Parsing as Language Modeling

Do Kook Choe
Brown University
Providence, RI
dc65@cs.brown.edu

Eugene Charniak
Brown University
Providence, RI
ec@cs.brown.edu

Abstract

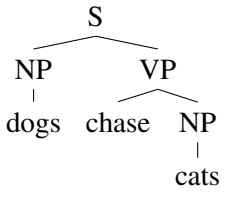
We recast syntactic parsing as a language modeling problem and use recent advances in neural network language modeling to achieve a new state of the art for constituency Penn Treebank parsing — 93.8 F_1 on section 23, using 2-21 as training, 24 as development, plus tri-training. When trees are converted to Stanford dependencies, UAS and LAS are 95.9% and 94.1%.

1 Introduction

Recent work on deep learning syntactic parsing models has achieved notably good results, e.g., Dyer et al. (2016) with 92.4 F_1 on Penn Treebank constituency parsing and Vinyals et al. (2015) with 92.8 F_1. In this paper we borrow from the approaches of both of these works and present a neural-net parse reranker that achieves very good results, 93.8 F_1, with a comparatively simple architecture.

In the remainder of this section we outline the major difference between this and previous work — viewing parsing as a language modeling problem. Section 2 looks more closely at three of the most relevant previous papers. We then describe our exact model (Section 3), followed by the experimental setup and results (Sections 4 and 5).

(a)



(b)

(S (NP dogs)$_{\mathrm{NP}}$ (VP chase (NP cats)$_{\mathrm{NP}}$)$_{\mathrm{VP}}$)$_{\mathrm{S}}$

Figure 1: A tree (a) and its sequential form (b). There is a one-to-one mapping between a tree and its sequential form. (Part-of-speech tags are not used.)

1.1 Language Modeling

Formally, a language model (LM) is a probability distribution over strings of a language:

$$P(\boldsymbol{x}) = P(x_1, \cdots, x_n) = \prod_{t=1}^{n} P(x_t | x_1, \cdots, x_{t-1}), \quad (1)$$

where $\boldsymbol{x}$ is a sentence and t indicates a word position. The efforts in language modeling go into computing $P(x_t|x_1, \cdots, x_{t-1})$, which as described next is useful for parsing as well.

1.2 Parsing as Language Modeling

A generative parsing model parses a sentence $(\boldsymbol{x})$ into its phrasal structure $(\boldsymbol{y})$ according to

$$\operatorname*{argmax}_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} P(\boldsymbol{x}, \boldsymbol{y}'),$$

where $\mathcal{Y}(\boldsymbol{x})$ lists all possible structures of $\boldsymbol{x}$. If we think of a tree $(\boldsymbol{x}, \boldsymbol{y})$ as a sequence $(\boldsymbol{z})$ (Vinyals et

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2331–2336,
Austin, Texas, November 1-5, 2016.

al., 2015) as illustrated in Figure 1, we can define a probability distribution over $(\boldsymbol{x}, \boldsymbol{y})$ as follows:

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{z}) = P(z_1, \cdots, z_m) = \prod_{t=1}^{m} P(z_t | z_1, \cdots, z_{t-1}), \quad (2)$$

which is equivalent to Equation (1). We have reduced parsing to language modeling and can use language modeling techniques of estimating $P(z_t | z_1, \cdots, z_{t-1})$ for parsing.

2 Previous Work

We look here at three neural net (NN) models closest to our research along various dimensions. The first (Zaremba et al., 2014) gives the basic language modeling architecture that we have adopted, while the other two (Vinyals et al., 2015; Dyer et al., 2016) are parsing models that have the current best results in NN parsing.

2.1 LSTM-LM

The LSTM-LM of Zaremba et al. (2014) turns $(x_1, \cdots, x_{t-1})$ into h_t, a hidden state of an LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2003; Graves, 2013), and uses h_t to guess x_t:

$$P(x_t | x_1, \cdots, x_{t-1}) = P(x_t | h_t) = \mathrm{softmax}(W h_t)[x_t],$$

where W is a parameter matrix and $[i]$ indexes ith element of a vector. The simplicity of the model makes it easily extendable and scalable, which has inspired a character-based LSTM-LM that works well for many languages (Kim et al., 2016) and an ensemble of large LSTM-LMs for English with astonishing perplexity of 23.7 (Jozefowicz et al., 2016). In this paper, we build a parsing model based on the LSTM-LM of Zaremba et al. (2014).

2.2 MTP

Vinyals et al. (2015) observe that a phrasal structure $(\boldsymbol{y})$ can be expressed as a sequence and build a machine translation parser (MTP), a sequence-to-sequence model, which translates $\boldsymbol{x}$ into $\boldsymbol{y}$ using a conditional probability:

$$P(\boldsymbol{y} | \boldsymbol{x}) = P(y_1, \cdots, y_l | \boldsymbol{x}) = \prod_{t=1}^{l} P(y_t | \boldsymbol{x}, y_1, \cdots, y_{t-1}),$$

where the conditioning event $(\boldsymbol{x}, y_1, \cdots, y_{t-1})$ is modeled by an LSTM encoder and an LSTM decoder. The encoder maps $\boldsymbol{x}$ into $\boldsymbol{h}^e$, a set of vectors that represents $\boldsymbol{x}$, and the decoder obtains a summary vector (h'_t) which is concatenation of the decoder's hidden state (h^d_t) and weighted sum of word representations $(\sum_{i=1}^{n} \alpha_i h^e_i)$ with an alignment vector $(\boldsymbol{\alpha})$. Finally the decoder predicts y_t given h'_t. Inspired by MTP, our model processes sequential trees.

2.3 RNNG

Recurrent Neural Network Grammars (RNNG), a generative parsing model, defines a joint distribution over a tree in terms of actions the model takes to generate the tree (Dyer et al., 2016):

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{a}) = \prod_{t=1}^{m} P(a_t | a_1, \cdots, a_{t-1}), \quad (3)$$

where $\boldsymbol{a}$ is a sequence of actions whose output precisely matches the sequence of symbols in $\boldsymbol{z}$, which implies Equation (3) is the same as Equation (2). RNNG and our model differ in how they compute the conditioning event $(z_1, \cdots, z_{t-1})$: RNNG combines hidden states of three LSTMs that keep track of actions the model has taken, an incomplete tree the model has generated and words the model has generated whereas our model uses one LSTM's hidden state as shown in the next section.

3 Model

Our model, the model of Zaremba et al. (2014) applied to sequential trees and we call LSTM-LM from now on, is a joint distribution over trees:

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{z}) = \prod_{t=1}^{m} P(z_t | z_1, \cdots, z_{t-1}) = \prod_{t=1}^{m} P(z_t | h_t) = \prod_{t=1}^{m} \mathrm{softmax}(W h_t)[z_t],$$

where h_t is a hidden state of an LSTM. Due to lack of an algorithm that searches through an exponentially large phrase-structure space, we use an n-best parser to reduce $\mathcal{Y}(\boldsymbol{x})$ to $\mathcal{Y}'(\boldsymbol{x})$, whose size is polynomial, and use LSTM-LM to find $\boldsymbol{y}$ that satisfies

$$\underset{\boldsymbol{y}' \in \mathcal{Y}'(\boldsymbol{x})}{\operatorname{argmax}} P(\boldsymbol{x}, \boldsymbol{y}'). \quad (4)$$

3.1 Hyper-parameters

The model has three LSTM layers with 1,500 units and gets trained with truncated backpropagation through time with mini-batch size 20 and step size 50. We initialize starting states with previous mini-batch's last hidden states (Sutskever, 2013). The forget gate bias is initialized to be one (Jozefowicz et al., 2015) and the rest of model parameters are sampled from $\mathcal{U}(-0.05, 0.05)$. Dropout is applied to non-recurrent connections (Pham et al., 2014) and gradients are clipped when their norm is bigger than 20 (Pascanu et al., 2013). The learning rate is $0.25 \cdot 0.85^{\max(\epsilon - 15,\, 0)}$ where ϵ is an epoch number. For simplicity, we use vanilla softmax over an entire vocabulary as opposed to hierarchical softmax (Morin and Bengio, 2005) or noise contrastive estimation (Gutmann and Hyvärinen, 2012).

4 Experiments

We describe datasets we use for evaluation, detail training and development processes.[1]

4.1 Data

We use the Wall Street Journal (WSJ) of the Penn Treebank (Marcus et al., 1993) for training (2-21), development (24) and testing (23) and millions of auto-parsed "silver" trees (McClosky et al., 2006; Huang et al., 2010; Vinyals et al., 2015) for tri-training. To obtain silver trees, we parse the entire section of the New York Times (NYT) of the fifth Gigaword (Parker et al., 2011) with a product of eight Berkeley parsers (Petrov, 2010)[2] and ZPar (Zhu et al., 2013) and select 24 million trees on which both parsers agree (Li et al., 2014). We do not resample trees to match the sentence length distribution of the NYT to that of the WSJ (Vinyals et al., 2015) because in preliminary experiments Charniak parser (Charniak, 2000) performed better when trained on all of 24 million trees than when trained on resampled two million trees.

Given $\boldsymbol{x}$, we produce $\mathcal{Y}'(\boldsymbol{x})$, 50-best trees, with Charniak parser and find $\boldsymbol{y}$ with LSTM-LM as Dyer et al. (2016) do with their discriminative and generative models.[3]

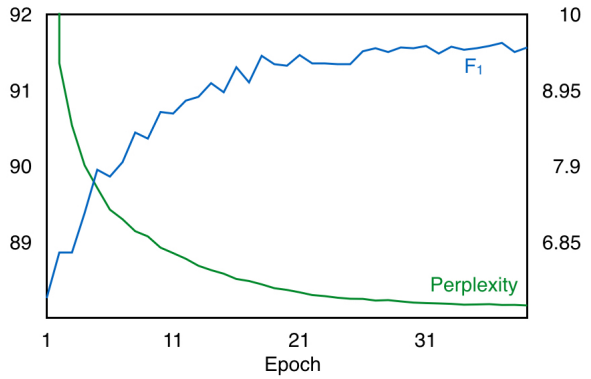


Figure 2: Perplexity and F_1 on the development set at each epoch during training.

n	Oracle	Final	Exact
10	94.0	91.2	39.8
50	96.7	91.7	40.0
51^o	100	93.9	49.7
100	96.3	91.7	39.9
500	97.0	91.8	40.0

Table 1: The performance of LSTM-LM (G) with varying n-best parses on the dev set. Oracle refers to Charniak parser's oracle F_1. Final and Exact report LSTM-LM (G)'s F_1 and exact match percentage respectively. To simulate an optimal scenario, we include gold trees to 50-best trees and rerank them with LSTM-LM (G) (51^o).

4.2 Training and Development

4.2.1 Supervision

We unk words that appear fewer than 10 times in the WSJ training (6,922 types) and drop activations with probability 0.7. At the beginning of each epoch, we shuffle the order of trees in the training data. Both perplexity and F_1 of LSTM-LM (G) improve and then plateau (Figure 2). Perplexity, the

[1] The code and trained models used for experiments are available at github.com/cdg720/emnlp2016.

[2] We use the reimplementation by Huang et al. (2010).

[3] Dyer et al. (2016)'s discriminative model performs comparably to Charniak (89.8 vs. 89.7).

	Base	Final
Vinyals et al. (2015)	88.3	90.5
Dyer et al. (2016)	89.8	92.4
LSTM-LM (G)	89.7	**92.6**

Table 2: F_1 of models trained on WSJ. Base refers to performance of a single base parser and Final that of a final parser.

model's training objective, nicely correlates with F_1, what we care about. Training takes 12 hours (37 epochs) on a Titan X. We also evaluate our model with varying n-best trees including optimal 51-best trees that contain gold trees (51^o). As shown in Table 1, the LSTM-LM (G) is robust given sufficiently large n, i.e. 50, but does not exhibit its full capacity because of search errors in Charniak parser. We address this problem in Section 5.3.

4.2.2 Semi-supervision

We unk words that appear at most once in the training (21,755 types). We drop activations with probability 0.45, smaller than 0.7, thanks to many silver trees, which help regularization. We train LSTM-LM (GS) on the WSJ and a different set of 400,000 NYT trees for each epoch except for the last one during which we use the WSJ only. Training takes 26 epochs and 68 hours on a Titan X. LSTM-LM (GS) achieves 92.5 F_1 on the development.

5 Results

5.1 Supervision

As shown in Table 2, with 92.6 F_1 LSTM-LM (G) outperforms an ensemble of five MTPs (Vinyals et al., 2015) and RNNG (Dyer et al., 2016), both of which are trained on the WSJ only.

5.2 Semi-supervision

We compare LSTM-LM (GS) to two very strong semi-supervised NN parsers: an ensemble of five MTPs trained on 11 million trees of the high-confidence corpus[4] (HC) (Vinyals et al., 2015); and an ensemble of six one-to-many sequence models trained on the HC and 4.5 millions of English-German translation sentence pairs (Luong et al., 2016). We also compare LSTM-LM (GS) to best performing non-NN parsers in the literature. Parsers' parsing performance along with their training data is reported in Table 3. LSTM-LM (GS) outperforms all the other parsers with 93.1 F_1.

5.3 Improved Semi-supervision

Due to search errors – good trees are missing in 50-best trees – in Charniak (G), our supervised and semi-supervised models do not exhibit their full potentials when Charniak (G) provides $\mathcal{Y}'(\boldsymbol{x})$. To mitigate the search problem, we tri-train Charniak (GS) on all of 24 million NYT trees in addition to the WSJ, to yield $\mathcal{Y}'(\boldsymbol{x})$. As shown in Table 3, both LSTM-LM (G) and LSTM-LM (GS) are affected by the quality of $\mathcal{Y}'(\boldsymbol{x})$. A single LSTM-LM (GS) together with Charniak (GS) reaches 93.6 and an ensemble of eight LSTM-LMs (GS) with Charniak (GS) achieves a new state of the art, 93.8 F_1. When trees are converted to Stanford dependencies,[5] UAS and LAS are 95.9% and 94.1%,[6] more than 1% higher than those of the state of the art dependency parser (Andor et al., 2016). Why an indirect method (converting trees to dependencies) is more accurate than a direct one (dependency parsing) remains unanswered (Kong and Smith, 2014).

6 Conclusion

The generative parsing model we presented in this paper is very powerful. In fact, we see that a generative parsing model, LSTM-LM, is more effective than discriminative parsing models (Dyer et al., 2016). We suspect building large models with character embeddings would lead to further improvement as in language modeling (Kim et al., 2016; Jozefowicz et al., 2016). We also wish to develop a complete parsing model using the LSTM-LM framework.

Acknowledgments

We thank the NVIDIA corporation for its donation of a Titan X GPU, Tstaff of Computer Science

[4]The HC consists of 90,000 gold trees, from the WSJ, English Web Treebank and Question Treebank, and 11 million silver trees, whose sentence length distribution matches that of the WSJ, parsed and agreed on by Berkeley parser and ZPar.

[5]Version 3.3.0.

[6]We use the CoNLL evaluator available through the CoNLL website: ilk.uvt.nl/conll/software/eval.pl. Following the convention, we ignore punctuation.

	Base	Oracle	Final	Gold	Silver
Huang et al. (2010)	-	-	92.8	WSJ (40K)	BLLIP (1.8M)
Shindo et al. (2012)	-	-	92.4	WSJ (40K)	-
Choe et al. (2015)	-	-	92.6	WSJ (40K)	NYT (2M)
Vinyals et al. (2015)	-	-	92.8	HC (90K)	HC (11M)
Luong et al. (2016)	-	-	93.0	HC (90K)	HC (11M)
Charniak (G) + LSTM-LM (G)	89.7	96.7	92.6	WSJ (40K)	-
Charniak (G) + LSTM-LM (GS)	89.7	96.7	93.1	WSJ (40K)	NYT (0/10M)
Charniak (GS) + LSTM-LM (G)	91.2	97.1	92.9	WSJ (40K)	NYT (24M/0)
Charniak (GS) + LSTM-LM (GS)	91.2	97.1	93.6	WSJ (40K)	NYT (24M/10M)
Charniak (GS) + E(LSTM-LMs (GS))	91.2	97.1	**93.8**	WSJ (40K)	NYT (24M/11.2M)

Table 3: Evaluation of models trained on the WSJ and additional resources. Note that the numbers of Vinyals et al. (2015) and Luong et al. (2016) are not directly comparable as their models are evaluated on OntoNotes-style trees instead of PTB-style trees. E(LSTM-LMs (GS)) is an ensemble of eight LSTM-LMs (GS). X/Y in Silver column indicates the number of silver trees used to train Charniak parser and LSTM-LM. For the ensemble model, we report the maximum number of trees used to train one of LSTM-LMs (GS).

at Brown University for setting up GPU machines and David McClosky for helping us train Charniak parser on millions trees.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Do Kook Choe, David McClosky, and Eugene Charniak. 2015. Syntactic parse fusion. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Lingpeng Kong and Noah A Smith. 2014. An empirical comparison of parsing methods for stanford dependencies. *arXiv preprint arXiv:1404.4314*.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task se-

quence to sequence learning. *International Conference on Learning Representations*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition. *Linguistic Data Consortium, LDC2011T07*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Ilya Sutskever. 2013. *Training recurrent neural networks*. Ph.D. thesis, University of Toronto.

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 28*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.