Schriftliche Wiederholungsprüfung zur Übung
Statistische Methoden in der maschinellen Sprachverarbeitung
SS 2016
Dozent: Helmut Schmid

Die Prüfung besteht aus drei Teilaufgaben mit jeweils 10 Punkten.

Inhaltliche Wiederholung aus der Vorlesung

Naive Bayes-Modelle können zur Desambiguierung von Wortbedeutungen auf Basis der Kontextwörter verwendet werden. Die beste Bedeutung $\hat{s}$ eines Wortes w im Kontext der Wörter $context(w)$ wird nach der folgenden Formel berechnet:

$$\hat{s} = \arg \max_{s \in senses(w)} p(s|w) \prod_{c \in context(w)} p(c|s) \tag{1}$$

$senses(w)$: Menge der Bedeutungen des Wort-Tokens w
$context(w)$: Liste der Kontext-Wörter

Anmerkung: Dieses Modell soll nicht nur ein ambiges Wort w sondern mehrere ambige Wörter desambiguieren.

Aufgabe 1: Naive-Bayes-Parameterschätzung

Schreiben Sie ein Programm, welches die Wahrscheinlichkeiten $p(s|w)$ und $p(c|s)$ aus Trainingsdaten schätzt.

Es existiert bereits eine Funktion *extract_data*, welche das Trainingskorpus einliest und als eine Liste von Listen zurückliefert. Ihr Programm beginnt daher mit der Zeile:

```
my @data = extract_data();
```

`@data` enthält eine Liste von Wortvorkommen mit der korrekten Bedeutung und den Kontextwörtern.

`$w = $data[1][0]` liefert Ihnen das 2. ambige Wort (z.B. "Bank"), das aus dem Korpus extrahiert wurde
`$s = $data[3][1]` liefert Ihnen die Bedeutung des 4. ambigen Wortes des Korpus (z.B. "Sitzbank")
`$context = $data[3][2]` liefert Ihnen eine Referenz auf eine Liste mit den Wörtern im Kontext des 4. ambigen Wortes des Korpus (z.B. die Liste "(sitzen, Mann, vor, Baum)")

Sie sollen die folgenden Wahrscheinlichkeiten schätzen:

- die Wahrscheinlichkeit der Bedeutung s gegeben das Wort w gemäß der Formel

$$p(s|w) = \frac{freq(w,s)}{\sum_{s'} freq(w,s')} \tag{2}$$

- die Wahrscheinlichkeit des Kontextwortes c gegeben die Wortbedeutung s gemäß der Formel

$$p(c|s) = \frac{freq(s,c)}{\sum_{c'} freq(s,c')} \quad (3)$$

Schritte:

- Berechnen Sie, wie oft das ambige Wort w mit der Bedeutung s auftaucht (= $freq(w,s)$)
- Berechnen Sie, wie oft das Kontextwort c bei der Bedeutung s auftaucht (= $freq(s,c)$)
- Schätzen Sie $p(s|w)$ nach der Formel (2)
- Schätzen Sie $p(c|s)$ nach der Formel (3)

(6 Punkte)

Aufgabe 2: Naive-Bayes-Anwendung

Schreiben Sie nun ein Programm, welches das Naive Bayes-Modell aus einer Datei liest und auf Daten anwendet.

Ihr Programm beginnt mit

```
my(%wsprob,%scprob);
read_params(\%wsprob,\%scprob);
my @data = extract_data();
```

Danach sind die Parameter und die Eingabedaten eingelesen. (Die Funktionen `read_params` und `extract_data` müssen Sie nicht programmieren.)

`$wsprob{Schwester}{SCHWESTER1}` enthält die Wahrscheinlichkeit der Bedeutung ,,SCHWESTER1" des Wortes ,,Schwester" und entspricht $p(s|w)$

`$scprob{SCHWESTER1}{Krankenhaus}` enthält die Wahrscheinlichkeit des Wortes ,,Krankenhaus" im Kontext der Bedeutung ,,SCHWESTER1" und entspricht $\Rightarrow p(c|s)$

`@data` enthält die Eingabedaten, wobei dieses Mal die Bedeutung (z.B. `$data[3][1]`) undefiniert ist, da Sie diese ja berechnen sollen.

Schritte:

- Iterieren Sie über die Liste der Eingabedaten `@data`
- Berechnen Sie die wahrscheinlichste Bedeutung s aller zu desambiguierenden Wörter $data[i][0]$ nach Formel (1) auf dem vorigen Blatt und speichern sie s in $data[i][1]$ (wobei i der Index des aktuellen zu desambiguierenden Wortes ist).

  Tipp: Die Menge der möglichen Bedeutungen des Wortes ,,Schwester" liefert der Befehl: `keys %{$wsprob{Schwester}}`

(6 Punkte)

Aufgabe 3: Perzeptron-Tagger

Nun sollen Sie noch den Viterbi-Algorithmus für den Perzeptron-Tagger implementieren. Er ist bis auf die Berechnung der Viterbi-Werte mit dem Viterbi-Algorithmus für HMMs identisch. Hier sind die Formeln für die Berechnung:

$$\begin{aligned} \delta_{\langle s \rangle}(0) &= 0 \\ \delta_t(0) &= -\inf \text{ falls } t \neq \langle s \rangle \\ \delta_t(k) &= \max_{t'} \delta_{t'}(k-1) + w \cdot f(w_{k-1}, w_k, w_{k+1}, t', t) \end{aligned}$$

"inf" steht für unendlich. $\langle s \rangle$ ist das Satzgrenzen-Tag. $\delta_t(k)$ ist die Viterbi-Wahrscheinlichkeit von Tag t an Position k. f(...) ist ein Merkmalsvektor.

Programmanfang:

```
my %weights = read_weights();     # liest die Gewichte ein
while (@satz = read_sentence()) { # liest den nächsten Satz ein
```

`read_sentence` gibt eine leere Liste zurück, wenn es keine weiteren Sätze mehr gibt.

Außer den Funktionen `read_weights` und `read_sentence` ist eine weitere Funktion `@features = get_features($w1,$w2,$w3,$prevtag,$tag)` gegeben, welche die Liste der Merkmale für das Wort `$w2` mit den Nachbarwörtern `$w1` und `$w3` und dem Tag `$tag` und dem Vorgängertag `$prevtag` liefert.

Die Variable `@tags` enthält die Liste aller Tags und ist ebenfalls gegeben.

grobe Schleifenstruktur (zur Erinnerung):

- Iteration über alle Sätze
- Initialisierung der Viterbitabelle `@viterbi`
- Iteration über alle Wortpositionen
- Iteration über alle möglichen Tags t
- Iteration über alle möglichen Vorgänger-Tags t2
- Berechnung des Scores von t durch Aufsummieren der Gewichte der Merkmale in `@features` und Addition zum Viterbi-Score des Vorgänger-Tags
- Viterbi-Maximierung (Speicherung eines Verweises in `$bestprev` nicht vergessen)
- Ausgabe der besten Tagfolge

(8 Punkte)

Viel Erfolg!