Schriftliche Prüfung zur Übung
Statistische Methoden in der maschinellen Sprachverarbeitung
WS 2017/18
Dozent: Helmut Schmid

Sie sollen eine Perzeptron-Tagger implementieren.

Kurze Inhaltliche Wiederholung

Perzeptron-Algorithmus

Wiederhole n mal

- Wähle ein Trainingsbeispiel (w, t)
- Wende den Klassifikator auf w an, welcher die Klasse t' liefert
- Wenn w falsch klassifiziert wurde (also $t' \neq t$), dann modifiziere den Gewichtsvektor θ durch Addition des Merkmalsvektors der korrekten Klasse und Subtraktion des Merkmalsvektors der falschen Klasse

$$\theta \leftarrow \theta + f(w, t) - f(w, t')$$

Viterbi-Algorithmus für loglineare Modelle

Initialisierung: $\delta_t(0) = \begin{cases} 0 & \text{falls } t = \langle s \rangle \\ -\inf & \text{sonst} \end{cases}$

Berechnung: (für $0 < k \leq n + 1$)

$$\begin{aligned} \delta_t(k) &= \max_{t'} \ \delta_{t'}(k-1) + s(t', t, w, k) \\ \psi_t(k) &= \arg\max_{t'} \ \delta_{t'}(k-1) + s(t', t, w, k) \end{aligned}$$

Ausgabe: (für $0 < k \leq n$)

$$\begin{aligned} t_{n+1} &= \langle s \rangle \\ t_k &= \psi_{t_{k+1}}(k+1) \end{aligned}$$

t, t' sind Tags
w ist die Wortfolge
k ist eine Wortposition
$\delta_t(k)$ ist die Viterbiwahrscheinlichkeit von Tag t an Wortposition k
$\psi_t(k)$ ist das beste Vorgängertag von Tag t an Position k
$s(t', t, w, k)$ ist der lokale Score, der sich durch Multiplikation von Gewichtsvektor θ und Merkmalsvektor $f(t', t, w, k)$ ergibt:

$$s(t', t, w, k) = \theta \cdot f(t', t, w, k)$$

Teil 1: Initialisierung

Definieren Sie eine Klasse `Tagger`, die wie folgt verwendet wird:

```
tagger = Tagger(pathname)
```

In der entsprechenden `__init__`-Funktion lesen Sie aus der Datei `pathname` die Trainingsdaten ein. Jede Zeile enthält ein Wort und ein Tag. Auf das Ende eines Satzes folgt immer eine Leerzeile. Speichern Sie die eingelesenen Daten als Liste von Paaren, die aus einer Wortliste und einer Tagliste bestehen, in der Variablen `self.data`.

Am Beginn und Ende jeder Tagfolge und Wortfolge muss das Grenzsymbol $\langle s \rangle$ hinzugefügt werden

Initialisieren Sie außerdem den Gewichtsvektor `self.weight` als leeres Dictionary.

Berechnen Sie die Menge der Tags in `self.tagset`. (6 Punkte)

Teil 2: Merkmalsextraktion

Schreiben Sie eine Methode `get_features(prevtag, tag, words, pos)` für die Klasse Tagger, welche den Merkmalsvektor berechnet und eine Liste mit einem Tag-Tag-Merkmal und einem Tag-Wort-Merkmal zurückgibt.

Beispiel:

Wenn Sie die Funktion mit `get_features('PRO', 'VVFIN', ['<s>','Es','schneit','<s>'], 2)` aufrufen, soll die Liste `['TT-PRO-VVFIN','TW-VVFIN-schneit']` zurückgegeben werden.

(3 Punkte)

Teil 3: Berechnung des lokalen Scores

Schreiben Sie eine Methode `local_score(prevtag, tag, words, pos)`. Diese ruft zunächst die Funktion `get_features` auf, um den Merkmalsvektor zu erhalten. Dann berechnet sie das Produkt aus Merkmalsvektor und Gewichtsvektor, indem sie über alle Merkmale in der Merkmalsliste iteriert und die Gewichte der Merkmale summiert. Die berechnete Summe wird zurückgegeben. (4 Punkte)

Teil 4: Viterbi-Algorithmus

Schreiben Sie nun eine Methode `viterbi(words)`, welche mit dem Viterbi-Algorithmus die beste Tagfolge berechnet.

Schritte:

- `vit_score` und `best_prev` als Liste von Dictionaries initialisieren.
- Für alle Wortpositionen, möglichen Tags (in `self.tagset`) und Vorgängertags
  - den lokalen Score berechnen, mit dem Viterbi-Score des Vorgänger-Tags addieren und (über alle Vorgänger-Tags) maximieren. Dabei das beste Vorgängertag in `best_prev` merken.
- schließlich die beste Tagfolge aus `best_prev` extrahieren und zurückgeben

(10 Punkte)

Teil 5: Training auf Satz

Schreiben Sie eine Methode `train_on_sent(words,tags)`, welche eine Wortfolge und eine Tagfolge als Argumente erhält und einen Trainingsschritt ausführt.

Schritte:

- Berechnen Sie die beste Tagfolge für die Wortfolge mit der Funktion `self.viterbi`.
- Iterieren Sie über alle relevanten Wortpositionen
  - Wenn die Tags an der aktuellen Wortposition in der korrekten und der berechneten Tagfolge verschieden sind, oder wenn die beiden Vorgänger-Tags verschieden sind
    * Berechne den Merkmalsvektor an der aktuellen Position für das korrekte Tagpaar
    * Erhöhe das Gewicht aller Merkmale aus diesem Merkmalsvektor um 1
    * Berechne den Merkmalsvektor an der aktuellen Position für das falsche Tagpaar
    * Reduziere das Gewicht aller Merkmale aus diesem Merkmalsvektor um 1
- Für die oben beschriebene Addition/Subtraktion der Merkmalsvektoren vom Gewichtsvektor schreiben Sie eine weitere Methode `update_weights(features, inc)`, um Redundanz zu vermeiden. Der Wert von "inc" ist hier entweder +1 oder -1.

(6 Punkte)

Teil 6: Training

Schreiben Sie eine Methode `train()`, welche 100000 Mal (mit random.choice) zufällig einen Satz der Trainingsdaten auswählt und damit die Funktion `train_on_sent` aufruft.

(1 Punkt)

(30 Punkte insgesamt)

Viel Erfolg!