Appendix

A Data Preprocessing

As discussed in the main paper, we subsample the train/test split of the OBWB dataset, respectively. Before this subsampling, we filter sentences with unknown token and punctuations in different form with PTB. By filtering these sentences, we ensure a fair evaluation of language modeling quality since we rule out the effect of unknown tokens in the OBWB dataset. We find that the quotation marks and commas have different forms with PTB dataset. Also, some sentences have brackets which conflict with the parsing brackets. Since OBWB is a large dataset with a billion words, we just filter all these sentences. The number of remaining sentences is: 534,951 sentences in training and 108,178 sentences in test. To make the data size comparable to the PTB dataset, we take the first 50K sentences in training as our OBWB training set, and the first 2.5K sentences in test as our OBWB test set, which is of similar scale to PTB dataset. We will release all preprocessed datasets to facilitate other researchers for further investigation.

B Hyperparameters

For all models, the weights are initialized from a uniform distribution on $[-1/\sqrt{d}, 1/\sqrt{d}]$ where d is the layer size. The initial learning rate is 0.001 for the parser (both encoder and decoder), and 20 for the joint generative model. We reduce the learning rate by half once the validation loss increases. Dropout rate is set as 0.5 for all models. The mini-batch size is 128 for the parser and 20 for the joint generative model. With regard to the network structure of parser, we use a 3-layer bi-directional GRU for the encoder, and a 3-layer standard GRU for the decoder. For the joint generative model, we use a 2-layer standard GRU. The hidden size is 256 for all models, including the GRU language model. In semi-supervised setting, we set the first 100 iterations as the accumulation period to train the baseline function without updating NVLM parameters. For importance sampling on PTB, we use a coefficient α and renormalize the flattened distribution.[3] For OBWB experiments, we use $\alpha = 0.7$ for all models.

[3]With a flattened distribution, the parser can generate parse trees with more diversity. We follow Dyer et al. (2016) to set $\alpha = 0.8$ for PTB experiments.

Model	F_1
Henderson (2004)	89.4
Socher et al. (2013)	90.4
Zhu et al. (2013)	90.4
Seq2Seq (Vinyals et al., 2015)	88.3
RNNG (Dyer et al., 2016)	92.4
Choe and Charniak (2016)	92.6
SO-RNNG (Kuncoro et al., 2017)	93.6
GA-RNNG (Kuncoro et al., 2017)	93.5
CharLSTM (Kitaev and Klein, 2018)	93.6
Suzuki et al. (2018)	94.1
NVLM-parser-only	87.6
NVLM	90.7

Table 5: Test parsing performance F_1 (in percentage) on PTB §23. Note that for fair comparison, results achieved by incorporating external resources, such as additional training data and pre-trained embeddings from external corpus, are not reported here. NVLM-parser-only refers to the parser alone in NVLM, and NVLM refers to the parser plus the joint generative model, where we use the joint generative model to rerank the parsing candidates.

C Parsing Performance

We report our parsing performance and several representative parsing models in Table 5. For NVLM, we use reranking method to further improve the parsing performance. Specifically, we sample 100 parse trees for each sentence, and then rerank them by the joint generative model. The one with the highest log likelihood is selected as the final parse for F_1 evaluation on PTB test dataset.

D Parsing Speed

To our best knowledge, most existing state-of-the-art parsers such as Seq2Seq (Vinyals et al., 2015) and RNNG (Dyer et al., 2016) are implemented on CPUs, and take several days to converge. Even the model has been trained, the parsing speed of new sentences is slow. We provide a highly efficient implementation of our parser on GPUs based on PyTorch (Paszke et al., 2017). Our parser is capable of parsing ∼300 sentences per second on a single NVIDIA GTX 1080 GPU, which is ∼1M sentences per hour.

We list training and testing speed of various parsers in Table 6. Speed of other parsers is estimated from their papers, open-sourced software

documents, and a parser comparison paper (Kong and Smith, 2014). Although this comparison is not exactly fair, since we implement on GPUs. However, we can still see that our parser is rather efficient.

E Gradient Derivation

We derive the gradient in Eq. (11) as follows.

$$
\begin{aligned}
&\nabla_{\theta_1}\tilde{\mathcal{J}}(\mathcal{D}_X) \\
=&\nabla_{\theta_1}\mathbb{E}_{P_{\theta_1}(y|x)}\Big[\log P_{\theta_2}(x,y) - \log P_{\theta_1}(y|x)\Big] \\
=&\nabla_{\theta_1}\sum_y P_{\theta_1}(y|x)\log P_{\theta_2}(x,y) - \\
&\nabla_{\theta_1}\sum_y P_{\theta_1}(y|x)\log P_{\theta_1}(y|x) \\
=&\sum_y \log P_{\theta_2}(x,y)\nabla_{\theta_1}P_{\theta_1}(y|x) - \\
&\sum_y \big(\log P_{\theta_1}(y|x)+1\big)\nabla_{\theta_1}P_{\theta_1}(y|x) \\
=&\sum_y \big(\log P_{\theta_2}(x,y) - \log P_{\theta_1}(y|x)\big)\nabla_{\theta_1}P_{\theta_1}(y|x) \\
=&\mathbb{E}_{P_{\theta_1}(y|x)}\Big[\nabla_{\theta_1}P_{\theta_1}(y|x)A(x,y)\Big], \qquad (14)
\end{aligned}
$$

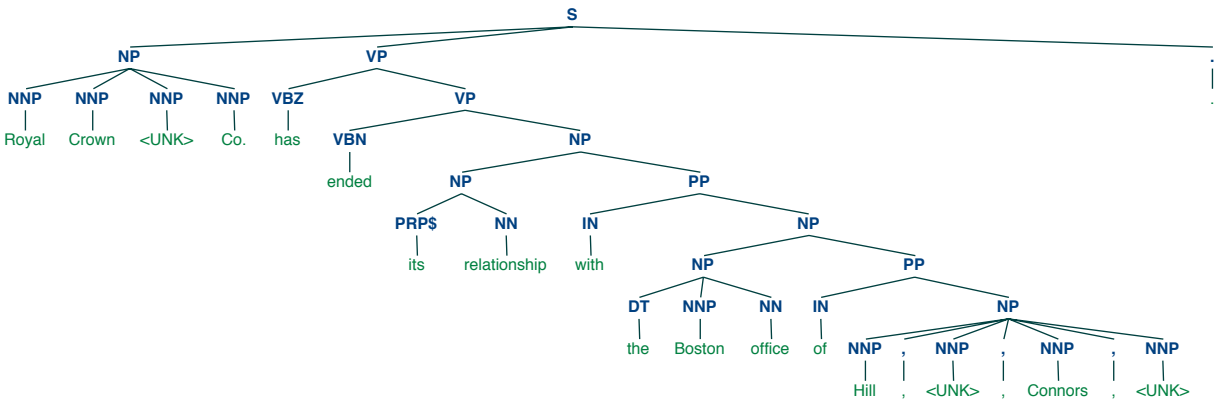
where $A(x,y) = \log P_{\theta_2}(x,y) - \log P_{\theta_1}(y|x)$. Note that we have $\sum_y \nabla_{\theta_1}P_{\theta_1}(y|x) = 0$ because summing over the probability leads to constant 1 and the gradient with respect to constant is 0.

F Case Study of Parser

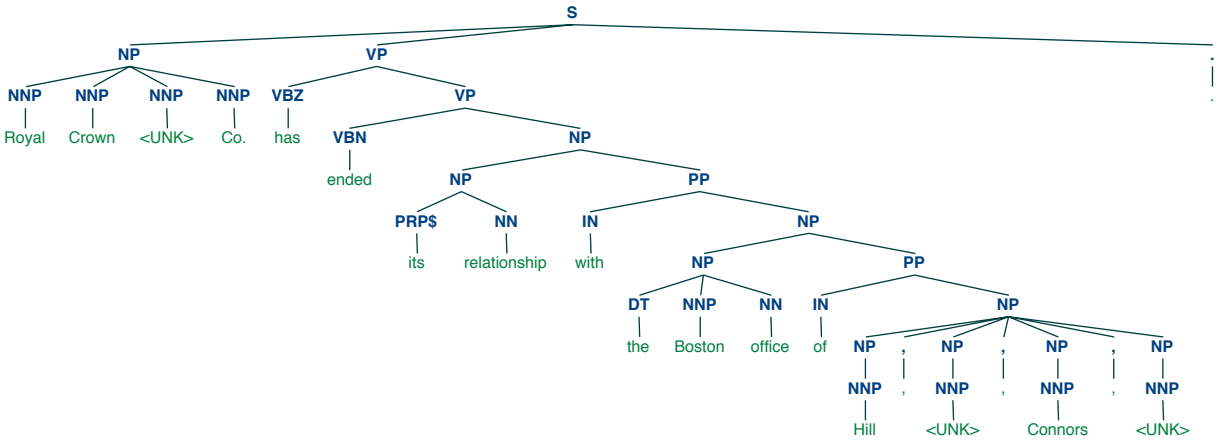
We randomly pick up a sentence from the PTB test dataset, and then visualize the ground-truth parse tree and the output of our parser, as shown in Figure 3. We see that although the sentence is not short (21 tokens), the parser generates almost perfect results except the redundant four "NP"s on the bottom right. This shows that our parser is indeed capturing the language structure, and such information can effectively improve language model or help it quickly land on new corpus.

Parser	Training Time (hours)	Parsing Speed (sentences / s)
Stanford PCFG Parser (Klein and Manning, 2003)	-	6
Stanford RNN Parser (Socher et al., 2013)	-	3
Berkeley Parser (Petrov et al., 2006)	-	10
Charniak Parser (Charniak and Johnson, 2005)	-	5
Seq2Seq (Vinyals et al., 2015)	-	120
RNNG (Dyer et al., 2016)	168	-
NVLM	12	300

Table 6: Comparison of parser training and testing speed.



(a) Ground-truth parse tree



(b) Parse tree generated by our parser

Figure 3: Case study of the performance of our parser.