Paper

https://browse.arxiv.org/pdf/2205.14135.pdf

Idea

- Combines both Random attention, window attention, and global attention

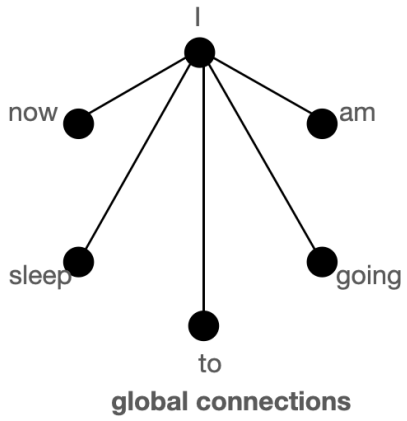
Architecture

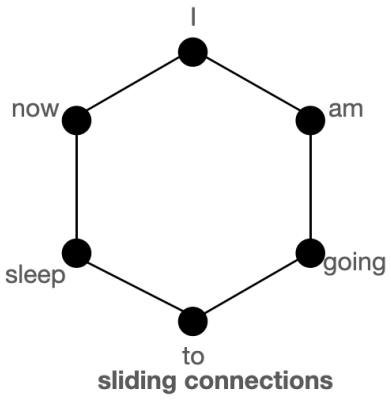
- On input sequence X = (x1 … x_n)
- And graph D whose vertex set is [n] = {1… n}

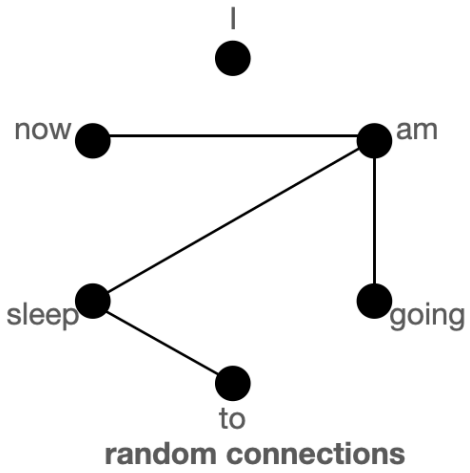
$$\text{ATTN}_D(\boldsymbol{X})_i = \boldsymbol{x}_i + \sum_{h=1}^{H} \sigma\left(Q_h(\boldsymbol{x}_i) K_h(\boldsymbol{X}_{N(i)})^T\right) \cdot V_h(\boldsymbol{X}_{N(i)}) \qquad \text{(AT)}$$

- This is just the standard attention equation
- Explains through graphs

Understanding the need for global, sliding, random keys with Graphs







Where is it applied

- Encoder side only

Proof

- Dhfksd

Sources

- https://www.youtube.com/watch?v=WVPE62Gk3EM&ab_channel=YannicKilcher
    - Shows attention
    - Explains star graph explanation
- https://huggingface.co/blog/big-bird
    - Explains graph

Reading Group Oct 8, 2023

- Why using the graphs?
  - Its easier to explain using graphs vs normal matrices]
  - “the quadratic complexity of self-attention can now be seen as a graph sparsification problem”
  - Erdos Renyi method
  - Can represent a graph through a Random walk of the graph?
  - Hugging face global atten graph diagram is wrong
- Is bigBird attention speed lower than original attention?
  - Not rly, its o(n)
- How is Big bird accuracy (knowing its a universal approximator) given equal sequence length?
  - It appears the sequence length varies in their experiments (as in model A has smaller seq length and they test it against BigBird with bigger context)
- Turing Completeness
  - As in is Big Bird a finite state machine and turing complete ?

B.2 Details of the Simulation

In this section, we give more details on the architecture of the encoder and decoder needed to implement our simulation strategy.

High Level Overview: Given the Turing machine M, we will show that a transformer with an appropriate encoder and decoder $\mathcal{T}_D$ can simulate each step of M's execution. Our simulation strategy will mostly follow Pérez et al. [72], except we will use a sparse attention mechanism. The main idea is to maintain the current Turing machine state $q^{(j)}$ and symbol under the head $s^{(j)}$ as part of the decoder sequence $\boldsymbol{Y}$ for all time step j so that we can always simulate the corresponding Turing machine transition $\delta(q^{(j)}, s^{(j)}) = (q^{(j)}, v^{(j)}, m^{(j)})$. The key difference will rise in Lemma B.4 of Pérez et al. [72], where full attention is used to select the appropriate symbol from tape history in one step. To accomplish the same task with sparse attention, we will exploit the associative property of max and break down the symbol selection over multiple steps. Thus, unlike Pérez et al. [72] one decoding step of our sparse transformer $\mathcal{T}_D$ does not correspond to one step of the Turing machine M. In particular, we will have two type of steps: compute step corresponding to update of M's state and intermediate steps corresponding to aggregating the max (which in turn is used for symbol selection). Let i denote the step of $\mathcal{T}_D$ and $g(i)$ denote the step of M being simulated at step i of the decoder. At each decoding step we want to maintain the current Turing machine state $q^{g(i)}$ and symbol under the $s^{g(i)}$ in $\boldsymbol{y}_i$. For roughly $O(\sqrt{i})$ intermediate steps the state will remain the same, while we aggregate information about relevant past output symbols through sparse attention. To maintain the same state for intermediate steps, we introduce an extra switching layer (App. B.2.3). Finally, at the next compute step we will make the transition to new state $q^{g(i)+1}$, new head movement $m^{g(i)}$, and new output symbol $v^{g(i)}$ to be written. Thereby we are able to completely simulate the

-