. For example "Vær velkommen" is repeated in the first line of every verse in "Vær velkommen Herrens år", and "Hvem sidder der bag skærmen" is also repeated in the first line of every verse in "Jens Vejmand". From a lyrical and rhythmic perspective, this seems to work pretty well. However, in "Det er hvidt herude" GPT-3 focuses way too much on using the wording "hvide" or "hvidt". It ends up using one of the two in each sentence, and it makes the song a bit repetitive.

To see the abilities of GPT-3 in another language, we tested GPT-3 on a Swedish and an English song - both of which are present in *Højskolesangbogen*. The continuation of "Vem kan segla forütan vind" looks very well. Again, GPT-3 uses the repetition of a central structure from the first verse throughout the second and third verses. It repeats the "Vem kan" while also constructing the lines as questions, which is a central feature of the first verse.

GPT-3's continuation of "Amazing Grace" is impressive. It is clear that it performs much better in English than in a low-resource language such as Danish. First of all, the structure mimics the original verse completely. So much, so that the generated verses fit perfectly with the original melody. Secondly, the generated verses actually rhyme in the same structure as the original verse, i.e. it follows an A-B-A-B structure so first and third line rhymes and second and fourth line rhymes. Thirdly, without analysing and interpreting the metaphors too much we can still say that they make more sense in the context of the first verse.

7 Discussion (NK)

7.1 Possible Improvements

In the following sections, we have outlined a few ideas for improvements and further research on this topic.

Naturally, the qualitative evaluation of the songs is problematic as it potentially introduces human biases into the assessments. Future studies should other people's evaluations of the generated lyrics. A few ideas are to see 1) how they would rate the quality of the song lyrics, and 2) how easy they are to sing. Secondly, we would also like to investigate, whether people can actually see from the generated verse that it is generated by GPT-3. One could present people with a range of songs, some of them entirely real, some of them with second and third verses generated by GPT-3. The participants are then to guess, whether a text is AI-generated or if it's original. A third option could be to present participants with both the original text and the generated text, and then they should guess which is original and which is generated.

All of these possibilities would say something about the quality of the lyrics generated. Or at least it would say something about GPT-3's abilities to generate lyrics that are good enough to mislead regular people.

Another interesting possibility could be to thoroughly compare the generated text and the original text. Are there any similarities or differences? It would be just as interesting to find similarities as differences.

8 Conclusion (JW and NK)

This project has explored the possibilities of using large language models, specifically GPT-3, to generate lyrics for songs of cultural content. By fine-tuning GPT-3 on the Danish "Højskolesangbogen," we were able to generate lyrics that were similar in style and content to those found in the "Højskolesangbogen."

Overall, our results show that large language models have the potential to generate high-quality lyrics that capture the essence and cultural significance of a particular songbook or music tradition. While further research is needed to fully understand the capabilities and limitations of these models in this context, our findings suggest that they could be a useful tool for songwriters and musicians looking to create new works inspired by specific cultural traditions.

Additionally, our project highlights the value of using scraped data as a source for training large language models. By using a dataset of lyrics scraped from the internet, we were able to fine-tune GPT-3 on a large and diverse collection of songs, which likely contributed to the model's ability to generate lyrics that were similar in style and content to those found in the "Højskolesangbogen."

With GPT-4 just around the corner, it is likely that we will see more improvements in lyrics generation such as adding nodes or chords to the songs.

References

Alammar, J. (2018). *The Illustrated Transformer.* Retrieved from `https://jalammar.github.io/illustrated-transformer/`

Ammu, Bhaskar. (2020, August). *GPT-3: All you need to know about the AI language model.* Retrieved 2022-12-28, from `https://www.sigmoid.com/blogs/gpt-3-all-you-need-to-know-about-the-ai-language-model/` (Section: Machine Learning)

Bache, S. M., & Wickham, H. (2022). *magrittr: A Forward-Pipe Operator for R.*

Bisong, E. (2019). *Building Machine Learning and Deep Learning Models on Google Cloud Platform.* Retrieved 2022-11-29, from `https://link.springer.com/book/10.1007/978-1-4842-4470-8`

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Amodei, D. (2020). Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 1877–1901). Curran Associates, Inc. Retrieved from `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014, October). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.* arXiv. Retrieved 2022-12-09, from `http://arxiv.org/abs/1409.1259` (arXiv:1409.1259 [cs, stat]) doi: 10.48550/arXiv.1409.1259

ChrisHMSFT. (2022). *Azure OpenAI models - Azure OpenAI.* Retrieved 2022-12-28, from `https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/models`

Faurholt, J. (2021). *Højskolesangbogen.*

Gudivada, V. N., & Arbabifard, K. (2018, January). Chapter 3 - Open-Source Libraries, Application Frameworks, and Workflow Systems for NLP. In V. N. Gudivada & C. R. Rao (Eds.), *Handbook of Statistics* (Vol. 38, pp. 31–50). Elsevier. Retrieved 2022-12-09, from `https://www`

.sciencedirect.com/science/article/pii/S0169716118300221 doi: 10.1016/bs.host.2018.07.007

Harris, C. R., Millman, K. J., Walt, S. J. v. d., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020, September). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Retrieved from https://doi.org/10.1038/s41586-020-2649-2 (Publisher: Springer Science and Business Media LLC) doi: 10.1038/s41586-020-2649-2

Hemingway, A. S. H., Colette. (2001). *Music in Ancient Greece | Essay | The Metropolitan Museum of Art | Heilbrunn Timeline of Art History.* Retrieved 2022-12-29, from https://www.metmuseum.org/toah/hd/grmu/hd_grmu.htm

Lek, S., & Park, Y. S. (2008, January). Artificial Neural Networks. In S. E. Jørgensen & B. D. Fath (Eds.), *Encyclopedia of Ecology* (pp. 237–245). Oxford: Academic Press. Retrieved 2022-12-09, from https://www.sciencedirect.com/science/article/pii/B9780080454054001737 doi: 10.1016/B978-008045405-4.00173-7

Lin, X. V., Mihaylov, T., Artetxe, M., Wang, T., Chen, S., Simig, D., ... Li, X. (2022, November). *Few-shot Learning with Multilingual Language Models.* arXiv. Retrieved 2022-12-29, from http://arxiv.org/abs/2112.10668 (arXiv:2112.10668 [cs]) doi: 10.48550/arXiv.2112.10668

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. v. d. Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56 – 61). doi: 10.25080/Majora-92bf1922-00a

R Core Team. (2022). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

RStudio Team. (2020). *RStudio: Integrated Development Environment for R.* Boston, MA: RStudio, PBC. Retrieved from http://www.rstudio.com/

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017, December). *Attention Is All You Need.* arXiv. Retrieved 2022-12-07, from http://arxiv.org/abs/1706.03762 (arXiv:1706.03762 [cs]) doi: 10.48550/arXiv.1706.03762

Wickham, H. (2022a). *rvest: Easily Harvest (Scrape) Web Pages.* Retrieved 2022-12-28, from https://rvest.tidyverse.org/authors.html

Wickham, H. (2022b). *stringr: Simple, Consistent Wrappers for Common String Operations.*

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. doi: 10.21105/joss.01686

Xie, Y. (2022). *bookdown: Authoring Books and Technical Documents with R Markdown.* Retrieved from https://github.com/rstudio/bookdown

A Appendix

A.1 AI generated lyrics

Det er så yndigt at følges ad

Det er så yndigt at følges ad
for to, som gerne vil sammen være,
da er med glæden man dobbelt glad
og halvt om sorgen så tung at bære;
ja, det er gammen //:
at rejse sammen, ://
når fjederhammen //:
er kærlighed. ://

Generated continuation

2. Det er så yndigt at følges ad for to,
når lyset slukkes af dagen,
og øjet vil lukkes med længsel
efter den, som er tæt ved ens siden;
ja, det er gammen //:
at rejse sammen, ://
når fjederhammen //:
er kærlighed. ://
3. Det er så yndigt at følges ad
for to, når vejen er brat,
så er hjælp hos hinanden at finde;
ja, det er gammen //:
at rejse sammen, ://
når fjederhammen //:
er kærlighed. ://

Det er hvidt herude

Det er hvidt herude,
kyndelmisse slår sin knude
overmåde hvas og hård,
hvidt forneden, hvidt foroven,
pudret tykt står træ i skoven
som udi min abildgård.

Generated continuation

2. Det er hvidt herude,
hvide huer over hoveder,
hvide hænder over nødder,
hvide vogne på de vogne,
hvide marker og hvide veje,

hvide skyer i den blåe.
3. Det er hvidt herude,
hvide vadestens-fodtrin står,
hvide træer i den skov,
hvide kornmarker til jorden,
hvide sæler i det hav,
hvide soler i den blåe.

Der er noget i luften

Der er noget i luften, jeg ved ikke hvad,
som forår, skønt skoven har mistet hvert blad.
Der er noget i luften, som rosernes duften,
som fuglenes fryd,
skønt rosen er falmet, og fuglen er draget mod syd.

Generated continuation

2. Der er noget i luften, som ved, hvad jeg har savnet,
som sørger på tavshed og kommer med lov.
Der er noget i luften, som stiger af graven,
som hvisker mig i øret: ”Jeg lever endnu!”
som våren og høsten, kan jeg også græde.
3. Der er noget i luften, som blomsterne fred,
skønt grønne skove kan græde over det grønne blad.
Der er noget i luften, som giver mig trøst,
som vinter og som forår,
skønt vinteren er slut og foråret er genvundet sin plads

Vær velkommen Herrens år

Vær velkommen, Herrens år,
og velkommen herhid!
Julenat, da vor Herre blev fød,
da tændte sig lyset i mørkets skød.
Velkommen, nytår, og velkommen her!

Generated continuation

2. Vær velkommen, nu tiden er kort,
vi skal glædes og græde og tordne løs.
Vær velkommen, mens tiden er god,
nu åbner sig Gud, hvad vi bør tage imod.
Velkommen nytår og velkommen her
3. Vær velkommen, nytårsnat,
da hjertet blev mildt og varmt!
Da klangen rørtes fra englehær,
da rosenrøde viste sig der.
Velkommen, nat, da engle kom!

Vem kan segla förutan vind

Vem kan segla förutan vind?
Vem kan ro utan åror?
Vem kan skiljas från vännen sin
utan att fälla tårar?

Generated continuation

2. Vem kan lita på löftesord,
om intet sinne därtill?
Vem kan njuta av frid och ro,
om intet hjärta är stilla?
3. Vem kan finna nåd utan synd,
när hjärtat är vårt fängelse?
Vem kan lämna sitt hus och sin gård
utan att drabbas av katastrof?

Danmark nu blunder den lyse nat

Danmark, nu blunder den lyse nat
bag ved din seng, når du sover.
Gøgen kukker i skov og krat,
Vesterhavet og Kattegat
synger, imens det dugger,
sagte som sang ved vugger.

Generated continuation

2. Tørres så blødt i den danske vår,
høres så sødt i den sommer.
Vugger dig frem til det lyse vår,
lyse som sommersangen.
Gøgen kukker i skov og krat,
Vesterhavet og Kattegatsynger,
3. Hver nat, hvor du sover så blødt,
sommer og vår dig vugger.
Tørres så blødt i den danske vår,
høres så sødt i den sommer.
Vugger dig frem til sommer
Det er den danske sommer

En rose så jeg skyde

En rose så jeg skyde
op af den frosne jord,
alt som os fordum spå'de
profetens trøsteord.
Den rose spired frem
midt i den kolde vinter

om nat ved Betlehem.

Generated continuation
2. I en sti da jeg gik,
jeg så en rosenfag,
jeg kyssed den og fægted
og båren den i min favn.
Min sjæl af den blev lys,
og min sti blev stadig kort,
jeg ånded rosenfag,
3. O, om jeg alene
fandt denne rose frem,
om jeg alene fik
den i mit hjertebrændt!
O, om jeg alene fik den til at gro,
og til at blomstre, og
få åndens rosenmagt!

2. Min rose står i jorden, i døde ting og grus, der findes livets lærdom, i hjertet trøst og nus.

Jens Vejmand
Hvem sidder dér bag skærmen
med klude om sin hånd,
med læderlap for øje
tog om sin sko et bånd?
Det er såmænd Jens vejmand,
der af sin sure nød
med hamren må forvandlede hårde sten til brød.

Generated continuation
2. Hvem sidder dér bag skærmen
med sorte strimler på,
med gylden knap for øjet
sin kappe, hård og blå
Det er såmænd den store,
høje, sorte dame,
som på det herlige kort
har arbejdet på.
3. Hvem sidder dér bag skærmen
og presser næven i,
og slår den i klaveret,
så spillerne forsvinder?
Det er såmænd det onde,
som jeg inde bær,
og som jeg kan ikke trække
ud af min krop igen.

Det var en lørdag aften
//: Det var en lørdag aften, jeg sad og vented dig, - ://
//: du loved mig at komme vist, men kom dog ej til mig! ://

Generated continuation
3 //: Og en søndag aften jeg sad og vented dig, - ://
//: du loved mig at komme vist, men kom dog ej til mig! ://
4 //: Og en mandag aften jeg sad og vented dig, - ://
//: du loved mig at komme vist, men kom dog ej til mig! ://
5 //: Og en tirsdag aften jeg sad og vented dig, - ://
//: du loved mig at komme vist, men kom dog ej til mig! ://
6 //: Og en onsdag aften jeg sad og vented dig, - :// //: du loved mig at komme vist, men kom dog ej til ://

Amazing grace
Amazing grace how sweet the sound
That saved a wretch like me!
I once was lost, but now I'm found,
Was blind but now I see.

Generated continuation
4. The Lord has promised good to me,
His word my hope secures;
He will my shield and portion be
As long as life endures.
5. Oh, when this flesh and heart shall fail,
And mortal life shall cease,
I shall possess, within the veil,
A life of joy and peace.