

Programmierpraktikum

Sommersemester 2007 - 04.05.2007

Björn Wilmsmann, B.A.
Sprachwissenschaftliches Institut
Ruhr-Universität Bochum
wilmsmann@linguistics.rub.de

Heute: Wiederholung

- Besprechen der letzten Aufgabe
- ‚Guter‘ Code
- Datenstrukturen in Perl
- außerdem: List::Util, EPIC, CPAN
- anschließend: WordNet

Tokenizer

- Mutual Information
- Zurücksetzen von Variablen in Schleifen
- Separatoren
- Dokumentation, Benennungen

Mutual Information

- Maß für die Information, die man über eine Variable y enthält, wenn eine Variable x gegeben ist
- je größer der Abstand der gemeinsamen Verteilung von x, y zum Produkt der Einzelverteilungen ist, desto mehr hängt x von y (und vice versa) ab

Mutual Information

- Definition über Wahrscheinlichkeiten

$$I(X;Y) = \sum_x \sum_y p(x,y) \cdot \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Mutual Information

- Korpus: das Haus, das Haus, das Haus, das Haus, das Auto, das Auto, das Auto, das Kind, das Kind, das Kind
 - $P(\text{das, Haus}) = 0.4$
 - $P(\text{das, Kind}) = 0.3$
 - $P(\text{das, Auto}) = 0.3$

Mutual Information

- Korpus: das Haus, das Haus, das Haus, das Haus, das Auto, das Auto, das Auto, das Kind, das Kind, das Kind
 - $P(\text{das}) = 0.5$
 - $P(\text{Haus}) = P(\text{Auto}) = 0.2$
 - $P(\text{Kind}) = 0.1$

Mutual Information

- $I(\text{das}, \text{Haus})$
 - $= \ln(P(\text{das}, \text{Haus}) / P(\text{das}) * P(\text{Haus})) / \ln(2)$
 - $= \ln(0.4 / 0.5 * 0.2) / \ln(2)$
 - $= 2$

Variablen in Schleifen

```
foreach $line (<$input>) {  
  push(@words, split(/\s+/, $line));  
  foreach $word (@words) {  
    ...  
  }  
}
```

Separatoren

- `split(/ +/, $line)` vs. `split(/\s+/, $line)`

Dokumentation

- Code sollte per Leerzeilen in sinnvolle Blöcke unterteilt werden
- jeder Block sollte kommentiert werden
- Variablen sollten eindeutig und mit ‚telling names‘ benannt werden
 - %hash vs. %ngrams
 - @array34 vs. @tokens

„Guter“ Code

- Zeilenumbrüche machen
- eine Operation nach der anderen
- keine tief verschachtelten Operationen
- keine tief verschachtelten Kontrollstrukturen

Datenstrukturen in Perl

- Variablen, Referenzen
- Arrays
- Hashes
- Verschachtelte Datenstrukturen
- Methodenaufrufe mit Datenstrukturen
- Codebeispiel

List::Util

- <http://search.cpan.org/~gbarr/Scalar-List-Utils-1.19/lib/List/Util.pm>
- use List::Util
- stellt verschiedene sinnvolle Methoden zum Verarbeiten von Listen zur Verfügung

List::Util

- first BLOCK LIST
- `$foo = first { defined($_) } @list # first defined value in @list`
- `$foo = first { $_ > $value } @list # first value in @list which is greater than $value`

List::Util

- `max LIST`
- `$foo = max 1..10 # 10`

List::Util

- `maxstr LIST`
- `$foo = maxstr 'A'..'Z' # 'Z'`
- `$foo = maxstr "hello","world" # "world"`

List::Util

- min LIST
- \$foo = min 1..10 # 1

List::Util

- minstr LIST
- \$foo = minstr 'A'..'Z' # 'A'
- \$foo = minstr "hello","world" # "hello"

List::Util

- reduce BLOCK LIST
- \$foo = reduce { \$a + \$b } 1 .. 10 # sum
- \$foo = reduce { \$a . \$b } @bar # concat

List::Util

- shuffle LIST
- @cards = shuffle 0..5 | # 0..5 | in a random order

List::Util

- `sum LIST`
- `$foo = sum 1..100 # $\sum_{x=1}^n x = (n / 2) * (n + 1)$`

EPIC

- Perl Plugin für Eclipse:
<http://e-p-i-c.sourceforge.net/> bzw. <http://e-p-i-c.sourceforge.net/updates>
- stellt verschiedene Hilfsmittel zur Perl-Programmierung zur Verfügung
- Syntaxüberprüfung, Runtime
- Regular Expression Tester

CPAN

- <http://cpan.org/>
- Comprehensive Perl Archive Network
- riesiges Archiv mit Perl Modulen für verschiedenste Anwendungszwecke
- wird von verschiedenen Sponsoren unterstützt, unter anderem der RUB, die einen eigenen Mirror zur Verfügung stellt.

CPAN

- `(sudo) perl -MCPAN -e shell`
- `install MODULE_NAME` (entspricht `make install`, impliziert `make` und `make test`)
- installiert das ausgewählte Modul im lokalen Pfad nebst aller Abhängigkeiten
- mittels `i /REGEX/` auch Suche nach Modulen möglich

WordNet

- zum Verständnis und zur Motivation sind zunächst allgemeine Informationen zum Semantic Web hilfreich

Semantic Web

- Motivation
 - Menschen können aus lückenhaften Vorgaben weitere Information ableiten
 - Menschen können Assoziationen zwischen Klassen herleiten
 - Menschen haben Zugriff auf verschiedenartige Informationsquellen

Semantic Web

- Motivation
 - das momentane WWW nutzt natürliche Sprache und verschiedene Medien (Grafik, Sound, Videos)
 - die Bedeutung der Inhalte können von Menschen leicht erschlossen werden
 - von Maschinen hingegen nur schwer

Semantic Web

- Motivation
 - viele Aufgaben (z.B. Urlaubsbuchungen, Literaturrecherche) erfordern das Zusammenspiel verschiedener Informationsquellen
 - dieses Zusammenspiel ist automatisiert nur bei einem einheitlichen Protokoll möglich

Semantic Web

- basiert auf Ideen von Tim Berners-Lee [1]
- soll Wissen leichter als bisher automatisch verarbeitbar machen
- Bedeutung von Begriffen ergibt sich aus deren Beziehungen untereinander
- Maschinen kennen keine Beziehungen zwischen Begriffen

Semantic Web

- Ziel des Semantic Web ist es
 - Entitäten und deren Beziehungen untereinander abzubilden
 - Informationen und Wissen maschinenlesbar zu machen
 - automatischen, agentenbasierten Systemen die Arbeit zu erleichtern

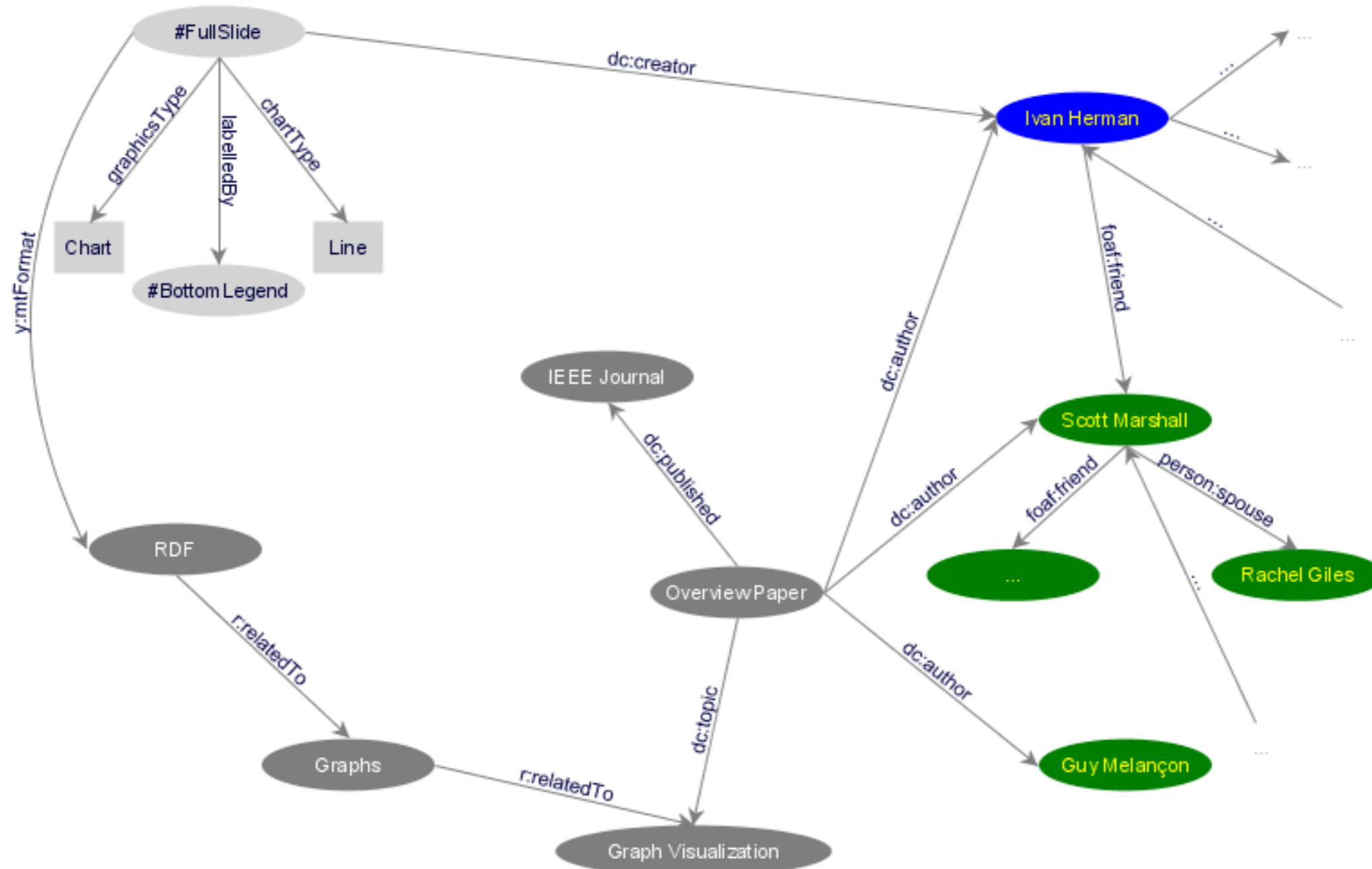
Semantic Web

- Netzwerk von Begriffen
- hierarchische Beziehungen zwischen Begriffen (,ist-Teil-von‘, ,ist-Unterklasse-von‘)
- analog semantischer Modellierung in ER Modellen
- Ontologien

Semantic Web

- bereits heutzutage verschiedene Anwendungen
 - biologische Taxonomien
 - Dokumentenclustering
 - Organisation von Informationen im ingenieurswissenschaftlichen Kontext (Beispiel: Ford)

Graphisches Beispiel



Semantic Web

- Verschiedene XML-Datenformate
 - RDF: Resource Description Framework
 - OWL: Web Ontology Language

RDF

- grundlegendes Dateiformat für das Semantic Web
- definiert sogenannte Tripel aus subject, predicate, object
- z.B. Subjekt: New York, Prädikat: hat-Abkürzungen, Objekt: NY

RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:terms="http://purl.org/dc/terms/">
  <rdf:Description rdf:about="urn:states:New%20York">
    <terms:alternative>NY</terms:alternative>
  </rdf:Description>
</rdf:RDF>
```

OWL

- baut auf RDF auf
- fügt weiteres Vokabular zur Definition von Ontologien hinzu

OWL

- Mengenbeziehungen
- Kardinalitäten
- Gleichheit
- Symmetriebeziehungen
- Aufzählungen

WordNet

- WordNet ist ein semantisches Lexikon für das Englische (andere Sprachen sind aber im gleichen Format verfügbar)
- enthält momentan ca. 150.000 Worte in ca. 115.000
- 207.000 Wort-Bedeutungs-Paare
- ca. 12 MB komprimierte Daten

WordNet

- Gruppiert Worte in Synsets (Synonym-Ringe, in einem Kontext äquivalente Terme)
- Beschreibt semantische Beziehungen zwischen Synsets
- Beispiel Synset: good, right, ripe -- (most suitable or right for a particular purpose; "a good time to plant tomatoes"; "the right time to act"; "the time is ripe for changes")

WordNet

- enthält Wortstämme, Informationen über Polysemie
- unterscheidet zwischen Nomen, Verben, Adjektiven und Adverbien
- Beziehungen für diese Klassen sind unterschiedlich

Substantive

- Hypernym: Oberklasse
- Hyponym: Unterklasse
- Meronym: ist-Teil-von
- Holonym: Inverses ist-Teil-von
- Koordinierte Begriffe: Haben eine gemeinsame Oberklasse

Verben

- Hypernym: Oberklasse
- Troponym: ist-Art-von (,flüstern‘ zu ,sprechen‘)
- Entailment: Implikation (,schnarchen‘ impliziert ,schlafen‘)
- Koordinierte Begriffe: Haben eine gemeinsame Oberklasse

Adjektive

- assoziierte Substantive
- Verbpartizipien

Adverbien

- zu Grunde liegendes Adjektiv

Beispiel

dog, domestic dog, *Canis familiaris*

=> canine, canid

=> carnivore

=> placental, placental mammal, eutherian, eutherian mammal

=> mammal

=> vertebrate, craniate

=> chordate

=> animal, animate being, beast, brute, creature, fauna

=> ...

Anwendungen

- Nutzung als lexikalische Ontologie
- Erkennen der Ähnlichkeit zwischen Worten: <http://www.d.umn.edu/~tpederse/similarity.html>

WordNet

- <http://wordnet.princeton.edu/>
- Links: <http://wordnet.princeton.edu/links>

WordNet

- Perl: <http://search.cpan.org/dist/WordNet-QueryData/>
- Java: <http://www.seas.gwu.edu/~simhaweb/software/jwordnet/>
- Python: NLTK

Aufgabe

- überarbeiten Sie Ihre bisherigen Lösungen
- probieren Sie schon einmal WordNet und das Perl Module `WordNet::QueryData` aus.

WordNet

- Download: <http://wordnet.princeton.edu/obtain> (WordNet 3.0 für Unices bzw. 2.1 für Windows)
- Entpacken
- Dictionary-Verzeichnis: `$WNHOME/dict/`

WordNet::QueryData

- Umgebungsvariable WNHOME setzen
 - Unices (BSD, OSX) und Unix-artige (Linux)
 - Bash: `.bash_profile`
 - csh: `setenv WNHOME /PFAD`
 - `export WNHOME="/PFAD"`

WordNet::QueryData

- Umgebungsvariable WNHOME setzen
 - Windows
 - Systemeinstellungen → System
 - Erweitert → Umgebungsvariablen
 - Neu: WNHOME, C:\PFAD

WordNet::QueryData

- `(sudo) perl -MCPAN -e shell`
- `install WordNet::QueryData`

WordNet::QueryData

```
use WordNet::QueryData;

my $wn = WordNet::QueryData->new;

print "Synset: ", join(", ", $wn->querySense("cat#n#7", "syns")), "\n";
print "Hyponyms: ", join(", ", $wn->querySense("cat#n#1", "hypo")), "\n";
print "Parts of Speech: ", join(", ", $wn->querySense("run")), "\n";
print "Senses: ", join(", ", $wn->querySense("run#v")), "\n";
print "Forms: ", join(", ", $wn->validForms("lay down#v")), "\n";
print "Noun count: ", scalar($wn->listAllWords("noun")), "\n";
print "Antonyms: ", join(", ", $wn->queryWord("dark#n#1", "ants")), "\n";
```

Referenzen

- [1] The Semantic Web (2001), Tim Berners-Lee et al.: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809E C588EF21>
- [2] Tutorial on Semantic Web Technologies (2005), Ivan Herman: <http://www.w3.org/2005/Talks/1214-Trento-IH/Overview.pdf>

**Vielen Dank für Ihre
Aufmerksamkeit!**