A Generative Parser with a Discriminative Recognition Algorithm

Jianpeng Cheng **Adam Lopez** and **Mirella Lapata**
School of Informatics, University of Edinburgh
jianpeng.cheng@ed.ac.uk, {alopez,mlap}@inf.ed.ac.uk

Abstract

Generative models defining joint distributions over parse trees and sentences are useful for parsing and language modeling, but impose restrictions on the scope of features and are often outperformed by discriminative models. We propose a framework for parsing and language modeling which marries a generative model with a discriminative recognition model in an encoder-decoder setting. We provide interpretations of the framework based on expectation maximization and variational inference, and show that it enables parsing and language modeling within a single implementation. On the English Penn Treebank, our framework obtains competitive performance on constituency parsing while matching the state-of-the-art single-model language modeling score.[1]

1 Introduction

Generative models defining joint distributions over parse trees and sentences are good theoretical models for interpreting natural language data, and appealing tools for tasks such as parsing, grammar induction and language modeling (Collins, 1999; Henderson, 2003; Titov and Henderson, 2007; Petrov and Klein, 2007; Dyer et al., 2016). However, they often impose strong independence assumptions which restrict the use of arbitrary features for effective disambiguation. Moreover, generative parsers are typically trained by maximizing the joint probability of the parse tree and the sentence—an objective that only indirectly relates to the goal of parsing. At test time, these models require a relatively expensive recognition algorithm (Collins, 1999; Titov and Henderson, 2007) to recover the parse tree, but the parsing performance consistently lags behind their discriminative competitors (Nivre et al., 2007; Huang, 2008; Goldberg and Elhadad, 2010), which are directly trained to maximize the conditional probability of the parse tree given the sentence, where linear-time decoding algorithms exist (e.g., for transition-based parsers).

In this work, we propose a parsing and language modeling framework that marries a generative model with a discriminative recognition algorithm in order to have the best of both worlds. The idea of combining these two types of models is not new. For example, Collins and Koo (2005) propose to use a generative model to generate candidate constituency trees and a discriminative model to rank them. Sangati et al. (2009) follow the opposite direction and employ a generative model to re-rank the dependency trees produced by a discriminative parser. However, previous work combines the two types of models in a goal-oriented, pipeline fashion, which lacks model interpretations and focuses solely on parsing.

In comparison, our framework unifies generative and discriminative parsers with a single objective, which connects to expectation maximization and variational inference in grammar induction settings. In a nutshell, we treat parse trees as latent factors generating natural language sentences and parsing as a posterior inference task. We showcase the framework using Recurrent Neural Network Grammars (RNNGs; Dyer et al. 2016), a recently proposed probabilistic model of phrase-structure trees based on neural transition systems. Different from this work which introduces separately trained discriminative and generative models, we integrate the two in an auto-encoder which fits our training objective. We show how the framework enables grammar induction, parsing and language

[1] Our code is available at https://github.com/cheng6076/virnng.git.

modeling within a single implementation. On the English Penn Treebank, we achieve competitive performance on constituency parsing and state-of-the-art single-model language modeling score.

2 Preliminaries

In this section we briefly describe Recurrent Neural Network Grammars (RNNGs; Dyer et al. 2016), a top-down transition-based algorithm for parsing and generation. There are two versions of RNNG, one discriminative, the other generative. We follow the original paper in presenting the discriminative variant first.

The discriminative RNNG follows a shift-reduce parser that converts a sequence of words into a parse tree. As in standard shift-reduce parsers, the RNNG uses a buffer to store unprocessed terminal symbols and a stack to store partially completed syntactic constituents. At each timestep, one of the following three operations[2] is performed:

- NT(X) introduces an open non-terminal X onto the top of the stack, represented as an open parenthesis followed by X, e.g., (NP.
- SHIFT fetches the terminal in the front of the buffer and pushes it onto the top of the stack.
- REDUCE completes a subtree by repeatedly popping the stack until an open non-terminal is encountered. The non-terminal is popped as well, after which a composite term representing the entire subtree is pushed back onto the top of the stack, e.g., (NP *the cat*).

The above transition system can be adapted with minor modifications to an algorithm that generates trees and sentences. In generator transitions, there is no input buffer of unprocessed words but there is an output buffer for storing words that have been generated. To reflect the change, the previous SHIFT operation is modified into a GEN operation defined as follows:

- GEN generates a terminal symbol and add it to the stack and the output buffer.

[2]To be precise, the total number of operations under our description is |X|+2 since the NT operation varies with the non-terminal choice X.

3 Methodology

Our framework unifies generative and discriminative parsers within a single training objective. For illustration, we adopt the two RNNG variants introduced above with our customized features. Our starting point is the generative model (§ 3.1), which allows us to make explicit claims about the generative process of natural language sentences. Since this model alone lacks a bottom-up recognition mechanism, we introduce a discriminative recognition model (§ 3.2) and connect it with the generative model in an encoder-decoder setting. To offer a clear interpretation of the training objective (§ 3.3), we first consider the parse tree as latent and the sentence as observed. We then discuss extensions that account for labeled parse trees. Finally, we present various inference techniques for parsing and language modeling within the framework (§ 3.4).

3.1 Decoder (Generative Model)

The decoder is a generative RNNG that models the joint probability $p(x, y)$ of a latent parse tree y and an observed sentence x. Since the parse tree is defined by a sequence of transition actions a, we write $p(x, y)$ as $p(x, a)$.[3] The joint distribution $p(x, a)$ is factorized into a sequence of transition probabilities and terminal probabilities (when actions are GEN), which are parametrized by a transitional state embedding $\mathbf{u}$:

$$\begin{aligned} p(x, a) &= p(a)p(x|a) \\ &= \prod_{t=1}^{|a|} p(a_t|\mathbf{u}_t)p(x_t|\mathbf{u}_t)^{\mathbb{I}(a_t=\text{GEN})} \end{aligned} \quad (1)$$

where $\mathbb{I}$ is an indicator function and $\mathbf{u}_t$ represents the state embedding at time step t. Specifically, the conditional probability of the next action is:

$$p(a_t|\mathbf{u}_t) = \frac{\exp(\mathbf{a}_t\mathbf{u}_t^{\mathrm{T}} + b_a)}{\sum_{a' \in \mathcal{A}} \exp(\mathbf{a}'\mathbf{u}_t^{\mathrm{T}} + b_{a'})} \quad (2)$$

where $\mathbf{a}_t$ represents the action embedding at time step t, $\mathcal{A}$ the action space and b_a the bias. Similarly, the next word probability (when GEN is invoked) is computed as:

$$p(w_t|\mathbf{u}_t) = \frac{\exp(\mathbf{w}_t\mathbf{u}_t^{\mathrm{T}} + b_w)}{\sum_{w' \in \mathcal{W}} \exp(\mathbf{w}'\mathbf{u}_t^{\mathrm{T}} + b_{w'})} \quad (3)$$

[3]We assume that the action probability does not take the actual terminal choice into account.

where $\mathcal{W}$ denotes all words in the vocabulary.

To satisfy the independence assumptions imposed by the generative model, $\mathbf{u}_t$ uses only a restricted set of features defined over the output buffer and the stack — we consider $p(a)$ as a context insensitive prior distribution. Specifically, we use the following features: 1) the stack embedding $\mathbf{d}_t$ which encodes the stack of the decoder and is obtained with a stack-LSTM (Dyer et al., 2015, 2016); 2) the output buffer embedding $\mathbf{o}_t$; we use a standard LSTM to compose the output buffer and $\mathbf{o}_t$ is represented as the most recent state of the LSTM; and 3) the parent non-terminal embedding $\mathbf{n}_t$ which is accessible in the generative model because the RNNG employs a depth-first generation order. Finally, $\mathbf{u}_t$ is computed as:

$$\mathbf{u}_t = \mathbf{W}_2 \tanh(\mathbf{W}_1[\mathbf{d}_t, \mathbf{o}_t, \mathbf{n}_t] + b_d) \quad (4)$$

where $\mathbf{W}$s are weight parameters and b_d the bias.

3.2 Encoder (Recognition Model)

The encoder is a discriminative RNNG that computes the conditional probability $q(a|x)$ of the transition action sequence a given an observed sentence x. This conditional probability is factorized over time steps as:

$$q(a|x) = \prod_{t=1}^{|a|} q(a_t|\mathbf{v}_t) \quad (5)$$

where $\mathbf{v}_t$ is the transitional state embedding of the encoder at time step t.

The next action is predicted similarly to Equation (2), but conditioned on $\mathbf{v}_t$. Thanks to the discriminative property, $\mathbf{v}_t$ has access to any contextual features defined over the entire sentence and the stack — $q(a|x)$ acts as a context sensitive posterior approximation. Our features[4] are: 1) the stack embedding $\mathbf{e}_t$ obtained with a stack-LSTM that encodes the stack of the encoder; 2) the input buffer embedding $\mathbf{i}_t$; we use a bidirectional LSTM to compose the input buffer and represent each word as a concatenation of forward and backward LSTM states; $\mathbf{i}_t$ is the representation of the word on top of the buffer; 3) to incorporate more global features and a more sophisticated look-ahead mechanism for the buffer, we also use an adaptive buffer embedding $\bar{\mathbf{i}}_t$; the latter is computed by having the stack embedding $\mathbf{e}_t$ attend to all remaining embeddings on the buffer with the attention function in Vinyals et al. (2015); and 4) the parent non-terminal embedding $\mathbf{n}_t$. Finally, $\mathbf{v}_t$ is computed as follows:

$$\mathbf{v}_t = \mathbf{W}_4 \tanh(\mathbf{W}_3[\mathbf{e}_t, \mathbf{i}_t, \bar{\mathbf{i}}_t, \mathbf{n}_t] + b_e) \quad (6)$$

where $\mathbf{W}$s are weight parameters and b_e the bias.

3.3 Training

Consider an auto-encoder whose encoder infers the latent parse tree and the decoder generates the observed sentence from the parse tree.[5] The maximum likelihood estimate of the decoder parameters is determined by the log marginal likelihood of the sentence:

$$\log p(x) = \log \sum_a p(x, a) \quad (7)$$

We follow expectation-maximization and variational inference techniques to construct an evidence lower bound of the above quantity (by Jensen's Inequality), denoted as follows:

$$\log p(x) \geq \mathbb{E}_{q(a|x)} \log \frac{p(x, a)}{q(a|x)} = \mathcal{L}_x \quad (8)$$

where $p(x, a) = p(x|a)p(a)$ comes from the decoder or the generative model, and $q(a|x)$ comes from the encoder or the recognition model. The objective function[6] in Equation (8), denoted by $\mathcal{L}_x$, is unsupervised and suited to a grammar induction task. This objective can be optimized with the methods shown in Miao and Blunsom (2016).

Next, consider the case when the parse tree is observed. We can directly maximize the log likelihood of the parse tree for the encoder output $\log q(a|x)$ and the decoder output $\log p(a)$:

$$\mathcal{L}_a = \log q(a|x) + \log p(a) \quad (9)$$

This supervised objective leverages extra information of labeled parse trees to regularize the distribution $q(a|x)$ and $p(a)$, and the final objective is:

$$\mathcal{L} = \mathcal{L}_x + \mathcal{L}_a \quad (10)$$

where $\mathcal{L}_x$ and $\mathcal{L}_a$ can be balanced with the task focus (e.g, language modeling or parsing).

[4] Compared to Dyer et al. (2016), the new features we introduce are 3) and 4), which we found empirically useful.

[5] Here, GEN and SHIFT refer to the same action with different definitions for encoding and decoding.

[6] See § 4 and Appendix A for comparison between this objective and the importance sampler of Dyer et al. (2016).

Learned word embedding dimensions	40
Pretrained word embedding dimensions	50
POS tag embedding dimensions	20
Encoder LSTM dimensions	128
Decoder LSTM dimensions	256
LSTM layer	2
Encoder dropout	0.2
Decoder dropout	0.3

Table 1: Hyperparameters.

3.4 Inference

We consider two inference tasks, namely parsing and language modeling.

Parsing In parsing, we are interested in the parse tree that maximizes the posterior $p(a|x)$ (or the joint $p(a, x)$). However, the decoder alone does not have a bottom-up recognition mechanism for computing the posterior. Thanks to the encoder, we can compute an approximated posterior $q(a|x)$ in linear time and select the parse tree that maximizes this approximation. An alternative is to generate candidate trees by sampling from $q(a|x)$, re-rank them with respect to the joint $p(x, a)$ (which is proportional to the true posterior), and select the sample that maximizes the true posterior.

Language Modeling In language modeling, our goal is to compute the marginal probability $p(x) = \sum_a p(x, a)$, which is typically intractable. To approximate this quantity, we can use Equation (8) to compute a lower bound of the log likelihood $\log p(x)$ and then exponentiate it to get a pessimistic approximation of $p(x)$.[7]

Another way of computing $p(x)$ (without lower bounding) would be to use the variational approximation $q(a|x)$ as the proposal distribution as in the importance sampler of Dyer et al. (2016). We discuss details in Appendix A.

4 Related Work

Our framework is related to a class of variational autoencoders (Kingma and Welling, 2014), which use neural networks for posterior approximation in variational inference. This technique has been previously used for topic modeling (Miao et al., 2016) and sentence compression (Miao and Blunsom, 2016). Another interpretation of the proposed framework is from the perspective of guided policy search in reinforcement learning (Bachman and Precup, 2015), where a generative parser is trained to imitate the trace of a discriminative parser. Further connections can be drawn with the importance-sampling based inference of Dyer et al. (2016). There, a generative RNNG and a discriminative RNNG are trained separately; during language modeling, the output of the discriminative model serves as the proposal distribution of an importance sampler $p(x) = \mathbb{E}_{q(a|x)} \frac{p(x,a)}{q(a|x)}$. Compared to their work, we unify the generative and discriminative RNNGs in a single framework, and adopt a joint training objective.

[7] As a reminder, the language modeling objective is $\exp(NLL/T)$, where NLL denotes the total negative log likelihood of the test data and T the token counts.

Discriminative parsers	Socher et al. (2013)	90.4
	Zhu et al. (2013)	90.4
	Dyer et al. (2016)	91.7
	Cross and Huang (2016)	89.9
	Vinyals et al. (2015)	92.8
Generative parsers	Petrov and Klein (2007)	90.1
	Shindo et al. (2012)	92.4
	Dyer et al. (2016)	93.3
This work	$\operatorname{argmax}_a q(a\|x)$	89.3
	$\operatorname{argmax}_a p(a, x)$	90.1

Table 2: Parsing results (F1) on the PTB test set.

5 Experiments

We performed experiments on the English Penn Treebank dataset; we used sections 2–21 for training, 24 for validation, and 23 for testing. Following Dyer et al. (2015), we represent each word in three ways: as a learned vector, a pretrained vector, and a POS tag vector. The encoder word embedding is the concatenation of all three vectors while the decoder uses only the first two since we do not consider POS tags in generation. Table 1 presents details on the hyper-parameters we used. To find the MAP parse tree $\operatorname{argmax}_a p(a, x)$ (where $p(a, x)$ is used rank the output of $q(a|x)$) and to compute the language modeling perplexity (where $a \sim q(a|x)$), we collect 100 samples from $q(a|x)$, same as Dyer et al. (2016).

Experimental results for constituency parsing and language modeling are shown in Tables 2 and 3, respectively. As can be seen, the single framework we propose obtains competitive parsing performance. Comparing the two inference

KN-5	255.2
LSTM	113.4
Dyer et al. (2016)	102.4
This work: $a \sim q(a\|x)$	99.8

Table 3: Language modeling results (perplexity).

methods for parsing, ranking approximated MAP trees from $q(a|x)$ with respect to $p(a, x)$ yields a small improvement, as in Dyer et al. (2016). It is worth noting that our parsing performance lags behind Dyer et al. (2016). We believe this is due to implementation disparities, such as the modeling of the reduce operation. While Dyer et al. (2016) use an LSTM as the syntactic composition function of each subtree, we adopt a rather simple composition function based on embedding averaging, which gains computational efficiency but loses accuracy.

On language modeling, our framework achieves lower perplexity compared to Dyer et al. (2016) and baseline models. This gain possibly comes from the joint optimization of both the generative and discriminative components towards a language modeling objective. However, we acknowledge a subtle difference between Dyer et al. (2016) and our approach compared to baseline language models: while the latter incrementally estimate the next word probability, our approach (and Dyer et al. 2016) directly assigns probability to the entire sentence. Overall, the advantage of our framework compared to Dyer et al. (2016) is that it opens an avenue to unsupervised training.

6 Conclusions

We proposed a framework that integrates a generative parser with a discriminative recognition model and showed how it can be instantiated with RNNGs. We demonstrated that a unified framework, which relates to expectation maximization and variational inference, enables effective parsing and language modeling algorithms. Evaluation on the English Penn Treebank, revealed that our framework obtains competitive performance on constituency parsing and state-of-the-art results on single-model language modeling. In the future, we would like to perform grammar induction based on Equation (8), with gradient descent and posterior regularization techniques (Ganchev et al., 2010).

Acknowledgments We thank three anonymous reviewers and members of the ILCC for valuable feedback, and Muhua Zhu and James Cross for help with data preparation. The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

A Comparison to Importance Sampling (Dyer et al., 2016)

In this appendix we highlight the connections between importance sampling and variational inference, thereby comparing our method with Dyer et al. (2016).

Consider a simple directed graphical model with discrete latent variables a (e.g., a is the transition action sequence) and observed variables x (e.g., x is the sentence). The model evidence, or the marginal likelihood $p(x) = \sum_a p(x, a)$ is often intractable to compute. Importance sampling transforms the above quantity into an expectation over a distribution $q(a)$, which is known and easy to sample from:

$$p(x) = \sum_a p(x, a) \frac{q(a)}{q(a)} = \mathbb{E}_{q(a)} w(x, a) \quad (11)$$

where $q(a)$ is the proposal distribution and $w(x, a) = \frac{p(x,a)}{q(a)}$ the importance weight. The proposal distribution can potentially depend on the observations x, i.e., $q(a) \triangleq q(a|x)$.

A challenge with importance sampling lies in choosing a proposal distribution which leads to low variance. As shown in Rubinstein and Kroese (2008), the optimal choice of the proposal distribution is in fact the true posterior $p(a|x)$, in which case the importance weight $\frac{p(a,x)}{p(a|x)} = p(x)$ is constant with respect to a. In Dyer et al. (2016), the proposal distribution depends on x, i.e., $q(a) \triangleq q(a|x)$, and is computed with a separately-trained, discriminative model. This proposal choice is close to optimal, since in a fully supervised setting a is also observed and the discriminative model can be trained to approximate the true posterior well. We hypothesize that the performance of their importance sampler is dependent on this specific proposal distribution. Besides, their training strategy does not generalize to an unsupervised setting.

In comparison, variational inference approach approximates the log marginal likelihood $\log p(x)$ with the evidence lower bound. It is a natural choice when one aims to optimize Equation (11)

directly:

$$\begin{aligned}\log p(x) &= \log \sum_a p(x,a)\frac{q(a)}{q(a)} \\ &\geq \mathbb{E}_{q(a)} \log \frac{p(x,a)}{q(a)}\end{aligned} \tag{12}$$

where $q(a)$ is the variational approximation of the true posterior. Again, the variational approximation can potentially depend on the observation x (i.e., $q(a) \triangleq q(a|x)$) and can be computed with a discriminative model. Equation (12) is a well-defined, unsupervised training objective which allows us to jointly optimize generative (i.e., $p(x,a)$) and discriminative (i.e., $q(a|x)$) models. To further support the observed variable a, we augment this objective with supervised terms shown in Equation (10), following Kingma et al. (2014) and Miao and Blunsom (2016).

Equation (12) can be also used to approximate the marginal likelihood $p(x)$ (e.g., in language modeling) with its lower bound. An alternative choice without lower bounding is to use the variational approximation $q(a|x)$ as the proposal distribution in importance sampling (Equation (11)). Ghahramani and Beal (2000) show that this proposal distribution leads to improved results of importance samplers.

References

Philip Bachman and Doina Precup. 2015. Data generation as sequential decision making. In *Advances in Neural Information Processing Systems*, MIT Press, pages 3249–3257.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Pennsylvania, Philadelphia.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1):25–70.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany, pages 32–37.

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)*. Beijing, China, pages 334–343.

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*. San Diego, California, pages 199–209.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research* 11(Jul):2001–2049.

Zoubin Ghahramani and Matthew J. Beal. 2000. Variational inference for Bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems*, MIT Press, pages 449–455.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California, pages 742–750.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Edmonton, Canada, pages 24–31.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*. Columbus, Ohio, pages 586–594.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, MIT Press, pages 3581–3589.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*. Banff, Canada.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 319–328.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*. New York, New York, USA, pages 1727–1736.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China, pages 1791–1799.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(2):95–135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1532–1543.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York, pages 404–411.

Reuven Y Rubinstein and Dirk P Kroese. 2008. *Simulation and the Monte Carlo method*. John Wiley & Sons.

Federico Sangati, Willem Zuidema, and Rens Bod. 2009. A generative re-ranking model for dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*. Paris, France, pages 238–241.

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: Long Papers-Volume 1*. Jeju Island, Korea, pages 440–448.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria, pages 455–465.

Ivan Titov and James Henderson. 2007. Constituent parsing with incremental sigmoid belief networks. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic, pages 632–639.

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*, MIT Press, pages 2773–2781.

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 (Volume 1: Long Papers)*. Sofia, Bulgaria, pages 434–443.