Document-Level Machine Translation with Hierarchical Attention

Experiments with baseline attention-based machine translator

Presented to Prof. Chris Pollett
Department of Computer Science
San Jose State University

In Partial Fulfillment
Of the Requirements of the Class
Fall 2022: CS 297

By
Yu-Tang Shen
November 2022

TABLE OF CONTENTS

I. Introduction on Attention Mechanism

LSTMs addressed the gradient vanishing problem, often found in models such as recurrent neural networks [1], but the vanishing effect was still noticeable when the distance between two related items in a sequence was beyond the capability of the memory. Bahdanau et al. [2] introduced the attention mechanism to mimic human translators: cutting a long sentence into smaller fragments and processing each of them. Instead of trying to memorize all the necessary context, the attention mechanism scanned through the passage and looked for the related terms, mimicking human translators paying attention to the keywords in passages. "Attention" described how related two items were. For instance, the term "device" had a high attention with "computer" could imply "device" was referring to "computer" in a sentence.

[3] claiming "attention is all you need" in 2017 further promoted the success of attention models in NMT. The model proposed by Bahdanau et al. [2] was not parallelizable since RNN was part of the implementation, while [3] designed a highly parallel model and improved the translation quality. Every component of the input sequence could be processed independently on different processors, where each of which computed the attention and determines the translation.

Attention mechanism applied three different information extraction functions to each component in the input sequence: query, key, and value, where all three functions are linear transformation of the input value as follows:

Query: $q_i = W_q a_i$

Key: $k_i = W_k a_i$

Value: $v_i = W_v a_i$

where W were trainable weights for different functions and a were inputs for to be parsed

With all *q, k, v* values for each component in the sequence, the attention score between component was computed as $\alpha_{1,2} = q_1 \cdot k_2$, where $\alpha_{1,2}$ is the attention score between first and second items. All *q, k, v* permutations were multiplied to obtain all attention scores, and all the attention scores would be normalized

and multiplied to all corresponding v_i. The output of an attention layer was $softmax(QK^T)V$, where Q, V were matrices of q and v values, and K^T was the transpose of k values.

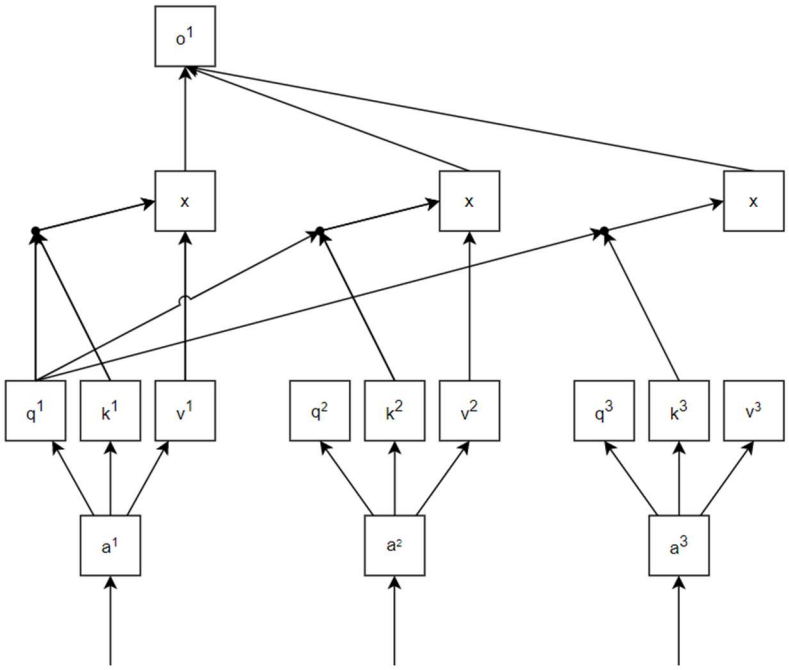

Fig 1. Output of a component after one attention layer

The advantages of attention model compared to other seq2seq model such as LSTM included being more efficient and to resolve relationships that were further from each other. Each LSTM node needed to wait for its previous node to complete to proceed, i.e., LSTM nodes were dependent to each other. And such dependency made parallel execution impossible. However, attention mechanism computed three different functions on each input and could be parallelized. For example, in Fig 1., o^1, o^2, and o^3 could be computed in parallel, because they were independent to each other. Although LSTM could preserve information coming from positions that were far from the current ones, the information was diluted after each LSTM node such that the information would eventually vanish. Instead of solving the gradient vanishing problem, LSTM offered a mitigated remedy. Attention mechanism provided a seemingly thorough solution to the gradient vanishing problem: by performing attention to the entire sequence, all the relevant information could be observed from every position.

Ideally, performing attention to all components in sequences would preserve all information embedded between them, but doing so would require a great amount of computing resources. To calculate attention between components in a n-length sequence, it would require n^2 computations: calculating attention for one component required $(q_i k_0 + q_i k_1 + \cdots + q_i k_n)v_i$, and therefore for all the components n^2 computations were needed. Thus, attention models such as Transformer [3], BERT [4], etc., had maximum input length limitations such that the models could be efficient and powerful.

Attention mechanisms were then deployed to applications such as graph analysis tasks [5], summarizing, text classification, sentiment analysis, and computer vision [6]. As described in the previous paragraph, attention models limited the window to perform attention, and since more applications started utilizing attention mechanisms, protocols to alleviate the hardware stress were delivered. [7] presented a pattern for performing attention: instead of computing attention to all permutations, compute attention for those in the following three categories: i) ones that were close to each other, ii) one of the components in the permutation was the first or second component, and iii) ones that were randomly selected. In Fig 2., the new attention pattern proposed by [7] was presented.

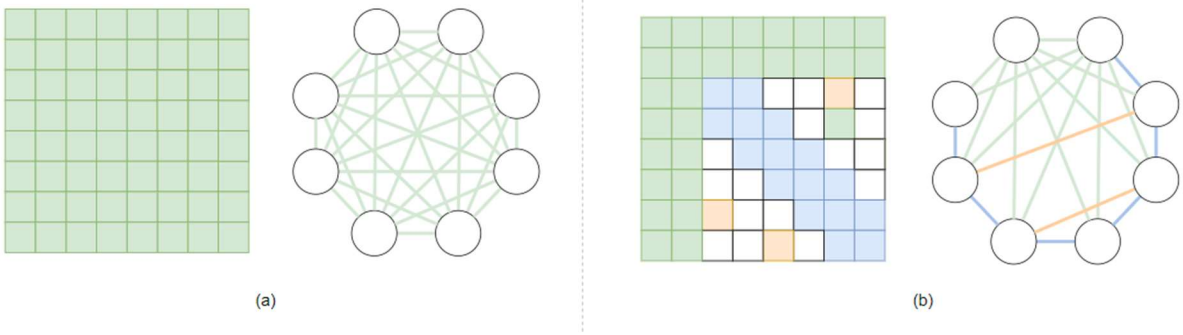

Fig 2. (a) Full attention compared to (b) Big Bird attention

[7] demonstrated similar, some even better, results compared to other attention models. The first kind of pattern listed in the previous paragraph, corresponding to blue ones in Fig 2(b), were inspired by the observation that contexts in NLP displayed a high locality of reference, and was also supported by [8]. The second type of pattern was designed by theoretical analysis by the authors and was proven critical to the

model performance. By attending all tokens to O(1) specific tokens preserved the expressive power of the model. Finally, each token randomly selected *r* tokens to attend to such that the distance between any two tokens was logarithmic to the number of tokens.

However, there was no evidence listed in [7] indicating the proposed model could achieve similar results within a same amount of time. The paper proved that the Big Bird model required more layers to achieve tasks that could be achieved by one layer of full-attention. Although Big Bird required less hardware resources, it was uncertain if the configuration yielded a shorter computation time.

II. Attention Mechanism Configuration

With numerous attention-based model available [5] [6], this report aimed to conduct experiment on the ones proposed by [3] and [7]. This report used the news corpus in [9] as the data for training and testing, where the longest English sentence had 575 words and the longest Chinese sentence had 713 words. In both configurations, SentencePiece [10] tokenization was adopted.

1. Attention model

   The Transformer model was configured with one layer of encoder and one layer of decoder, where full attention (compute attention for all token permutation) was deployed.

2. Big Bird model

   Although [7] didn't experiment Big Bird on translation tasks, this report tried to explore its capability to translate English sentences to Chinese. The most similar task done on the original paper was summarization, so this report adapted the configuration used in summarization task. The translation model was configured with 12 layers of Big Bird encoder and 6 layers of Big Bird decoder.

III. EXPERIMENT

1. Attention model

The Transformer model obtained a BLEU score of 10.45 after training for 30 epochs. It was much lower than the original paper listed, where it achieved 41 BLEU score on English to French translation. The original paper trained with 6 encoder layers and 6 decoder layer on 45 million pairs of sentences, while the experiment conducted in the paper had 1 encoder layer and 1 decoder layer trained on 130 thousand pairs of sentences.

However, some intuitive results would be shown to provide insights on the attention model. In Fig 3., some acceptable results were shown. The example showed that the model could learn phonetics ("Diego" was translated in an acceptable but different way), semantics ("can" , "how", and "find out" were all translated into synonyms), and sentence structures (the model successfully attend 2005 to tokens that were in the front and put it in the front).

```
[Source]: diego had to get off at the same stop .
[Prediction]: 迪戈必须在同一个停止停止 。
[Original]: 蒂亚哥也在同一个车站下了车 。

[Source]: do you think he can get down from the tree ?
[Prediction]: 你觉得他可以从树上下来 ?
[Original]: 你认为他能从树上下来吗 ?

[Source]: people love to find out " how " to do things .
[Prediction]: 人们喜欢找出 " 怎么做 " 的事情 。
[Original]: 人们都喜欢了解 " 如何 " 做事情 。

[Source]: statistical data from the chinese authorities shows that bilateral
          trade between portugal and china in the first half of 2005 totaled us$908 million .
[Prediction]: 统计数据显示 , 2005年上半年葡萄牙和中国当局之间的贸易数据显示 ,
          2005年上半年增长9.08亿美元 。
[Original]: 中国官方的统计数字显示 , 2005年上半年的中葡双边贸易额为9.08亿美元 。

[Source]: senior u . s . and pakistani military leaders met this week on
          an american aircraft carrier to discuss the violence .
[Prediction]: 美国和巴基斯坦军事领导人本星期在美国飞机上讨论了暴力活动 。
[Original]: 美国和巴基斯坦军方高级领导人本周在美国一艘航空母舰上讨论了暴力问题 。
```

Fig 3. Some acceptable results

Still, some unwanted results were also observed. In Fig 4., the stuttering translation saw on LSTM NMT model was observed in this model. In Fig 5., domain-specific jargons were wrongly translated.

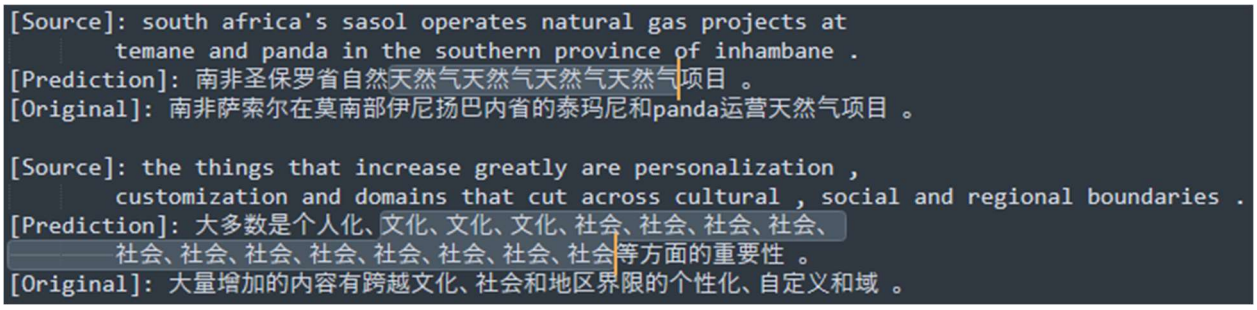

Fig 4. Some stuttering translation

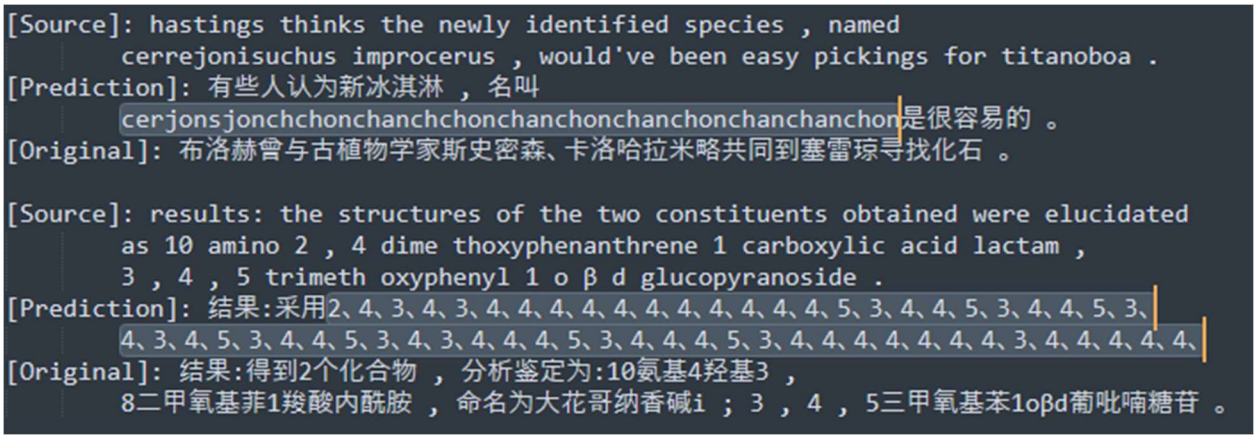

Fig 5. Scientific jargon translation

2. Big Bird model

At the time this report was written, the experiments on Big Bird model couldn't be properly trained. Currently, the training process could run for couple steps until running out of memory, and errors inherited from external libraries couldn't be resolved yet.

```
2022-11-28 21:31:49.910955: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 4 Chunks of size 108527616 totalling 414.00MiB
2022-11-28 21:31:49.910973: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 1 Chunks of size 110100480 totalling 105.00MiB
2022-11-28 21:31:49.910991: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 1 Chunks of size 130285568 totalling 124.25MiB
2022-11-28 21:31:49.911011: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 1 Chunks of size 138412032 totalling 132.00MiB
2022-11-28 21:31:49.911030: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 27 Chunks of size 150994944 totalling 3.80GiB
2022-11-28 21:31:49.911048: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 1 Chunks of size 158072832 totalling 150.75MiB
2022-11-28 21:31:49.911068: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 2 Chunks of size 177733632 totalling 339.00MiB
2022-11-28 21:31:49.911086: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 1 Chunks of size 220712960 totalling 210.49MiB
2022-11-28 21:31:49.911104: I tensorflow/tsl/framework/bfc_allocator.cc:1098] 6 Chunks of size 276824064 totalling 1.55GiB
2022-11-28 21:31:49.911123: I tensorflow/tsl/framework/bfc_allocator.cc:1102] Sum Total of in-use chunks: 15.48GiB
2022-11-28 21:31:49.911141: I tensorflow/tsl/framework/bfc_allocator.cc:1104] total_region_allocated_bytes_: 16635121664 memory_limit_: 16635121664 available bytes: 0 curr_region_allocatio
n_bytes_: 17179869184
2022-11-28 21:31:49.911162: I tensorflow/tsl/framework/bfc_allocator.cc:1110] Stats:
Limit:                  16635121664
InUse:                  16621086208
MaxInUse:               16621086208
NumAllocs:                     1967
MaxAllocSize:            276824064
Reserved:                        0
PeakReserved:                    0
LargestFreeBlock:                0

2022-11-28 21:31:49.911191: W tensorflow/tsl/framework/bfc_allocator.cc:492] ***************************************************************************************************
2022-11-28 21:31:49.911246: W tensorflow/core/framework/op_kernel.cc:1830] OP_REQUIRES failed at cwise_ops_common.h:123 : RESOURCE_EXHAUSTED: OOM when allocating tensor with shape[4,3072,3
072] and type float on /job:localhost/replica:0/task:0/device:CPU:0 by allocator mklcpu
INFO:tensorflow:training_loop marked as finished
I1128 21:31:57.411848 140567906538304 error_handling.py:115] training_loop marked as finished
WARNING:tensorflow:Reraising captured error
W1128 21:31:57.437516 140567906538304 error_handling.py:149] Reraising captured error
Traceback (most recent call last):
  File "/mnt/c/Users/yutan/Desktop/bb/bigbird/venv/lib/python3.9/site-packages/tensorflow/python/client/session.py", line 1378, in _do_call
    return fn(*args)
  File "/mnt/c/Users/yutan/Desktop/bb/bigbird/venv/lib/python3.9/site-packages/tensorflow/python/client/session.py", line 1361, in _run_fn
    return self._call_tf_sessionrun(options, feed_dict, fetch_list,
  File "/mnt/c/Users/yutan/Desktop/bb/bigbird/venv/lib/python3.9/site-packages/tensorflow/python/client/session.py", line 1454, in _call_tf_sessionrun
    return tf_session.TF_SessionRun_wrapper(self._session, options, feed_dict,
tensorflow.python.framework.errors_impl.ResourceExhaustedError: OOM when allocating tensor with shape[4,3072,3072] and type float on /job:localhost/replica:0/task:0/device:CPU:0 by allocat
or mklcpu
         [[{{node gradients/bert_1/encoder/layer_5/dense_1/Pow_grad/mul}}]]
Hint: If you want to see a list of allocated tensors when OOM happens, add report_tensor_allocations_upon_oom to RunOptions for current allocation info. This isn't available when running i
n Eager mode.


During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/mnt/c/Users/yutan/Desktop/bb/bigbird/run_translation.py", line 548, in <module>
    app.run(main)
  File "/mnt/c/Users/yutan/Desktop/bb/bigbird/venv/lib/python3.9/site-packages/absl/app.py", line 308, in run
```

Fig 6. Big Bird running out of memory

IV. Conclusion

Attention models provided a way to better address the gradient vanishing problem; despite the experiment results listed in section III wasn't long enough to claim that attention model indeed preserved the context information while LSTM models couldn't, it demonstrated being capable of retaining such information.

Another observation that was yet to be discovered on the prior LSTM model was that the Transformer model capable of learning phonetic information to translate names by the way it sounded. Nonetheless, as stated in section I, with the input length exceeding the attention window and uncertainty on Big Bird being faster than full attention models, a solution enabling long sequences to benefit from the power of attention model was wanted.

REFERENCES

[1] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 6, no. 02, pp. 107-116, 1998.

[2] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473,* 2014.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems,* vol. 30, pp. 11-21, 2017.

[4] D. Jacob, C. Ming-Wei, L. Kenton and T. Kristina, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

[5] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed and E. Koh, "Attention models in graphs: A survey," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 13, no. 6, pp. 1-25, 2019.

[6] S. Chaudhari, V. Mithal, G. Polatkan and R. Ramanath, "An attentive survey of attention models," *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 12, no. 5, pp. 1-32, 2021.

[7] Z. Manzil, G. Guru, D. Avinava, A. Joshua, A. Chris, O. Santiago, P. Philip, R. Anirudh, W. Qifan, Y. Li and A. Amr, "Big Bird: Transformers for Longer Sequences," *arXiv,* 2020.

[8] K. Clark, U. Khandelwal, O. Levy and C. D. Manning, "What Does BERT Look At? An Analysis of BERT's Attention," *arXiv,* 2019.

[9] L. Tian, D. F. Wong, L. S. Chao, P. Quaresma, F. Oliveira, Y. Lu, S. Li, Y. Wang and L. Wang, "UM-Corpus: A Large English-Chinese Parallel Corpus for Statistical Machine Translation," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014.

[10] T. Kudo and J. Richardson, *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing,* arXiv, 2018.