The ABC of computational Text Analysis

07: Data Sources and Ethics

Alex Flückiger
9 April 2020

Outline

- deepen the understanding of regex
- learn about available data resources
- think about ethics + bias → next session

Ready for some interaction?

Join the fun 

access with a code **51 63 60**: https://www.menti.com

access directly: https://www.menti.com/sqs8kqsf6g

Recap last Lecture

- regex to extract + replace particular text
- match text with generalized patterns
- 2 types of symbols

  literal: `abc`

  meta: `\w \s [^abc] *`

Replacing + Removing

stream editor (sed)

- advanced find + replace using regex

  `sed "s/WHAT/WITH/g" FILE`

- `sed` replaces any sequence, `tr` only single symbols

```
echo "hello" | sed "s/llo/y/g"      # replace "llo" with a "y"

# by setting the g flag in "s/llo/y/g",
# sed replaces all occurences, not only the first one
```

Contextual Replacing

- reuse match with grouping `(pattern)`
  - `\1` *equals the expression inside first pair of parentheses*
  - `\2` *expression of second pair*
  - ...

```
# swap order of name (last first -> first last)
echo "Lastname Firstname" | sed -E "s/(.*) (.*)/\2 \1/"

# egrep also supports grouping
# match any two digit pair
egrep -r "([0-9])\1([0-9])\2"
```

Even more Symbols

- `\b` word boundary

  `word\b` *does not match* `words`

- `^` begin of line and `$` end of line

  `^A` *matches only* `A` *at line start*

- `|` disjunction (OR)

  `(Mr|Mrs|Mr\.|Mrs\.) Green` *matches alternatives*

Something actually useful

combining regular expressions with frequency analysis

```
# get political areas
# by counting words ending with "politik"
egrep -rioh '\w*politik\b' */*.txt | sort | uniq -c | sort -h

# get ideologies/concepts
# by counting words ending with -ismus
egrep -rioh '\w*ismus\b' */*.txt | sort | uniq -c | sort -h
```

Greediness Trap

- greedy ~ match the longest string possible
- quantifiers `*` or `+` are greedy
- non-greedy by excluding some symbols

  `[^EXCLUDE_SYMBOLS]` *instead of* `.*`

```
# greedy: an apple, other apple
echo 'an apple, other apple' | egrep 'a.*apple'

# non-greedy: an apple
echo 'an apple, other apple' | egrep 'a[^,]*apple'
```

Better Tokenization

- tokenization ~ splitting into words

```
# new, improved approach
cat text.txt | tr -sc "[a-zäöüA-ZÄÖÜ0-9-]" "\n"

# old approach
cat text.txt | tr ' ' '\n'
```

Questions

In-class: Game

1. Make sure that your local copy of the Github repository KED2020 is up-to-date with `git pull`. Go to the party programmes in `materials/party_programmes/txt`.
2. Use `egrep` to extract all uppercased words like `UNO, OECD, SP` and count their frequency.
3. Post the most frequent abbreviation from 2) on www.menti.com.

```
# Some not so random hints
piping with |
sort
uniq -c
egrep -roh */*.txt
```

In-class: Exercises I

1. Use `egrep` to extract words following any of these strings: `der die das`.

2. Do the self-check on the next slide.

3. Use `sed -E` to remove the table of content, the footer and the page number in the programme of the Green Party. Check the PDF to get a visual impression and test your regular expression with `egrep` to see if you match the correct parts in the document.

In-class: Self-Check

equivalent patterns

```
a+ == aa*                    # "a" once or more than once
a? == (a|_)                  # "a" once or nothing
a{3} == aaa                  # three "a"
a{2,3} == (aa|aaa)           # two or three "a"
[ab] == (a|b)                # "a" or "b"
[0-9] == (0|1|2|3|4|5|6|7|8|9)  #any digit
```

In-class: Exercise II

1. Since you know about RegEx, we can use a more sophisticated tokenizer to split a text into words. What is the difference between the old and new approach? Test it and check the helper page with `man`.

```
# new, improved approach
cat text.txt | tr -sc "[a-zäöüA-ZÄÖÜ0-9-]" "\n"
# old approach
cat text.txt | tr ' ' '\n'
```

In-class: Exercise III

1. Count all the bigrams (sequence of two words) using character sets and quantifiers. What about trigrams (three words)?
2. Extract the words following numbers (also consider numbers like: `1'000, 1,000 or 5%`). Then, count all the words while excluding the numbers themselves. Hint: Pipe another grep to remove the digits.
3. You are ready to come up with your own patterns...

Data Sources

Forms of Data

- content data

  clean, plain text data
  preferable `.txt`

- metadata ~ information about the actual data

  publishing date, authors, source, version
  preferable as `.csv`

What data sources are there?

- broadly social
  - *newspapers + magazines*
  - *websites + social media*
  - *reports by NGOs/GOs*
- scientific
  - *journals*
- economic
  - *business plans/reports*
  - *contracts*
  - *patents*
- any textual source…

Interesting Publishers

- Nexis Uni

  newspaper, business + legal reports (international)
  licensed by the university

- HathiTrust

  massive collection of books (international)
  open, requires agreement

- Project Gutenberg

  huge collection of books (international)
  open-access

- JSTOR

  scientific articles across disciplines
  requires account, only 1-3-grams

Dataset Search

- Harvard Dataverse

  free scientific data repository
- Google Dataset Search

  Google for datasets basically
- corpora by the Department of Computational Linguistics @ UZH

🤓 search for your topic followed by *corpus* or *text collection*

Some great historical Corpora

ready off the shelf, machine-readable

- 1 August speeches by Swiss Federal Councillors

  ask me for access to this corpus
- Human Rights Texts
- United Nations General Debate Corpus

Online Computational Text Analysis

- Impresso

  many historical newspapers + magazines (LU, CH)
  requires account

- bookworm HathiTrust

  great filtering by metadata
  credible scientific source

- Google Ngram Viewer

  no filtering option
  useful for quick analysis

Copyright

- restricted access to high quality data 
- check the rights to process the data

ToDo: Next steps

- submit exercise #2
- check out the data resources
- think about the mini-project

  groups
  data


Have a nice
Easter break!

References