Parsing mit neuronalen Netzen: Netzwerk

In dieser Übung implementieren Sie das neuronale Netzwerk mit der Klasse **Parser**.

Aus Effizienzgründen werden die **Wortrepräsentationen** nicht aus der gesamten Buchstabenfolge berechnet sondern aus **Suffixen/Präfixen** fester Länge. Dies erleichtert die Parallelverarbeitung ohne Qualitätseinbußen.

Die Eingabe des Netzwerkes besteht daher aus zwei Tensoren: Der erste Tensor enthält alle **Wortsuffixe** fester Länge (z.B. 10). Kürzere Wörter werden am Anfang mit Paddingsymbolen aufgefüllt. Der zweite Tensor enthält alle **Wortpräfixe** derselben Länge in umgekehrter Reihenfolge, ebenfalls mit Padding.

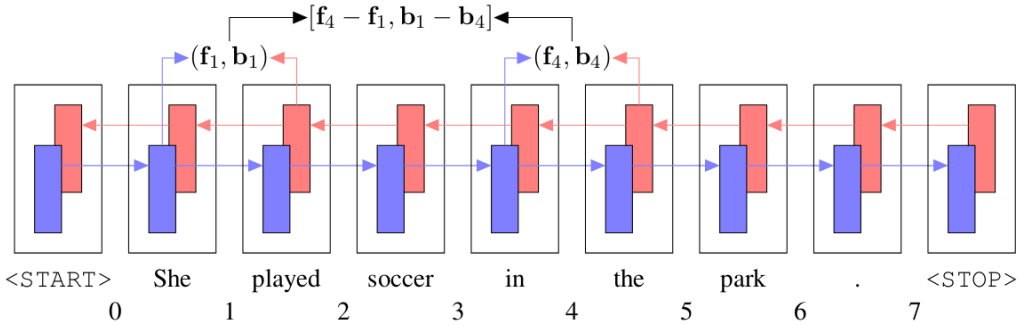
Die Tensoren speichern Buchstaben-IDs und haben die Dimension N x L, wobei N die Satzlänge und L die Präfix-/Suffix-Länge ist. Diese Vorverarbeitung müssen Sie nicht implementieren.

Das neuronale **Netzwerk** erledigt folgende Aufgaben:

- Berechnung der **Wortrepräsentationen**: Ein Forward-LSTM und ein Backward-LSTM berechnen jeweils eine Repräsentation für jedes Eingabewort auf Basis der Embeddings der Buchstaben des Wort-Suffixes bzw. -Präfixes. Die Endzustände der beiden LSTMs werden konkateniert und ergeben die Wortrepräsentation.

  Anm.: Da die beiden LSTMs auf unterschiedlichen Eingaben operieren, können Sie hier kein bidirektionales LSTM verwenden.

- Berechnung der **Span-Repräsentationen**: Ein (tiefes) bidirektionales LSTM verarbeitet den zweidimensionalen Tensor mit Wortrepräsentationen und liefert eine Repräsentation für jede Satzposition. Sie teilen diese Repräsentationen in zwei Hälften. Die erste Hälfte liefert die Forward-Repräsentation, die zweite Hälfte die Backward-Repräsentation. Damit Sie mit diesen beiden neuen Tensoren – wie im Schaubild gezeigt – die Span-Repräsentationen berechnen können, entfernen Sie beim forward-Tensor das letzte Element und beim backward-Tensor das erste Element.



  Nun berechnen Sie durch Subtraktion der Startrepräsentationen aller Spans von ihren Endrepräsentationen die Spanrepräsentationen. Mit der PyTorch-Methode `combinations` können Sie die Start- und Endpositionen aller Spans generieren. Mit `forward[endpositions] - forward[startpositions]` und `backward[startpositions] - backward[endpositions]` können Sie dann

alle forward- und backward-Differenzvektoren parallel berechnen. Die Differenzvektoren konkatenieren Sie.

Da Sie für die Spanberechnungen auch Repräsentationen für die Positionen 0 und Satzlänge+1 brauchen, müssen Sie am Anfang und Ende des BiLSTM-Eingabe-Tensors einen Nullvektor als Dummy-Eingabe hinzufügen. (Sie können dafür auch die PyTorch-Funktion `pad` verwenden.)

- Berechnung der **Bewertungen** der Kategorien: Ein Feedforward-Netzwerk mit einer Hidden Layer transformiert jede Spanrepräsentation in einen Vektor, der für jede syntaktische Kategorie einen Score liefert. Die Größe der Hidden Layer sollte frei wählbar sein. Die Bewertung des Labels "keine Konstituente" (mit der ID 0) sollte überall auf 0 gesetzt werden.

In dieser Übungsaufgabe implementieren Sie noch nicht das Gesamtsystem, sondern nur das neuronale Netzwerk. Sie können also noch nicht auf realen Daten trainieren.

Aufruf: python Parser.py

Vorüberlegungen

- Wie **gliedern** Sie das Netzwerk sinnvoll in Teilnetzwerke?
- Welche Argumente sollten die **Konstruktoren** der Teilnetzwerke erhalten?
- Welche Argumente sollten die **Forward**-Funktionen der Teilnetzwerke erhalten und welche Rückgabewerte sollten sie liefern?
- Wo sollte **Dropout** angewendet werden?
- Wie können Sie ihr Netzwerk separat **testen**?

Anmerkung: Aus Effizienzgründen sollten Sie möglichst viel durch Verwendung von Tensor-Operationen parallel verarbeiten. (Eine parallele Verarbeitung mehrerer Sätze müssen Sie aber nicht versuchen.)