KI-Praktikum

M. Helmert, M. Brenner, J. Smaus, M. Westphal
Sommersemester 2009

Universität Freiburg
Institut für Informatik

Projekt 5: Natürliche Sprache

Abgabe: 2. August 2009

Präsenzaufgaben

Aufgabe 1

Die Datei `grammars.py` enthält grundlegende Datenstrukturen zur Beschreibung von Grammatiken sowie einfache Beispiele.

(a) Benutzen Sie die Funktion `generate_random()` um Zufallssätze mit der gegebenen Grammatik *E1* zu erzeugen. Erweitern Sie in einer Kopie *E2* von *E1* das Lexikon um zusätzliche Beispiele der einzelnen Wortarten.

(b) Erweitern Sie das Lexikon von *E2* um die Wortart *Adjective* und einige Beispiele. Erweitern Sie dann die Grammatik so, dass Nominalphrasen der Form *Art Adjective Noun* erzeugt werden können.

(c) Erweitern Sie die Grammatik so, dass Präpositionalphrasen erzeugt werden können. Präpositionalphrasen haben die Form *Preposition NP*, also z.B. „on the beach“.
Die Grammatik hat damit eine entscheidende Veränderung erfahren. Welche?

(d) Verändern Sie die Grammatik so, dass in einer Nominalphrase beliebig viele Adjektive (also auch keines!) vor dem Nomen stehen können.

Aufgabe 2

Die Datei `parsing.py` enthält die Implementation eines Chart-Parsers.

(a) Testen Sie die Funktion `parse_phrase()` mit verschiedenen Grammatiken und Sätzen. Verwenden Sie den Parameter `trace`, um Informationen über den Ablauf des Parsingprozesses zu erhalten.

(b) Erweitern Sie `parsing.py` um die Möglichkeit, Syntaxbäume (parse trees) zu visualisieren, ähnlich wie in Abbildung 1.

(c) Finden Sie Beispiele für mehrdeutige Sätze in den Grammatiken, die Sie bisher erstellt haben.

```
S
  Statement
    NP
      Art : the
      Adj : green
      N : gremlin
    VP
      V : is
      Adj : hungry
```

Abbildung 1: Ein einfacher Syntaxbaum.

Aufgabe 3

Modellieren Sie eine Grammatik *E3* in *deutscher* Sprache mit Regeln und Lexikon, die *E1* entsprechen.

(a) Vergleichen Sie die durch `generate_random()` im Englischen und im Deutschen erzeugten zufälligen Sätze. Was fällt auf?

(b) Was ist notwendig, um im Deutschen Nominalphrasen mit Adjektiven zu beschreiben? Erweitern Sie *E3* analog zu Aufgabe 1(b) so, dass Sätze wie „Das hungrige Monster frisst die suesse Schokolade“ geparst werden können.

(c) Frage: Wie wachsen Grammatik und Lexikon bei steigender Zahl von Fällen, Geschlechtern u.s.w.?

(d) Frage: Wie bildet man die Dativ-Formen von Nominalphrasen im Deutschen, also “Das gehört . . . ” u.s.w.?

(e) Modellieren Sie schriftlich diese NP-Regel als Teil einer Definitive Clause Grammar.

Aufgabe 4

Auf `http://norvig.com/spell-correct.html` finden Sie einen Artikel von Peter Norvig (ja genau, *der* Peter Norvig), in dem ein Rechtschreibkorrekturprogramm für die englische Sprache angegeben wird.

(a) Probieren Sie das Programm an einigen Beispielworten aus.

(b) Im Gegensatz zum Beispiel zur Rechtschreibkorrektur in Google wird ein als korrekt bekanntes Wort niemals „korrigiert“, selbst wenn das Wort so selten ist, dass ein Tippfehler mutmaßlich wahrscheinlicher ist, als dass wirklich dieses Wort gemeint ist.

Überzeugen Sie sich hiervon, indem Sie die Datei `big.txt` so verändern, dass sie das obige Phänomen in extremer Form offenbart.

Erklären Sie das Phänomen mit Hilfe des Begriffs „Fehlermodell“, und verweisen Sie auf die entsprechende Stelle im Code.

Verändern Sie den Code derart, dass auch korrekte Wörter „korrigiert“ werden, wenn es ein häufigeres Wort in Editierdistanz 1 gibt.

Hausaufgaben

Aufgabe 5

Ein *referentieller Ausdruck (RA)* ist, vereinfacht gesagt, eine Nominalphrase, die ein Sprecher verwendet, um ein Objekt gegenüber einem Hörer eindeutig zu identifizieren. Dies ist für Kommunikation in natürlicher Sprache von zentraler Bedeutung, da Objekte normalerweise keine eindeutigen, von allen Gesprächsteilnehmern verwendeten Namen haben. Durch die Verwendung eines RAs, z.B. in dem Satz "Gib mir die blaue Tasse!", können beide Gesprächsteilnehmer sicherstellen, vom gleichen Objekt zu sprechen.
Es existieren zahlreiche Algorithmen zur Generierung referentieller Ausdrücke (GRE). Gemeinsam ist ihnen, dass eine Nominalphrase gesucht wird, die

(a) das Objekt beschreibt, und

(b) alle anderen Objekte (die sogenannten Distraktoren) *nicht* beschreibt.

Der wohl einfachste denkbare GRE-Algorithmus ist der sogenante "Full Brevity" Algorithmus, der den kürzestmöglichen RA der Form *the Adj* Noun* erzeugt.[1] Dazu wird eine Breitensuche über alle möglichen RAs, die den Referenten beschreiben, in wachsender Länge durchgeführt, und für jeden RA geprüft, ob er *nur* den Referenten beschreibt. Ist dies der Fall, wurde ein kürzestmöglicher RA gefunden.

In der Datei `monsterworld.py` finden Sie die Beschreibungen diverser äußerst gefährlicher Monster. Implementieren sie den "Full Brevity" Algorithmus, so dass für jedes Monster ein RA erzeugt wird, der dieses eindeutig identifiziert.

Aufgabe 6

Sie sind die Entwickler eines Informationssystems über die (wie bereits erwähnt, äußerst gefährlichen) Monster aus der Datei `monsterworld.py`.

(a) Entwickeln Sie eine Grammatik *E4* (basierend auf *E2*), so dass einfache Anfragen an das System in natürlicher Sprache (Englisch) geparst werden können. Z.B. sollen folgende Anfragen möglich sein:

- *"Where is the big red monster?"*
- *"What is the name of the green goblin?"*
- *"Describe Gizmo"*.

Anmerkung: Die Namen der Monster sollen in der Datenbank gespeichert werden. Sie sollten deshalb dem Lexikon von *E4* zur Laufzeit automatisch hinzugefügt werden.

[1] Psycholinguistische Untersuchungen haben gezeigt, dass beim Menschen Kürze nicht das entscheidende Auswahlkriterium bei der Auswahl eines geeigneten RAs ist.

(b) Überlegen Sie sich eine geeignete Datenstruktur zur Speicherung der Informationen über die Monster, um solche Anfragen beantworten zu können.

(c) Implementieren Sie eine Funktion `resolve_reference()`, die für eine geparste Nominalphrase ein Monster aus der Datenbank findet, das der Beschreibung entspricht.
Anmerkung: Bedenken Sie die Möglichkeit syntaktischer sowie semantischer Ambiguität (mehrere mögliche Parses bzw. mehrere mögliche Referenten).

(d) Implementieren Sie eine Funktion `interpret_command()`, so dass aus einer geparsten Anfrage mit Hilfe von `resolve_reference()` eine Interpretation der Anfrage gewonnen wird und für diese aus der Datenbank eine Antwort ausgelesen wird.

(e) Generieren Sie abhängig von der Anfrage eine passende englische Antwort. Z.B. soll auf „What is the name of the green goblin?“ die Antwort lauten „Its name is Norman“. Für die Generierung von Beschreibungen sollten Sie Ihre Implementation des „Full Brevity“ Algorithmus aus Aufgabe 5 verwenden.

Aufgabe 7

(Fortsetzung zu Aufgabe 4)

(a) Erweiterung von Teilaufgabe (b): Ein korrektes Wort soll nur dann „korrigiert“ werden, wenn es deutlich seltener ist als ein anderes Wort in Editierdistanz 1. Die Definition von „deutlich“ ist Ihnen überlassen und soll durch einige Tests kalibriert und dokumentiert werden.

(b) Gehen Sie aus von dem Code der vorherigen Teilaufgabe (d.h., auch korrekte Wörter werden evtl. „korrigiert“).

Verfeinern Sie die bisherige Editierdistanz 1: Änderungen, wo ein Wort durch Ersetzung eines Einzelbuchstabens durch den entsprechenden Doppelbuchstaben erhalten wird oder umgekehrt, sollen eine geringere Editierdistanz erhalten. Verändern Sie das Programm entsprechend. Das Programm sollte folgende Korrekturen berechnen: fil → fill; rabit → rabbit; watter → water; tooy → toy; flor → floor; dor → door; peper → pepper; mater → matter; biding → bidding; pol → poll; cel → cell; fil → fill; til → till; buz → buzz; bok → book; faty → fatty; funy → funny; winer → winner; spining → spinning.

(c) Implementieren Sie die T9-Funktion, die Sie von Ihrem Mobiltelefon kennen. Z.B. sollte `t9('438')` = `'get'` sein und `t9('2655328')` = `'collect'`.

Aufgabe 8 (Freiwillig)

Integrieren Sie allen für dieses Aufgabenblatt entwickelten Code! Entwickeln Sie ein Dialogsystem, dem man Informationen über Monster geben und Fragen dazu

stellen kann – natürlich darf man sich dabei auch mal einen Tippfehler erlauben . . .
Das könnte z.B. so aussehen:

> Benutzer: Gizmo is a small white furry gremlin.
> System: Aha.
> Benutzer: What is the name of the small germlin? *(Tippfehler!)*
> System: Its name is Gizmo.
> Benutzer: . . .

Viel Spaß!