LEARNING FROM GRAPH-BASED STRUCTURES IN NLP

A PREPRINT

Tarun Sunkaraneni
CS6190: Probabilistic Modeling
University of Utah
tarun.sunkaraneni@utah.edu

December 14, 2019

ABSTRACT

Ever since representation learning of word embeddings became powerful in understanding textual information, Natural Language processing has been able to make great advances in area such as translation, question answering and inference. Even though most NLP models extract and utilize sequential information efficiently, graphical structures aren't prominently used in prominent tasks. This paper aims to compare compare and contrast the effectiveness of using Bayesian Neural Networks (BNN) on for dependency parsing.

1 Introduction

Transition-based dependency parsing aims to model a transition sequence from an initial configuration to some terminal configuration, which results in a target dependency tree, as shown in Figure 1. Many traditional parsers suffer from poorly generalized features. Higher level features such as part of speech and word attributes have assisted performance, but almost all of these traditional parsers had relied on manually designed feature templates. [2] Introduced a parser which used (i) neural network architecture that gives good accuracy and speed, (ii) develops a neural network architecture that gives good accuracy and speed, and (iii) introduces the *cube* activation to capture higher-order interaction features. That project will re-implement their findings using bayesian neural networks to compare viability in training.

2 Transition-based dependency parsing

This project employs the arc-standard transistion systems. In the arc- standard system, a configuration $c = (s, b, A)$ relies on a stack s, a buffer b, and a set of dependency arcs A. The initial configuration for a sentence $w_1, ..., w_n$ is $s = [ROOT], b = [w_1, ..., w_n], A =$**null**. A configuration c is terminal if the buffer is empty and the stack contains the single node $ROOT$, and the parse tree is given by A_c. Denoting $s_i(i = 1, 2, ...)$ as the ith top element on the stack, and $b_i(i = 1, 2, ...)$ as the i^{th} element on the buffer, the arc-standard system defines three types of transitions:

- LEFT-ARC(l): adds an arc $s1 \rightarrow s2$ with label l and removes $s2$ from the stack. Pre-condition: $|s| \geq 2$.
- RIGHT-ARC(l): adds an arc $s2 \rightarrow s1$ with label l and removes $s1$ from the stack. Pre-condition: $|s| \geq 2$.
- SHIFT: moves $b1$ from the buffer to the stack. Precondition: $|b| \geq 1$.

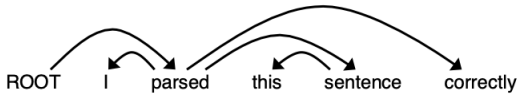
3 Neural Network Based Parser

The original paper uses sets S^w, S^t, S^t to denote the sentence *word, POS, label*. The paper builds a standard neural network with one hidden layer, where the corresponding embeddings are used to convert a value to a vector. To introduce higher-order feature interaction,

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$
$$\hat{y} = ReLU(hW_2 + b_2)$$

$$J(\theta) = CE(y, \hat{y}) = -\sum_{i=1}^{N_c} y_i log \hat{y}_i$$

The cube activation allows all three terms to interact when necessary.



stack	buffer	new dependency	transition
[ROOT]	[I, parsed, this, sentence, correctly]		Initial Configuration
[ROOT, I]	[parsed, this, sentence, correctly]		SHIFT
[ROOT, I, parsed]	[this, sentence, correctly]		SHIFT
[ROOT, parsed]	[this, sentence, correctly]	parsed→I	LEFT-ARC

Figure 1: The Arc-standard parsing scheme

4 Tests

Our classifier used embedding size $h = 20$, hidden layer size $h = 200$, the Adam optimizer with default parameters. I used the English Penn Treebank dataset for my experiments. I report unlabeled attachment scores (UAS). For the Bayesian Neural network setup, I use the **Edward** library with a Gaussian kernel for the weights and rely on **KLqp** (Variational inference with KL divergence). Since I experiment with at most 2 hidden layer neural networks, I only train for 10 epochs.

Table 1: Performance

BNN	**hidden activations**	**hidden layer config**	**UAS**
No	[Cube]	[200]	88
No	[Tanh]	[200]	84
No	[Cube, ReLU]	[200, 100]	87
No	[Cube, Tanh]	[200, 100]	85
Yes	[Cube]	[200]	89
Yes	[Tanh]	[200]	87
Yes	[Cube, ReLU]	[200, 100]	65
Yes	[Cube, Tanh]	[200, 100]	74

5 Discussion

Training using Edward significantly increased our training time. Because it is a harder training scheme, the bayesian neural network architecture could not generalize with 10 epochs. However, it is worth noting that the best performance was actually an identical configuration as [2] as a bayesian neural network. This hints that the Variational inference objective may be effective at dependency parsing.

The code for this project is availible at https://github.com/TarunSunkaraneni/bayesian-dependency-parsing.

References

[1] Timothy Dozat and Christopher Manning. Simpler but More Accurate Semantic Dependency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL, 2018.

[2] A fast and accurate dependency parser using neu- ral networks. In of the 2014 Con- ference on Empirical Methods in Natural Lan- guage Processing (EMNLP). pages 740–750. Chen, Danqi and Christopher D Manning. 2014.