

# Programmieraufgabe Perl

## Minimum Edit Distance

Wintersemester 2005/2006

„*Minimum Edit Distance*“ ist ein einfacher Algorithmus zur Korrektur von Tippfehlern. Er berechnet die minimal nötige Anzahl an Lösch- oder Einfügeoperationen, um von einer Zeichenkette zu einer anderen zu gelangen und ermöglicht so, zu einem falsch geschriebenen Wort eine Liste der wahrscheinlichsten (d.h. „ähnlichsten“) richtigen aus einem Lexikon auszuwählen. Dabei können Löschungen, Einfügungen und Ersetzungen gleich gewichtet werden oder anhand einer Tabelle von empirisch ermittelten häufigen Tippfehlern oder Distanzen auf der jeweils verwendeten Tastatur verschiedene Wahrscheinlichkeiten erhalten. Eine genauere Beschreibung des Algorithmus findet sich in Jurafsky and Martin: *Speech and Language Processing* (CLUE-Bibliothek *CL ju 3*). In Pseudocode sieht er folgendermaßen aus (Achtung: die beiden ersten *for*-Schleifen fehlen in der Beschreibung bei Jurafsky/Martin!):

```
function MIN-EDIT-DISTANCE(target,source) returns min-distance
  n ←LENGTH(target)
  m ←LENGTH(source)
  Create a distance matrix distance[n+1,m+1]
  distance[0,0] ←0
  for each column i from 1 to n+1 do
    distance[i,0] ←distance[i-1,0] + ins-cost(targeti)
  for each row j from 1 to m+1 do
    distance[0,j] ←distance[0,j-1] + ins-cost(sourcej)
  for each column i from 1 to n+1 do
    for each row j from 1 to m+1 do
      distance[i,j] ←MIN(distance[i-1,j] + ins-cost(targeti),
                        distance[i-1,j-1] + subst-cost(sourcej,targeti),
                        distance[i,j-1] + del-cost(sourcej),
```

D.h.: es wird ein zweidimensionales Array angelegt, das so breit wie der Ziel- und so hoch wie der Quellstring ist. Dieses Array wird spaltenweise gefüllt mit Werten, die sich aus den jeweils angrenzenden Arrayelementen und den „Kosten“ einer Löschung, Einfügung oder Ersetzung ergeben. Am Ende steht die minimale Distanz in *Array*[*n,m*].

## 1 Aufgabe

1. Schreiben Sie eine Funktion *MinEditDistance*, die zwei Strings als Argumente nimmt und die numerische Distanz zurückgibt. Die Kosten für eine

Einfügung/Löschung (*ins-cost()* und *del-cost()*) können Sie als Konstante 1 setzen. Die Kosten einer Ersetzung sind 2, falls sich die beiden Strings an Position *i* bzw. *j* unterscheiden, ansonsten 0.

2. Erzeugen Sie eine Wortliste aus dem LIMAS-Korpus, z.B. mit einer leicht modifizierten Version des Frequenzlistengenerators aus dem Seminar, oder mit Unix-Kommandos.
3. Schreiben Sie ein Programm, das als Kommandozeilenargumente erstens den Dateinamen einer Wortliste und zweitens ein zu prüfendes Wort nimmt, und dieses Wort mittels *MinEditDistance* mit allen Wörtern der Liste vergleicht. Ausgegeben werden soll eine Liste der Wörter mit der niedrigsten Distanz.

## 2 Tips

1. `use constant NAME => Wert;` definiert echte Konstanten
2. Eine kleine Unterfunktion *subst\_cost* erleichtert die Formulierung des `MIN()`-Ausdrucks. Da ein dreiwertiges `min()` nicht existiert, lässt sich `min(a,b,c)` durch `min(a,min(b,c))` ersetzen.
3. Die Liste der Wörter mit der niedrigsten Distanz lässt sich wie für die Frequenzliste mittels eines Hashes erzeugen, das für jedes Wort die entsprechende Distanz speichert und am Ende nach Einträgen mit der niedrigsten gefundenen gefiltert wird. Deutlich speichersparender ist aber ein Array, das nur diejenigen Wörter mit der jeweils niedrigsten Distanz enthält. Wird ein Wort mit niedrigerer Distanz als die aktuell niedrigste gefunden, kann dem Array eine leere Liste zugewiesen und das neue Wort neu eingetragen werden.

## 3 Abgabe

Lösungen bitte bis in drei Wochen (6. Februar 2006) per Email an `bethke@linguistik.uni-erlangen.de` mailen, möglichst vom Linguistik-Account und der Einfachheit halber unter Angabe der Matrikelnummer (und am liebsten PGP-signiert :-))

Viel Erfolg!