KED2020 Exercise 2: RegEx NLP

Alex Flückiger

2020

Requirements

Your submission needs to comply with the following requirements:

- Deadline: 16 April 2020, 23:59 Hand in your solutions via the respective exercise module on OLAT. Keep in mind to submit on time, as the module is only open until midnight.
- Submission: a single, executable shell script.
- Naming schema: *SURNAME*_KED2020_ex_2.sh
  Replace *SURNAME* with your surname.
- Your shell script should look similar to the template provided here. Add all the commands that you have used in this exercise to this script. Additionally, write a short comment for each command that explains what it does.
- Find solutions individually. Ask friends or Google whenever you feel lost. In terms of programming, Google may be your best friend. When you struggle to understand a concept or have technical issues, feel free to post in the forum on OLAT.

1 Parse Information with RegEx

Sometimes when you download newspaper articles from a publisher, they do not provide the data in a well-structured format. There may be only a single file comprising multiple articles, and in which the metadata is not properly separated from the actual content. In the folder of this exercise, you find such a file named `newspaper_articles.txt`. It is a real-world example that makes the use of regular expressions indispensable.

In this task, you systematically extract the metadata from all the articles in this document. In our case, the metadata comes at the beginning of each newspaper article and looks like this:

```
Von BRUNO VANONI, BERN.
591 words
26 February 2004
Tages Anzeiger
TANZ
German
(c) 2004 Tages Anzeiger Homepage Address: https://www.tagesanzeiger.ch/
```

You need to write three commands to extract the following information using regular expressions and to write the resulting output into a new file:

1. number of words (e.g., 591 words, as given in the file, don't count with `wc`)
2. publishing date (e.g., 26 February 2004)
3. names of authors (e.g., Von BRUNO VANONI). Try to write a pattern that doesn't match the location (e.g., Bern).

Write the patterns as specific, generalizable, and simple as possible. Moreover, try to cover all the cases without getting false positives. Covering all the cases may be too hard[1], just push the limit as far as you can. It is more important that you learn about performing RegEx than to get a perfect solution.

To be clear, you need a single command for each of these three subtasks, similar to the one below. If this command fails on your system for any reason, get in touch. There as there may be a technical problem that needs troubleshooting.

```
### EXAMPLE COMMAND
# extract with a pattern and write the matches into a file
egrep -ho "words" newspaper_articles.txt > new.txt
```

Add the commands to your script and write a short comment for each of them.

While coming up with a solution, you may find the following commands useful along the way:

[1] An example that may be hard to match: *Von MIT KRISTA SAGER* SPRACH WERNER BOSSHARDT*

```
### OTHER USEFUL COMMANDS
# have a look at the document
more newspaper_articles.txt
# useful to check what grep matches before you write into a file
egrep -ho "words" newspaper_articles.txt
# egrep does not support the meta-character \d to match any digit
# use [0-9] instead
```

2 Removing Parts of a Document

In the first task, you have extracted relevant metadata. Thus, these parts can be removed from the document to keep only the actual content of the newspaper articles.

Use the same patterns[2] that you have written in the first task. Pipe multiple `sed` commands to remove the pieces in one go and save the final, cleaned output as a new file. Your command should have the same structure as below:

```
# replace the pattern with an empty sequence to remove it
cat name.txt | sed -E "s/pattern//g" | sed -E "s/pattern2//g" > clean.txt
```

Feel free to improve the patterns or to write new ones to remove even more parts of the document in the attempt to clean the data further.

Add the commands to your script and write a short comment for each of them.

3 Test your script

This task is a simple sanity check. You don't need to include it in your submission.

Your deliverable has to be a runnable script comprising all the commands to accomplish the tasks above. To test your script, run it with the commands below. Once you start the script again, it executes all the commands, one after another. Everything should be reconstructed accordingly. If not, correct the script so that it runs through without any issue.

```
# run script
bash script_name.sh
```

4 Feedback

Please answer the following questions at the end of your script. Start your answers with the # symbol to make them comments that are ignored when running the script.

1. Do you have any questions concerning the exercise or the commands?

[2] Reuse the pattern only, not the entire command with egrep.

2. How long did it take to solve this exercise? Give a fair estimation.