

Programmierpraktikum

Sommersemester 2007 - 27.04.2007

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

Eclipse / Aufgabe

- Bisherige Erfahrungen?
- Alle gut mit Subclipse / SVN zurecht gekommen?
- Wie lief das Lösen der Aufgabe?

EPIC

- Perl Plugin für Eclipse
- stellt verschiedene Hilfsmittel zur Perl-Programmierung zur Verfügung
- Syntaxüberprüfung
- Runtime
- Regular Expression Tester

Guter Code in Perl

- `@P=split//, ".URRUU\c8R";@d=split//, "\nrekah xinU / lreP rehtona tsuJ";sub p{`
- `@p{"r$p", "u$p"}=(P,P);pipe"r$p", "u$p";++$p;($q*=2)+=$f=!fork;map{$P=$P[$f^ord`
- `($p{$_})&6];$p{$_}= / ^$P/ix?$P:close$_}keys%p}p;p;p;p;p;map{$p{$_}=~/^[P.]/&&`
- `close$_}%p;wait until$?;map{/^r/&&<$_}&&%p;$_=$d[$q];sleep rand(2)if/\S/;print`

Guter Code in Perl

- Dieses Programm gibt aus „Just another Perl / Unix hacker“, hat NUR den zweiten Preis im 5. jährlichen Obfuscated Perl Contest gewonnen und soll als abschreckendes Beispiel dienen

Guter Code in Perl

- Pragas
 - use strict;
 - use warnings;
 - use Data::Dumper;

Guter Code in Perl

- Pragmas für Datenstrukturen
 - `Scalar::Util`
 - `List::Util`
 - `List::MoreUtils`

Guter Code in Perl

- Benennung
 - einheitliche Nutzung von `_` (C-Style) oder Binnengroßschreibung (Java-Style)
 - Boolean: `is_open`
 - Arrays: `@elements`
 - Hashes: `%mother_of`

Guter Code in Perl

- Referenzen: `$text_ref`
- alle Schleifen labeln, in denen `next`, `last`, `redo` etc. verwendet werden

Guter Code in Perl

- TEST:
- while (\$test == 1) {
-
- last TEST;
- }

Guter Code in Perl

- siehe dazu auch ein Klassiker: Go To Statement Considered Harmful, Edsger W. Dijkstra (1968)
- next, last, redo sind natürlich nicht ganz dasselbe wie GOTO, richtig angewendet können sie allerdings ähnlich verheerend wirken

Guter Code in Perl

- Dateioperationen
- indirekte Filehandles benutzen, keine Barewords (globaler Skopus), also `open($file)` statt `open(FILE)`

Guter Code in Perl

- Kontrollstrukturen
 - Postfix-Notation nur bei Kontrollstrukturen (next, die, last etc.), nicht bei Variablendeklaration
 - ~~my \$total = 100 if (\$total == 0);~~

Perl Module

- Was sind Perl Module?
- Wie sind Perl Module aufgebaut?
- Wie baue ich selber Perl Module?

Was sind Perl Module?

- the Perl way of reusing software components
- erlaubt prozedurale und objektorientierte Programmierweise
- da Perl ursprünglich keine objektorientierte Sprache war, wurde die Objektorientierung mit Hilfskonstrukten realisiert

Aufbau

- siehe Beispiel

Perl Module selbst gemacht

- siehe Beispiel

Textnormalisierung

- Allgemein
 - Konvertierung aller Buchstaben in Groß- oder Kleinbuchstaben
 - Zeichensetzung entfernen
 - Diakritika entfernen
 - Abkürzungen ausschreiben

Textnormalisierung

- Spezieller
- ähnlich klingende Worte auf eine gemeinsame Äquivalenzklasse abbilden:
Soundex
- Stemming: Porter Stemmer

Hintergrund

- Bedeutung im Information Retrieval
- Stemming und phonetische Abbildung erhöhen möglicherweise den Recall
- allerdings besteht dadurch die Gefahr, die Precision zu reduzieren
- rein linguistische Anwendung eher selten

Ein bisschen IR Theorie

- Information Retrieval: Suche nach Informationen in Dokumenten in Datenbanken, Hypertext-Strukturen
- Recall und Precision dienen als Benchmarking (Leistungsmaß) für Information Retrieval Systeme
- dazu zunächst einige Definitionen

Ein bisschen IR Theorie

- D = Corpus von n Dokumenten
- Q = Menge der möglichen Anfragen auf D
- für jede Anfrage $q \in Q$ existiert eine Untermenge $D_q \subseteq D$ der für q relevanten Dokumente

Ein bisschen IR Theorie

- für jede Anfrage $q \in Q$ existiert eine Untermenge $D_r \subseteq D$ der für q tatsächlich zurückgegebenen Dokumente
- $k = |D_r|$

Ein bisschen IR Theorie

- $\text{recall}(k) = (1 / |D_q|) \sum_{i=1}^k r_i$
- $\text{precision}(k) = (1 / k) \sum_{i=1}^k r_i$
- Was wollen uns diese Formeln sagen?

Ein bisschen IR Theorie

- Recall: Verhältnis der Anzahl der relevanten Dokumente in der Ergebnismenge zur Anzahl der tatsächlich relevanten Dokumente im Corpus
- Precision: Verhältnis der Anzahl der relevanten Dokumente in der Ergebnismenge zur Anzahl der Dokumente in der Ergebnismenge

Ein bisschen IR Theorie

- Ideal sind Werte von jeweils 100 % (bzw. 1)
- je nach verwendetem System und Modell (z.B. Bayes vs. Vektorraummodell) stark unterschiedlich

Ein bisschen IR Theorie

- Recall und Precision sind korreliert
- Erhöhung des einen Wertes birgt immer die Gefahr der Reduktion des jeweils anderen
- Je mehr sich ein Wert dem Grenzwert 0 nähert, desto mehr nähert sich der andere Wert dem Grenzwert 1
- siehe auch: Chakrabarti (2003, p. 54-56)

Soundex

- Kurzwiederholung Phonetik
- siehe <http://en.wikipedia.org/wiki/Soundex>
- phonetischer Algorithmus für das Englische
- indiziert Worte anhand ihrer Lautstruktur
- entwickelt und patentiert von Robert Russell und Margaret Odell

Soundex

- Varianten
 - Reverse Soundex
 - NYSIIS
 - Celko Improved Soundex
 - Metaphone
 - Daitch-Mokotoff Soundex

Soundex

- Soundex Code für ein Wort: $[A-Z]\{1\}[0-9]\{3\}$
- der Buchstabe ist der Anfangsbuchstabe des Worts
- die Ziffern kodieren die restlichen Konsonanten des Wortes

Soundex

- ähnlich klingende Konsonanten erhalten dieselbe Ziffer
- z.B. [bfpv] = 1
- Vokale beeinflussen die Kodierung, werden aber selbst nicht kodiert

Soundex

- normalisiere auf Kleinschreibung
- behalte den ersten Buchstaben des Wortes
- entferne alle Vorkommen der folgenden Buchstaben an Positionen $\neq 1$: a, e, h, i, o, u, w, y
- weise den verbleibenden Buchstaben (außer dem ersten) Zahlen zu

Soundex

- [bfpv] = 1
- [cgjkqsxz] = 2
- [dt] = 3
- [l] = 4
- [mn] = 5
- [r] = 6

Soundex

- falls im Initialzustand zwei oder mehrere Buchstaben aufeinander folgten, die der gleichen Ziffer zugewiesen wurden, entferne alle Ziffern außer der ersten (tricky)
- optionaler Left-Shift bis Stringlänge = 4
- gebe die ersten 4 Zeichen zurück

Soundex

- Übung
 - Robert
 - Rupert
 - jingle
 - angel

Porter Stemmer

- Kurzwiederholung Morphologie
- Erfinder: Martin Porter
- ursprünglich im Computer Laboratory, Cambridge, UK als Teil eines IR Systems entwickelt
- entfernt Endungsmorpheme von Worten des Englischen in mehreren Schritten

Porter Stemmer

- Konsonanten (c): Buchstaben $\{a, e, i, o, u, y_nach_Konsonant\}$
- Vokale (v): alle übrigen Buchstaben
- C: String, der aus einem oder mehreren Konsonanten besteht
- V: String, der aus einem oder mehreren Vokalen besteht

Porter Stemmer

- jedes englische Wort lässt durch folgenden regulären Ausdruck darstellen
- $(C)^*?(VC)^*?(V)^*?$
- die Anzahl der gefundenen (VC) stellt dabei die sogenannte Measure (m) dar

Porter Stemmer

- Beispiele
 - $m = 0$: TR, EE, TREE, BY
 - $m = 1$: TROUBLE, OATS, TREES, IVY
 - $m = 2$: TROUBLES, PRIVATE, ORRERY

Porter Stemmer

- Regeln
 - (Bedingung) $S1 \rightarrow S2$
 - „falls das Wort mit dem Suffix $S1$ endet und der Stamm vor $S1$ die Bedingung erfüllt, ersetze $S1$ mit $S2$ “

Porter Stemmer

- Bedingungen
 - m: Measure des Stamms
 - *S: Stamm endet mit S
 - *v*: Stamm enthält Vokal
 - *d: Stamm endet mit doppeltem Konsonanten

Porter Stemmer

- Bedingungen
- *o: Stamm endet mit CVC, wobei das zweite C kein W, X oder Y ist (z.B. -WIL, -HOP)

Porter Stemmer

- sequentielle Abarbeitung eines 7-schrittigen Algorithmus
- 1. Pl. Nouns and 3rd P. Sg. Verbs
- 2.
 - a. Verbal Past Tense and Progressive Forms
 - b. Cleanup

Porter Stemmer

- 3.Y → I
- 4. Derivational Morphology I: Multiple suffixes
- 5. Derivational Morphology II: More multiple suffixes

Porter Stemmer

- 6. Derivational Morphology III: Single suffixes
- 7.
 - a. Cleanup
 - b. Cleanup

Porter Stemmer

- siehe auch: Jurafsky & Martin (200, Appendix B)
- Homepage des Erfinders: <http://www.tartarus.org/~martin/PorterStemmer/>
- Perl: <http://search.cpan.org/~ulpfr/perlindex-1.502/lib/Text/English.pm>

Porter Stemmer

- Perl: <http://search.cpan.org/~ulpfr/perlindex-1.502/lib/Text/English.pm>
- Java: Komponente von Lucene (<http://lucene.apache.org/>)
- Python: Komponente des Natural Language Toolkit (<http://nltk.sourceforge.net/>)

Aufgabe

- Schreiben Sie ein Normalizer-Modul, das
 - einen String nimmt und die darin enthaltenen Worte mittels zweier Methoden normalisiert und jeweils in einem Hash speichert und ausgibt
 - mit dem Soundex-Algorithmus
 - mit dem Porter Algorithmus

Aufgabe

- Implementieren Sie den Soundex Algorithmus selbst
- Nutzen Sie für den Porter Algorithmus eines der vorhandenen Module
- Abgabe bis zum 02.05.2007, 00:00 Uhr per SVN
- Abgabe alleine oder in Zweierteams

**Vielen Dank für Ihre
Aufmerksamkeit!**