Fast and Accurate Non-Projective Dependency Tree Linearization

Xiang Yu, Simon Tannert, Ngoc Thang Vu, Jonas Kuhn
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart, Germany
firstname.lastname@ims.uni-stuttgart.de

Abstract

We propose a graph-based method to tackle the dependency tree linearization task. We formulate the task as a Traveling Salesman Problem (TSP), and use a biaffine attention model to calculate the edge costs. We facilitate the decoding by solving the TSP for each subtree and combining the solution into a projective tree. We then design a transition system as post-processing, inspired by non-projective transition-based parsing, to obtain non-projective sentences. Our proposed method outperforms the state-of-the-art linearizer while being 10 times faster in training and decoding.

1 Introduction

Surface realization is the task of generating a sentence from a syntactic or semantic representation. In several shared tasks (Belz et al., 2011; Mille et al., 2018, 2019), the input representations are unordered dependency trees. The state-of-the-art system (Yu et al., 2019a) in the Surface Realization Shared Task 2019 (SR'19) takes a pipeline approach, where the first step is linearization, namely ordering the tokens in the dependency tree. They use a Tree-LSTM to encode each token with awareness of the whole tree, then apply the divide-and-

Several works have followed this formulation. Among others, Zaslavskiy et al. (2009) formulate the word ordering in phrase-based machine translation as a TSP, and show that it achieves better performance and speed than beam search decoding with the same bigram language model. Horvat and Byrne (2014) explore higher-order n-gram language models for TSP-based word ordering, which transforms into a much larger TSP graph. All of the aforementioned works operate on a bag of words *without syntax*, which is a TSP graph of non-trivial size with little information about the internal structure. Much effort has been put into incorporating more powerful decoding algorithms such as Integer Programming (Germann et al., 2001) and Dynamic Programming (Tillmann and Ney, 2003).

Our work differs from the previous work on TSP-based word ordering in several aspects. (1) Linearization is a special case of word ordering *with syntax*, where we can use a tree-structured encoder to provide better representation of the tokens. (2) We adopt the divide-and-conquer strategy to break down the full tree into subtrees and order each subtree separately, which is faster and more reliable with an approximate decoder. (3) We apply deep biaffine attention (Dozat and Manning, 2016), which has yielded great improvements in dependency parsing, and reinterpret it as a bigram