Appendices

Appendix A Further Analysis.

The *context length* of an autoregressive language model refers to the number of previous tokens that the model can make use of when predicting the next token. A vanilla transformer operating on segments of length N has a context length of 0 for the first token, and $N-1$ for the last token, and thus has an average context length of $N/2$. The prediction quality for a vanilla transformer is not uniform across the segment; it makes poor predictions for tokens near the beginning of the segment due to lack of context, and better predictions near the end.

With sliding window attention, each layer attends to the previous layer within a window of W tokens from the current position. The context length for a single layer is thus W, no matter where in the segment the token occurs. The simple case where $W = N$ corresponds to Transformer-XL. A Transformer-XL model achieves a large improvement in average perplexity over a vanilla model, simply because it can make much better predictions for the tokens at the beginning of each segment.

In a model with multiple layers, the *theoretical receptive field* (TRF) is defined as the maximum distance that information could potentially propagate through the model. This definition is similar to "context length", but the TRF is "theoretical", because the model may have a hard time actually learning to use that much context in practice. For example, the TRF of an LSTM is infinite, but in practice an LSTM has difficulty transmitting information over more than a few hundred tokens. The *effective context length* of an LSTM is much less than the TRF might suggest.

For a sliding-window model (or Transformer-XL), the TRF is $W \cdot L$, where L is the number of layers. However, the model can still only attend directly to the previous W tokens. Although the TRF is much higher, making use of the additional context requires multiple "hops" of attention, which is more difficult for the model to learn. This problem is especially acute in Transformer-XL ($N = W$), because the very first "hop" of attention will be to keys and values in the cache, which is not differentiable. Our sliding window model ($N >> W$) has an identical TRF to Transformer-XL, but it can differentiate across multiple blocks, which gives it a higher effective context length in practice, and thus better perplexity.

The TRF of the block-recurrent transformer is infinite. We also show that the effective context length also seems to be quite large in practice, since we observe cases in which the model is able to accurately predict information across distances of more than 60k tokens.

A.1 Computational Complexity

The computational complexity of attention in a recurrent layer is $\mathcal{O}((W^2 + S^2 + 2SW) \cdot N/W)$, where N is the segment length, W is the window length, and S is the number of states. N/W is the number of blocks, and each block does self attention W^2, state self-attention S^2, and attention between tokens/states and states/tokens $2SW$.

The complexity of sliding-window attention is $\mathcal{O}(W^2 \cdot N/W) = \mathcal{O}(W \cdot N)$.

A.2 Comparison between context, recurrence, and memory

The arXiv data set contains latex code, with lots of complicated syntax that can benefit from long-range attention (e.g. theorems, jargon, citations). As a result, adding memory, recurrence, or just increasing the window size of the Transformer-XL baseline yields a larger benefit on arXiv than it does for PG19.

There also seems to be a qualitative difference in how the models are using attention on these datasets. We hypothesize that dealing with complicated syntax requires direct (single-"hop") attention, which is why the `XL:2048` model does well on arXiv. In contrast, natural language novels don't have complicated syntax, but may benefit from a better understanding of subtle relationships between words (e.g. which character the word "her" refers to). These subtle interactions require multi-layer, multi-"hop" attention, which is easier to train in the more differentiable `Slide:13L` model. On PG19, `Slide:13L` actually does better than `XL:2048`, despite having a much shorter window size, while on arXiv, the situation is reversed.

There may be a similar relationship between recurrence and kNN memory. The Memorizing Transformer does direct, single-"hop" attention, using kNN lookup. Recurrence, in contrast, summarizes and compresses text into a set of recurrent states.

Both the Memorizing Transformer and the Block-Recurrent Transformer achieve very similar perplexity, and based on qualitative studies, they seem to use the additional memory or states primarily for long-range name lookups. However, recurrence may be better at capturing more subtle long-range information, like writing style, while memory is better at precision lookups of facts, like citations. The fact that the Memorizing Transformer does better than the Block-Recurrent transformer on arXiv, but not PG19, would seem to support this hypothesis, but further experiments on downstream tasks are necessary to confirm this hypothesis.

A.3 Comparison against Longformer

The idea of sliding window attention was popularized by the Longformer [33], but the full Longformer is a much more complicated model than the `Slide` model that we present here.

The full Longformer uses several different attention patterns, and sliding-window attention is only one of them. Longformer also uses dilated attention and sparse global attention, both of which are implemented with custom CUDA kernels. Moreover, LongFormer uses different window sizes in each layer, and it uses a multi-phase training regimen of pre-training and fine-tuning, following a curriculum that gradually increases window size and sequence length. We do none of these things.

Appendix B Gate Initialization and Training Stability

We observed that training stability is quite sensitive to how the gates are initialized. Recurrence has a failure mode where the model learns to completely ignore the recurrent state, in which case its performance reverts to that of the non-recurrent transformer. Moreover, this situation appears to be a local optimum; once the model has reached this point, it does not recover.

Our hypothesis is that learning the recurrent transition function is a much more difficult task than learning to attend directly to the input tokens. As a result, the vertical direction trains much faster than the horizontal direction, especially early in training. This may lead to a situation in which the recurrent states are much less informative than the input tokens, and the model learns to ignore them.

To avoid this failure mode, gate initialization requires special care. Moreover, proper gate initialization depends on the optimizer. We use the Adafactor optimizer [51], which normalizes gradients with respect to their variance, and then multiplies them by the native scale of the underlying parameter. Thus, if a bias term is initialized to 0, its native scale will be 0, the gradient updates will be very small, and the bias will tend to remain small over the course of training. If a bias term is initialized to 1 (which tells the forget gate to "remember", and is standard practice in LSTMs) then the initial updates will be large, and the model will learn to ignore the recurrent states before they have the chance to learn anything useful.

We compromise by initializing the bias terms of the gates to small but non-zero values, using a normal distribution with mean 0 and a standard deviation of 0.1. The weight matrices of the gates are also initialized to small values, using a truncated normal distribution with a standard deviation of $\sqrt{(\frac{0.1}{f_{\text{in}}})}$ where f_{in} is the dimension of $\boldsymbol{h}_t$.

We add a constant of -1 and +1 to the input and forget gates (see Eq. 4) to initially bias the gate to "remember" without affecting the size of the updates that Adafactor will apply. Using this initialization trick, the recurrent cell reliably learns to make use of the recurrent state.

Appendix C Training Details

For PG19, we do both token-level modeling, and character-level modeling. In our initial experiments, we use a pre-trained 32k sentencepiece vocabulary from T5 [39] for the token-level modeling. We use the Adafactor optimizer [51], a learning rate schedule with inverse square root decay, 1000 warmup steps, and a dropout rate of 0.05. The learning rate is 1.0; when combined with warmup and the decay schedule, this yields an applied learning rate of 0.03, decaying to 0.0014. This learning rate

and schedule were borrowed from other language models; we did not attempt to do a hyperparameter sweep to identify the optimum learning rate and schedule. We train for 500k steps with 32 replicas on Google V4 TPUs; training takes approximately 48 hours. Reported results are for the "test" split.

For the later scaling experiment on PG19, we switched the learning rate schedule to cosine decay, as recommended in [43], with a maximum rate of 0.01, and a minimum of 0.001. We did a brief experiment with a learning rate of 0.02 and 0.005, before settling on 0.01. The change in learning rate schedule resulted in a significant improvement. We also switched from the 32k T5 vocabulary to a 32k custom sentencepiece vocabulary trained on PG19. Our custom vocabulary has higher bits-per-token, but fewer tokens, and thus has slightly better word-level perplexity.

For arXiv, we use a pre-trained 32k vocabulary from LaMDA [52]. Due to the large number of mathematical symbols in LaTeX, many tokens are only one character, so the bits-per-token numbers are lower than for PG19. We dropped the learning rate to 0.5 after observing some instabilities when training on longer (4096) segment lengths. Reported results are for the "test" split.

The GitHub dataset is much larger than PG19, and has very high variance, due to the fact that it contains code written in many different programming languages and coding styles. Consequently, there was a lot of noise in the results, which made it difficult to accurately compare models. We reduced the noise by using a batch size that is 4x larger than for PG19. As with the PG19 scaling study, we use a cosine decay learning rate schedule, but the maximum LR is 0.005; this is half the LR used for PG19. Because of the increased batch size, these models ran for only 250k steps. The GitHub experiment uses a pre-trained 32k vocabulary from LaMDA. Reported results are for the "validation" split.

Due to the large number of experiments, we did not have the computational resources to run all experiments multiple times, and consequently do not provide error bars for all experiments. However, for the headline numbers on PG19-tokens, we ran each experiment 3 times, with both different random seeds and with dataset shuffling. Actual measured error bars on PG19 were very low, between 0.002 and 0.007. The numbers in Table 1 are rounded to the nearest 0.01, which means that the error bars must be rounded up to match the precision of the reported results. E.g. for `Rec:fixed:skip` on PG19-tokens, an average of 3 runs has a mean of 3.525 and a standard deviation of 0.0047; we round this *up* to 3.53 ± 0.01. Note that it is not possible to obtain truly accurate error bars from such a small number of runs; by rounding the error up, we provide a conservative estimate of the actual error.

C.1 Dataset licensing and other issues.

PG19 consists of works in the public domain, and consequently it is a public dataset that is freely available to other researchers. Due to the age of the texts, some of the books do contain potentially offensive material.

The arXiv dataset consists of documents for which the author has given express permission for their work to be distributed by `arxiv.org`. However, because the author still retains copyright, these articles cannot necessarily be redistributed in the form of a public dataset, nor will we publish a model that has been pre-trained on this data. We obtained access to this dataset via private channels.

The Github dataset consists of code with open-source licenses, which permit the code to be downloaded, compiled, or modified. Similar to arXiv, however, because the authors retain copyright, this code cannot necessarily be redistributed as a public dataset, nor will we publish a model that has been pre-trained on this data. We obtained access to this dataset via private channels.

C.2 Selection of data sets

Having a standardized data set is import for the purpose of comparing published results, and historically, papers on long-context language modeling have used enwik8 or wiki-103 as benchmarks [34][33]. However, these datasets are not particularly good benchmarks for our purposes.

The purpose of our experiments is to see whether block recurrence can transmit information over very long distances: we show retrieval over 60k+ tokens. We chose PG19 specifically because we believe it to be a good dataset for these sorts of experiments. It consists only of long, book-length works, it is much larger than enwik8 or wiki-103, it is publicly available, and has been cited in other published work. Arxiv and github are (sadly) not public, but they similarly have long documents in the 50k+ token range.

Enwik8 is not a corpus of long articles. In fact, it doesn't even split the text into separate articles at all; it's just a single text dump that concatenates a bunch of short unrelated articles together. Attempting to split it on article boundaries yields a data set in which the majority of "articles" are merely stubs, with HTML boilerplate and no actual text. Enwik8 is a fine benchmark for data compression, which was the purpose for which it was originally intended, but it is less than ideal for long-range language modeling.

Wiki-103 is significantly better, because it does break the text into articles, and it eliminates the boilerplate, but the average length is still only 3.6k tokens per article, which is less than the segment length used in our experiments, and much shorter than the 50k-100k tokens of PG19.

C.3 Comparisons with previously published results.

It is well-known that transformer perplexity scales with the number of parameters. However, the choice of vocabulary, learning rate schedule, batch size, number of training steps, and optimizer also make a large difference to the final headline perplexity numbers. The Chinchilla scaling study [43] demonstrates that a change to the learning rate schedule can have a large effect; we also observed improvements from a change to vocabulary as well. Not all vocabularies are created equal, even for vocabularies which have the same size, and are trained primarily on English-language text. Published perplexity numbers between different models cannot be meaningfully compared unless all other variables are strictly controlled.

Appendix D Window Size and Number of Recurrent States

Ablation results for the window size is is shown in Table 3 (a). Decreasing window size leads to worse perplexity in both the recurrent model and Transformer-XL, but the penalty is smaller for the recurrent model.

Ablation results for the number of recurrent states is shown in Table 3 (b). Increasing the number of recurrent states makes a small but measurable improvement up to 1024 states, but is worse at 2048 states. The window size was 512 for this experiment.

Table 3: Changing the window size (a) and number of recurrent states (b) on PG19.

Window size	`Rec:fixed:skip`	`Slide:12L`
128	3.58	3.69
256	3.54	3.64
512	3.53	3.60

Number of states	`Rec:fixed:skip`
128	3.54
256	3.535
512	3.53
1024	3.51
2048	3.55

Appendix E Block-Feedback

In the block-feedback variation of our model, the entire stack of 12 layers is applied to the first block of tokens. The recurrent state for that block is then extracted from the recurrent layer, and the state is broadcast to all other layers when processing the next block. Because the recurrent layer is placed high in the stack, this means that the lower layers of the transformer can cross-attend to a higher layer, which is computationally more powerful, much like the feedback transformer [24]. Results are shown in Table 1.

Recurrence with feedback is significantly more expensive than the non-feedback version, because all 12 layers now have a cross-attention module, instead of just the recurrent layer. In our experiments, feedback increased the step time by approximately 35-40%, and the additional queries also increase the number of parameters.

Adding feedback improves perplexity in most cases, but the improvement seems to depend on the data set. The effect of feedback also depends on the gate configuration. In particular, block feedback dramatically improves the performance of the LSTM gate. This could be because the recurrent states, and thus the gate, get a gradient from all all layers of the transformer, instead of just one.

Appendix F Scaling Plot Details

Table 4: Bits per token on PG19 at various model scales. The data in this table was used for the scaling plot in Figure 3.

	40.6M	81.2M	162M	325M	650M	1.3B
`Rec:fixed:skip`	3.96	3.79	3.59	3.44	3.29	3.22
`XL:512:13-layer`	4.03	3.86	3.67	3.51	3.39	3.31

Performance on large text datasets, such as PG-19, is highly correlated with the number of trainable parameters; larger models tend to perform better. However, training large models can be expensive, and not all researchers have access to the necessary amount of compute to beat our new state of the art, or even to reproduce our results.

Table 4 provides the numeric results from our scaling study, which covers scales from small 40M parameter models that can be easily trained on a single machine, to our largest 1.3B parameter model. Configuration files for all scales are provided in the open source release. Our scaling strategy is to increase the dimensions of the various parts of the transformer: embedding size, MLP hidden layer size, number of heads, head size, and number of layers. Each factor of 2 increase in the number of parameters scales either one or two of these dimensions.

Appendix G Qualitative Analysis Results

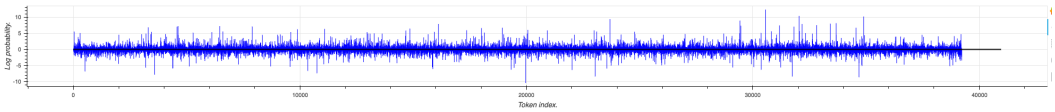

Figure 4: Difference in per-token cross-entropy loss.

The following are excerpts from our qualitative study. We selected five books at random from PG19 test set, and ran two different models on each book. The first model is a 13-layer Transformer-XL, and the second is the `Rec:fixed:skip` configuration of the Block-Recurrent Transformer. For each token, we compute the difference between the cross-entropy loss (i.e., the negative log likelihood (NLL)) output by both models, and and then sort the results.

Figure 4 shows an example of the per-token difference in NLL between the two models on the first book; the x-axis is the index of the token. On average, the recurrent model does slightly better than Transformer-XL, but it does not necessarily make a better prediction for any individual token.

The following excerpts show the top 4 tokens where the Block-Recurrent Transformer made a better prediction than Transformer-XL; these tokens correspond to spikes in Figure 4. We show the token number, the NLL returned by the recurrent model, the NLL returned by Transformer-XL, and an excerpt of text, with the token itself marked with `|token|`. Almost all of the top tokens are proper names of characters and places. In all cases except one, the mispredicted name does not appear within the attention window of the previous 512 tokens. These names are thus invisible to Transformer-XL, but visible to the recurrent model.

Note that these are not cherry picked examples; the five books are chosen at random. Moreover, the same pattern still holds if the search is expanded to the top 40 tokens for each book. In fact, even the names are often the same; Transformer-XL often seems to mispredict the same names over and over again; these are likely the names of main characters.

Sorting the other way, to show the top tokens where Transformer-XL does better than the recurrent model, does not show the same pattern. There are still plenty of proper names, but it is usually cases where both models fail to predict the name. Moreover, the names are mixed with more common words as well.

Memoirs of Marie Antoinette Queen Of France Vol. 7.

```
(30555, 0.3696011, 12.688916)
```

```
physician’s house to make inquiries as to  the cause of so long an absence.
G|omin| and Larne had not yet ventured to  follow this advice, when next

(32043, 0.20177796, 10.513703)
with the autopsy arrived at the  outer gate of the Temple.  These were
Dum|ang|in, head physician of the  Hospice de l’Unite; Pelletan,

(34896, 1.3752508, 11.534926)
who had acted as courier to Louis  XVI. during the flight to Varennes,
and Tur|gi|, who had waited on the  Princesses in the Temple.  It was

(34896, 1.3752508, 11.534926)
.    In all the evidence there appeared but two serious facts, attested
by  Latour-|du|-Pin and Valaze, who deposed to them because they could
```

Notes: “Gomin” appears multiple times in the top 40 results.

An Unsentimental Journey through Cornwall

```
(983, 0.80441666, 11.626528)
OUGH CORNWALL          [Illustration: FALMOUTH, FROM FLUSHING.]
|DAY| THE FIRST      I believe in holidays. Not in a frantic rushing

(23606, 0.655162, 11.265186)
there was not the slightest use  in getting up, I turned round and took
another sleep.          |DAY| THE FIFTH      "Hope for the best, and be

(11297, 0.405902, 10.426135)
ball.    "Ma’am, if you go slow and steady, with me before and Cur|gen|ven
behind,  you’ll _not_ fall."    Nor did I. I record it

(38021, 1.3457009, 11.167879)
our journey; going over the same ground which  we had traversed already,
and finding Praden|ack| Down as bleak and  beautiful as ever. Our first
```

Notes: The word “DAY” appears earlier in the table of contents, but not within the previous 512 tokens. “Curgenven” appears multiple times in the top 40.

The Good Hope **by Herman Heijermans Jr.**

```
(3975, 0.7425603, 15.8969145)
to us."    It matters nothing that this gospel of Life has often been
preached. He|ij|ermans has caught the spirit of it as well as the  letter.

(7385, 0.26241615, 15.000762)
must scratch the stones.    CLEM. Tomorrow afternoon, then.    COB. T|ja|!
I’ll be here, then. Good day, Miss. [To Barend.] Good

(42638, 0.73628587, 15.407666)
hundred guilders. Bejour. [Rings off;  at the last words Kne|irt|je has
entered.]    KNEIRTJE. [Absently.] I----[

(25798, 0.12310324, 14.402218)
They exeunt, dragging Barend.]    KNEIR. Oh, oh----    TRU|US|. [With
anxious curiosity, at side door.] What was the matter,  Kneir?
```

Notes: The word “Tja” is not a proper name, but is a Dutch word that is likely unique to this particular book. It appears multiple times in the top 40 results. The name “Truus” appears multiple times in the top 40.

***Travels in Morocco Volume 2* by James Richardson**

Notes: The second example in this list is not a good one, because both models mispredict the name. We thus include a fifth example as well. The word "Toser" appears frequently in the top 40, and is also the only case where the proper name appears within the attention window of 512 tokens.

```
(61049, 2.0328524, 18.696453)
, some of them as black as [Redacted]s. Many people in  T|oser| have sore
eyes, and several with the loss of one eye, or nearly  so; opthal

(63453, 9.254061, 24.413633)
Tunis;" but the restrictive system established by the Turks  during late
years at Ghad|umes|, has greatly damaged the trade between the  Jereed and

(66149, 0.8768588, 14.105822)
enterprizing fellow,  worthy of imitation. He calculated the distance from
Ghabs to T|oser| at  200 miles. There are a number of towns in the

(64161, 1.0309317, 14.152167)
and ourselves went to Wedyen, a town and  date-wood about eight miles
from T|oser|, to the left. The date-grove is  extensive, and there are

(27282, 0.04982663, 12.551973)
, is a very ancient city, situate upon the right bank of the  river
Boura|gra|g, and near its mouth. This place was captured in 1263, by
```

India's Love Lyrics by Laurence Hope

```
(15694, 0.07477246, 11.22447)
rieving            A wasted chance.  Fate knows no tears.       Verses:
Fa|iz|Ulla       Just in the hush before dawn     A little wistful wind is

(3135, 0.0126608405, 9.759871)
afloat     In the little noontide of love's delights           Between
two Nights.            Val|go|vind's Boat Song       Waters glisten and
sunbeams quiver,

(23183, 0.1419289, 9.481476)
sleep, the touch of your lips on my mouth.            His Rubies: Told by
Val|go|vind       Along the hot and endless road,       Calm and erect,
with haggard eyes,'

(26886, 0.13428347, 9.46193)
off sheath,       And find there is no Rival like the Past.     Verse
by T|aj|Mahomed       When first I loved, I gave my very soul    Utterly
```

G.1 Clearing the recurrent state

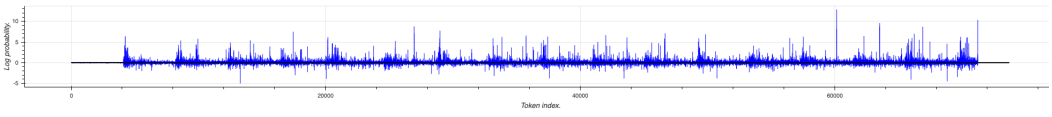

Figure 5: Difference in per-token cross-entropy loss with state clearing.

Our second qualitative study is structured similarly to the first, except that instead of comparing two different models, we compare two different runs of the same model: the `Rec:fixed:skip` configuration. The first run processes the book normally, while the second run clears the recurrent states at the beginning of each 4096-token segment. In the second run, the model can use recurrence within a segment to look beyond the local attention window of 512 tokens, but it cannot use recurrence to carry information from one segment to the next.

This experiment is somewhat cleaner than the first, because both runs are done with the same pre-trained model, which has the same parameters. Figure 5 shows the difference in per-token cross-entropy loss for the first book. There is no difference between the two runs for the first segment, and the biggest differences in subsequent segments are often clustered near the start of each new segment.

The overall pattern is very similar to the first qualitative experiment: most of the tokens involve proper names. We verified that in most cases, the mispredicted name not only does not occur within the 512-token attention window, but does not occur within the 4096-token segment. In addition to proper names, chapter titles and illustration captions occur frequently within the top 40 results; the recurrent model seems to be remembering these from a previous occurrence in the table of contents.

Perhaps most interestingly, in two of the books, one of the highest ranked mispredictions was the title and author of the book itself. The Gutenberg project inserts boilerplate at both the beginning and end of each book; the title and author are listed multiple times at the beginning, and once at the end. This experiment thus shows that the model is able to "remember" this information in the recurrent state, across a distance of 60,000 tokens or more.

Baby Mine, by Margaret Mayo

Note that the 2nd misprediction **is the title of the book itself**, at token number 71,244.

```
(60153, 0.00438364, 12.798895)
nerves, but you  needn't worry, I've got everything fixed.
Donneg|hey| sent a special  officer over with me. He's outside

(71244, 2.3727102, 12.660636)
sunlight and shadows of his  ample, well kept grounds.      End of the
Project Gutenberg EBook of| Baby| Mine, by Margaret Mayo    ***

(63536, 4.132567, 13.688847)
Alfred, with the air of a  connoisseur.    "She sure is," admitted
Don|neg|hey, more and more disgruntled as he felt  his reputation for

(26947, 3.7521973, 12.516711)
with a sigh of thanksgiving he hurried upstairs to his unanswered mail.
CHAPTER XIII    When Alfred| Hardy| found himself on the train bound for
```

The 'Patriotes' of '37 by Alfred D. Decelles

```
(38759, 0.85374373, 11.140007)
24, 125,  126.    Cote, Dr Cyri|le|, 89, 108, 118, 120;

(40181, 0.35475963, 10.291676)
17-26, 129-30.    Nelson, Dr Wolf|red|, a follower of Papineau, 37, 60

(28756, 0.0010796335, 9.256147)
on a well-reasoned plan of  action.  Most of the leaders--Wolf|red| Nelson,
Thomas Storrow Brown,  Robert Bouchette, and Amury Girod--were

(28772, 1.7782648, 10.611834)
leaders--Wolfred Nelson, Thomas Storrow Brown,  Robert Bouchette, and
Am|ury| Girod--were strangers to the men under  their command; and none of
```

Another Brownie Book, by Palmer Cox

In this case, **the top 4 results include the author of the book**. This book has a lot of illustrations which are listed in the table of contents, and make up many of the other results in the top 40.

```
(16442, 0.087840006, 8.706101)
```

```
[Illustration]    [Illustration]          [Illustration]    THE BROWNIES
AND THE TUG|BO|AT.      [Illustration]    While Brownies strayed along a

(24225, 0.04784866, 8.266858)
[Illustration]            End of the Project Gutenberg EBook of Another
Brownie Book, by Palmer| Cox|   ***

(24224, 5.651471, 11.840027)
Illustration]    [Illustration]            End of the Project Gutenberg
EBook of Another Brownie Book, by| Palmer| Cox    ***

(4460, 1.0682098, 6.6397715)
To secret haunts without delay.    [Illustration]          [Illustration]
THE BROWNIES AT| AR|CHERY.      [Illustration]    [Illustration]
```

***The Life Of Thomas Paine Vol. II.* (of II) by Moncure Daniel**

By the end of the book, the recurrent model is quite sure that "Paine" is a main character, but not if its memory keeps getting erased.

```
(170457, 0.98941976, 12.209277)
they could. The scandal branched into  variants. Twenty-five years later
pious Grant| Thor|burn promulgated that  Paine had run off from Paris

(120592, 0.6974209, 10.396527)
my friends and accept the same to yourself."    As the Commissioners did
not leave when they expected,| Paine| added  several other letters to

(169152, 0.18432029, 9.516743)
whose composition the farrier no doubt supposed  he had paid the editor
with stories borrowed from "Old|ys|," or not  actionable. Cheetham

(139870, 0.53406477, 9.646917)
nor was any letter received from  him. This was probably the most important
allusion in a letter of| Paine|,  dated New York, March 1, 1804, to "C
```

Appendix H Token level cross-entropy

In addition to qualitative studies comparing a single document, we also compared the average bits-per-token over all documents in the PG19 test set. It has been observed that in vanilla transformer architectures, this token-wise cross-entropy often diverges at long segment lengths [53].

In Figure 6 we plot the cumulative cross-entropy, which is the average bits-per-token (i.e. $\log_2$ perplexity) averaged up to the given length, and we compare the Block Recurrent Transformer against the Transformer-XL baseline. Performance of the two architectures is comparable for the first few thousand tokens, but the recurrent architecture clearly outperforms Transformer-XL at longer document lengths.

Appendix I Vocabulary Ablation Experiments

The authors of the PG-19 dataset propose *word-level perplexity* (WLP) as a measure of model performance [15]. However, it is only possible to compute WLP if there is an accurate count of the number of words. A published word count is provided for the PG19 data set [15], but a word count is not available for the other data sets in this paper, such as arXiv and github. There is not even a clear definition of what constitutes a "word" in source code or LaTeX documents, which have a lot of symbols. Consequently, we do not provide WLP for these data sets.

WLP also glosses over some details of tokenization; in particular, the word count ignores whitespace. SentencePiece [44] is often advertised as being a subword tokenizer that preserves whitespace, but in

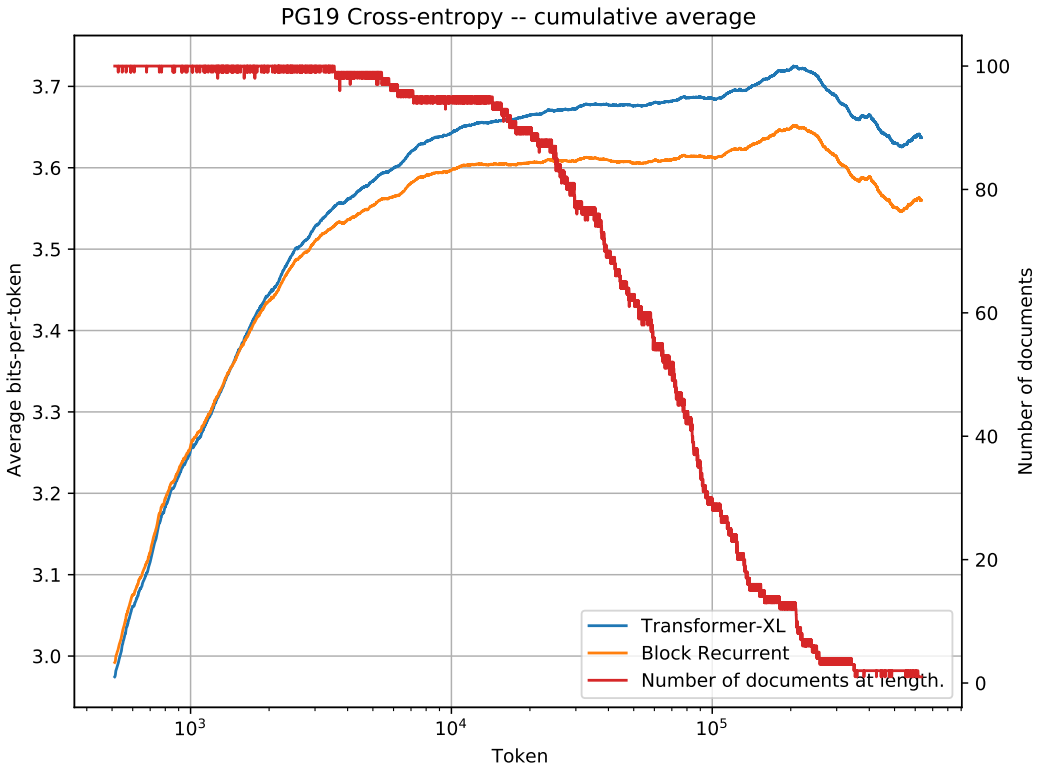

Figure 6: Cumulative cross-entropy on PG19 of a 13-layer Transforemer-XL and Block Recurrent model. Though comparable at the first few thousand tokens, the recurrent model performs better at longer sequences. In red we show the number of documents at a given token length.

fact that is only partially true. Depending on how it is configured, SentencePiece may normalize text by mapping multiple whitespace characters (e.g. tab) to space, merge internal whitespace characters, or strip all newlines from the text.

Thus, depending on vocabulary, a model trained with a SentencePiece tokenizer may or may not be predicting whitespace and newlines. If it does predict them, then that whitespace is not captured by the word count, which can affect the WLP numbers.

Measuring bits-per-character (BPC) rather than WLP is a potentially simpler and fairer comparison that works with any data set and tokenizer, but it runs into the same problem with whitespace. If the tokenizer merges or strips whitespace from the input text, as many SentencePiece vocabularies do, then it is not possible to get a consistent character count across different vocabularies.

I.1 Effect of vocabulary on model perplexity

The vocabulary itself can also make a significant difference. We argue that the tokenizer and vocabulary should be considered as part of the model, and comparisons using either WLP or BPC are only valid between models that use the same tokenizer and vocabulary. Using WLP to judge model innovations without taking into account other differences can lead to incorrect conclusions. We give an empirical example of how using different vocabularies can lead to significant differences in WLP.

The total cross-entropy loss $L_S = -\sum_{t=0}^{N_S} \log(p_t|p_{<t})$ where S is the chosen tokenization scheme and N_S is the number of tokens in the data when tokenized by S. Models using the same tokenization scheme can be compared by their *token-level perplexity* $\text{PPL}_S = \exp(\frac{L_S}{N_S})$ where $\frac{L_S}{N_S}$ is the *average loss*. Lower perplexity is better.

WLP is defined to be $\exp(\frac{L_S}{N_W})$ where N_W is the number of "words" as defined by the authors of the PG-19 datasets. For the PG-19 test set, $N_W = 6,966,499$ [15]. Thus, WLP is simply a rescaling of the average loss, i.e. WLP = $\exp(\frac{L_S}{N_S}\frac{N_S}{N_W})$, where we'll refer to $\frac{N_S}{N_W}$ as the *scale factor* – the number of tokens per word.

I.1.1 Vocabulary size

The size of the vocabulary has three obvious effects on the model. First, a larger vocabulary has more parameters, and thus can store more information in the embedding table.

Second, a larger vocabulary will typically have longer and rarer words, so the average length of each token (characters-per-token) will also be longer, and the *scale factor* (tokens-per-word) will be smaller. The context length of the model *in tokens* is usually fixed, so having longer tokens means that the model can attend over longer distances in the input text.

Third, having longer tokens also means that the model will process more of the training data on each training step (again because the number of tokens per step is fixed), and thus complete more epochs over the data set.

I.1.2 Vocabulary quality

Even if two vocabularies are the same size, there may be a difference in vocabulary quality. For example, if a multi-lingual vocabulary is used on English-language text, then only a portion of its total capacity will be used to capture English words, and it will behave much like a smaller vocabulary.

A more subtle issue is that there are many ways in which text can be tokenized. A tokenizer which does a good job of capturing natural semantic segmentation into subwords can be expected to perform better than one that doesn't, e.g. "`walk|ed`" vs. "`wal|ked`."

I.2 Experimental results

We train the `Rec:fixed:skip` model with a segment length of 4096 for 500k steps employing an inverse square root decay schedule while only varying the choice of vocabulary (all other hyperparameters are as in Section 4).

The byte-level vocabulary does not use a tokenizer, and operates on the raw ASCII text. The T5 vocabulary [39] was trained on multi-lingual data from Common Crawl (commoncrawl.org), consists of 32k tokens, and does not preserve whitespace. We also test on a 32k sized version of the LaMDA vocabulary [54]. To test the effect of vocabulary size, we also train a number of SentencePiece models on the PG19 training set with sizes ranging from 512 to 128k.

Results are shown in Table 5 and Figure 7. For SentencePiece vocabularies trained on PG19, we observe that a larger vocabulary has a significant effect on the scale factor, although the effect starts to diminish with more than 32k entries (see Figure 7). The average bits per token increases as the tokens become longer.

Table 5: Test perplexity of the `Rec:bias:skip` model when using different tokenizers. Results are given in bits-per-token, so WLP is $2^{(\text{bits-per-token} \cdot \text{scale factor})}$. STE stands for Tensorflow's deprecated SubwordTextEncoder as used by other baselines; SPM stands for SentencePiece model [44].

vocab	size	token count	scale factor	avg bits-per-token	WPL
T5 [39]	32k	10,523,460	1.5106	3.525	40.08
LaMDA [52]	32k	12,053,681	1.7302	3.121	42.23
PG19 STE	32k	11,097,364	1.5930	3.352	40.49
PG19 SPM	32k	10,229,476	1.4684	3.647	40.93
PG19 SPM	128k	9,638,472	1.3835	3.830	39.37
PG19 SPM	96k	9,711,259	1.3940	3.801	**39.36**
PG19 SPM	64k	9,856,687	1.4149	3.766	40.18
PG19 SPM	32k	10,229,476	1.4684	3.647	40.93
PG19 SPM	8k	11,559,455	1.6593	3.289	43.94
PG19 SPM	4k	12,622,101	1.8118	3.021	44.43
PG19 SPM	1k	15,865,499	2.2774	2.477	49.91
PG19 SPM	512	18,124,332	2.6016	2.125	46.16
bytes	256	41,288,269	5.9267	0.960	51.61

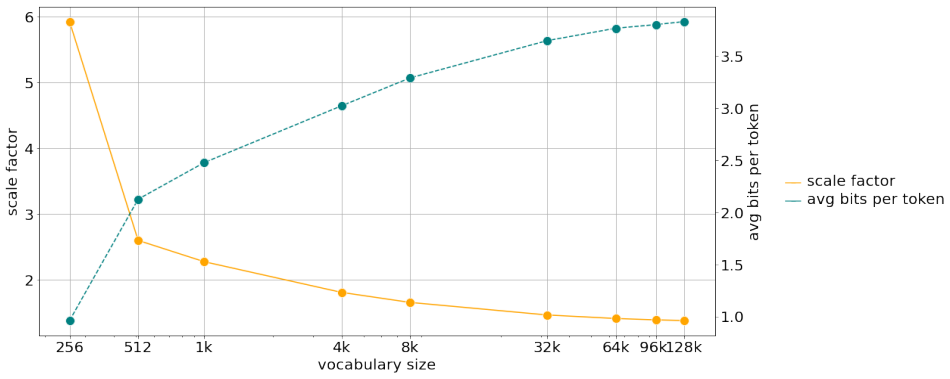

Figure 7: Word-perplexity scale factor and average bits per token for sentencepiece vocabularies trained on PG19 with different sizes.