Stringvergleich: Minimum Edit Distance

Katja Markert

Institut für Computerlinguistik
Uni Heidelberg
markert@cl.uni-heidelberg.de

November 5, 2019

Bisher und jetzt

- Bisher: Generierung und Erkennung von Strings
- mit der Hilfe von regulären Ausdrücken und Automaten
- Jetzt: Vergleich von Strings

Warum Stringvergleich?

1. **Rechtschreibkorrektur:** *giraffe* bessere Verbesserung für *graffe* als *graffity* (wenn man Kontext ignoriert). Warum?
2. **Rechtschreibkorrektur:** 0.05% - 38% der Wörter eines Textes sind inkorrekt, je nach Textsorte (siehe J&M,2nd edition, Ch 2)
3. ähnliche Argumente bei OCR oder handwriting recognition
4. **Koreferenzerkennung:** *United States*, *States*, *United States of America* wahrscheinlicher die gleiche Koreferenzkette als *United States*, *Mexico* und *Guatemala*

Rechtschreibkorrektur: Erkennung von Nichtwörtern

- Lexikonbasiert
- Markiere alle Wörter, die nicht im Lexikon verzeichnet sind
- Voraussetzung: korrekte Tokenisierung! (s. nächste Vorlesung)
- Würden Sie zu dieser Erkennung am liebsten ein besonders großes und vollständiges Lexikon benutzen?
- Ist die Erkennung von Nichtwörtern ausreichend zur Rechtschreibkorrektur?

Nicht-Wort erkannt! Wie korrigiere ich?

- Suche nach der wahrscheinlichsten korrekten Wortform
- Distanzmaß zwischen inkorrekter und potentiell korrekter Wortform
  – *graffe* vs. *giraffe, graffity, raffle*
- Einfachste Möglichkeit: **Minimum Edit Distance**
- berücksichtigt Kontext nicht
- nicht probabilistisch

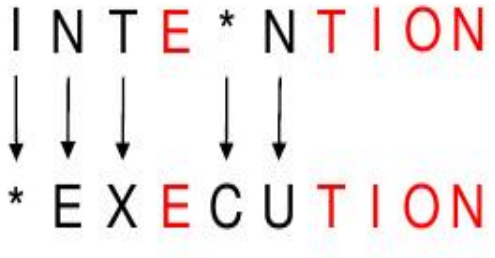
Minimum Edit Distance

Definition

MED zwischen zwei Zeichenketten ist die minimale Anzahl von Operationen, die benötigt wird, um die eine Zeichenkette in die andere zu überführen. Operationen sind hierbei *einfügen, ersetzen, löschen*.

Beispiel:

- 3 mal Ersetzen
- 1 mal Löschen
- 1 mal Einfügen



Alignierung/alignment zeigt die Korrespondenz zwischen zwei Zeichenketten und bestimmt Operationen.

Kosten der Editieroperationen

- Levenshtein Distanz (1966): Jede der drei Operationen hat die gleichen Kosten (=1) $\longrightarrow$ MED zwischen *Intention* und *Execution* = 5
- Alternative:
  - Einfügen und Löschen: Kosten 1
  - Ersetzen = Löschen + Einfügen. Kosten 2

  MED zwischen *Intention* und *Execution* = 8
- Kosten können also verschieden sein. MED ist kostenabhängig.

Wo ist das Problem?

Der Raum aller Editiersequenzen ist riesig

- Können nicht alle naiv durchsuchen
- Viele verschiedene Pfade führen zum gleichen Zustand.
- Sollten uns nur den kürzesten Pfad zu jedem Zustand merken

Lösung: Dynamische Programmierung

- Klasse von Algorithmen, die auf Tabellen operieren
- Probleme werden auf Teilprobleme zurückgeführt
- Berechnete Teilergebnisse werden abgespeichert
- Ein einmal berechnetes Teilproblem muss nicht nochmals berechnet werden

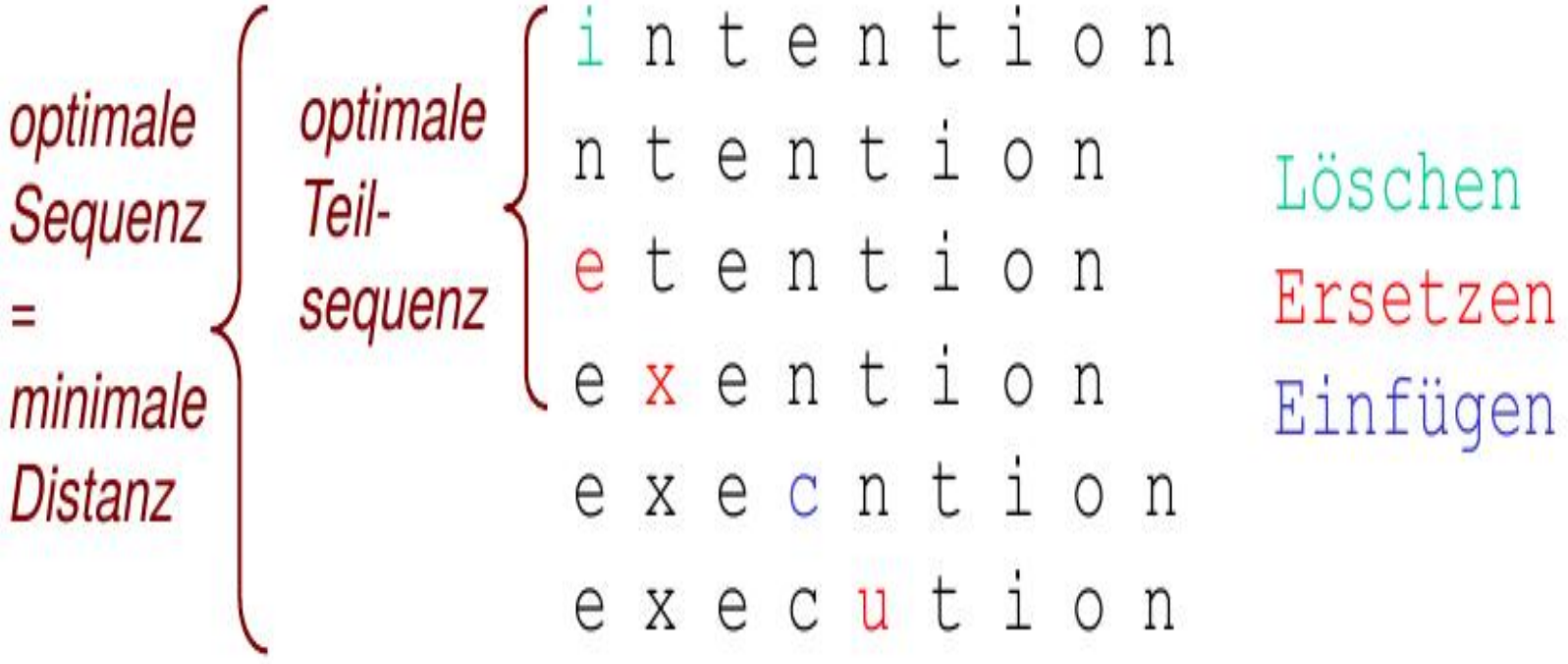
Varianten, die wir noch sehen werden

- Viterbi Algorithmus für HMM POS Tagging
- CKY (und Earley) Algorithmus (Chart Parsing)

Intuition

Haupteinsicht

Ist eine Sequenz von Änderungsoperationen optimal (minimal), so ist auch jede Teilsequenz optimal (minimal).



Intuition

Seien V, W Zeichenketten und x, y einzelne Zeichen. Dann gilt:

$$
\begin{aligned}
md(Vx, Wy) \quad = \quad & min(\\
& c_{sub}(x, y) + md(V, W), \\
& c_{del}(x) + md(V, Wy), \\
& c_{ins}(y) + md(Vx, W) \\
&)
\end{aligned}
$$

Beispiel:

$$
\begin{aligned}
md(inte, exec) \quad = \quad & min(\\
& c_{sub}(e, c) + md(int, exe), \\
& c_{del}(e) + md(int, exec), \\
& c_{ins}(c) + md(inte, exe) \\
&)
\end{aligned}
$$

Formalisiert für zwei gegebene Zeichenketten

Für eine Zeichenkette $X = x_1 \dots x_n$, definieren wir $X[1 \dots i] := x_1 \dots x_i$

Zwischen zwei Zeichenketten $X = x_1 \dots x_n$, $Y = y_1 \dots y_m$, sei $md(i,j) := md(X[1 \dots i], Y[1 \dots j])$.

Also ist $md(X,Y) = md(n,m)$

Formalisiert für zwei gegebene Zeichenketten

$X = x_1 \dots x_n$ source, $Y = y_1 \dots y_m$ target

Initialisierung:

- $md(0,0) = 0$
- $md(i,0) = md(i-1,0) + c_{del}([X(i)])$ für $1 < i \leq n$;
- $md(0,j) = md(0,j-1) + c_{ins}([Y(j)])$ für $1 < j \leq m$;

Rekursion:

Für alle $i = 1 \dots n$ für alle $j = 1 \dots m$

$$
\begin{aligned}
md(i,j) \quad = \quad & min(\\
& md(i-1,j) + c_{del}(X[i]), \\
& md(i-1,j-1) + c_{sub}(X[i], Y[j]), \\
& md(i,j-1) + c_{ins}(Y[j]) \\
&)
\end{aligned}
$$

Ausgabe: $md(n,m)$ ist Distanz

Mit Levenshtein

Kosten insertion, deletion 1, substitution 2, gleich lassen Null:

$$
\begin{aligned}
md(i,j) \quad = \quad & min(\\
& md(i-1,j)+1, \\
& md(i-1,j-1)+(2 \ \ oder \ \ 0), \\
& md(i,j-1)+1) \\
&)
\end{aligned}
$$

Als Tabelle

Tabelle von Größe (n+1) Zeilen und (m+1) Spalten.

X/Y	#	1	.	j	.	.	m
#							
1							
.							
.			i-1,j-1	i-1,j			
i			i, j-1	i,j			
.							
.							
n							

- i-1,j-1 $\rightarrow$ i,j: Ersetzung/substitution
- i-1,j $\rightarrow$ i,j: Löschen/deletion von source zu target
- i,j-1 $\rightarrow$ i,j: Einfügen/insertion von source zu target

Flexible Kosten

Die obige Formalisierung erlaubt flexible Kosten. Im folgenden Beispiel benutzen wir

- $c_{del}(X[i]) = 1$, unabh. von $X[i]$;
- $c_{ins}(Y[j]) = 1$ unabh. von $Y[j]$;
- $c_{sub}(X[i], Y[j]) = 2$ für $X[i] \neq Y[j]$,
- $c_{sub}(X[i], Y[j] = 0$ für $X[i] = Y[j]$

Beispiel: Initialisierung

X/Y	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1									
n	2									
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

Beispiel: Anfang

X/Y	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	**2**	**?**							
n	2									
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

Beispiel: Weiter

X/Y	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	**7**	**?**		
n	2									
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

Beispiel: Weiter

X/Y	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	7	**6**	7	8
n	2									
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

Beispiel: Weiter

X/Y	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	7	6	7	8
n	2	3	4	5	6	7	8	**7**	8	**7**
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

Bitte den Rest ausfüllen!

Beispiel: Weiter

X/Y	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	7	6	7	8
n	2	3	4	5	6	7	8	7	8	7
t	3	4	5	6	7	8	7	8	9	8
e	4	3	4	5	6	7	8	9	10	9
n	5	4	5	6	7	8	9	10	11	10
t	6	5	6	7	8	9	8	9	10	11
i	7	6	7	8	9	10	9	8	9	10
o	8	7	8	9	10	11	10	9	8	9
n	9	8	9	10	11	12	11	10	9	**8**

$md(X, Y) = 8$

Beispiel: Alignierung

- Wir können uns merken, wo unsere Werte hergekommen sind
- Dies ergibt ein (oder mehrere) *alignments* und damit eine Serie von substitutions, deletions, insertions.
- Jeder vollständige Pfad absteigender Kosten von (n, m) (untere rechte Ecke) nach (0,0) repräsentiert eine minimale Editiersequenz.

	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	**1**	↖←↑ 2	↖←↑ 3	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖ 6	← 7	← 8
n	2	↖←↑ **3**	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↑ 7	↖←↑ 8	↖ 7
t	3	↖←↑ 4	↖←↑ **5**	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖ 7	←↑ 8	↖←↑ 9	↑ 8
e	4	↖ 3	← 4	↖← **5**	← **6**	← 7	←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9
n	5	↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ **8**	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖↑ 10
t	6	↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖ **8**	← 9	← 10	←↑ 11
i	7	↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9	↖ **8**	← 9	← 10
o	8	↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↑ 10	↑ 9	↖ **8**	← 9
n	9	↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖←↑ 12	↑ 11	↑ 10	↑ 9	↖ **8**

Bild von Jurafsky und Martin, 3rd edition.

Laufzeit und Speicher

Laufzeit: $O(nm)$

Speicher = Tabellengröße: $O(nm)$

Andere Gewichte?

Kann es Sinn machen, unterschiedliche Gewichte für unterschiedliche Löschungen, Substitutionen etc zu verwenden?

Wenn Sie *Nanen* getippt hatten (fälschlicherweise), ist es dann wahrscheinlicher, dass Sie *Nasen* oder *Namen* tippen wollten (kontextunabhängig)?

Andere Gewichte?

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																								
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Von Dan Jurafsky

Minimum Edit Distance für Evaluation

Evaluation von MT und Spracherkennung

Spokesman	confirms		senior	government	adviser	was	shot	
Spokesman	said	the	senior		adviser	was	shot	dead
	S	I		D				I

Zusammenfassung

Sie können:

- Die Minimum Edit Distanz zwischen zwei Zeichenketten berechnen
- Die Alignierung ableiten
- Das Konzept der dynamischen Programmierung beschreiben

Aufgaben

- * Jurafsky und Martin, Chapter 2.5 (3rd edition, online)
- * Aufgabenblatt 3
- Interaktive Demo für Levenshtein Distanz (mit Ausgabe aller minimalen Editiersequenzen):
  `http://www.let.rug.nl/kleiweg/lev/`

Referenzen

- * Jurafsky und Martin, Chapter 2, 3rd edition (online)
- Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press.
- Bellman, R. (1957). *Dynamic Programming.* Princeton University Press.