6. Übung zur Vorlesung “Internet-Suchmaschinen” im Sommersemester 2009

Prof. Dr. Gerd Stumme, Wi.-Inf. Beate Krause

01. Juli 2009

1 String-Metriken

1. Berechnen Sie die Levenstein-Metrik $lev($“carsten”, “christina”$)$. Geben Sie das Berechnungsschema an, und nennen Sie die einzelnen Umformungsschritte einer kürzesten Umformung in der Art: test $\rightarrow$ tost $\rightarrow$ toast

   Lösungsvorschlag:

		C	H	R	I	S	T	I	N	A
	0	1	2	3	4	5	6	7	8	9
C	1	0	1	2	3	4	5	6	7	8
A	2	1	1	2	3	4	5	6	7	7
R	3	2	2	1	2	3	4	5	6	7
S	4	3	3	2	2	2	3	4	5	6
T	5	4	4	3	3	3	2	3	4	5
E	6	5	5	4	4	4	3	3	4	5
N	7	6	6	5	5	5	4	4	3	4

   Eine kürzeste Umformung wäre also

   carsten $\rightarrow$ chrsten $\rightarrow$ christen $\rightarrow$ christin $\rightarrow$ christina

2. Modifizieren Sie den Algorithmus auf Folie 60, so dass er die Editier-Distanz berechnet.

Lösungsvorschlag:

Im Unterschied zur Levenshtein-Metrik sind bei der Editier-Distanz keine Austauschoperationen möglich, sondern nur Lösch- sowie Einfügeoperationen. Damit ergeben sich für die Belegung der $LEV[i,j]$ die folgende Möglichkeiten: Es sei

$$\delta(i,j) := \begin{cases} 0 & \text{für } s[i] = t[j], \\ \infty & \text{für } s[i] \neq t[j]. \end{cases}$$

Dann gilt:

$$EDIT[i,j] := min \begin{cases} EDIT[i-1,j]+1 & \text{(Löschung)} \\ EDIT[i,j-1]+1 & \text{(Einfügung)} \\ EDIT[i,j-1]+\delta(i,j) & \text{(gleicher Buchstabe)} \end{cases}$$

3. Modifizieren Sie den Algorithmus auf Folie 60, so dass er die Damerau-Levenstein-Distanz berechnet.

   Lösungsvorschlag:

   Für die Damerau-Levenstein-Distanz wird eine Distanzfunktion benötigt, die angibt, ob der $i$te Buchstabe des Wortes s mit dem $j$ten Buchstaben des Wortes t übereinstimmt oder nicht:

   $$d(s_i,t_j) := \begin{cases} 0 & \text{für } s_i = t_j, \\ \infty & \text{für } s_i \neq t_j. \end{cases}$$

   Die Damerau-Levenstein-Metrik ist dann definiert durch:

   $$DLEV[0,0] := 0$$

   $$DLEV[i,j] := min \begin{cases} DLEV[i-1,j]+1 \\ DLEV[i,j-1]+1 \\ DLEV[i-1,j-1]+d(s_i,t_j) \\ DLEV[i-2,j-2]+d(s_{i-1},t_j)+d(s_i,t_{j-1})+1 & \text{(für } i \geq 2\text{)} \end{cases}$$

4. Berechnen Sie die Editier- und die Damerau-Levenstein-Distanz von cbabac zu abcabbbaa und vergleichen Sie sie mit der Levenstein-Distanz.

   Lösungsvorschlag:

   Es gilt $dlev(\text{“}cbabac\text{”}, \text{“}abcabbbaa\text{”}) = lev(\text{“}cbabac\text{”}, \text{“}abcabbbaa\text{”}) = 5$ und $edit(\text{“}cbabac\text{”}, \text{“}abcabbbaa\text{”}) = 8$.

2 Soundex

1. Vergleichen Sie die folgenden Wörter mit Soundex!
   - through, thru, trough Lösungsvorschlag:

     T62, T6, T62
   - Mr, Mayer, Meier Lösungsvorschlag:

     M6, M6, M6
   - Smith, Schmidt, Schmitz Lösungsvorschlag:

     S53, S253, S253
   - data, date, dito Lösungsvorschlag:

     D3, D3, D3
2. Sehen Sie Probleme und Verbesserungsmöglichkeiten?

   Lösungsvorschlag:

   Soundex ist hier manchmal zu grob (etwa bei Mr/Mayer, through/trough), andererseits auch zu feinfühlig, etwa bei dem stummen “g” in through/thru.

   Zudem sind einige der Regeln sprachabhängig. So ergibt es im Englischen einen Sinn, *h* und *y* wie Vokale zu behandeln, im Deutschen weniger.
3. Schlagen Sie vor, wie Sie die Probleme in Soundex beheben können.

   Lösungsvorschlag:

   Eine größere Menge von Regeln und Ausnahmen helfen, solche sprachlichen Feinheiten zu berücksichtigen.

   Phonix z. B. beinhaltet über 100 Regeln, die Fälle wie *ough* $\rightarrow$ *ow, kn* $\rightarrow$ *n, chr* $\rightarrow$ *kr* abdecken (Gadd T., PHONIX: The Algorithm, Program, 24(4), p381-402, 1990).
4. Welche neuen Nachteile bringen Ihre Verbesserungen mit sich?

   Lösungsvorschlag:

   Neben der größeren Komplexität erhöht solch ein umfangreiches Regelwerk vor allem die Sprachabhängigkeit. Phonix in der vorliegenden Form ist z. B. nur für die englische Sprache zu gebrauchen.

3 XML/XPath

1. Betrachten Sie das XML-Dokument auf Seite 27 des Kapitels “Strukturelle Anfragen” (`<library ... >`).

   Geben Sie XPath-Ausdrücke, die folgende Teile des Dokuments auswählen:

   a) alle Bücher

      Lösungsvorschlag:

      /library/author/book
      //book

   b) alle Bücher von William Smart

      Lösungsvorschlag:

      /library/author[@name=“William Smart”]/book
      //author[@name=“William Smart”/book

2. Funktionieren die folgenden Ausdrücke auf dem Dokument auf Seite 19 genau so?

   a) alle Personen

      Lösungsvorschlag:

      //Person geht, nicht aber /db/Person o.ä., da hier Personen an verschieden Stellen im Baum vorkommen.

   b) alle Personen, die Robert heißen. *Tip:* Den Textinhalt von Kindelementen kann man prüfen, indem man einen relativen Pfad statt `@attribut` in die eckige Klammer schreibt!

      Lösungsvorschlag:

      //Person[Name/Vorname = ’Robert’]

4 Eigenschaften von Texten/Power Laws

1. Begründen Sie die Aussage von Luhn auf Seite 6 des Kapitels: Warum sind besonders häufige und besonders seltene Wörter nicht sehr nützlich?

   Lösungsvorschlag:

   Besonders häufige Wörter kommen in fast jedem Dokument vor und sind somit nicht geeignet, relevante von nicht relevanten Dokumenten zu unterscheiden.

Besonders seltene Wörter dagegen sind unter Umständen so selten, dass sie auch praktisch nie angefragt werden. Dadurch ergeben sie ebenfalls keinen Nutzen beim Information Retrieval, auch wenn sie für sehr spezifische Anfragen perfekte Resultate liefern können (100% Precision, 100% Recall).

2. Auch die Grade von Webseiten sind nach einem Potenzgesetz (*power law*) verteilt. Die folgende Tabelle gibt einige Ingrade einer Menge von Webseiten zu einem Zeitpunkt im Jahr 1999 wieder:

Grad	Anzahl Seiten
1	63100000
10	501200
100	4000
1000	32
5000	1

Bestimmen Sie den Koeffizienten c im Potenzgesetz!

Lösungsvorschlag:

Grad	Anzahl Seiten	Log Grad	Log Anz	Diff
1	63100000	0	7,8	
10	501200	1	5,7	-2,1
100	4000	2	3,6	-2,1
1000	32	3	1,51	-2,1
5000	1	3,7	0	-2,15

Die Steigung im Log-Log-Plot – also der Exponent c im Potenzgesetz – ist hier also etwa -2.1.

5 Praxisaufgabe (Abgabe: 14.07.09)

Sie haben bereits einen invertierten Index auf Dokumenten, verschiedene Anfragemöglichkeiten und einen Spider implementiert.

Bauen Sie diese Komponenten zu einer Suchmaschine zusammen! Dazu soll Ihre Web-Schnittstelle so weiter entwickelt werden, dass die folgende Funktionalität angeboten werden kann:

- Eingabe von zu crawlenden Basis-URLs und einer Schranke (Anzahl Seiten, Crawltiefe o.ä.). Diese Seiten sollen aus dem Web geholt und zum Index hinzugefügt werden.
- Ranking auf Basis von tf-idf Gewichten als Antwort für eine Suchanfrage
- Anzeige der gerankten Dokumente (Links), wenn möglich des Kontexts