Reguläre Mustererkennung

Übersicht

- Suche von Zeichenketten mittels regulärer Ausdrücke
- Eine Anwendung von Endlicher Automaten Technik
- Reguläre Suchmuster:
  - Zeichenketten
  - Zeichenklassen
  - Verankerungen
  - Optionalität
  - Disjunktion, Gruppierung
  - Wiederholungen
- Suchstrategien: Eifrig und gierig!
- Ersetzen mit Regulären Ausdrücken

Mustererkennung

Mustererkennung (*pattern matching*) in Zeichenketten

- In Textverarbeitungsprogrammen und vielen Programmiersprachen
  - Hier: Mustererkennung wie es in Perl oder JavaScript 1.2 zur Verfügung steht
  - Leider gibt es von Programm zu Programm kleinere und grössere Unterschiede in der konkreten Syntax. Die Prinzipen selbst sind allerdings stabil.
- Zeilenweise Mustererkennen
  - Z.B. grep-Tool: Zeige alle Zeilen, in denen ein Muster vorkommt
  - Hier: Zeige das gefundene Muster in einer Zeichenkette
- Einmaliges vs. mehrfaches Erkennen
  - Hier: Nur sogenannter *first match*
- Erweiterungen mit Ersetzen
  - Mehrfaches Ersetzen bringt zusätzliche Schwierigkeiten…

Zeichenketten

Wörtliche Zeichen

- Zeichenketten (mit relevanter Gross-/Kleinschreibung)

  `/Peter Pan/` matcht "*Aber Peter Pan sagte:*"

  `/die/` matcht "*Die Radieschen schmecken.*"

Zeichen mit Sonderbedeutung

- Die Zeichen `.?()[]{}*+|^$\` müssen mit `\` geschützt werden.
  - Dem schützenden Steuerzeichen (*escape char*) zusammen mit dem geschützten Zeichen sagt man *escape sequence*.

    `/z\.B\. nicht\?/` matcht "*Wer mag das z.B. nicht?*"

Zeichen für Sonderzeichen

- Tabulator (`\t`), Zeilenvorschub (`\n`), Wagenrücklauf (`\r`)

  `/\tTabulatoren\t/` matcht "*Viele	Tabulatoren	*"

Zeichenklassen

Zeichenauswahl

- Zwischen eckigen Klammern stehen alternative Zeichen.

  `/[dD]ie/` matcht "*Die Birne*" oder "*das Radieschen*"

  `/[1234567890]h/` matcht "*Das dauert 2 h.*" oder "*Das dauert 3 h.*"

Zeichenausschluss

- Das Dach in `[^chars]` schliesst alle nachfolgenden Zeichen aus.

  `/[^aeiou][^aeiou]/` matcht "*abba*"

Zeichenbereiche

- Der Bindestrich erlaubt Bereiche von nachfolgenden Zeichenkodes.

  `/Kapitel [0-9]/` matcht "*Kapitel 4.2*"

  `/[A-Z][A-Z][A-Z]/` matcht "*laut BBC wurde*"

Vorgefertigte Zeichenklassen

Wildcards: Vorgefertigte Zeichenklassen

- Der Punkt steht für ein einzelnes Zeichen ausser \n.

  `/ ... /` matcht *"Aber der Frosch sprach"*

- \d steht für eine Ziffer von 0 bis 9

  `/CH\-\d\d\d\d/` matcht *"CH-8580 Amriswil"*

- \s steht für Layoutzeichen (Leerzeichen, Zeilenende, Tabulatoren)

  `/,\s/` matcht *"Er kommt, ich weiss es."*

- \w steht für ein alphanumerisches Zeichen oder Unterstrich
  - Orientiert sich an Bezeichnern in Programmiersprachen wie C oder Perl

    `/\w\w\w\w/` matcht "Was, 99$ kostet das?"

Grosse Wildcards sind ausgeschlossene kleine:

- `\D = [^\d], \S = [^\s], \W =[^\w]`

Verankerungen

Verankerungen: Positionieren von Suchmustern

- Mit dem Dach suchen wir am Anfang der Zeichenkette

  `/^die/` matcht *"die Radieschen"*

- Mit dem Dollar suchen wir am Ende der Zeichenkette

  `/die$/` matcht *"die Parodie"*

- Mit \b suchen wir an einer Wortgrenze

  `/\bdies\b/` matcht *"Den Radieschen ist der 'dies academicus' egal."*
  - Als Wortgrenze zählen nebst allen Zeichen, die \W matchen noch Anfang und Ende von Zeichenketten.

- Mit \B suchen wir nicht an einer Wortgrenze.
  - Praktisch: `/\B...\b/` matcht Suffixe, `/\b...\B/` matcht Präfixe

Die Verankerungen selbst matchen keine Zeichen!

Optionalität, Gruppierung, Disjunktion

Optionale Zeichen

- Das Fragezeichen macht das vorausgehende Zeichen optional:

  `/Microsofts?/` matcht *"Microsoft verschenkt Programme."*

Optionale Muster

- Dank Klammerung lassen sich reguläre Muster optional setzen:

  `/(Bill )?Gates/` matcht *"Gates als Wohltäter"*

Disjunktion

- Der senkrechte Strich ist ein Infixoperator, der alternative Ausdrücke trennt – mit niedrigste Präzedenz.

  `/das|die/` matcht *"Gates verärgert die Aktionäre."*

  `/, (die|das|der)/` matcht *"Das Programm, das niemand kauft."*

Wiederholungen

+: Mindestens einmal repetieren

- Zeichen repetieren

  `/\d+/` matcht *"Er feiert den 25. Geburtstag."*

- Um Muster zu repetieren, braucht es Klammern:

  `/([Hh]a)+!/` matcht *"Er: Hahahaha!"*

*: Beliebig oft wiederholen

- Beliebig heisst null oder mehr Mal...

  `/x*/` matcht "" oder *"xxx"*

  `/the*/` matcht *"theee"*

  `/Ha(ha)*!/` matcht *"Hahaha!"*

- Achtung: Was matcht `/h*/` in "d.h."?

Matching-Strategien

Matching ist mehrdeutig

`/h*/` matcht nicht bloss "*d.h.*", sondern noch *ε* vor und nach allen Buchstaben!
D.H. "*εd.h.*", "*dε.h.*", "*d.εh.*", "*d.hε.*", "*d.h.ε*"

Strategie I: Sei eifrig! (*eager*)

- Matche die am weitesten links stehende Zeichenkette!

`/h*/` matcht zuerst "*εd.h.*"

Matching ist mehrdeutig

`/a!*/` passt nicht bloss auf "*ha!!!*", sondern auch auf "*ha!!!*", "*ha!!!*" und "*ha!!!*"

Strategie II: Sei gierig! (*greedy*)

- Matche soviele Zeichen wie möglich mit einem Ausdruck!

`/a!*/` matcht "*ha!!!*"

Wiederholungen II

{*n*}: Genau *n* Mal wiederholen!

`/\d{2}(-\d{3}){2}/` matcht "*Immatrikulationsnummer 99-723-362*"

{*min*,*max*}: Mindestens *min* Mal und höchstens *max* Mal wiederholen

`/ha{3,6}!/` matcht "*haaaa!*", aber nicht "*haa!*" oder "*haaaaaaa!*"

{*n*,}: Mindestens *n* Mal wiederholen!

- Beispiel: Suche Sätze, die mindestens 3 Komma enthalten!
  `/(.*,.*){3,}/`
- Beispiel: Suche Sätze, die genau 3 Komma enthalten!
  `/^([^,]*,[^,]*){3}$/`

Ersetzen mit Regulären Ausdrücken

Oft soll die gefundene Zeichenkette ersetzt werden.

- Ersetzungsoperator: **s/***Suchmuster***/***Ersetzungstext***/**
  `s/z\. ?B\./zum Beispiel/`

Oft sollen nur bestimmte Teile der gefundenen Zeichenkette modifiziert werden.

- Geklammerte Ausdrücke im Suchmuster stehen im Ersetzungstext als nummerierte Register ($*n*) zur Verfügung
  - n-tes Register enthält Ausdruck mit n-ter öffnender Klammer (von links nach rechts)
    `s/(\w+) \w+ (\w+)/$2 $1/` modifiziert "*das alte Haus*" zu "*Haus das*"
- Beispiel "Satzendeerkennung"
  Einsetzen eines Leerzeichen vor Satzendpunkten
  `s/(\. ["\(«']) / $1/` modfiziert "Er log. "Soso!" zu "Er log . "Soso!"

Literatur

Minitutorat mit Beispielen und Übungen zum Trainieren

http://www.cl.unizh.ch/clab/regex/ilap_regextut/

Literatur

- Friedl, Jeffrey E. F. (1998): Reguläre Ausdrücke.
  - "Bibel der Regulären Ausdrücke": Widmet sich umfassend dem Umgang mit Regulären Ausdrücken in verschiedenen UNIX-Tools und Perl.
- Jurafsky, D., Martin, J. (2000): Speech and Language Processing: An Introduction to Natural Language Processing. S.21-33.
  - Verständliche Einführung mit einigen praktischen Beispielen.
- Handbücher zur Programmiersprache PERL
- Hilfe zu MS Word