LC-CRF-Wortart-Tagger 2: Optimierung und Tagging

1) Optimierung des LC-CRF-Tagger-Trainings

In der letzten Aufgabe haben Sie einen CRF-Tagger trainiert. Korrigieren Sie diesen Code mit Hilfe der erhaltenen Kommentare. Nun sollen Sie das Trainingsprogramm verbessern durch

- Evaluierung der Tagging-Genauigkeit auf den Developmentdaten nach jeder Epoche und Speicherung des besten Gewichtsvektors
- L_1-Regularisierung
- Beschleunigung durch Pruning an zwei Stellen:

  Im Forward-Algorithmus berechnen Sie die lexikalischen Scores aller Tags an der aktuellen Position und iterieren dann nur über die Tags mit einem Score größer dem maximalen Score plus log(Schwellwert) (wobei der Schwellwert bspw. bei 0.001 liegt).

  Wenn Sie über alle Tags an Position i iteriert haben, dann eliminieren Sie in forward[i] alle Tags, deren Alpha-Scores (logarithmierte Darstellung) über dem maximalen alpha-Score plus log(Schwellwert) liegt. Der backward-Algorithmus iteriert nur über die Tags an Position i, die in forward[i] eingetragen sind.
- Beschleunigung durch Caching:

  Die Berechnung der Merkmalsvektoren und der Scores ist rechenintensiv und erfolgt mehrfach. Sie können das Programm beschleunigen, indem Sie diese Werte in Caches speichern. Als Key dient jeweils das Argument-Tag(-Paar) plus Position. Die Caches werden nach jedem Satz gelöscht, damit der Speicherplatzbedarf begrenzt bleibt.

  Die Funktion für die Berechnung eines Merkmalsvektors oder Scores schaut zunächst im entsprechenden Cache nach, ob das Ergebnis bereits berechnet wurde. Falls ja, wird das Ergebnis im Cache zurückgegeben. Andernfalls wird der Wert berechnet, im Cache gespeichert und zurückgegeben.
- neuer Aufruf: **crf-train.py train-file dev-file param-file**

2) Implementierung eines Taggerprogrammes

Außerdem sollen Sie ein Programm schreiben, welches die gespeicherten Parameter einliest und dann Eingabesätze mit dem Viterbi-Algorithmus annotiert. Die Eingabesätze werden aus einer Datei eingelesen, die ein Wort pro Zeile enthält, wobei auf jeden Satz eine Leerzeile folgt. Die Ausgabe erfolgt im gleichen Format wie die Trainingsdaten.

Aufruf: **crf-annotate.py param-file text.txt**

Vorüberlegungen

- Wie implementieren Sie die Evaluierung?
- Wie kann die L_1-Regularisierung effizienter gemacht werden?
- Welche Teilaufgaben umfasst das Taggerprogramm?

Optimieren Sie den L_1-Regularisierungsfaktor und die Lernrate auf Developmentdaten und berechnen Sie zum Schluss die Tagginggenauigkeit auf den Testdaten. Auf den Developmentdaten sind über 97% Genauigkeit möglich. Das Training kann über eine Stunde dauern.

Schicken Sie alle Programme, den optimalen Metaparameter und die erzielte Genauigkeit auf den Testdaten an `schmid@cis.lmu.de`.