DH 1: Sprache und Text

Reguläre Ausdrücke

Andreas Blombach, Stephanie Evert
Lehrstuhl für Korpus- und Computerlinguistik
https://www.linguistik.phil.fau.de

Friedrich-Alexander-Universität
Philosophische Fakultät und Fachbereich Theologie

Reguläre Ausdrücke

- Reguläre Ausdrücke (regex/regexp) sind Zeichenketten, die Suchmuster vorgeben
- weit verbreitet für Volltextsuche (z.B. grep) und in vielen Texteditoren implementiert (Emacs, Atom, …)
  - Aufgabe: Finde die Textteile, auf die der Suchausdruck passt

- in Korpusabfragen auf zwei Ebenen eingesetzt:
  - für passende Wortformen und Annotationen (über Einzelzeichen)
    → „matcht“ stets vollständige Zeichenketten ≠ Volltextsuche
  - um lexikalisch-grammatische Muster zu beschreiben (über Tokens)
- verschiedene Implementierungen/Spielarten/„flavours“; wir verwenden PCRE
  - POSIX, PCRE = Perl Compatible Regular Expressions, Python, Oniguruma, …

Web-Oberflächen zum Herumspielen

- CWB Wordlist Explorer:
  http://corpora.linguistik.uni-erlangen.de/cgi-bin/demos/regex/wordlist_explorer.perl

  mit regulären Ausdrücken in Frequenzlisten verschiedener Korpora suchen

- Volltextsuche & RegEx-Debugging
  - https://regexr.com/
  - https://regex101.com/
  - http://regviz.org/ (für JavaScript, nicht PCRE)
  - https://www.debuggex.com/

- Kreuzworträtsel mit regulären Ausdrücken
  https://regexcrossword.com/

PCRE: Perl Compatible Regular Expressions

- `(…)?` = optional (einmal oder keinmal)
- `(…)*` = beliebig oft (auch gar nicht)
- `(…)+` = beliebig oft, aber mindestens einmal
- `(…){3}` = genau dreimal
- `(…){2,4}` = zwischen zwei- und viermal (2, 3, 4)
- `(…){4,}` = beliebig oft, aber mindestens viermal
  - Quantoren beziehen sich auf Ausdruck unmittelbar davor – wenn das mehrere Zeichen sein sollen, Klammern nicht vergessen!
- `(…|…|…)` = Alternativen
- `.` = beliebiges Zeichen (matchall)
  - v.a.: `.?` (optionales Zeichen), `.*` (beliebige Zeichenkette), `.+`
- Sonderzeichen suchen: `\.` = ., `\*` = *, `\?` = ?, `\+` = +, …

PCRE

- `[aeiou]` = *ein* Zeichen aus der Menge
  - `[a-z]` = `[abc … z]` und `[A-Z]` = `[ABC … Z]` (Achtung: keine Umlaute, <ß> usw.)
  - `[0-9]` = `[0123456789]`
- `[^aeiou]` = alles(!) außer `[aeiou]`
- Vordefinierte Zeichenklassen:
  - `\w` = Buchstaben, Ziffern und `_` (word character)
  - `\s` = Leerraumzeichen (Leerzeichen, Tabulator, Zeilenumbruch, sonstige Steuerzeichen)
  - `\d` = Ziffern
  - `\pL` = Buchstaben, `\p{Ll}` = Kleinbuchst., `\p{Lu}` = Großbuchst.
  - `\pN` = Ziffern, `\p{Cyrillic}` = kyrillische Zeichen, …
    - siehe https://www.pcre.org/original/doc/html/pcrepattern.html#SEC5

PCRE

Erweiterung für Volltextsuche:

- irrelevant für Korpusanfragen, die vollständige Zeichenketten liefern

- `(…)??`, `(…)*?`, `(…)+?`
  = so wenige Wiederholungen wie möglich
  - reguläre Ausdrücke sind normalerweise gierig (greedy), versuchen also, so viel zurückzugeben wie möglich – das kann unerwartete und unerwünschte Ergebnisse verursachen
- `^…` = Anker für Zeilenanfang
- `…$` = Anker für Zeilenende
  - Vorsicht: `^(…|…)$` ≠ `^…|…$`
  - `^` und `$` sind in manchen Korpusabfragesystemen auch als Anker für Wortanfang bzw. -ende verwendbar (da im Hintergrund jedes Wort in einer eigenen Zeile steht)
- `\b` = Anker für Wortgrenze (Wortanfang oder Wortende)
- Schöne PCRE-Kurzübersicht: https://www.debuggex.com/cheatsheet/regex/pcre

Suche im Kontext: *look-around assertions*

- Ausdrücke in einem bestimmten Kontext finden, ohne dass dieser Kontext im Ergebnis mitgeliefert wird

`…(?=…)`	=	*positive look-ahead* (Kontext muss Suchausdruck folgen)
`…(?!…)`	=	*negative look-ahead* (Kontext darf Suchausdruck **nicht** folgen)
`(?<=…)…`	=	*positive look-behind* (Kontext muss Suchausdruck vorangehen)
`(?<!…)…`	=	*negative look-behind* (Kontext darf Suchausdruck **nicht** vorangehen)

Gruppierungen und Rückwärtsreferenzen

- Runde Klammern erzeugen sog. *capturing groups*
  - `(\d{2}):(\d{2})` → Gruppen 1 (Stunden) und 2 (Minuten)
  - lässt sich z.B. mit Python oder Perl für Informationsextraktion verwenden
  - `(?:…)` = *non-capturing groups* (→ auch zur Kontrolle der Numerierung)
- Rückwärtsreferenzen auf Gruppen: **`\1`**, **`\2`**, …
  - `([a-z]+)-\1` → *fifty-fifty*, *wah-wah*, *ack-ack*, …
- Texteditoren: Ersetzen mit reg. Ausdrücken
  - Gruppen können in den Ersetzungstext eingefügt werden
  - üblicherweise mit **`$1`**, **`$2`**, …
  - “text processing for everybody” (→ *Find in Project*)