Levenshtein-Abstand

Grundlagen: Levenshtein-Abstand, Viterbi-Algorithmus

Der Levenshtein-Abstand (String-Edit-Distanz) berechnet die minimale Anzahl von Löschungs-, Einfügungs- und Ersetzungsoperationen, die notwendig sind, um einen String a in einen String b zu überführen. Beispielsweise kann *aufgetaucht* mit zwei Löschoperationen, einer Ersetzungsoperation und einer Einfügeoperation in *auftauchen* transformiert werden.

Den Levenshtein-Abstand von $a = a_1...a_n$ und $b = b_1...b_m$ berechnet man am besten mit Hilfe einer Matrix d der Dimension $(n+1) \times (m+1)$, deren Einträge $d[i,k]$ den minimalen Levenshtein-Abstand zwischen dem Präfix $a_1...a_i$ und dem Präfix $b_1...b_k$ angeben. $d[i,k]$ wird mit dynamischer Programmierung wie folgt berechnet:

$$d[i,k] := min \begin{pmatrix} d[i-1,k]+1, \\ d[i-1,k-1]+[a_i \neq b_k], \\ d[i,k-1]+1 \end{pmatrix}$$

Der Klammeroperator $[P]$ bildet den Wahrheitswert P auf 1 ab, falls P wahr ist, und sonst auf 0. $d[n,m]$ liefert den Levenshtein-Abstand von a und b. $d[0,0]$ wird mit 0 initialisiert.

Schreiben Sie ein Programm, welches eine Datei mit einem Wortpaar pro Zeile einliest (mit Tabulator als Trennzeichen) und jeweils den Levenshtein-Abstand berechnet und auf den Bildschirm ausgibt.

Aufruf: `python levenshtein.py wordpairs.txt`

Erweitern Sie anschließend das Programm so, dass es zusätzlich alle optimalen Alignierungen der Buchstaben berechnet und ausgibt. Die optimalen Alignierungen des Stringpaares *aa a* sind:

a:a a:
a: a:a

Geben Sie nur diese Programmvariante ab. Bei dieser Aufgabe kommt es besonders auf kurzen und einfachen Code an.

Vorüberlegungen

- Welcher Operation entspricht jeder Term in der Minimum-Operation oben jeweils?
- Rechnen Sie das Beispiel *abgibt-abgegeben* durch.
- Die Felder in der ersten Zeile und der ersten Spalte der Matrix sind Spezialfälle. Überlegen Sie, was hier anders ist.
- Wie können Sie die optimalen Alignierungen der Buchstabenfolgen aus der Matrix extrahieren?