Übung zur Vorlesung “Einführung in die Computerlinguistik und Sprachtechnologie”

Wintersemester 2016/2017, Prof. Dr. Udo Hahn, Sven Büchel
Übungsblatt 6 vom 23.12.2016
Abgabe bis 05.01.2017, 23.59 Uhr; per Email (PDF-Format) an
sven-eric.buechel@uni-jena.de

Aufgabe 1

Unten stehend sehen Sie eine einfache Variante des Porter-Stemmers in Pseudocode.

a) Verstehen

Geben Sie für die nachstehenden Wörter an, in welchen Zeilen des Pseudocodes jeweils eine Veränderung des Eingabeworts durchgeführt wird.

- hospitalization
- unpredictability
- hopelessness
- replacements

Dabei reicht es, wenn Sie die jeweiligen Zeilen innerhalb der Funktionen `step1`...`step6` angeben.

Lösung

- hospitalization: D14, D26, E3.
- inpredictability: C3, D23, E8.
- hopelessness: D29.
- replacements: A5, E10.

b) Trace

Schreiben Sie eine verkürzte Trace für folgenden Befehl:

```
H1        print stem_word("lemmatizations").
```

Wie aus der Übung bekannt, sollte die Trace folgende Elemente beinhalten:

- Die Zeilennummer der aktuellen Operation.
- Variablenzuweisungen, z.B.: `x` ← `"wort"`
- Funktionsaufrufe, z.B.: `fun(`↓ `"parameter")`
- Funktionsrückgaben, z.B.: `fun(`↑ `"rueckgabe")`
- Ausgaben, z.B.: `print "ausgabe"`

Schreiben Sie jeweils nur einen dieser Verarbeitungsschritte pro Zeile Ihrer Trace, auch wenn mehrere mehrere in einer Programmzeile ausgeführt werden. Achten Sie auf konsequente Einrückung! Operationen des Hauptprogramms werden nicht eingerückt, Operationen innerhalb von Funktionen sind jeweils eine Ebene weiter eingerückt als die aufrufende Instanz.

Aufgrund des Umfang des Pseudocode, kürzen Sie die Trace bitte folgendermaßen ab:

- Für die 5 Hilfsfunktionen (`m`, `star_d`, `star_vowel_star`, `star_o` und `replace`) schreiben Sie bitte *nicht* die internen Verarbeitungsschritte auf, sondern nur den Funktionsaufruf und die Rückgabe.
- Häufig wird die Funktion `replace` aufgerufen und die Rückgabe in die Variable `raw` geschrieben, ohne dass sich hierdurch deren Wert ändert (z.B. `raw ← replace(raw, "at", "ate")`). Schreiben Sie diese Funktionsaufrufe, -rückgaben und Variablenzuweisung nur dann auf, wenn sich der Wert der Variable ändert (also tatsächliche eine Modifkation des Wortes stattfindet). In allen anderen Fällen, deuten Sie dies bitte nur durch Auslassungszeichen unter Angabe der so zusammengefassten Zeilen an (z.B. `E10...E14` *keine Änderung*).

Lösung

```
H1 stem_word(↓"lemmatizations")
        P1 word ←"lemmatizations"
        2  stem ←"lemmatizations"
        3  step1(↓"lemmatizations")
                A1 raw ←"lemmatizations"
                2...3 keine Aenderungen
                5  replace(↓"lemmatizations", ↓"s", ↓"")
                5  replace(↑"lemmatization")
                A6 processed ←"lemmatization"
        3  step1(↑"lemmatization")
        3  stem ←"lemmatization"
        4  step2(↓"lemmatization")
                B1 raw ←"lemmatization"
                3  keine Aenderung
                16 processed ←"lemmatization"
        4  step2(↑"lemmatization")
        4  stem ← "lemmatization"
        5  step3(↓"lemmtization")
                C1 raw ←"lemmatization"
                4  processed ← "lemmatization"
        5  step3(↑"lemmatization")
        5  stem ← "lemmatization"
        6  step4(↓"lemmatization")
                D1 raw ← "lemmatization"
                3...13 keine Aenderung
                14 replace(↓"lemmatization", ↓"ization", ↓"ize")
                14 replace(↑"lemmatize")
                14 raw ← "lemmatize"
                15...29 keine Aenderung
                30  processed ← "lemmatize"
        6  step4(↑"lemmatize")
        6  stem ← "lemmatize"
        7  step5(↓"lemmatize")
                E1 raw ← "lemmatize"
                2  m(↓"lemmatize")
                2  m(↑3)
                3...19 keine Aenderung
                20  replace(↓"lemmatize", ↓"ize",↓"")
                20  replace(↑"lemmat")
                21  processed ← "lemmat"
        7  step5(↑"lemmat")
        7  stem ← "lemmat"
        8  step6(↓"lemmat")
                F1 raw ← "lemmat"
                2  m(↓"lemmat")
                2  m(↑2)
                3  keine Aenderung
                4  m(↓"lemmat")
```

```
                4  m(↑2)
                4  star_o(↓"lemmat")
                4  star_o(↑TRUE)
                6  m(↓"lemmat")
                6  m(↑2)
                8  processed ← "lemmat"
        8  step6(↑"lemmat")
        8  stem ← "lemmat"
2  stem_word(↑"lemmat")
2  print "lemmat"
```

Porter Stemmer

```
        //prueft ob aktuelles wort auf doppelkonsonant endet
V1      def star_d(↓word, ↑out)
2               out ← word[-1] = word[-2]

W1      def replace(↓word, ↓original, ↓replacement, ↑new_word)
2               if word ends with replacement
3                       //ersetzt die Endung von word durch replacement
4                       word ← word[:-length(ending)] + replacement
5               new_word ← word

        //zaelt Vokal-Konsonanten-Sequenzen im akutellen Wort (entspricht m() )
X1      def m(↓ word, ↑ count)
2               vowels ← ["a", "e", "i", "o", "u", "y"]
3               count ← 0
4               for i in 1...length(word)-1
5                       if word[i] not in vowels and word[i-1] in vowels
6                               count ← count+1

        // Prueft ob ein Vokal innerhalb des Stamms ist (entspricht *v*)
Y1      def star_vowel_star(↓word, ↓suffix, ↑out)
2               vowels ← ["a", "e", "i", "o", "u", "y"]
3               out ← FALSE
4               stem ← word[:-length(suffix)]
5               for v in vowels
6                       if v in stem
7                               out ← TRUE

        //prueft ob aktuelles Wort auf mit KONSONANT-VOKAL-KONSONANT endet
Z1      def star_o(↓word, ↑out)
2               a ← ["w", "x", "y"]
3               vowels ← ["a", "e", "i", "o", "u", "y"]
4               out ← word[-3] not in vowels and word[-2] in vowels and\
                        word[-1] not in vowels and word[-1] not in a

A1      def step1 (↓raw, ↑processed)
2               raw ← replace(raw, "sses", "ss")
3               raw ← replace(raw, "ies", "i")
4               if not raw[-2] = "s"
5                       raw ← replace(raw, "s", "")
6               processed ← raw

B1      def step2 (↓raw, ↑processed)
2               if m(raw) > 1
3                       raw ← replace(raw, "eed", "ee")
4               if (star_vowel_star(raw, "ed") and raw ends with "ed") or\
                        (star_vowel_star(raw, "ing") and raw ends with "ing")
```

```
5                       raw ← replace(raw, "ed", "")
6                       raw ← replace(raw, "ing", "")
7                       //Sonderschritt zum Wiederdranhaengen von "e"
8                       raw ← replace(raw, "at", "ate")
9                       raw ← replace(raw, "bl", "ble")
10                      raw ← replace(raw, "iz", "ize")
11                      //Doppelkonsonant vereinfachen
12                      if (not raw[-1] in ["l", "s", "z"]) and (star_d(raw))
13                              raw ← raw[:-1]
14                      if m(raw) = 1 and star_o(raw)
15                              raw ← raw + "e"
16              processed ← raw

C1      def step3(↓ raw, ↑ processed)
2         if star_vowel_star(raw, "y")
3               raw ← replace(raw, "y", "i")
4         processed ← raw

D1      def step4 (↓ raw, ↑ processed)
2         if m(raw) > 0
3           raw ← replace(raw, "ational", "ate")
4           raw ← replace(raw, "tional", "tion")
5           raw ← replace(raw, "enci", "ence")
6           raw ← replace(raw, "anci", "ance")
7           raw ← replace(raw, "izer", "ize")
8           raw ← replace(raw, "abli", "able")
9           raw ← replace(raw, "ational", "ate")
10          raw ← replace(raw, "alli", "al")
11          raw ← replace(raw, "entli", "ent")
12          raw ← replace(raw, "eli", "e")
13          raw ← replace(raw, "ousli", "ous")
14          raw ← replace(raw, "ization", "ize")
15          raw ← replace(raw, "ation", "ate")
16          raw ← replace(raw, "ator", "ate")
17          raw ← replace(raw, "alism", "al")
18          raw ← replace(raw, "iveness", "ive")
19          raw ← replace(raw, "fulness", "ful")
20          raw ← replace(raw, "ousness", "ous")
21          raw ← replace(raw, "aliti", "al")
22          raw ← replace(raw, "iviti", "ive")
23          raw ← replace(raw, "biliti", "ble")
24          raw ← replace(raw, "icate", "ic")
25          raw ← replace(raw, "ative", "")
26          raw ← replace(raw, "alize", "al")
27          raw ← replace(raw, "iciti", "ic")
28          raw ← replace(raw, "ful", "")
29          raw ← replace(raw, "ness", "")
30        processed ← raw

E1      def step5 (↓ raw, ↑ processed)
2         if m(raw) > 1
3           raw ← replace(raw, "al", "")
4           raw ← replace(raw, "ance", "")
5           raw ← replace(raw, "ence", "")
6           raw ← replace(raw, "er", "")
7           raw ← replace(raw, "ic", "")
8           raw ← replace(raw, "able", "")
9           raw ← replace(raw, "ant", "")
10          raw ← replace(raw, "ement", "")
11          raw ← replace(raw, "ent", "")
```

```
12          raw ← replace(raw, "tion", "")
13          raw ← replace(raw, "sion", "")
14          raw ← replace(raw, "ou", "")
15          raw ← replace(raw, "ism", "")
16          raw ← replace(raw, "ate", "")
17          raw ← replace(raw, "iti", "")
18          raw ← replace(raw, "ous", "")
19          raw ← replace(raw, "ive", "")
20          raw ← replace(raw, "ize", "")
21        processed ← raw

F1      def step6 (↓ raw, ↑ processed)
2         if m(raw) > 1
3           raw ← replace(raw, "e", "")
4         if m(raw) = 1 and not star_o(raw)
5           raw ← replace(raw, "e", "")
6         if m(raw) > 1 and raw ends with "ll"
7           raw ← replace(raw, "ll", "l")
8         processed ← raw



P1      def stem_word (↓word, ↑stem)
2         stem ← lower_case(word) //alle Grossbuchstaben zu Kleinbuchstaben
3         stem ← step1(stem)
4         stem ← step2(stem)
5         stem ← step3(stem)
6         stem ← step4(stem)
7         stem ← step5(stem)
8         stem ← step6(stem)
```