Schriftliche Prüfung zur Übung
Statistische Methoden in der maschinellen Sprachverarbeitung
WS 2015/16
Dozent: Helmut Schmid

Aufgabe

Sie sollen den **Forward-Backward-Algorithmus** für einen Bigramm-Tagger implementieren. Die Verarbeitung erfolgt wie immer satzweise.

Definition der Forward-Wahrscheinlichkeiten $\alpha_t(k)$

$$\begin{aligned} \alpha_t(0) &= \begin{cases} 1 & \text{falls } t = \langle s \rangle \\ 0 & \text{sonst} \end{cases} \\ \alpha_t(k) &= \sum_{t' \in T} \alpha_{t'}(k-1)\; p(t|t')\; p(w_k|t) \qquad \text{für } 1 \leq k \leq n+1 \end{aligned}$$

t, t' sind hier Tags, k ist eine Wortposition, $\langle s \rangle$ ist das Satzgrenzen-Tag. $w_1, ..., w_n$ sind die Wörter des Satzes. w_0 und w_{n+1} sind Satzgrenzentags.

Sie können den Forward-Algorithmus fast wie den Viterbi-Algorithmus implementieren. Die Maximierung des Viterbi-Algorithmus muss lediglich durch eine Summe ersetzt werden, und es werden keine Rückwärtszeiger gespeichert.

Verwenden Sie z.B. ein Array von Hash-Tabellen *forward[k]{tag}*, um die Forward-Wahrscheinlichkeiten zu speichern. Folgende Datenstrukturen und Funktionen sind gegeben und müssen nicht implementiert werden:

- n: eine Variable mit der Länge des Eingabesatzes
- *word[1...n+1]*: ein Array/Liste, in dem die Wörter gespeichert sind. Das erste Wort hat den Index 1, das letzte den Index n.

  $word[n+1] = \epsilon$ ist ein spezielles Satzende-Token, für das die Funktion *lexprob* die korrekten Wahrscheinlichkeiten liefert (also 1 falls das Tag $\langle s \rangle$ ist, 0 sonst).
- *tagset*: ein Array/Liste, in dem die Menge der möglichen Tags gespeichert ist.
- *lexprob(t,w)*: Funktion, die $p(w|t)$ zurückliefert
- *contextprob(t,t')*: Funktion, die $p(t'|t)$ zurückliefert

Berechnen Sie Forward-Wahrscheinlichkeiten für alle möglichen Tags. Das heißt, iterieren Sie jeweils über alle Tags in *tagset* (statt über eine Menge möglicher Tags des Wortes).

Tipp: Sie können drei ineinander geschachtelte Schleifen verwenden, in denen Sie über alle Positionen $k = 1...n+1$, alle aktuellen Tags $t \in tagset$ und alle vorherigen Tags $t' \in tagset$ iterieren.

$$
\begin{aligned}
\beta_t(n+1) &= \begin{cases} 1 & \text{falls } t = \langle s \rangle \\ 0 & \text{sonst} \end{cases} \\
\beta_t(k) &= \sum_{t' \in T} p(t'|t)\, p(w_{k+1}|t')\, \beta_{t'}(k+1) \qquad \text{für } 0 \leq k \leq n
\end{aligned}
$$

und die Aposteriori-Wahrscheinlichkeiten

$$
\gamma_t(k) \;=\; \frac{\alpha_t(k)\; \beta_t(k)}{\alpha_{\langle s \rangle}(n+1)} \qquad \text{für } 1 \leq k \leq n
$$

Dann extrahieren Sie die besten Tags

$$
t_k \;=\; \arg\max_t \gamma_t(k) \qquad \text{für } 1 \leq k \leq n
$$

und geben sie aus.

Sie verwenden den Forward-Backward-Algorithmus hier also zum Taggen.[1]

Die Berechnung der Backward-Wahrscheinlichkeiten geht ähnlich wie die Berechnung der Forward-Wahrscheinlichkeiten, aber Sie iterieren in umgekehrter Reihenfolge n...0.

Am besten schreiben Sie vier Funktionen – forward, backward, posterior und printtags – für die vier Schritte.

[1]Dies ist ein alternativer Tagging-Algorithmus, der den Vorteil hat, dass zu jedem Tag die Aposteriori-Wahrscheinlichkeit als Maß der Zuverlässigkeit ausgegeben werden kann.