Stringdistanzberechnung nach Levenshtein mit TUSCRIPT

Grundlagen und Anwendung

TUSTEP-Workshop, Blaubeuren, 9. Januar 2015

Matthias Schneider
schneiderm@uni-trier.de
Kompetenzzentrum für elektronische
Erschließungs- und Publikationsverfahren
in den Geisteswissenschaften/
Trier Center for Digital Humanities
Univeristät Trier
www.kompetenzzentrum.uni-trier.de
www.m-schneider.eu

Grundlagen – Was ist die Levenshtein-Distanz?

Definitionen

> **Levenshtein distance** —— a distance metric between two strings, not necessarily of the same length, given by the minimum number of symbol insertions, deletions and substitutions required to transform one string into the other, e. g. the Levenshtein distance between `zeitgeist` and `preterit` is 6.
> Stephen (1994), S. 208.

> Given strings x and y, with $|x|$ = m, $|y|$ = n, where $m, n > 0$ and $m \leq n$ and a generalised Levenshtein string-distance measure, d, find $d(x, y)$.
> When given as file difference problem, x and y represent two files, where x_i is the i^{th} line of x and y_j is the j^{th} line of y. The requirement is then to determine a minimal sequence of editing operations necessary to transform x into y.
> Stephen (1994), S. 42.

> We will say that a code K can correct s deletions, insertions, and reversals if any binary word can be obtained from no more than one word in K by s or fewer deletions, insertions, or reversals.
> Levenshtein (1966), S. 708 f.

> The three basic types of error in strings of discrete symbols are: (1) replacement, (2) insertion, (3) deletion of a symbol. (Interchange of two consecutive symbols is reduced to two of the above basic operations in many ways.)
> Kohonen (1985), S. 309.

- Levenshtein-Distanz LD(x, y) = Maß für den Unterschied respektive die Ähnlichkeit zwischen beliebig langen Strings/Zeichenketten/Symbolketten x, y[1]
- je größer die Levensthein-Distanz, umso geringer ist die Ähnlichkeit der Vergleichstrings (Eriksen et al. (2005), S. 310)
- LD ist symmetrisch; x und y können jeweils als Quell- oder Zielstring behandelt werden, ohne dass sich die LD ändert (Eriksen et al. (2005), S. 310); LD(x, y) = LD(y, x)
- Berechnung = minimale Anzahl von Änderungsoperationen (Umstellen, Hinzufügen, Löschen von Symbolen/Characters/Buchstaben), Wertung jeder Operation jeweils = 1
- Sonderfall *Damerau-Levenshtein-Distanz*: wertet Transposition von zwei Symbolen als Fehlerwert 1 (in der einfachen LD würde dieser Fall als 2 gewertet, da zwei Substitutionen notwendig sind)
- LD ist zugleich eine Verallgemeinerung der Hamming-Distanz; =Distanz zweier Strings gleicher (!) Länge, äquivalent zur Anzahl unterschiedlicher Symbolpositionen der Strings, (Stephen (1994), S. 207)

---

1 Alternative Bezeichnung: Levenshtein Edit Distance (LED). Vgl. einführend auch Michaela Geierhos, http://www.cis.uni-muenchen.de/~micha/praesentationen/rechtschreibkorrektur/Levenshtein.html (07.01.2015).

Beispiele (vgl. Matrizes im Anhang)

Gegeben seien ein String x mit dem Inhalt `haus` und ein String y mit dem Inhalt `baum`. Da der erste und der letzte Buchstabe von x substituiert werden müssen, um y zu erhalten (h → b, s → m), ergibt sich LD(haus, baum) = 2.

Gegeben seien ein String x mit dem Inhalt `spotten` und ein String y mit dem Inhalt `unsportlich`.
Um y aus x zu erhalten, sind folgende Schritte notwendig:
- Einfügung von `un` am Anfang (LD = LD + 2)
- Substitution von `t` durch `r` (LD = LD + 1)
- Substitution von `en` durch `li` (LD = LD + 2)
- Einfügung von `ch` (LD = LD + 2)
- LD(spotten, unsportlich) = 7

Variante: gewichtete Levenshtein-Distanz[2]

- ungewichtete Levenshtein-Distanz: Löschen, Hinzufügen und Tausch jeweils d = 1 (s. o.)
- gewichtete Levenshtein-Distanz: unterschiedliche Bewertung der einzelnen Operationen, z. B. Löschen = 2, Hinzufügen = 1, Substituieren = 1,5 in Abhängigkeit von Gegenstand und Forschungsfrage bzw. Anwendung
- Bsp.: Gewichtung anhand der Wahrscheinlichkeit auftretender Fehler ausgehend von den vorkommenden Symbolen (nach Kohonen (1985), S. 310):
  `WLD(A,B) = min {pa(i) + qb(i) + rc(i)}`
  mit a = Ersetzungen, b = Einfügungen, c = Löschen von Symbolen sowie p, q, und r als Wahrscheinlichkeitskoeffizienten

Standardisierung der Ergebnisse[3]

Ausgangsproblem: Die absolute LD ist nur im Bezug auf zwei konkrete Strings aussagekräftig und nicht zwischen unterschiedlichen Paaren vergleichbar (Tougard, Eriksen (2006), S. 242).
Exemplarisch hierzu drei Vergleiche, die allesamt eine LD von 3 aufweisen, sich jedoch durch die unterschiedlichen Umfänge der Vergleichstrings stark voneinander unterscheiden:

LD(phrasen, hase) = 3 → 3/7 des Strings muss verändert werden.

LD(philologie, philosophie) = 3 → 3/10 des Strings muss verändert werden.

LD({vulgata_1}, {vulgata_2}) = 3 → inhaltlich vernachlässigbare Abweichung.[4]

2 Garland et al. (2012) S. 1437.

3 Garland et al. (2012), S. 1423–1426

4 Bei diesem Beispiel ist der Vergleich von zwei Textversionen der Vulgata gemeint, die sich an lediglich 3 Stellen voneinander unterscheiden und damit im Gegensatz zu den vorhergehenden Beispielen eine nur minimale Abweichnung aufweisen.

Standardisierung durch inhärente Relativierung: Levenshtein Similarity Index (LSI)

$$\mathrm{LSI}(a, b) = 1 - \frac{\mathrm{LD}(a, b)}{\max\ (\mathrm{len}(a),\ \mathrm{len}(b))}$$

- normalisierte LD mit Bezug auf den längeren der beiden Vergleichstrings zum Zwecke der Vergleichbarkeit
- LSI wird auf das Intervall [0,1] standardisiert
- $0 \leq |\mathrm{LSI}| \leq 1$ mit 0 = keine Ähnlichkeit, 1 = Identität

Medianbildung zwecks Vergleich von Samples

- generalized median M (nach Kohonen (1985), S. 311):

$$\sum_{i=1}^{n} = \mathrm{d}[x(i), M] = \min$$

→ der generalisierte Median ist definiert als diejenige Zeichenfolge, welche die geringste LD zu allen Vergleichsstrings der betreffenden Gruppe aufweist

S := set median, welcher dem Sample angehören muss

M := generalisierter Median, der nicht Teilmenge des untersuchten Samples sein muss

> The set median is found easily, by computing all the mutual distances between the given elements, and searching for that element which has the minimum sum of distances from the other elements.
> The (generalized) median is then found by systematically varying each of the symbol positions of the set median, making ›errors‹ of all the three types over the whole alphabet, and checking whether the sum of distances from the other elements is decreased.[…]
> […] no causal reason exists for the median being equal to the correct string; the median is always computed relative to the error statistics.
> Kohonen (1985), S. 311 f.

- je höher die Fehlerrate der untersuchten Strings, umso robuster sind die Ergebnisse der Mediantests im Vergleich zu anderen Methoden (Kohonen (1985), S. 313)[5]

Laufzeit

> Since the standard LED uses dynamic programming, its time complexity is $O(n^2)$, where n is the maximum of the lenghts of two strings being matched. If there are m entries in a dictionary, the run tims of the LED algorithm is $O(mn^2)$.
> Kulyukin, Vanka, Wang (2013), S. 64.

5 Vgl. für Experimente mit den Medianberechnungen insb Kohonen (1985) und Eriksen et al. (2005).

Beispielwerte:
`LD(bech1.tf, vulgata-u.tf) = 4126969`[6]
Laufzeit unter Lenovo X220 mit i5–2520M @2,5 GHz, 8 GB RAM, SSD Samsung 840 Evo mit 1TB: rd. 13 Minuten
Laufzeit unter Kenko S–860 mit i7–5820K @3,3 GHz, 16 GB RAM, SSD Samsung 850 Pro mit 512 GB: rd. 3 Minuten

Anwendungen für Stringdistanzanwendungen

> In many string-processing applications there is a need to measure the degree of similarity between two strings.
> Stephen (1994), S. 39

> The Levenshtein distance method presents a simple but powerful technique that can be applied to any behaviour produced in a sequence to assess similarity.
> Garland et al. (2012), S. 1437

– Nutzung von erweiterten und normalisierten Levenshtein-Distanzen zur Berechnung von Ähnlichkeiten bei Buckelwalgesängen unterschiedlicher Populationen auf individueller Ebene sowie Populationsebene:

> Here we use a quantitative analysis technique, the Levenshtein distance, to assess similarity in sequences of displays at both the population and individual levels. […] The Levenshtein distance is applicable to all behavioural data produced in sequences ad its use is not limited to acoustical studies.
> Garland et al. 2012, S. 1413.

– Untersuchung von Veränderungen bei männlichen Walgesängen zur Paarungszeit, Veränderung auf Ebene von Populationsgruppen über die Zeit hinweg sowie zwischen unterschiedlichen Populationsgruppen (Eriksen et al. (2005))[7]
– Analyse von Ähnlichkeiten bei Sprachen[8]
– Analye der Ähnlichkeitsgrade von dialektaler Aussprache im Niederländischen (Heeringa (2004))

---

6 `bech1.tf` (14.101 Zeichen) entspricht dem Text »Seelenlos« von Ludwig Bechstein, welcher in den Beispielen von #*SATZ mitgelierfert wird. `bech2.tf` (13.903 Zeichen) unterscheidet sich von `bech1.tf` durch die Umcodierung der Anführungszeichen. Statt `<anf> und </anf>` ist in `bech2.tf` das doppelte Hochkomma (") genutzt worden. `LD(bech1.tf, bech2.tf) = 242. vulgata-u.tf` (4.140.204 Zeichen) ist eine Textfassung der Vulgata ohne jegliche Interpuktion. Lediglich die Versangaben stehen als Tags am Datensatzanfang. Die im Rahmen der Laufzeitangaben genannten Differenzen sind als technische Benchmarks zu verstehen, welche keinerlei Anspruch auf inhaltliche Aussagefähigkeit erheben möchten.

7 Hier Nutzung des Leveshtein Similarity Index sowie generalisierter Mediane in Verbindung mit weiteren statistischen Methoden. Ergebnis: »The Tongan humpback whale song follows the same structure of song and song change as in other parts of the Southern Hemisphere as well as the Northern Hemisphere.« (Eriksen et al. (2005), S. 310, 325). Hierzu auch Tougaard, Eriksen (2005).

8 Zulu, Botha und Barnard (2008) untersuchen die Ähnlichkeitsgrade der 11 offiziellen südafrikanischen Sprachen. Diese Untersuchung soll bspw. bei der Entscheidung helfen, in welche Sprachen etwa Unternehmensdokumente übersetzt werden sollten, um eine möglichst große Zahl an Rezipienten erreichen zu können, falls nicht jede Sprache berücksichtigt werden kann.

- Versionskontrolle unterschiedlicher Dokumente
- approximate string matching (z. B. Suchmaschinen o. ä., Jain, Rao (2013))
- Vorkontrolle für Vergleiche (Ersteinschätzung der Unterschiede)
- Verlinken von Lemmalisten aus unterschiedlichen Wörterbüchern (z. B. mit max. $LD(x, y) = 2$)
- Erstellung von historischen Namensregistern über ein Korpus (s. o.)
- Vorbereitung von Listen zur Kontrolle von Eingabedaten, bei denen lediglich bestimmte Begriffspaare ausgegeben werden (z. B. $LD(x, y) \leq 3$), woraufhin die korrekte Schreibweise festgelegt oder Korrekturen vorgenommen werden können
- Berechnung von Hilfsdateien für Registererstellung:
  Ausgangspunkt: Steuerlisten
  Vorhaben: Erstellung Personenregister
  Idee: quadratische LD des Index Verborum berechnen, um Ähnlichkeiten festzustellen und Namensvarianten schneller zu erschließen, die in einer alphabetisch sortierten Liste weit voneinander entfernt stünden (Iohannes, Johannes)
- OCR-Korrektur, Abgleich der OCR-Ergebnisse mit digitalen Wörterbüchern; Ziel: OCR-Input für Nährstoffkalkulation bei Diabetes via Smartphone (vgl. Kulyukin, Vanka, Wang (2013)[9]

Levenshtein-Implementation in TUSCRIPT

Zur Berechnung der Levenshtein-Distanz stehen in TUSCRIPT zwei neue Funktionen zur Verfügung: `DISTANCE(a,b)` für die Berechnung der einfachen Distanz ohne Berücksichtigung von Unterschieden in der Groß- und Kleinschreibung sowie `DISTANCE_EXACT` für die Berechnung der Distanz unter Berücksichtigung von Unterschieden in der Groß- und Kleinschreibung. Letztere werden bei `DISTANCE_EXACT` wie Hinzufügen, Löschen oder Substituieren als Erhöhung um 1 gewertet.

Vergleich einfacher Strings

Die im Workshop-Rahmen verwendete Segmentdatei `lev_segm` mit Codebeispielen kann wie folgt verwendet werden:

- Anmelden der Datei im betreffenden Projekt (`#AN,lev_segm`)
- Definition der Segmentdatei als Makrodatei (`#DE,1:lev_segm`)
- Aufruf des gewünschten Segments mitsamt den obligatorischen Parametern, z. B.:

```
$einf_lev,baum,haus  oder
$lsi_ber,bech1.tf,bech2.tf,prot_bech.rtf
```

9 Die Autoren testen drei Verfahren für die Korrektur der OCR-Ergebnisse: n-gram-Abgleiche, die Nutzung der Levenshtein-Distanz sowie ihr eigens entwickeltes Verfahren »Skip Trie Matching«. Letzteres weist für den skizzierten Zweck Vorteile hinsichtlich der Laufzeit sowie der Ergebnisse auf.

Für die folgenden Code-Beispiele die Konventionen der Zeilen 01–03.

```
01 $$!
02 $$ MODE TUSCRIPT, {}
03 Ausführung z.B. mit dem Kommando $?$,scriptdatei
```

```
string1 = "Haus"
string2 = "Baum"
lev = DISTANCE (string1, string2)
→ lev = 2
```

```
string1 = "Wein"
string1 = "Bier"
lev = DISTANCE (string1, string2)
→ lev = 4
```

Unterscheidung von Groß- und Kleinschreibung mit d = d + 1

```
string1 = "Haus"
string2 = "haus"
lev = DISTANCE (string, string2)
→ lev = 0
```

```
string1 = "Haus"
string2 = "haus"
lev_ex = DISTANCE_EXACT (string, string2)
→ lev_ex = 1
```

Umgang mit Umlauten, Diakritika und »Sonderzeichen« (ß, đ…)

- TUSTEP codiert Buchstaben mit Diakritika intern in Form mehrerer Zeichen; z. B.:
  é = %/e,
  d̲̊ = %−−#;o#.d
- Vergleich von Strings ohne Vorbereitung (data pre-processing) kann zu unerwünschten resp. unerwarteten Ergebnissen führen

Optionen zum pre-processing in Abhängigkeit von Forschungsfrage, Vorlage, Sprache (…)

- Buchstaben mit Diakritika auf Grundbuchstaben zurückführen (z.B »d« statt »#.d«)
- mehrere Diakritika auf eine einheitliche Codierung vereinheitlichen zur Bewahrung der rechnerischen Distanz zum Grundbuchstaben (z. B. »$a« statt »%−%−−a«)
- Auswertung sämtlicher TUSTEP-internen Codes

Literatur

CHEN, Yoke Yie / YONG, Suet-Peng / ISHAK, Adzlan, Email Hoax Detection System Using Levenshtein Distance Method, in: Journal of Computers, Jg. 9, Nr. 2/2014, S. 441–446.

ERIKSEN, Nina / MILLER, Lee A. / Tougaard, Jakob et al., Cultural Change in the Songs of Humpback Whales (*Megaptera novaeangliae*) from Tonga, in: Behaviour, Jg. 142, Nr. 3/2005, S. 305–328.

GARLAND, Ellen C. / Lilley, Matthew S. et al., Improved Version of the Levenshtein Distance Method for Comparing Sequence Information in Animals' Vocalisations. Tests Using Humpback Whale Song, in: Behaviour, Jg. 149, Nr. 13/14/2012, S. 1413–1441.

GOLIĆ, Jovan Dj. / MIHALJEVIĆ, Miodrag J., A Generalized Correlation Attack on a Class of Stream Ciphers Based on the Levenshtein Distance, in: Journal of Cryptology Nr. 3/1991, S. 201–212.

HEERINGA, Wilbert, Measuring Dialect Pronunciation Differences using Levenshtein Distance, (Groningen Dissertations in Linguistics, 46), Gronigen 2004.

HELWEG, David A. / CATO, Douglas H. / Jenkins, Peter F. et al., Geographic Variation in South Pacific Humpback Whale Songs, in: Behaviour, Jg. 135, Nr. 1/1998, S. 1–27.

HICHAM, Gueddah / ABDALLAH, Yousfi / MOSTAPHA, Belkasmi, Introduction of the Weight Edition Errors in the Levenshtein Distance, in: International Journal of Advanced Research in Artificial Intelligence, Jg. 1, Nr. 5/2012, S. 30–32.

INTURI, Anitha Rani / RAMADEVI, Jujjuri, An Unsupervised Approach for Mining Multiple Web Databases, in: International Journal of Electronics and Computer Science Engineering, Jg. 1, Nr. 4/2012, S. 2148–2151.

JAIN, Shivani / RAO, A.L.N., A Comparative Performance Analysis of Approximate String Matching, in: International Journal of Innovative Technology and Exploring Engineering, Jg. 3, Nr. 5/2013, S. 123–128.

JINAN, Fiaidhi / SABAH, Mohammed / AMINUL, Islam, Towards Identifying Personalized Twitter Trending Topics Using the Twitter Client RSS Feeds, in: Journal of Emerging Technologies in Web Intelligence, Jg. 4, Nr. 3/2012, S. 221–226.

KOHONEN, Teuvo, Median Strings, in: Pattern Recognition Letters, Jg. 3, Nr. 5/1985, S. 309–313.

KULYUKIN, Vladimir / VANKA, Aditya / WANG, Haitao, Skip Trie Matching. A Greedy Algorithm for Real-Time OCR Error Correction on Smartphones, in: International Journal of Digital Information and Wireless Communication, Jg. 3, Nr. 3/2013, S. 56–65.

LEVENSHTEIN, V. I., Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, Orig. russ. in: Doklady Akademii Nauk SSSR, Jg. 163, Nr. 4/1965, S. 845–848, in: Cybernetics and Control Theory, Jg. 10, Nr. 8/1966, S. 701–710.

LI, Yujian / BO, Liu, A Normalized Levenshtein Distance Metric, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, Jg. 29, Nr. 6/2007, S. 1091–1095.

MARGOLIASH, Daniel / STAICER, Cynthia A. / INOUE, Sue A., Stereotyped and Plastic Song in Adult Indigo Buntings, *Passerina Cynea, in:* Animal Behaviour, Nr. 42/1991, S. 367–388.

MARGOLIASH, Daniel / STAICER, Cynthia A. / INOUE, Sue A., The Process of Syllable Acquisition in Adult Indigo Buntings (Passerina Cyanea), in: Behaviour, Jg. 131, Nr. 1/2/1994, S. 39–64.

PETRONI, Filippo / SERVA, Maurizio, Measures of Lexical Distance between Languages, in: Physica A, Jg. 389, Nr. 11/2010, S. 2280–2283.

STEPHEN, Graham A., String Searching Algorithms, (Lecture Notes Series on Computing, 6), Singapur 1994.

TOUGAARD, Jakob / ERIKSEN, Nina, Analysing Differences among Animal Songs Quantitatively by Means of the Levenshtein Distance Measure, in: Behaviour, Jg. 143, Nr. 2/2006, S. 239–252.

ZULU, P. N. / BOTHA, G. / BARNARD, E., Orthographic Measures of Language Distances Between the Official South African Languages, in: Literator, Jg. 29, Nr. 1/2008, S. 185–204.

		b	**a**	**u**	**m**
	[0]	1	2	3	4
h	1	[1]	2	3	4
a	2	2	[1]	2	3
u	3	3	2	[1]	2
s	4	4	3	2	[2]

h a u s
b a u m

		u	n	s	p	o	r	t	l	i	c	h
	0	1	2	3	4	5	6	7	8	9	10	11
s	1	1	2	2	3	4	5	6	7	8	9	10
p	2	2	2	3	2	3	4	5	6	7	8	9
o	3	3	3	3	3	2	3	4	5	6	7	8
t	4	4	4	4	4	3	3	3	4	5	6	7
t	5	5	5	5	5	4	4	3	4	5	6	7
e	6	6	6	6	6	5	5	4	4	5	6	7
n	7	7	6	7	7	6	6	5	5	5	6	7

```
  spo t ten
unsportlich

  spo tt en
unsportlich

  spott  en
unsportlich

  spo tte n
unsportlich

  spott e n
unsportlich

  spotte  n
unsportlich

  spo tten
unsportlich

  spott en
unsportlich

  spotte n
unsportlich

  spotten
unsportlich
```

http://www.let.rug.nl/kleiweg/lev/
(07.01.2015)

Ergebnisse Ähnlichkeitsberechnung (Skript LSI_BER) bech1.tf~bech2.tf:

Zeilen bech1.tf = 58
Zeilen bech2.tf = 58

Zeichenanzahl bech1.tf = 14101
Zeichenanzahl bech2.tf = 13903

einfache Levenshtein-Distanz:
LD(bech1.tf, bech2.tf) = 242

exakte Levenshtein-Distanz:[1]
LD_ex(bech1.tf, bech2.tf) = 242

Levenshtein Similarity Index:[2]
LSI(bech1.tf, bech2.tf) = 0.98284

[1] Bei der »exakten« Berechnung wird unterschiedliche Groß- und Kleinschreibung mit einem Unterschied von 1 gewertet.

[2] Formel: $LSI = 1 - \frac{LD}{MAX(LEN(bech1.tf),LEN(bech2.tf))}$

Der Index variiert zwischen 0 und 1 mit 0 als Indikator für keinerlei Übereinstimmung und 1 als Zeichen für Identität.

Ergebnisse Ähnlichkeitsberechnung spotten~unsportlich:

Zeichenanzahl spotten = 7
Zeichenanzahl unsportlich = 11

einfache Levenshtein-Distanz:
LD(spotten, unsportlich) = 7

Levenshtein Similarity Index:[1]
LSI(spotten, unsportlich) = 0.36364

[1] Formel: $LSI = 1 - \frac{LD}{MAX(LEN(spotten),LEN(unsportlich))}$

Der Index variiert zwischen 0 und 1 mit 0 als Indikator für keinerlei Übereinstimmung und 1 als Zeichen für Identität.