Center of Attention

Investigating the Effect of Alternate Attention Mechanisms on Efficiency and Model Utility

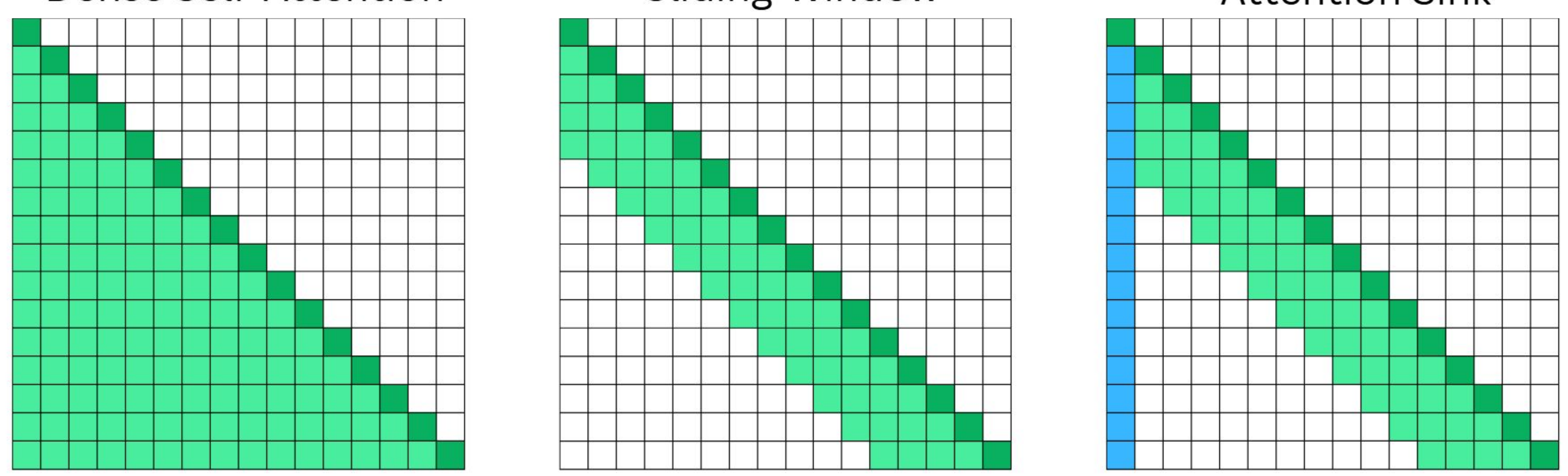
James Ding[1], Anisha Palaparthi[1], Hollie Zheng[1]

Stanford University

Project Overview

- LLMs and Transformer Models have increased in popularity
- Rely on self-attention to achieve state-of-the-art performance, but this operation creates a **bottleneck** in runtime due to **quadratic complexity**
- **(Dense) Self Attention**: Allows model to extract dependencies and relevant info from tokens through a **scaled-dot-product** calculation - runs in **O(n^2)!**
- **Accelerating Attention**: Three approaches to accelerating attention:
  - **Sparse Attention***: Reduce O(n^2) to polynomial complexity by using select token to compute attention
  - **Kernel Methods**: Replace softmax with a feature map approx. to reduce computational complexity
  - **Hardware Optimization***: Leverage hardware parallelism to reduce the runtime of dense self-attention computation
- Explore alternate attention mechanisms that reduce computational complexity (**Sliding Window Attention, Attention Sink**) or optimize hardware for attention computation (**FlashAttention**)
- Investigate the **tradeoffs** between **model performance** and **runtime** in 3 downstream tasks
- **FlashAttention** best mitigates these tradeoffs with **high performance** and **low runtime** compared to the baseline

Methods



- **Flash Attention**: Address **memory bottleneck** in long sequences by minimizing costly off-chip access through an **IO-aware** approach using **Tiling** and **Recomputation**. Relies on computation of additional statistics *m(x), l(x)*

$$m(x) = m([x^{(1)}\ x^{(2)}]) = \max(m(x^{(1)}), m(x^{(2)})), \quad f(x) = \left[e^{m(x^{(1)})-m(x)} f(x^{(1)}) \quad e^{m(x^{(2)})-m(x)} f(x^{(2)})\right],$$

$$\ell(x) = \ell([x^{(1)}\ x^{(2)}]) = e^{m(x^{(1)})-m(x)} \ell(x^{(1)}) + e^{m(x^{(2)})-m(x)} \ell(x^{(2)}), \quad \mathrm{softmax}(x) = \frac{f(x)}{\ell(x)}.$$

- **Sliding Window**: Divides K and Q matrices into **chunks.** Each chunk is multiplied with its corresponding chunk to produce a banded matrix.

$$\mathbf{K}_i = \mathbf{K}[:,:,iw:(i+2)w,:] \qquad \mathbf{Q}_i = \mathbf{Q}[:,:,iw:i(i+2)w,:] \qquad \forall i \in \left\{0,\dots,\left\lfloor \frac{s}{w} \right\rfloor - 1\right\} \quad \mathbf{A}_i = \mathbf{Q}_i\mathbf{K}_i^\top \qquad \forall i \in \left\{0,\dots,\left\lfloor \frac{s}{w} \right\rfloor - 1\right\}$$

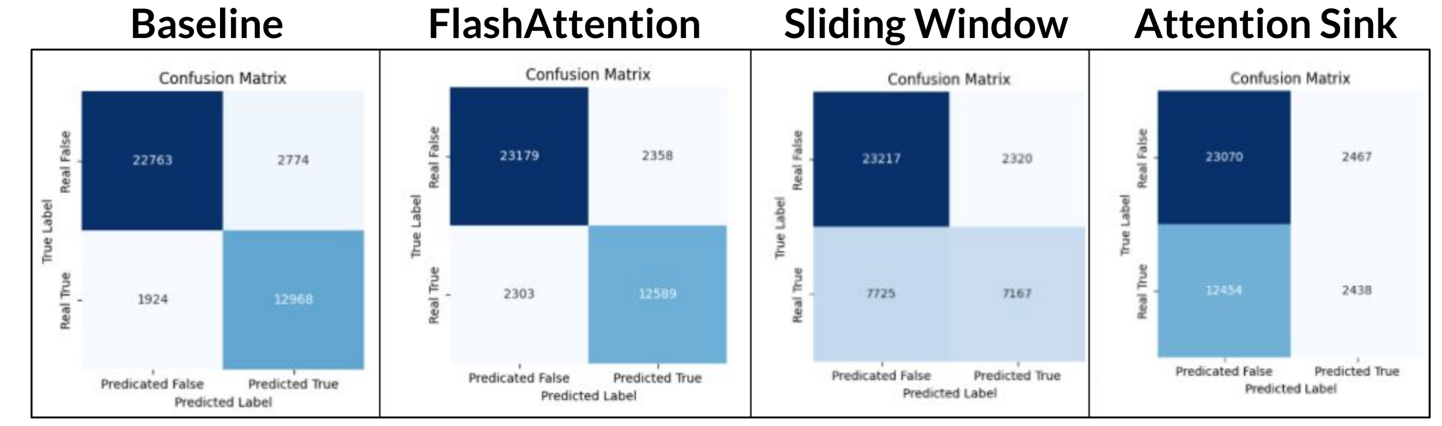
$$\mathbf{A}[:,:,iw:(i+2)w, iw:(i+2)w] = \mathbf{A}_i, \mathbf{A}[:,:,j,k] = 0 \text{ otherwise}$$

- **Attention Sink**: Similar to Sliding Window, but includes **first few tokens** for offloading attention (these have **high attention scores** when current token has nothing to attend to).

$$\mathbf{S}_i = \mathbf{Q}_0\mathbf{K}_i^\top \quad \mathbf{A}[:,:,iw:(i+2)w, iw:(i+2)w] = \mathbf{A}_i, \mathbf{A}[:,:,iw:(i+2)w, 0:2w] = \mathbf{S}_i$$

Discussion & Future Research

- **On Model Utility:** Flash Attention **matched** Baseline, while Sliding Window, Attention Sink saw **reduced performance** across downstream tasks.
  - Modifying attention mechanisms leads to **different points of failure** in each model. In Paraphrase Detection, Baseline and FlashAttention have balanced false positives/negatives, but **Attention Sink** was more prone to **false positives**. Likely because questions have similar beginning structures, and Attention Sink pays higher attention to earlier tokens.



- **On Model Runtime**: Flash Attention has the **greatest speedup**, while Sliding Window and Attention Sink saw **smaller speedups**.
  - Demonstrated that **reducing computation** complexity is **less effective** than reducing **IO bottleneck** at speeding up attention.
- **Tradeoffs:** Baseline maintains best performance but much slower. Sliding Window, Attention Sink provide speedup at **significant performance cost**.
  - FlashAttention **best mitigates tradeoff**, achieving higher speedups in training/evaluation time with minimal drop in model utility
- **Future Research**: investigate tradeoffs of alternate attention mechanisms (including kernel methods) and on other downstream tasks

Dataset and Metrics

- **Data**:
  - **Sentiment Analysis**: **SST** contains 12k single-sentence movie reviews labeled on a **five-point scale. CFIMDB** contains 2k multi-sentence reviews with **binary labels**.
  - **Paraphrase Detection**: 400k Quora question pairs with **binary labels.**
  - **Sonnet Generation**: Based on 154 **14-line sonnets** written by Shakespeare.
- **Performance**: Sentiment and Paraphrase is based on **accuracy**; Sonnet is based on **CHRF**.
  - Additional Metrics (for binary classification): F1 Score, Precision, Recall, and Confusion Matrices
- **Efficiency:** Training and Evaluation Time
- **Experiment Environment**: NVIDIA L4 GPU from GCP

Experiments