Tokenisierung und Frequenzlisten

tr

tr stammt aus den *GNU* Coreutils[3]:

Ein Auszug aus der *Manpage* mit den wichtigsten Funktionen:

```
tr - translate or delete characters

-c, -C, --complement
       use the complement of SET1

-d, --delete
       delete characters in SET1, do not translate

-s, --squeeze-repeats
       replace each input sequence of a repeated character that is
       listed in SET1 with a single occurrence of that character
```

Beispiele dazu:

```
echo "hallo" | tr -d a

hllo
```

Es werden somit alle *a's* aus dem Text gelöscht. Der Befehl:

```
echo "12stefan4545" | tr -d -c '[[:digit:]]'
```

Würde alle Zeichen entfernen, die nicht auf die Zeichenklasse *digit* zutreffen. Da ür wird die *-c* Option verwendet.

Um alle aufeinanderfolgenden Leerzeichen durch ein Leerzeichen zu ersetzen, geht man so vor (man beachte die Option *-s*:

```
echo "   test  x" | tr -s ' ' ' '
```

Um alle Punktuationszeichen und Kontrollsequenzen aus einem Text zu entfernen, verwendet man die Zeichenklassen *punct* und *cntrl*:

```
echo -e ':-!stefan\n' | tr -d '[[:punct:]][[:cntrl:]]'
```

[3]Siehe dazu http://www.gnu.org/software/coreutils/

Suche in Dateien: *grep*

Mittels dem global/regular expression/print (kurz: *grep*) Werkzeug kann muster-basiert in z.B. Texten gesucht werden:

```
NAME
      grep, egrep, fgrep - print lines matching a pattern
```

```
SYNOPSIS
      grep [OPTIONS] PATTERN [FILE...]
      grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
      grep searches the named input FILEs (or standard input if
      no files are named, or if a single hyphen-minus (-) is
      given as file name) for lines containing a match to the
      given PATTERN.
      By default, grep prints the matching lines.

      In addition, two variant programs egrep and fgrep
      are available. egrep is the same as grep -E. fgrep
      is the same as grep -F. Direct invocation as
      either egrep or fgrep is deprecated, but is
      provided to allow historical applications that
      rely on them to run unmodified.
```

Folgende Metazeichen können für einen regulären Ausdruck zum Suchen verwendet werden:

```
A regular expression may be followed by one of several repetition
operators:
?      The preceding item is optional and matched at most once.
*The preceding item will be matched zero or more times.
+The preceding item will be matched one or more times.
{n}    The preceding item is matched exactly n times.
{n,}   The preceding item is matched n or more times.
{,m}   The preceding item is matched at most m times. This is a GNU extension.
{n,m}  The preceding item is matched at least n times, but not more than m times.
```

Dabei können folgende Zeichenklassen ebenfalls verwendet werden:

`[:alnum:], [:alpha:], [:cntrl:], [:digit:], [:graph:], [:lower:], [:print:], [:punct:], [:space:], [:upper:]` und `[:xdigit:]`.

Hinweis: Wird *grep* ohne den Parameter *-E* aufgerufen, der ür die *erweiterten* reg-uläre Ausdrücke steht, so müssen alle Metazeichen wie “(” durch ein Backslash *escapt* werden!

Beispiele

Im folgenden wird dieser Text als Beispiel verwendet (*blog.txt*)

```
Der Fehler ist ein simpler Programmierfehler gewesen, der im Rahmen eines
Forschungsprojektes entstanden ist. T-Systems und BND oder andere
Geheimdienste waren zu keiner Zeit beteiligt und zu meiner späteren
Anstellung bei T-Systems bestand zu keiner Zeit ein Zusammenhang. Dass T-
Systems im RFC genannt wird, liegt an der verspäteten Fertigstellung des
RFCs und es ist üblich, den bei der Fertigstellung aktuellen Arbeitgeber
anzugeben.
```

Es sollen nun alle potenziellen Satzanfänge und Satzenden ausgegeben werden:

```
cat blog.txt | grep -o "\.\s*[[:upper:]][[:lower:][:punct:][:upper:]]*"

. T-Systems

. Dass
```

und

```
cat blog.txt | grep -o "[[:lower:][:upper:]]*\."

ist.

Zusammenhang.

4anzugeben.
```

Um Akronyme zu finden:

```
cat blog.txt | grep -o "[[:upper:]]\{2,\}"

BND

RFC

RFC
```

Mit der Option *-f* können auch mehrere reguläre Ausdrücke aus einer Datei eingele-sen werden:

```
-f FILE, --file=FILE
       Obtain patterns from FILE, one per line. The empty file
       contains zero patterns, and therefore matches nothing.
       (-f is specified by POSIX.)
```

Mit der *-i* Option kann Groß- und Kleinschreibung ignoriert werden:

```
-i, --ignore-case
       Ignore case distinctions in both the PATTERN and the input
       files.
       (-i is specified by POSIX.)
```

Beispiel:

```
cat /etc/os-release | grep -i pretty_name

PRETTY_NAME="Arch Linux"
```

Die *-v* Option erlaubt es einen regulären Ausdruck zu invertieren:

```
-v, --invert-match
            Invert the sense of matching, to select non-matching
            lines.
            (-v is specified by POSIX.)
```

Beispiel:

```
echo -e "Hallo\nStefan\nHallo\nHallo" | grep -vi hallo
        2Stefan
```

Mit der *-n* Option kann die Zeilennummer des Treffers ausgegeben werden:

```
-n, --line-number
Prefix each line of output with the 1-based line number within its
input file. (-n is specified by POSIX.)
```

Beispiel:

```
echo -e "Hallo\nStefan\nHallo\nHallo" | grep -ni stefan
Stefan
```

Die Option *-A NUM* erlaubt es *NUM* Zeilen nach einem Treffer auszugeben:

```
-A NUM, --after-context=NUM
            Print NUM lines of trailing context after matching
            lines. Places a line containing a group separator (--)
            between contiguous groups of matches. With the -o or --
            only-matching option, this has no effect and a warning
            is given.
```

Beispiel:

```
echo -e "Erste Zeile\nZweite Zeile\nStefan\nVierte Zeile" | grep -A 1 -ni \
stefan

3:Stefan
4-Vierte Zeile
```

Die Option *-B NUM* erlaubt es *NUM* Zeilen vor einem Treffer auszugeben:

```
-B NUM, --before-context=NUM
       Print NUM lines of leading context before matching
       lines. Places a line containing a group separator (--)
       between contiguous groups of matches. With the -o or -
       -only-matching option, this has no effect and a
       warning is given.
```

Beispiel:

```
echo -e "Erste Zeile\nZweite Zeile\nStefan\nVierte Zeile" | grep -B 2 -ni \
stefan

1-Erste Zeile
2-Zweite Zeile
3:Stefan
```

agrep

Das Werkzeug *agrep* (ür *approximately*) ermöglicht eine Suche mi els Levenshtein-Abstandes.

```
NAME
      agrep - print lines approximately matching a pattern

SYNOPSIS
      agrep [OPTION]...   PATTERN [FILE]...

DESCRIPTION
      Searches for approximate matches of PATTERN in each FILE or
      standard input. Example: `agrep -2 optimize foo.txt'
      outputs all
      lines in file `foo.txt' that match "optimize" within two
      errors.
                                                                    E
      .g.
      lines which contain "optimise", "optmise", and "opitmize"
      all match
```

Dabei können folgende Optionen ür die Anzahl der Ersetzungen, Ein ügungen und Löschungen verwendet werden:

```
Approximate matching settings:
-D NUM, --delete-cost=NUM
       Set cost of missing characters to NUM.

-I NUM, --insert-cost=NUM
       Set cost of extra characters to NUM.

-S NUM, --substitute-cost=NUM
       Set cost of incorrect characters to NUM.  Note that a
       deletion
       (a missing character) and an insertion (an extra character) together
       constitute a substituted character, but the cost will be the that of
       a deletion and an insertion added together. Thus, if the const of a
       substitution is set to be larger than the sum of the costs of
       deletion and insertion, direct substitutions will never be done.
```

Folgende Beispieldatei wird verwendet (*dict.txt*):

```
Aber
Und
TLS
UDP
BND
RFC
Backdoor
Payload
openssl
Code
Audit
fail
```

Nun können folgende Suchen gemacht werden:

```
agrep "nail" -1 dict.txt
fail
```

```
agrep "TNT" -2 dict.txt
TLS
BND
```

```
agrep "openssh" -2 dict.txt
openssl
```

```
agrep -s "BNT" -2 dict.txt
TLS
BND
Backdoor
```

Quellenangaben und Literaturhinweise

Der Beispieltext ist erreichbar unter http://blog.fefe.de/?ts=adba343f.

Mehr Informationen rund um den Levenshtein Abstand findet man u.a. in dem *In-formation Retrieval* Buch von Manning, Raghavan und Herrn Schütze. Das Kapital dazu ist online ver ügbar unter http://nlp.stanford.edu/IR-book/html/htmledition/edit-distance-1.html.

Nähere Informationen zu *grep* kann man in der Online Dokumentation finden. Dort sind auch sehr viele Beispiele beschrieben, siehe http://www.gnu.org/software/grep/ manual/html_node/index.html.

sed und *awk*

Mittels den Werkzeugen *sed* und *awk* können Zeichenke en sehr umfangreich bearbeitet werden.

Sed stream editor

sed ist ein Werkzeug zum Bearbeiten von Text-Strömen:

```
NAME
       sed - stream editor for filtering and transforming text

SYNOPSIS
       sed [OPTION]... {script-only-if-no-other-script} [input-file]...

DESCRIPTION
       Sed is a stream editor. A stream editor is used to perform
       basic text transformations on an
       input stream (a file or input from a pipeline). While in some ways
       similar to an editor which permits scripted edits
       (such as ed), sed works by making only one pass over the input(s), and is
       consequently more efficient. But it is sed's ability
       to filter text in a pipeline which particularly distinguishes
       it from other types of editors.
```

Wichtige *sed* Optionen sind u.a.:

```
-e script, --expression=script

        add the script to the commands to be executed

-f script-file, --file=script-file

        add the contents of script-file to the commands to be executed
```

Beispiele ür Ersetzungen:

```
echo "Sie gehen und Sie laufen." | sed -e 's/Sie/sie/'

sie gehen und Sie laufen.
```

Dabei wird das erste Vorkommen von “Sie” durch “sie” ersetzt. Sollen alle Vorkommen ersetzt werden, so verwendet man die Option *g* am Ende:

```
echo "Sie gehen und Sie laufen." | sed -e 's/Sie/sie/g'
```

```
sie gehen und sie laufen.
```

Um alle Zeilen, die mit einem Kommentarzeichen beginnen auszugeben, kann man die Option *p* verwenden:

```
echo -e "# Das ist ein Kommentar\nZweite Zeile" | sed -n -e '/^#/p'

# Das ist ein Kommentar
```

Zeichenklassen können selbstverständlich in *sed* ebenfalls verwendet werden:

```
echo "Test . Anfang . Ende." | sed 's/\s\([[:punct:]]\)/\1/g'

Test. Anfang. Ende.
```

awk

awk ist eine Skriptsprache mit der man strukturierte Textdateien auswerten und bear-beiten kann. *awk* setzt sich aus den Anfangsbuchstaben der Nachnamen der Entwick-ler zusammen: Alfred V. Aho, Peter J. Weinberger und Brian W. Kernighan.

awk kann sowohl Befehle von der Kommandozeile aus bearbeiten, als auch eigene Skriptdateien verarbeiten:

```
PROLOG
      This manual page is part of the POSIX Programmer's Manual.
      The Linux implementation of this interface may differ
      (consult the corresponding Linux manual page for details of Linux
      behavior), or the interface may not be implemented on Linux.

NAME
      awk — pattern scanning and processing language

SYNOPSIS
      awk [-F sepstring] [-v assignment]... program [argument...]

      awk [-F sepstring] -f progfile [-f progfile]... [-v assignment]...
           [argument...]
```

Beispiele:

```
echo "1;Peter;Müller;Waldstraße" | awk -F";" '{print $4}'
Waldstraße
```

Per *-F* Argument kann ein Feldtrenner angegeben werden - standardmässig ist dieser auf das Leerzeichen gesetzt. Per *$* Variable kann dann direkt auf das gewünschte Feld zugegriffen werden.

Eine *awk* Skriptdatei kann u.a. so aussehen - das folgende Beispielskript berechnet durchschni liche Wortlänge:

```
# Autor: Stefan Schweter
# Programm: calculateKorpus.awk
# Beschreibung: Berechnung von durchschnittlicher Wortlänge
# Verwendung:

#awk -f calculate_average_wordlength.awk

BEGIN {
print "Durchschnittliche Wortlänge wird berechnet"
FS = " ";
}

{
              char_counter = char_counter + length($1);

}
END {
print "Mittelwert der Wortlänge: " char_counter/NR;
}
```

Zu Testen beispielsweise mit:

```
echo -e "Test\nBeispiel\nhier\nkommt\nes" | awk -f
calculate_average_wordlength.awk
Durchschnittliche Wortlänge wird berechnet
Mittelwert der Wortlänge: 4.6
```

Qellenangaben und Literaturhinweise

Empfehlenswert rund um das Thema *sed* und *awk* ist das *sed & awk* Buch von Dougherty und Robbins. Inhaltsverzeichnis kann hier eingesehen werden: http://shop.oreilly.com/product/9781565922259.do