DYNAMIC EVALUATION OF TRANSFORMER LANGUAGE MODELS

Ben Krause, Emmanuel Kahembwe, Iain Murray, & Steve Renals
School of Informatics, University of Edinburgh
Edinburgh, Scotland, UK
ben.krause,e.kahembwe,i.murray,s.renals@ed.ac.uk

ABSTRACT

This research note combines two methods that have recently improved the state of the art in language modeling: Transformers and dynamic evaluation. Transformers use stacked layers of self-attention that allow them to capture long range dependencies in sequential data. Dynamic evaluation fits models to the recent sequence history, allowing them to assign higher probabilities to re-occurring sequential patterns. By applying dynamic evaluation to Transformer-XL models, we improve the state of the art on enwik8 from 0.99 to 0.94 bits/char, text8 from 1.08 to 1.04 bits/char, and WikiText-103 from 18.3 to 16.4 perplexity points. Code to replicate our results is available[1].

1 INTRODUCTION

Language modeling is a commonly used machine learning benchmark with applications to speech recognition, machine translation, text generation, and unsupervised learning in natural language processing tasks. LSTMs (Hochreiter and Schmidhuber, 1997) conventionally used for language modeling have been shown to use relatively shorter contexts to make predictions (Khandelwal et al., 2018). Several recent improvements to language modeling have resulted from models with increased ability to use long-range dependencies. This work combines two specific advances: Transformers (Vaswani et al., 2017) and dynamic evaluation (Mikolov et al., 2010; Krause et al., 2018). Transformers can model long range dependencies through stacked layers of self-attention, and dynamic evaluation exploits certain types of long range dependencies by adapting parameters based on the observed sequence history. Dynamic evaluation can be applied to any language model at test time, but to our knowledge, no previous work has applied dynamic evaluation to Transformers.

Transformers use a combination of a self-attention mechanism and positional embeddings to encode information about the sequence history (Vaswani et al., 2017). The use of self-attention provides shorter paths for information to travel, which is conjectured to be one of the main reasons that transformers achieve better results on common language modeling benchmarks compared to other models (Dai et al., 2019). Moreover, transformers trained on very large datasets can generalize to other NLP tasks, and generate realistic samples that are coherent over long time frames (Radford et al., 2019).

Dynamic evaluation adapts models to the recent sequence history via gradient descent in order to exploit re-occurring sequential patterns. Natural language tends to have long range dependencies associated with the style and word usage of particular passages of texts; and dynamic evaluation can exploit these dependencies via online model adaptation. Transformers with a large memory cache also potentially have the capability of adapting to the style of the recent sequence history, although it is unclear to what extent they learn to do this in practice. Dynamic evaluation and Transformers have each shown their respective capabilities to use thousands of timesteps of context to improve predictions (Krause et al., 2018; Dai et al., 2019), but it is unclear how much overlap there is between the type of long-range dependencies exploited by Transformers and dynamic evaluation. If Transformers are able to fully adapt to the style of the recent sequence history, there should be little to no advantage of using dynamic evaluation. Therefore, in this work, we explore the utility of applying dynamic evaluation to Transformers.

[1] www.github.com/benkrause/dynamiceval-transformer

2 TRANSFORMERS

A number of variants of Transformers have been suggested for language modeling (Al-Rfou et al., 2018; Liu et al., 2018; Baevski and Auli, 2019; Radford et al., 2018), but in this work, we focus on the Transformer-XL architecture of Dai et al. (2019), which uses segment-level attention recurrence and a relative positional encoding mechanism to generalize to longer attention lengths than seen during training. Transformer-XL has recently improved state-of-the-art results on a number of common language modeling benchmarks.

The Transformer-XL, like the regular Transformer, contains stacked self-attention layers and position-wise feedforward operations. The Transformer-XL processes sequence segments in parallel across time in each forward pass. The hidden states from these sequence segments are cached in a memory so that future sequence segments can apply attention over them. We refer to Dai et al. (2019) for the full details of the model.

3 DYNAMIC EVALUATION

Dynamic evaluation is a gradient descent based adaptation method that can be applied to auto-regressive sequence modeling problems. Auto-regressive sequence models use the following factorization to assign a probability to a sequence $x_{1:T} = \{x_1, \ldots, x_T\}$:

$$P(x_{1:T}) = P(x_1) \prod_{t=2}^{T} P(x_t | x_{1:t-1}). \tag{1}$$

The model predicts a distribution over the next sequence element $P(x_t|x_{1:t-1})$ (or a sequence segment $x_{t_1:t_2}|x_{1:t_1-1}$). The model observes the true x_t and takes a loss based on the cross entropy prediction error, $\mathcal{L}_t$. The gradient $\nabla\mathcal{L}_t$ is then used to update the network before proceeding to the next sequence element. As in all autoregressive models, dynamic evaluation only conditions on sequence elements that it has already predicted, and so evaluates a valid log-probability for each sequence. Dynamic evaluation is illustrated graphically in figure 1.

The gradient descent adjusted weights can be interpreted as a memory that can better capture re-occurring patterns that occur in linguistic sequences. Dynamic evaluation updates were shown to have the ability to increase probabilities of words that occur in a sequence, as well as words with similar embeddings to words that occur in the sequence (Krause et al., 2018). This capability gives dynamic evaluation the potential to better model recently seen words, as well as to adapt more broadly to the style and topic of a sequence.

Following Krause et al. (2018), which applies dynamic evaluation to RNNs at the sequence segment level, we apply dynamic evaluation to Transformer-XL models at the sequence segment level. Since Transformer-XL models are designed to process sequences in segments, we align the sequence segments used for Transformer-XL (Dai et al., 2019) with the sequence segments used to compute the gradient for dynamic evaluation. The gradient is computed once for each sequence segment (after taking a loss on the segment), and backpropagation is truncated to be contained within a single sequence segment.

There is a large space of potential optimizers that can be used for dynamic evaluation, and we evaluate two in this work. We consider a simple baseline that uses stochastic gradient descent with a fixed learning rate to update the weights of the network on each segment. We also consider the more complex dynamic evaluation optimizer (Krause et al., 2018) which uses an update rule related to RMSprop (Tieleman and Hinton, 2012), except that gradient statistics are computed from the training data, and weights are decayed back to the original parameters learned during training.

4 EXPERIMENTS

We applied dynamic evaluation to pretrained Transformer-XL models from Dai et al. (2019) on two character-level datasets and one word-level dataset. We chose these 3 datasets because they all

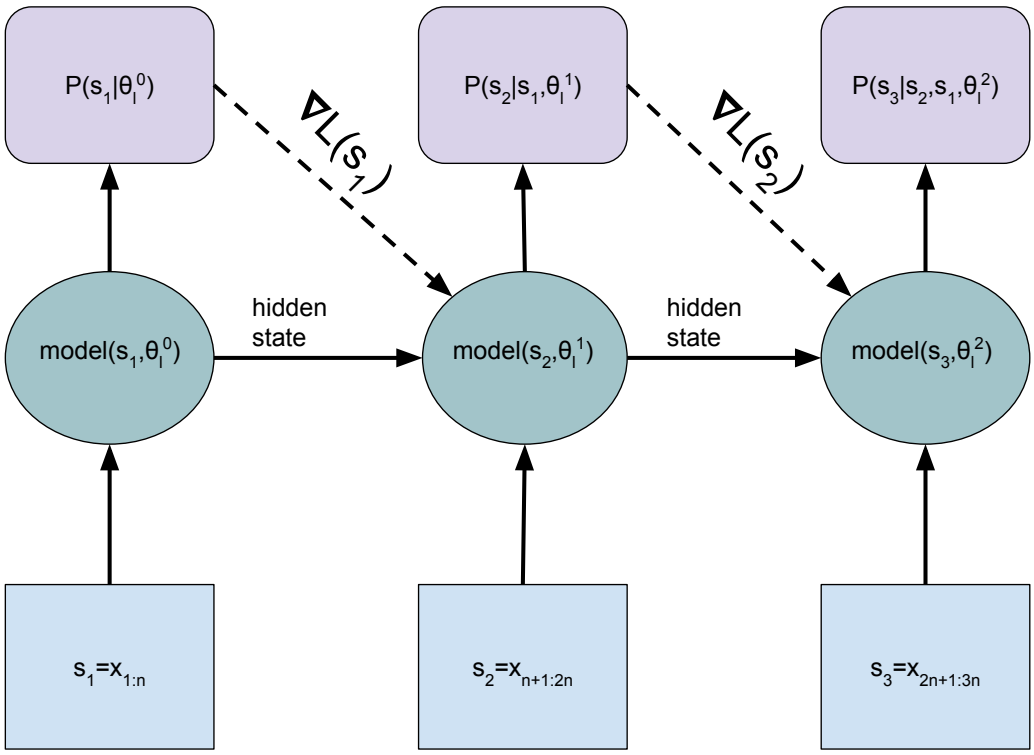

Figure 1: Illustration of dynamic evaluation (figure from Krause et al. (2018)). The model evaluates the probability of sequence segments s_i of length n. The gradient $\nabla\mathcal{L}(s_i)$ with respect to the log probability of s_i is used to update the model parameters θ_l^{i-1} to θ_l^i before the model progresses to the next sequence segment. Dashed edges are what distinguish dynamic evaluation from static (normal) evaluation.

contain long-range dependencies that span across sentences and paragraphs. Details of the model training can be found in Dai et al. (2019), and we downloaded models using their code[2].

We measured the performance of two types of dynamic evaluation; one which used the optimizer from Krause et al. (2018), which we refer to as "RMS dynamic eval + decay", and one that used stochastic gradient descent, which we refer to as "SGD dynamic eval". Following Krause et al. (2018), we tuned hyperparameters for dynamic evaluation on the validation sets before evaluating on the test sets.

4.1 Character-level experiments

We use two datasets to evaluate dynamic evaluation on character-level Transformer-XL models; enwik8 (Hutter, 2006) and text8[3]. enwik8 is a byte-level data set derived from Wikipedia that in addition to English text, also includes markup, special characters, and text in other languages. enwik8 contains 90M characters for training, 5M for validation, and 5M for testing. We noticed a slight anomaly in the preprocessing of enwik8 in the code released by Dai et al. (2019) that caused it to have 204 unique tokens (rather than the standard 205 tokens used in most results, for instance in Graves (2013)), and our results also contain this anomaly since we use pretrained models from their work. text8 is derived from the same data as enwik8, but is preprocessed to only contain an alphabet of 27 characters (lowercase a–z plus spaces). text8 also uses a 90M–5M–5M split for training, validation, and testing. Following Dai et al. (2019), we used sequence segments of 128 and a memory cache of length 3800 for both datasets. Results for enwik8 and text8 are reported in Table 1 and Table 2

[2] https://github.com/kimiyoung/transformer-xl
[3] http://mattmahoney.net/dc/textdata

Model	# of params	test
Hyper LSTM (Ha et al., 2017)	25M	1.34
HM-LSTM (Chung et al., 2017)	35M	1.32
Recurrent highway networks (Zilly et al., 2017)	46M	1.27
FS-LSTM (Mujika et al., 2017)	47M	1.25
awd-LSTM (Merity et al., 2018)	47M	1.23
Transformer + aux losses (Al-Rfou et al., 2018)	235M	1.06
Multiplicative LSTM (Krause et al., 2016)	46M	1.24
Multiplicative LSTM + dynamic eval (Krause et al., 2018)	46M	1.08
Transformer-XL (Dai et al., 2019)	277M	0.993
Transformer-XL + SGD dynamic eval	277M	0.946
Transformer-XL + RMS dynamic eval + decay	277M	**0.940**

Table 1: Character-level cross-entropy (bits/char) on enwik8. As noted in Section 4.1, there is a slight difference in the data used in the final three results and previous work.

Model	# of params	test
HM-LSTM (Chung et al., 2017)	35M	1.29
Recurrent highway networks (Zilly et al., 2017)	45M	1.27
Transformer + aux losses (Al-Rfou et al., 2018)	235M	1.13
Multiplicative LSTM (Krause et al., 2016)	45M	1.27
Multiplicative LSTM + dynamic eval (Krause et al., 2018)	45M	1.19
Transformer-XL (Dai et al., 2019)	277M	1.085
Transformer-XL + SGD dynamic eval	277M	1.042
Transformer-XL + RMS dynamic eval + decay	277M	**1.038**

Table 2: Character-level cross-entropy (bits/char) on text8.

respectively. Applying Dynamic evaluation improves the Transformer-XL by a noticeable margin, achieving state of the art on both of these character-level datasets.

4.2 WORD-LEVEL EXPERIMENTS

We evaluate dynamic evaluation on word-level Transformer-XL using the WikiText-103 dataset (Merity et al., 2017), which is also comprised of Wikipedia text. WikiText-103 contains 103 million training tokens, and a vocabulary size of 268k. Given the large vocabulary size, the pretrained model we re-evaluate from Dai et al. (2019) used an adaptive softmax output layer (Grave et al., 2017a) to make training faster. Results for WikiText-103 are reported in Table 3. There was no noticeable validation advantage to using a decay rate, so we refer to the dynamic evaluation optimizer for this experiment simply as "RMS dynamic eval", since the decay rate was tuned to be set to zero. Dynamic evaluation gave a 9% perplexity improvement to Transformer-XL on WikiText-103.

The results on WikiText-103 are the first that we know of that apply dynamic evaluation with an adaptive softmax output layer. Adaptive softmax reduces the computational expense of the output layer at the cost of giving the model less expressiveness at modeling rare words. When training a network from scratch, such a trade-off is sensible, since it is difficult to learn a good representation of rare words. However, when dynamically adapting to the recent sequence history, the adaptive softmax layer may make adapting to recent rare words more challenging. There is potential for future work improving the combination of dynamic evaluation and adaptive softmax, for instance by hybridizing it with the neural cache method (Grave et al., 2017b). The neural cache learns a non-parametric

Model	# of params	valid	test
LSTM+ neural cache (Grave et al., 2017b)	-	-	40.8
GCNN-14 (Dauphin et al., 2017)	-	-	37.2
QRNN (Merity et al., 2018)	151M	32.0	33.0
LSTM + hebbian + cache (Rae et al., 2018)	-	29.7	29.9
Transformer + adaptive input (Baevski and Auli, 2019)	247M	19.8	20.5
Transformer-XL* (Dai et al., 2019)	257M	17.3	18.1
Transformer-XL + SGD dynamic eval	257M	16.3	17.0
Transformer-XL + RMS dynamic eval	257M	15.8	**16.4**

Table 3: Word-level perplexity on WikiText-103. *We report our results using the pretrained model from Dai et al. (2019) using a batch size of 1, and achieved a slightly lower perplexity than in the original paper (18.1 vs 18.3).

output layer that is independent of the network's output layer, which may potentially allow for more expressive adaptation to rare words in models with an adaptive softmax.

5 Conclusion

Dynamic evaluation was able to give moderate improvements to strong Transformer network baselines, and improves the state of the art on all three datasets evaluated. These results demonstrate that the types of long range dependencies used by dynamic evaluation and Transformers are somewhat different, as applying dynamic evaluation to Transformers leads to further improvements. These improvements are not nearly as large as when dynamic evaluation has been applied to weaker models, suggesting that Transformers are by themselves somewhat more capable of modeling re-occurring patterns in sequences than past architectures. However, Transformers still struggle to fully exploit these repetitions, even in these experiments where training and testing data came from the same domain. Transformers may struggle to adapt even more when there is a shift between training and testing data. Our results therefore motivate future work on enhancements and architectures for adaptive sequence modeling, as current Transformer models cannot fully deal with adaptation on their own.

References

Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. (2018). Character-level language modeling with deeper self-attention. *arXiv:1808.04444*.

Baevski, A. and Auli, M. (2019). Adaptive input representations for neural language modeling. *ICLR*.

Chung, J., Ahn, S., and Bengio, Y. (2017). Hierarchical multiscale recurrent neural networks. *ICLR*.

Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv:1901.02860*.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *ICML*.

Grave, E., Joulin, A., Cissé, M., Jégou, H., et al. (2017a). Efficient softmax approximation for GPUs. In *ICML*.

Grave, E., Joulin, A., and Usunier, N. (2017b). Improving neural language models with a continuous cache. *ICLR*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv:1308.0850*.

Ha, D., Dai, A., and Lee, Q. (2017). Hypernetworks. *ICLR*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

Hutter, M. (2006). The human knowledge compression prize. *URL http://prize.hutter1.net*.

Khandelwal, U., He, H., Qi, P., and Jurafsky, D. (2018). Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Krause, B., Kahembwe, E., Murray, I., and Renals, S. (2018). Dynamic evaluation of neural sequence models. *ICML*.

Krause, B., Lu, L., Murray, I., and Renals, S. (2016). Multiplicative LSTM for sequence modelling. *arXiv:1609.07959*.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *ICLR*.

Merity, S., Keskar, N. S., and Socher, R. (2018). An analysis of neural language modeling at multiple scales. *arXiv:1803.08240*.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. *ICLR*.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Mujika, A., Meier, F., and Steger, A. (2017). Fast-slow recurrent neural networks. *NIPS*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://openai.com/blog/language-unsupervised/*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *URL https://openai.com/blog/better-language-models/*.

Rae, J. W., Dyer, C., Dayan, P., and Lillicrap, T. P. (2018). Fast parametric learning with activation memorization. *ICML*.

Tieleman, T. and Hinton, G. E. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *NIPS*.

Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2017). Recurrent highway networks. *ICLR*.