NSHRDLU?

Samuel R. Bowman[*†]
sbowman@stanford.edu

Jon Gauthier[†‡]
angeli@stanford.edu

Christopher Potts[*]
cgpotts@stanford.edu

Christopher D. Manning[*†§]
manning@stanford.edu

[*]Stanford Linguistics
[†]Stanford NLP Group
[‡]Stanford Symbolic Systems
[§]Stanford Computer Science

This project aims to build neural network models that can jointly learn to parse sentences and to use those parses to guide semantic composition.

Table 1 shows the sequence of model designs that we plan to build.

1 Models

1.1 Model 0

Model 0 in its simplest form computes the same function as a plain TreeRNN, but uses a stack and a recurrent input reader instead of a tree as its graph structure. **TODO:** Model 0 figure. We expect it to be substantially faster than conventional TreeRNN implementations, since it readily supports batched matrix–vector multiplications, allowing it to fast CPU libraries and to function well on GPUs.

1.2 Model 1

Model 1 adapts Model 0 to use a stack and a buffer, making it more closely resemble a shift–reduce parser, and laying the groundwork for a model which can parse novel sentences at test time. The structre of the model is shown in Figure 1.

The model runs for a fixed number of transition steps: $2N - 3$. In its starting configuration, it contains a stack that is prepopulated with the first two words of the sentence (since SHIFTSHIFTis the only legal operation sequence for the first two timesteps of a true shift-reduce parser), as well as a buffer (a queue) prepopulated with all of the remaining words in the sentence. Both the stack and buffer represent words using their embeddings.

At each timestep at test time, the model combines views of the stack and buffer (the top element of the buffer and the top two elements of the stack, highlighted in yellow) as the input to a tracking LSTM (red). This LSTM's output is fed into a sigmoid operation classifier (blue) which chooses between the SHIFTand REDUCEoperations. If SHIFTis chosen, one word embedding is popped from the buffer and pushed onto the stack. If REDUCEis chosen, the buffer is left as is, and the top two elements of the stack are popped and composed using a learned composition function (green), with the result placed back on top of the stack.

Supervision The model is trained using two objective functions simultaneously. The semantic objective function is computed by feeding the value from the top of the stack at the final timestep—the full sentence encoding—into a downstream neural network model for some semantic task, like a sentiment classifier or an entailment classifier. The gradients from that classifier propagate to every part of the model except the operation classifier (blue). The syntactic objective function takes the form of direct supervision on the operation classifier (blue) which encourages that classifier to produce the same sequence of operations that an existing parser would produce for that sentence. The gradients from the syntactic objective function propagate to every part of the model but the downstream semantic model.

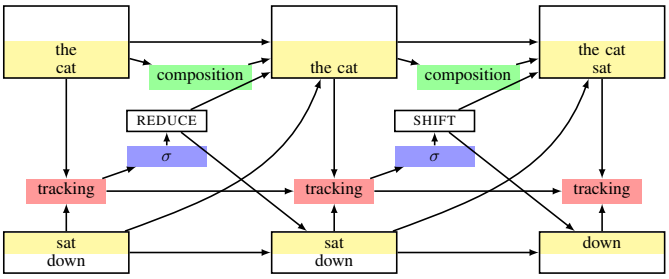
At training time, following the strategy used in LSTM text decoders, the decisions made by the operation classifier (blue) is discarded, and the model instead uses the correct operation as specified in the (already parsed) training corpus. At test time, this signal is not available, and the model uses its own predicted operations.

1.3 Model 2

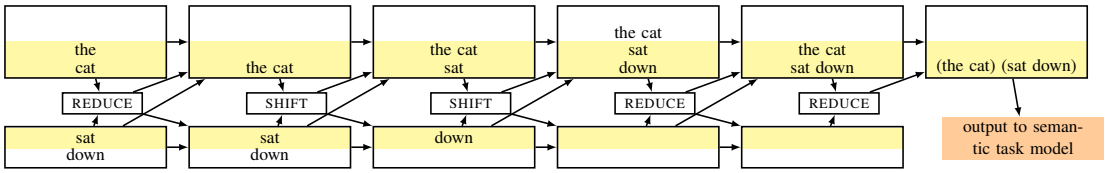
Model 2 makes a small change to Model 1 that is likely to substantially change the dymanics of learning: It uses the operation sequence predicted by the operation classifier (blue) at training time as well as at test time. It may be possible to accelerate Model 2 training by initializing it with parameters learned by Model 1.

Name	Stack Representation	Input Representation	Ops Classifier	Op Predictions Used in Training
Model 0	Discrete	Op. sequence	N	–
Model 1	Discrete	**Discrete Buffer**	**Y: Directly supervised**	**N**
Model 2	Discrete	Discrete Buffer	Y: Directly supervised	**Y**
Model 3	**Soft**	**Soft Buffer**	Y: Directly supervised	Y
Model 4	Soft	Soft Buffer	**Y: Indirectly supervised**	Y

Table 1: Model variants, ordered by increasing reliance on learning. Bolding indicates the differences between each model and its parent model.



(a) The Model 1/2 network unrolled for two transitions on the input *the cat sat down*.



(b) The fully unrolled Model 1/2 network for *the cat sat down* with some layers omitted for clarity.

Figure 1: Two views of Models 1 and 2 (which use equivalent model graphs). In both views, the lower boxes represent the input buffer, and the upper boxes represent the stack. Yellow highlighting indicates which portions of these data structures are visible to the tracking LSTM and to the composition function.

1.4 Model 3

Model 3 modifies Model 2 by introducing the soft stack/soft queue from (Grefenstette et al., 2015) in place of the hard, conventional stack and buffer. The soft stack makes it possible to for the model to predict smooth distributions over operations of the form (0.93 SHIFT, 0.07 REDUCE), instead of making hard decisions. These soft decisions allow for gradient information to flow from the stack and the buffer back into the operation classifier (blue). This is crucial to our ultimate goal, as it makes it possible for semantic considerations to influence the model's parsing decisions.

1.5 Model 4

Model 4 modifies Model 3 by removing the direct supervision signal from the operation classifier (blue), instead forcing the operation classifier to learn solely from the gradient provided by the downstream supervision task. It may be possible to accelerate or otherwise improve Model 4 training by initializing it with parameters learned by Model 3.

2 Other possible model features

2.1 Encoding the contents of the stack and buffer

The tracking LSTM (red) needs access to the top of the buffer and the top two elements of the stack in order to make even minimally informed decisions about whether to shift or reduce. It could benefit further from additional information about broader sentential context. This can be provided by running new LSTMs along the elements of each of the stack and the buffer (following Dyer et al. 2015) and feeding the result into the tracking LSTM.

2.2 Contextually-informed composition

The composition function in the basic model (green) combines only the top elements of the stack, without using any further information. It

may be possible to encourage the composition function to learn to do some amount of context-sensitive interpretation/disambiguation by adding a connection from the tracking LSTM (red) directly into the composition function.

2.3 Constituent labels

It would be possible to train any of the parse-supervised models (1–3) to learn explicit part of speech operations, expanding the op set dramatically to something like {SHIFT, REDUCE-NP, REDUCE-S, REDUCE-PP, ...}.

3 Implementation notes

The size of the stack should be N for sentences of N words, in case the first reduce merges the final two words. The size of the buffer should be $N-2$.

Acknowledgments

References

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. *arXiv preprint arXiv:1506.02516*.