

## 6. Aufgabenblatt zum Softwarepraktikum “Muster- und Bilderkennung” im Grundstudium

---

Alle für die Bearbeitung der Aufgaben benötigten Dateien finden Sie in folgendem Archiv:

<http://www-i6.informatik.rwth-aachen.de/web/Teaching/LabCourses/▶SS06/Softwarepraktikum/Aufgabe6/lm-ambikeys.tgz>

### Aufgabe 1:

Erstellen Sie eine Klasse, die es ermöglicht, ein Sprachmodell im ARPA-Format einzulesen und für gegebene Wortfolgen (Uni- und Bigramme) Wahrscheinlichkeiten nach folgendem Schema berechnet:

$$P(w_i|w_{i-1}) = \begin{cases} P(w_i|w_{i-1}) & \text{für vorhandenen Eintrag } w_{i-1} \ w_i \\ \alpha(w_{i-1}) \cdot P(w_i) & \text{sonst} \end{cases}$$

ARPA-Format:

```
\data\  
ngram 1=6998  
ngram 2=33978  
  
\1-grams:  
-1.777618      a      -0.3105851  
-4.038711      a.m.   -0.4448084  
-4.184839      abandon -0.1696519  
...  
  
\2-grams:  
-2.799451      a chair 0.1895401  
-3.089627      a chance  
...
```

Format für die `\N-grams`:-Einträge ( $N = 1, 2, \dots$ ):

1. Logarithmus (Basis 10) der bedingten Wahrscheinlichkeit für  $P(w_i|w_{i-n+1}^{i-1})$
2.  $n$ -Gramm  $w_{i-n+1}, \dots, w_{i-1}, w_i$
3. *optional*: Backoff-Gewicht  $\alpha(w_{i-n+1}^i)$  des  $n$ -Gramms (Logarithmus Basis 10)

## Aufgabe 2:

Implementieren Sie eine mehrdeutige Tastatur mit folgenden Eigenschaften:

- Einlesen einer Definitionsdatei ( $\rightsquigarrow$  `t9.key`), die pro Zeile die jeweiligen Buchstaben einer Taste enthält (Codierung der mehrdeutigen Tastatur)
- Laden eines Lexikons ( $\rightsquigarrow$  `80days.tok.dic`) und Erstellen der Datenstruktur, die zu jedem Code (Folge von Tastendrücken) die möglichen exakten Wortfolgen und Vervollständigungen effizient ermittelt (als Datenstruktur eignet sich der in Aufgabe 3 implementierte Präfix-Baum)

Beispiel:

	exact matches	completions
code 7325	<i>real</i>	<i>really</i>
	<i>peak</i>	<i>realise</i>
	<i>seal</i>	<i>reality</i>
		<i>reckon</i>
		<i>reckoning</i>
		<i>secluded</i>

- Methoden, die die möglichen Wortfolgen eines Codes und den schon disambiguierten Kontext (hier: das Vorgängerwort) anhand von Wahrscheinlichkeiten des in Aufgabe 1 implementierten Sprachmodells neu bewerten

## Aufgabe 3:

Testen Sie Ihre Implementierung, indem Sie die Sätze aus `80days.tok.test` mehrdeutig von einem Evaluationsmodul tippen lassen. Verwenden Sie hierbei das Sprachmodell in `80days.tok.lm`. Benutzen Sie als Bewertungsmaß

$$\text{KSPC} = \frac{\sum_w (K_w + P_w + 1)}{\sum_w N_w},$$

wobei  $K_w$  die Länge des getippten Codes des Wortes  $w$ ,  $P_w$  dessen Position in der Vorschlagsliste und  $N_w$  die Wortlänge ist (KSPC = keystrokes per character).

Untersuchen Sie, welche KSPC man mit exakten Vorschlagslisten und Vorschlagslisten mit Vervollständigung sowohl für Uni- als auch Bigramme erreicht. Hierbei soll das Tippen eines Wortes abgebrochen werden, sobald das Wort als beste Hypothese in der Vorschlagsliste auftaucht ( $P_w = 1, K_w < N_w$ ). Wenn das ganze Wort eingegeben werden musste, muss es anschließend in der Liste lokalisiert werden ( $P_w \geq 1$ ).

## Abgabe:

Abnahmetermin: **Donnerstag, 06. Juli**

Abnahmeort: **CIP-Pool**

Bitte senden Sie den Quellcode der Lösungen **spätestens einen Tag vor Abnahme** per e-mail an die Betreuer **{hasan,matusov}@i6.informatik.rwth-aachen.de**. Achten Sie auch darauf, dass Ihre Programme unter Linux kompilierbar sind, bei keiner Eingabe abstürzen, jeden Bedienungsfehler mit einer sinnvollen Fehlermeldung quittieren und beim Aufruf mindestens eine kurze Hilfe ausgeben!