Die AI::Categorizer-Experienz

Steffen Schwigon
<schwigon@webit.de>

webit! Gesellschaft für neue Medien mbH

Übersicht

- **Exposition**
  - Projektkontext
  - Textkategorisierung
- **Durchführung**
  - AI::Categorizer
  - Hierarchische Kategorisierung
- **Reprise**
  - Erfahrungen

Projektkontext

Ausschreibungswesen

- Ausschreibungswesen
- Bekanntmachungstexte
- Ausschreibungsblätter
- Online-Veröffentlichung
- Online-Recherche
  - Inhaltsbeschreibung
  - CPV-Nummern

CPV-Nummern

- Numerischer Code aus 8 Ziffern
- Katalog aus 8300 CPV-Nummern
- mehr Ziffern = zunehmende Präzisierung

```
01000000  Erzeugnisse der Landwirtschaft, des Gartenbaus,
          der Jagd und zugehörige Erzeugnisse.
01100000  Feldfrüchte und Erzeugnisse des Erwerbsgartenbaus.
01110000  Getreide und andere Feldfrüchte
01111120  Weichweizen
01111200  Mais
...
45000000  Bauarbeiten.
45110000  Abbruch von Gebäuden sowie allgemeine
          Abbruch- und Erdbewegungsarbeiten.
45111211  Sprengarbeiten.
45111212  Enttrümmerungsarbeiten.
```

Projekt: CPV-Vergabe

- Ausschreibungsplattform *ausschreibungs-abc*
- Dienstleister vergibt CPV-Nummern
- Manuelle Arbeit
  - Zeitknappheit
  - aufwendig
- Vorschläge von einer Maschine
  - Maschine lernt
  - überwachende Tätigkeit
- abstrakt: Dokumente zu Kategorien zuordnen

Textkategorisierung

Grundprinzip

- Dokumente mit statistischen Gemeinsamkeiten bilden gemeinsame Kategorien
- Beispiel:
  - Böse Spamtexte
    - *"Lolita mag Dich"*
    - *"Earn money fast"*
  - Gute Texte
    - *"Sehr geehrte Damen und Herren"*
    - *"Am Samstag ist Hamburgerparty"*
- War einfach, nur zwei Kategorien: Spam vs. NichtSpam

Ausschreibungswesen

- Bekanntmachungstexte
- 8300 Kategorien

AI::Categorizer

AI::Categorizer

- Perl Framework für Textkategorisierung
  - Autor: Ken Williams
- Abstraktion rund um Kategorisierungsarbeiten
  - Dokumente, Kategorien
  - Qualitätsmessung, Experimente
- verschiedene Algorithmen/Bibliotheken
  - NaiveBayes
  - DecisionTree
  - kNN (k Nearest Neighbor)
  - SVM (Support Vector Machine)
  - Weka (ein Java-Framework)

Grundprinzip

- Antrainieren (Lernen)
  - vorhandene Dokumente mit vorhandenen Kategorien
  - statistische Eigenschaften ermitteln
  - speichern

- Kategorisieren
  - neue Dokumente zu Kategorien zuordnen
  - statistische Gemeinsamkeiten mit einer Kategorie
  - zuordnen

SYNOPSIS: Anlernen

```
use AI::Categorizer::Learner::NaiveBayes;
use AI::Categorizer::Document;
use AI::Categorizer::KnowledgeSet;

push @docs, new AI::Categorizer::Document
 ( name            => $doc_id,
   content         => { text => $bekanntmachungstext, ... },
   content_weights => { text => $wichtung_text, ... },
   categories      => [ '45111210', ... ] );

my @categories = ('45111210', ...) ;

my $k = new AI::Categorizer::KnowledgeSet
 ( documents  => \@docs,
   categories => \@categories );

my $nb = new AI::Categorizer::Learner::NaiveBayes (%params);

$nb->train      ( knowledge_set => $k );
$nb->save_state ('filename');
```

SYNOPSIS: Kategorisieren

```
use AI::Categorizer::Learner::NaiveBayes;
use AI::Categorizer::Document;
use AI::Categorizer::Hypothesis;

my $nb = AI::Categorizer::Learner::NaiveBayes->restore_state
                                                ('filename');

my $document = new AI::Categorizer::Document
 ( name            => $doc_id,
   content         => { text => $bekanntmachungstext, ... },
   content_weights => { text => $wichtung_text, ... },
 );

my $hypothesis = $nb->categorize ($document);

$best_category  = $hypothesis->best_category;
@all_categories = $hypothesis->categories;
```

Hierarchische Kategorisierung

Teile und herrsche

- Theoretisch 8300 Kategorien
- CPV-Nummern enthalten Hierarchie
  - 45_1_1_1_2_1_1

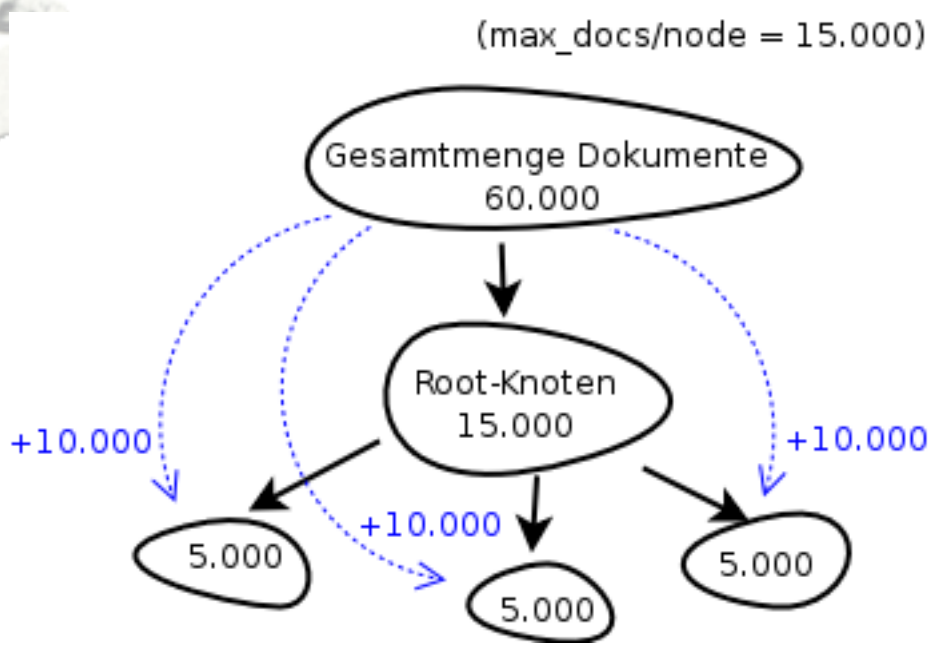
```
45000000  Bauarbeiten.
45100000  Baureifmachung.
45110000  Abbruch von Gebäuden sowie allgemeine
          Abbruch- und Erdbewegungsarbeiten.
45111000  Abbrucharbeiten, Baureifmachung und Abräumung.
45111200  Baureifmachung und Abräumung.
45111210  Spreng- und Enttrümmerungsarbeiten.
45111211  Sprengarbeiten.
```

- Baumstruktur

Hierarchisch lernen

- Baum aus Kategorisierern
- Wurzelknoten lernt nur Unterscheidung erste 2 Ziffern
- Weitere Knoten lernen nur je eine Folgeziffer
- Für jede Zifferfolge (Gruppe von Dokumenten) einen Kategorisierer

Grundprinzip Lernen



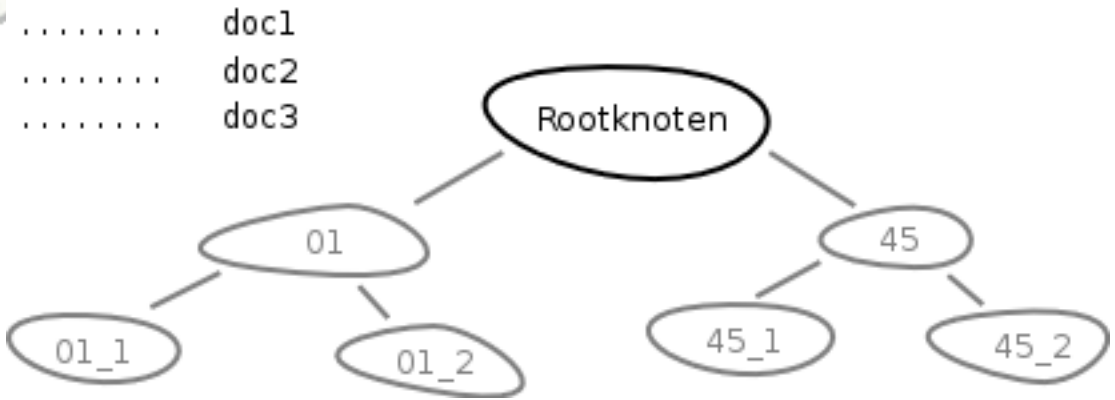
- Problem *Ausdünnen*

Hierarchisch kategorisieren

- Frisches Dokument
- Obersten Kategorisierer laden
- erste 2 Ziffern ermitteln, z.B. 45
- Kategorisierer für 45 laden
- Folgeziffer ermitteln, z.B. 45_1
- Kategorisierer für 45_1 laden, Folgeziffer ermitteln, usw.

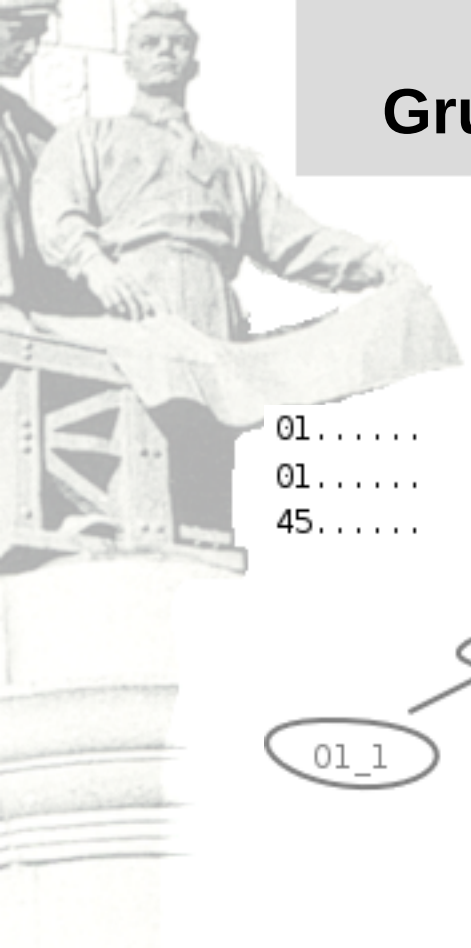
- Wichtig bei vielen Dokumenten:
  - jeden Kategorisierer nur einmal verwenden

Grundprinzip Kategorisieren

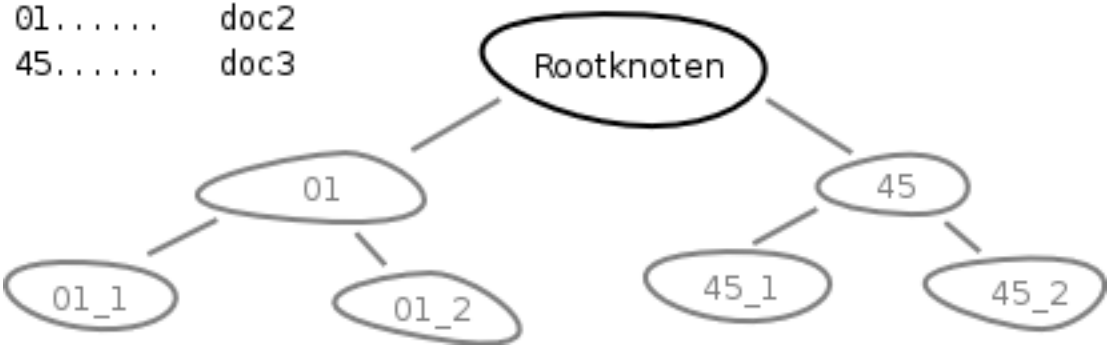


- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren

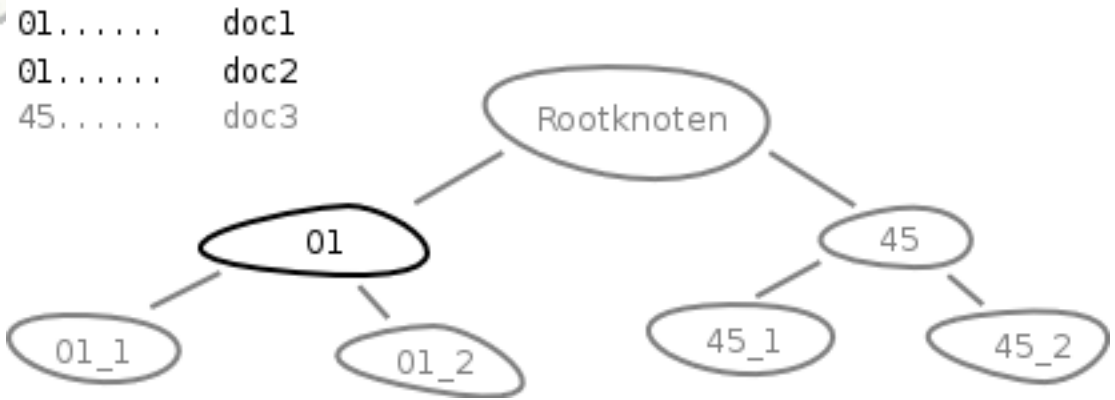


```
01......   doc1
01......   doc2
45......   doc3
```



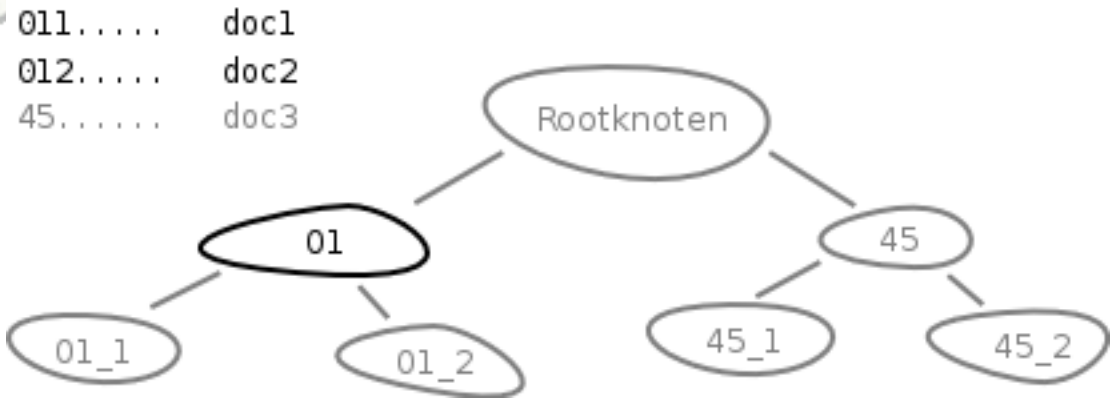
- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



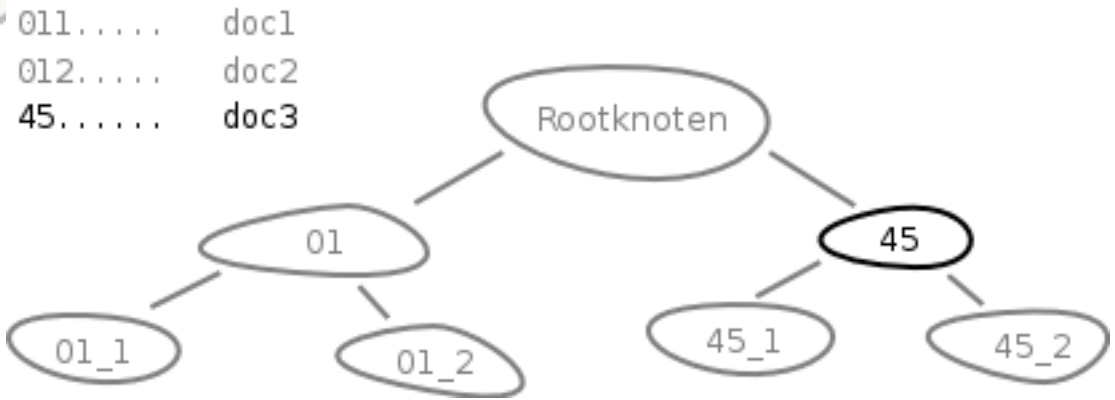
- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



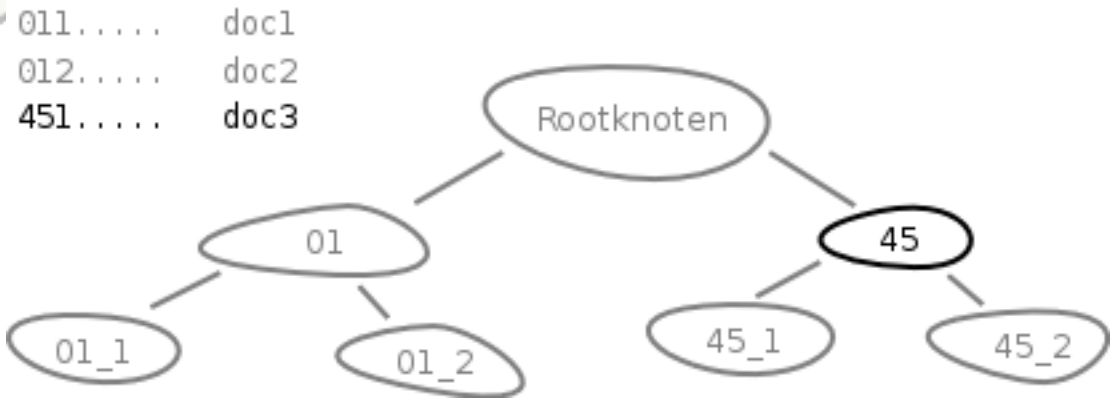
- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



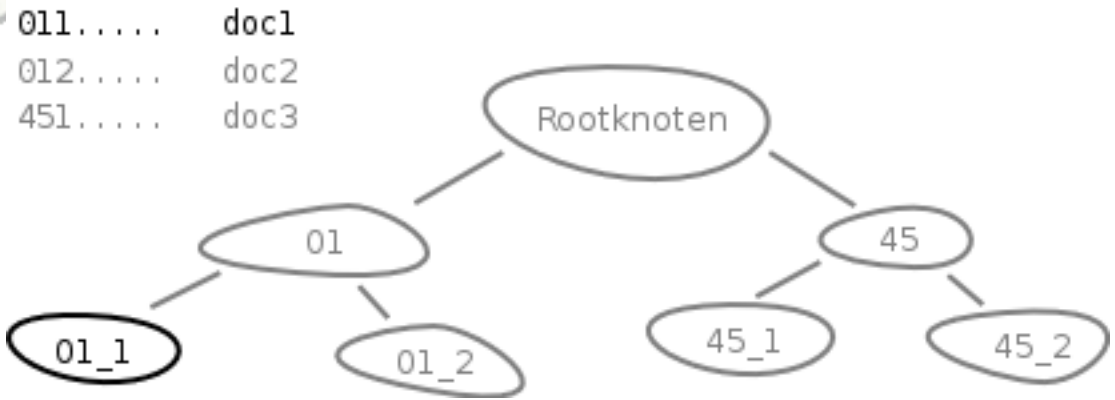
- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



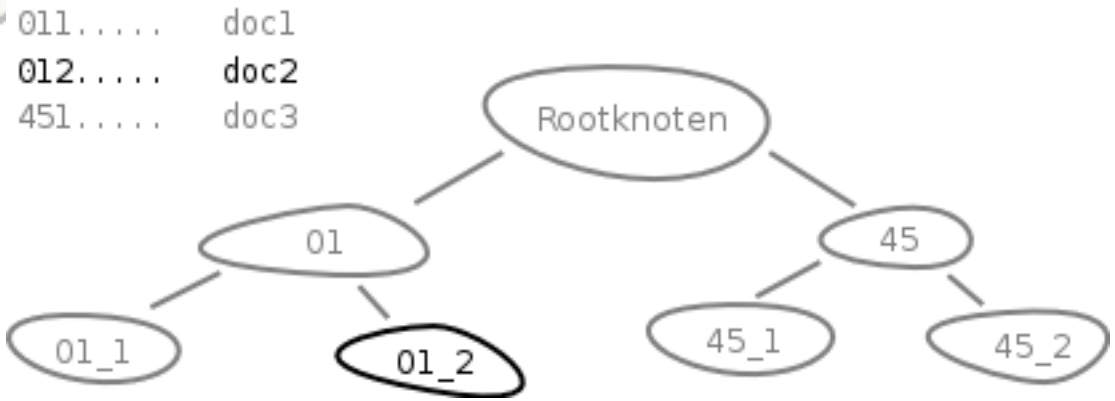
- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



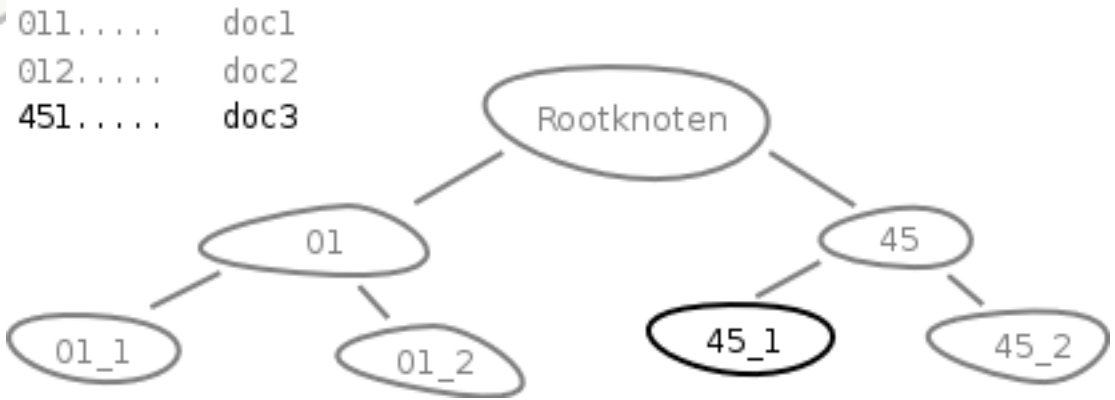
- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



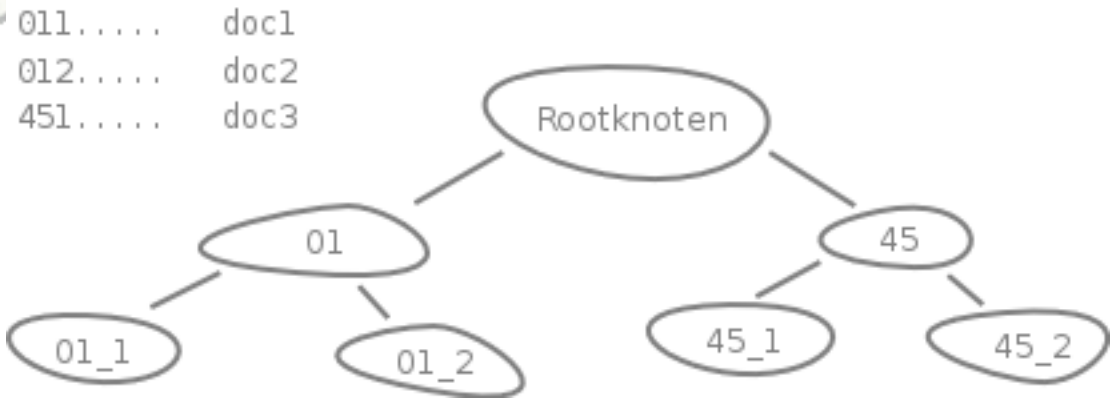
- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



- Jeden Kategorisierer *nur einmal verwenden*

Grundprinzip Kategorisieren



- Jeden Kategorisierer *nur einmal verwenden*

Loslösen vom CPV-Schema

- Hierarchie derzeit mit CPV-Nummernschema
- reines hierarchisches Framework herauslösen
- Zurück ins CPAN?

Erfahrungen

Ein unendliches Thema

- Nichttriviales Kategorisierungsproblem nicht trivial lösbar
- Viel Feintuning
- Qualitätsmessung
  - Kennzahlen: *precision*, *recall*, *accuracy*, *F1*
- Parametrisierung
  - des Frameworks
  - der Algorithmen
- Optimierung des konkreten Anwendungsfalles
  - Herauslösen wichtiger Textteile
  - Metainfos

Probleme

- Rechtzeitig Dokumentenmenge beschaffen
  - vergleichbar bzgl. Inhalt und Struktur
  - sonst keine Parametrisierung möglich
- Speicher, Zeit und Rechenlast
- Zeit für Parametrisierung
  - Parameter finden für genau dieses Kategorisierungsproblem
- Nicht alle Kategorisierer geeignet
  - NaiveBayes - schnell & einfach, medium Qualität
  - SVM - "halbschnell", komplexer, gute Qualität

Fazit

- Kategorisierungsproblem?
- AI::Categorizer ist gute Wahl
- Reserven durch Parametrisierbarkeit und Offenheit
- Zugang zu leistungsfähigen Kategorisierbibliotheken
  - SVM funktioniert inzwischen auch hierarchisch
- Perl für die Verarbeitung von Texten

Fazit

- Kategorisierungsproblem?
- AI::Categorizer ist gute Wahl
- Reserven durch Parametrisierbarkeit und Offenheit
- Zugang zu leistungsfähigen Kategorisierbibliotheken
  - SVM funktioniert inzwischen auch hierarchisch
- Perl für die Verarbeitung von Texten

- Ende

DRESDEN PERL MONGERS
*
*
use style;