Parsing Related Works

Shiyue Zhang

Deep Biaffine Attention for Neural Dependency Parsing

- Basic info:
  - Authors
    - Timothy Dozat, Stanford University, tdozat@stanford.edu
    - Christopher D. Manning, Stanford University, manning@stanford.edu
  - Status
    - arXiv 6 Nov 2016
    - Under review as a conference paper at ICLR 2017
- Key words:
  - Graph-based dependency parsing, BiLSTM, Biaffine attention
- Main Contributions:
  - Biaffine attention, adam beta2=0.9

Deep Biaffine Attention for Neural Dependency Parsing

- Model
- Dependency arc prediction
  - Score potential arcs

$$\mathbf{h}_i^{(arc\text{-}dep)} = \mathbf{MLP}^{(arc\text{-}dep)}(\mathbf{r}_i)$$
$$\mathbf{h}_j^{(arc\text{-}head)} = \mathbf{MLP}^{(arc\text{-}head)}(\mathbf{r}_j)$$

$$s_{ij}^{(arc)} = \mathbf{h}_i^{\top(arc\text{-}dep)} U^{(arc)} \mathbf{h}_j^{(arc\text{-}head)} + \mathbf{w}^{\top(arc)} \mathbf{h}_j^{(arc\text{-}head)}$$

- Dependency relation prediction
  - Score potential relations

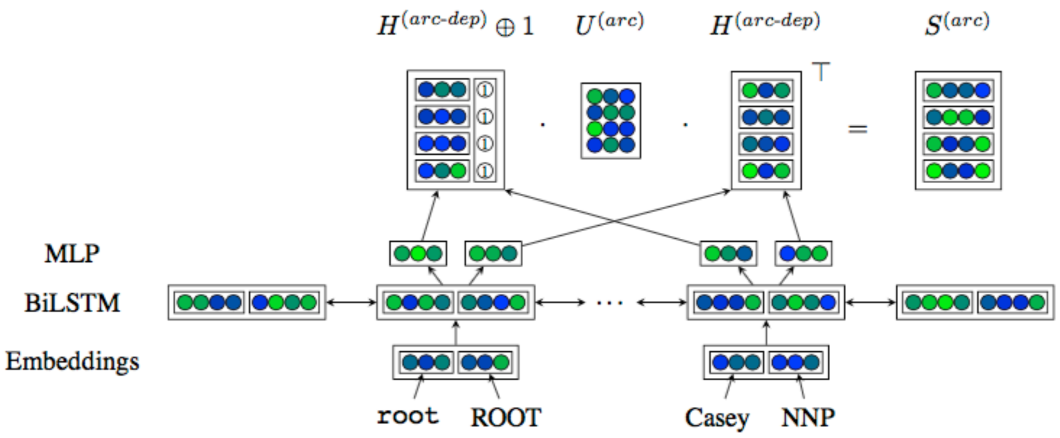

Figure 2: Deep biaffine neural dependency parser applied to the sentence "Casey hugged Kim". We concatenate a vector of ones to $H^{(arc\text{-}dep)}$ to make the scorer biaffine rather than bilinear.

$$\mathbf{h}_i^{(rel\text{-}dep)} = \mathbf{MLP}^{(rel\text{-}dep)}(\mathbf{r}_i)$$
$$\mathbf{h}_{y_i^{(arc)}}^{(rel\text{-}head)} = \mathbf{MLP}^{(rel\text{-}head)}(\mathbf{r}_{y_i^{(arc)}})$$
$$\mathbf{s}_i^{(rel)} = \mathbf{h}_i^{(rel\text{-}dep)} \mathbf{U}^{(rel)} \mathbf{h}_{y_i^{(arc)}}^{(rel\text{-}head)} + W^{(rel)} \left(\mathbf{h}_i^{(rel\text{-}dep)} \oplus \mathbf{h}_{y_i^{(arc)}}^{(rel\text{-}head)}\right) + \mathbf{b}^{(rel)}$$

Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations

- Basic info:
  - Authors
    - Eliyahu Kiperwasser, Bar-Ilan University, elikip@gmail.com
    - Yoav Goldberg, Bar-Ilan University, yoav.goldberg@gmail.com
  - Status
    - arXiv 14 Mar 2016
    - *Transactions of the Association for Computational Linguis- tics*, 4:313–327, 2016.
- Key words:
  - Dependency parsing, BiLSTM, Attention
- Main Contributions:
  - BiLSTM, Attention

Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations

- Model
- Transition-based Parser
  - feature function ϕ(c) is the concatenated BiLSTM vectors of the top 3 items on the stack and the first item on the buffer.

Algorithm 1 Greedy transition-based parsing

1: **Input:** sentence $s = w_1, \ldots, x_w,\ t_1, \ldots, t_n$, parameterized function $\text{SCORE}_\theta(\cdot)$ with parameters θ.
2: $c \leftarrow \text{INITIAL}(s)$
3: **while not** $\text{TERMINAL}(c)$ **do**
4: $\quad \hat{t} \leftarrow \arg\max_{t \in \text{LEGAL}(c)} \text{SCORE}_\theta(\phi(c), t)$
5: $\quad c \leftarrow \hat{t}(c)$
6: **return** $tree(c)$

$$\phi(c) = v_{s_2} \circ v_{s_1} \circ v_{s_0} \circ v_{b_0}$$
$$v_i = \text{BILSTM}(x_{1:n}, i)$$

- Graph-based Parser

$$\begin{aligned} parse(s) &= \arg\max_{y \in \mathcal{Y}(s)} score_{global}(s, y) \\ &= \arg\max_{y \in \mathcal{Y}(s)} \sum_{(h,m) \in y} score(\phi(s, h, m)) \\ &= \arg\max_{y \in \mathcal{Y}(s)} \sum_{(h,m) \in y} MLP(v_h \circ v_m) \end{aligned}$$
$$v_i = \text{BIRNN}(x_{1:n}, i)$$

Bi-directional Attention with Agreement for Dependency Parsing

- Basic info:
  - Authors
    - Hao Cheng, Hao Fang, University of Washington, {chenghao, hfang}@uw.edu
    - Xiaodong He, Jianfeng Gao, Li Deng, Microsoft Research, {xiaohe, jfgao, deng}@microsoft.com
  - Status
    - arXiv 22 Sep 2016
    - EMNLP 2016.
- Key words:
  - Dependency parsing, BiLSTM, Attention
- Main Contributions:
  - BiLSTM, Attention

Bi-directional Attention with Agreement for Dependency Parsing

- Paper said "To the best of our knowledge, this is the first at- tempt to apply memory network models to graph- based dependency parsing". But to be honest it is a bidirectional attention model.
- Components:
  - Memory: $\mathbf{m}_j = \left[\mathbf{h}_j^l; \mathbf{h}_j^r\right]$
  - Query:

$$s_{t,j} = \mathbf{v}^T \phi\left(\mathbf{C}\mathbf{m}_j + \mathbf{D}\mathbf{q}_t\right) \qquad \mathbf{a}_t = \text{softmax}(\mathbf{s}_t)$$

$$\tilde{\mathbf{m}}_t = \sum_{j=1}^{n} a_{t,j}\mathbf{m}_j \qquad \mathbf{q}_t = \text{GRU}\left(\mathbf{q}_{t-1}, [\tilde{\mathbf{m}}_t; \mathbf{x}_t]\right)$$

  - Prediction:

$$\mathbf{y}_t = \text{softmax}\left(\mathbf{U}\left[\tilde{\mathbf{m}}_t^l;\ \tilde{\mathbf{m}}_t^r\right] + \mathbf{W}\left[\mathbf{q}_t^l;\ \mathbf{q}_t^r\right]\right)$$