

Zusammenfassung zur Vorlesung Basismodul Computerlinguistik

Approximatives Matching & Levenshtein-Automaten

23.12.2021

1 Levenshtein-Automaten

- Approximatives Matching
- Levenshtein-Abstand
- Nicht-deterministische Levenshtein-Automaten

2 Aufgabenstellung des Abschlussprojekts

Lookup → Approximatives Matching

- Anhand eines **Lexikonautomaten** wird das genaue Suchen durchgeführt. Es ist zu ermitteln, ob ein gegebenes Wort in der Sprache des Automaten existiert.
- In der Praxis wird jedoch Approximatives Matching häufiger angefordert.
- Für ein nicht im Lexikon existierendes Wort sollen möglichst **passende Korrekturkandidaten** gefunden werden.
→ nach **Levenshtein-Abstand**

- Rechtschreibkorrektur bei Texteditoren
- Korrektur von Suchanfragen, Vorschlagssuche
- Ermittlung von Korrekturkandidaten für fehlerhaft digitalisierte Wörter (OCR)

Gegeben ein **Pattern** P , ein **Lexikon** D und ein kleiner **Grenzwert** k , soll auf effiziente Weise die Menge aller **Einträge** W in D ermittelt werden, sodass der Levenshtein-Abstand zwischen P und W nicht größer als k ist.

Definition:

Der (Standard-) **Levenshtein-Abstand** zwischen zwei Wörtern P und W , geschrieben $d_L(P, W)$, ist die minimale Anzahl primitiver Editieroperationen die notwendig sind, um P in W umzuwandeln.

- Ersetzungen
- Löschungen
- Einfügungen

Berechnung mittels **dynamischer Programmierung**

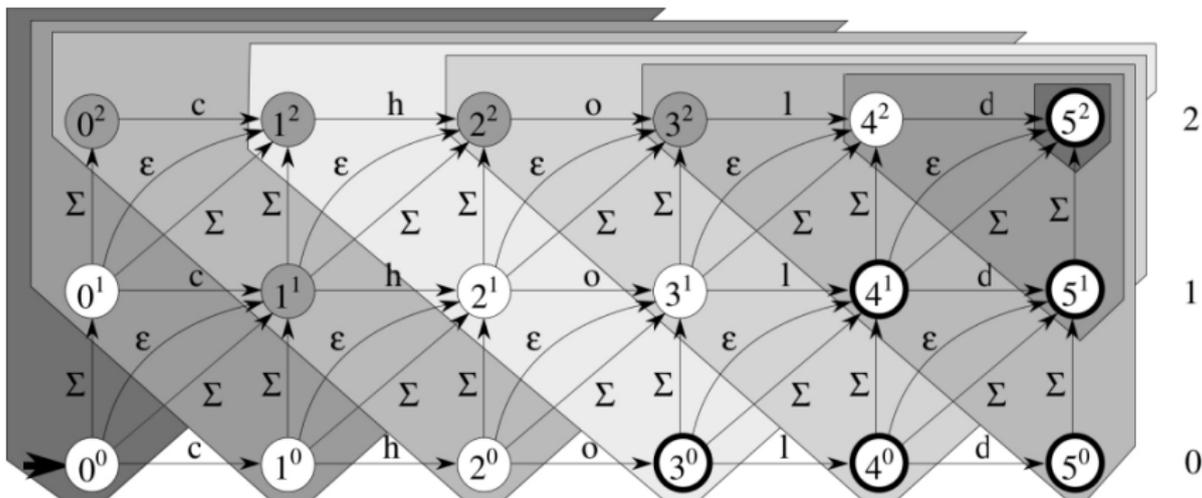
- $d_L(\epsilon, W) = |W|$
- $d_L(P, \epsilon) = |P|$
- $d_L(Pa, Wb) = \begin{cases} d_L(P, W) & \text{falls } a = b \\ 1 + \min(d_L(P, W), d_L(Pa, W), d_L(P, Wb)) & \text{falls } a \neq b \end{cases}$

Nicht-deterministische Levenshtein-Automatene

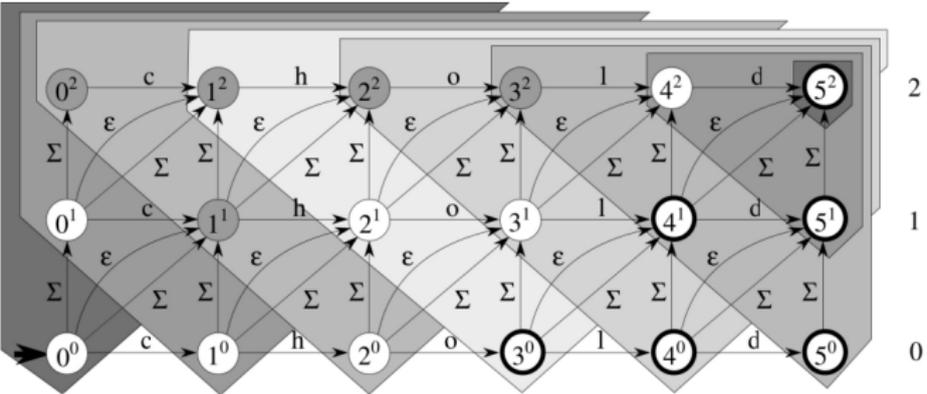
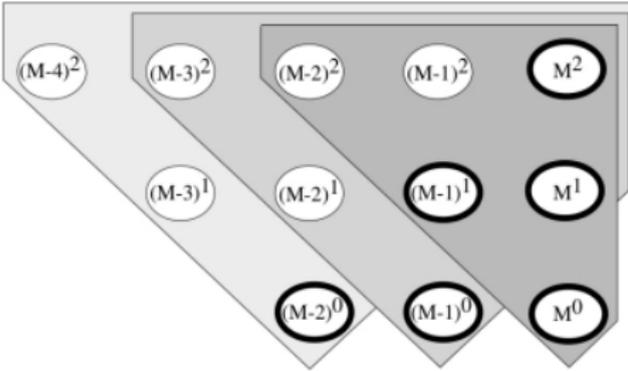
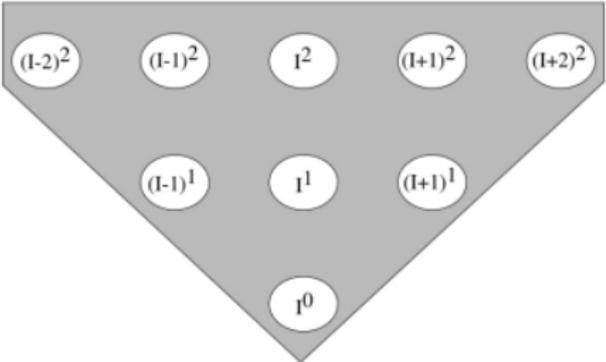
Grundidee

Für jedes **Pattern** P wird ein nicht-deterministischer **Stockwerksautomat** $A_n(P)$ aufgebaut, der als Sprache alle Wörter W mit Levenshtein-Abstand $d_L(P, W) \leq n$ hat.

Z.B. Stockwerkautomat $A_2(chold)$ für das Pattern $chold$ und maximalen Editierabstand 2



Dreieckbereich



Vorgehensweise bei Übergängen

- 1 Aktualisieren der aktiven Zustände
- 2 Eliminieren aller überflüssigen Zustände

→ **bis:**

- Eingabewort komplett eingelesen wird **oder**
- es keinen passenden Übergang mehr gibt

Definition

Der charakteristische Vektor $\vec{\chi}(w, V)$ für ein Symbol $w \in \Sigma$ in einem Wort $V = v_1 \cdot v_n \in \Sigma^*$ ist der **Bitvektor** der Länge n , bei dem das i -te Bit auf 1 gesetzt ist, genau dann wenn $w = v_i$, sonst auf 0.

- Um die Länge der charakteristischen Vektoren zu standardisieren, wird an den Beginn des Patterns P eine Folge von n speziellen Symbolen '\$' angefügt.
- Die **Länge** des Bitvektoren ist $2n+2$ für $0 \leq i \leq |P| - n - 1$. Für alle späteren Positionen nehmen die Vektoren kontinuierlich ab.

Der Link zur Aufgabenstellung des Abschlussprojekts:
`https://www.cip.ifi.lmu.de/~nie/p1/
Abschlussprojekt_de.html`