Incremental Parsing with Minimal Features Using Bi-Directional LSTM

James Cross and **Liang Huang**
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon, USA
{crossj,liang.huang}@oregonstate.edu

Abstract

Recently, neural network approaches for parsing have largely automated the combination of individual features, but still rely on (often a larger number of) atomic features created from human linguistic intuition, and potentially omitting important global context. To further reduce feature engineering to the bare minimum, we use bi-directional LSTM sentence representations to model a parser state with only three sentence positions, which automatically identifies important aspects of the entire sentence. This model achieves state-of-the-art results among greedy dependency parsers for English. We also introduce a novel transition system for constituency parsing which does not require binarization, and together with the above architecture, achieves state-of-the-art results among greedy parsers for both English and Chinese.

1 Introduction

Recently, neural network-based parsers have become popular, with the promise of reducing the burden of manual feature engineering. For example, Chen and Manning (2014) and subsequent work replace the huge amount of manual feature combinations in non-neural network efforts (Nivre et al., 2006; Zhang and Nivre, 2011) by vector embeddings of the atomic features. However, this approach has two related limitations. First, it still depends on a large number of carefully designed atomic features. For example, Chen and Manning (2014) and subsequent work such as Weiss et al. (2015) use 48 atomic features from Zhang and Nivre (2011), including select third-order dependencies. More importantly, this approach inevitably leaves out some nonlocal information which could be useful. In particular, though such a model can exploit similarities between words and other embedded categories, and learn interactions among those atomic features, it cannot exploit any other details of the text.

We aim to reduce the need for manual induction of atomic features to the bare minimum, by using bi-directional recurrent neural networks to automatically learn context-sensitive representations for each word in the sentence. This approach allows the model to learn arbitrary patterns from the entire sentence, effectively extending the generalization power of embedding individual words to longer sequences. Since such a feature representation is less dependent on earlier parser decisions, it is also more resilient to local mistakes.

With just three positional features we can build a greedy shift-reduce dependency parser that is on par with the most accurate parser in the published literature for English Treebank. This effort is similar in motivation to the stack-LSTM of Dyer et al. (2015), but uses a much simpler architecture.

We also extend this model to predict phrase-structure trees with a novel shift-promote-adjoin system tailored to greedy constituency parsing, and with just two more positional features (defining tree span) and nonterminal label embeddings we achieve the most accurate greedy constituency parser for both English and Chinese.

2 LSTM Position Features

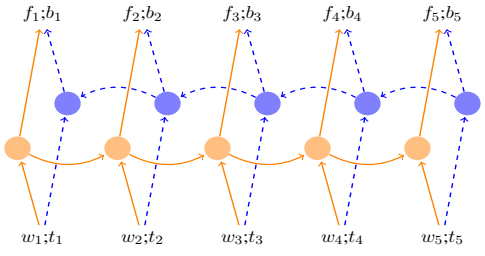


Figure 1: The sentence is modeled with an LSTM in each direction whose input vectors at each time step are word and part-of-speech tag embeddings.

The central idea behind this approach is exploiting the power of recurrent neural networks to let the model decide what apsects of sentence context are important to making parsing decisions, rather than relying on fallible linguistic information (which moreover requires leaving out information which could be useful). In particular, we model an input sentence using Long Short-Term Memory networks (LSTM), which have made a recent resurgence after being initially formulated by Hochreiter and Schmidhuber (1997).

The input at each time step is simply a vector representing the word, in this case an embedding for the word form and one for the part-of-speech tag. These embeddings are learned from random initialization together with other network parameters in this work. In our initial experiments, we used one LSTM layer in each direction (forward and backward), and then concatenate the output at each time step to represent that sentence position: that word in the entire context of the sentence. This network is illustrated in Figure 1.

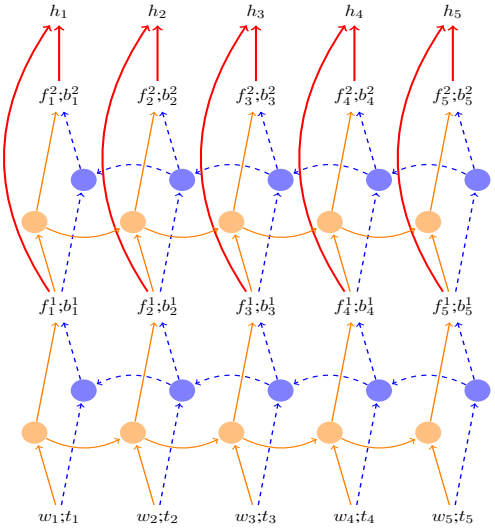


Figure 2: In the 2-Layer architecture, the output of each LSTM layer is concatenated to create the positional feature vector.

It is also common to stack multiple such LSTM layers, where the output of the forward and backward networks at one layer are concatenated to form the input to the next. We found that parsing performance could be improved by using two bidirectional LSTM layers in this manner, and concatenating the output of both layers as the positional feature representation, which becomes the input to the fully-connected layer. This architecture is shown in Figure 2.

Intuitively, this represents the sentence position by the word in the context of the sentence up to that point and the sentence after that point in the first layer, as well as modeling the "higher-order" interactions between parts of the sentence in the second layer. In Section 5 we report results using only one LSTM layer ("Bi-LSTM") as well as with two layers where output from each layer is used as part of the positional feature ("2-Layer Bi-LSTM").

input:	$w_0 \dots w_{n-1}$	
axiom	$\langle \epsilon,\ 0 \rangle : \emptyset$	
shift	$\dfrac{\langle S,\ j \rangle : A}{\langle S \vert j,\ j+1 \rangle : A}$	$j < n$
$\mathsf{re}_{\curvearrowleft}$	$\dfrac{\langle S \vert s_1 \vert s_0,\ j \rangle : A}{\langle S \vert s_0,\ j \rangle : A \cup \{s_1 \curvearrowleft s_0\}}$	
goal	$\langle s_0,\ n \rangle : A$	

Figure 3: The arc-standard dependency parsing system (Nivre, 2008) ($\mathsf{re}_{\curvearrowleft}$ omitted). Stack S is a list of heads, j is the start index of the queue, and s_0 and s_1 are the top two head indices on S.

	dependency	constituency
positional	s_1, s_0, q_0	$s_1, s_0, q_0, s_1.\mathsf{left}, s_0.\mathsf{left}$
labels	-	$s_0.\{\mathsf{left}, \mathsf{right}, \mathsf{root}, \mathsf{head}\}$ $s_1.\{\mathsf{left}, \mathsf{right}, \mathsf{root}, \mathsf{head}\}$

Table 1: Feature templates. Note that, remarkably, even though we do labeled dependency parsing, we do *not* include arc label as features.

3 Shift-Reduce Dependency Parsing

We use the arc-standard system for dependency parsing (see Figure 4). By exploiting the LSTM architecture to encode context, we found that we were able to achieve competitive results using only three sentence-position features to model parser state: the head word of each of the top two trees on the stack (s_0 and s_1), and the next word on the queue (q_0); see Table 1.

The usefulness of the head words on the stack is clear enough, since those are the two words that are linked by a dependency when taking a reduce action. The next incoming word on the queue is also important because the top tree on the stack should not be reduced if it still has children which have not yet been shifted. That feature thus allows

$$
\begin{array}{ll}
\text{input:} & w_0 \ldots w_{n-1} \\
\text{axiom} & \langle \epsilon,\ 0 \rangle \colon \emptyset \\
\textsf{shift} & \dfrac{\langle S,\ j \rangle}{\langle S \mid j,\ j+1 \rangle} \quad j < n \\
\textsf{pro}(X) & \dfrac{\langle S \mid t,\ j \rangle}{\langle S \mid X(t),\ j \rangle} \\
\textsf{adj}_{\curvearrowright} & \dfrac{\langle S \mid t \mid X(t_1 ... t_k),\ j \rangle}{\langle S \mid X(t, t_1 ... t_k),\ j \rangle} \\
\text{goal} & \langle s_0,\ n \rangle
\end{array}
$$

Figure 4: Our shift-promote-adjoin system for constituency parsing ($\textsf{adj}_{\curvearrowright}$ omitted).

the model to learn to delay a right-reduce until the top tree on the stack is fully formed, shifting instead.

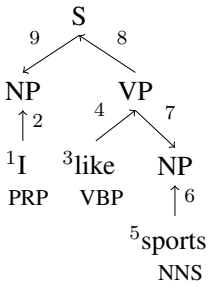
3.1 Hierarchical Classification

The structure of our network model after computing positional features is fairly straightforward and similar to previous neural-network parsing approaches such as Chen and Manning (2014) and Weiss et al. (2015). It consists of a multilayer perceptron using a single ReLU hidden layer followed by a linear classifier over the action space, with the training objective being negative log softmax.

We found that performance could be improved, however, by factoring out the decision over structural actions (i.e., shift, left-reduce, or right-reduce) and the decision of which arc label to assign upon a reduce. We therefore use separate classifiers for those decisions, each with its own fully-connected hidden and output layers but sharing the underlying recurrent architecture. This structure was used for the results reported in Section 5, and it is referred to as "Hierarchical Actions" when compared against a single action classifier in Table 3.

4 Shift-Promote-Adjoin Constituency Parsing

To further demonstrate the advantage of our idea of minimal features with bidirectional sentence representations, we extend our work from dependency parsing to constituency parsing. However, the latter is significantly more challenging than the former under the shift-reduce paradigm because:



[1]shift (I)	[6]pro (NP)
[2]pro (NP)	[7]$\textsf{adj}_{\curvearrowright}$
[3]shift (like)	[8]pro (S)
[4]pro (VP)	[9]$\textsf{adj}_{\curvearrowright}$
[5]shift (sports)	

Figure 5: Shift-Promote-Adjoin parsing example. Upward and downward arrows indicate promote and (sister-)adjunction actions, respectively.

- we also need to predict the nonterminal labels
- the tree is not binarized (with many unary rules and more than binary branching rules)

While most previous work binarizes the constituency tree in a preprocessing step (Zhu et al., 2013; Wang and Xue, 2014; Mi and Huang, 2015), we propose a novel "Shift-Promote-Adjoin" paradigm which does not require any binariziation or transformation of constituency trees (see Figure 5). Note in particular that, in our case only the Promote action produces a new tree node (with a non-terminal label), while the Adjoin action is the linguistically-motivated "sister-adjunction" operation, i.e., attachment (Chiang, 2000; Henderson, 2003). By comparison, in previous work, both Unary-X and Reduce-L/R-X actions produce new labeled nodes (some of which are auxiliary nodes due to binarization). Thus our paradigm has two advantages:

- it dramatically reduces the number of possible actions, from $3X + 1$ or more in previous work to $3 + X$, where X is the number of nonterminal labels, which we argue would simplify learning;
- it does not require binarization (Zhu et al., 2013; Wang and Xue, 2014) or compression of unary chains (Mi and Huang, 2015)

There is, however, a more closely-related "shift-project-attach" paradigm by Henderson (2003). For the example in Figure 5 he would use the following actions:

> shift(I), project(NP), project(S), shift(like), project(VP), shift(sports), project(NP), attach, attach.

The differences are twofold: first, our Promote action is head-driven, which means we only promote the head child (e.g., VP to S) whereas his Project action promotes the *first* child (e.g., NP to S); and secondly, as a result, his Attach action is always right-attach whereas our Adjoin action could be either left or right. The advantage of our method is its close resemblance to shift-reduce dependency parsing, which means that our constituency parser is jointly performing both tasks and can produce both kinds of trees. This also means that we use head rules to determine the correct order of gold actions.

We found that in this setting, we did need slightly more input features. As mentioned, node labels are necessary to distinguish whether a tree has been sufficiently promoted, and are helpful in any case. We used 8 labels: the current and immediate predecessor label of each of the top two stacks on the tree, as well as the label of the left- and rightmost adjoined child for each tree. We also found it helped to add positional features for the leftmost word in the span for each of those trees, bringing the total number of positional features to five. See Table 1 for details.

5 Experimental Results

We report both dependency and constituency parsing results on both English and Chinese.

All experiments were conducted with minimal hyperparameter tuning. The settings used for the reported results are summarized in Table 6. Networks parameters were updated using gradient backpropagation, including backpropagation through time for the recurrent components, using ADADELTA for learning rate scheduling (Zeiler, 2012). We also applied dropout (Hinton et al., 2012) (with $p = 0.5$) to the output of each LSTM layer (separately for each connection in the case of the two-layer network).

We tested both types of parser on the Penn Treebank (PTB) and Penn Chinese Treebank (CTB-5), with the standard splits for each of training, development, and test sets. Automatically predicted part of speech tags with 10-way jackknifing were used as inputs for all tasks except for Chinese dependency parsing, where we used gold tags, following the traditions in literature.

5.1 Dependency Parsing: English & Chinese

Table 2 shows results for English Penn Treebank using Stanford dependencies. Despite the minimally designed feature representation, relatively few training iterations, and lack of precomputed embeddings, the parser performed on par with state-of-the-art incremental dependency parsers, and slightly outperformed the state-of-the-art greedy parser.

The ablation experiments shown in the Table 3 indicate that both forward and backward contexts for each word are very important to obtain strong results. Using only word forms and no part-of-speech input similarly degraded performance.

Parser	Dev		Test	
	UAS	LAS	UAS	LAS
C & M 2014	92.0	89.7	91.8	89.6
Dyer et al. 2015	93.2	90.9	93.1	90.9
Weiss et al. 2015	-	-	93.19	91.18
+ Percept./Beam	-	-	93.99	92.05
Bi-LSTM	93.31	91.01	93.21	91.16
2-Layer Bi-LSTM	93.67	91.48	93.42	91.36

Table 2: Development and test set results for shift-reduce dependency parser on Penn Treebank using only (s_1, s_0, q_0) positional features.

Parser	UAS	LAS
Bi-LSTM Hierarchical†	93.31	91.01
† - Hierarchical Actions	92.94	90.96
† - Backward-LSTM	91.12	88.72
† - Forward-LSTM	91.85	88.39
† - tag embeddings	92.46	89.81

Table 3: Ablation studies on PTB dev set (wsj 22). Forward and backward context, and part-of-speech input were all critical to strong performace.

Figure 6 compares our parser with that of Chen and Manning (2014) in terms of arc recall for various arc lengths. While the two parsers perform similarly on short arcs, ours significantly outpeforms theirs on longer arcs, and more interestingly our accuracy does not degrade much after length 6. This confirms the benefit of having a global sentence repesentation in our model.

Table 4 summarizes the Chinese dependency parsing results. Again, our work is competitive with the state-of-the-art greedy parsers.

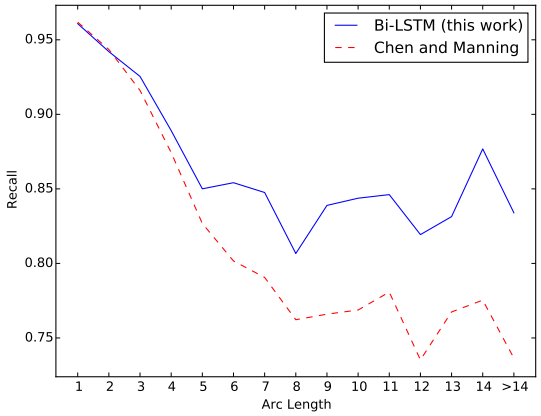


Figure 6: Recall on dependency arcs of various lengths in PTB dev set. The Bi-LSTM parser is particularly good at predicting longer arcs.

Parser	Dev		Test	
	UAS	LAS	UAS	LAS
C & M 2014	84.0	82.4	83.9	82.4
Dyer et al. 2015	87.2	85.9	87.2	85.7
Bi-LSTM	85.84	85.24	85.53	84.89
2-Layer Bi-LSTM	86.13	85.51	86.35	85.71

Table 4: Development and test set results for shift-reduce dependency parser on Penn Chinese Treebank (CTB-5) using only (s_1, s_0, q_0) position features (trained and tested with gold POS tags).

5.2 Constituency Parsing: English & Chinese

Table 5 compares our constituency parsing results with state-of-the-art incremental parsers. Although our work are definitely less accurate than those beam-search parsers, we achieve the highest accuracy among greedy parsers, for both English and Chinese.[1,2]

Parser	b	English		Chinese	
		greedy	beam	greedy	beam
Zhu et al. (2013)	16	86.08	90.4	75.99	85.6
Mi & Huang (05)	32	84.95	90.8	75.61	83.9
Vinyals et al. (05)	10	-	90.5	-	-
Bi-LSTM	-	89.75	-	79.44	-
2-Layer Bi-LSTM	-	**89.95**	-	**80.13**	-

Table 5: Test F-scores for constituency parsing on Penn Treebank and CTB-5.

[1]The greedy accuracies for Mi and Huang (2015) are from Haitao Mi, and greedy results for Zhu et al. (2013) come from duplicating experiments with code provided by those authors.

[2]The parser of Vinyals et al. (2015) does not use an explicit transition system, but is similar in spirit since generating a right bracket can be viewed as a reduce action.

	Dependency	Constituency
Embeddings		
Word (dims)	50	100
Tags (dims)	20	100
Nonterminals (dims)	-	100
Pretrained	No	No
Network details		
LSTM units (each direction)	200	200
ReLU hidden units	200 / decision	1000
Training		
Training epochs	10	10
Minibatch size (sentences)	10	10
Dropout (LSTM output only)	0.5	0.5
L2 penalty (all weights)	none	1×10^{-8}
ADADELTA ρ	0.99	0.99
ADADELTA ϵ	1×10^{-7}	1×10^{-7}

Table 6: Hyperparameters and training settings.

6 Related Work

Because recurrent networks are such a natural fit for modeling languages (given the sequential nature of the latter), bi-directional LSTM networks are becoming increasingly common in all sorts of linguistic tasks, for example event detection in Ghaeini et al. (2016). In fact, we discovered after submission that Kiperwasser and Goldberg (2016) have concurrently developed an extremely similar approach to our dependency parser. Instead of extending it to constituency parsing, they also apply the same idea to graph-based dependency parsing.

7 Conclusions

We have presented a simple bi-directional LSTM sentence representation model for minimal features in both incremental dependency and incremental constituency parsing, the latter using a novel shift-promote-adjoin algorithm. Experiments show that our method are competitive with the state-of-the-art greedy parsers on both parsing tasks and on both English and Chinese.

Acknowledgments

We thank the anonymous reviewers for comments. We also thank Taro Watanabe, Muhua Zhu, and Yue Zhang for sharing their code, Haitao Mi for producing greedy results from his parser, and Ashish Vaswani and Yoav Goldberg for discussions. The authors were supported in part by DARPA FA8750-13-2-0041 (DEFT), NSF IIS-1449278, and a Google Faculty Research Award.

References

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree-adjoining grammar. In *Proc. of ACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Reza Ghaeini, Xiaoli Z. Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proc. of ACL*.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of NAACL*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351.

Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2755–2763.

Zhiguo Wang and Nianwen Xue. 2014. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of ACL*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL*, pages 188–193.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of ACL 2013*.