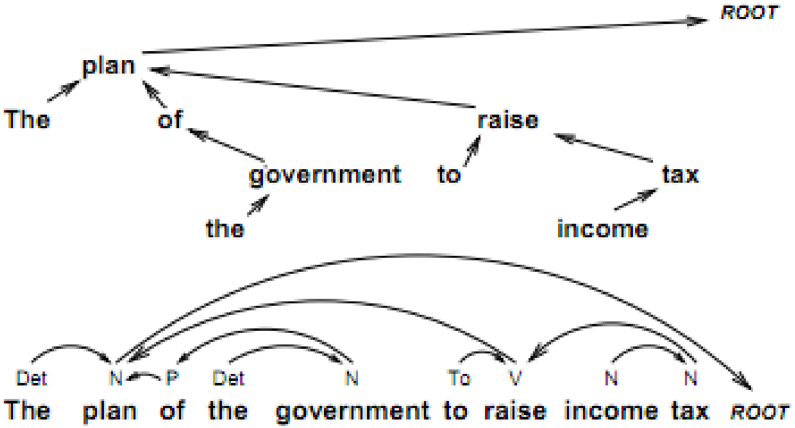
An Effective Neural Network Model for Graph-based Dependency Parsing

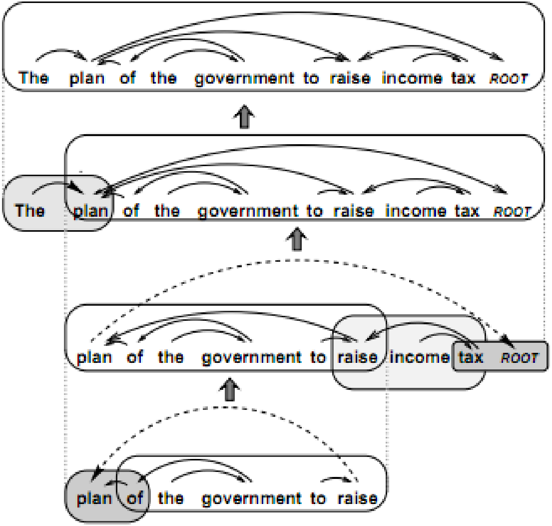
A reimplementation

Wenzhe Pei, Tao Ge, Baobao Chang

Introduction



Introduction



Introduction

- Given a sentence x, graph-based models formulate the parsing process as a searching problem:

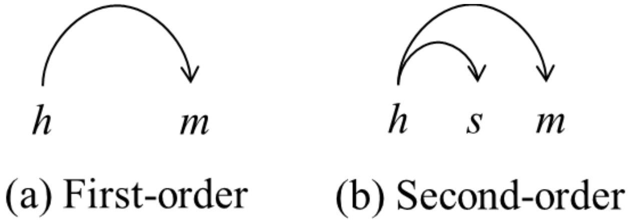
$$y^*(x) = \text{argmax}_{\hat{y} \in Y(x)} \, Score(x, \hat{y}(x); \theta)$$

Introduction

- For efficient decoding, previous work uses factorization

$$Score(x, \hat{y}(x); \theta) = \sum_{c \in \hat{y}(x)} ScoreF(x, c; \theta)$$

- c is the subgraph of the tree $\hat{y}(x)$



Introduction

- The most common choice for the score function is

$$ScoreF(x, c; \theta) = \mathbf{w} \cdot \mathbf{f}(x, c)$$

- Problems
  - A mass of features could put the model at risk of overfitting
  - Feature extraction slows down the parsing speed
  - Feature design requires domain expertise

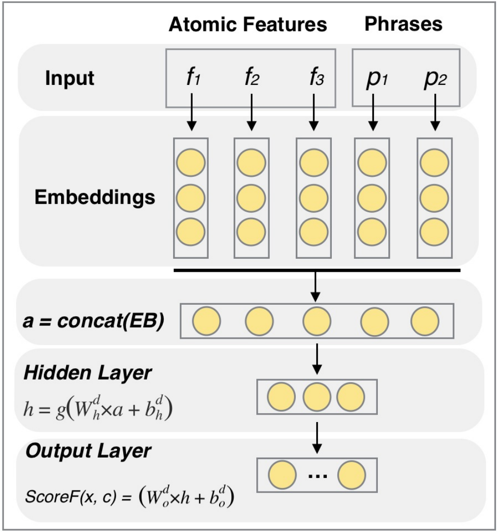
Introduction

- In this paper, we present a neural network model for graph-based parsing

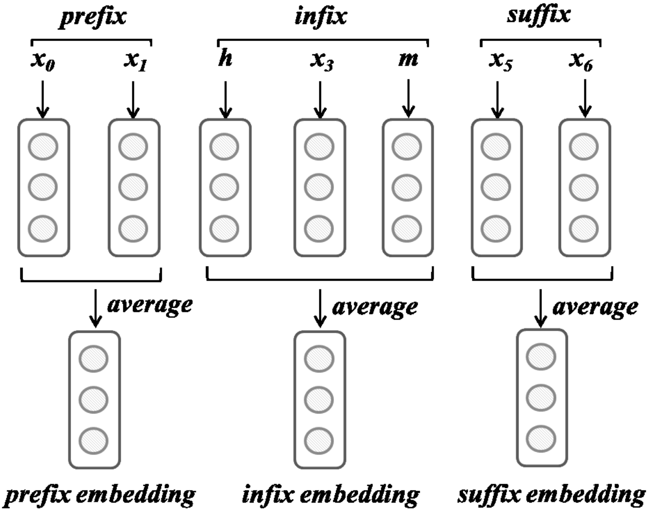
$$ScoreF(x, c; \theta) = NN(x, c)$$

- Our model can
  - Learn feature combinations automatically
  - Exploit phrase-level information through distributed representation
  - Generalize to any graph-based models (first-order, second-order, third-order, . . .)

Model Details



Model Details



Model Details

- Learning feature combinations
  - New activation function: *tanh-cube*

$$g(l) = tanh(l^3 + l)$$

  - Intuitively, the cube term in each hidden unit directly models feature combinations in a multiplicative way

$$(w_1 a_1 + w_2 a_2 + \cdots + w_n a_n + b)^3 = \sum_{i,j,k} (w_i w_j w_k) a_i a_j a_k + \sum_{i,j} b(w_i w_j) a_i a_j \ldots$$

Improvements

- Speed up Eisner algorithm parsing by taking advantage of precomputed word embeddings
- Generate CUDA code for filling in the dynamic programming table

Experiments

- Dataset
  - English Penn TreeBank
  - Use Stanford POS Tagger for POS-tagging
  - Speed testing using all sentences from Wikipedia