Parsing mit neuronalen Netzen: Training

Nun sollen Sie das Parser-Training implementieren.

Korrigieren Sie zunächst die Fehler in Ihrem Code aus der letzten Aufgabe mit Hilfe der erhaltenen Korrekturvorschläge.

Erstellen Sie dann eine Datei **train-parser.py**, welche eine Klasse *Data* für die Datenvorverarbeitung und Ihre Klasse *Parser* für das Netzwerk importiert. Die Klasse *Data* enthält Funktionen zur Vorverarbeitung der Daten. Sie können den Code an der Adresse `https://www.cis.uni-muenchen.de/~schmid/lehre/Experimente/data/Code-Aufgabe10.zip` herunterladen.

Dort definieren Sie eine Funktion **loss**, welche die Log-likelihood der korrekten Span-Labels mit der CrossEntropyLoss-Funktion berechnet und zurückgibt.

Implementieren Sie außerdem eine Funktion **num_errors**, welche die Zahl der falsch gelabelten Spans berechnet und zurückgibt. Sie können die beiden Funktionen *loss* und *num_errors* auch zu einer Funktion *loss* zusammenfassen, die beides berechnet.

Schreiben Sie nun eine Funktion **train**. Erzeugen Sie darin zunächst ein Objekt der Klasse *Data* mit dem Befehl

```
data = Data(path_train, path_dev)
```

Dann trainieren Sie für n Epochen (bspw. n=50) auf den Trainingsdaten *data.train_parses*, die Sie zu Beginn jeder Epoche mit *random.shuffle* umordnen. *data.train_parses* ist eine Liste von Parsebäumen. Jeder Parsebaum ist ein Paar bestehend aus der Liste der Wörter und der Liste der Konstituenten. Jede Konstituente ist ein Tripel (Label, Startposition, Endposition). Mit der Methode *data.label2ID* können Sie die Labels auf IDs abbilden. Mit der Methode *data.words2charIDvec(words)* können Sie eine Liste von Wörtern in eine Liste von Suffixen und eine Liste von Präfixen konvertieren, wobei jedes Suffix/Präfix aus einer Liste von Buchstaben-IDs besteht. In der Datei Data.py ist auch der Index der "no-constituent"-Spanklasse in der Variablen *NoConstLabelID* verfügbar.

Nach jeder Epoche berechnen Sie die Zahl der falsch gelabelten Spans in den Development-Daten. Wenn die Zahl der Fehler die bisher kleinste war, speichern Sie das Netzwerk mit der Methode *torch.save*. Außerdem rufen Sie die Methode *data.store_parameters(filename)* auf, um die Tabellen für die Abbildung von Buchstaben und Labels auf Zahlen-IDs zu speichern.

Bei einer guten Implementierung kann die Zahl der falsch gelabelten Konstituenten in den Development-Daten im Laufe des Trainings unter 6000 sinken.

Schicken Sie bitte mit dem Code eine Tabelle mit der Anzahl der Fehler nach jeder Epoche.

Vorüberlegungen

- Welche Argumente müssen Sie den Funktionen *loss* und *num_errors* übergeben?
- Wie berechnen Sie das Loss am besten?
- Wie berechnen Sie die Zahl der falsch gelabelten Spans?
- Welche Schritte sind für das Training erforderlich?