Breaking the Transformer Bottleneck

Christopher Benka, Nicholas Hammer, Kelly Su

June 10, 2021

Abstract

The expressiveness of Softmax-based models are limited by the Softmax bottleneck. A solution to this issue is the Mixture of Softmaxes (MoS) method, previously shown to improve the performance of RNN-language models. However, RNN-language models are no longer state-of-the-art on language modeling tasks, with GPT and BERT cementing transformer-based language models as ultramodern. Little attention, though, has been given to the effect of the Softmax bottleneck on Transformer-based language models.

In this paper, we explore the effect of the Softmax Bottleneck on an encoder-only Transformer for the purpose of language modeling. Being a Softmax-based method, we find that the Transformer architecture also suffers from the Softmax bottleneck. Experimental results indicate applying a Mixture of Softmaxes reduces perplexity by 6.7% on the Penn TreeBank dataset and 14.1% on the WikiText-2 dataset. We also show that MoS on Transformers further mitigates the perplexity for up to 10 Softmaxes and declines at 15 Softmaxes.

1 Introduction

The Transformer model is an ultramodern, attention-based deep learning model that has rapidly become the architecture of choice for language-modeling problems since its onset in 2017 [1]. It has since replaced older RNN models such as long short-term-memory (LSTM) and led to the development of pre-trained systems such as the Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT) [2, 3]. Like RNNs, Transformers are designed to handle sequential input data and also use small latent embedding vectors and Softmaxes. Unlike RNNs, though, they do not require that the sequential data be processed in order. They utilize self-attention and multi-headed attention mechanisms to improve language and context dependencies across corpora and typically consist of an encoder-decoder architecture. The last decoder is followed by a final linear transformation and then a Softmax layer that produces the output probabilities over the vocabulary. In this paper, we are particularly interested in improving this Softmax layer.

The Softmax layer is commonly used as an output activation function for modeling categorical probability distributions in many deep learning applications. However, the Softmax's distributed word embeddings do not have sufficient capacity to model natural language, as they are limited by a relatively small dimensionality. Since the length of the hidden vector is much smaller than the vocabulary size, the Softmax function cannot completely learn the true probability distribution. This is termed the "Softmax bottleneck", where the Softmax layer does not effectively capture the context and relational dependencies of natural language. To address this limitation, the Mixture of Softmaxes solution was proposed, originally motivated by translating language modeling into a matrix factorization problem. MoS is more expressive in modelling natural language and has shown to improve perplexity on RNN models [4].

2 Previous Works

Zhilin Yang and his colleagues at Carnegie Mellon University initially termed the "Softmax Bottleneck" as well as proposed MoS as a solution. The paper illustrated that MoS mitigates the negative effects of the bottleneck in LSTMs and substantially improves current state-of-the-art benchmarks by up to 3.6 points in perplexity. It achieves 47.69 on the Penn Treebank, 40.68 on the WikiText-2 dataset, and outperforms the baseline of the 1B Word dataset by over 5.6 points in perplexity [4].

An additional proposed solution was considered in a non-parametric model, namely n-grams.

With enough parameters, n-gram models are not restricted by parametric forms and are efficient at modeling natural language. This method is an “easy fix”, however, as it significantly increases the number of parameters when compared to Softmax, which can cause overfitting [5].

The Transformer model was introduced in [1] and pre-trained models BERT and GPT followed respectively. Both models have shown to improve performance over a wide variety of NLP tasks, including but not limited to sentiment detection, text summarization, and translation [2, 3]. Both BERT and GPT utilize the Softmax layer in the architecture of their final decoder. Given that the Transformer model has surpassed RNN models and is the current state-of-the-art NLP architecture, we investigate if MoS can also benefit Transformers.

3 Model

To summarize the crux of the Softmax bottleneck, let θ be the set of parameters, X be the set of samples, and $\mathbf{H}_\theta \in \mathbb{R}^{B\times d}$, $\mathbf{W}_\theta \in \mathbb{R}^{M\times d}$, $\mathbf{A} \in \mathbb{R}^{N\times M}$ be matrices of context vectors, word embeddings, and log probabilities of the true data distribution, respectively. With the Softmax activation function, we aim to model $\mathbf{A}$ with $\mathbf{A}' = \mathbf{H}_\theta \mathbf{W}_\theta^T$ such that

$$\Pr_\theta(X \mid c) = \Pr{}^*(X \mid c)$$

for every context c, where the probabilities come from $\mathbf{A}^*$ and $\mathbf{A}$, respectively. The issue, then, is stated in *Corollary 1* in [4]:

Corollary 1 *If* $d < \operatorname{rank}(\mathbf{A}) - 1$, *for any function family* $\mathcal{U}$ *and any model parameter* θ, *there exists a context* c *in* $\mathcal{L}$ *such that* $\Pr_\theta(X \mid c) \neq \Pr^*(X \mid c)$.

The authors of [4] hypothesize that since natural language is so context-dependent, the matrix $\mathbf{A}$ should be high-rank – certainly much higher than d – and thus the Softmax bottleneck limits the expressiveness of models which use it, including Transformers, by not modeling the probabilities generated after the true probability distribution.

MoS is a high-rank language model that was proven by the aforementioned authors to alleviate this issue. For K Softmax distributions and T contexts, this model formulates the conditional distribution as

$$\Pr_\theta(x \mid c) = \sum_{k=1}^{K} \pi_{c,k} \frac{\exp \mathbf{h}_{c,k}^T \mathbf{w}_x}{\sum_{x'} \exp \mathbf{h}_{c,k}^T \mathbf{w}_{x'}}$$

where $\pi_{c,k}$ is a probability distribution of the mixture weight of the k-th component and $\mathbf{h}_{c,k}$ is the k-th context vector associated with context c. The prior and the context vectors for context c_t are parameterized as $\pi_{c_t,k} = \frac{\exp \mathbf{w}_{\pi,k}^T \mathbf{g}_t}{\sum_{k'=1}^{K} \exp \mathbf{w}_{\pi,k}^T \mathbf{g}_t}$ and $\mathbf{h}_{c_t,k} = \tanh(\mathbf{W}_{h,k}\mathbf{g}_t)$. Here, $\mathbf{w}_{\pi,k}$ and $\mathbf{W}_{h,k}$ are model parameters and $\mathbf{g}_t$ is the hidden state output of the model employed with respect to context c_t, where $t \in [T]$. Essentially, by increasing the number of Softmaxes under consideration and taking the weighted-average (according to some distribution π), the rank of $\mathbf{A}'$ increases, thereby producing a highly expressive model.

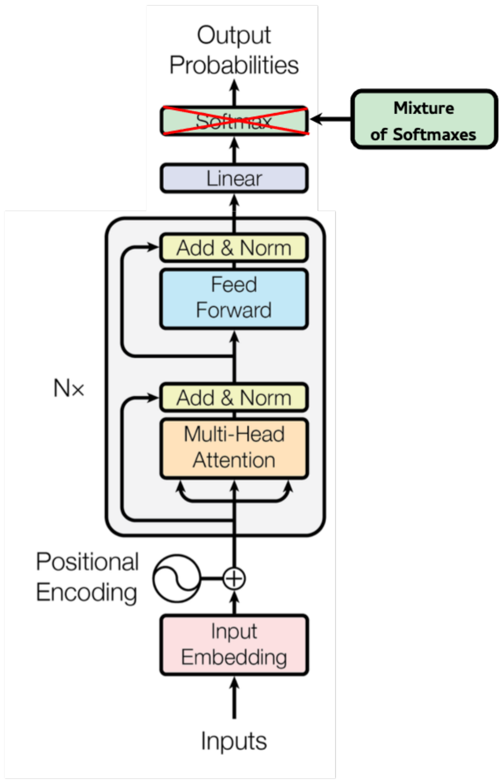

Figure 1: Encoder-Only Transformer-MoS [1]

To readily see the effects of Mixture of Softmaxes, we examine an encoder-only Transformer that replaces the original decoder which has multiple layers of self-attention blocks with a simply linear layer; this architecture is essentially the same as BERT. The experimental Transformer-MoS variation uses the MoS solution instead of the usual Softmax layer after the decoder. Since MoS increases the expressiveness of the model by allowing for different Softmaxes to ‘focus’ on different parts of the true probability distribution, we expect the Transformer-MoS model to perform significantly better than the stock Transformer.

	Penn TreeBank		**WikiText-2**	
Dataset	Transformer	Transformer-MoS	Transformer	Transformer-MoS
Validation	100.04	91.88 8.9% reduction	162.83	140.57 15.8% reduction
Test	92.95	86.89 6.7% reduction	153.38	131.75 14.1% reduction

Table 1: Perplexities for Transformer and Transformer-MoS. Markedly, Transformer-MoS outperforms the baseline Transformer model on both Penn TreeBank and Wikitext-2.

4 Experiments

We compare the performance of the standard encoder-only Transformer and Transformer-MoS models on two language modeling datasets, Penn TreeBank and WikiText-2. WikiText-2 is a collection of two million tokens extracted from the set of verified “Good” and “Featured” articles on Wikipedia with a vocabulary size of around thirty-three thousand. The Penn Treebank dataset is made up of 2,499 stories from a Wall Street Journal collection of 98,732 stories for syntactic annotation.

We evaluate the models’ validation and test perplexity on each dataset. To maintain a fair comparison, hyperparameters defining the transformer-encoder portion of the architecture are held consistent between the Transformer and Transformer-MoS. For each dataset, we perform a manual search of hyperparameters that decrease validation perplexity. To examine the efficacy of MoS as a function of the number of mixtures, we also train the Transformer-MoS with a varying number of mixtures on the Penn TreeBank dataset. We record test perplexities and training time to explore the trade-off in accuracy and time to train.

4.1 Training Details

We train the baseline and Transformer-MoS on an NVIDIA RTX-2080ti GPU for 50 epochs and use stochastic gradient descent to update the learned parameters with an initial learning rate of 7. At the end of each epoch, the model being trained is evaluated on the validation set, where the learning rate is divided by 1.75 anytime the validation loss does not improve. The full encoders of these models consist of 4 Transformer encoder layers. We use an input and output dimension of 200, 200 for number of feed forward hidden units, and .2 for dropout regularization. The training data is split into batches of size 20 with 35 source tokens.

5 Results

5.1 Language Modeling

Table 1 summarizes our primary results. The Transformer-MoS outperforms the baseline Transformer model on both language modeling datasets. On the Penn TreeBank dataset, the Transformer-MoS achieves a 6.7% reduction in perplexity. With an RTX-2080ti, training the Transformer-MoS model takes 32 minutes and training the Transformer model takes 21 minutes. On WikiText-2, the Transformer-MoS also outperforms the baseline model, reducing perplexity by 14.1%.

5.2 Variation of Mixtures

N Mixtures	PPL	Training Time
2	89.2	19.20 ms/batch
5	88.58	21.59 ms/batch
10	86.89	28.34 ms/batch
15	90.69	41.47 ms/batch

Table 2: Perplexities and training time on Penn TreeBank with varying number of Softmaxes.

To evaluate the effect of the number of mixtures on the performance of the proposed model, we vary the number of Softmaxes used in the decoder on the Penn TreeBank dataset. As summarized in Table 2, increasing the number of Softmaxes up to 10 mixtures increases the performance of the model. At 15 mixtures, the model’s performance begins to degrade, likely due to overfitting.

6 Discussion

Our theoretical discussion and experimental results indicate that the Transformer architecture is also limited by the Softmax bottleneck. This result is unsurprising as like RNNs, Transformers relies on low dimensional word embeddings with dimensionality insufficient to model the expressiveness of language. Fortunately, the MoS technique applied to Transformers is able to break the Softmax Bottleneck, producing a noticeable reduction in perplexity on both language modeling datasets. Our results further establish the Softmax Bottleneck as an issue that implementations of Transformers must attend to. The code, Transformer, and Transformer-MoS models are available on Github[1].

We are, however, concerned with the efficiency of the MoS technique. Applying the MoS technique results in nearly 1M added parameters and increases training time by approximately 1.5 times. Further research should focus on techniques that more efficiently address the Softmax Bottleneck, particularly in ways to reuse the parameters between Softmaxes.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Alec Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[4] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank rnn language model, 2018.

[5] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1, 1995.

[1]https://github.com/ChrisBenka/Transformer_MOS