Enhancing the Inside-Outside Recursive Neural Network Reranker for Dependency Parsing

Phong Le
Institute for Logic, Language and Computation
University of Amsterdam, the Netherlands
p.le@uva.nl

Abstract

We propose solutions to enhance the Inside-Outside Recursive Neural Network (IORNN) reranker of Le and Zuidema (2014). Replacing the original softmax function with a hierarchical softmax using a binary tree constructed by combining output of the Brown clustering algorithm and frequency-based Huffman codes, we significantly reduce the reranker's computational complexity. In addition, enriching contexts used in the reranker by adding subtrees rooted at (ancestors') cousin nodes, the accuracy is increased.

1 Introduction

Using neural networks for syntactic parsing has become popular recently, thanks to promising results that those neural-net-based parsers achieved. For constituent parsing, Socher et al. (2013) using a recursive neural network (RNN) got an F1 score close to the state-of-the-art on the Penn WSJ corpus. For dependency parsing, the inside-outside recursive neural net (IORNN) reranker proposed by Le and Zuidema (2014) is among the top systems, including the Chen and Manning (2014)'s extremely fast transition-based parser employing a traditional feed-forward neural network.

There are many reasons why neural-net-based systems perform very well. First, Bansal et al. (2014) showed that using word-embeddings can lead to significant improvement for dependency parsing. Interestingly, those neural-net-based systems can transparently and easily employ word-embeddings by initializing their networks with those vectors. Second, by comparing a count-based model with their neural-net-based model on perplexity, Le and Zuidema (2014) suggested that predicting with neural nets is an effective solution for the problem of data sparsity. Last but not least, as showed in the work of Socher and colleagues on RNNs, e.g. Socher et al. (2013), neural networks are capable of 'semantic transfer', which is essential for disambiguation.

We focus on how to enhance the IORNN reranker of Le and Zuidema (2014) by both reducing its computational complexity and increasing its accuracy. Although this reranker is among the top systems in accuracy, its computation is very costly due to its softmax function used to compute probabilities of generating tokens: all possible words in the vocabulary are taken into account. Solutions for this are to approximate the original softmax function by using a hierarchical softmax (Morin and Bengio, 2005), noise-contrastive estimation (Mnih and Teh, 2012), or factorization using classes (Mikolov et al., 2011). A cost of using those approximations is, however, drop of the system performance. To reduce the drop, we use a hierarchical softmax with a binary tree constructed by combining Brown clusters and Huffman codes.

We show that, thanks to the reranking framework and the IORNN's ability to overcome the problem of data sparsity, more complex contexts can be employed to generate tokens. We introduce a new type of contexts, named *full-history*. By employing both the hierarchical softmax and the new type of context, our new IORNN reranker has significantly lower complexity but higher accuracy than the original reranker.

2 The IORNN Reranker

We firstly introduce the IORNN reranker (Le and Zuidema, 2014).

Proceedings of the 14th International Conference on Parsing Technologies, pages 87–91,
Bilbao, Spain; July 22–24, 2015.

2.1 The ∞-order Generative Model

The reranker employs the generative model proposed by Eisner (1996). Intuitively, this model is top-down: starting with ROOT, we generate its left dependents and its right dependents. We then generate dependents for each ROOT's dependent. The generative process recursively continues until there is no dependent to generate. Formally, this model is described by the following formula

$$P(T(H)) = \prod_{l=1}^{L} P\left(H_l^L | \mathcal{C}_{H_l^L}\right) P\left(T(H_l^L)\right) \times \prod_{r=1}^{R} P\left(H_r^R | \mathcal{C}_{H_r^R}\right) P\left(T(H_r^R)\right) \quad (1)$$

where H is the current head, $T(N)$ is the subtree rooted at N, and $\mathcal{C}_N$ is the context to generate N. H^L, H^R are respectively H's left dependents and right dependents, plus EOC (End-Of-Children), a special token to inform that there are no more dependents to generate. Thus, $P(T(ROOT))$ is the probability of generating the entire T.

Le and Zuidema's ∞-order generative model is defined as the Eisner's model in which the context $\mathcal{C}_D^\infty$ to generate D contains *all* of D's generated siblings, its ancestors and theirs siblings. Because of very large fragments that contexts are allowed to hold, traditional count-based methods are impractical (even if we use smart smoothing techniques). They thus introduced the IORNN architecture to estimate the model.

2.2 Estimation with the IORNN

Each tree node y carries three vectors: inner representation $\mathbf{i}_y$, representing y, outer representation $\mathbf{o}_y$, representing the *full* context of y, and partial outer representation $\bar{\mathbf{o}}_y$, representing the *partial* context $\mathcal{C}_y^\infty$ which generates the token of y.

Without loss of generality and ignoring directions for simplicity, we assume that the model is generating dependent y for node h conditioning on context $\mathcal{C}_y^\infty$ (see Figure 1). Under the approximation that the inner representation of a phrase equals the inner representation of its head, and thanks to the recursive definition of full/partial contexts ($\mathcal{C}_y^\infty$ is a combination of $\mathcal{C}_h^\infty$ and y's previously generated sisters), the (partial) outer representations of y are computed as follows.

$$\bar{\mathbf{o}}_y = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{b}_o + \mathbf{r})$$

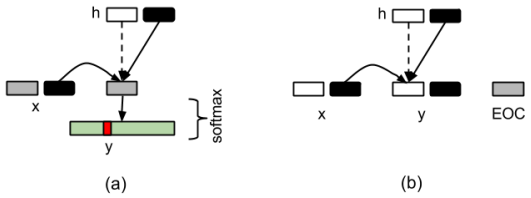

Figure 1: The process to (a) generate y, (b) compute outer representation $\mathbf{o}_y$, given head h and sibling x. Black, grey, white boxes are respectively inner, partial outer, and outer representations. (Le and Zuidema, 2014)

where $\mathbf{r} = \mathbf{0}$ if y is the first dependent of h; otherwise, $\mathbf{r} = \frac{1}{|\bar{\mathcal{S}}(y)|}\sum_{v\in\bar{\mathcal{S}}(y)} \mathbf{W}_{dr(v)}\mathbf{i}_v$, where $\bar{\mathcal{S}}(y)$ is the set of y's sisters generated before. And,

$$\mathbf{o}_y = f(\mathbf{W}_{hi}\mathbf{i}_h + \mathbf{W}_{ho}\mathbf{o}_h + \mathbf{b}_o + \mathbf{s})$$

where $\mathbf{s} = \mathbf{0}$ if y is the only dependent of h (ignoring EOC); otherwise $\mathbf{s} = \frac{1}{|\mathcal{S}(y)|}\sum_{v\in\mathcal{S}(y)} \mathbf{W}_{dr(v)}\mathbf{i}_v$ where $\mathcal{S}(y)$ is the set of y's sisters. $dr(v)$ is the dependency relation of v with h. $\mathbf{W}_{hi/ho/dr(v)}$ are $n \times n$ real matrices, and $\mathbf{b}_o$ is an n-d real vector.

The probability $P(w|\mathcal{C}_y^\infty)$ of generating a token w at node y is given by

$$softmax(w, \bar{\mathbf{o}}_y) = \frac{e^{u(w,\bar{\mathbf{o}}_y)}}{\sum_{w'\in V} e^{u(w',\bar{\mathbf{o}}_y)}} \quad (2)$$

where $\left[u(w_1, \bar{\mathbf{o}}_y), ..., u(w_{|V|}, \bar{\mathbf{o}}_y)\right]^T = \mathbf{W}\bar{\mathbf{o}}_y + \mathbf{b}$ and V is the set of all possible tokens (i.e. vocabulary). $\mathbf{W}$ is a $|V| \times n$ real matrix, $\mathbf{b}$ is an $|V|$-d real vector.

2.3 The Reranker

Le and Zuidema's (mixture) reranker is

$$T^* = \arg\max_{T\in\mathcal{D}(S)} \alpha \log P(T(ROOT)) + (1-\alpha)s(S,T) \quad (3)$$

where $\mathcal{D}(S)$ and $s(S,T)$ are a k-best list and scores given by a third-party parser, and $\alpha \in [0,1]$.

3 Reduce Complexity

The complexity of the IORNN reranker above for computing $P(T(ROOT))$ is approximately[1]

$$O = l \times (3 \times n \times n + n \times |V|)$$

[1]Look at Figure 1, we can see that each node requires four matrix-vector multiplications: two for computing children's (partial) outer representation, one for computing sisters' (partial) outer representations, and one for computing the softmax.

where l is the length of the given sentence, n and $|V|$ are respectively the dimensions of representations and the vocabulary size (sums of vectors are ignored because their computational cost is small w.r.t $l \times n \times n$). In Le and Zuidema's reranker, $|V| \approx 14000 \gg n = 200$. It means that the reranker spends most of its time on computing $softmax(w, \bar{\mathbf{o}}_y)$ in Equation 2. This is also true for the complexity in the training phase.

To reduce the reranker's complexity, we need to approximate this softmax. Mnih and Teh (2012) propose using the noise-contrastive estimation method which is to force the system to discriminate correct words from randomly chosen candidates (i.e., noise). This approach is very fast in training thanks to fixing normalization factors to one, but slow in testing because normalization factors are explicitly computed. Vaswani et al. (2013) use the same approach, and also fix normalization factors to one when testing. This, however, doest not guarantee to give us properly normalized probabilities. We thus employ the hierarchical sofmax proposed by Morin and Bengio (2005) which is fast in both training and testing and outputs properly normalized probabilities.

Assume that there is a binary tree whose leaf nodes each correspond to a word in the vocabulary. Let $(u_1^w, u_2^w, ..., u_L^w)$ be a path from the root to the leaf node w (i.e. u_1^w = root and $u_L^w = w$). Let $L(u)$ the left child of node u, and $[x]$ be 1 if x true and -1 otherwise. We then replace Equation 2 by

$$P(w|\mathcal{C}_y^\infty) = \prod_{i=1}^{L-1} \sigma\big([u_{i+1}^w = L(u_i^w)]\mathbf{v}_{u_i^w}^T \times \bar{\mathbf{o}}_y\big)$$

where $\sigma(z) = 1/(1+e^{-z})$. If the binary tree is perfectly balanced, the new complexity is approximately $l \times (3 \times n \times n + n \times \log(|V|))$, which is less than $4l \times n \times n$ if $|V| < 2^n$ (1.6×10^{60} as $n = 200$ in the Le and Zuidema's reranker).

Constructing a binary tree for this hierarchical softmax turns out to be nontrivial. Morin and Bengio (2005) relied on WordNet whereas Mikolov et al. (2013) used only frequency-based Huffmann codes. In our case, an ideal tree should reflect both semantic similarities between words (e.g. leaf nodes for 'dog' and 'cat' should be close to each other), and word frequencies (since we want to minimize the complexity). Therefore we propose combining output of the Brown hierarchical clustering algorithm (Brown et al., 1992) and frequency-based Huffman codes.[2] Firstly, we use the Brown algorithm to find c hierarchical clusters ($c = 500$ in our experiments).[3] We then, for each cluster, compute the Huffman code for each word in that cluster.

4 Enrich Contexts

Although suggesting that predicting with neural networks is a solution to overcome the problem of sparsity, Le and Zuidema's reranker still relies on two widely-used independence assumptions: (i) the two Markov chains that generate dependents in the two directions are independent, given the head, and (ii) non-overlapping subtrees are generated independently.[4] That is why its partial context (e.g. the red-dashed shape in Figure 2) used to generate a node ignores: (i) sisters in the other direction and (ii) ancestors' cousins and their descendants.

We, in contrast, eliminate those two assumptions by proposing the following top-down left-to-right generative story. From the head node, we generate its dependents from left to right. The partial context to generate a dependent is the whole fragment that is generated so far (e.g. the blue shape in Figure 2). We then generate subtrees rooted at those nodes also from left to right. The full context given to a node to generate the subtree rooted at this node is thus the whole fragment that is generated so far (e.g. the combination of the blue shape and the blue-dotted shape in Figure 2). In this way, the model always uses the whole up-to-date fragment to generate a dependent or to generate a subtree rooted at a node. To our knowledge, these contexts, which contain full derivation histories, are the most complete ones ever used for graph-based parsing.

Extending the IORNN reranker in this way is straight-forward. For example, we first generate a subtree $tr(x)$ rooted at node x in Figure 3. We then compute the inner representation for $tr(x)$: if $tr(x)$ contains only x then $\mathbf{i}_{tr(x)} = \mathbf{i}_x$; otherwise

$$\mathbf{i}_{tr(x)} = f(\mathbf{W}_h^i \mathbf{i}_x + \frac{1}{|\mathcal{S}(x)|} \sum_{u \in \mathcal{S}(x)} \mathbf{W}_{dr(u)}^i \mathbf{i}_{tr(u)} + \mathbf{b}^i)$$

[2] Another reason not to use the Brown clustering algorithm alone is that the algorithm is inefficient with high numbers of clusters ($|V| \approx 14000$ in this case).

[3] We run Liang (2005)'s implementation at `https://github.com/percyliang/brown-cluster` on the training data.

[4] Le and Zuidema rephrased this assumption by the approximation that the meaning of a phrase equals to the meaning of its head.

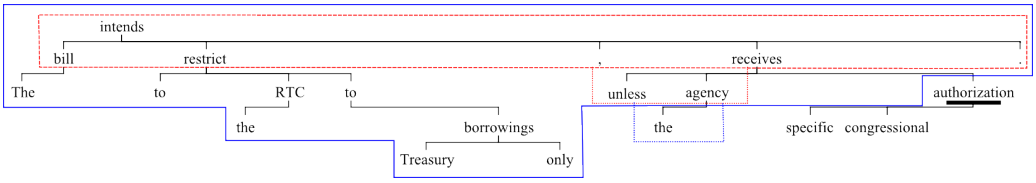

Figure 2: Context used in Le and Zuidema's reranker (red-dashed shape) and full-history context (blue-solid shape) to generate token 'authorization'.

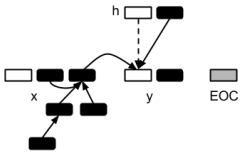

Figure 3: Compute the full-history outer representation for y.

where $\mathcal{S}(x)$ is the set of x's dependents, $dr(u)$ is the dependency relation of u with x, $\mathbf{W}^i_{h/dr(u)}$ are $n \times n$ real matrices, and $\mathbf{b}^i$ is an n-d real vector.[5]

5 Experiments

We use the same setting reported in Le and Zuidema (2014, Section 5.3). The Penn WSJ Treebank is converted to dependencies using the Yamada and Matsumoto (2003)'s head rules. Sections 2-21 are for training, section 22 for development, and section 23 for testing. The development and test sets are tagged by the Stanford POS-tagger trained on the whole training data, whereas 10-way jackknifing is used to generate tags for the training set. For the new IORNN reranker, we set $n = 200$, initialise it with the 50-dim word embeddings from Collobert et al. (2011). We use the MSTParser (with the 2nd-order feature mode) (McDonald et al., 2005) to generate k-best lists, and optimize k and α (Equation 3) on the development set.

Table 1 shows the comparison of our new reranker against other systems. It is a surprise that our reranker with the proposed hierarchical softmax alone can achieve an almost equivalent score with Le and Zuidema's reranker. We conjecture that drawbacks of the hierarchical softmax compared to the original can be lessened by probabilities of generating other elements like POS-tags, dependency relations. Adding enriched contexts, our reranker achieves the second best accuracy among those systems.

System	UAS
MSTParser (baseline)	92.06
Koo and Collins (2010)	93.04
Zhang and McDonald (2012)	93.06
Martins et al. (2013)	93.07
Bohnet and Kuhn (2012)	93.39
Reranking	
Hayashi et al. (2013)	93.12
Le and Zuidema (2014)	93.12
Our reranker (h-softmax only, $k = 45$)	93.10
Our reranker ($k = 47$)	**93.27**

Table 1: Comparison with other systems on section 23 (excluding punctuation).

Because in this experiment no words have paths longer than $20 \ll n = 200$, our new reranker has a significantly lower complexity than the one of Le and Zuidema's reranker. On a computer with an Intel Core-i5 3.3GHz CPU and 8GB RAM, it takes 20 minutes to train this reranker, which is implemented in C++, and 2 minutes to evaluate it on the test set.

6 Conclusion

Solutions to enhance the IORNN reranker of Le and Zuidema (2014) were proposed. We showed that, by replacing the original softmax function with a hierarchical softmax, the reranker's computational complexity significantly decreases. The cost of this, which is drop on accuracy, is avoided by enriching contexts with subtrees rooted at (ancestors') cousin nodes. The new reranker, according to experimental results on the Penn WSJ Treebank, has even higher accuracy than the old one.

Acknowledgments

We thank Willem Zuidema and four anonymous reviewers for helpful comments.

[5] Note that the overall computational complexity increases linearly with $l \times n \times n$. For instance, for computing $P(T(ROOT))$, the increase is approximately $2l \times n \times n$ since each node requires maximally two more matrix-vector multiplications.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds: a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87. Association for Computational Linguistics.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Katsuhiko Hayashi, Shuhei Kondo, and Yuji Matsumoto. 2013. Efficient stacked dependency parsing by forest reranking. *Transactions of the Association for Computational Linguistics*, 1(1):139–150.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Percy Liang. 2005. Semi-supervised learning for natural language. In *MASTERS THESIS, MIT*.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *AISTATS*, volume 5, pages 246–252. Citeseer.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*, pages 1387–1392. Citeseer.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of International Conference on Parsing Technologies (IWPT)*, pages 195–206.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331. Association for Computational Linguistics.