



University of
Zurich^{UZH}

Institute of Computational Linguistics

Programmiertechniken in der Computerlinguistik I

Herbstsemester 2015

2. Sitzung, 24. Sept. 2015

Martin Volk



University of
Zurich^{UZH}

Institute of Computational Linguistics

Thema: grep und reguläre Ausdrücke

Reguläre Ausdrücke (EN: regular expressions)

- sind Muster über Zeichenketten
- definieren eine einfache formale Sprache



grep

grep und Zählen (count) der Ergebnisse

grep -c pattern file zählt per Datei

grep pattern file | wc -l zählt die
gesamte Ausgabe



grep

grep für ein Muster xx und Anzeige des Kontexts

grep -B 2 xx file Zeige zwei Zeilen des
vorangehenden Kontexts (Before)

grep -A 4 xx file Zeige vier Zeilen des
folgenden Kontexts (After)

grep -C 3 xx file Zeige drei Zeilen des
vorangehenden und folgenden Kontexts (Context)



University of
Zurich^{UZH}

Institute of Computational Linguistics

grep

Gegeben ein Text aus dem Text+Berg-Korpus mit Part-of-Speech Tags und Lemmas im 3-Spalten-Format.

grep VV	findet alle Verben im Deutschen
grep ADJ	findet alle Adjektive im Deutschen
grep ADV	findet alle Adverbien
grep befreien	findet Lemma und Wortform
grep '^befreien'	findet Wortform
grep '^be.*en\t'	findet alle Wortformen mit <i>be...en</i>
grep '\tbe.*en\$'	findet alle Lemmas mit <i>be...en</i>

Page 5



University of
Zurich^{UZH}

Institute of Computational Linguistics

grep

grep -o '^be.*en\t' gibt statt der Zeile nur den Treffer
aus → erlaubt Sortieren und Zählen der Hits

grep --colour '^be.*en\t' markiert den Treffer in der
Zeile

Page 6



grep und Erweiterungen

grep bietet die Grundfunktionalität der Suche mit regulären Ausdrücken.

Erweiterungen von grep

egrep

grep -E ist wie **egrep**

grep -P erlaubt reguläre Ausdrücke wie in Perl. Sehr mächtig! Empfohlen!



grep

Befehl	Erklärung
grep '\tbe..en\$'	findet alle Lemmas mit 2 Zeichen zwischen be und en
grep '\tbe.[aeiou].en\$'	findet alle Lemmas mit Vokal zwischen 2 Buchstaben
grep '\tbe.[äöü].en\$'	findet alle Lemmas mit Umlaut zwischen 2 Buchstaben
grep '^[bg]e..en\t'	findet alle Wortformen mit be..en und ge..en



Wichtig!

Ein regulärer Ausdruck sucht immer den längsten möglichen String (= *greedy pattern matching*).

Unterscheide:

`grep -o '^be.*en\t'` findet alle Wortformen mit *be...en*

und

`grep -o '^be.*en'` findet alle **Zeilen** mit *be...en*



grep

`grep aa`


`grep -i "aa"` Case-insensitive! findet "aa" "AA", "aA", "Aa"

`grep [aeiou][aeiou]` findet alle Vokal-Paare!

`grep [aeiou][aeiou][aeiou]` findet alle Vokal-Tripel!

`grep [aeiou][aeiou][aeiou][aeiou]` findet alle Vokal-Quadrupel! Gibt es wirklich!

`grep '[aeiou]\{4\}'` mit Anf-Zeichen!

**University of
Zurich**^{UZH}
Institute of Computational Linguistics

grep

grep `'^[1234567890]'` findet Wörter, die mit einer Ziffer beginnen

grep `'^[1234567890][a-z]'` findet Wörter, die mit einer Ziffer + einem Buchstabe beginnen (Beachte grosse und kleine Buchstaben und Umlaute)

grep `-E '^\d[a-z]'` Alternative Notation!

Page 11

**University of
Zurich**^{UZH}
Institute of Computational Linguistics

Wichtige Symbole

<code>\n</code>	neue Zeile
<code>\t</code>	Tabulator
<code>\d</code>	Ziffern 0-9 ("digits")
<code>\w</code>	Buchstaben [A-Za-z], Unterstrich und Ziffern
<code>\s</code>	Leerschlag, Tabulator, neue Zeile

Page 12



grep -P

grep -P '(ADJ ADV)'	mit Alternativen
grep -P 'th?al'	findet <i>thal</i> und <i>tal</i>
grep -P 't(h)al'	ist äquivalent zu
grep -P 'th?al'	!!!
grep -P '^schnee*'	ist äquivalent zu
grep -P '^schne+'	!!!



grep -P

grep -P '([aeiou])\1'	findet alle Vokalpaare
Beachte die Wiederaufnahme des ersten Treffers (durch runde Klammern markiert) mit \1	
grep -P '^(\\w)\1'	findet alle Buchstaben- oder Ziffernpaare am Wortanfang.
grep -P '^(\\d)\1(\\d)\2'	findet zwei gleiche Ziffernpaare.



Reguläre Ausdrücke - Eine rekursive Definition

Jedes Zeichen (auch ein leeres) ist ein regulärer Ausdruck.

Beispiel: "a"

Die **Sequenz** (Folge) von regulären Ausdrücken r_1 und r_2 ist ein regulärer Ausdruck.

Beispiel: " $r_1 r_2$ "

Die **Alternative** von regulären Ausdrücken r_1 und r_2 ist ein regulärer Ausdruck.

Beispiel: " $r_1 | r_2$ "

Ein regulärer Ausdruck r mit **Optionalität, ein- oder mehrmaliger Wiederholung** ist ein regulärer Ausdruck.

Beispiel: " r^* "



Reguläre Ausdrücke

$a^+ == aa^*$ (a einmal oder mehrmals)

$a? == a|_$ (a oder nichts)

$a\{3\} == aaa$

$a\{2,3\} == aa|aaa$

$[ab] == a|b$

$\backslash d == 0|1|2|3|4|5|6|7|8|9$



Grenzen Regulärer Ausdrücke

aaabbb ist ein regulärer Ausdruck

a*b* ist ein regulärer Ausdruck

Wir können aber **nicht** allgemein formulieren, dass genauso viele **a** wie **b** erkannt werden sollen. Salopp: Reguläre Ausdrücke haben keine Zähler!

Formal:

aⁿbⁿ mit $n \in \mathbb{N}$ ist **keine** reguläre Sprache!

aⁿbaⁿ mit $n \in \mathbb{N}$ ist **keine** reguläre Sprache!

Aber

aⁿbⁿ mit $n \in \mathbb{N}, n < 10$ ist eine reguläre Sprache!



Reguläre Sprachen

= Sprachen, die mit regulären Ausdrücken definiert werden können.

Formal:

aⁿbⁿ mit $n \in \mathbb{N}$ ist **keine** reguläre Sprache!

Aber

aⁿbⁿ mit $n \in \mathbb{N}, n < 10$ ist eine reguläre Sprache!

aⁿb^m mit $n, m \in \mathbb{N}$ ist eine reguläre Sprache

aⁿb^mc^m mit $n, m \in \mathbb{N}, m < 5$ ist eine reg. Sprache

a^mb⁵c^m mit $m \in \mathbb{N}$ ist **keine** reguläre Sprache!



Reguläre Ausdrücke vs. Wildcards

sind nicht das selbe!

- Wildcard * == beliebige Anzahl Zeichen
- RegEx * == null oder mehr Wiederholungen eines reg. Ausdrucks

- Wildcard ? == ein beliebiges Zeichen
- RegEx ? == null oder 1 Instanz eines reg. Ausdrucks