Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Zihang Dai∗12, Zhilin Yang∗12, Yiming Yang1, Jaime Carbonell1, Quoc V. Le2, Ruslan Salakhutdinov1
1Carnegie Mellon University, 2Google Brain
{dzihang,zhiliny,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

Abstract
Transformers have a potential of learning longer-term dependency, but are limited by a fixed-length context in the setting of language modeling. We propose a novel neural architecture Transformer-XL that enables learning dependency beyond a fixed length without disrupting temporal coherence. It consists of a segment-level recurrence mechanism and a novel positional encoding scheme. Our method not only enables capturing longer-term dependency, but also resolves the context fragmentation problem. As a result, Transformer-XL learns dependency that is 80% longer than RNNs and 450% longer than vanilla Transformers, achieves better performance on both short and long sequences, and is up to 1,800+ times faster than vanilla Transformers during evaluation. Notably, we improve the state-of-the-art results of bpc/perplexity to 0.99 on enwiki8, 1.08 on text8, 18.3 on WikiText-103, 21.8 on One Billion Word, and 54.5 on Penn Treebank (without finetuning). When trained only on WikiText-103, Transformer-XL manages to generate reasonably coherent, novel text articles with thousands of tokens. Our code, pretrained models, and hyperparameters are available in both Tensorflow and PyTorch1.

1 Introduction
Language modeling is among the important problems that require modeling long-term dependency, with successful applications such as unsupervised pretraining (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018). However, it has been a challenge to equip neural networks with the capability to model long-term dependency in sequential data. Recurrent neural networks (RNNs), in particular Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), have been a standard solution to language modeling and obtained strong results on multiple benchmarks. Despite the wide adaption, RNNs are difficult to optimize due to gradient vanishing and explosion (Hochreiter et al., 2001), and the introduction of gating in LSTMs and the gradient clipping technique (Graves, 2013) might not be sufficient to fully address this issue. Empirically, previous work has found that LSTM language models use 200 context words on average (Khandelwal et al., 2018), indicating room for further improvement.

On the other hand, the direct connections between long-distance word pairs baked in attention mechanisms might ease optimization and enable the learning of long-term dependency (Bahdanau et al., 2014; Vaswani et al., 2017). Recently, Al-Rfou et al. (2018) designed a set of auxiliary losses to train deep Transformer networks for character-level language modeling, which outperform LSTMs by a large margin. Despite the success, the LM training in Al-Rfou et al. (2018) is performed on separated fixed-length segments of a few hundred characters, without any information flow across segments. As a consequence of the fixed context length, the model cannot capture any longer-term dependency beyond the predefined context length. In addition, the fixed-length segments are created by selecting a consecutive chunk of symbols without respecting the sentence or any other semantic boundary. Hence, the model lacks necessary contextual information needed to well predict the first few symbols, leading to inefficient optimization and inferior performance. We refer to this problem as context fragmentation.

To address the aforementioned limitations of fixed-length contexts, we propose a new architecture called Transformer-XL (meaning extra long). We introduce the notion of recurrence into our
deep self-attention network. In particular, instead of computing the hidden states from scratch for each new segment, we reuse the hidden states obtained in previous segments. The reused hidden states serve as memory for the current segment, which builds up a recurrent connection between the segments. As a result, modeling very long-term dependency becomes possible because information can be propagated through the recurrent connections. Meanwhile, passing information from the previous segment can also resolve the problem of context fragmentation. More importantly, we show the necessity of using relative positional encodings rather than absolute ones, in order to enable state reuse without causing temporal confusion. Hence, as an additional technical contribution, we introduce a simple but more effective relative positional encoding formulation that generalizes to attention lengths longer than the one observed during training.

Transformer-XL obtained strong results on five datasets, varying from word-level to character-level language modeling. Transformer-XL is also able to generate relatively coherent long text articles with thousands of tokens (see Appendix E), trained on only 100M tokens.

Our main technical contributions include introducing the notion of recurrence in a purely self-attentive model and deriving a novel positional encoding scheme. These two techniques form a complete set of solutions, as any one of them alone does not address the issue of fixed-length contexts. Transformer-XL is the first self-attention model that achieves substantially better results than RNNs on both character-level and word-level language modeling.

3 Model

Given a corpus of tokens \( x = (x_1, \ldots, x_T) \), the task of language modeling is to estimate the joint probability \( P(x) \), which is often auto-regressively factorized as \( P(x) = \prod_t P(x_t | x_{<t}) \). With the factorization, the problem reduces to estimating each conditional factor. In this work, we stick to the standard neural approach to modeling the conditional probability. Specifically, a trainable neural network is used to encode the context \( x_{<t} \) into a fixed size hidden state, which is multiplied with the word embeddings to obtain the logits. The logits are then fed into the Softmax function, yielding a categorical probability distribution over the next token.

3.1 Vanilla Transformer Language Models

In order to apply Transformer or self-attention to language modeling, the central problem is how to train a Transformer to effectively encode an arbitrarily long context into a fixed size representation. Given infinite memory and computation, a simple solution would be to process the entire context sequence using an unconditional Transformer decoder, similar to a feed-forward neural network. However, this is usually infeasible with the limited resource in practice.

One feasible but crude approximation is to split the entire corpus into shorter segments of man-
ageable sizes, and only train the model within each segment, ignoring all contextual information from previous segments. This is the idea adopted by Al-Rfou et al. (2018). We call it the vanilla model and visualize it in Fig. 1a. Under this training paradigm, information never flows across segments in either the forward or backward pass. There are two critical limitations of using a fixed-length context. First, the largest possible dependency length is upper bounded by the segment length, which is a few hundred on character-level language modeling (Al-Rfou et al., 2018). Therefore, although the self-attention mechanism is less affected by the vanishing gradient problem compared to RNNs, the vanilla model is not able to fully exploit this optimization advantage. Second, though it is possible to use padding to respect the sentence or other semantic boundaries, in practice it has been standard practice to simply chunk long text into fixed-length segments due to improved efficiency (Peters et al., 2018; Devlin et al., 2018; Al-Rfou et al., 2018). However, simply chunking a sequence into fixed-length segments will lead to the context fragmentation problem as discussed in Section 1.

During evaluation, at each step, the vanilla model also consumes a segment of the same length as in training, but only makes one prediction at the last position. Then, at the next step, the segment is shifted to the right by only one position, and the new segment has to be processed all from scratch. As shown in Fig. 1b, this procedure ensures that each prediction utilizes the longest possible context exposed during training, and also relieves context fragmentation issue encountered in training. However, this evaluation procedure is extremely expensive. We will show that our proposed architecture is able to substantially improve the evaluation speed.

3.2 Segment-Level Recurrence with State Reuse

To address the limitations of using a fixed-length context, we propose to introduce a recurrence mechanism to the Transformer architecture. During training, the hidden state sequence computed for the previous segment is fixed and cached to be reused as an extended context when the model processes the next new segment, as shown in Fig. 2a. Although the gradient still remains within a segment, this additional input allows the network to exploit information in the history, leading to an ability of modeling longer-term dependency and avoiding context fragmentation. Formally, let the two consecutive segments of length $L$ be $s_\tau = [x_\tau,1, \cdots, x_\tau,L]$ and $s_{\tau+1} = [x_{\tau+1,1}, \cdots, x_{\tau+1,L}]$ respectively. Denoting the $n$-th layer hidden state sequence produced for the $\tau$-th segment $s_\tau$ by $h_{\tau,n} \in \mathbb{R}^{L \times d}$, where $d$ is the hidden dimension. Then, the $n$-th layer hidden state for segment $s_{\tau+1}$ is produced (schematically) as follows,

$$h_{\tau+1,n}^\top = \text{SG}(h_{\tau,n}^\top \odot h_{\tau,n+1}^\top),$$

$$q_{\tau+1,n}, k_{\tau+1,n}, v_{\tau+1,n} = h_{\tau+1,n}^\top W_q, h_{\tau+1,n}^\top W_k, h_{\tau+1,n}^\top W_v,$$

$$h_{\tau+1} = \text{Transformer-Layer}(q_{\tau+1,n}^\top, k_{\tau+1,n}^\top, v_{\tau+1,n}^\top),$$

where the function $\text{SG}(\cdot)$ stands for stop-gradient, the notation $[h_u \odot h_v]$ indicates the concatenation of two hidden sequences along the length dimension, and $W_\cdot$ denotes model parameters. Compared to the standard Transformer, the critical difference lies in that the key $k_{\tau+1,n}^\top$ and value $v_{\tau+1,n}^\top$ are conditioned on the extended context $h_{\tau,n+1}^\top$ and hence $h_{\tau,n+1}^\top$ cached from the previous segment. We emphasize this particular design by the green paths in Fig. 2a.

With this recurrence mechanism applied to every two consecutive segments of a corpus, it essentially creates a segment-level recurrence in the hidden states. As a result, the effective context being utilized can go way beyond just two segments. However, notice that the recurrent dependency between $h_{\tau,n}^\top$ and $h_{\tau,n+1}^\top$ shifts one layer downwards.
per-segment, which differs from the same-layer recurrence in conventional RNN-LMs. Consequently, the largest possible dependency length grows linearly w.r.t. the number of layers as well as the segment length, i.e., \( O(N \times L) \), as visualized by the shaded area in Fig. 2b. This is analogous to truncated BPTT (Mikolov et al., 2010), a technique developed for training RNN-LMs. However, different from truncated BPTT, our method caches a sequence of hidden states instead of the last one, and should be applied together with the relative positional encoding technique described in Section 3.3.

Besides achieving extra long context and resolving fragmentation, another benefit that comes with the recurrence scheme is significantly faster evaluation. Specifically, during evaluation, the representations from the previous segments can be reused instead of being computed from scratch as in the case of the vanilla model. In our experiments on enwiki8, Transformer-XL is up to 1,800+ times faster than the vanilla model during evaluation (see Section 4).

Finally, notice that the recurrence scheme does not need to be restricted to only the previous segment. In theory, we can cache as many previous segments as the GPU memory allows, and reuse all of them as the extra context when processing the current segment. Thus, we can cache a pre-defined length-\( M \) old hidden states spanning (possibly) multiple segments, and refer to them as the memory \( m^t \in \mathbb{R}^{M \times d} \), due to a clear connection to the memory augmented neural networks (Graves et al., 2014; Weston et al., 2014). In our experiments, we set \( M \) equal to the segment length during training, and increase it by multiple times during evaluation.

### 3.3 Relative Positional Encodings

While we found the idea presented in the previous subsection very appealing, there is a crucial technical challenge we haven’t solved in order to reuse the hidden states. That is, how can we keep the positional information coherent when we reuse the states? Recall that, in the standard Transformer, the information of sequence order is provided by a set of positional encodings, denoted as \( U \in \mathbb{R}^{L_{\text{max}} \times d} \), where the \( i \)-th row \( U_i \) corresponds to the \( i \)-th absolute position within a segment and \( L_{\text{max}} \) prescribes the maximum possible length to be modeled. Then, the actual input to the Transformer is the element-wise addition of the word embeddings and the positional encodings. If we simply adapt this positional encoding to our recurrence mechanism, the hidden state sequence would be computed schematically by

\[
\begin{align*}
    h_{r+1} &= f(h_r, E_{s_{r+1}} + U_{1:L}) \\
    h_r &= f(h_{r-1}, E_{s_r} + U_{1:L}),
\end{align*}
\]

where \( E_{s_r} \in \mathbb{R}^{L \times d} \) is the word embedding sequence of \( s_r \), and \( f \) represents a transformation function. Notice that, both \( E_{s_r} \) and \( E_{s_{r+1}} \) are associated with the same positional encoding \( U_{1:L} \). As a result, the model has no information to distinguish the positional difference between \( x_{r,j} \) and \( x_{r+1,j} \) for any \( j = 1, \ldots, L \), resulting in a sheer performance loss.

In order to avoid this failure mode, the fundamental idea is to only encode the relative positional information in the hidden states. Conceptually, the positional encoding gives the model a temporal clue or “bias” about how information should be gathered, i.e., where to attend. For the same purpose, instead of incorporating bias statically into the initial embedding, one can inject the same information into the attention score of each layer. More importantly, it is more intuitive and generalizable to define the temporal bias in a relative manner. For instance, when a query vector \( q_{r,i} \) attends on the key vectors \( k_{r,j} \), it does not need to know the absolute position of each key vector to identify the temporal order of the segment. Instead, it suffices to know the relative distance between each key vector \( k_{r,j} \) and itself \( q_{r,i} \), i.e. \( i - j \). Practically, one can create a set of relative posi-
can easily distinguish the representations of \( x_{r,j} \) and \( x_{r+1,j} \) from their different distances, making the state reuse mechanism feasible. Meanwhile, we won’t lose any temporal information, as the absolute position can be recovered recursively from relative distances.

Previously, the idea of relative positional encodings has been explored in the context of machine translation (Shaw et al., 2018) and music generation (Huang et al., 2018). Here, we offer a different derivation, arriving at a new form of relative positional encodings, which not only has a one-to-one correspondence to its absolute counterpart but also enjoys much better generalization empirically (see Section 4). Firstly, in the standard Transformer (Vaswani et al., 2017), the attention score between query \( q_i \) and key vector \( k_j \) within the same segment can be decomposed as

\[
A_{i,j}^{\text{abs}} = E_q^\top W_q^\top W_k E_k + E_q^\top W_q^\top W_k R_{i,j}.
\]

Following the idea of only relying on relative positional information, we propose to reparameterize the four terms as follows

\[
A_{i,j}^{\text{rel}} = E_q^\top W_q^\top W_k E_k + E_q^\top W_q^\top W_k R_{i,j} + U_q^\top W_q U_j + U_q^\top W_q U_j.
\]

- The first change we make is to replace all appearances of the absolute positional embedding \( U_j \) for computing key vectors in term (b) and (d) with its relative counterpart \( R_{i-j} \). This essentially reflects the prior that only the relative distance matters for where to attend. Note that \( R \) is a sinusoid encoding matrix (Vaswani et al., 2017) without learnable parameters.
- Secondly, we introduce a trainable parameter \( u \in \mathbb{R}^d \) to replace the query \( U_j^\top W_q \) in term (c). In this case, since the query vector is the same for all query positions, it suggests that the attentive bias towards different words should remain the same regardless of the query position. With a similar reasoning, a trainable parameter \( v \in \mathbb{R}^d \) is added to substitute \( U_j^\top W_q \) in term (d).
- Finally, we deliberately separate the two weight matrices \( W_{k,E} \) and \( W_{k,R} \) for producing the content-based key vectors and location-based key vectors respectively.

Under the new parameterization, each term has an intuitive meaning: term (a) represents content-based addressing, term (b) captures a content-dependent positional bias, term (c) governs a global content bias, and (d) encodes a global positional bias.

In comparison, the formulation in Shaw et al. (2018) only has terms (a) and (b), dropping the two bias terms (c) and (d). Moreover, Shaw et al. (2018) merge the multiplication \( W_k R \) into a single trainable matrix \( R \), which abandons the inductive bias built into the original sinusoid positional encoding (Vaswani et al., 2017). In contrast, our relative positional embedding \( R \) adapts the sinusoid formulation. As a benefit of the inductive bias, a model trained on a memory of some certain length can automatically generalize to a memory several times longer during evaluation.

Equipping the recurrence mechanism with our proposed relative positional embedding, we finally arrive at the Transformer-XL architecture. For completeness, we summarize the computational procedure for a \( N \)-layer Transformer-XL with a single attention head here. For \( n = 1, \ldots, N \):

\[
\mathbf{h}_n = \text{Masked-Softmax} (\mathbf{a}_n) + \mathbf{h}_{n-1}
\]

where

\[
\begin{align*}
\mathbf{a}_n & := \text{LayerNorm} (\text{Linear} (\mathbf{h}_n) + \mathbf{h}_{n-1}) \\
\mathbf{h}_n & := \text{Positionwise-Feed-Forward} (\mathbf{h}_n)
\end{align*}
\]

with \( \mathbf{h}_0 := \mathbf{E}_{w} \) defined as the word embedding sequence. In addition, it is worth mentioning that a naive way to compute \( A \) requires computing \( W_{k,R} R_{i-j} \) for all pairs \((i, j)\), whose cost is quadratic w.r.t. the sequence length. However, noticing that the value of \( i-j \) only ranges from zero to the sequence length, we show a simple computation procedure in Appendix B, which reduces the cost to be linear w.r.t. the sequence length.

4 Experiments

4.1 Main Results

We apply Transformer-XL to a variety of datasets on both word-level and character-level language...
modeling to have a comparison with state-of-the-art systems, including WikiText-103 (Merity et al., 2016), enwik8 (LLC, 2009), text8 (LLC, 2009), One Billion Word (Chelba et al., 2013), and Penn Treebank (Mikolov and Zweig, 2012).

WikiText-103 is the largest available word-level language modeling benchmark with long-term dependency. It contains 103M training tokens from 28K articles, with an average length of 3.6K tokens per article, which allows testing the ability of long-term dependency modeling. We set the attention length to 384 during training and 1600 during evaluation. We adopted adaptive softmax and input representations (Baevski and Auli, 2018; Grave et al., 2016a). As shown in Table 1, Transformer-XL reduces the previous state-of-the-art (SoTA) perplexity from 20.5 to 18.3, which demonstrates the superiority of the Transformer-XL architecture.

The dataset enwik8 contains 100M bytes of unprocessed Wikipedia text. We compare our architecture with the previous results in Table 2. Under the model size constraint, the 12-layer Transformer-XL achieves a new SoTA result, outperforming the 12-layer vanilla Transformer from Al-Rfou et al. (2018) by 0.05, while both Transformer variants have a large margin over conventional RNN-based models. Notably, our 12-layer architecture achieves the same result as the 64-layer network from Al-Rfou et al. (2018), using only 17% of the parameter budget. In order to see whether better performances can be obtained by increasing the model size, we train 18-layer and 24-layer Transformer-XLs with increased model sizes. With the attention length 784 during training and 3,800 during evaluation, we obtained a new SoTA result and our method is the first to break through 1.0 on widely-studied character-level benchmarks. Different from Al-Rfou et al. (2018), Transformer-XL does not need any auxiliary losses, and thus all benefits are credited to a better architecture.

Similar to but different from enwik8, text8 contains 100M processed Wikipedia characters created by lowering case the text and removing any character other than the 26 letters a through z, and space. Due to the similarity, we simply adapt the best model and the same hyper-parameters on enwik8 to text8 without further tuning. The comparison with previous methods is summarized in Table 3. Again, Transformer-XL achieves the new SoTA result with a clear margin.
### Table 5: Comparison with state-of-the-art results on Penn Treebank.

| Model | #Param | PPL |
|-------|--------|-----|
| Inan et al. (2016) - Tied Variational LSTM | 24M | 73.2 |
| Zilly et al. (2016) - Variational RHN | 23M | 65.4 |
| Zoph and Le (2016) - NAS Cell | 25M | 64.0 |
| Merity et al. (2017) - AWD-LSTM | 24M | 58.8 |
| Pham et al. (2018) - Efficient NAS | 24M | 58.6 |
| Liu et al. (2018) - Differentiable NAS | 23M | 56.1 |
| Yang et al. (2017) - AWD-LSTM-MoS | 22M | 55.97 |
| Melis et al. (2018) - Dropout tuning | 24M | 55.3 |
| Ours - Transformer-XL | 24M | **54.52** |
| Merity et al. (2017) - AWD-LSTM+Finetune\(^\d\) | 24M | 57.3 |
| Yang et al. (2017) - MoS+Finetune\(^\d\) | 22M | **54.44** |

Table 5: Comparison with state-of-the-art results on Penn Treebank. † indicates using two-step finetuning.

One Billion Word does not preserve any long-term dependency because sentences have been shuffled. Consequently, this dataset mainly tests the ability of modeling only short-term dependency. The comparison between Transformer-XL and the other methods is shown in Table 4. Although Transformer-XL is mainly designed to better capture longer-term dependency, it dramatically improves the single-model SoTA from 23.7 to 21.8. Specifically, Transformer-XL significantly outperforms a contemporary method using vanilla Transformers (Baevski and Auli, 2018), suggesting the advantage of Transformer-XL is generalizable to modeling short sequences.

We also report the results on word-level Penn Treebank in Table 5. Similar to AWD-LSTM (Merity et al., 2017), we apply variational dropout and weight average to Transformer-XL. With proper regularization, Transformer-XL achieves a new SoTA result among models without two-step finetuning. Penn Treebank has only 1M training tokens, which implies that Transformer-XL also generalizes well even on small datasets.

#### 4.2 Ablation Study

We conduct two sets of ablation studies to examine the effects of two proposed techniques used in Transformer-XL: the recurrence mechanism and the new positional encoding scheme.

The first study is performed on WikiText-103, which requires modeling long-term dependency. The results are reported in Table 6. Among the compared encoding schemes, Shaw et al. (2018) is relative, while Vaswani et al. (2017) and Al-Rfou et al. (2018) are absolute. “Full” and “half” losses refer to applying a cross entropy loss to all or the recent half positions in the segment. We found that absolute encodings only work well with half losses because half losses exclude positions with very short attention lengths during training for better generalization. Table 6 shows that both the recurrence mechanism and our encoding scheme are necessary to achieve the best performance, as well as generalizing to longer attention sequences during evaluation time. Although the backpropagation length during training is only 128, with the two techniques the attention length can be increased to 640 at test time. In the standard setting with 151M parameters, the perplexity decreases as the attention length increases.

Since the recurrence mechanism costs additional memory, we also compare Transformer-XL with baselines under the same GPU memory constraints. As shown in Table 10 in Appendix A, despite using a shorter backpropagation length, Transformer-XL remains superior to the baselines.

The second study targets at isolating the effects of resolving the context fragmentation problem from the benefit of capturing longer context length. In order to achieve this goal, we deliberately choose a dataset that does not require long-term dependency, so that any improvement from establishing the recurrence can be attributed to solving the context fragmentation. Specifically, we perform this controlled experiment on the One Billion Word dataset, which can only benefit from removing the context fragmentation. We train a 20-layer Transformer-XL with ~0.3B parameters for 400K steps. As shown in Table 7, using segment-level recurrence substantially improves performance even when long-term dependency is not needed, which is consistent with our previous discussion that the recurrence mechanism resolves the context fragmentation problem. Moreover, our relative positional encodings is also superior to Shaw et al. (2018) on short sequences.

#### 4.3 Relative Effective Context Length

Khandelwal et al. (2018) proposed a method to evaluate the Effective Context Length (ECL) of a sequence model. ECL is the longest length to which increasing the context span would lead to a gain more than a threshold. However, ECL ignores the fact that it is harder to get improvement when a model already achieves a lower perplexity using only a shorter context, and thus it is not suitable for fair comparison among multiple models. We instead propose a new metric...
Table 6: Ablation study on WikiText-103. For the first two blocks, we use a slightly smaller model (128M parameters). † indicates that the corresponding row is reduced to the same setting as the Transformer network in (Al-Rfou et al., 2018), except that two auxiliary losses are not implemented in our experiments. “PPL init” refers to using the same length as training. “PPL best” indicates the perplexity obtained by using the optimal length. “Attn Len” is the shortest possible attention length during evaluation to achieve the corresponding result (PPL best). Increasing the attention length during evaluation improves performance only when our positional encoding is used. The “Transformer-XL (151M)” setting uses a standard parameter budget as previous work (Merity et al., 2018), where we observe a similar effect when increasing the attention length during evaluation.

Table 7: Ablation study on One Billion Word, a dataset without long-term dependency.

Table 8: Relative effective context length (RECL) comparison. See text for the definition of RECL and r. The first three models and the last four models are compared as two model groups when we calculate RECL (RECL is computed on a model group rather than a single model). Each group has the same parameter budget.

Table 9: Slowdown in terms of running time during evaluation. Evaluation is based on per-token time on one GPU.
5 Conclusions

Transformer-XL obtains strong perplexity results, models longer-term dependency than RNNs and Transformer, achieves substantial speedup during evaluation, and is able to generate coherent text articles. We envision interesting applications of Transformer-XL in the fields of text generation, unsupervised feature learning, image and speech modeling.

Acknowledgments

ZD and YY were supported in part by National Science Foundation (NSF) under the grant IIS-1546329 and by the DOE-Office of Science under the grant ASCR #KJ040201. ZY and RS were supported in part by the Office of Naval Research grant N000141812861, the NSF grant IIS1763562, the Nvidia fellowship, and the Siebel scholarship.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Lilion Jones. 2018. Character-level language modeling with deeper self-attention. arXiv preprint arXiv:1808.04444.

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.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint arXiv:1312.3005.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. arXiv preprint arXiv:1609.01704.

Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. 2016. Recurrent batch normalization. arXiv preprint arXiv:1603.09025.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In Advances in neural information processing systems, pages 3079–3087.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. arXiv preprint arXiv:1612.09083.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Adji Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. arXiv preprint arXiv:1611.01702.

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, Moustapha Cissé, David Grangier, and Hervé Jégou. 2016a. Efficient softmax approximation for gpus. arXiv preprint arXiv:1609.04309.

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

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

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. arXiv preprint arXiv:1410.5401.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. arXiv preprint arXiv:1609.09106.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in current nets: the difficulty of learning long-term dependencies. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. 2018. An improved relative self-attention mechanism for transformer with application to music generation. arXiv preprint arXiv:1809.04281.

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

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. 2018. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, pages 10434–10443.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Noam Shazeer, Joris Pelemans, and Ciprian Chelba. 2014. Skip-gram language modeling using sparse non-negative matrix probability estimation. *arXiv preprint arXiv:1412.1454*.

Trieu H Trinh, Andrew M Dai, Thang Luong, and Quoc V Le. 2018. Learning longer-term dependencies in rnns with auxiliary losses. *arXiv preprint arXiv:1803.00144*.

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.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2017. Topic compositional neural language model. *arXiv preprint arXiv:1712.09783*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. 2016. On multiplicative integration with recurrent neural networks. In *Advances in neural information processing systems*, pages 2856–2864.

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*.

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

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.
A Ablation Study with Memory Constraints

| Backprop Len | Recurrence | Encoding | Loss | ppix best | ppix init | Attn Len |
|--------------|------------|----------|------|-----------|-----------|----------|
| 128          | ✔          | Ours     | Full | 26.77     | 27.02     | 500      |
| 128          | ✔          | Ours     | Partial | 28.33 | 28.69 | 460 |
| 176          | ⨿         | Ours     | Full | 27.98     | 28.43     | 400      |
| 172          | ⨿         | Ours     | Partial | 28.83 | 28.83 | 120      |

Table 10: Ablation study on WikiText-103 with the same GPU memory constraints.

B Efficient Computation of the Attention with Relative Positional Embedding

As we discussed in section 3.3, the naive way of computing the $W_{k,R}R_{i-j}$ for all pairs $(i,j)$ is subject to a quadratic cost. Here, we present a simple method with only a linear cost. Firstly, notice that the relative distance $i-j$ can only be integer from 0 to $M + L - 1$, where $M$ and $L$ are the memory length and segment length respectively. Hence, the rows of the matrix

$$Q := \begin{bmatrix} R_{0}^\top \quad R_{1}^\top \quad \cdots \quad R_{M+L-1}^\top \end{bmatrix}$$

$$W_{k,R}^\top = \begin{bmatrix} [W_{k,R}R_{M+L-1}]^\top \\ [W_{k,R}R_{M+L-2}]^\top \\ \vdots \\ [W_{k,R}R_{0}]^\top \end{bmatrix} \in \mathbb{R}^{(M+L) \times d}$$

consist of all possible vector outputs of $W_{k,R}R_{i-j}$ for any $(i,j)$. Note that we have defined $Q$ in a reversed order, i.e., $Q_k = W_{k,R}R_{M+L-1-k}$, to make further discussion easier.

Next, we collect the term $(b)$ for all possible $i,j$ into the following $L \times (M + L)$ matrix,

$$B = \begin{bmatrix} q_0^T W_{k,R}R_{M} & \cdots & q_0^T W_{k,R}R_{0} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{L-1}^T W_{k,R}R_{M+L-1} & \cdots & q_{L-1}^T W_{k,R}R_{0} & 0 & \cdots & 0 \end{bmatrix}$$

$$= \begin{bmatrix} q_0^T Q_0 & \cdots & q_0^T Q_M & q_0^T Q_{M+1} & \cdots & q_0^T Q_{M+L-1} \\ q_1^T Q_0 & \cdots & q_1^T Q_M & q_1^T Q_{M+1} & \cdots & q_1^T Q_{M+L-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{L-1}^T Q_0 & \cdots & q_{L-1}^T Q_M & q_{L-1}^T Q_{M+1} & \cdots & q_{L-1}^T Q_{M+L-1} \end{bmatrix}$$

Then, we further define

$$\tilde{B} = qQ^\top = \begin{bmatrix} q_0^T Q_0 & \cdots & q_0^T Q_M & q_0^T Q_{M+1} & \cdots & q_0^T Q_{M+L-1} \\ q_1^T Q_0 & \cdots & q_1^T Q_M & q_1^T Q_{M+1} & \cdots & q_1^T Q_{M+L-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{L-1}^T Q_0 & \cdots & q_{L-1}^T Q_M & q_{L-1}^T Q_{M+1} & \cdots & q_{L-1}^T Q_{M+L-1} \end{bmatrix}$$

Now, it is easy to see an immediate relationship between $B$ and $\tilde{B}$, where the $i$-th row of $B$ is simply a left-shifted version of $i$-th row of $\tilde{B}$. Hence, the computation of $B$ only requires a matrix multiplication $qQ^\top$ to compute $\tilde{B}$ and then a set of left-shifts.

Similarly, we can collect all term $(d)$ for all possible $i,j$ into another $L \times (M + L)$ matrix $D$.

$$D = \begin{bmatrix} v^T Q_{L-1} & \cdots & v^T Q_{M+L-1} & 0 & \cdots & 0 \\ v^T Q_{L-2} & \cdots & v^T Q_{M+L-2} & v^T Q_{M+L-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ v^T Q_0 & \cdots & v^T Q_M & v^T Q_{M+1} & \cdots & v^T Q_{M+L-1} \end{bmatrix}$$
Then, we can follow the same procedure to define
\[
\tilde{d} = [Qv]^{T} = \left[ v^{T}Q_{0} \cdots v^{T}Q_{M} v^{T}Q_{M+1} \cdots v^{T}Q_{M+L-1} \right].
\]
Again, each row of \(D\) is simply a left-shift version of \(\tilde{d}\). Hence, the main computation cost comes from the matrix-vector multiplication \(\tilde{d} = [Qv]^{T}\), which is not expensive any more.

C Details About RECL

Figure 3: Visualizing unnormalized relative perplexity gains with \(r = 0.1\).

Figure 4: Perplexity vs context length.

In this section, we describe the details of the metric RECL. Let \(\mathcal{M} = \{m_{1}, m_{2}, \cdots, m_{N}\}\) be a model group consisting of \(N\) models. Let \(l_{i}(c, t)\) denote the loss of model \(m_{i}\) on the \(t\)-th token in the corpus with a context length \(c\). Concretely, the loss can be written as

\[
l_{i}(c, t) = -\log P_{m_{i}}(x_{t}|x_{t-1}, \cdots, x_{t-c})
\]

where \(P_{m_{i}}\) is the probability distribution given by model \(m_{i}\), and \(x_{t}\) is the \(t\)-th token in the corpus. Given a short context length \(c\) and a long context length \(c'\) such that \(c' \geq c\), we can further define a baseline for each position \(t\),

\[
b(c, t) = \min_{i=1}^{N} l_{i}(c, t)
\]

The relative loss of \(m_{i}\) w.r.t. the model group \(\mathcal{M}\) is written as

\[
f_{i}(c, c') = \frac{1}{|T|} \sum_{t \in T} \min \left( b(c, t), l_{i}(c', t) \right)
\]

The above equation uses the minimum loss of all models on the short length \(c\) as a baseline, and only losses smaller than the baseline will be effectively counted towards the relative loss. This enables fair
comparison between multiple models because all models with a long context length \( c' \) need to improve over the same baseline. Sometimes we only care about those positions where the baseline performs poorly (which means short-term dependency with context length \( c \) is not sufficient), so given a ratio parameter \( r \), we define the set \( T \) is the above equation as

\[
T = \text{top}-r \text{ positions } t \text{ with largest } b(c, t)
\]

The relative gain is subsequently defined as the relative perplexity reduction:

\[
g_i(c, c') = \frac{\exp f_i(c, c) - \exp f_i(c, c')}{\exp f_i(c, c)}
\]

Given a step size \( \Delta \), we then use an algorithm to find the RECL by thresholding the relative gain:

1. Set initial short context length \( c \), and long context length \( c' = c + \Delta \)
2. Compute \( g_i(c, c') \). If \( g_i(c, c') < 0.01 \), return RECL = \( c \). If \( g_i(c, c') \geq 0.01 \), set \( c = c' \), \( c' = c + \Delta \) and go to step 1.

In Figure 3, we visualize the unnormalized relative perplexity gains \( \exp f_i(c, c) - \exp f_i(c, c') \) with various pairs of \( (c, c') \) when \( r = 0.1 \). It is clear that Transformer-XL has a longer RECL compared to RNNs and other baselines because the relative gains are substantially larger.

For reference, we plot the perplexities with varying context lengths in Figure 4. The y-axis denotes the “normal” perplexity (not calibrated by baselines).

D Attention Visualization

In this section, we provide some visualization of the attention learned by the SoTA model on the WikiText-103 validation set. Recall that, this model has 16 10-head transformer layers and relies on a memory of length 640.

![Figure 5](image_url)

Figure 5: Average attention over the previous 640 tokens, where each row corresponds to a attention head and each column corresponds to a relative location. There are totally 160 attention heads, and every 10 heads come from a single layer. Darker colors indicate higher values.

The first visualization aims at revealing the overall trend of where the model is attending. Specifically, for each attention head of each layer, we average the attention distributions of all tokens in the validation set. This is shown in Fig. 5. As we can see, the overall trend is to focus more on the nearby tokens than the faraway ones. However, it is also very clear that some attention heads have a wider attention distribution over the entire memory span, notably the head 8 from layer 1, head 78 from layer 8, and the head 158 from layer 16.

Since we are focused on learning long-range dependency, we are especially interested in these heads with a wider attention span. Thus, in the second set of visualization, we pick the three notable heads mentioned above, and visualize their attention behavior for a randomly chosen position, as shown in Fig. 6. Here, we see three different patterns of wider attention:

- For the head 8 in the 1st layer, we see an almost uniform attention over the entire memory span. This is quite intuitive, as lower-level layers needs to screen the entire memory span to decide where to focus for higher-level layers
• For the head 78 in the 8th layer (a middle-level layer), we see a very sparse attention pattern scattered in all ranges of the memory. Again, this well fits our intuition that as information accumulates, the network may focus on some particular position with special interests.

• For the head 158 in the 16th layer (i.e. the last layer), each target location (corresponding to each row) has its own distinct sparse focus, differing from head 78 where target locations largely share the same attentive location in memory. Meanwhile, the pattern is also different from the case of head 8, where a few locations are clearly attended more than others.

Finally, as we have discussed in section 3.3, the attention score can be decomposed into four intuitive terms. Here, we want to further investigate how these four terms contribute to the overall attention trend in Fig. 5. Since the term (c) represents the global content bias, i.e., the prior importance of each word regardless of the context, we will leave it out and focus on the terms (a), (b) and (d). So, for each term, we take the Softmax w.r.t. the memory span and average the resulted distribution of all tokens in the validation set. The results are visualized in Fig. 7:

• Since term (a) is fully content-based addressing, when averaging over all target words, the result is essentially uniform over the entire context, except for a few very close words, which are likely to be semantically similar to the target word.

• The overall trend of term (b) highly resembles that of the entire attention distribution in Fig. 5. It suggests that the global trend of focusing on the nearby context is largely contributed by this content-dependent positional bias.

• The overall trend of term (d) is also focusing more on nearby words. However, compared to the trend of term (b), it is clearly flatter and biases towards a longer context.
Figure 7: Visualization of the three terms in computing the attention score. Each row corresponds to an attention head and each column corresponds to a relative location.

### E Generated Text

In this section, we present some generated text from our best model trained the Wikitext-103 dataset. We seed our Transformer-XL with a context of at most 512 consecutive tokens randomly sampled from the test set of Wikitext-103. Then, we run Transformer-XL to generate a *pre-defined* number of tokens (500 or 1,000 in our case). For each generation step, we first find the top-40 probabilities of the next-step distribution and sample from top-40 tokens based on the re-normalized distribution. To help reading, we detokenize the context, the generated text and the reference text. Three generated examples are shown in Tables 11, 12, and 13. Note that we do not perform any cherry picking and present the first three examples we generate in the paper. In the text, “= text =”, “== text ==” and “=== text ===” denote the Wikipedia page title, section title and subsection title, respectively, due to the original data preprocessing procedure of Wikitext-103 (Merity et al., 2016).

As we can see, though only trained on 100M tokens, Transformer-XL is a strong model at generating long text articles, particularly in the following aspects:

- Transformer-XL is able to structurally maintain the sectional arrangement of Wikipedia.
- Transformer-XL manages to semantically stay on the same topic throughout the course of generation.
- Long-range references are common in the generated text.
- Transformer-XL often generates novel content that is not present in the training data.

For more detailed explanation of the interesting observations in each example, please refer to the corresponding caption.
Despite the overall excellence of the generation quality, the model can only perceive the seed context and hallucinate what to generate based on the limited knowledge (100M tokens only) it is trained on. As a result, the generated text sometimes looks clearly relevant but not close enough or to the point compared to what human writer would do. That said, we believe this issue is mostly a problem of limited training data size and could be alleviated by using a larger training set.

Context:
Kershaw started the 2010 season by posting a 3.07 ERA in April, but did so by walking 22 batters in 29 innings. On May 4, he had his worst start of his career against the Milwaukee Brewers at Dodger Stadium, throwing just 57 pitches in 11/3 innings, while retiring only four of the 13 batters he faced — including the pitcher. He was booed loudly upon being pulled from the game. Kershaw said after the game, “I didn’t give our team any kind of chance. It’s just not a good feeling to let your teammates down, let everybody down. It stings, it hurts. I’ve got to figure things out.” Kershaw rebounded his next start by pitching an 8 inning two-hitter and out-dueling the then undefeated Ubaldo Jiménez. He credited his control of the slider being the major turning point for him. Later in the season, he was suspended for five games after hitting Aaron Rowand of the Giants with a pitch in a game on July 20. The incident occurred after both teams were given a warning following Giants ace Tim Lincecum hitting Matt Kemp earlier in the game. He threw his first career complete game shutout on September 14, 2010 also against San Francisco and finished the season with a record of 13–10 and a 2.59 ERA in 32 starts, pitching 204/3 innings and recording 212 strikeouts.

Example 1 – 500 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The training data size and could be alleviated by using a larger training set.

After finishing the 2010 season strong, the Dodgers named Kershaw as the Opening Day Starter for the 2011 season. On May 29, he pitched the second complete-game shutout of his career, striking out 10 while winning a two-hitter against the Florida Marlins, 8–0; he also had two singles and an RBI, scoring twice in the game. He produced his third career shutout on June 20, a two-hit, 11-strikeout effort against the Detroit Tigers. Kershaw became the first Dodgers starter to strike out the side in the 9th inning since Sandy Koufax’s perfect game. In his next start, on June 26, Kershaw pitched another complete game (against the Los Angeles Angels of Anaheim). He became the first Dodger starter to have back-to-back complete game victories since Jeff Weaver in the 2005 season and the first Dodger to have double-digit strikeouts in consecutive starts since Chan-Ho Park in the 2000 season. He was awarded the National League Player of the Week award for the week of June 20 – 26 as a result of those two starts. Midway through June, Kershaw had amassed 32 career victories

Reference:

XL Generation:

Context:

On May 17, 2013, Kershaw sustained another back injury during spring training, pertaining to the seed context and hallucinating the experience of Kershaw in Royals across the generated text.

On November 17, he was honored with the National League Cy Young Award, the NL and the Sporting News (TSN) National League Pitcher of the Year Award, and the Rawlings Gold Glove Award as the top fielding pitcher in the National League by the Baseball Writers’ Association of America. In a game against the Reds on September 14, 2013, Kershaw pitched his 20th game of the season and became the first pitcher in the history of Major League Baseball to have pitched 20 complete games in a single season.

Table 11: Example 1 – 500 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The sample is randomly generated without any cherry picking.

Original Wikipedia page: https://en.wikipedia.org/wiki/Clayton_Kershaw

There are many interesting observations from this example:

- Firstly, Kershaw never went to Royals in real life. Despite that, Transformer-XL stays on the fully imagined topic and keeps hallucinating the experience of Kershaw in Royals across the generated text.

- Secondly, notice that XL correctly tracks the chronological order from 2011 to 2012 and to the finally 2013 season in the section titles.

- In addition, notice that Transformer-XL accurately uses the the phrase “another back injury” in the 2013 season paragraph, since it has talked about one earlier injury in the 2012 season. This shows again Transformer-XL’s ability of capturing long-term dependency.
This demonstrates a degree of generalization instead of memorization. Please refer to the original book content at the end of the page, as marked by the single "=" line. This suggests the model has the ability of identifying the end of a topic/page, and randomly starting with a new topic.

Even more interestingly, a newly-started page is on a book called "The Tale of Mrs. Tittlemouse". Transformer-XL manages to copy the same book title and some related information from the training set, but hallucinates novel content of the book. This demonstrates a degree of generalization instead of memorization. Please refer to the original book content at the Wikipedia page.

Table 12: Example 2 – 500 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The sample is randomly generated without any cherry picking.

This example exhibit some additional interesting properties of Transformer-XL:

- After finishing the last paragraph of the seed context, both the reference and generated text start a new topic (i.e., Wikipedia page), as marked by the single "=" line. This suggests the model has the ability of identifying the end of a topic/page, and randomly starting with a new topic.

- Even more interestingly, a newly-started page is on a book called “The Tale of Mrs. Tittlemouse”. Transformer-XL manages to copy the same book title and some related information from the training set, but hallucinates novel content of the book. This demonstrates a degree of generalization instead of memorization. Please refer to the original book content at the Wikipedia page.

Reference:

- The Heart of Ezra Greer is a 1917 American silent drama film produced by the Thanhouser Company and directed by Emile <unk>. The film focuses on Ezra Greer, a successful middle-aged man who searches for his college age daughter, Mary. The film was released on 13 November 1910 and was accompanied by a greeting card by Potter's brother-in-law, Frederick Warne. The tale was well received by critics and children, and sold over 600,000 copies in its first few years. In the first year of its publication, Potter’s ‘The Tale of Mrs. Tittlemouse’ won children’s book awards in England, America and Australia. It was selected as a “recommended reading” by children in the US and Canada.

Usage:

- The film follows Ezra Greer, a middle-aged man who has worked hard since his youth. He cares deeply for his motherless daughter, Mary, but was unable to attend the annual commencement at her co-educational college. He is advised to find Mary by making the case for his own daughter to return from college, but Mary leaves with her romantic interest, Jack. After a carriage accident in which the baby is injured, Ezra and Jack rush to the hospital and find Mary as a nurse crying over the child. The film ends with the marriage of Jack and Mary. The film was released by Pathé on October 7, 1917. The film was the final release from Thanhouser and was deemed to be an average film by most reviewers. Criticism for the film hinged on far-fetched coincidences to drive the plot. The film is presumed lost.

XL Generation:

- 700,000 in 2013 on the black market. In 2004 the plant was reported to be responsible for a net profit of around US $ 3 million in Brazil. A 2012 study found A. speciosa to cause agricultural production at least 2 – 3 times greater than those achieved without prior knowledge of land management.

Context:

- Species range across the Neotropics from Mexico in the north to Bolivia, Paraguay, and southern Brazil in the south. According to <unk> and coauthors, three species are found in Mexico, four in Central America, and 62 in South America. Three species are present in the Caribbean — two in Trinidad and Tobago, along the southern edge of the region, and one in Haiti.

- = Distribution =

- Species range across the Neotropics from Mexico in the north to Bolivia, Paraguay, and southern Brazil in the south. According to <unk> and coauthors, three species are found in Mexico, four in Central America, and 62 in South America. Three species are present in the Caribbean — two in Trinidad and Tobago, along the southern edge of the region, and one in Haiti.

- = Habitat and ecology =

- <unk> includes both large trees and small acaulescent palms which occupy a number of different ecological niches. Dense stands of some of the larger species are conspicuous elements on the landscape, while smaller species are found in both in the forest understory and in savannas. Disturbance has been implicated in the formation of vegetation dominated by large <unk> species. In seasonally dry Amazonian forests the density of large adult A. <unk> palms was correlated with canopy openness; the species also dominates savannas formed by repeated forest fires in Trinidad and Tobago. A. <unk> speciosa forms pure stands in many parts of Brazil where natural forest vegetation has been cleared. Similarly, stands of A. <unk> in Bahia, Brazil (which are cultivated for <unk> fibre) are managed using fire — the seedlings survive cutting and burning, and are able to dominate burned forest patches. The fruit are dispersed by animals; fruit which are not dispersed frequently suffer seed predation by <unk> beetles. Certain species of <unk> have been mentioned as examples of "anachronistic" species which are adapted for dispersal by now-extinct Pleistocene megafauna. On <unk> Island, <unk>, in the Brazilian Amazon, <unk> <unk> fruit were consumed by tapiro, collared peccaries, deer and primates. Rodents, including agoutis, fed upon the fruit and, as the fruit availability declined, they fed on the seeds. Other dispersers of <unk> fruit include Crested <unk> which consume the fruit and disperse the seeds of A. <unk> in the Brazilian Pantanal.

- = Uses =

- <unk> species have a long history of human utilisation. <unk> <unk> <unk> <unk> seeds have been found in archaeological sites in Colombia dating back to 9000 BP. A variety of species remain important sources of edible oil, thatch, edible seeds and fibre. The leaves of <unk> <unk> and A. <unk> are used extensively for thatching. Several species are oil palms, with A. speciosa among the most important economically. Products extracted from A. speciosa were reported to support over 300,000 households in the Brazilian state of Maranhão in 2005, and in 1985 it was estimated to support over 450,000 households throughout the Brazil. <unk> fibres, extracted from the leaf bases of A. <unk>, are commercially important, and generated about US $
The Battle of Dürrenstein (also known as the Battle of <unk> or Battle of <unk> and Battle of <unk>), German: <unk> bei <unk>), on 11 November 1805 was an engagement in the Napoleonic Wars during the War of the Third Coalition. Dürrenstein (modern <unk>) is located in the <unk> Valley, on the River Danube, 73 kilometers (45 mi) upstream from Vienna, Austria. The river makes a sharp bend there, and it was fought in the flood plain between the river and the mountains. At Dürrenstein a combined force of Russian and Austrian troops trapped a French division commanded by Théodore Maxime Gazan. The French division was part of the newly created VIII Corps, the so-called Corps Mortier, under command of Édouard Mortier. In pursuing the Austrian retreat from Bavaria, Mortier had over-extended his three divisions along the north bank of the Danube. Mikhail Kutuzov, commander of the Coalition force, enticed Mortier to send Gazan's division into a trap and French troops were caught in a valley between two Russian columns. They were rescued by the timely arrival of a second division, under command of Pierre Dupont de l'Étang. The battle extended well into the night. Both sides claimed victory. The French lost more than a third of their participants, and Gazan's division experienced over 40 percent losses. The Austrians and Russians also lost over 16 percent of their troops. Perhaps the most significant was the death in the French staff. The battle was fought three weeks after the Austrian capitulation at Ulm and three weeks before the Russo-Austrian defeat at the Battle of Austerlitz. After Austerlitz Austria withdrew from the war. The French demanded a high indemnity and Francis II abdicated as Holy Roman Emperor, releasing the German states from their allegiance to the Holy Roman Empire.

In a series of conflicts from 1803-15 known as the Napoleonic Wars, various European powers formed five coalitions against the First French Empire. Like the wars sparked by the French Revolution (1789), these further revolutionized the formation, organization and training of European armies and led to an unprecedented militarization, mainly due to mass conscription. Under the leadership of Napoleon, French power rose quickly as the Grande Armée conquered most of Europe, and collapsed rapidly.
French approached Vienna, the Prussians had already surrendered. As the Austrians did not want to allow the war to continue, they decided to abandon their territories in the north and move their army to the north and west, cutting off Charles from Vienna. The Battle of Warsaw was fought on 23 November 1805 between the French Army of the Danube and the Austrian Army of Styria in the vicinity of Warsaw and Pressburg (modern Trnava, Slovakia). At that time Habsburg forces reached the top of the hill at bayonet point. The Austrian cavalry unsuccessfully tried to fend off the French, but the Austrian infantry broke and ran. In this engagement alone, the Austrians lost more than half their reserve artillery park, 6,000 (out of 8,000 total participants) dead, wounded or captured and four colors. Reisch’s column also failed to destroy the bridges across the Danube. Napoleon’s lightning campaign exposed the Austrian indecisive command structure and poor supply apparatus. Mack

Table 13: Example 3 – 1,000 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The sample is randomly generated without any cherry picking.

Original Wikipedia page: https://en.wikipedia.org/wiki/Battle_of_D%C3%BCrenstein.

- Although this example is significantly longer, we can see that Transformer-XL is still able to stay on the same topic and makes up non-existing stories about the Napoleon wars.
- Notably, from the second section on, the generated text correctly follows a fine-grained chronological order on the level of month and day to narrate events in 1805, except a mistake (1804 instead of 1805) near the end of the paragraph. To ease reading which we have highlighted all the date related phrases by magenta in the generation.