UNIVERSITY OF
LUCERNE

# The ABC of Computational Text Analysis

## #5 Basic NLP with Command-line

Alex Flückiger
Faculty of Humanities and Social Sciences
University of Lucerne

31 March 2023

# Recap last Lecture

- perform shell commands 🕹️
  - navigate filesystem
  - create/copy/move/remove files
- complete assignment ✍️

# Get around in your filesystem 

```
.
├── README.md
└── lectures
    ├── images
    │   └── ai.jpg
    └── md
        ├── KED2023_01.md
        └── KED2023_02.md
```

Example location of the course material: `/home/alex/KED2023`

- `pwd` get the path to the current directory
- `cd ..` go one folder up
- `cd FOLDERNAME` go one folder down into FOLDERNAME
- `ls -l` see the content of the current folder

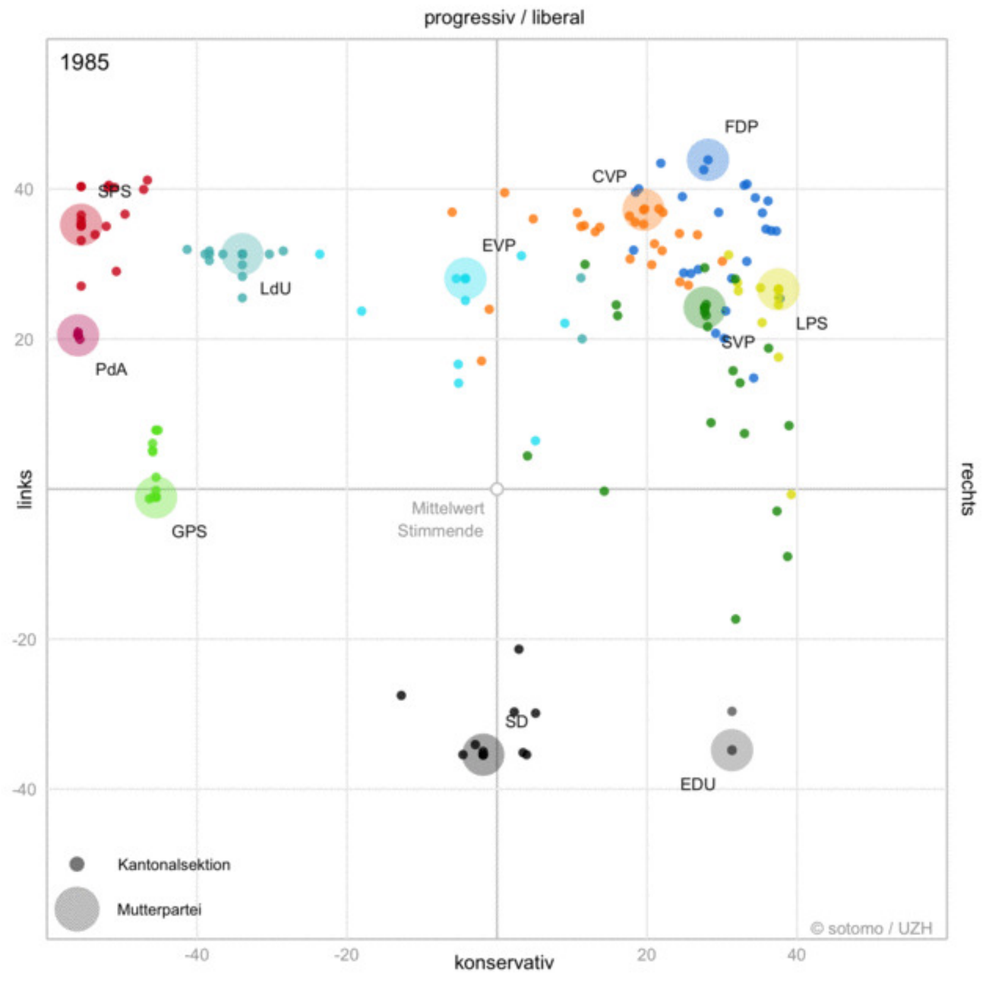
# Outline

- corpus linguistic using the shell 🔪

  counting, finding, comparing

- analyzing programmes of Swiss parties 📊

# When politics changes, language changes.




*historical development of Swiss party politics (Tagesanzeiger)*

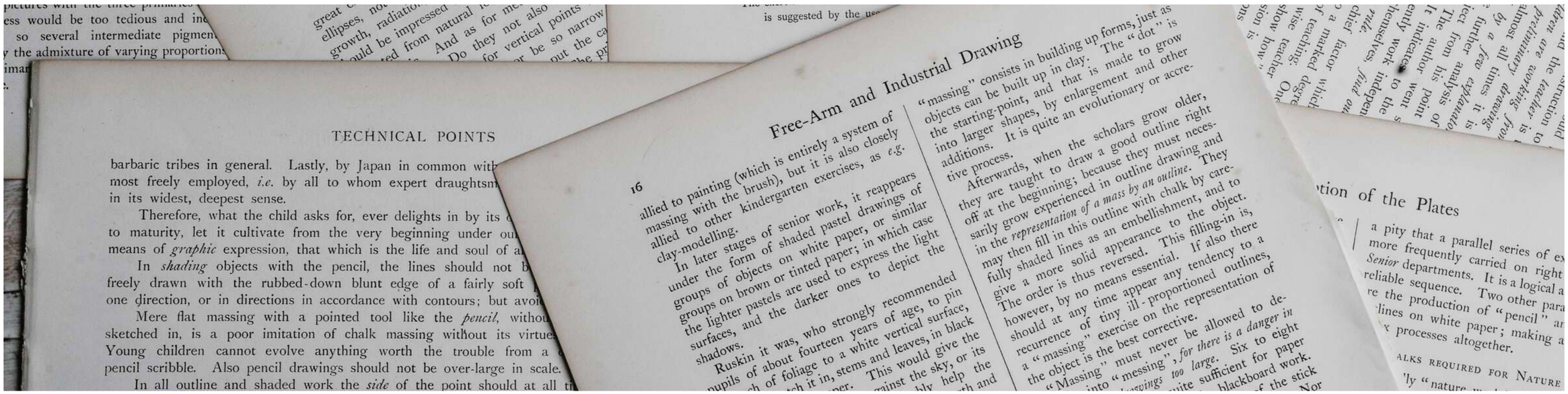
# Processing a Text Collection

1. each document as individual file (`.txt`)

    use shell for quick analysis

2. a dataset of documents (`.csv`, `.tsv`, `.xml`)

    use Python for indepth analysis




*Processing a collection of documents (src)*

# Counting Things

# Frequency Analysis

- frequency ~ measure of relevance
- bag of words approach
- simple
- powerful


*text as a bag of words (src)*

# Key Figures of Texts

```
wc *.txt     # count number of lines, words, characters
```

# Word Occurrences

## show in context

```
egrep -ir "computational" folder/        # search in all files in fol

# common egrep options:
# -i              search case-insensitive
# -r              search recursively in all subfolders
# --colour        highlight matches
# --context 2     show 2 lines above/below match
```

## count words

```
egrep -ic "big data" *.txt       # count across all txt-files, ignore
```

# Word Frequencies

## steps of the algorithm

1. split text into one word per line (tokenize)
2. sort words alphabetically
3. count how often each word appears

```
# piping steps to get word frequencies
cat text.txt | tr " " "\n" | sort | uniq -c | sort -h > wordfreq.txt

# explanation of individual steps:
tr " " "\n"      # replace spaces with newline
sort -h          # sort lines alphanumerically
uniq -c          # count repeated lines
```

# Word Frequencies

- absolute frequency
- relative frequency

  `= n_occurrences / n_total_words`

  independent of size
- **statistical validation of variation**

  significance tests between corpora

# Convert Stats into Dataset

- convert to `.tsv` file
- useful for further processing
  e.g., import in Excel

```
# convert word frequencies into tsv-file
# additional step: replace a sequence of spaces with a tabulator
cat text.txt | tr " " "\n" | sort | uniq -c  | sort -h | \
tr -s " " "\t"  > test.tsv
```

# In-class: Matching and counting

1. Print the following sentence in your command line using `echo`.

```
echo "There are a few related fields: NLP, computational linguistics,
```

2. How many words are in this sentence? Use the pipe operator to combine the comm
3. Match the words `computational` and colorize its occurences in the sentence usi
4. Get the frequencies of each word in this sentence using `tr` and other commands.

# Preprocessing

# Common Preprocessing

## Refining results with

- lowercasing
- replace symbols
- join lines
- trimming header + footer
- splitting into multiple files
- using patterns to remove/extract parts 

# Lowercasing

## reduce word forms

```
echo "ÜBER" | tr "A-ZÄÖÜ" "a-zäöü"  # fold text to lowercase
```

# Removing and Replacing Symbols

```
echo "3x3" | tr -d "[:digit:]"      # remove all digits
cat text.txt | tr -d "[:punct:]"    # remove punctuation like .,:;?!

tr "Y" "Z"                          # replace any Y with Z
```

# Standard Preprocessing

## save a preprocessed document

```
# lowercase, no punctuation, no digits
cat speech.txt | tr "A-ZÄÖÜ" "a-zäöü" | \
tr -d "[:punct:]" | tr -d "[:digit:]" > speech_clean.txt
```

# Join Lines

```
cat test.txt | tr -s "\n" " "    # replace newlines with spaces
```

# Trim Lines

```
cat -n text.txt          # show line numbers
sed "1,10d" text.txt     # remove lines 1 to 10
```

# Splitting Files

```
# splits file at every delimiter into a stand-alone file
csplit huge_text.txt  "/delimiter/" {*}
```

# Check Differences between Files

sanity check after modification

```
# show differences side-by-side and only differing lines
diff -y --suppress-common-lines text_raw.txt text_proc.txt
```

Where there is a shell,
there is a way. 

# Organizing Code

- Git tracks file changes and allows for version management
- GitHub is a popular hosting platform based on Git
  - share code and collaborate
  - repository = project folder

🤓 Published code and data are parts of the endeavour of open science.

# Questions?

# In-class: Getting ready

1. Change into your local copy of the GitHub course repository KED2023 and update it with `git pull`. When you haven't cloned the repository, follow section 5 of the installation guide .

   You find some party programmes (Grüne, SP, SVP) in `materials/data/swiss_party_programmes/txt`. The programmes are provided in plain text which I have extracted from the publicly available PDFs.

2. Have a look at the content of some of these text files using `more`.

# In-class: Analyzing Swiss Party Programmes I

1. Compare the absolute frequencies of single terms or multi-word expressions of your choice (e.g., Ökologie, Sicherheit, Schweiz)…

   across parties

   historically within a party

   Use the file names as filter to get various aggregation of the word counts.

2. Pick terms of your interest and look at their contextual use by extracting relevant passages. Does the usage differ across parties or time?

**Share your insights with the class using Etherpad.**

# In-class: Analyzing Swiss Party Programmes II

1. Convert the word frequencies per party into a `tsv` dataset. Compute the relative word frequency instead of the absolute frequency using any spreadsheet software (e.g. Excel). Are your conclusions still valid after accounting for the size?
2. Can you refine the results with further preprocessing of the data?
3. What is the size of the vocabulary of this data collection (number of unique words)?

**Pro Tip** 🤓: Use `egrep` to look up commands in the `.md` course slides

# Additional Resources

When you look for useful primers on Bash, consider the following resources:

- Tutorial Basic Text Analysis by W. Turkel
- Tutorial Pattern Matching + KWIC by W. Turkel