Abschnitt 4

Reguläre Ausdrücke

Grundlagen Regulärer Ausdrücke

- Reguläre Ausdrücke (engl. *regular expressions* (regex)) sind **abstrakte Ausdrücke** zur beschreiben von Zeichenfolgen
- Ein Regex beschreibt ein Muster (**Pattern**), auf das eine Zeichenfolge passen kann, oder auch nicht (**Matching**)
- Regex können daher für eine verbesserte Suche eingesetzt werden, z.B.:
  - `Herrn? M[ae][iy]er` findet *Herr Meier, Herrn Meier, Herr Maier,...*
  - `\w+\.\w+@uni-jena.de` findet alle FSU-Email-Adressen
  - `\d{4,6}\.\d{4,8}` findet Telefonnummern

Allgemeines zur Syntax

- Die meisten Zeichen ändern ihre Bedeutung nicht, stehen also für sich selber. Darunter
  - Buchstaben (`a, b, c, ...`)
  - Ziffern (`0, 1, 2,...,9`)
  - Das Leerzeichen ␣
- Andere Zeichen kommen in Regex Sonderbedeutungen zu, insbesondere Klammern und manche Satzzeichen. Diesen muss ein Backslash vorangestellt werden, um trotzdem ihre *normale* Bedeutung darzustellen. Z.B:
  - `\.`
  - `\(`
  - `\?`
  - `\\`

Disjunktion und Negation

- Darstellung unterschiedlicher möglicher Zeichen oder Zeichengruppen (**Disjunktion**)
  - Einzelne Zeichen mit eckigen Klammern: `M[ae][iy]er`
  - Eine Spanne (**Range**) von Zeichen mit eckigen Klammer und Minus: `[0-9], [A-Za-z]`
  - Disjunktion beliebiger Zeichenfolgen mit dem Pipe-Operator: `(Herrn|Frau)`
- **Negation** von Zeichen durch Zirkumflex und eckigen Klammern
  - `[^a]`, jedes beliebige Symbol außer “a”
  - `[^0-9]`, keine Zahlen

Zeichenklassen

- `\d`, Ziffern, entspricht `[0-9]`
- `\D`, keine Ziffern
- `\w`, alphanumierische Zeichen (`[a-zA-Z0-9]`)
- `\W`, nicht-alphanumerische Zeichen
- `\s`, Whitespace-Zeichen (Space `\r`, Tab `\t`, Newline `\n`)
- `\S`, Nicht-Whitespace-Zeichen (Zahlen, Buchstaben, Satzzeichen)
- `\b`, Wortgrenzen-Zeichen (boundary), z.B. Whitespace und Interpunktion

Quantoren

- Quantoren (Quantifier) geben an, wie häufig das *vorausgehende* Zeichen wiederholt werden darf
- `?`, null oder ein Mal
- `+`, 1 bis *n*
- `*` , 0 bis *n*
- `{n}`, genau *n* Mal
- `{n,m}`, zwischen und *n* und *m* Mal
- Darüber hinaus gibt viele weitere, insbesondere für UTF-8

Wildcard

- Der Punkt `.` dient als *Wildcard* und matcht jedes beliebige Zeichen.
- Z.B. `w.rf` matcht “wirf” und “warf”, aber auch “wurf”, “werf”, “w4rf”,...
- Gerade die Kombination von Wildcard und Quantifiern (v.a. `+` und `*`) kann zu unerwarteten Effekten führen:
  - Regex: `dies.*\b`
  - *soll* matchen: dieser, dieses, dies,...
  - Text: *Sie findet diesen Sommer besonders schön.*
  - Matcht: *diesen Sommer besonders schön.*

Greediness

- Üblicherweise matchen Regex mit Quantoren die *längsmögliche* Zeichenfolge (**greedy**). Z.B.:
  - Regex: `a+h`
  - Text: *Aaaaaaah!*
  - Match: *aaaaaah* (und nicht *ah*)!
- Das Fragezeichen hinter dem Quantifer ändert dessen Verhalten, so dass jetzt die *kürzestmögliche* Zeichenfolge gematcht wird (**non-greedy, lazy**).
- Beispiel `dies.*?\b`

Escaping

- Um Zeichen mit Sonderbedeutungen darzustellen müssen diese **escaped** werden (Backslash voranstellen)
  - `\(\)`
  - `\{\}`
  - `\[\]`
  - `\.`
  - `\+`
  - `\?`
  - `\|`
  - `\\`

Cheat-Sheet Reguläre Ausdrücke

Disjunktion:

- `[Bb]`
- `[A-Z], [A-Za-z]`
- `(o|ou)`

Negation:

- `[^a], [^0-9]`

Quantoren:

- `?` 0 oder 1
- `+` 1 bis n
- `*` 0 bis n
- `{n}` genau n
- `{n,m}` n bis m

Zeichenklassen:

- `\d` Ziffern, `\D` keine Ziffern
- `\w` Alphanumerische
- `\s` Whitespace
- `\b` Wortgrenze

Wildcard:

- `.` matcht jedes Zeichen

Non-Greediness:

- `?`, z.B. `a*?`

Übung

Auf welche Art von Zeichenkette passen jeweils die folgenden Regex?

- `diese?`
- `g\wb`
- `(1\d|20)\d{2}`
- `\w+\.\w+@[\w\-]+\.\w{2,3}`