Übung zur Vorlesung "Computerlinguistik I"

Wintersemester 2016/17, Prof. Dr. Udo Hahn, Sven Büchel
Übungsblatt 5 vom 25.11.2016
Abgabe bis 01.12.16, 23.59 Uhr; per Email (PDF-Format) an
sven-eric.buechel@uni-jena.de

Aufgabe 1 : Algorithmen verstehen (4 Punkte)

a) Beschreibung (2 Punkte)

Beschreiben Sie **präzise** in ganzen Sätzen, was der folgende Algorithmus für die englische Sprache macht. Gehen Sie dabei vor allem auf die Gesamtaufgabe des Programms ein. Würden Sie einen präziseren Namen als `"umwandlung"` vorschlagen?

```
umwandlung(↓eingabewort, ↑ausgabewort)
    endung ← eingabewort[length(eingabewort)−3:length(eingabewort)]
    if endung = "ies"
        ausgabewort ← eingabewort[0:length(eingabewort)−3]
        ausgabewort ← ausgabewort + "y"
    else
        ausgabewort ← eingabewort
```

Lösung: Der Algorithmus nimmt ein Wort als `eingabewort` entgegen und speichert die letzten drei Zeichen des Wortes in der Variablen `endung`. Diese Variable wird daraufhin untersucht, ob sie die Zeichenkette `"ies"` enthält. Ist dies der Fall, wird diese Endung durch `"y"` ersetzt und das Ergebnis in `ausgabewort` gespeichert. Damit werden Wörter, die auf ”ies“ enden so umgeformt, dass ”ies“ entfernt und durch ”y“ ersetzt wird. Das bedeutet, dass englische Substantive, die im Singular auf ”y“ und im Plural auf ”ies“ enden (city, baby, party...nicht zu verwechseln mit den eingedeutschten Varianten, die im Deutschen auch im Plural auf ”ys“ enden (”Handys“)), korrekt von ihrer Plural- auf ihre Singularform zurück geführt werden. Ebenso mit Verben, die im Infinitiv auf ”y“ und in der dritten Person Singular auf ”ies“ enden. Andere Wörter bleiben unmodifiziert.

b) Identifikation von Fehlerquellen (2 Punkte)

Hat der Algorithmus aus der vorigen Teilaufgabe Unzulänglichkeiten? Welche? Beschreiben Sie informell, wie die Probleme das Algorithmus' behoben werden könnten. Benutzen Sie auch hier bitte ganze Sätze!

Lösung: Unabhängig von der linguistischen Interpretation ist der Algorithmus anfällig für Eingaben mit weniger als 3 Zeichen. Allgemein geht der Algorithmus für Wörter fehl, die zwar auf ”ies“ enden, deren Rückführung auf eine ”y“ Endung aber keinen Sinn ergibt. Beispiele dafür sind einige Verben wie ”lie“ (he lies) oder ”die“ (he dies). Lösungsansatz hierfür wäre etwa ein Wörterbuch, in dem unregelmäßige Formen aufgeführt sind und gegen das abgeglichen werden kann. Zudem könnte es helfen, die Wortart des Eingabewortes zu kennen und die Reduktion nur für Substantive und Verben durchzuführen. Falls ein Substantiv vorliegt, wäre es zudem hilfreich zu wissen, ob es überhaupt im Plural steht. Außerdem könnte ein Grundformwörterbuch helfen: Man könnte testen, ob eine zurückgeführte Wortform im Grundformwörterbuch zu finden ist. Falls nicht, könnte man davon ausgehen, es mit einem Sonderfall zu tun zu haben und die Reduktion nicht durchführen. Der Algorithmus könnte für solche Fälle das Wort unverändert zurück geben. Damit würde davon ausgegangen, dass Wörter, die nicht bearbeitet werden können, bereits in ihrer Grundform stehen oder zumindest in keiner Form, die dieser Algorithmus bearbeitet.

Aufgabe 2 : Algorithmus zur Satzsegmentierung (6 Punkte)

Eine der grundlegendsten Aufgabe in der Computerlinguistik ist es, in einem Fließtext Sätze zu erkennen. Ein Programm zur Satzerkennung bekommt als Eingabe einen Text, beispielsweise "Peter lief nach draußen. Seine Jacke wurde vom Regen durchweicht.". Die Ausgabe des Programms wären dann die Sätze "Peter lief nach draußen." und "Seine Jacke wurde vom Regen durchweicht."

a) (2 Punkt)

Eine naive Herangehensweise zur Satzerkennung ergibt sich, wenn jeder Punkt im Text als Satzgrenze aufgefasst wird. Welche Probleme sehen Sie bei diesem Vorgehen? Nennen Sie mindestens vier Beispiele.

Lösung: Der Punkt ist in seiner graphematischen Bedeutung ambig. Obwohl die meisten Punkte in gewöhnlichen Texten tatsächlich ein Satzende markieren, führen Punkte von

- Abkürzungen
- Datumsangaben
- Zeitangaben
- Dezimalangaben (1.000)
- etc.

zur fälschlichen Satztrennung. Als Unteraufgabe der Satzerkennung gilt es daher, diejenigen Punkte zu identifizieren, die kein Satzende markieren.

b) (4 Punkte)

Geben Sie einen einfachen Algorithmus an, der einen Text entgegen nimmt und diesen satzweise ausgibt. Hinweise:

- Als Hilfestellung sei die Funktion `getWords(text)` gegeben. Diese Funktion bekommt einen Text, trennt ihn an Leerzeichen und gibt die so erhaltenen Wörter (ggf. mit Interpunktion!) in einer Liste zurück. Die Leerzeichen selbst gehen dabei verloren. Beispiel: `words` ← `getWords("Peter lief nach draußen.")` zerlegt den eingegeben Satz in die Elemente `"Peter"`, `"lief"`, `"nach"` und `"draußen."` (man beachte, dass das letzte Element auch den Punkt enthält, da er nicht durch ein Leerzeichen abgetrennt ist). Die Variable `words` enthält nun alle diese Wörter. Wie gehabt kann mit `words[0]` auf das 1te Wort, mit `words[1]` auf das 2te Wort etc. zugegriffen werden.
- Als weitere Hilfe sei eine Liste aller Titel `titleList` gegeben. Diese Liste enthalte beispielsweise "Dr.", "Prof.", "Dipl." etc. Diese Liste kann verwendet werden, um zu überprüfen, ob ein Wort eine Abkürzung oder ein Satzende darstellt. Bsp: `if wort in titleList: ...`
- Eine einzelne Variable kann beliebig viele Zeichen und Wörter enthalten. Betrachten Sie folgendes Beispiel:

  ```
  wort1 ← "Peter"
  wort2 ← "lief"
  satz ← wort1 + " " + wort2
  ```

  Die Variable `satz` enthält nun die Zeichenkette `"Peter lief"` (man beachte das Leerzeichen, das wieder hinzugefügt werden muss).
- Eine Variable kann "geleert" werden, indem man ihr die leere Zeichenkette zuweist: `satz` ← `""`.

Lösung:

```
text ← <eingabe>
words ← getWords(text)
sentence ← ""
for i from 0 to length(words)−1 :     # auch moeglich mit: for word in words
    sentence ← sentence + " " + words[i]
    if words[i][length(words[i])−1] = ".":
        if words[i] not in titleList
            print sentence
            sentence ← ""
```