Improving Neural Language Models by Segmenting, Attending, and Predicting the Future

Hongyin Luo[1] **Lan Jiang**[2] **Yonatan Belinkov**[1] **James Glass**[1]
[1]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA
{hyluo, belinkov, glass}@mit.edu
[2]School of Information Sciences, University of Illinois at Urbana–Champaign
Champaign, IL 61820, USA
lanj3@illinois.edu

Abstract

Common language models typically predict the next word given the context. In this work, we propose a method that improves language modeling by learning to align the given context and the following phrase. The model does not require any linguistic annotation of phrase segmentation. Instead, we define syntactic heights and phrase segmentation rules, enabling the model to automatically induce phrases, recognize their task-specific heads, and generate phrase embeddings in an unsupervised learning manner. Our method can easily be applied to language models with different network architectures since an independent module is used for phrase induction and context-phrase alignment, and no change is required in the underlying language modeling network. Experiments have shown that our model outperformed several strong baseline models on different data sets. We achieved a new state-of-the-art performance of 17.4 perplexity on the Wikitext-103 dataset. Additionally, visualizing the outputs of the phrase induction module showed that our model is able to learn approximate phrase-level structural knowledge without any annotation.

1 Introduction

Neural language models are typically trained by predicting the next word given a past context (Bengio et al., 2003). However, natural sentences are not constructed as simple linear word sequences, as they usually contain complex syntactic information. For example, a subsequence of words can constitute a phrase, and two non-neighboring words can depend on each other. These properties make natural sentences more complex than simple linear sequences.

Most recent work on neural language modeling learns a model by encoding contexts and matching the context embeddings to the embedding of the next word (Bengio et al., 2003; Merity et al., 2017; Melis et al., 2017). In this line of work, a given context is encoded with a neural network, for example a long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) network, and is represented with a distributed vector. The log-likelihood of predicting a word is computed by calculating the inner product between the word embedding and the context embedding. Although most models do not explicitly consider syntax, they still achieve state-of-the-art performance on different corpora. Efforts have also been made to utilize structural information to learn better language models. For instance, parsing-reading-predict networks (PRPN; Shen et al., 2017) explicitly learn a constituent parsing structure of a sentence and predict the next word considering the internal structure of the given context with an attention mechanism. Experiments have shown that the model is able to capture some syntactic information.

Similar to word representation learning models that learns to match word-to-word relation matrices (Mikolov et al., 2013; Pennington et al., 2014), standard language models are trained to factorize context-to-word relation matrices (Yang et al., 2017). In such work, the context comprises all previous words observed by a model for predicting the next word. However, we believe that context-to-word relation matrices are not sufficient for describing how natural sentences are constructed. We argue that natural sentences are generated at a higher level before being decoded to words. Hence a language model should be able to predict the following sequence of words given a context. In this work, we propose a model that factorizes a context-to-phrase mutual information matrix to learn better language models. The context-to-phrase mutual information matrix describes the relation among contexts and the probabilities of

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1483–1493
Florence, Italy, July 28 - August 2, 2019.

phrases following given contexts. We make the following contributions in this paper:

- We propose a phrase prediction model that improves the performance of state-of-the-art word-level language models.
- Our model learns to predict approximate phrases and headwords without any annotation.

2 Related Work

Neural networks have been widely applied in natural language modeling and generation (Bengio et al., 2003; Bahdanau et al., 2014) for both encoding and decoding. Among different neural architectures, the most popular models are recurrent neural networks (RNNs; Mikolov et al., 2010), long short-term memory networks (LSTMs; Hochreiter and Schmidhuber, 1997), and convolutional neural networks (CNNs; Bai et al., 2018; Dauphin et al., 2017).

Many modifications of network structures have been made based on these architectures. LSTMs with self-attention can improve the performance of language modeling (Tran et al., 2016; Cheng et al., 2016). As an extension of simple self-attention, transformers (Vaswani et al., 2017) apply multi-head self-attention and have achieved competitive performance compared with recurrent neural language models. A current state-of-the-art model, Transformer-XL (Dai et al., 2018), applied both a recurrent architecture and a multi-head attention mechanism. To improve the quality of input word embeddings, character-level information is also considered (Kim et al., 2016). It has also been shown that context encoders can learn syntactic information (Shen et al., 2017).

However, instead of introducing architectural changes, for example a self-attention mechanism or character-level information, previous studies have shown that careful hyper-parameter tuning and regularization techniques on standard LSTM language models can obtain significant improvements (Melis et al., 2017; Merity et al., 2017). Similarly, applying more careful dropout strategies can also improve the language models (Gal and Ghahramani, 2016; Melis et al., 2018). LSTM language models can be improved with these approaches because LSTMs suffer from serious over-fitting problems.

Recently, researchers have also attempted to improve language models at the decoding phase. Inan et al. (2016) showed that reusing the input word embeddings in the decoder can reduce the perplexity of language models. Yang et al. (2017) showed the low-rank issue in factorizing the context-to-word mutual information matrix and proposed a multi-head softmax decoder to solve the problem. Instead of predicting the next word by using only similarities between contexts and words, the neural cache model (Grave et al., 2016) can significantly improve language modeling by considering the global word distributions conditioned on the same contexts in other parts of the corpus.

To learn the grammar and syntax in natural languages, Dyer et al. (2016) proposed the recurrent neural network grammar (RNNG) that models language incorporating a transition parsing model. Syntax annotations are required in this model. To utilize the constituent structures in language modeling without syntax annotation, parse-read-predict networks (PRPNs; Shen et al., 2017) calculate syntactic distances among words and computes self-attentions. Syntactic distances have been proved effective in constituent parsing tasks (Shen et al., 2018a). In this work, we learn phrase segmentation with a model based on this method and our model does not require syntax annotation.

3 Syntactic Height and Phrase Induction

In this work, we propose a language model that not only predicts the next word of a given context, but also attempts to match the embedding of the next phrase. The first step of this approach is conducting phrase induction based on syntactic heights. In this section, we explain the definition of syntactic height in our approach and describe the basics ideas about whether a word can be included in an induced phrase.

Intuitively, the syntactic height of a word aims to capture its distance to the root node in a dependency tree. In Figure 1, the syntactic heights are represented by the red bars. A word has high syntactic height if it has low distance to the root node.

A similar idea, named syntactic distance, is proposed by Shen et al. (2017) for constructing constituent parsing trees. We apply the method for calculating syntactic distance to calculate syntactic height. Given a sequence of embeddings of input words $[x_1, x_2, \cdots, x_n]$, we calculate their syntactic heights with a temporal convolutional net-

work (TCN) (Bai et al., 2018).

$$d_i = W_d \cdot [x_{i-n}, x_{i-n+1}, \cdots, x_i]^T + b_d \quad (1)$$

$$h_i = W_h \cdot ReLU(d_i) + b_h \quad (2)$$

where h_i stands for the syntactic height of word x_i. The syntactic height h_i for each word is a scalar, and W_h is a $1 \times D$ matrix, where D is the dimensionality of d_i. These heights are learned and not imposed by external syntactic supervision. In Shen et al. (2017), the syntactic heights are used to generate context embeddings. In our work, we use the syntactic heights to predict induced phrases and calculate their embeddings.

We define the phrase induced by a word based on the syntactic heights. Consider two words x_i and x_k. x_k belongs to the phrase induced by x_i if and only if for any $j \in (i, k)$, $h_j < max(h_i, h_k)$. For example, in Figure 1, the phrase induced by the red marked word **the** is "the morning flights", since the syntactic height of the word **morning**, $h_{morning} < h_{flights}$. However, the word "to" does not belong to the phrase because $h_{flights}$ is higher than both h_{the} and h_{to}. The induced phrase and the inducing dependency connection are labeled in blue in the figure.

Note that this definition of an induced phrase does not necessarily correspond to a phrase in the syntactic constituency sense. For instance, the words "to Houston" would be included in the phrase "the morning flights to Houston" in a traditional syntactic tree. Given the definition of induced phrases, we propose phrase segmenting conditions (PSCs) to find the last word of an induced phrase. Considering the induced phrase of the i-th word, $s_i = [x_i, x_{i+1}, \cdots, x_j]$. If x_j is not the last word of a given sentence, there are two conditions that x_j should satisfy:

1. (PSC-1) The syntactic height of x_j must be higher than the height of x_i, that is

$$h_j - h_i > 0 \quad (3)$$

2. (PSC-2) The syntactic height of x_{j+1} should be lower that x_j.

$$h_j - h_{j+1} > 0 \quad (4)$$

Given the PSCs, we can decide the induced phrases for the sentence shown in Figure 1. The last word of the phrase induced by "United" is "canceled", and the last word of the phrase induced by "flights" is "Houston". For the word assigned the highest syntactic height, its induced phrase is all remaining words in the sentence.

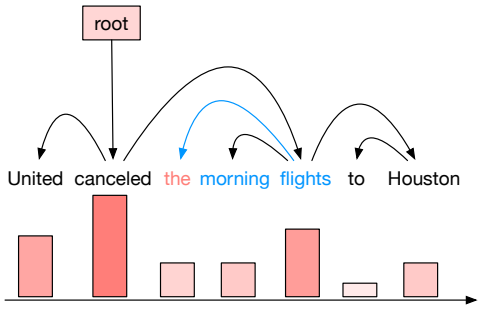


Figure 1: Groundtruth dependency tree and syntactic heights of each word.

4 Model

In this work, we formulate multi-layer neural language models as a two-part framework. For example, in a two-layer LSTM language model (Merity et al., 2017), we use the first layer as phrase generator and the last layer as a word generator:

$$[c_1, c_2, \cdots, c_T] = RNN^1([x_1, x_2, \cdots, x_T]) \quad (5)$$

$$[y_1, y_2, \cdots, y_T] = RNN^2([c_1, c_2, \cdots, c_T]) \quad (6)$$

For a L-layer network, we can regard the first L_1 layers as the phrase generator and the next $L_2 = L - L_1$ layers as the word generator. Note that we use y_i to represent the hidden state output by the second layer instead of h_i, since h_i in our work is defined as the syntactic height of x_i. In the traditional setting, the first layer does not explicitly learn the semantics of the following phrase because there is no extra objective function for phrase learning.

In this work, we force the first layer to output context embeddings c_i for phrase prediction with three steps. Firstly, we predict the induced phrase for each word according to the PSCs proposed in Section 3. Secondly, we calculate the embedding of each phrase with a head-finding attention. Lastly, we align the context embedding and phrase embedding with negative sampling. The word generation is trained in the same way as standard language models. The diagram of the model is shown in Figure 2. The three steps are described next.

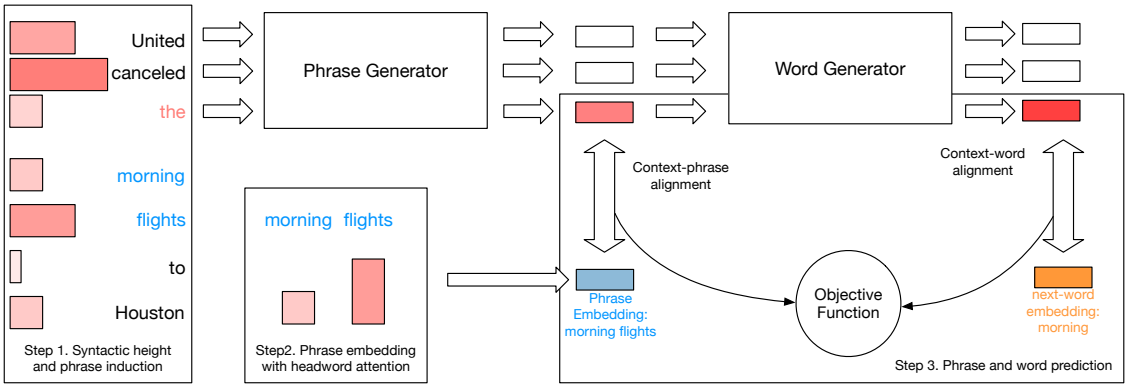


Figure 2: The 3-step diagram of our approach. The current target word is "the", the induced phrase is "morning flights", and the next word is "morning". The context-phrase and context-word alignments are jointly trained.

4.1 Phrase Segmentation

We calculate the syntactic height and predict the induced phrase for each word:

$$h_i = TCN([x_{i-n}, x_{i-n+1}, \cdots, x_i]) \quad (7)$$

where $TCN(\cdot)$ stands for the TCN model described in Equations (1) and (2), and n is the width of the convolution window.

Based on the proposed phrase segmenting conditions (PSCs) described in the previous section, we predict the probability of a word being the first word outside a induced phrase. Firstly, we decide if each word, $x_{j-1}, j \in (i+1, n]$, satisfies the two phrase segmenting conditions, PSC-1 and PSC-2. The probability that x_j satisfies PSC-1 is

$$p^1_{psc}(x_j) = \frac{1}{2} \cdot (f^{HT}(h_j - h_i) + 1) \quad (8)$$

Similarly, the probability that x_j satisfies PSC-2 is

$$p^2_{psc}(x_j) = \frac{1}{2} \cdot (f^{HT}(h_j - h_{j+1}) + 1) \quad (9)$$

where f_{HT} stands for the HardTanh function with a temperature a:

$$f^{HT}(x) = \begin{cases} -1 & x \leq -\frac{1}{a} \\ a \cdot x & -\frac{1}{a} < x \leq \frac{1}{a} \\ 1 & x > \frac{1}{a} \end{cases}$$

This approach is inspired by the context attention method proposed in the PRPN model (Shen et al., 2017).

Then we can infer the probability of whether a word belongs to the induced phrase of x_i with

$$p^{ind}(x_j) = \prod_{k=1}^{j} \hat{p}(x_k) \quad (10)$$

where $p^{ind}(x_i)$ stands for the probability that x_i belongs to the induced phrase, and

$$\hat{p}(x_k) = \begin{cases} 1 & k \leq i+1 \\ 1 - p^1_{psc}(x_{k-1}) \cdot p^2_{psc}(x_{k-1}) & k > i+1 \end{cases}$$

Note that the factorization in Equation 10 assumes that words are independently likely to be included in the induced phrase of x_i.

4.2 Phrase Embedding with Attention

Given induced phrases, we can calculate their embeddings based on syntactic heights. To calculate the embedding of phrase $s = [x_1, x_2, \cdots, x_n]$, we calculate an attention distribution over the phrase:

$$\alpha_i = \frac{h_i \cdot p^{ind}(x_i) + c}{\sum_j h_j \cdot p^{ind}(x_j) + c} \quad (11)$$

where h_i stands for the syntactic height for word x_i and c is a constant real number for smoothing the attention distribution. Then we generate the phrase embedding with a linear transformation:

$$s = W \cdot \sum_i \alpha_i \cdot e_i \quad (12)$$

where e_i is the word embedding of x_i. In training, we apply a dropout layer on s.

4.3 Phrase and Word Prediction

A traditional language model learns the probability of a sequence of words:

$$p(x_1, x_2, \cdots, x_n) = p(x_1) \cdot \prod_i p(x_{i+1}|x_1^i) \quad (13)$$

where x_1^i stands for $x_1, x_2, \cdots, x_i$, which is the context used for predicting the next word, x_{i+1}. In most related studies, the probability $p(x_{i+1}|x_1^i)$

is calculated with the output of the top layer of a neural network y_i and the word representations e_{i+1} learned by the decoding layer:

$$p(x_{i+1}) = Softmax(e_{i+1}^T \cdot y_i) \quad (14)$$

The state-of-the-art neural language models contain multiple layers. The outputs of different hidden layers capture different level of semantics of the context. In this work, we force one of the hidden layers to align its output with the embeddings of induced phrases s_i. We apply an embedding model similar to Mikolov et al. (2013) to train the hidden output and phrase embedding alignment. We define the context-phrase alignment model as follows.

We first define the probability that a phrase ph_i can be induced by context $[x_1, \ldots, x_i]$.

$$p(ph_i|x_1^i) = \sigma(c_i^T \cdot s_i) \quad (15)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$, and c_i stands for the context embedding of $x_1, x_2, \cdots, x_i$ output by a hidden layer, defined in Equation 5. s_i is the generated embedding of an induced phrase. The probability that a phrase ph_i cannot be induced by context $[x_1, \ldots, x_i]$ is $1 - p(ph_i|x_1^i)$. This approach follows the method for learning word embeddings proposed in Mikolov et al. (2013).

We use an extra objective function and the negative sampling strategy to align context representations and the embeddings of induced phrases. Given the context embedding c_i, the induced phrase embedding s_i, and random sampled negative phrase embeddings s_i^{neg}, we train the neural network to maximize the likelihood of true induced phrases and minimize the likelihood of negative samples. we define the following objective function for context i:

$$l_i^{CPA} = 1 - \sigma(c_i^T \cdot s_i) + \frac{1}{n}\sum_{j=1}^{n} \sigma(c_i^T \cdot s_j^{neg}) \quad (16)$$

where n stands for the number of negative samples. With this loss function, the model learns to maximize the similarity between the context and true induced phrase embeddings, and minimize the similarity between the context and negative samples randomly selected from the induced phrases of other words. In practice, this loss function is used as a regularization term with a coefficient γ:

$$l = l^{LM} + \gamma \cdot l^{CPA} \quad (17)$$

It worth noting that our approach is model-agnostic and and can be applied to various architectures. The TCN network for calculating the syntactic heights and phrase inducing is an independent module. In context-phrase alignment training with negative sampling, the objective function provides phrase-aware gradients and does not change the word-by-word generation process of the language model.

5 Experiments

We evaluate our model with word-level language modeling tasks on Penn Treebank (PTB; Mikolov et al., 2010), Wikitext-2 (WT2; Bradbury et al., 2016), and Wikitext-103 (WT103; Merity et al., 2016) corpora.

The PTB dataset has a vocabulary size of 10,000 unique words. The entire corpus includes roughly 40,000 sentences in the training set, and more than 3,000 sentences in both valid and test set.

The WT2 data is about two times larger the the PTB dataset. The dataset consists of Wikipedia articles. The corpus includes 30,000 unique words in its vocabulary and is not cleaned as heavily as the PTB corpus.

The WT103 corpus contains a larger vocabulary and more articles than WT2. It consists of 28k articles and more than 100M words in the training set. WT2 and WT103 corpora can evaluate the ability of capturing long-term dependencies (Dai et al., 2018).

In each corpus, we apply our approach to publicly-available, state-of-the-art models. This demonstrates that our approach can improve different existing architectures. Our trained models will be published for downloading. The implementation of our models is publicly available.[1]

5.1 Penn Treebank

We train a 3-layer AWD-LSTM language model (Merity et al., 2017) on PTB data set. We use 1,150 as the number of hidden neurons and 400 as the size of word embeddings. We also apply the word embedding tying strategy (Inan et al., 2016). We apply variational dropout for hidden states (Gal and Ghahramani, 2016) and the dropout rate is 0.25. We also apply weight dropout (Merity et al., 2017) and set weight dropout rate as 0.5. We apply stochastic gradient descent (SGD) and averaged SGD (ASGD; Polyak and Juditsky, 1992)

[1] https://github.com/luohongyin/PILM

Model	#Params	Dev PPL	Test PPL
Inan et al. (2016) – Tied Variational LSTM	24M	75.7	73.2
Zilly et al. (2017) – Recurrent Highway Networks	23M	67.9	65.7
Shen et al. (2017) – PRPN	-	-	62.0
Pham et al. (2018) – Efficient NAS	24M	60.8	58.6
Melis et al. (2017) – 4-layer skip LSTM (tied)	24M	60.9	58.3
Shen et al. (2018b) – ON-LSTM	25M	58.3	56.2
Liu et al. (2018) – Differentiable NAS	23M	58.3	56.1
Merity et al. (2017) – AWD-LSTM	24M	60.7	58.8
Merity et al. (2017) – AWD-LSTM + finetuning	24M	60.0	57.3
Ours – AWD-LSTM + Phrase Induction - NS	24M	61.0	58.6
Ours – AWD-LSTM + Phrase Induction - Attention	24M	60.2	58.0
Ours – AWD-LSTM + Phrase Induction	24M	59.6	57.5
Ours – AWD-LSTM + Phrase Induction + finetuning	24M	57.8	55.7
Dai et al. (2018) – Transformer-XL	24M	56.7	54.5
Yang et al. (2017) – AWD-LSTM-MoS + finetuning	22M	**56.5**	**54.4**

Table 1: Experimental results on Penn Treebank dataset. Compared with the AWD-LSTM baseline models, our method reduced the perplexity on test set by 1.6.

Model	#Params	Dev PPL	Test PPL
Inan et al. (2016) – Variational LSTM (tied)	28M	92.3	87.7
Inan et al. (2016) – VLSTM + augmented loss	28M	91.5	87.0
Grave et al. (2016) – LSTM	-	-	99.3
Grave et al. (2016) – LSTM + Neural cache	-	-	68.9
Melis et al. (2017) – 1-Layer LSTM	24M	69.3	69.9
Melis et al. (2017) – 2-Layer Skip Conn. LSTM	24M	69.1	65.9
Merity et al. (2017) – AWD-LSTM + finetuning	33M	68.6	65.8
Ours – AWD-LSTM + Phrase Induction	33M	68.4	65.2
Ours – AWD-LSTM + Phrase Induction + finetuning	33M	**66.9**	**64.1**

Table 2: Experimental results on Wikitext-2 dataset.

for training. The learning rate is 30 and we clip the gradients with a norm of 0.25. For the phrase induction model, we randomly sample 1 negative sample for each context, and the context-phrase alignment loss is given a coefficient of 0.5. The output of the second layer of the neural network is used for learning context-phrase alignment, and the final layer is used for word generation.

We compare the word-level perplexity of our model with other state-of-the-art models and our baseline is AWD-LSTM (Merity et al., 2017). The experimental results are shown in Table 1. Although not as good as the Transformer-XL model (Dai et al., 2018) and the mixture of softmax model (Yang et al., 2017), our model significantly improved the AWD-LSTM, reducing 2.2 points of perplexity on the validation set and 1.6 points of perplexity on the test set. Note that the "finetuning" process stands for further training the language models with ASGD algorithm (Merity et al., 2017).

We also did an ablation study without either headword attention or negative sampling (NS). The results are listed in Table 1. By simply averaging word vectors in the induced phrase Without the attention mechanism, the model performs worse than the full model by 0.5 perplexity, but is still better than our baseline, the AWD-LSTM model. In the experiment without negative sampling, we only use the embedding of true induced

Model	#Params	Test PPL
Grave et al. (2016) – LSTM	-	48.7
Bai et al. (2018) – TCN	-	45.2
Dauphin et al. (2017) – GCNN-8	-	44.9
Grave et al. (2016) – LSTM + Neural cache	-	40.8
Dauphin et al. (2017) – GCNN-14	-	37.2
Merity et al. (2018) – 4-layer QRNN	151M	33.0
Rae et al. (2018) – LSTM + Hebbian + Cache	-	29.9
Dai et al. (2018) – Transformer-XL Standard	151M	24.0
Baevski and Auli (2018) – Adaptive input	247M	20.5
Dai et al. (2018) – Transformer-XL Large	257M	18.3
Ours – Transformer-XL Large + Phrase Induction	257M	**17.4**

Table 3: Experimental results on Wikitext-103 dataset.

phrases to align with the context embedding. It is also indicated that the negative sampling strategy can improve the performance by 1.1 perplexity. Hence we just test the full model in the following experiments.

5.2 Wikitext-2

We also trained a 3-layer AWD-LSTM language model on the WT2 dataset. The network has the same input size, output size, and hidden size as the model we applied on PTB dataset, following the experiments done by Merity et al. (2017). Some hyper-parameters are different from the PTB language model. We use a batch size of 60. The embedding dropout rate is 0.65 and the dropout rate of hidden outputs is set to 0.2. Other hyper-parameters are the same as we set in training on the PTB dataset.

The experimental results are shown in Table 2. Our model improves the AWD-LSTM model by reducing 1.7 points of perplexity on both the validation and test sets, while we did not make any change to the architecture of the AWD-LSTM language model.

5.3 Wikitext-103

The current state-of-the-art language model trained on Wikitext-103 dataset is the Transformer-XL (Dai et al., 2018). We apply our method on the state-of-the-art Transformer-XL Large model, which has 18 layers and 257M parameters. The input size and hidden size are 1024. 16 attention heads are used. We regard the first 14 layers as the phrase generator and the last 4 layers as the word generator. In other words, the context-phrase alignment is trained with the outputs of the 14th layer.

The model is trained on 4 Titan X Pascal GPUs, each of which has 12G memory. Because of the limitation of computational resources, we use our approach to fine-tune the officially released pre-trained Transformer-XL Large model for 1 epoch. The experimental results are shown in Table 3. Our approach got 17.4 perplexity with the officially released evaluation scripts, significantly outperforming all baselines and achieving new state-of-the-art performance[2].

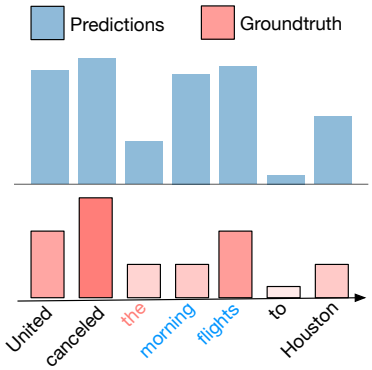
6 Discussion

In this section, we show what is learned by training language models with the context-phrase alignment objective function by visualizing the syntactic heights output by the TCN model and the phrases induced by each target word in a sentence. We also visualize the headword attentions over the induced phrase.

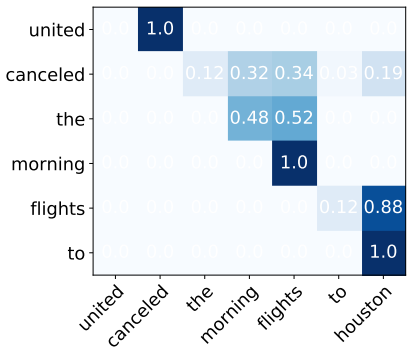
The first example is the sentence showed in Figure 1. The sentence came from Jurafsky and Martin (2014) and did not appear in our training set. Figure 1 shows the syntactic heights and the induced phrase of "the" according to the ground-truth dependency information. Our model is not given such high-quality inputs in either training or evaluation.

Figure 3 visualizes the structure learned by our phrase induction model. The inferred syntactic heights are shown in Figure 3a. Heights assigned

[2]We did not show Dev PPLs in Table 3 since only the correct approach to reproduce the test PPL was provided with the pretrained Transformer-XL model.

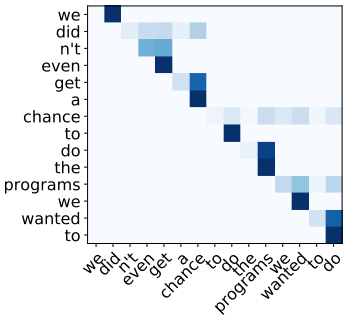


(a) Syntactic heights of each word.

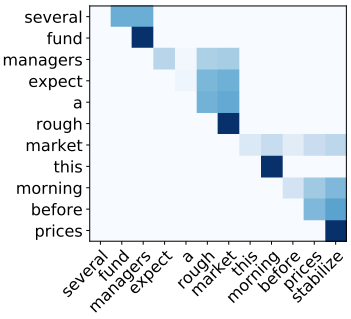


(b) Induced phrases and headword attentions.

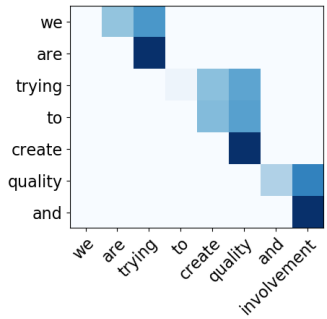
Figure 3: Examples of induced phrases and corresponding headword attention for generating the phrase embedding. The word of each row stands for the target word as the current input of the language model, and the values in each row in the matrices stands for the words consisting the induced phrase and their weights.



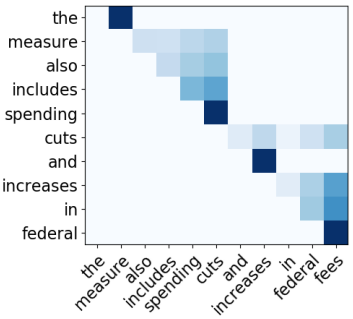
(a)



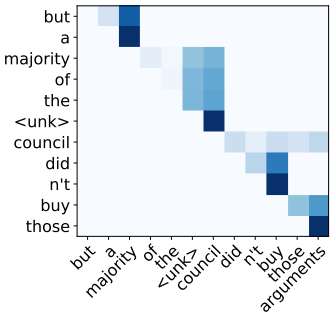
(b)



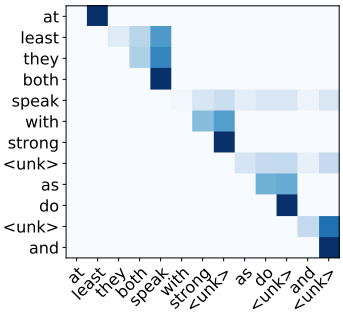
(c)



(d)



(e)



(f)

Figure 4: Examples of phrase inducing and headword attentions.

to words "the" and "to" are significantly lower than others, while the verb "canceled" is assigned the highest in the sentence. Induced phrases are shown in Figure 3b. The words at the beginning of each row stand for the target word of each step. Values in the matrix stand for attention weights for calculating phrase embedding. The weights are calculated with the phrase segmenting conditions (PSC) and the syntactic heights described in Equations 8 to 11. For the target word "united", $h_{united} < h_{canceled}$ and $h_{canceled} > h_{the}$, hence the induced phrase of "united" is a single word "canceled", and the headword attention of "canceled" is 1, which is indicated in the first row of Figure 3b. The phrase induced by "canceled" is the entire following sequence, "the morning flights to houston", since no following word has a higher syntactic height than the target word. It is also shown that the headword of the induced phrase of "canceled" is "flights", which agrees with the dependency structure indicated in Figure 1.

More examples are shown in Figure 4. Figures 4a to 4d show random examples without any unknown word, while the examples shown in Figures 4e and 4f are randomly selected from sentences with unknown words, which are marked with the UNK symbol. The examples show that the phrase induction model does not always predict the exact structure represented by the dependency tree. For example, in Figure 4b, the TCN model assigned the highest syntactic height to the word "market" and induced the phrase "expect a rough market" for the context "the fund managers". However, in a ground-truth dependency tree, the verb "expect" is the word directly connected to the root node and therefore has the highest syntactic height.

Although not exactly matching linguistic dependency structures, the phrase-level structure predictions are reasonable. The segmentation is interpretable and the predicted headwords are appropriate. In Figure 4c, the headwords are "trying", "quality", and "involvement". The model is also robust with unknown words. In Figure 4e, "the <unk> council" is segmented as the induced phrase of "but a majority of". In this case, the model recognized that the unknown word is dependent on "council".

The sentence in Figure 4f includes even more unknown words. However, the model still correctly predicted the root word, the verb "speak". For the target word "with", the induced phrase is "strong <unk>". Two unknown words are located in the last few words of the sentence. The model failed to induce the phrase "<unk> and <unk>" for the word "do", but still successfully split "<unk>" and "and". Meanwhile, the attentions over the phrases induced by "speak", "do", and the first "<unk>" are not quite informative, suggesting that unknown words made some difficulties for headword prediction in this example. However, the unknown words are assigned significantly higher syntactic heights than the word "and".

7 Conclusion

In this work, we improved state-of-the-art language models by aligning context and induced phrases. We defined syntactic heights and phrase segmentation rules. The model generates phrase embeddings with headword attentions. We improved the AWD-LSTM and Transformer-XL language models on different data sets and achieved state-of-the-art performance on the Wikitext-103 corpus. Experiments showed that our model successfully learned approximate phrase-level knowledge, including segmentation and headwords, without any annotation. In future work, we aim to capture better structural information and possible connections to unsupervised grammar induction.

References

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2018. Transformer-xl: Language modeling with longer-term dependency.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

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

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*, volume 3. Pearson London.

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

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Gábor Melis, Charles Blundell, Tomáš Kočiskỳ, Karl Moritz Hermann, Chris Dyer, and Phil Blunsom. 2018. Pushing the bounds of dropout. *arXiv preprint arXiv:1805.09208*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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)*, pages 1532–1543.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Jack W Rae, Chris Dyer, Peter Dayan, and Timothy P Lillicrap. 2018. Fast parametric learning with activation memorization. *arXiv preprint arXiv:1803.10049*.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2017. Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018a. Straight to the tree: Constituency parsing with neural syntactic distance. *arXiv preprint arXiv:1806.04168*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018b. Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. *arXiv preprint arXiv:1601.01272*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2017. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4189–4198. JMLR. org.