

Programmierpraktikum

Sommersemester 2007 - 20.04.2007

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

Heute: Aufwärmrunde

- Tokenisierung und Erkennen von Abkürzungen

Was ist ein Token?

- ein einzelnes Vorkommen eines Wortes in einem Text
- konkretes Wort

Was ist ein Type?

- Arten von Vorkommen
- repräsentiert alle Vorkommen eines Wortes in einem Corpus
- abstraktes Wort, Lexikoneintrag

Was ist Tokenisierung?

- „Tokenisierung bezeichnet in der Computerlinguistik die Segmentierung eines Textes in Einheiten der Wortebene (manchmal auch Sätze, Absätze o.ä.).“ (<http://de.wikipedia.org/wiki/Tokenisierung>)
- siehe auch Halama (2004a), Kap. 3.3.1 (pp. 218-223)

Was ist Tokenisierung?

- Tokenisierung ist Voraussetzung für jede weitere Verarbeitung von Texten im Information Retrieval bzw. allgemein in der Sprachverarbeitung
- ein grundlegendes Verständnis von Problemstellungen beim Tokenisieren kann daher nicht schaden

Einfacher Ansatz

- Trennung der Token nach bestimmten Zeichen
- \s für Worte
- [.:;!?] für Sätze
- [\r\n] für Absätze

Probleme

- in vielen Sprachen hören Worte nicht unbedingt nach einem Leerzeichen auf (Englisch, alle romanischen Sprachen)
- Worte werden per Konvention (Latein) oder grammatisch bedingt (polysynthetische Sprachen wie z.B. Finnisch) aneinander gereiht

Probleme

- im Gegensatz zu [::!?] ist . ambig
- neben Satzendezeichen auch
 - Trennsymbol bei Zahlen: dt. 3.298,24;
engl. 3,298.24
 - Nutzung in Abkürzungen und Akronymen: Abk.; U.S.A.

Probleme

- ambige Abkürzungen
 - Dr. kann z.B. für *doctor* oder *drive* stehen
 - „He stopped to see Dr.White“
 - „He stopped at Meadows Dr.White Falcon was still open.“

Probleme

- Klitika im Englischen
 - „they‘re“ besteht aus zwei Token
- aber auch Deppenapostroph im Deutschen
 - „Harry‘s Hardwarehölle“
 - „Tiefpreise für PC‘s“

Verbesserte Ansätze

- alle erwähnten Problemstellungen sind relevant
- im Folgenden wird es aber um das Erkennen von Abkürzungen bzw. Kollokationen gehen
- Qualitätskriterium: Baseline (bei Abkürzungen je nach Anwendung 50 bis 90 % richtig erkannter Interpunktion)

Verbesserte Ansätze

- Symbolische und statistische Algorithmen
- Filterung anhand von
 - Regeln, regulären Ausdrücken, allgemein Wortstruktur
 - Statistischer Abhängigkeit der Wortvorkommen, Erkennen von Kollokationen

Symbolischer Ansatz

- Filterung anhand von
 - regulären Ausdrücken
 - Abkürzungslisten

Symbolischer Ansatz

- Vorteile
 - schnelles Prototyping
 - kompakt
- Nachteile
 - robustes System erfordert viel Feintuning
 - langwierige, ermüdende Arbeit

Symbolischer Ansatz

- ein Verfahren für das Englische
(Grefenstette / Tapanainen (1994), siehe Halama (2004a), Kap. 3.3.1 (pp. 221-222))

Symbolischer Ansatz

- Zeichensequenz: $c_1, c_2 \dots c_n$; $c_n = \cdot$
- Corpus C ist eine Multimenge
- Lexikon L ist eine Menge von Einträge mit beliebigen Zeichensequenzen $c_1, c_2 \dots c_n$
- Lexikon A ist die Menge der bereits erkannten Abkürzungen

Symbolischer Ansatz

- $c_{n+1} = [a-z,;] \vee c_{n+1} = // \wedge c_{n+2} = [a-z] \rightarrow$

Abkürzung

- $c_1 = [a-z] \wedge c_1, c_2 \dots c_{n-1} \in L \rightarrow$ keine

Abkürzung

- $c_1 = [A-Z] \wedge c_1, c_2 \dots c_n \in A \rightarrow$ Abkürzung

Symbolischer Ansatz

- $c_1 = [A-Z] \wedge c_1, c_2 \dots c_n, c_{n+1} \in C$ mit $c_{n+1} \neq // \rightarrow$ keine Abkürzung
- $c_1 = [A-Z] \wedge (V(c_1, c_2 \dots c_n) = 1 \vee V(c_1, c_2 \dots c_n) = 2) \rightarrow$ keine Abkürzung
- andernfalls \rightarrow Abkürzung

Statistischer Ansatz

- macht sich die Eigenschaften von Bigramm-Modellen zu Nutze
- Unigramme, die häufig gemeinsam in Bigrammen auftreten, weisen eine statistische Abhängigkeit auf
- Mutual Information: $I(X; Y) = E \left\{ \log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right) \right\}$

Statistischer Ansatz

- Erkennen von Abkürzungen wird so zu Kollokationserkennung
- Kollokation: Regelmäßig miteinander auftretende Worte
- Abkürzungen sind in diesem Sinne Kollokationen, deren zweites Element nur zufällig kein zweites Wort, sondern ein Punkt ist.

Statistischer Ansatz

- Vorteile
 - automatisches Lernen, kein Feintuning
 - Kollokationen und Abkürzungen
- Nachteile
 - Sparse Data
 - großer Datensatz

Und nun?

- beide Ansätze sind nicht ohne Weiteres problemlos implementierbar
- für regelbasierte Ansätze fehlt uns (noch) das Lexikon
- naive statistische Ansätze führen in der Praxis oftmals zu falschen Positiven

Und nun?

- wir kombinieren daher eine Klassifikation mittels Mutual Information mit grundlegenden regelbasierten Heuristiken
- eine Sequenz wird nur als Abkürzung interpretiert, wenn sie eine hohe Mutual Information aufweist und ein weiteres Kriterium für Abkürzungen erfüllt

Und nun?

- solche Kriterien sind
 - relative Kürze der Sequenz
 - mehrere Punkte in einer Sequenz

Aufgabe

- Erstellen Sie ein Modul, das
 - den Pfad zu einem Corpus (Textdatei; wer mag, kann auch URLs bzw. HTML/XML Input verarbeiten lassen) als Argument nimmt

Aufgabe

- dieses Corpus zunächst nach dem primitivem Ansatz in ein Array von Unigram Token, sowie ein Array von Bigram Token aufteilt, diese jeweils zählt und die gezählten Häufigkeiten in einen Hash mit den im Corpus enthaltenen Unigram-/Bigram-Typen speichert
- die Gesamtzahl der Token aufaddiert

Aufgabe

- die Wahrscheinlichkeiten für die einzelnen Unigramme und Bigramme berechnet
- Mutual Information mit symbolischen Heuristiken kombiniert, um Abkürzungen zu erkennen
- die erkannten Abkürzungen als Hash mit ihrer Häufigkeit als Wert zurückgibt

Aufgabe

- Schreiben Sie ein Hauptprogramm, welches
 - das erstellte Modul sinnvoll aufruft
 - das Hash in eine zuvor als Argument übergebene Textdatei sortiert (absteigend nach Häufigkeit) ausgibt

Aufgabe

- Stubs in Perl, Python und Java, sowie die Aufgabenstellung und die Folien werden ab heute nachmittag in Ihrem SVN-Ordner liegen: https://rebelyell.linguistics.rub.de/svn/courses/pp/work/students/nachname_des_students
- Nutzernamen und Passwörter folgen per E-Mail

Aufgabe

- Abgabe: Bis zum 26.04.2007, 0:00 Uhr per SVN

**Vielen Dank für Ihre
Aufmerksamkeit!**