Second-Order Unsupervised Neural Dependency Parsing

Songlin Yang1, Yong Jiang2, Wenjuan Han3, Kewei Tu4
1School of Information Science and Technology, ShanghaiTech University
2Alibaba DAMO Academy, Alibaba Group
3Department of Computer Science, National University of Singapore

Motivation
Most of the unsupervised dependency parsers are based on first-order probabilistic generative models that only consider local parent-child information, which lacks expressiveness.

Proposed Solution
We proposed a second-order extension of unsupervised neural dependency models that incorporate grandparent-child or sibling information. To reduce the number of parameters, we use the decomposed trilinear function to score. To reduce the number of grammar rules, we use the agreement-based learning framework to jointly train a second-order unlexicalized model and a first-order lexicalized model.

Learning via Gradient Method
\[
\nabla_x (\log p(x)) = \sum_{z \in T(x)} p_z(z | x) \nabla_z \log p_z(x, z) \\
= \sum_{z \in T(x)} p_z(z | x) \sum_{r \in R} c(r, x, z) \nabla_r \log p_r(r) \\
= \sum_{r \in R} e(r, x) \nabla_r \log p_r(r)
\]
where \(c(r, x, z)\) is the number of times rule \(r\) is used in dependency parse tree \(z\) of sentence \(x\), and \(e(r, x)\) is the expected count of rule \(r\) in sentence \(x\), \(T(x)\) is the whole set of parse tree. In the EM algorithm, we need to calculate the expected counts of grammar rules for the entire training data set; while in mini-batch gradient ascent, we calculate the expected counts of grammar rules in the mini-batch, which is similar to online EM algorithm.

Neural Architecture and Decomposed Trilinear Scoring Function
We use a shared POS embedding layer and apply different linear transformations to get role-specific representations for child, parent, grandparent, or sibling, then go through MLP layers to encode the direction and valence information. We use a decomposed trilinear function to compute the unnormalized rule probability from the three vectors \(h_c, h_p, h_v\).

Experiments
We propose a second-order extension of unsupervised neural dependency models that incorporate grandparent-child or sibling information. To reduce the number of parameters, we use the decomposed trilinear function to score. To reduce the number of grammar rules, we use the agreement-based learning framework to jointly train a second-order unlexicalized model and a first-order lexicalized model.

Learning via Gradient Method
\[
\nabla_x (\log p(x)) = \sum_{z \in T(x)} p_z(z | x) \nabla_z \log p_z(x, z) \\
= \sum_{z \in T(x)} p_z(z | x) \sum_{r \in R} c(r, x, z) \nabla_r \log p_r(r) \\
= \sum_{r \in R} e(r, x) \nabla_r \log p_r(r)
\]
where \(c(r, x, z)\) is the number of times rule \(r\) is used in dependency parse tree \(z\) of sentence \(x\), and \(e(r, x)\) is the expected count of rule \(r\) in sentence \(x\), \(T(x)\) is the whole set of parse tree. In the EM algorithm, we need to calculate the expected counts of grammar rules for the entire training data set; while in mini-batch gradient ascent, we calculate the expected counts of grammar rules in the mini-batch, which is similar to online EM algorithm.

Second-Order Parsing
The picture shows the dynamic-programming structures and derivations of second-order (grandparent-child variant) parsing model (Koo and Collin, 2010). We design dynamic programming algorithms adapted from their ideas to calculate the marginal likelihood and obtain the Viterbi parse tree.

Joint parsing is helpful and joint training improves the performance of both models.

Experiments
We propose a second-order extension of unsupervised neural dependency models that incorporate grandparent-child or sibling information. To reduce the number of parameters, we use the decomposed trilinear function to score. To reduce the number of grammar rules, we use the agreement-based learning framework to jointly train a second-order unlexicalized model and a first-order lexicalized model.

Learning via Gradient Method
\[
\nabla_x (\log p(x)) = \sum_{z \in T(x)} p_z(z | x) \nabla_z \log p_z(x, z) \\
= \sum_{z \in T(x)} p_z(z | x) \sum_{r \in R} c(r, x, z) \nabla_r \log p_r(r) \\
= \sum_{r \in R} e(r, x) \nabla_r \log p_r(r)
\]
where \(c(r, x, z)\) is the number of times rule \(r\) is used in dependency parse tree \(z\) of sentence \(x\), and \(e(r, x)\) is the expected count of rule \(r\) in sentence \(x\), \(T(x)\) is the whole set of parse tree. In the EM algorithm, we need to calculate the expected counts of grammar rules for the entire training data set; while in mini-batch gradient ascent, we calculate the expected counts of grammar rules in the mini-batch, which is similar to online EM algorithm.