The ABC of computational Text Analysis

05: Basic NLP with Command-line

Alex Flückiger
26 March 2020

Recap last Lecture

- using shell for file handling
- complete assignment

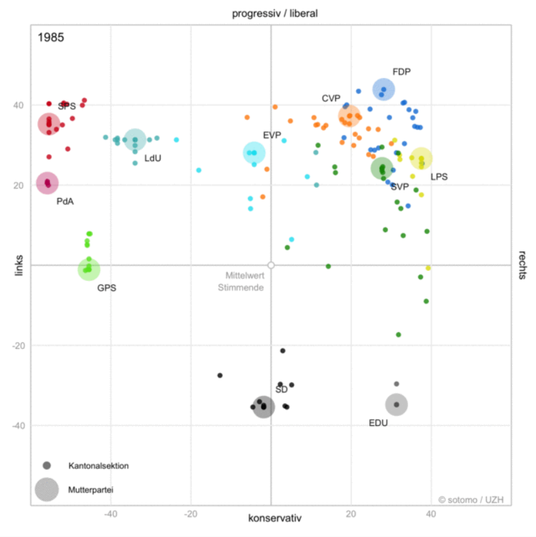
Outline

- 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)

Starting Point

- each text as separate file ✅

  start with shell

- document collection as dataset

  proceed with Python?

Counting Things

Frequency Analysis

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

Key Figures of Texts

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

Word Occurrences

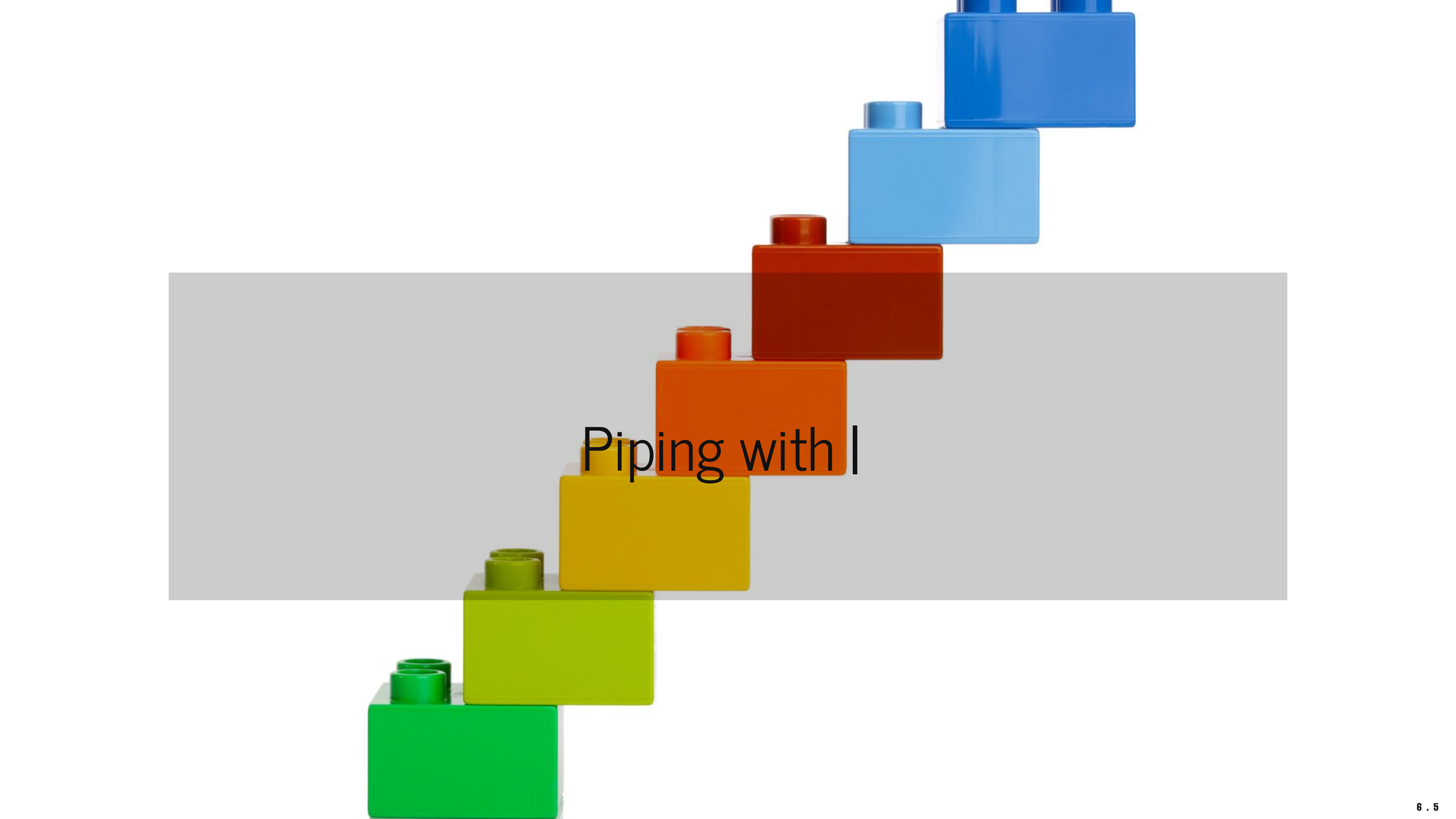
show in context

```
grep -ir 'data'          # search in current folder, ignore case

# other options
# --colour to highlight pattern
# --context 2 to show 2 lines above/below
```

count words

```
grep -ic 'data' */*.txt # count in all txt-files, ignore case
```



Piping with |

Frequencies for all Words

steps of algorithm

1. split text into one word per line (tokenize)
2. sort words
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                 # sort lines alphanumerically
uniq -c              # count repeated lines
```

Word Frequencies

- absolute frequency
- relative frequency

  rel frq = occurrences / total_words

  independent of size
- statistical validation of variation

  significance tests between corpora

Convert Stats into Dataset

- convert to tsv
- useful for further processing

  export to 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 -n | \
tr -s ' ' '\t'  > test.tsv
```

Preprocessing to refine Results

Common Preprocessing

- 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 preprocessed documents

```
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
```

Trimming

```
more -N text.txt        # show line numbers
sed '1,10d' text.txt    # remove the first 10 lines
```

Splitting

```
# 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 

Questions

In-class: Getting ready

1. Update your local copy of the Github repository KED2020 with `git pull`. You find some party programmes (Grüne, SP, SVP) in `materials/party_programmes`. The programmes are provided in plain text which I have extracted from the official PDFs.
2. Fool around with individual commands to get a feeling of them before you proceed with the actual analysis.

In-class: Analyzing Swiss Party Programmes I

1. Compare the absolute frequencies of single terms or multi-word expressions …

   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 from 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. Get the number of unique words rather than the total number of words of a document (piping).

Pro Tip 🤓: Use `grep` to look up commands in the 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