
Simplified Attention : Kernels are all you need!

Rishabh Sharma*

Department of Computer Science
University of Wisconsin-Madison
rsharma228@wisc.edu

Abstract

In this assignment, we propose a novel modification to the self-attention mechanism used in transformer architectures. Traditional self-attention mechanisms rely on two distinct vectors, the key and the query, to compute the attention weights that determine how much focus each token should place on others. Our approach simplifies this process by utilizing the same vector for both the key and the query. This design reduces the complexity of the attention computation, and creates a symmetric self-attention matrix. We show empirically that this reduced attention can still preserve much of the useful contextual relationships between tokens; and work well on generation tasks.

1 Motivation

In a typical self-attention layer, every token has three vectors, the query (q), key (k) and value (v). Self-attention is computed by first doing a dot product each key-value pair in sequence. This gives us a "attention-matrix" which we can then pass each row through a soft-max activation and multiply each score with the respective value vector. This gives us the self-attention for this sentence.

$$\text{Traditional attention matrix: } \begin{bmatrix} q_1 k_1 & q_1 k_2 & q_1 k_3 & \dots \\ q_2 k_1 & q_2 k_2 & q_2 k_3 & \dots \\ q_3 k_1 & q_3 k_2 & q_3 k_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Note that the matrix above is asymmetric. But, here's the thing—do we really need both the query and key vectors for this? A possible reason for having an asymmetric matrix is to allow the model to learn directional relationships. For example in a sentence "A frog jumps into the river." The self-attention of the word 'frog' wrt 'river' can be different for 'river' wrt 'frog'. But we claim that positional embedding are enough to allow the model to learn this directional relationship.

Based on the motivation above we propose to simplify things by using the same vector, z , for both the query and key. Now, the attention matrix looks like this:

$$\text{Our simplified attention matrix: } \begin{bmatrix} z_1 z_1 & z_1 z_2 & z_1 z_3 & \dots \\ z_2 z_1 & z_2 z_2 & z_2 z_3 & \dots \\ z_3 z_1 & z_3 z_2 & z_3 z_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

By using z as both the query and key, we simplify the attention calculation, reduce the number of parameters, and make the model more computationally efficient.

*

2 Empirical Setup

To evaluate the performance of our simplified attention mechanism, we conduct experiments using a toy character-level GPT model. In this setup, the task is next-character prediction, where the model is trained to predict the next character in a sequence rather than the next token.

For the simplified attention model, we modify the architecture to use the same vector for both the key and query in the self-attention mechanism, as described earlier. We evaluate both models using different configurations of embedding dimensions and attention heads to examine the performance trade-offs between traditional and simplified self-attention.

2.1 Training Setup

Both models are trained on a toy dataset of Shakespeare’s text, tokenised at the character level. We use the following settings as the default (unless explicitly specified)

- Optimizer: Adam with a learning rate of $1e-3$ (with cosine weight decay)
- Batch size: 64
- Context length: 256 characters
- number of heads: 6
- number of layers: 6
- embedding dimension: 384

Later we will show the performance of this baseline on a different dataset (text scraped from the novel ‘IT’ by Stephen King) as an ablation study. We will also study the transfer learning performance of the baseline model.

2.2 Evaluation

We measure the train loss, validation loss throughout the training. Although the validation cross entropy loss tells us how close the generated text is to the validation set, we use additional metrics to evaluate the performance of the model. We use BERTScore[1] as a ‘specific’ metrics to compare the generated text with the validation set, and perplexity[2] as a ‘general’ metric to evaluate the generated text.

2.2.1 Specific Metric

The BERTscore measures the pairwise cosine similarity between each token of the generated text and the validation set text in the bert-embedding space. This is a good way to measure the similarities between the two sets Without penalising the semantic differences (which happens in the n-gram based evaluation methods). For eg. a sample reference sentence "it is freezing today" can be generated as "the weather today is very cold", the word ‘freezing’ is not present in the generated text, which will lower the recall for the model. Also the generated text has additional words ‘weather’ and ‘cold’ which will penalise the precision in word-matching algorithms. BERTScore is a better metric to evaluate in such cases.

2.2.2 General Metric

Perplexity of a distribution gives us an idea of the amount of information contained (when derived from entropy). In the context of evaluating a generative language model, a low value of perplexity means the model is certain about the distribution it generated.

$$\text{Perplexity}(P) = 2^{H(P)} = 2^{-\sum_x P(x) \log_2 P(x)}$$

3 Results

We first present the results for the baseline GPT model with traditional self-attention. Figure 3 shows the training and validation loss over the course of training. We have tried different architectures to



Figure 1: *

(a) Train loss on the baseline (b) Train loss on a 12 heads model (c) Train loss on 768 emb-dim

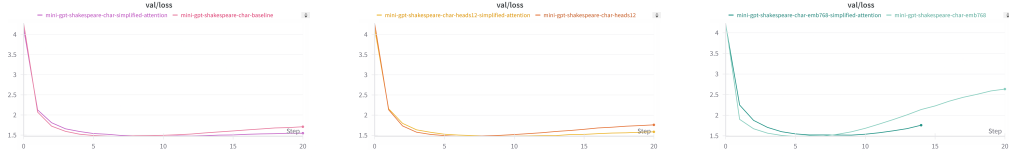


Figure 2: *

(d) Val loss on the baseline (e) Val loss on a 12 heads model (f) Val loss on a 768 emb-dim model

Figure 3: Training and validation curves for the toy character-level GPT model. (a)-(c) show the training metrics, while (d)-(f) display the corresponding validation metrics. Each plot has two models, a baseline attention model, and a simplified attention model

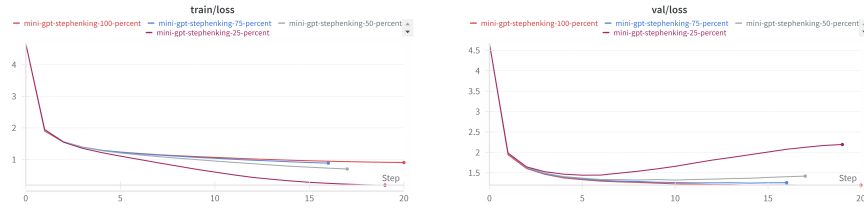


Figure 4: Training and validation plots for models given different quantities of training data. (The validation set is constant in these comparisons)

test how robust are our findings. The first observation is the model the baseline model is over-fitting on the training dataset, but the over-fitting of the simplified attention model is lagging behind its baseline. But both models achieve very similar lowest validation scores, as we see in Table 1. Also the simplified attention model is performing very similar if not better in the other evaluation metrics.

3.1 Results on a different dataset

In this section we will try a different dataset taken from the novel 'IT' by Stephen King. We try giving the model different proportions of data, and we measure the trend of performance across different proportions. The performance is fairly what was expected, as seen in Figure 4, the model is unable to overfit on the full dataset, but overfits very quickly on the smaller training sets. Also Table 2 shows the performance generally increases with dataset size. (We were unable to try the simplified attention here due to time constraints)

3.2 Results on transfer learning

Here we are checking whether a model pre-trained on the Shakespeare dataset reduces the time required to learn the new Stephen King dataset. Typically the model takes 8-10 epochs to learn the distribution of the Stephen King dataset Figure 5, but when we start from the pre-trained model, we start getting good results after 2 epochs. The val loss goes down to 1.205 and the BERTScore and perplexity values are 0.794 and 3.832. these numbers are comparable with the model trained from scratch on the complete dataset in Table 2, but with a small number iterations required. This shows the model trained on the Shakespeare dataset can be easily transferred to the Stephen King dataset.

Table 1: Comparison of model pairs based on validation score, BERTscore, perplexity, and a sample of generated text. The SA is the simplified attention model. (BERTScore and perplexity are averaged over 10 samples)

Model	Val Loss	BERTScore	Perplexity	Sample Generated Text
baseline	1.469	0.8199	4.011	AUTOLYCUS: I am a pressed cocklaine to that speak you: sir, I can perceive your highness that you have done evils of all distill
SA	1.468	0.8253	3.843	MENENIUS: We will tell thee! When she ridde. The market-plague in his country, Lest as he be, her parting to her country,
baseline w12heads	1.472	0.8172	3.963	Clown: If you were the thing indeed William that is so, and doubt call him now passion his burther mother
SA w12heads	1.477	0.8216	3.761	Provost: A graciously indeed liege, my lord: the princes With falls of my royalty gracious liege.
baseline w768dim	1.487	0.8060	4.174	What is the world, which we should so bury To take and some gallows: but he was called by The thought with my faces and honour
SA w768dim	1.517	0.8189	3.772	First Gentleman: Now queen, that did holy surcess of him, and how my noble look in his tomb.

Table 2: Comparison of different amounts of data given to the model on a new dataset (BERTScore and perplexity are averaged over 10 samples, validation set is not changed across these evaluations)

Percentage of Data	Val Loss	BERTScore	Perplexity	Sample Generated Text
25	1.442	0.7772	3.723	The revrious of the previous The tail was not really all the came for Emerican American and Georgie's Dead, Eddie was the upstairs, who had drawn up with a story of the rain who struggled around him. The blood showed up to his eyes showed in his shoulder
50	1.322	0.7942	3.531	She stood there alone at the house again, and the three bullets had been gone with men. The dryers were a circle of a mon Provost: A graciously indeed liege, my lord: the princes With falls of my royalty gracious liege.
75	1.248	0.7983	3.277	
100	1.200	0.7932	3.193	

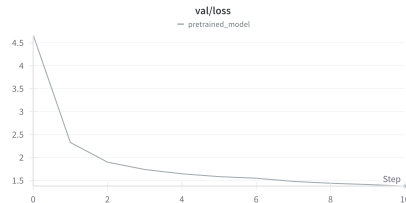


Figure 5: Validation plot for finetuning a model originally trained on the Shakespeare dataset to the Stephen King dataset, the model quickly starts performing well on the new dataset

4 Conclusion and Discussion

Based on the results seen in previous sections, we can say that the simplified attention layer works well on this toy task of character prediction across different architectures. However it is uncertain whether this will scale-up when trying with bigger tasks and larger architectures. (We wanted to try this on a larger task like replicating the full gpt-2 training pipeline, but couldn't due to compute constraints) The gains achieved from this are also not very significant, we only reduce the number of parameters in the attention layers by a third, which is not a lot when we account for the fully connected layers. However if this approach can scale up, there is potential for huge efficiency gains in both training and inference.

5 Prompting Experiments on Chat-GPT

5.1 Zero-Shot Prompting vs In-Context Prompting

Taking a break from training and simplified attention, in this section we will try to evaluate OpenAI's Chat-GPT model on some fun tasks. We start by giving a word problem with big numbers, chat-gpt fails in this case. You can have a look at Figure 6, however, giving a single example improves the performance by a lot, which you can see in Figure 7.

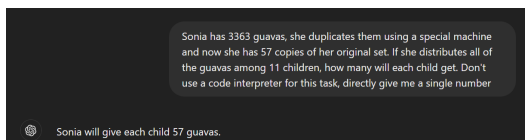


Figure 6: Chat-GPT fails on a word problem with big numbers

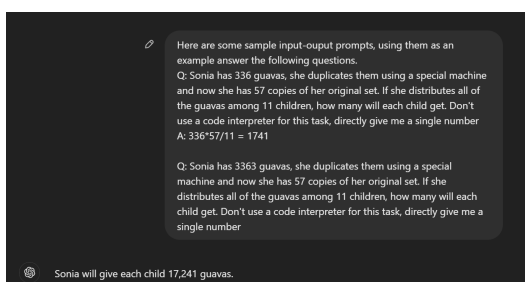


Figure 7: Showing an example brings the model very close to the actual output

5.2 Majority Voting

Trying the same problem shown in Figure 6, and generating multiple responses does not work as well. Even when varying the temperature parameter no prompt is able to give the correct answer here. I tried to find other prompts with this property, but mostly the Chat-GPT (o) is able to answer it directly. When the model is wrong (in problems like arithmetic, reasoning based word problems), the majority is also wrong. Asking the Chat-GPT model to find me such a prompt generates this response:

Here's a prompt that can produce varied outputs based on temperature settings, which might lead to a correct answer through a majority vote:

Prompt: "What are the implications of the Schrödinger's cat thought experiment in quantum mechanics, and how do they relate to the observer effect? Discuss different interpretations, including Copenhagen, Many-Worlds, and pilot-wave theory, and their philosophical ramifications."

By varying the temperature, responses may differ significantly, potentially leading to a consensus on the key ideas related to the thought experiment.

5.3 ROT13

Trying Zero-Shot rot13, the model is unable to answer correctly. Figure 8.

Moving to a "In-context" learning prompt, providing additional answers also doesn't work as well. Figure 9

Adding more information to the prompt by adding the definition and decoded versions also didn't work. Figure 10

In order to get the correct response from the model, we have to give some additional instructions asking the model to think step-by-step. This finally allows the model to answer correctly. Figure 11

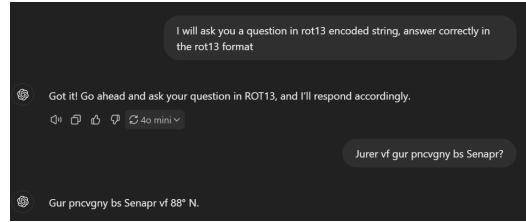


Figure 8: Chat-GPT fails on zero shot ROT13. Question is "What is the Capital of France" the decoded answer is "The capital of France is 88° N"

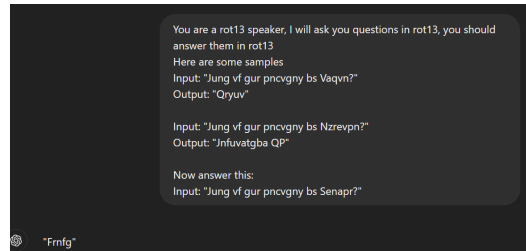


Figure 9: Chat-GPT fails on in context ROT13. The in-context questions answers are "What is the Capital of India" "Delhi" and "What is the capital of America" "Washington DC"

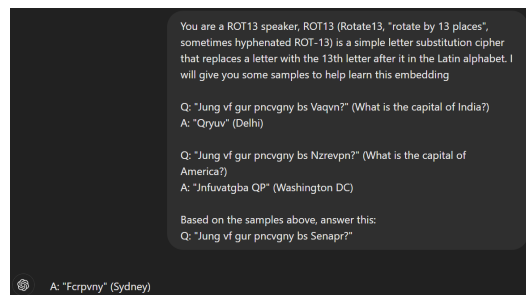


Figure 10: Chat-GPT fails with more instructions provided in the prompt

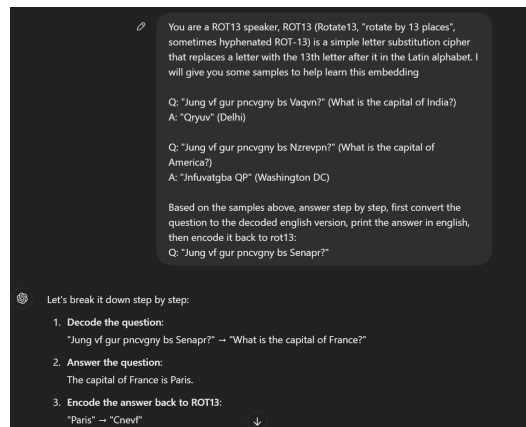


Figure 11: Chat-GPT works on rot13 with chain of thought prompting

References

- [1] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 [cs.CL]. URL: <https://arxiv.org/abs/1904.09675>.

- [2] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. “Perplexity—a measure of the difficulty of speech recognition tasks”. In: *The Journal of the Acoustical Society of America* 62.S1 (Aug. 2005), S63–S63. ISSN: 0001-4966. DOI: 10.1121/1.2016299. eprint: https://pubs.aip.org/asa/jasa/article-pdf/62/S1/S63/11558910/s63_5_online.pdf. URL: <https://doi.org/10.1121/1.2016299>.