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

Zihang Dai[*12], **Zhilin Yang**[*12], **Yiming Yang**[1], **Jaime Carbonell**[1],
Quoc V. Le[2], **Ruslan Salakhutdinov**[1]
[1]Carnegie Mellon University, [2]Google 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 PyTorch[1].

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

[*]Equal contribution. Order determined by swapping the one in Yang et al. (2017).

[1]https://github.com/kimiyoung/transformer-xl

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.

2 Related Work

In the last few years, the field of language modeling has witnessed many significant advances, including but not limited to devising novel architectures to better encode the context (Bengio et al., 2003; Mikolov et al., 2010; Merity et al., 2016; Al-Rfou et al., 2018), improving regularization and optimization algorithms (Gal and Ghahramani, 2016) , speeding up the Softmax computation (Grave et al., 2016a) , and enriching the output distribution family (Yang et al., 2017).

To capture the long-range context in language modeling, a line of work directly feeds a representation of the wider context into the network as an additional input. Existing works range from ones where context representations are manually defined (Mikolov and Zweig, 2012; Ji et al., 2015; Wang and Cho, 2015) to others that rely on document-level topics learned from data (Dieng et al., 2016; Wang et al., 2017).

More broadly, in generic sequence modeling, how to capture long-term dependency has been a long-standing research problem. From this perspective, since the ubiquitous adaption of LSTM, many efforts have been spent on relieving the vanishing gradient problem, including better initialization (Le et al., 2015), additional loss signal (Trinh et al., 2018), augmented memory structure (Ke et al., 2018) and others that modify the internal architecture of RNNs to ease the optimization (Wu et al., 2016; Li et al., 2018). Different from them, our work is based on the Transformer architecture and shows that language modeling as a real-world task benefits from the ability to learn longer-term dependency.

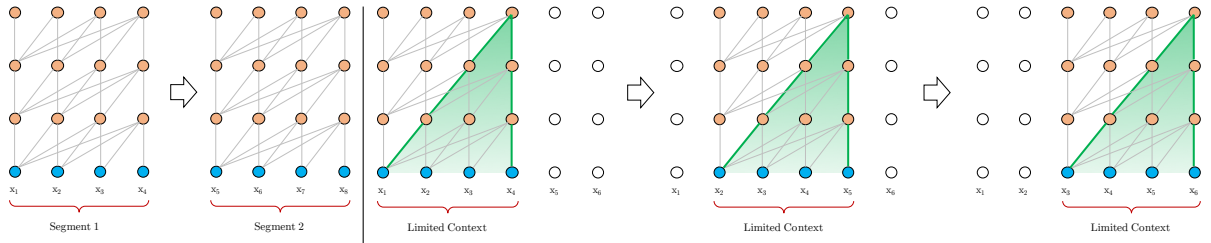
3 Model

Given a corpus of tokens $\mathbf{x} = (x_1, \ldots, x_T)$, the task of language modeling is to estimate the joint probability $P(\mathbf{x})$, which is often auto-regressively factorized as $P(\mathbf{x}) = \prod_t P(x_t \mid \mathbf{x}_{<t})$. 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 $\mathbf{x}_{<t}$ 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-



(a) Train phase. (b) Evaluation phase.

Figure 1: Illustration of the vanilla model with a segment length 4.

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.

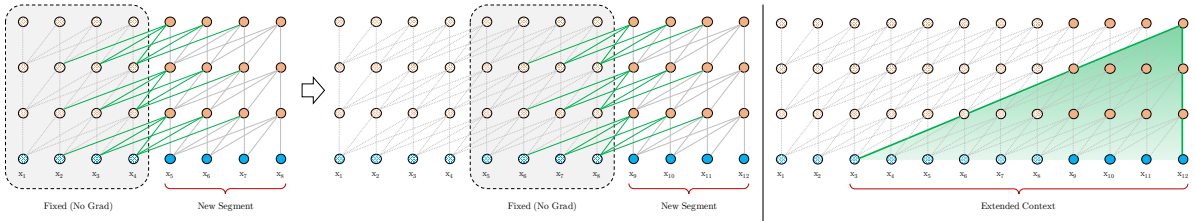
3.2 Segment-Level Recurrence with State Reuse

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 $\mathbf{s}_\tau = [x_{\tau,1}, \cdots, x_{\tau,L}]$ and $\mathbf{s}_{\tau+1} = [x_{\tau+1,1}, \cdots, x_{\tau+1,L}]$ respectively. Denoting the n-th layer hidden state sequence produced for the τ-th segment $\mathbf{s}_\tau$ by $\mathbf{h}_\tau^n \in \mathbb{R}^{L\times d}$, where d is the hidden dimension. Then, the n-th layer hidden state for segment $\mathbf{s}_{\tau+1}$ is produced (schematically) as follows,

$$\widetilde{\mathbf{h}}_{\tau+1}^{n-1} = \left[\text{SG}(\mathbf{h}_\tau^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}\right],$$
$$\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n = \mathbf{h}_{\tau+1}^{n-1}\mathbf{W}_q^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_k^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_v^\top,$$
$$\mathbf{h}_{\tau+1}^n = \text{Transformer-Layer}\left(\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n\right).$$

where the function SG$(\cdot)$ stands for stop-gradient, the notation $[\mathbf{h}_u \circ \mathbf{h}_v]$ indicates the concatenation of two hidden sequences along the length dimension, and $\mathbf{W}_\cdot$ denotes model parameters. Compared to the standard Transformer, the critical difference lies in that the key $\mathbf{k}_{\tau+1}^n$ and value $\mathbf{v}_{\tau+1}^n$ are conditioned on the extended context $\widetilde{\mathbf{h}}_{\tau+1}^{n-1}$ and hence $\mathbf{h}_\tau^{n-1}$ 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 $\mathbf{h}_{\tau+1}^n$ and $\mathbf{h}_\tau^{n-1}$ shifts one layer downwards



(a) Training phase. (b) Evaluation phase.

Figure 2: Illustration of the Transformer-XL model with a segment length 4.

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 predefined length-M old hidden states spanning (possibly) multiple segments, and refer to them as the memory $\mathbf{m}_\tau^n \in \mathbb{R}^{M \times d}$, 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 $\mathbf{U} \in \mathbb{R}^{L_{\max} \times d}$, where the i-th row $\mathbf{U}_i$ corresponds to the i-th *absolute* position within a segment and $L_{\max}$ 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

$$\mathbf{h}_{\tau+1} = f(\mathbf{h}_\tau, \mathbf{E}_{\mathbf{s}_{\tau+1}} + \mathbf{U}_{1:L})$$
$$\mathbf{h}_\tau = f(\mathbf{h}_{\tau-1}, \mathbf{E}_{\mathbf{s}_\tau} + \mathbf{U}_{1:L}),$$

where $\mathbf{E}_{\mathbf{s}_\tau} \in \mathbb{R}^{L \times d}$ is the word embedding sequence of $\mathbf{s}_\tau$, and f represents a transformation function. Notice that, both $\mathbf{E}_{\mathbf{s}_\tau}$ and $\mathbf{E}_{\mathbf{s}_{\tau+1}}$ are associated with the same positional encoding $\mathbf{U}_{1:L}$. As a result, the model has no information to distinguish the positional difference between $x_{\tau,j}$ and $x_{\tau+1,j}$ 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 $q_{\tau,i}$ attends on the key vectors $\mathbf{k}_{\tau,\leq i}$, 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 $k_{\tau,j}$ and itself $q_{\tau,i}$, i.e. $i-j$. Practically, one can create a set of relative posi-

tional encodings $\mathbf{R} \in \mathbb{R}^{L_{\max} \times d}$, where the i-th row $\mathbf{R}_i$ indicates a relative distance of i between two positions. By injecting the relative distance dynamically into the attention score, the query vector can easily distinguish the representations of $x_{\tau,j}$ and $x_{\tau+1,j}$ 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

$$\mathbf{A}^{\text{abs}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.$$

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

$$\mathbf{A}^{\text{rel}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.$$

- The first change we make is to replace all appearances of the absolute positional embedding $\mathbf{U}_j$ for computing key vectors in term (b) and (d) with its relative counterpart $\mathbf{R}_{i-j}$. This essentially reflects the prior that only the relative distance matters for where to attend. Note that $\mathbf{R}$ 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 $\mathbf{U}_i^\top \mathbf{W}_q^\top$ 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 $\mathbf{U}_i^\top \mathbf{W}_q^\top$ in term (d).
- Finally, we deliberately separate the two weight matrices $\mathbf{W}_{k,E}$ and $\mathbf{W}_{k,R}$ 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 $\mathbf{W}_k \mathbf{R}$ into a single trainable matrix $\hat{\mathbf{R}}$, which abandons the inductive bias built into the original sinusoid positional encoding (Vaswani et al., 2017). In contrast, our relative positional embedding $\mathbf{R}$ 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 $n = 1, \ldots, N$:

$$\begin{aligned}
\widetilde{\mathbf{h}}_\tau^{n-1} &= \left[\text{SG}(\mathbf{m}_\tau^{n-1}) \circ \mathbf{h}_\tau^{n-1}\right] \\
\mathbf{q}_\tau^n, \mathbf{k}_\tau^n, \mathbf{v}_\tau^n &= \mathbf{h}_\tau^{n-1} \mathbf{W}_q^{n\top}, \widetilde{\mathbf{h}}_\tau^{n-1} \mathbf{W}_{k,E}^{n\top}, \widetilde{\mathbf{h}}_\tau^{n-1} \mathbf{W}_v^{n\top} \\
\mathbf{A}_{\tau,i,j}^n &= \mathbf{q}_{\tau,i}^{n\top} \mathbf{k}_{\tau,j}^n + \mathbf{q}_{\tau,i}^{n\top} \mathbf{W}_{k,R}^n \mathbf{R}_{i-j} \\
&\quad + u^\top \mathbf{k}_{\tau,j}^n + v^\top \mathbf{W}_{k,R}^n \mathbf{R}_{i-j} \\
\mathbf{a}_\tau^n &= \text{Masked-Softmax}(\mathbf{A}_\tau^n) \mathbf{v}_\tau^n \\
\mathbf{o}_\tau^n &= \text{LayerNorm}(\text{Linear}(\mathbf{a}_\tau^n) + \mathbf{h}_\tau^{n-1}) \\
\mathbf{h}_\tau^n &= \text{Positionwise-Feed-Forward}(\mathbf{o}_\tau^n)
\end{aligned}$$

with $\mathbf{h}_\tau^0 := \mathbf{E}_{\mathbf{s}_\tau}$ defined as the word embedding sequence. In addition, it is worth mentioning that a naive way to compute $\mathbf{A}$ requires computing $\mathbf{W}_{k,R}^n \mathbf{R}_{i-j}$ 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

Model	#Param	PPL
Grave et al. (2016b) - LSTM	-	48.7
Bai et al. (2018) - TCN	-	45.2
Dauphin et al. (2016) - GCNN-8	-	44.9
Grave et al. (2016b) - LSTM + Neural cache	-	40.8
Dauphin et al. (2016) - GCNN-14	-	37.2
Merity et al. (2018) - QRNN	151M	33.0
Rae et al. (2018) - Hebbian + Cache	-	29.9
Ours - Transformer-XL Standard	151M	**24.0**
Baevski and Auli (2018) - Adaptive Input$^\diamond$	247M	20.5
Ours - Transformer-XL Large	257M	**18.3**

Table 1: Comparison with state-of-the-art results on WikiText-103. $^\diamond$ indicates contemporary work.

Model	#Param	bpc
Ha et al. (2016) - LN HyperNetworks	27M	1.34
Chung et al. (2016) - LN HM-LSTM	35M	1.32
Zilly et al. (2016) - RHN	46M	1.27
Mujika et al. (2017) - FS-LSTM-4	47M	1.25
Krause et al. (2016) - Large mLSTM	46M	1.24
Knol (2017) - cmix v13	-	1.23
Al-Rfou et al. (2018) - 12L Transformer	44M	1.11
Ours - 12L Transformer-XL	41M	**1.06**
Al-Rfou et al. (2018) - 64L Transformer	235M	1.06
Ours - 18L Transformer-XL	88M	1.03
Ours - 24L Transformer-XL	277M	**0.99**

Table 2: Comparison with state-of-the-art results on enwik8.

Model	#Param	bpc
Cooijmans et al. (2016) - BN-LSTM	-	1.36
Chung et al. (2016) - LN HM-LSTM	35M	1.29
Zilly et al. (2016) - RHN	45M	1.27
Krause et al. (2016) - Large mLSTM	45M	1.27
Al-Rfou et al. (2018) - 12L Transformer	44M	1.18
Al-Rfou et al. (2018) - 64L Transformer	235M	1.13
Ours - 24L Transformer-XL	277M	**1.08**

Table 3: Comparison with state-of-the-art results on text8.

Model	#Param	PPL
Shazeer et al. (2014) - Sparse Non-Negative	33B	52.9
Chelba et al. (2013) - RNN-1024 + 9 Gram	20B	51.3
Kuchaiev and Ginsburg (2017) - G-LSTM-2	-	36.0
Dauphin et al. (2016) - GCNN-14 bottleneck	-	31.9
Jozefowicz et al. (2016) - LSTM	1.8B	30.6
Jozefowicz et al. (2016) - LSTM + CNN Input	1.04B	30.0
Shazeer et al. (2017) - Low-Budget MoE	~5B	34.1
Shazeer et al. (2017) - High-Budget MoE	~5B	28.0
Shazeer et al. (2018) - Mesh Tensorflow	4.9B	24.0
Baevski and Auli (2018) - Adaptive Input$^\diamond$	0.46B	24.1
Baevski and Auli (2018) - Adaptive Input$^\diamond$	1.0B	23.7
Ours - Transformer-XL Base	0.46B	23.5
Ours - Transformer-XL Large	0.8B	**21.8**

Table 4: Comparison with state-of-the-art results on One Billion Word. $^\diamond$ indicates contemporary work.

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.

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†	24M	57.3
Yang et al. (2017) - MoS+Finetune†	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

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	**27.02**	**26.77**	**500**
-	✓	Shaw et al. (2018)	Full	27.94	27.94	256
-	✓	Ours	Half	28.69	28.33	460
-	✗	Ours	Full	29.59	29.02	260
-	✗	Ours	Half	30.10	30.10	120
-	✗	Shaw et al. (2018)	Full	29.75	29.75	120
-	✗	Shaw et al. (2018)	Half	30.50	30.50	120
-	✗	Vaswani et al. (2017)	Half	30.97	30.97	120
Transformer (128M)†	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
Transformer-XL (151M)	✓	Ours	Full	23.43	**23.09**	**640**
					23.16	450
					23.35	300

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.

Method	PPL
Ours	**25.2**
With Shaw et al. (2018) encodings	25.7
Without recurrence	27.1

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

Model	$r = 0.1$	$r = 0.5$	$r = 1.0$
Transformer-XL 151M	**900**	**800**	**700**
QRNN	500	400	300
LSTM	400	300	200
Transformer-XL 128M	**700**	**600**	**500**
- use Shaw et al. (2018) encoding	400	400	300
- remove recurrence	300	300	300
Transformer	128	128	128

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.

called *Relative Effective Context Length* (RECL). RECL is defined on a model group instead of a single model, and the gain of a long context is measure by the relative improvement over the *best* short context model. As such, the model group shares the same baseline to enable fair comparison. RECL also has a parameter r, which means constraining the comparison on top-r hard examples. See Appedix C for more details about RECL. As shown in Table 8, Transformer-XL manages to model dependency of 900 words long on average with $r = 0.1$. The RECL of Transformer-XL is 80% and 450% longer than recurrent networks and Transformer respectively. Both the recurrence mechanism and our positional encodings contribute to a longer RECL. This further substantiates our argument that Transformer-XL is able to model longer-term dependency.

Attn Len	How much Al-Rfou et al. (2018) is slower
3,800	1,874x
2,800	1,409x
1,800	773x
800	363x

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

4.4 Generated Text

Trained only on WikiText-103 which is medium-sized, Transformer-XL is already able to generate relatively coherent articles with thousands of tokens without manual cherry picking, despite minor flaws. Please refer to Appendix E for samples.

4.5 Evaluation Speed

Finally, we compare the evaluation speed of our model with the vanilla Transformer model (Al-Rfou et al., 2018). As shown in Table 9, due to the state reuse scheme, Transformer-XL achieves an up to 1,874 times speedup during evaluation.

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 Llion 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:1612.08083*.

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 B 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 recurrent 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*.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. *arXiv preprint arXiv:1511.03962*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Sekitoshi Kanai, Yasuhiro Fujiwara, Yuki Yamanaka, and Shuichi Adachi. 2018. Sigsoftmax: Reanalysis of the softmax bottleneck. *arXiv preprint arXiv:1805.10829*.

Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. 2018. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pages 7650–7661.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Bryon Knol. 2017. cmix v13. http://www.byronknoll.com/cmix.html.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork rnn. *arXiv preprint arXiv:1402.3511*.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. 2016. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. 2018. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5457–5466.

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

MultiMedia LLC. 2009. Large text compression benchmark.

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*.

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*.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

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 and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. *SLT*, 12(234-239):8.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Asier Mujika, Florian Meier, and Angelika Steger. 2017. Fast-slow recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 5915–5924.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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*.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

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	pplx best	pplx 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.

Table 10 compares Transformer-XL with baseline under the same memory budget. Transformer-XL still outperforms the baseline even with a shorter backprop length.

B Efficient Computation of the Attention with Relative Positional Embedding

As we discussed in section 3.3, the naive way of computing the $\mathbf{W}_{k,R}\mathbf{R}_{i-j}$ 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

$$\mathbf{Q} := \begin{bmatrix} \mathbf{R}_{M+L-1}^\top \\ \mathbf{R}_{M+L-2}^\top \\ \vdots \\ \mathbf{R}_1^\top \\ \mathbf{R}_0^\top \end{bmatrix} \mathbf{W}_{k,R}{}^\top = \begin{bmatrix} [\mathbf{W}_{k,R}\mathbf{R}_{M+L-1}]^\top \\ [\mathbf{W}_{k,R}\mathbf{R}_{M+L-2}]^\top \\ \vdots \\ [\mathbf{W}_{k,R}\mathbf{R}_1]^\top \\ [\mathbf{W}_{k,R}\mathbf{R}_0]^\top \end{bmatrix} \in \mathbb{R}^{(M+L)\times d}$$

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

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

$$\mathbf{B} = \begin{bmatrix} q_0^\top \mathbf{W}_{k,R}\mathbf{R}_M & \cdots & q_0^\top \mathbf{W}_{k,R}\mathbf{R}_0 & 0 & \cdots & 0 \\ q_1^\top \mathbf{W}_{k,R}\mathbf{R}_{M+1} & \cdots & q_1^\top \mathbf{W}_{k,R}\mathbf{R}_1 & q_1^\top \mathbf{W}_{k,R}\mathbf{R}_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{M+L-1} & \cdots & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{M+L-1} & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{L-1} & \cdots & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_0 \end{bmatrix}$$

$$= \begin{bmatrix} q_0^\top \mathbf{Q}_{L-1} & \cdots & q_0^\top \mathbf{Q}_{M+L-1} & 0 & \cdots & 0 \\ q_1^\top \mathbf{Q}_{L-2} & \cdots & q_1^\top \mathbf{Q}_{M+L-2} & q_1^\top \mathbf{Q}_{M+L-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{Q}_0 & \cdots & q_{L-1}^\top \mathbf{Q}_M & q_{L-1}^\top \mathbf{Q}_{M+1} & \cdots & q_{L-1}^\top \mathbf{Q}_{M+L-1} \end{bmatrix}$$

Then, we further define

$$\widetilde{\mathbf{B}} = \mathbf{q}\mathbf{Q}^\top = \begin{bmatrix} q_0^\top \mathbf{Q}_0 & \cdots & q_0^\top \mathbf{Q}_M & q_0^\top \mathbf{Q}_{M+1} & \cdots & q_0^\top \mathbf{Q}_{M+L-1} \\ q_1^\top \mathbf{Q}_0 & \cdots & q_1^\top \mathbf{Q}_M & q_1^\top \mathbf{Q}_{M+1} & \cdots & q_1^\top \mathbf{Q}_{M+L-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{Q}_0 & \cdots & q_{L-1}^\top \mathbf{Q}_M & q_{L-1}^\top \mathbf{Q}_{M+1} & \cdots & q_{L-1}^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

Now, it is easy to see an immediate relationship between $\mathbf{B}$ and $\widetilde{\mathbf{B}}$, where the i-th row of $\mathbf{B}$ is simply a left-shifted version of i-th row of $\widetilde{\mathbf{B}}$. Hence, the computation of $\mathbf{B}$ only requires a matrix multiplication $\mathbf{q}\mathbf{Q}^\top$ to compute $\widetilde{\mathbf{B}}$ 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 $\mathbf{D}$,

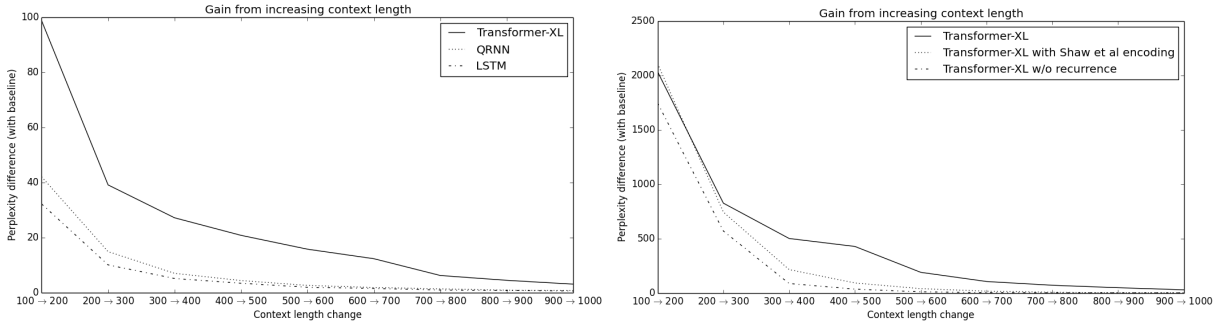
$$\mathbf{D} = \begin{bmatrix} v^\top \mathbf{Q}_{L-1} & \cdots & v^\top \mathbf{Q}_{M+L-1} & 0 & \cdots & 0 \\ v^\top \mathbf{Q}_{L-2} & \cdots & v^\top \mathbf{Q}_{M+L-2} & v^\top \mathbf{Q}_{M+L-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v^\top \mathbf{Q}_0 & \cdots & v^\top \mathbf{Q}_M & v^\top \mathbf{Q}_{M+1} & \cdots & v^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

Then, we can follow the same procedure to define

$$\widetilde{\mathbf{d}} = [\mathbf{Q}v]^\top = \begin{bmatrix} v^\top \mathbf{Q}_0 & \cdots & v^\top \mathbf{Q}_M & v^\top \mathbf{Q}_{M+1} & \cdots & v^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

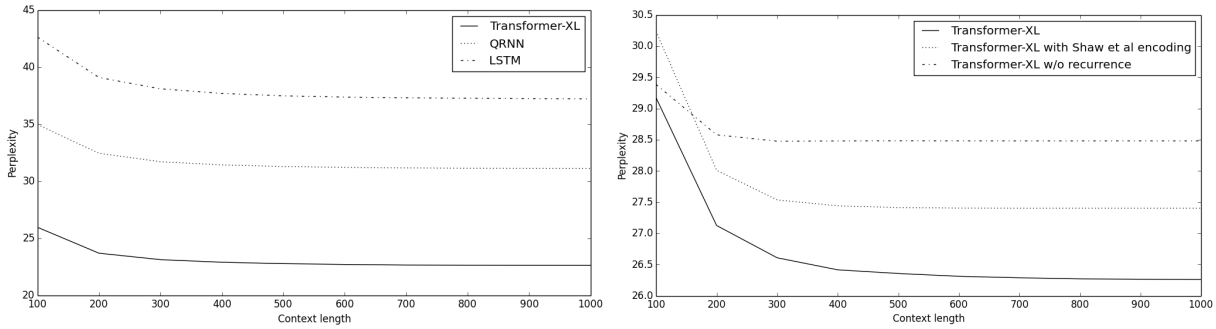
Again, each row of $\mathbf{D}$ is simply a left-shift version of $\widetilde{\mathbf{d}}$. Hence, the main computation cost comes from the matrix-vector multiplication $\widetilde{\mathbf{d}} = [\mathbf{Q}v]^\top$, which is not expensive any more.

C Details About RECL



(a) Transformer-XL vs RNNs (b) Transformer-XL vs Baseline

Figure 3: Visualizing unnormalized relative perplexity gains with $r = 0.1$.



(a) Transformer-XL vs RNNs (b) Transformer-XL vs Baseline

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

$$f_i(c, c') = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \min \big(b(c, t), l_i(c', t) \big)$$

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 $\mathcal{T}$ is the above equation as

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

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 Δ, 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.

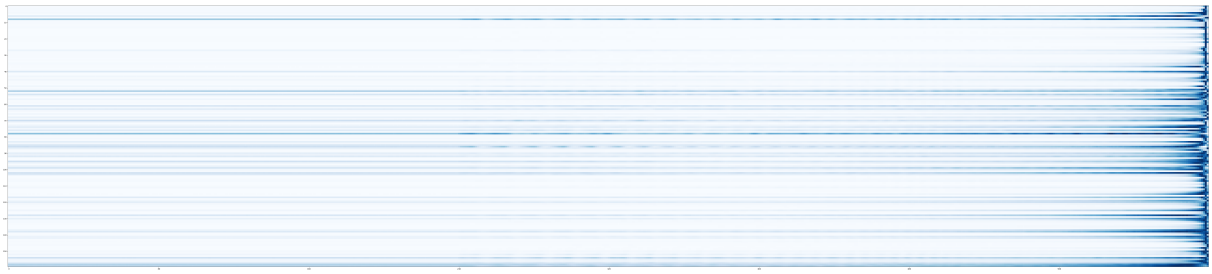
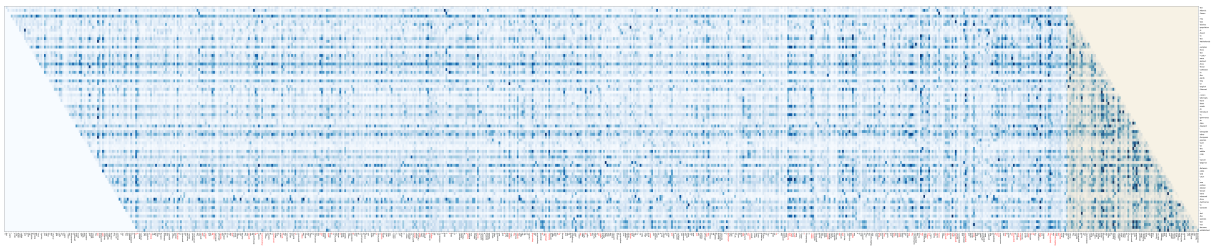


Figure 5: Average attention over the previous 640 tokens, where each row corresponds to a attention head and each column corresponds to a relative location. There are totally 160 attention heads, and every 10 heads come from a single layer. Darker colors indicate higher values.

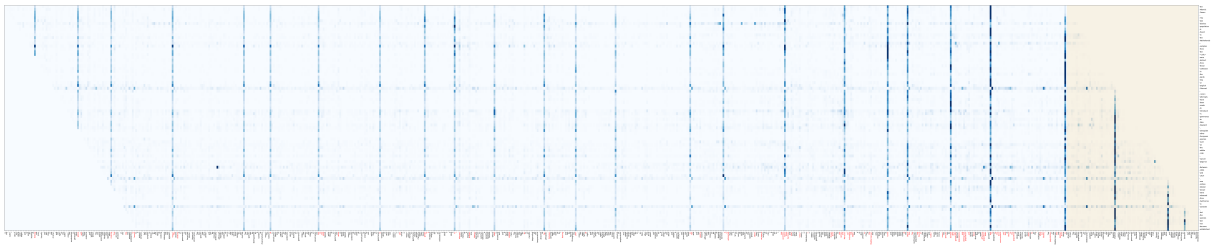
The first visualization aims at revealing the overall trend of where the model is attending. Specifically, for each attention head of each layer, we average the attention distributions of all tokens in the validation set. This is shown in Fig. 5. As we can see, the overall trend is to focus more on the 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:

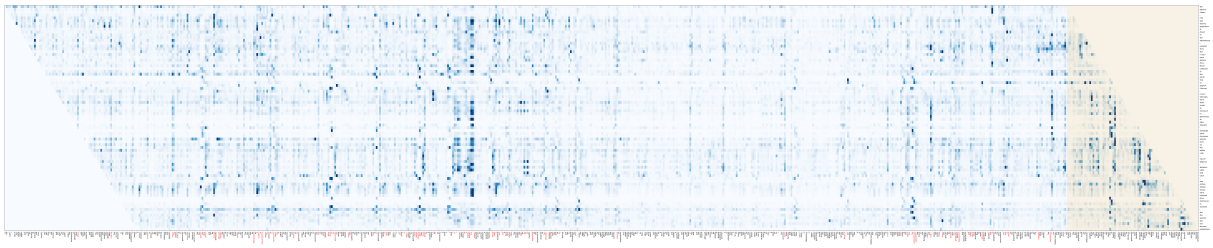
- For the head 8 in the 1st layer, we see an almost uniform attention over the entire memory span. This is quite intuitive, as lower-level layers needs to screen the entire memory span to decide where to focus for higher-level layers



(a) Head 8 from layer 1.



(b) Head 78 from layer 8.



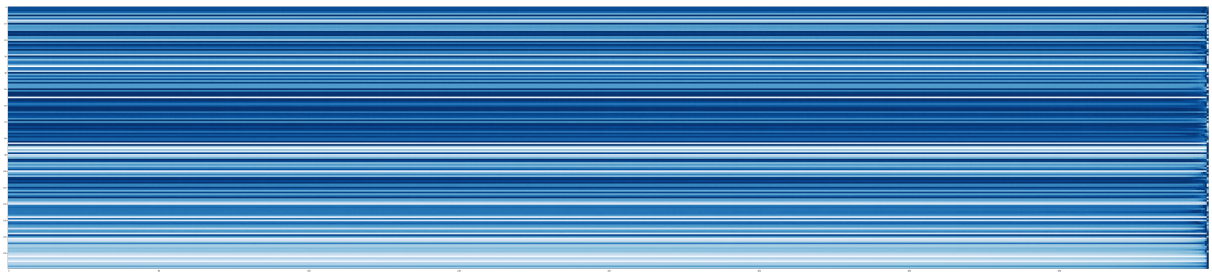
(c) Head 158 from layer 16.

Figure 6: Visualization of the three heads with a wide attention range. Each row corresponds to a target location/token and each column corresponds to a context location/token. Tokens in the memory that have top 20% attention values are highlighted in red.

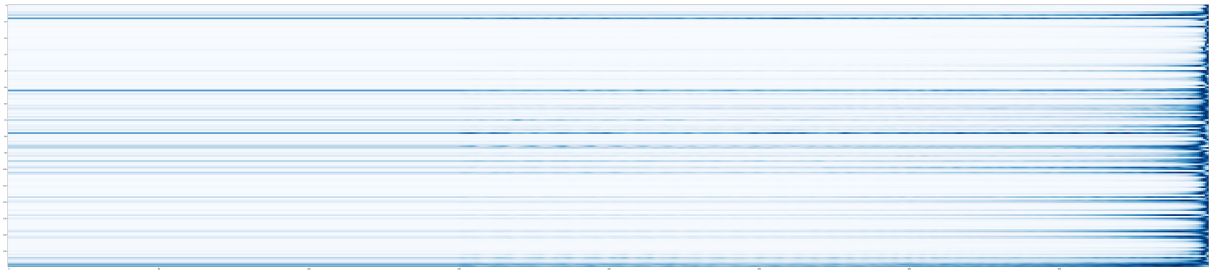
- For the head 78 in the 8th layer (a middle-level layer), we see a very sparse attention pattern scattered in all ranges of the memory. Again, this well fits our intuition that as information accumulates, the network may focus on some particular position with special interests.
- For the head 158 in 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:

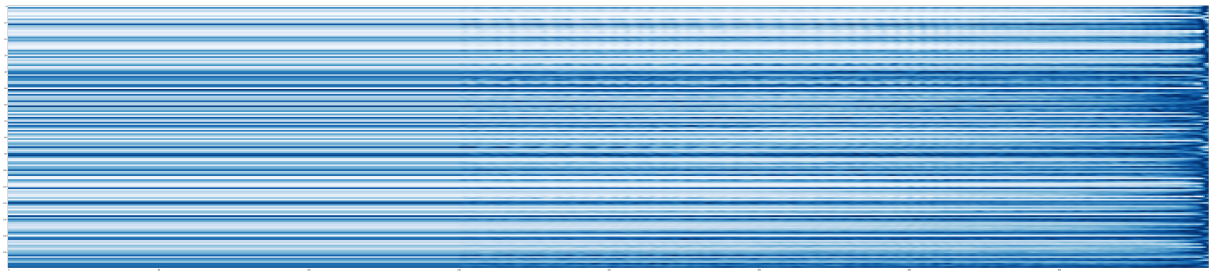
- Since term (a) is fully content-based addressing, when averaging over all target words, the result is essentially uniform over the entire context, except for a few very close words, which are likely to be semantically similar to the target word.
- The overall trend of term (b) highly resembles that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



(a) Term (a).



(b) Term (b).



(c) Term (d).

Figure 7: Visualization of the three terms in computing the attention score. Each row corresponds to a 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 the 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 tile, 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 2.91 ERA in 32 starts, pitching 2041 / 3 innings and recording 212 strikeouts.

= = = = 2011 season: 1st Cy Young Award = = = =

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

XL Generation:

, tying him with Luis Castillo for the third best start total in the league and trailing only Johnny Bench and Eddie Mathews (37). The Dodgers signed Kershaw to a one-year deal on July 14, 2011, with a player option for 2012, in a deal worth just under $ 300,000. The Dodgers entered the 2011 season as the major league leaders, and Kershaw had the best record in the National League in saves (39), games started (78) and walks (120); he also had the lowest win / loss ratio in the league (1.87), and he had the worst on-base percentage (.349). He had the best ERA (2.00), most wins (16), or most shutouts (29) in his career (25), while leading the league in losses per nine innings pitched (4.01). Following the 2011 season, the Dodgers traded Kershaw to the Kansas City Royals on July 29 for pitcher Javier Torres.

= = = Kansas City Royals = = =

= = = = 2012 season = = = =

During spring training, Kershaw played very well. He was selected to spring training as a relief pitcher for the Royals for the 2012 season. After an injury to closer Javier Vázquez, he was activated on April 29 to replace Matt Holliday in the Royals ' starting rotation. In his only start with the Royals, on August 6, 2012, Kershaw struck out five batters in seven innings pitched to help the Royals to their first victory in franchise history. On September 27, 2012, it appeared Kershaw was going to pitch a complete game shutout against the Detroit Tigers, but did not manage to do so since the Tigers won 3 – 1. At the conclusion of the season, Kershaw was named Major League Baseball's Most Valuable Player, was chosen to the All-Star Game at Busch Stadium and was named to the All-Star Game as the starting pitcher at shortstop. The Royals announced on February 4, 2013 that Kershaw would spend 2013 as starting pitcher, though he was expected to miss the season as a result of a shoulder injury.

= = = = 2013 season = = = =

On May 17, 2013, Kershaw sustained another back injury and did not start in August and October 2013. He appeared in 22 starts, all starts, finishing with a strikeout-to-walk ratio of 1.50 and a 2.91 ERA. He also had the third most strikeouts in the league: 10. On May 20, 2013, he

Reference:

, a 3.15 ERA and 593 career strikeouts in 568.2 innings. According to the Elias Sports Bureau, Kershaw was the first 23-year-old pitcher to have that many victories, an ERA that low and an average of more than one strikeout per inning since ERA became an official statistic in 1910. Kershaw was selected to the National League team for the 2011 Major League Baseball All-Star Game, his first All-Star selection. In the month of July, Kershaw was 4 – 1 with a 2.02 ERA and NL-leading 45 strikeouts, earning him the National League Pitcher of the Month Award. On August 23, he struck out Matt Holliday of the St. Louis Cardinals for his 200th strikeout of the season and became the 10th Dodger pitcher to record back-to-back 200 strikeout seasons and the first since Chan-Ho Park did it in the 2001 season. Kershaw finished the 2011 season by leading the NL with 21 wins, 248 strikeouts and a 2.28 ERA, winning the NL pitching Triple Crown, the first Triple Crown winner since Jake Peavy of the 2007 San Diego Padres and the first Dodger since Sandy Koufax won it in the 1966 season. Justin Verlander of the Detroit Tigers won the American League Triple Crown the same season, marking the first major-league season since 1924 to feature Triple Crown-winning pitchers in both leagues. Kershaw's 21 wins were the most by a Dodger pitcher since Orel Hershiser won 23 during the 1988 season. His ERA was the lowest by a Dodger since Hershiser's 2.03 in the 1985 season, his strikeouts were the most by a Dodger since Koufax's 317 in 1966 and his 233 1 / 3 innings pitched were the most since Chan Ho Park pitched 234 in 2001. Since 1965 when Koufax did it, Peavy and Kershaw are only two pitchers in the National League have led the league in wins, strikeouts, ERA, and WHIP (walks plus hits per inning pitched). Kershaw also became just the second <unk> to have a 240-plus strikeouts in a season before the age of 24, joining Vida Blue. After the season, Kershaw was awarded the Warren Spahn Award as the best left-handed pitcher in 2011, the Players Choice Award for Most Outstanding National League pitcher, the Gold Glove Award as the top fielding pitcher in the NL and the Sporting News (TSN) National League Pitcher of the Year. He was additionally selected as the starting pitcher for the TSN NL All-Star Team. On November 17, he was honored with the National League Cy Young Award, making him the youngest Cy Young winner since Dwight Gooden

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 injure in the 2012 season. This shows again Transformer-XL's ability of capturing long-term dependency.

Context:

= = 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 understorey 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. <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 tapirs, 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> 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 $

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.

= The Tale of Mrs. Tittlemouse =

The Tale of Mrs. Tittlemouse is a children's book written and illustrated by H. M. Potter. It was first published in February 1910 by Frederick Warne & Co. and included illustrations by Frederick Warne. Mrs. Tittlemouse was intended to be a children's book, rather than a musical, and the tale describes her attempts to find her own home. Mrs. Tittlemouse is an average kitten and the tale follows a mischievous rabbit named Mr. Tittlemouse who tries to kill her in an attempt to get a ride on a horse. Potter later wrote of the rabbit and her attempts to kill her, " If [she] were a fox, I should like to have been able to show her the way. " Potter's work was inspired by The Tale of Little Black Sambo and her two nieces, Petunia and Petunia. It was the first book in the Potter family to be published in a hardback format, and was also the first book in the Peter Rabbit series. The tale 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.

= = Plot summary = =

The tale opens with the pet rabbit Mrs. Tittlemouse wandering through a forest in a small litter of four. He is shot and badly injured by a deer after she attempts to escape. Mrs. Tittlemouse decides that she wants to find her own home, because she is eager to go on her own. She goes alone to the farm where she makes a little money by selling a few seeds and building a small cabin in the woods. She is approached by a wealthy hunter named Mr. Tittlemouse, who tries to kill her but Mrs. Tittlemouse kills him by stuffing a rope into his nose and killing him. She is rescued by Mr. Tittlemouse's wife Ruth, but Mrs. Tittlemouse then leaves the woodland with the baby. When she is spotted by

Reference:

20 million in annual income to Brazilian farmers in 1996.

= The Heart of Ezra Greer =

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 wayward Mary was romanced and abandoned by Jack <unk>, later bearing his child. Once Ezra becomes broke he finds employment as the valet for Jack <unk>. After Jack's engagement to a cabaret girl, Mary becomes upset and leaves her child at Jack's home. Contrary to Jack's wishes, Ezra keeps the child and Jack ultimately reveals that the child is his own. Ezra convinces Jack to make things right and Ezra convinces the cabaret girl to leave 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.

= = Plot = =

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 awaits for her to return from college, but Mary leaves with her romantic interest, Jack <unk>. On promise of marriage and wealth, Mary is romanced and gives birth to a fatherless child. Without word from his daughter, Ezra resigns from his job and attempts to seek her out and finds a poor motherless child, Marie. With Ezra's money exhausted he seeks employment and finds it as the valet of Jack. One day, Mary seeks an announcement of Jack's engagement to a cabaret girl known as " The Baby Vamp ". Bitter over the prospect of her child's future, she leaves the child at Jack's home during his absence with a note. Jack orders Ezra to take the baby to an orphanage, but Marie begs Ezra to keep him. After continually seeing the child, Jack is overcome with remorse and explains to Ezra and seeks his advice. Not knowing he was making the case for his own daughter, Ezra convinces Jack to seek out Mary and forget the Baby Vamp. The Baby

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.

Original Wikipedia page: https://en.wikipedia.org/wiki/The_Tale_of_Mrs._Tittlemouse.

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 "= title =" 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.

Context:
= Battle of Dürenstein =
The Battle of Dürenstein (also known as the Battle of <unk>, 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. Dürenstein (modern <unk>) is located in the <unk> Valley, on the River Danube, 73 kilometers (45 mi) upstream from Vienna, Austria. The river makes a crescent-shaped curve between <unk> and nearby Krems an der Donau and the battle was fought in the flood plain between the river and the mountains. At Dürenstein 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 <unk> 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 had heavy <unk> to 16 <unk> perhaps the most significant was the death in action of Johann Heinrich von Schmitt, one of Austria's most capable chiefs of 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.
= = Background = =
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

XL Generation:
after the French conquest of Italy. This victory marked the beginning of the Second Coalition. Napoleon's rapid advance caused Austria – Russia, Britain and Russia to make peace negotiations. The Russian army under Prince Mikhail Mikhailovich Mikhailovich Chaffee, commander of the Imperial Russian Army in Switzerland, was able to drive off the Austrians in the Battle of Stockach (1805) and to take Pressburg (modern Ö3 Austria) from the Austrians. At an early stage the Austro-Hungarian fleet had been damaged in a storm off Cape Matapan; this allowed the Allies to launch a full-scale invasion of Italy in February 1805. The Battle of Austerlitz was the decisive French victory against Napoleon and the largest naval battle in history, involving more modern-day European navies. The French military situation worsened as Napoleon faced several threats related to his newly formed Army of Europe, particularly Russia itself, which was now on the verge of collapse. The Russian general and friend of Napoleon, Alexander, had been dissatisfied with the conduct of the Austrians; he was still in conflict with Emperor Napoleon, the French Republic's king, who had declared war on Austria during the summer of 1804. With the war over, Napoleon decided to launch a second invasion of Italy in October 1805.
= = Prelude = =
In July 1805, the French 1st Army entered southern Italy. The army, under the command of Marshal Marmont, were reinforced by a few battalions of infantry under Claude General Auguste de Marmont at the town of Philippsburg and another battalion at Belluno. On 17 September 1805, the army marched from Belluno towards Krems. By 29 September, they had reached Belluno and conducted its advance against a small Austrian force. By 31 September, the whole force had been reinforced by a brigade from the Army of Tyrol under the command of Pierre Augereau. The Austrians were now under the command of Marshal Jean Victor Marie Moreau, a member of the Directory. Moreau had taken command of the Austrian invasion force in the spring of 1805. His command included the VI Corps commanded by Jean Baptiste Drouet de Ney and the VI Corps commanded by Generals Jean Victor Marie Moreau and Joseph Souham. Ney's corps consisted of the III. Corps and VI. Corps, which consisted of the III Corps and VI. Corps, located in the Austrian Netherlands, was commanded by Friedrich Joseph, Count Baillet de Latour. Moreau's army consisted of six divisions and several associated brigades.
= = Aftermath = =
= = = First Coalition forces = = =
On 9 October 1805 the French Army of the Danube was attacked by an Austrian army under Archduke Charles at the Battle of Austerlitz. Although Charles and Charles had not had much time to regroup, on 10 October, he launched his attack on the Polish forces under Friedrich Joseph, Count of Lauenburg. After three days, Charles' army captured Lauenburg. The French forces pursued the Austrians to the Silesian border, where they encountered strong Austrian resistance. These conflicts forced the Austrians to retreat into Tyrol and Austria agreed to a truce. The Austrian army, commanded by Wenzel Anton Karl, Count of Merveldt, was reduced to around 10,000 men. It was initially planned that Archduke Charles would launch a counter-attack against the French army on the same day, as Napoleon had hoped, but this was not carried out. On 25 October, Merveldt left Styria for Tyrol. On the same day, Austria launched its new offensive against the French at Ulm. Charles withdrew his army from the region for a third time at the Battle of Elchingen, under the overall command of the Austrian generals, Ferdinand and Friedrich Wilhelm of Jülich-Cleves-Berg. To prevent Archduke Charles from escaping from the battlefield, the commander of the Habsburg army, Archduke Charles, planned to occupy the fortress Linz; instead, he decided to force Franz von Hipper to surrender the city. However, as Charles moved to the south, Moreau arrived on the scene with additional soldiers – including the entire Imperial Guard – and defeated the Austrians at the Battle of Hohenlinden on 28 October. The loss of Linz resulted in Austria's complete defeat at Hohenlinden. In the meantime, the French Army of Observation and Preparedness was reorganized into the Army of the Danube under Feldzeugmeister (Colonel-General) Friedrich Freiherr von Hotze. The army was composed of the I, IV, VI, VI, VII, VIII and IX Corps. With reinforcements from Italy and France, it formed new battalions, companies, and squadrons in the Austrian army. On 17 November 1804, at the Battle of Jena-Auerstadt the Army of Silesia and the Army of Silesia joined forces, but by the time that the

Reference:
after the disastrous invasion of Russia in 1812. Napoleon's empire ultimately suffered complete military defeat in the 1813 – 14 campaigns, resulting in the restoration of the Bourbon monarchy in France. Although Napoleon made a spectacular return in 1815, known as the Hundred Days, his defeat at the Battle of Waterloo, the pursuit of his army and himself, his abdication and banishment to the Island of Saint Helena concluded the Napoleonic Wars.
= = Danube campaign = =
From 1803-06 the Third Coalition fought the First French Empire and its client states (see table at right). Although several naval battles determined control of the seas, the outcome of the war was decided on the continent, predominantly in two major land operations in the Danube valley: the Ulm campaign in the upper Danube and the Vienna campaign, in the middle Danube valley. Political conflicts in Vienna delayed Austria's entry into the Third Coalition until 1805. After hostilities of the War of the Second Coalition ended in 1801, Archduke <unk> emperor's <unk> advantage of the subsequent years of peace to develop a military restructuring plan. He carefully put this plan into effect beginning in 1803 – 04, but implementation was incomplete in 1805 when Karl Mack, Lieutenant Field Marshal and Quartermaster-General of the Army, implemented his own restructuring. Mack bypassed Charles ' methodical approach. Occurring in the field, Mack's plan also undermined the overall command and organizational structure. Regardless, Mack sent an enthusiastic report to Vienna on the military's readiness. Furthermore, after misreading Napoleon's maneuvers in Württemberg, Mack also reported to Vienna on the weakness of French dispositions. His reports convinced the war party advising the emperor, Francis II, to enter the conflict against France, despite Charles ' own advice to the contrary. Responding to the report and rampant anti-French fever in Vienna, Francis dismissed Charles from his post as generalissimo and appointed his <unk> brother-in-law, Archduke Ferdinand, as commander. The inexperienced Ferdinand was a poor choice of replacement for the capable Charles, having neither maturity nor aptitude for the assignment. Although Ferdinand retained nominal command, day-to-day decisions were placed in the hands of Mack, equally ill-suited for such an important assignment. When Mack was wounded early in the campaign, he was unable to take full charge of the army. Consequently, command further devolved to Lieutenant Field Marshal Karl Philipp, Prince of Schwarzenberg, an able cavalry officer but inexperienced in the command of such a large army.
= = = Road to Ulm = = =
The campaign in the upper Danube valley began in October, with several clashes in Swabia. Near the Bavarian town of Wertingen, 40 kilometers (25 mi) northwest of Augsburg, on 8 October the 1st Regiment of dragoons, part of Murat's Reserve Cavalry Corps, and grenadiers of Lannes ' V Corps surprised an Austrian force half its size. The Austrians were arrayed in a line and unable to form their defensive squares quickly enough to protect themselves from the 4,000 dragoons and 8,000 grenadiers. Nearly 3,000 Austrians were captured and over 400 were killed or wounded. A day later, at another small town, <unk> south of the Danube <unk> French 59th Regiment of the Line stormed a bridge over the Danube and, humiliatingly, chased two large Austrian columns toward Ulm. The campaign was not entirely bad news for Vienna. At Haslach, Johann von Klenau arranged his 25,000 infantry and cavalry in a prime defensive position and, on 11 October, the overly confident General of Division Pierre Dupont de l'Étang attacked Klenau's force with fewer than 8,000 men. The French lost 1,500 men killed and wounded. Aside from taking the Imperial Eagles and <unk> of the 15th and 17th Dragoons, Klenau's force also captured 900 men, 11 guns and 18 ammunition wagons. Klenau's victory was a singular success. On 14 October Mack sent two columns out of Ulm in preparation for a breakout to the north: one under Johann Sigismund Riesch headed toward Elchingen to secure the bridge there, and the other under Franz von Werneck went north with most of the heavy artillery. Recognizing the opportunity, Marshal Michel Ney hurried the rest of his VI Corps forward to re-establish contact with Dupont, who was still north of the Danube. In a two-pronged attack Ney sent one division to the south of Elchingen on the right bank of the Danube. This division began the assault at Elchingen. At the same time another division crossed the river to the east and moved west against Riesch's position. After clearing Austrian pickets from a bridge, the French attacked and captured a strategically located abbey at

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	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.