Script generated by TTT

Title: Seidl: GAD (12.07.2016)

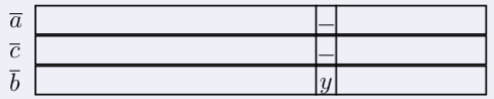
Date: Tue Jul 12 14:22:48 CEST 2016

Duration: 48:11 min

Pages: 26

Beziehung zwischen Edit- und Alignment-Distanz

Beweis.



Skizze: $s_{n+1} = (x, y)$ ist eine Insertion

- $(\overline{a}, \overline{c})$ ist jetzt wegen der Spalte $(-,-)$ kein Alignment mehr.
- jedoch nur noch interessant: Alignment $(\overline{a}, \overline{b})$ von a und b

$$\begin{aligned} w(\overline{a}, \overline{b}) &= w(\overline{a}, \overline{c}) + w(-, y) \\ &\quad \text{Nach Induktionsvoraussetzung} \\ &\leq w(s') + w(s_{n+1}) = w(s) \end{aligned}$$

□

Beziehung zwischen Edit- und Alignment-Distanz

Also ist die Alignment-Distanz durch die Edit-Distanz beschränkt, falls die zugrunde liegende Kostenfunktion eine Metrik ist:

Folgerung

Ist $w : \overline{\Sigma}_0^2 \to \mathbb{R}_+$ *eine metrische Kostenfunktion, dann gilt für alle* $a, b \in \Sigma^*$*:*

$$\overline{d}_w(a, b) \leq d_w(a, b).$$

Zusammengefasst ergibt sich für den Fall einer metrischen Kostenfunktion die Gleichheit von Edit- und Alignment-Distanz:

Theorem

Ist w *eine Metrik, dann gilt für* $a, b \in \Sigma^*$*:* $d_w(a, b) = \overline{d}_w(a, b)$.

Globale Alignments

Problem

GLOBALES ALIGNMENT

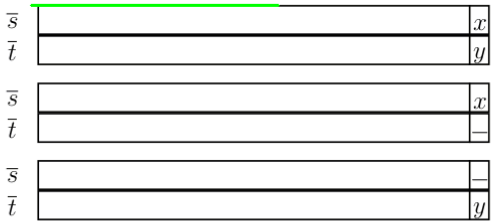
Eingabe: $s \in \Sigma^n$, $t = \in \Sigma^m$,
w*: Kostenfunktion für Distanz- oder Ähnlichkeitsmaß* μ.
Gesucht: *optimales globales Alignment* $(\overline{s}, \overline{t})$ *für* s *und* t*, d.h.*
$\mu(s, t) = w(\overline{s}, \overline{t})$

- zunächst mit Distanzmaßen
- Abwandlung für Ähnlichkeitsmaße meist offensichtlich

Optimale Alignments

- Annahme: sei $(\overline{s}, \overline{t})$ ein optimales Alignment für s und t
- 3 Möglichkeiten, wie die letzte Spalte $(\overline{t}_{|\overline{t}|}, \overline{s}_{|\overline{s}|})$ dieses optimalen Alignments aussehen kann:
  - Entweder wurde $x = s_n$ durch $y = t_m$ substituiert (oben)
  - oder es wurde das letzte Zeichen $x = s_n$ in s gelöscht (Mitte)
  - oder es wurde das letzte Zeichen $y = t_m$ in t eingefügt (unten).

Optimale Alignments



Skizze: Optimales Alignment mit Substitution/Match, Insertion bzw. Deletion am Ende

- In allen drei Fällen ist das Alignment, das durch Streichen der letzten Spalte entsteht, also $(\overline{s}_1 \cdots \overline{s}_{|\overline{s}|-1}, \overline{t}_1 \cdots \overline{t}_{|\overline{t}|-1})$, ebenfalls ein optimales Alignment für
  - $s_1 \cdots s_{n-1}$ mit $t_1 \cdots t_{m-1}$,
  - $s_1 \cdots s_{n-1}$ mit $t_1 \cdots t_m$ bzw.
  - $s_1 \cdots s_n$ mit $t_1 \cdots t_{m-1}$.

Optimale Alignments

Lemma

Sei $(\overline{a}, \overline{b})$ ein optimales Alignment für $a, b \in \Sigma^*$ für ein gegebenes Distanz- oder Ähnlichkeitsmaß.
Für $i \leq j \in [1 : |\overline{a}|]$ ist dann $(\overline{a}_i \cdots \overline{a}_j, \overline{b}_i \cdots \overline{b}_j)$ ein optimales Alignment für $a' = \overline{a}_i \cdots \overline{a}_j|_\Sigma$ und $b' = \overline{b}_i \cdots \overline{b}_j|_\Sigma$.

Optimale Alignments

Beweis.

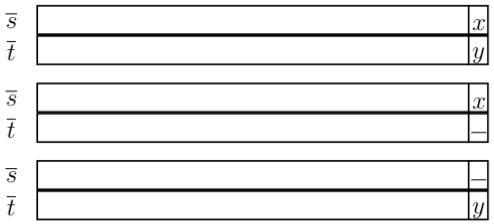
- Sei $(\overline{a}, \overline{b})$ ein optimales Alignment für $a, b \in \Sigma^*$.
- Für einen Widerspruchsbeweis nehmen wir an, dass $(\overline{a}_i \cdots \overline{a}_j, \overline{b}_i \cdots \overline{b}_j)$ kein optimales Alignment für $a', b' \in \Sigma$ ist.
- Sei also $(\widetilde{a'}, \widetilde{b'})$ ein optimales Alignment für a' und b'.
- Dann ist aber nach Definition der Kosten eines Alignments (unabhängig, ob Distanz- oder Ähnlichkeitsmaß) das Alignment

$$(\overline{a}_1 \cdots \overline{a}_{i-1} \cdot \widetilde{a'} \cdot \overline{a}_{j+1} \cdots \overline{a}_n, \overline{b}_1 \cdots \overline{b}_{i-1} \cdot \widetilde{b'} \cdot \overline{b}_{j+1} \cdots \overline{b}_n)$$

ein besseres Alignment als $(\overline{a}, \overline{b})$ (Widerspruch)

□

Optimale Alignments



Skizze: Optimales Alignment mit Substitution/Match, Insertion bzw. Deletion am Ende

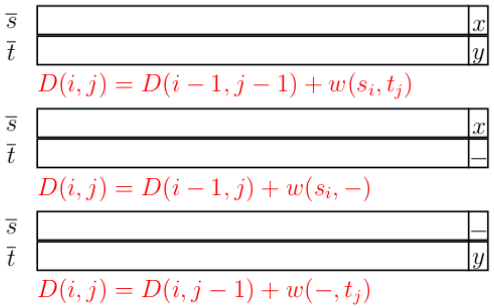
- In allen drei Fällen ist das Alignment, das durch Streichen der letzten Spalte entsteht, also $(\overline{s}_1 \cdots \overline{s}_{|\overline{s}|-1}, \overline{t}_1 \cdots \overline{t}_{|\overline{t}|-1})$, ebenfalls ein optimales Alignment für
  - $s_1 \cdots s_{n-1}$ mit $t_1 \cdots t_{m-1}$,
  - $s_1 \cdots s_{n-1}$ mit $t_1 \cdots t_m$ bzw.
  - $s_1 \cdots s_n$ mit $t_1 \cdots t_{m-1}$.

Needleman-Wunsch-Algorithmus

- berechnet ein optimales Alignment für zwei Sequenzen $s = s_1 \cdots s_n \in \Sigma^n$ und $t = t_1 \cdots t_m \in \Sigma^m$
- benutzt Matrix $D(i,j) = \mu(s_1 \cdots s_i, t_1 \cdots t_j)$, in der jeweils die Distanz eines optimalen Alignments für $s_1, \ldots s_i$ und $t_1, \ldots t_j$ gespeichert wird
- rekursive Berechnung der Matrix:

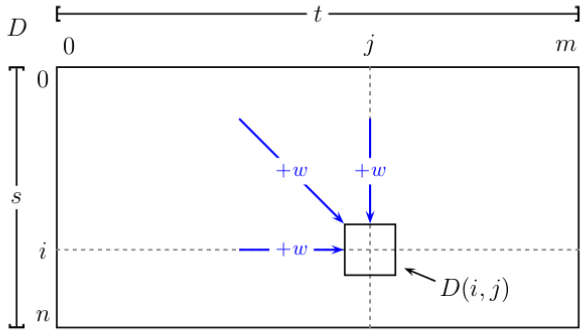
$$D(i,j) \;=\; \min \left\{ \begin{array}{lcl} D(i-1,j-1) & + & w(s_i,t_j), \\ D(i-1,j) & + & w(s_i,-), \\ D(i,j-1) & + & w(-,t_j) \end{array} \right\}.$$

Needleman-Wunsch-Algorithmus



Skizze: Erweiterung eines optimales Alignment zu $s_1 \cdots s_i$ mit $t_1 \cdots t_j$

Needleman-Wunsch-Algorithmus



Skizze: Berechnung optimaler Alignments nach Needleman-Wunsch

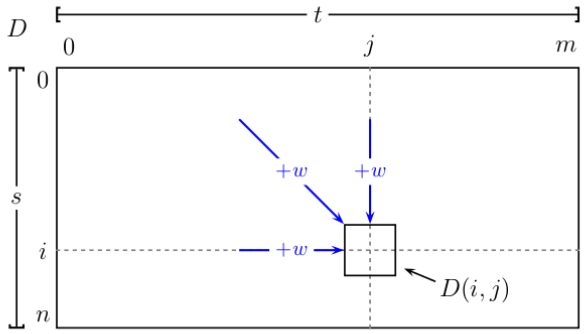
Needleman-Wunsch-Algorithmus

Prozedur SeqAlign(char $s[]$, int n, char $t[]$, int m)

$D[0,0] := 0$;
for $(i := 1; i \leq n; i{+}{+})$ **do**
  $D[i,0] := D[i-1,0] + w(s_i, -)$;
for $(j := 1; j \leq m; j{+}{+})$ **do**
  $D[0,j] := D[0,j-1] + w(-, t_j)$;
for $(i := 1; i \leq n; i{+}{+})$ **do**
  for $(j := 1; j \leq m; j{+}{+})$ **do**

$$D[i,j] := \min \left\{ \begin{array}{l} D[i-1,j] + w(s_i, -), \\ D[i,j-1] + w(-, t_j), \\ D[i-1,j-1] + w(s_i, t_j) \end{array} \right\};$$

Needleman-Wunsch-Algorithmus



Skizze: Berechnung optimaler Alignments nach Needleman-Wunsch

Needleman-Wunsch-Algorithmus

Prozedur SeqAlign(char $s[]$, int n, char $t[]$, int m)

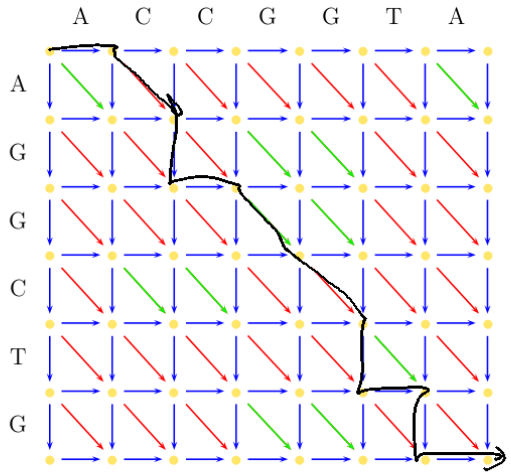
$D[0,0] := 0$;
for $(i := 1; i \leq n; i{+}{+})$ **do**
  $D[i,0] := D[i-1,0] + w(s_i, -)$;
for $(j := 1; j \leq m; j{+}{+})$ **do**
  $D[0,j] := D[0,j-1] + w(-, t_j)$;
for $(i := 1; i \leq n; i{+}{+})$ **do**
  for $(j := 1; j \leq m; j{+}{+})$ **do**

$$D[i,j] := \min \left\{ \begin{array}{l} D[i-1,j] + w(s_i, -), \\ D[i,j-1] + w(-, t_j), \\ D[i-1,j-1] + w(s_i, t_j) \end{array} \right\};$$

Needleman-Wunsch-Algorithmus: Beispiel

- Visualisierung am Beispiel: $s = AGGCTG$ und $t = ACCGGTA$
- 1. Schritt: Aufstellen des Edit-Graphen
- In Abhängigkeit von der jeweiligen Operation werden unterschiedliche Pfeile eingefügt:
  - blaue horizontale bzw. vertikale Pfeile: Insertionen bzw. Deletionen
  - rote diagonale Pfeile: Substitutionen
  - grüne diagonale Pfeile: Matches

Needleman-Wunsch-Algorithmus: Beispiel

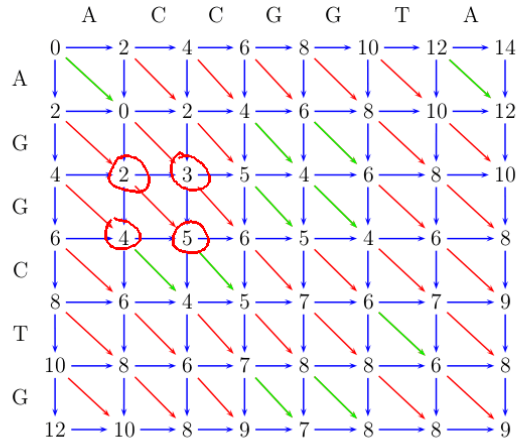


Skizze: Edit-Graph für s und t ohne Distanzen

Needleman-Wunsch-Algorithmus

- Nun werden die jeweiligen Distanzen des aktuellen Alignments (mit Hilfe der Rekursionsformel) eingetragen.
- In diesem Beispiel verursachen Einfügungen und Löschungen Kosten 2.
- Substitutionen verursachen hier Kosten 3.
- Ein Match verursacht keine Kosten (also 0).
- In der rechten unteren Ecke ($D(n,m)$) findet sich zum Schluss die Distanz eines optimalen Alignments.

Needleman-Wunsch-Algorithmus: Beispiel



Match $= 0$, Indel $= 2$, Subst $= 3$

Skizze: Edit-Graph für s und t mit Distanzen

Needleman-Wunsch-Algorithmus

Prozedur SeqAlign(char $s[]$, int n, char $t[]$, int m)

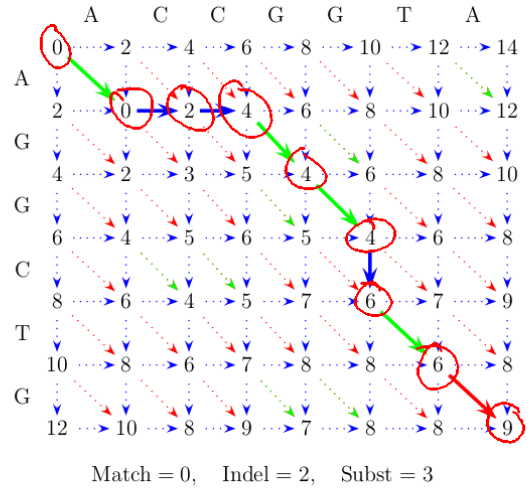
$D[0,0] := 0$;
for $(i := 1; i \le n; i{+}{+})$ **do**
  $D[i,0] := D[i-1,0] + w(s_i, -)$;
for $(j := 1; j \le m; j{+}{+})$ **do**
  $D[0,j] := D[0,j-1] + w(-, t_j)$;
for $(i := 1; i \le n; i{+}{+})$ **do**
  for $(j := 1; j \le m; j{+}{+})$ **do**

$$D[i,j] := \min \left\{ \begin{array}{l} D[i-1,j] + w(s_i,-), \\ D[i,j-1] + w(-,t_j), \\ D[i-1,j-1] + w(s_i,t_j) \end{array} \right\};$$

Needleman-Wunsch-Algorithmus

- Damit haben wir zwar den Wert eines optimalen Alignments für s und t bestimmt, kennen das Alignment an sich jedoch noch nicht.
- Dafür wird nun ein Pfad im Graphen von rechts unten nach links oben gesucht, der minimale Kosten verursacht.
- Gestartet wird in der rechten unteren Ecke.
- Als Vorgängerknoten wird nun der Knoten gewählt, der zuvor als Sieger bei der Minimum-Bildung hervorging. (Liefern mehrere Knoten die gleichen minimalen Kosten, kann einer davon frei gewählt werden. Meist geht man hier in einer vorher fest vorgegeben Reihenfolge bei Unentschieden vor, z.B. Insertion vor Substitution vor Deletion.)
- So verfährt man nun immer weiter, bis man in der linken oberen Ecke ankommt.

Needleman-Wunsch-Algorithmus: Beispiel



Match = 0, Indel = 2, Subst = 3

Skizze: Pfad im Edit-Graphen zur Bestimmung des Alignments

Needleman-Wunsch-Algorithmus: Beispiel

Nun kann man das optimale Alignment für s und t angeben. Dieses muss nur noch aus dem Edit-Graphen (entlang des gefundenen Pfades) abgelesen werden:

s :	A	–	–	G	G	C	T	G
t :	A	C	C	G	G	–	T	A

Beispiel: Optimales globales Alignment von s mit t

Needleman-Wunsch-Algorithmus

Theorem

Das optimale globale paarweise Sequenzen-Alignment für s und t mit $n = |s|$ und $m = |t|$ sowie die zugehörige Alignment-Distanz lassen sich mit dem Prinzip der Dynamischen Programmierung in Zeit $O(nm)$ und mit Platz $O(nm)$ berechnen.

Needleman-Wunsch-Algorithmus

Prozedur SeqAlign(char $s[]$, int n, char $t[]$, int m)

$D[0,0] := 0$;
for $(i := 1; i \leq n; i{+}{+})$ **do**
  $D[i,0] := D[i-1,0] + w(s_i, -)$;
for $(j := 1; j \leq m; j{+}{+})$ **do**
  $D[0,j] := D[0,j-1] + w(-, t_j)$;
for $(i := 1; i \leq n; i{+}{+})$ **do**
  for $(j := 1; j \leq m; j{+}{+})$ **do**

$$D[i,j] := \min \left\{ \begin{array}{l} D[i-1,j] + w(s_i, -), \\ D[i,j-1] + w(-, t_j), \\ D[i-1,j-1] + w(s_i, t_j) \end{array} \right\};$$

Needleman-Wunsch-Algorithmus

Theorem

Das optimale globale paarweise Sequenzen-Alignment für s und t mit $n = |s|$ und $m = |t|$ sowie die zugehörige Alignment-Distanz lassen sich mit dem Prinzip der Dynamischen Programmierung in Zeit $O(nm)$ und mit Platz $O(nm)$ berechnen.

Datenkompression

Problem:

- Dateien enthalten oft viel Redundanz (z.B. Wiederholungen) und nehmen mehr Speicherplatz ein als erforderlich

$\Rightarrow$ mit Wissen über die Struktur der Daten und Informationen über die Häufigkeit von Zeichen bzw. Wörtern kann man die Datei so kodieren, dass sie weniger Platz benötigt (Kompression)