Konstituentenparsing mit neuronalen Netzen

Grundlage: Gaddy et al. (Link auf Kursseite)

Strategie:

- Berechne für jede mögliche Konstituente (i, k, l) mit Startposition i, Endposition k und Kategorie l eine Bewertung $s(i, k, l)$.
- Die Bewertung eines Parsebaumes T ist die Summe der Bewertungen aller enthaltenen Konstituenten:

$$s(T) = \sum_{(i,k,l) \in T} s(i, k, l)$$

- Der Parser sucht den am besten bewerteten Parse:

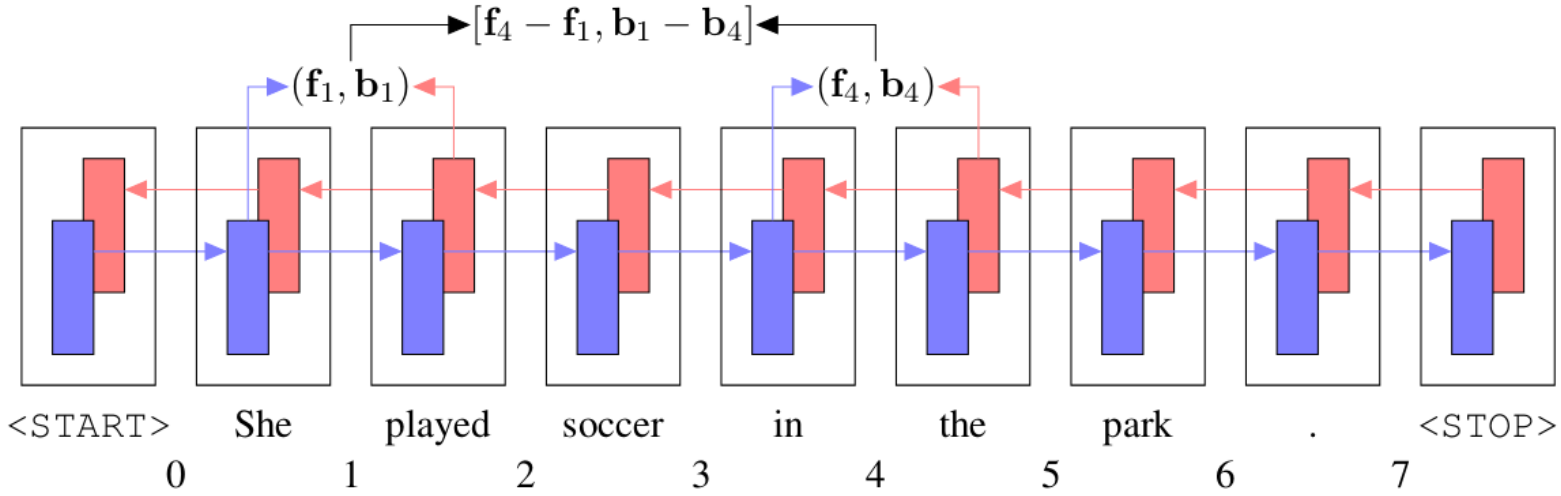
$$\hat{T} = \arg\max_{T} s(T)$$

Anm.: Die Labels der Tochter-Konstituenten haben keinen Einfluss auf den Score $s(i, k, l)$ einer Konstituente.

Repräsentation der Konstituenten

- Ein BiLSTM berechnet eine Vorwärts-Repräsentation f_i und eine Rückwärts-Repräsentation b_i für jede Satzposition i.
- Die Repräsentation r_{ik} einer Konstituente (i, k) wird als Konkatenation zweier Differenzvektoren definiert:

$$\mathbf{r}_{ik} = [\mathbf{f}_k - \mathbf{f}_i, \mathbf{b}_i - \mathbf{b}_k]$$



Bewertung der Konstituenten-Kategorien

Die Bewertungen der Kategorien werden durch ein neuronales Netz mit einer Hidden Layer aus der Konstituenten-Repräsentation berechnet:

$$s(i,k,l) = [\mathbf{W}_2 g(\mathbf{W}_1 \mathbf{r}_{ik} + z_1) + z_2]_l$$

g ist hier die Aktivierungsfunktion, z.B. ReLU.

Die Ausgabeebene enthält für jede Konstituenten-Kategorie ein Neuron und verwendet keine Aktivierungsfunktion.

Parsing

- verwendet den CKY-Algorithmus
- Kettenregeln der Form VP $\rightarrow$ V werden durch komplexe Kategorien VP-V ersetzt.
- Nicht-binäre Konstituenten wie VP $\rightarrow$ V NP PP werden geparst, indem zunächst V und NP zu einer Hilfskonstituente der Kategorie $\emptyset$ zusammengefasst werden, wobei $s(i, k, \emptyset) = 0$.

Parsing

- Berechnung der Viterbi-Scores von Konstituenten der Länge 1

$$\delta(i, i+1) = \max_l s(i, i+1, l)$$

  = Bewertung der besten Kategorie l

- Berechnung der Viterbi-Scores von Konstituenten der Länge $k - i > 1$

$$\delta(i, k) = \max_l s(i, k, l) + \max_{i<j<k} [\delta(i, j) + \delta(j, k)]$$

Wichtig: Die beste Kategorie l und die beste binäre Zerlegung der Konstituente können unabhängig voneinander berechnet werden.

Parsing

- Der Parser berechnet bottom-up alle Viterbi-Scores $\delta(i, k)$.
- Er merkt sich dabei für jede (mögliche) Konstituente die beste Kategorie und die beste binäre Zerlegung.
- Schließlich wird der beste Parsebaum top-down extrahiert.

Pseudocode

compute $score[i, k, sym]$ for all i, k, sym # Bewertungen aller möglichen Konstituenten
for l in 1...n **do** # Für alle Konstituentenlängen
    for i in 0...n-l **do** # Für alle Startpositionen
        k = i+l # Endposition
        $vscore[i, k] = \max_{sym} score[i, k, sym]$ # Beste Bewertung
        $label[i, k] = \arg\max_{sym} score[i, k, sym]$ # Beste Kategorie
        if l > 1 **then**
            $split[i, k] = \arg\max_{i<j<k} vscore[i, j] + vscore[j, k]$ # Beste Zerlegung
            $vscore[i, k] += vscore[i, split[i, k]] + vscore[split[i, k], k]$
output(0, n) # Parsebaum ausgeben

def output(i,k)
    print "(", label[i,k]
      # Kettenregeln und Dummy-Konstituenten sind hier nicht berücksichtigt
    if k == i+1 **then**
        print word[i]
    else
        output(i, split[i,k]); output(split[i,k], k) # Rekursion
    print ")"

Parser-Training

Wir verwenden einen gegenüber dem Originalparser leicht vereinfachten Trainingsalgorithmus und maximieren die Summe der logarithmierten Wahrscheinlichkeiten der korrekten Labels für alle möglichen Konstituenten (i,k):

$$\sum_{i=0}^{n-1} \sum_{k=i+1}^{n} \log p_{i,k,\text{label}[i,k]}$$

label$[i, k]$ ist gleich der ID der korrekten Kategorie, falls $w_i, ..., w_{k-1}$ eine Konstituente ist, und andernfalls gleich der ID von $\emptyset$.

$p_{i,k,l} = [\text{softmax}(\text{score}_{i.k})]_l$ ist die Wahrscheinlichkeit der Kategorie mit der ID l für die Wortfolge $w_i, ..., w_{k-1}$.