트랜스포머 모델의 활성화 희소성 활용 방안

박승현 [1], 박대진 [2]
[1] 경북대학교 전자전기공학부 석박통합과정
[2] 경북대학교 전자공학부 교수

ijjh0435@knu.ac.kr, boltanut@knu.ac.kr

Strategies for Leveraging Activation Sparsity in Transformer Architectures

Seung-Hyun Park[1], Dae-Jin Park[2]
[1]School of Electronic and Electrical Engineering, Kyungpook National University
[2]School of Electronics Engineering, Kyungpook National University

요 약

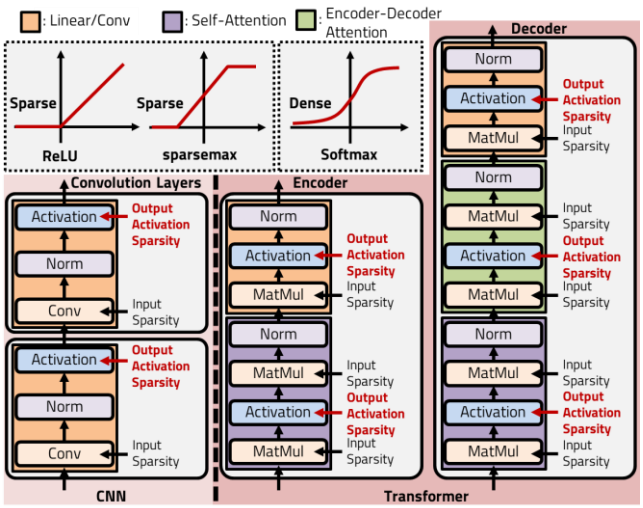
본 연구는 트랜스포머 모델의 어텐션 메커니즘에서 출력 희소성을 유도하기 위한 활성화 함수 대체 기법을 제안하고, 그 효과를 정량적으로 분석한다. 기존 softmax 기반 어텐션은 전 범위에 걸쳐 비선택적인 활성화를 발생시키며, 이는 연산 비효율성과 해석의 어려움을 초래한다. 이를 해결하기 위해 softmax 를 sparsemax 또는 ReLU 로 대체하고, 후자의 경우 bit-shift 기반 정규화를 적용해 하드웨어 효율을 높였다. 실험 결과, 두 대체 함수는 평균 80% 이상의 출력 희소성을 유도하였으며, 추론 속도, 메모리 사용량, 전력 소모 측면에서 유의미한 개선을 보였다. 성능 지표인 BLEU 점수는 softmax 대비 소폭 감소하였으나, sparsemax 의 경우 희소성과 정확도 간의 균형이 가장 우수하였다. 제안한 방식은 연산 자원이 제한된 환경에서도 트랜스포머 모델의 효율적 운용과 해석 가능성을 동시에 확보할 수 있는 실용적 대안을 제공한다.

1. 서론

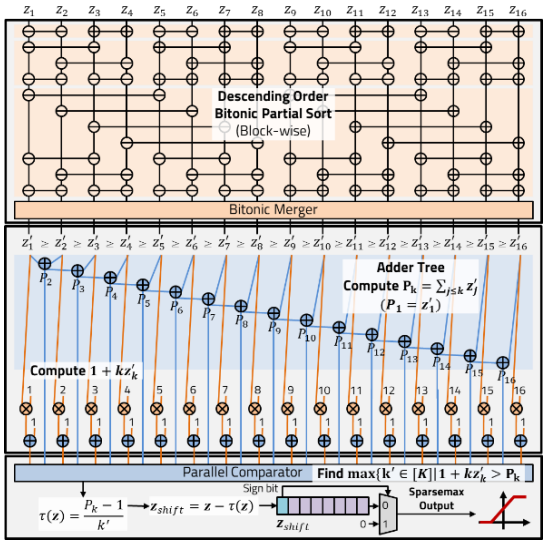
희소성은 정보 표현 및 처리 과정에서 일부 요소만이 활성화되거나 사용되는 현상을 의미하며, 자연계 및 생물학적 시스템에서도 널리 관찰된다. 예를 들어, 생물학적 뇌는 전체 뉴런 중 극히 일부만이 특정 자극에 반응함으로써 에너지 효율성과 정보 선택성을 동시에 달성한다 [1,2]. 이러한 특성은 인공신경망에도 영감을 주며, 최근에는 모델의 효율성과 해석 가능성을 높이는 데 중심적인 역할을 하고 있다 [3,4].

인공신경망에서 희소성은 일반적으로 그림 1 과 같이 세 가지로 나뉜다. 첫째, 입력 가중치 희소성은 학습 과정에서 중요도가 낮은 연결 가중치를 제거하거나 무시함으로써 모델의 파라미터 수를 줄이는 방식이다. 이는 주로 모델 압축 및 연산량 감소를 목적으로 하며, pruning 과 같은 기법을 통해 발생한다 [5]. 둘째, 입력 활성화 희소성은 한 층의 입력으로 들어오는 값들 중 다수가 0 이 되는 경우로, 보통 이전 층의 활성화 함수에 의해 유도된다. 이는 현재 층의 연산에서 곱셈이나 덧셈을 생략할 수 있는 기회를 제공한다 [6]. 셋째, 출력 활성화 희소성은 특정 층의 출력 값들 중 다수가 0 이 되는 현상을 의미한다. 이는 다음 층에서 입력으로 활용되는 값들에 영향을 주며, 후속 연산에서도 희소성을 유도할 수 있다 [7].

특히, 활성화 희소성은 입력에 따라 활성화되는 뉴



(그림 1) CNN 과 Transformer 모델에서 발생하는 희소성



(그림 2) Bitonic sort 를 활용한 sparsemax 가속 유닛 구조

런이 달라지는 입력 의존적 동적 희소성의 특성을 가지며, 이는 입력에 맞는 정보 처리 경로만을 선택적으로 활성화함으로써 연산 효율을 높이고, 불필요한 계산을 줄이며, 추론 속도를 개선하는 데 직접적으로 기여할 수 있다. 또한, 이러한 희소성은 하드웨어 친화적인 구조와 결합될 경우 메모리 전송 비용을 줄이고, 처리 지연을 감소시키는 효과도 가져온다 [8].

트랜스포머 모델의 MLP 계층이나 어텐션 블록 내부에서는 훈련이 진행될수록 점점 더 많은 희소성을 갖게 되며, 실제로 정보 전달에 기여하는 차원은 일부에 불과하다 [9]. 이러한 활성화 희소성은 단순히 연산량을 줄이는 데 그치지 않고, 모델의 일반화 성능, 노이즈에 대한 강건성, 추론 안정성 등 다양한 측면에서 긍정적인 효과를 가져올 수 있다. 뿐만 아니라, 일부 뉴런만이 의미 있는 출력을 생성하는 구조는 모델이 어떤 내부 표현을 통해 특정 예측을 수행하는지를 보다 명확하게 파악할 수 있는 기반을 제공하므로, 모델 해석 가능성을 향상시키는 데에도 유리하다 [10].

따라서, 활성화 희소성은 대규모 신경망의 비효율성을 줄이고 효율성과 성능을 동시에 향상시킬 수 있는 핵심적인 메커니즘이다. 나아가, 뉴런 수준에서의 정보 흐름을 구조적으로 제한함으로써 복잡한 딥러닝 모델의 작동 원리를 보다 뚜렷하게 분석할 수 있는기회를 제공한다는 점에서도 그 가치가 크다. 본 논문에서는 트랜스포머 모델에서 나타나는 활성화 희소성의 성질을 분석하고, 이를 실질적인 계산 최적화와 추론 효율화에 어떻게 활용할 수 있는지를 탐구하고자 한다.

2. 어텐션 레이어에서의 출력 활성화 희소성

트랜스포머의 self-attention 메커니즘에서 사용되는 softmax 함수는 입력 쿼리와 키 간의 유사도를 정규화하여 어텐션 스코어로 변환하는 데 사용된다. 그러나 softmax 는 본질적으로 모든 요소에 대해 양의 값을 출력하며, 희소성을 생성하지 못한다는 한계가 있다. 이를 해결하기 위해 본 연구에서는 어텐션 레이어에서 softmax 를 희소성이 있는 두 가지 함수, sparsemax 와 ReLU 로 대체하는 방법을 제안한다.

Sparsemax 는 softmax 의 대안으로, 출력 벡터의 일부 요소를 0 으로 만들 수 있는 선형적 정규화 함수이다 [11]. 이는 다음과 같은 과정으로 연산된다: 1) 입력 벡터 $z \in \mathbb{R}^n$을 내림차순으로 정렬한 후, 2) k 를 만족하는 최대 인덱스를 찾는다.

$$k = max\{j \in [n]: 1 + jz_{(j)} > \sum_{i=1}^{k} z_{(i)}\}$$

3) threshold τ를 계산한다.

$$\tau = \frac{1}{k}(\sum_{i=1}^{k} z_{(i)} - 1)$$

4) 최종 출력은 다음과 같다.

$$sparsemax(z)_i = \max(z_i - \tau, 0)$$

이 방식은 어텐션 분포에서 불필요한 요소를 0 으로 제거하여, 출력 희소성을 유도한다. 하지만 sorting 과 threshold 연산이 필요해 하드웨어에서는 구현 복잡도가 높을 수 있다. 그림 2 는 bitonic sorting 으로 구현한 sparsemax 연산 유닛 구조를 나타낸다. sparsemax 함수는 입력 벡터를 정렬하는 과정이 핵심인데, bitonic sort 는 하드웨어에서 병렬 처리가 용이하고 파이프라이닝 구조로 구현하기에 적합하다. 따라서 bitonic sort 를 사용하면 정렬 연산의 지연을 최소화하면서도 일정한 계산 구조를 유지할 수 있어 고속 sparsemax 구현에 적합하다.

또 다른 접근으로는 softmax 대신 ReLU 를 사용하는 방법이다 [12]. ReLU 는 단순한 연산으로 양수 요소만을 통과시키므로 희소한 분포를 유도할 수 있다. 그러나 정규화가 없어 입력 길이에 따라 분산이 커지고 학습이 불안정해질 수 있으므로, 이를 보정하기 위한 scaling 기법이 필요하다.

$$attention_{ij} = \frac{ReLU(q_i^T k_j)}{s}$$

여기서 s 는 입력 길이 n 에 따라 증가하는 분산을 줄이기 위한 scaling factor 로 다음과 같이 정의된다:

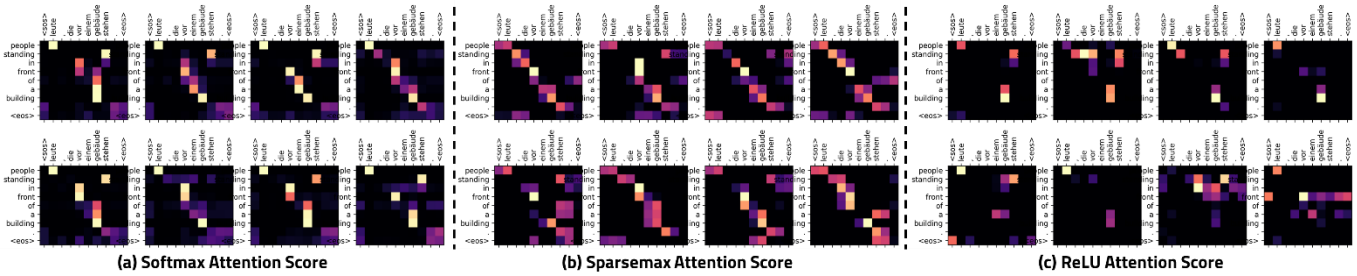
$$s = \gamma \cdot \sqrt{\frac{n}{2}}$$

이 때, γ는 학습 가능한 스케일 파라미터이다. 추가적으로, 하드웨어 상의 곱셈을 제거하기 위해 본 연구에서는 s 를 2 의 지수승으로 근사하고, bit-shift 연산으로 구현할 수 있도록 최적화하였다.

$$k = \lfloor \log_2(s) \rfloor, bit\_shift\_value = 2^{k+1}$$

따라서 어텐션 score 계산은 다음과 같이 구현할 수 있다.

$$attention_{ij} = \frac{ReLU(q_i^T k_j)}{2^{k+1}}$$



(그림 3) 다양한 활성화 함수의 어텐션 스코어 비교, 어두울수록 0 에 가까움

(표 1) 활성화 함수/레이어별 희소성 분석

Layer	Softmax	Sparsemax	ReLU
Encoder-Self	0.0000	0.7959	0.8034
Encoder-FF	0.8453	0.8518	0.8687
Decoder-Self	0.0000	0.8413	0.8511
Decoder-Cross	0.0000	0.7537	0.8534
Decoder-FF	0.8173	0.8448	0.8282

(표 2) 모델 정확도 비교 (BLUE Score)

Activation Function	BLEU Score
Softmax	34.67
Sparsemax	32.65
ReLU	31.14

이 방식은 연산 효율성과 메모리 접근 비용을 동시에 줄일 수 있으며, 희소성이 존재하는 경우 출력값 대부분이 0 이 되므로 활성화 희소성을 활용하여 후속 연산을 건너뛸 수 있는 장점을 가진다.

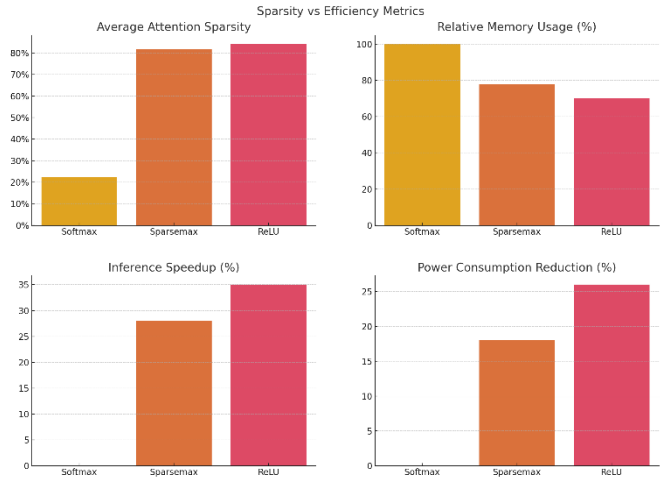
3. 어텐션 스코어를 통한 모델의 해석 가능성 비교

본 실험에 사용된 모델은 기본적인 Transformer 기반의 encoder-decoder 구조로, 입력 차원 7853, 출력 차원 5893, 은닉 차원 256 의 설정을 가지며, 인코더와 디코더는 각각 3 개의 층으로 구성되어 있다 [13]. 각 층에는 8 개의 어텐션 헤드와 피드포워드 차원 256, 드롭아웃 0.1 이 적용되었다. 이 구조는 어텐션 메커니즘의 변형이 실제 번역 성능과 어텐션 분포에 어떤 영향을 미치는지를 관찰하기에 적합하다.

트랜스포머 모델의 해석 가능성은 주로 어텐션 스코어 분포를 통해 평가할 수 있다. 어텐션 분포가 희소할수록 정보가 집중된 경로를 시각적으로 식별하기 쉬우며, 이는 모델이 어떤 입력 요소에 주목했는지를 명확히 보여줄 수 있다는 점에서 해석 가능성 측면에서 유리하다. 본 연구에서는 Softmax, Sparsemax, ReLU 를 어텐션 활성화 함수로 각각 적용하고, 이들 간의 어텐션 희소성, 해석 용이성, 성능(BLEU 점수)을 비교 분석하였다.

그림 3 은 동일한 입력에 대해 세 가지 어텐션 함수가 생성한 스코어 맵을 시각화한 것이다. Softmax 함수를 사용하였을 경우, 분포가 연속적이며 모든 위치에서 양의 가중치를 할당한다. 특정 영역에 집중은 있으나, 전체적으로 작은 양의 값이 분산된 형태를 띄므로 모델의 해석이 상대적으로 어렵다.

반면, Sparsemax 함수를 사용하였을 경우, 비교적 뚜렷한 집중을 보이며, 일부 요소는 완전히 0 으로 제거되어 희소성이 높다. 어텐션이 선택적으로 작동하는 패턴이 시각적으로 잘 드러난다. ReLU 를 사용하는



(그림 4) 추론시 희소성, 메모리, 성능, 전력 분석

경우에는 매우 높은 희소성을 나타내며, 많은 영역에서 어텐션 값이 0 에 수렴한다. 집중된 위치는 명확하나, 일부 경우에는 과도한 제거로 인해 맥락 정보가 소실될 가능성도 있다.

4. 어텐션 희소성 및 BLEU 점수 분석

표 1 은 softmax, sparsemax, ReLU 함수에 대해 트랜스포머 모델에서의 희소성을 보여준다. Softmax 는 전체 어텐션 스코어 분포가 비교적 연속적이며 0 이 아닌 값들을 많이 포함하므로 평균 희소성이 33% 수준으로 낮다. 반면, Sparsemax 와 ReLU 는 각각 81.8%, 84.1%에 이르는 높은 출력 희소성을 나타내며, 대부분의 어텐션 위치가 0 으로 비활성화된다. 이러한 희소성은 어텐션 맵에서 선택적 집중을 가능하게 하며, 정보 흐름이 시각적으로 명확하게 드러난다.

희소성이 높아질수록 표 2 와 같이 BLEU 점수는 일정 부분 감소하는 경향을 보인다. Softmax 는 BLEU 34.67 로 가장 높은 번역 성능을 보였고, Sparsemax 는 32.65, ReLU 는 31.14 를 기록했다. Sparsemax 는 희소성과 성능 사이의 균형이 가장 우수한 편이다. 희소성이 높아졌음에도 성능 저하는 제한적이었기 때문이다.

계층별 희소성 수치에서도 Sparsemax 와 ReLU 는 encoder-decoder 의 모든 self-attention, cross-attention 계층에서 약 75~85%의 희소성을 보였다. 반면 Softmax 는 encoder-self 와 decoder-encoder 계층에서 희소성을 보이지 않는다. Softmax 는 입력 간 모든 상호작용을 평균적으로 고려하는 반면, Sparsemax 와 ReLU 는 정보 흐름을 선택적으로 제한하는 방식이다.

5. 연산 성능 및 자원 사용 비교

희소성이 증가하면 계산량이 줄어든다. 특히 GPU 나 특화된 하드웨어에서는 0 연산을 건너뛰는 방식으로 추론 속도를 크게 향상시킬 수 있다. 실험 결과, 그림 4 와 같이 Sparsemax 는 약 28%, ReLU 는 약 35%의 추론 속도 향상을 보였다. Softmax 는 baseline 으로 비교를 위한 기준 역할만 수행한다.

메모리 사용량도 희소성과 비례해 감소한다. 중간 결과 값이 줄어들고, 저장 및 연산을 위한 공간이 절약된다. Sparsemax 는 약 22%, ReLU 는 약 30%의 메모리 절약 효과를 보였다. 시퀀스 길이가 길어질수록 이 효과는 더욱 커진다.

전력 소모 역시 감소한다. ReLU 는 가장 높은 희소성과 간결한 연산 구조 덕분에 약 26%의 전력 절감을 기록했고, Sparsemax 는 약 18%를 기록했다. Softmax 는 연산 경로가 밀집돼 있어 여전히 가장 많은 전력을 소비한다.

6. 결론

본 연구에서는 트랜스포머 모델에서 어텐션 활성화 함수를 softmax 에서 sparsemax 또는 ReLU 로 대체함으로써 출력 희소성을 유도하고, 이를 통해 해석 가능성과 연산 효율성을 동시에 향상시킬 수 있음을 보였다. 실험 결과, sparsemax 와 ReLU 는 softmax 에 비해 2 배 이상의 희소성을 유도하며, 어텐션 스코어 분포를 보다 선택적으로 만들었다. 이는 시각적으로 더 뚜렷한 정보 흐름을 제공하고, 중요한 입력 요소에 대한 집중이 명확히 드러나는 등 모델의 해석 가능성 측면에서 유의미한 개선으로 이어졌다. 성능 측면에서는 softmax 가 BLEU 기준 가장 높은 번역 정확도를 보였지만, sparsemax 와 ReLU 도 비교적 적은 성능 손실로 희소성을 확보할 수 있었다. 특히 sparsemax 는 희소성과 성능 사이의 균형이 우수해, 해석 가능성과 연산 최적화가 동시에 중요한 응용 환경에서 유리한 선택이 될 수 있다. 반면 ReLU 는 가장 높은 희소성과 전력 효율, 추론 속도를 제공하지만, 상대적으로 성능 저하가 더 큰 편이다. 희소성이 유도되면 메모리 사용량 감소, 연산량 절감, 전력 소모 감소라는 실질적 이점이 발생하며, 하드웨어 수준에서의 최적화 여지가 커진다. 본 연구에서는 ReLU 기반 스케일링 기법에 2 의 지수승 제약을 도입해 bit-shift 기반으로 효율적으로 구현할 수 있는 구조를 제안하였다. 이러한 접근은 경량 트랜스포머 모델 개발이나 실시간 추론 시스템에 적용 가능한 방향성을 제시하며, 향후 하드웨어 친화적인 희소 모델 설계에 기반이 될 수 있다.

사사문구

본 논문은 과학기술정보통신부의 재원으로 정보통신기획평가원 (No. 2022-0-01170, 50%, No. RS-2023-00228970, 50%)의 지원을 받아 수행된 연구임.

참고문헌

[1] K. Jason and G. David, “Imaging input and output of neocortical networks in vivo,” Proceedings of the National Academy of Sciences, vol. 102, no. 39, pp. 14063-14068, 2005.

[2] P. Cindy and I. Jeffry, “Odor representations in olfactory cortex: “sparse” coding, global inhibition, and oscillations,” Neuron, vol. 62, no. 6, pp. 850-861, 2009.

[3] T. Hastie, T. Robert and W. Martin, "Statistical learning with sparsity," Monographs on statistics and applied probability, vol. 143, no. 143, pp. 8, 2015.

[4] J. Sebastian et al., “Sparse is enough in scaling transformers,” Advances in Neural Information Processing Systems, vol. 34, pp. 9895-9907, 2021.

[5] S. Han, J. Pool, J. Tran, W. Dally, “Learning Both Weights and Connections for Efficient Neural Network,” Advances in Neural Information Processing Systems, 2015.

[6] E. Qin, A. Samajdar, H. J. Kwon, V. Nadella, S. Srinivasan, D. Das, B Kaul, T Krishna, “SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training,” IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 58-70, 2020.

[7] Z. Fan, W. Li, Z. Wang, T. Liu, H. Wu, Y. Liu, M. Wu, X. Wu, X. Ye, D. Fan, N. Sun, X. An, “Accelerating Convolutional Neural Networks by Exploiting the Sparsity of Output Activation,” IEEE Transactions on Parallel and Distributed Systems, Vol. 34, No. 12, pp. 3253-3265, 2023.

[8] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, Y. Chen, “Cambricon-X: An Accelerator for Sparse Neural Networks,” 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1-12, 2016.

[9] L. Zonglin et al., “The Lazy Neuron Phenomenon: On Emergence of Activation Sparsity in Transformers,” arXiv preprint, arXiv:2210.06313, 2022.

[10] L. James et al., “Training-Free Activation Sparsity in Large Language Models,” arXiv preprint, arXiv:2408.14690, 2024.

[11] A. Martins and R. Astudillo, “From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification,” International conference on machine learning, pp. 1614-1623, 2016.

[12] K. Shen et al., “A Study on ReLU and Softmax in Transformer,” arXiv:2302.06461, 2023.

[13] A. Vaswani et al., “Attention is All You Need,” Advances in neural information processing systems 30, 2017.