Übung 11

Perzeptron-Tagger

Laden Sie das manuell getaggte Korpus an der Adresse `www.cis.uni-muenchen.de/~schmid/lehre/data/tiger.txt.gz` herunter und dekomprimieren Sie es. Im Korpus folgt auf jeden Satz eine Leerzeile.

Schreiben Sie ein Programm, welches einen Tagger mit dem Perzeptron-Algorithmus trainiert. Sie sollten die folgende einfache Menge von Merkmalen verwenden und später erweitern, wenn Sie möchten.

- vorhergehendes Tag + Tag
- Wort + Tag
- Suffix + Worttyp + Tag (für Suffixe der Länge 1 bis 5, Worttyp repräsentiert die Groß-/Kleinschreibung)

Das Trainingsprogramm liest die Trainingsdaten ein und trainiert dann für eine vorgegebene Anzahl von Iterationen (z.B. 20000). In jeder Iteration wird ein Trainingssatz zufällig ausgewählt. Mit Hilfe des Viterbi-Algorithmus wird die am besten bewertete Tagfolge für den Satz berechnet (siehe unten). Wenn diese nicht mit der manuell annotierten Tagfolge übereinstimmt, werden die Gewichte angepasst, indem der Merkmalsvektor der korrekten Tagfolge zum Gewichtsvektor addiert und der Merkmalsvektor der falschen Tagfolge subtrahiert wird.

Der hier verwendete Viterbi-Algorithmus gleicht dem Viterbi-Algorithmus für HMMs. Statt logarithmierte Wahrscheinlichkeiten aufzuaddieren, werden jedoch lokale Scores summiert. Der Gewichtsvektor wird mit einem Nullvektor initialisiert. Am besten verwenden Sie hier ein `defaultdict(float)`. Dann bekommt jedes neue Gewicht automatisch zunächst den Wert 0. Hier sind die Formeln für die Berechnung der Viterbi-Werte:

$$\begin{aligned} \delta_t(0) &= \begin{cases} 0 & \text{falls t = <s>} \\ -\inf & \text{sonst} \end{cases} \\ \delta_t(i) &= \max_{t'} \delta_{t'}(i-1) + \theta \cdot \mathbf{f}(t', t, w_1^n, i) \end{aligned}$$

$\theta \cdot \mathbf{f}(t', t, w_1^n, i)$ ist das Produkt von Gewichtsvektor und lokalem Merkmalsvektor.

Am Ende des Trainings speichern Sie die Menge aller Tags und das Dictionary mit den Gewichten in einer pickle-Datei.

Anschließend schreiben Sie ein Programm, welches die Ausgabe des Trainingsprogrammes einliest und Sätze annotiert. Sie können hierfür den Code des Trainingsprogrammes recyceln.

Das Tagging-Ergebnis sollte im folgenden Format ausgegeben werden:

```
Das  PDS
ist  VAFIN
ein  ART
Satz NN
.    $.
```

Schritte:

- Funktion *read_data*, welche die Trainingsdaten in eine Liste von Listenpaaren einliest (wie beim HMM-Tagger).
- Funktion *get_features*, welche für ein gegebenes Wort, Tag und Vorgängertag den Merkmalsvektor berechnet. Der Merkmalsvektor ist eine Liste von Strings. Für die Eingabe *liest*, *VVFIN*, *PN* liefert die Funktion die Liste "TT-PN-VVFIN", "TW-VVFIN-liest", "TS-VVFIN-liest-k", "TS-VVFIN-iest-k", "TS-VVFIN-est-k", ... Die Werte der Merkmale sind jeweils 1 und werden nicht explizit repräsentiert.
- Funktion *local_score*, welche für einen gegebenen Merkmalsvektor die Summe seiner Gewichte berechnet. Da die Merkmalswerte alle 1 sind, können Sie so das Ergebnis der Multiplikation von Merkmalsvektor und Gewichtsvektor berechnen.
- Funktion *viterbi*, welche die Liste der Wörter eines Satzes als Argument bekommt und die Tagfolge mit dem höchsten Gesamtscore berechnet und zurückgibt.
- Funktion *update_weights*, welche eine Wortfolge, eine korrekte Tagfolge und eine vom Tagger gelieferte Tagfolge als Argumente nimmt, über alle Wortpositionen bis zum Endetoken iteriert, die Merkmalsvektoren an der aktuellen Position für beide Tagfolgen berechnet und für jedes Merkmal 1 von seinem Gewicht subtrahiert bzw. zu ihm addiert.
- Funktion *train*, welche die Trainingsdaten einliest und dann für N Iterationen folgende Schritte ausführt:
  - zufällige Auswahl eines Trainingssatzes (mit der Pythonfunktion *random.choice*)
  - Taggen durch Aufruf der Funktion *viterbi*
  - Anpassen des Gewichtsvektors durch Aufruf der Funktion *update_weights*

  Speicherung des Gewichtsvektors jeweils nach Training auf 1000 Sätzen

Anmerkung: Das Training kann mehrere Stunden dauern. Am besten speichern Sie den Gewichtsvektor in regelmäßigen Abständen. Dann können Sie testen, nach wievielen Iterationen die Ergebnisse am besten sind.