	Train on PTB	Train on OBWB
Test on PTB	112.3	**419.9**
Test on OBWB	**242.2**	139.3

Table 1: Test perplexity of RNN language model, which performs terribly when training and testing on different datasets.

available, we can separately train both components. On new corpus without tree annotations, we fix the pre-trained parser and train the generative model either from scratch or with warm-up. The pre-trained parser is armed with grammar knowledge, thus it boosts up our language model to land on new corpus. Our proposed framework also supports end-to-end joint training of the two components, so that we can fine-tune the language model. Experimental results show that our proposed framework is effective in all leaning schemes, which achieves good performance on two popular benchmark datasets. With the help of shared grammar, our language model converges significantly faster to a lower perplexity on new corpus.

Our contributions in this paper are summarized as follows:

- *Grammar-sharing framework*: We propose a framework for grammar-sharing language modeling, which incorporates the common grammar knowledge into language modeling. With the shared grammar, our framework helps language model efficiently transfer to new corpus with better performance and using shorter time.
- *End-to-end learning*: Our framework can be end-to-end trained without syntactic annotations. To tackle the technical challenges in end-to-end learning, we use variational methods that exploit policy gradient algorithm for joint training.
- *Efficient software package*: We provide a highly efficient implementation of our work on GPUs. Our parser is capable of parsing one million sentences per hour on a single GPU. See Appendix D for details.

2 Model

In this section, we first provide an overview of the proposed framework, then briefly introduce how components work together, and finally present the probabilistic formulation of each component.

2.1 Framework

As shown in Figure 1, neural variational language model (NVLM) consists of two probabilistic components: 1) a constituency parser $P_{\theta_1}(y|x)$ which models the conditional probability of the parse tree y (a syntax tree without terminal tokens) given the input sentence x (a sequence of terminal tokens); 2) a joint generative model $P_{\theta_2}(x, y)$ which models the joint probability of the sentence and the parse tree.

Constituency Parsing. Our parser can work independently, which takes as input a sentence x and parses x according to

$$\operatorname*{argmax}_{y' \in \mathcal{Y}(x)} P_{\theta_1}(y'|x), \tag{1}$$

where $\mathcal{Y}(x)$ denotes the collection of all possible parses of x. Our parser can also cooperate with the joint generative model as

$$\operatorname*{argmax}_{y' \sim P_{\theta_1}(y|x)} P_{\theta_2}(x, y'), \tag{2}$$

where the parsing candidates y' sampled from $P_{\theta_1}(y|x)$ are fed into the generative model $P_{\theta_2}(x, y)$ to be reranked.

Language Modeling. Statistical language models are typically formulated as

$$\begin{aligned} P(x) &= P(x_1, x_2, \ldots, x_{L_x}) \\ &= \prod_{t=1}^{L_x} P(x_t|x_{<t}), \end{aligned} \tag{3}$$

where x_t denotes the t-th token in the sentence x, the length of x is denoted as L_x, and $x_{<t}$ indicates all tokens before x_t. To evaluate NVLM as a language model, we need to marginalize the joint probability as $P(x) = \sum_{y' \in \mathcal{Y}(x)} P(x, y')$. This is extremely hard to compute due to the exponentially large space of $\mathcal{Y}(x)$. We use importance sampling technique to overcome this computational intractability, which is detailed in Section 4.

With treebank data such as Penn Treebank (Marcus et al., 1993), we have pairs of (x, y) to train the two components respectively, and get high-quality language model with the parser providing grammar knowledge. However, due to the expensive cost of accurate parsing annotation,