Schriftliche Prüfung zur Übung
Statistische Methoden in der maschinellen Sprachverarbeitung
WS 2016/17
Dozent: Helmut Schmid

Die Aufgaben können in Python oder Perl gelöst werden.

Aufgabe 1) Implementieren Sie eine Funktion **compute_discount(freq)**. Der Funktion wird ein Dictionary of Dictionaries übergeben mit Werten freq[word][tag] (oder analog ein Perl-Hash %freq mit $freq{word}{tag}). Sie soll nach der Formel

$$\delta = \frac{n_1}{n_1 + 2n_2}$$

den Discount berechnen und zurückgeben. n_1 (n_2) ist die Zahl der einmal (zweimal) aufgetretenen Wort-Tag-Paare. (8 Punkte)

Aufgabe 2) Implementieren Sie eine weitere Funktion **estimate_prob**, welche dasselbe Argument freq erhält und die folgenden Werte berechnet:

$$p^*(t|w) = \frac{f(w,t) - \delta}{\sum_{t'} f(w,t')}$$

und ein Dictionary of Dictionaries (oder einen Hash) mit den berechneten Werten zurückgibt. Zur Berechnung des Discounts δ können Sie die Funktion aus Aufgabe 1 aufrufen. (10 Punkte)

Aufgabe 3) Implementieren Sie nun eine Funktion **compute_backoff**, welche ein Dictionary of Dictionaries prob mit prob[word][tag] (oder einen analogen Perl-Hash) als Argument erhält, die Backoff-Faktoren berechnet und in einem Dictionary zurückgibt.

$$\text{backoff}(w) = 1 - \sum_t p^*(t|w)$$

(6 Punkte)

>>>>>>>>>>>>>>> weiter auf der nächsten Seite >>>>>>>>>>>>>>>>

Aufgabe 4) Hier ist ein Ausschnitt aus der Musterlösung für den Trigramm-HMM-Tagger. Die Funktion erhält eine Liste von Wörtern als Argument und berechnet die logarithmierten Viterbiwahrscheinlichkeiten. *lex_probs* ist eine Funktion, welche die möglichen Tags eines Wortes liefert. *context_prob* gibt die kontextuelle Wahrscheinlichkeit zurück.

```
def viterbi( words ):
   tokens = words + u'<s>'

   # Initialisierung der Viterbi-Tabelle
   vitscore = [dict()]
   vitscore[0][(u'<s>',u'<s>')] = 1.0

   # über alle Satzpositionen iterieren
   for i in range( tokens ):
      vitscore.append(dict())
      lexprob = lex_probs( tokens[k] )  # die möglichen Tags nachschlagen
      for tagpair in vitscore[i]:
         tag1, tag2 = tagpair
         for tag in lexprob:
            p = vitscore[tagpair] + log(context_prob(tagpair, tag) * lexprob[tag])
            p = p / log(apriori_tag_prob[tag])
            newpair = (tag2,tag)
            if newpair not in vitscore[i+1] or vitscore[i][newpair] < p:
               vitscore[i+1][tagpair] = p
```

Finden und korrigieren Sie mindestens 6 der 8 Fehler im Programm. (6 Punkte)

(30 Punkte insgesamt)