Wanda Pruning에 기반한 한국어 언어 모델 경량화

윤준호[1], 서대룡[2], 전동현[3], 강인호[4], 나승훈[*5]
전북대학교[1,*5], 네이버[2,3,4]
{hoho5702, nash}@jbnu.ac.kr, {daeryong.seo, donghyeon.jeon, once.ihkang}@navercorp.com

Wanda Pruning for Lightweighting Korean Language Model

Jun-Ho Yoon[1], Daeryong Seo[2], Donghyeon Jeon[3], Inho Kang[4], Seung-Hoon Na[*5]
Jeonbuk National University[1,*5], NAVER Corporation[2,3,4]

요 약

최근에 등장한 대규모 언어 모델은 다양한 언어 처리 작업에서 놀라운 성능을 발휘하고 있다. 그러나 이러한 모델의 크기와 복잡성 때문에 모델 경량화의 필요성이 대두되고 있다. Pruning은 이러한 경량화 전략 중 하나로, 모델의 가중치나 연결의 일부를 제거하여 크기를 줄이면서도 동시에 성능을 최적화하는 방법을 제시한다. 본 논문에서는 한국어 언어 모델인 Polyglot-Ko에 Wanda[1] 기법을 적용하여 Pruning 작업을 수행하였다. 그리고 이를 통해 가중치가 제거된 모델의 Perplexity, Zero-shot 성능, 그리고 Fine-tuning 후의 성능을 분석하였다. 실험 결과, Wanda-50%, 4:8 Sparsity 패턴, 2:4 Sparsity 패턴의 순서로 높은 성능을 나타냈으며, 특히 일부 조건에서는 기존의 Dense 모델보다 더 뛰어난 성능을 보였다. 이러한 결과는 오늘날 대규모 언어 모델 중심의 연구에서 Pruning 기법의 효과와 그 중요성을 재확인하는 계기가 되었다.

주제어: 언어 모델 경량화, Pruning

1. 서론

최근 몇 년 동안 인공 지능과 자연어 처리 분야는 놀라운 발전을 이룩하였다. 이 발전의 핵심에는 트랜스포머[2]를 기반으로 한 GPT, BLOOM, OPT와 같은 대규모 언어 모델이 자리잡고 있다. 이들 모델은 수십억에서 수조 개의 파라미터를 포함하며, 다양한 언어와 도메인에서 뛰어난 성능을 발휘한다. 이런 대규모 모델의 우수한 성능은 그 활용 가능성을 다양한 분야로 확장시켰다.[3]

그렇지만, 대규모 모델들의 등장은 동시에 여러 문제점도 함께 가져왔다.[4] 첫째, 모델의 크기 증가에 따라 학습과 추론에 필요한 컴퓨팅 자원이 기하급수적으로 증가하게 되었다. 이로 인해 모델 학습에 소요되는 시간과 비용이 크게 증가하였다. 둘째, 대규모 모델은 저장과 전송 모두에 큰 부담을 주게 되었다. 특히, 자원이 제한된 환경, 예를 들면 모바일 환경에서는 이런 대규모 모델의 활용이 매우 제한적이다. 다양한 태스크에서 우수한 성능을 보이는 GPT-175B 모델은 FP16으로 총 320GB 이상의 저장공간을 필요로 하며, 추론을 위해 최소 5개의 80GB의 메모리를 갖춘 A100 GPU가 필요하다.

이러한 대규모 언어 모델의 대중화를 목표로 높은 계산 비용을 완화하기 위한 다양한 연구들이 이루어지고 있다. 그 중 주목할만한 성과를 보이는 연구들은 모델의 파라미터를 더 낮은 Bit 수준으로 낮추는 Quantization에 집중되어 있다. 대규모 언어 모델 Quantization의 연구는 실제로도 상당한 리소스 절감으로 이어졌으며, 오늘날 QLoRA[5]와 같은 방법을 통해 모델을 4-8bit로 낮추어 고사양의 GPU를 보유하지 않은 환경에서도 수십억 파라미터 모델을 학습하고 추론할 수 있게 되었다.

Quantization 이외에도 모델을 압축하기 위한 노력으로는 Pruning 기법이 주목받고 있다. Pruning은 신경망 내에 중요도가 낮은 뉴런이나 연결을 제거하여 모델의 크기를 줄이는 방식이다. 이는 Lottery Ticket Hypothesis(LTH)[6]에서 제안된, 가중치가 잘 설정된 작은 서브넷을 사용하면 원래의 큰 신경망과 유사한 성능을 낼 수 있다는 아이디어와 유사하다. 이 기법은 모델의 크기를 줄이면서도 기존의 성능을 유지하거나, 때로는 Over-parameterized 된 모델을 경량화해 성능을 더욱 향상시키는 것을 목표로 한다. 이외에도, Pruned 모델의 일반화 능력을 향상시키기 위한 연구가 진행되고 있으며, 그 중요성이 점점 강조되고 있다.

본 논문에서는 한국어 언어 모델에 Pruning 기법을 적용하여 그 효과에 대해 깊게 탐구하고자 한다. 본 논문의 주요 기여는 다음과 같다.

- 한국어 모델인 Polyglot-Ko를 Pruning 후 한국어 위키 Perplexity 측정
- Pruned Polyglot-Ko의 한국어 자연어 이해 task에 대한 Zero-shot 평가
- Pruned Polyglot-Ko를 Fine-tuning 후 성능 비교
- Pruned Polyglot-Ko를 KoAlpaca v1.1으로 학습 후 성능 비교

2. 관련 연구

2.1 모델 경량화

모델 압축은 딥러닝 모델의 매개변수 크기를 줄이고 지연 시간을 줄이는 것으로, 계산 비용을 줄이고 메모리 사용량을 최적화하는 데 중요한 역할을 하며 그 필요성이 크게 증가하고 있다. 이는 자원이 제한된 환경에서 유용하며, 더 빠른 추론 시간과 적은 에너지 소비로 이어진다. 모델을 압축하기 위한 선행 연구들은 Quantization, Knowledge Distillation, Pruning과 같은 기타 기법 등 여러 카테고리로 나눌 수 있다.

Quantization[5][7][8]은 파라미터 값을 제한된 수의 가능한 값으로 매핑하여 모델 크기를 줄이는 것으로 기존 모델의 부동 소수점 수를 줄이는 데 그 목적이 있다. 예를 들어, 32비트 부동 소수점을 8비트 정수로 변환하는 것이다. 이를 통해 압축된 모델은 기존의 모델보다 메모리 사용량이 훨씬 적으며, 부동 소수점 연산보다 더욱 빠른 연산을 수행하여 계산 속도가 향상된다는 이점이 있다. 다만 기존의 파라미터 값을 다른 범위로 매핑하면서 정보가 손실되므로 성능 저하가 발생할 수 있다.

Knowledge Distillation[9][10]은 파라미터 수가 큰 Teacher 모델에서 작은 Student 모델로 정보를 전달하는 방법인데, Student 모델이 성능 저하 없이 Teacher 모델의 성능을 달성할 수 있도록 한다. Teacher 모델의 출력(일반적으로 특정 레이블에 대한 softmax 함수에 의해 생성된 확률)을 분포 형태로 변형하여 학습 시에 Teacher 모델의 손실과 Student 모델의 손실을 동시에 반영하는 형태이다. 작은 Student 모델도 복잡한 Teacher 모델의 출력을 잘 근사시킬 수 있으므로 기존의 큰 모델의 성능을 보일 수 있다는 장점이 있으나, 두 모델 간의 지식 증류 과정에 추가적인 학습 시간 및 리소스를 필요로 한다.

Pruning[1][4][11]은 신경망에서 중요하지 않다고 판단되는 가중치를 제거하여 신경망의 복잡성과 크기를 줄인다. Pruning에는 다양한 전략이 있지만, 대표적으로는 가중치의 크기가 작은 연결을 제거하는 Magnitude-based Pruning[12][13]이 있다. 이론적으로 Pruning을 통해 불필요한 가중치를 제거하여 모델의 크기를 줄일 수 있으며, 가중치의 제거로 인해 초당 부동소수점 연산 횟수가 줄어 추론 시간이 단축된다. 다만 너무 많은 가중치를 제거하면 모델의 성능에 영향을 미칠 수 있어 적절한 비율을 찾는 것이 중요하며, 모델 별 그 비율이 상이하다는 단점이 있다.

각 모델 압축 기법은 고유한 장점과 단점을 가지고 있으며, 특정 상황과 태스크의 요구 사항에 따라 적합한 기법을 선택해야 한다.

2.2 Pruning

Pruning에는 다양한 전략들이 있으며, 주요 기법으로는 Unstructured Pruning[12], Structured Pruning[14]이 있다. Unstructured Pruning은 개별적인 가중치를 독립적으로 제거하는 방식으로, 대표적으로는 가중치의 절대값을 기준으로 작은 값을 갖는 가중치를 제거하는 Magnitude-based Pruning[12][13]이 있다. Structured Pruning은 개별 가중치가 아닌 모델 내의 채널이나 필터 등의 구조를 제거하는 것으로 정형화된 네트워크 구조를 유지할 수 있다. 두 기법은 해석하는 시각에 따라 다르지만 일반적으로 반대되는 장, 단점을 갖는다. Unstructured Pruning의 경우 파라미터 수준에서의 최적화를 수행하기에 높은 압축률을 적용하여도 좋은 성능을 보여준다. 그러나 sparse 행렬 연산을 위한 불규칙적인 메모리 접근을 가능하게 하는 전용 하드웨어나 라이브러리 없이는 실질적인 추론 속도 개선을 이뤄내지 못한다. 반대로 Structured Pruning의 경우 제거한 구조에 대해서는 행렬 연산을 하지 않아도 되므로 전용 하드웨어나 라이브러리 없이도 Pytorch와 같은 딥러닝 프레임워크와 잘 호환되어 실질적인 추론 속도를 개선할 수 있다는 장점이 있으나, 세밀한 최적화의 어려움으로 높은 압축률을 달성하기 어렵다.

2.3 2:4 Sparsity

2:4 Sparsity[15]는 행렬 내의 4개의 연속된 값에 대해 최소 2개는 0으로 만들어 50%의 희소성을 부여하는 기법으로 모델의 정확도 유지와 하드웨어 가속 달성 간의 균형을 해결하기 위한 일종의 Pruning 기법이다. 오늘날 잘 다듬어진 Unstructured Pruning의 경우 정확도는 높으나, 열악한 메모리 접근으로 현대 벡터와 행렬 연산의 장점을 제대로 활용하지 못한다. 2:4 Sparsity 패턴의 구조화된 특성은 Sparse 행렬 연산에 최적화된 아키텍처에서 효율적인 메모리 엑세스가 가능하도록 한다. 다른 Sparsity 패턴에 비해 압축된 형식으로 저장하는데 있어 메모리 오버헤드가 적으며 행렬 곱셈 연산에 대한 연산 처리량을 두 배로 늘릴 수 있다. NIVIDA Ampere GPU 아키텍처는 Sparse Tensor Core를 이용해 2:4 Sparsity 패턴을 갖는 행렬에 대한 연산을 가속화한다.

3. 한국어 언어 모델 Pruning

3.1 Pruning 기법: Wanda

본 논문에서는 Wanda[1]를 Pruning 기법으로 채택하였는데, 그 이유는 다음과 같다. 일반적인 Pruning 방식들은 Pruning 후의 성능을 복원하기 위해 대규모 재학습 과정을 필요로 한다. 이는 대규모 모델에서는 상당한 비용을 수반하기에 현실적으로 GPT와 같은 규모의 모델에 적용하기에 어려운 점이 있다. 이후 SparseGPT[11]처럼 재학습 없이 대규모 언어 모델을 Pruning 할 수 있는 방법도 소개되었지만, SparseGPT[11]도 Pruning 때문에 발생하는 성능 저하를 보완하기 위해 남은 가중치를 갱신하는데 많은 계산 비용이 필요하다.

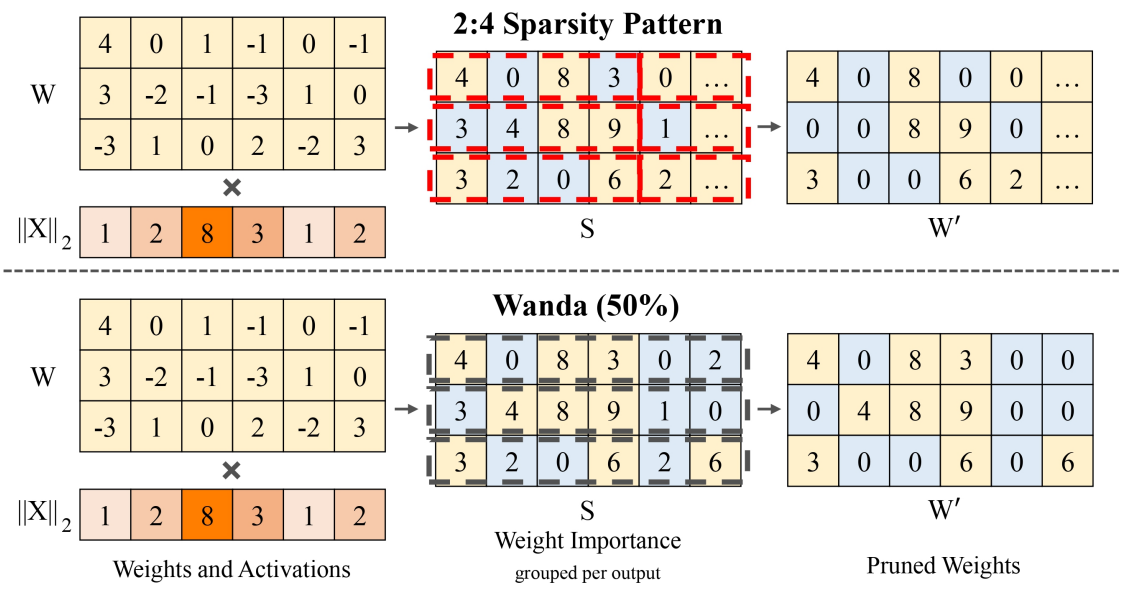

그림 1. Wanda 기법의 일반 Pruning(50%)과 2:4 Sparsity 패턴 Pruning 그림. 가중치 행렬 **W**와 입력 특징 활성화 **X**가 주어질 때, 가중치의 크기와 입력 활성화의 l_2 norm 값을 요소별 곱하여 가중치 중요도를 계산. 가중치 중요도를 바탕으로 가중치 제거시, 전역적인 비교가 아닌 출력별(가중치 내의 각 행)로 비교한다.

Wanda[1]는 대규모 재학습 및 복잡한 가중치 갱신없이 기본적인 Magnitude-based Pruning을 입력 활성화 값을 고려하는 방향으로 확장하여, 가중치 행렬의 중요도를 구한 후 일정 비율만큼 제거한다. Wanda[1]는 단일 포워드 패스 내에서의 최적화를 통해 Magnitude-based Pruning과 같은 단순성을 유지하면서 역전파나 헤시안 행렬을 통한 그래디언트 계산 그리고 제거된 가중치들을 보상하기 위한 가중치 업데이트 없이 사용할 수 있는 기법이다. 그럼에도 Wanda[1]는 Magnitude-based Pruning의 성능을 크게 능가하며 기존의 우수한 Pruning 기법인 SparseGPT[11]와 동등하거나 대부분의 경우 더 나은 성능을 발휘한다.

$$\mathbf{S}_{ij} = |\mathbf{W}_{ij}| \cdot ||\mathbf{X}_j||_2 \quad (1)$$

Wanda[1]는 모델의 가중치 행렬 **W**와 입력 **X**에 대해서 **W** 내의 각 가중치의 중요도 **S**를 (1)과 같은 수식을 통해 정의한다.

3.2 모델 및 Pruning 최적화

본 논문에서는 Pruning 기법을 적용하기 위한 대상으로 한국어 언어 모델인 Polyglot-Ko를 선택하였다. 이 Polyglot-Ko 모델은 EleutherAI의 Polyglot 프로젝트[1]의 일부로, TuNiB[2]에서 수집된 1.2TB의 한국어 데이터를 기반으로 Polyglot을 특화하여 한국어로 모델링한 결과물이다. 이 모델은 다양한 다국어 모델들과의 비교에서 뛰어난 처리 능력을 보여주며, 현대의 한국어 자연어 처리 연구에서 복잡하고 다양한 형태소 구조를 가진 한국어 문장의 이해와 생성에 큰 기여를 하고 있다.

Wanda[1] 기법에서는 가중치 제거를 위해 가중치의 중요도를 계산할 때 입력 활성화 값을 포함하기에, 이를 위해서는 입력으로 사용되는 보정 데이터가 필요하다. 보정 데이터는 특정 도메인이나 태스크에 국한되지 않는, 일반적인 목적을 지닌 데이터를 활용한다. 예를 들어, 영어 언어 모델에서는 대채로 C4 데이터셋이 이러한 목적으로 활용된다. 본 논문에서는 Polyglot-Ko 모델의 Pruning을 위해 C4 데이터셋의 다국어 버전인 mC4 데이터셋[3] 중 한국어 부분을 보정 데이터로 선택하여, Huggingface 플랫폼에서 해당 데이터를 가져와 활용하였다.

Wanda[1]는 연속된 가중치들을 대상으로 가중치 메트릭을 사용하여 2:4 Sparsity 패턴으로 변환할 수 있게 설계되었다. 본 논문에서는 향후 NVIDIA의 Ampere GPU 아키텍처에 기반한 추론 속도 향상을 목표로, 기본적인 Wanda[1] Pruning 방식뿐만 아니라, 2:4 Sparsity 패턴과 이를 연속된 8개의 가중치 값들 중에서 4개를 제거하는 방식으로 확장한 4:8 Sparsity 패턴을 모두 적용하였다. 모든 방식에서 동일한 비율로 가중치를 제거하기 위해, Wanda Pruning에는 50% 비율을 적용하였다.

[1]https://github.com/EleutherAI/polyglot
[2]https://tunib.ai/
[3]https://huggingface.co/datasets/mc4

표 1. Polyglot-Ko 모델들의 zero-shot task F1 결과

Params	Method	Sparsity	COPA	HellaSwag	BoolQ	SentiNeg	Mean
5.8B	Dense	0%	77.45	48.53	43.56	33.94	50.87
	Wanda	50%	**74.46**	**44.2**	**35.59**	38.38	48.15
		4:8	71.15	40	35.03	51.54	49.43
		2:4	66.54	35.27	33.56	**67.7**	**50.76**
12.8B	Dense	0%	79.37	48.28	48.18	91.17	66.75
	Wanda	50%	**76.27**	**60.2**	**49.23**	**77.49**	**65.79**
		4:8	74.67	45.72	39.42	73.4	58.30
		2:4	71.95	41.93	34.65	72.22	55.18

표 2. Polyglot-Ko 모델들의 한국어 위키 Perplexity

		Polyglot-Ko	
Method	Sparsity	5.8B	12.8B
Dense	0%	11.39	11.18
Wanda	50%	**15.39**	**13.00**
	4:8	23.12	15.10
	2:4	50.52	20.24

4. 실험 및 결과

4.1 Pruned 모델의 Perplexity

Polyglot-Ko 모델에 Pruning을 적용한 후, 5.8B 모델과 12.8B 모델을 한국어 위키의 Perplexity(PPL)를 기준으로 성능 평가를 진행하였다. 한국어 위키 데이터는 Korpora 프로젝트[4]에서 가져온 텍스트를 사용하였다.

결과는 표 2에 나타나 있으며, 전체적으로 2:4와 4:8 Sparsity 패턴에서 PPL이 증가하였다. 그러나 Wanda-50%의 경우 Dense 모델과 유사한 PPL을 보여주었고, 성능 순서는 Wanda-50%, 4:8 Sparsity 패턴, 2:4 Sparsity 패턴 순이었다. 특히 2:4 Sparsity 패턴의 5.8B 모델에서는 PPL이 급증하는 모습을 보였지만, 12.8B 모델에서는 상승폭이 비교적 낮았다. 이를 통해 동일 모델에 비해 파라미터 수가 많을수록 Pruning 이후에도 안정적인 성능을 보이는 것을 확인할 수 있었다.

4.2 Pruned 모델의 Zero-shot

Polyglot-Ko 모델에 Pruning을 적용한 후, 5.8B 모델과 12.8B 모델에서 Zero-shot 성능을 평가하였다. 평가에는 SKT에서 공개한 한국어 자연어 이해 벤치마크인 KoBEST[16] 중 COPA, HellaSwag, BoolQ, SentiNeg를 이용하였다.

결과는 표 1에 나타나 있으며, 대부분의 태스크에서는 표 2과 유사한 순서로 Wanda-50%, 4:8 Sparsity 패턴, 2:4 Sparsity 패턴이 좋은 성능을 보였다. 특히 5.8B 모델의 SentiNeg 결과에서는 Pruning을 적용한 모델들이 기존의 Dense 모델보다 전반적으로 더 우수한 성능을 보였으며, 2:4 Sparsity 패턴에서는 성능이 거의 2배 가까이 높아진 것을 확인할 수 있었다. 이는 SentiNeg 데이터셋이 가중치가 제거된 모델에 더 적합하게 작용했거나, Pruning을 통해 불필요한 가중치들을 효과적으로 제거하여 중요한 특성만을 활용한 결과라고 해석할 수 있다. 모델의 파라미터 수와 관련하여, 표 2과 유사하게 12.8B 모델은 동일한 모델 기준으로 파라미터 수가 더 많아 5.8B 모델보다 전반적으로 안정적인 모습을 보여준다.

4.3 Pruned 모델의 Fine-tuning

Polyglot-Ko 모델에 Pruning을 적용한 후, 그 결과로 저하된 언어 모델의 능력을 Fine-tuning을 통해 회복할 수 있을지 알아보기 위해 KorQuAD v1.0[5]을 기반으로 한국어 MRC 모델로써의 능력을 평가해보고자 한다.

자원의 한계로 인해 Polyglot-Ko 5.8B 모델과 12.8B 모델을 모두 Full Fine-tuning 하는 것은 어려웠기에 5.8B 모델에 대해서만 LoRA를 활용하여 Parameter-Efficient Fine-Tuning(PEFT)을 진행하였다. 이때 LoRA의 하이퍼파라미터 설정은 Alpaca-LoRA[6]를 참고하였고, 모델의 Self-attention 및 MLP 모듈에 Adapter를 추가하였다. 전체 5.8B 파라미터 중 약 14M개, 즉 전체의 0.25% 파라미터만이 학습에 사용되었다.

입력 데이터의 길이 설정은 쿼리를 96, 전체 시퀀스를 512로 하였고, 쿼리와 문서 사이에 <|SEP|>토큰을 추가하고 시퀀스 마지막에 <|EOS|>토큰을 추가하여 시퀀스를 구성하였다.

[4]https://github.com/ko-nlp/Korpora

[5]https://korquad.github.io/KorQuad%201.0/

[6]https://github.com/tloen/alpaca-lora

표 3. Koalpaca 데이터셋으로 fine-tuning 한 Polyglot-Ko 5.8B 모델에 대한 few-shot NSMC 벤치마크 성능 비교

Model	Peft	Method	Sparsity	Prompt1	Prompt2
Polyglot-Ko	✗	Dense	0%	57.52	72.23
KoAlpaca-Polyglot	✗	Dense	0%	69.39	76.83
KoAlpaca-Polyglot(Ours)	LoRA	Dense	0%	82.4	78
		Wanda	50%	77.2	73.8
			4:8	**78.8**	**77.4**
			2:4	72.8	68.6

표 4. Polyglot-Ko 5.8B 모델의 KorQuad v1.0 F1 결과

Method	Sparsity	KorQuad
Dense	0%	67.15
Wanda	50%	**66.83**
	4:8	66.03
	2:4	65.85

MRC의 전처리 및 후처리는 EnlipleAI의 KorQuAD 챌린지[7]의 소스코드를 Polyglot-Ko 모델에 맞게 수정하여 적용하였고, 성능 평가는 SQuAD v1.1 기반의 KorQuAD v1.0 공식 평가 스크립트사용하여 F1 점수로 비교하였다. 이 F1 점수는 모델의 응답과 정답을 음절 단위로 비교해서 정답과 겹치는 부분을 고려한 점수이다.

평가 결과는 표 4에 나타나 있으며, 이전의 결과와 유사하게 Wanda-50%, 4:8 Sparsity 패턴, 2:4 Sparsity 패턴 순으로의 성능이 나타났다. 이전 PPL과 Zero-shot 측정과는 달리, LoRA 파라미터 학습으로 인해 큰 성능 저하는 발생하지 않았음을 확인하였다.

4.4 Pruned 모델의 Instruction-tuning

Polyglot-Ko 모델에 Pruning을 적용한 뒤, Instruction-tuning을 수행하였을 때, 기존 모델에 해 얼마나 성능이 변화하는지를 비교하고자 한다.

사용된 데이터셋은 KoAlpaca[8]이며, 섹션 4.3에서 언급한 것처럼 자원의 한계로 인해 5.8B 모델만을 대상으로 LoRA를 활용하여 PEFT를 진행하였다. LoRA의 하이퍼파라미터 설정은 섹션 4.3과 동일하게 설정하였고, 5.8B 파라미터 중 약 14M개, 즉 전체의 0.25% 파라미터만이 학습에 사용되었다.

성능 평가는 NSMC 벤치마크를 사용하였고, KoAlpaca의 github[9]에서 제공하는 Few-shot 설정 및 2가지 프롬프트를 기반으로 하였다. 그 결과는 표 3에 나와 있으며, 4:8 Sparsity 패턴에서 가장 Dense 모델의 성능을 유지하는 모습을 보였고, PEFT가 적용되지 않은 Full Fine-tuning 결과보다 좋은 성능을 보였다.

5. 결론

본 논문에서는 한국어 언어 모델인 Polyglot-Ko를 Wanda[1] 기법을 적용하여 Pruning을 진행하였고 그 성능을 평가하였다. 대체로 Wanda-50%, 4:8 Sparsity 패턴, 2:4 Sparsity 패턴의 순서로 우수한 성능을 보였으나, 일부 경우에서는 4:8 Sparsity 패턴이나 2:4 Sparsity 패턴이 더 우수했고, Dense 모델을 능가하는 모습을 보이기도 하였다.

한계점으로는 Wanda[1] 기법은 보정 데이터의 질에 의존적인 면이 있는데, 본 논문에서는 mC4 데이터셋의 한국어 부분만을 이용하였지만, 다양한 보정 데이터셋을 활용하여 Wanda[1] 기법의 한국어 Pruning 성능을 평가하는 연구가 필요해 보인다. 이를 통해 보정 데이터의 질과 모델 성능 간의 관계를 더 깊이 이해할 수 있을 것이다. 또한 최적의 하이퍼파라미터에 대한 깊은 탐색의 부재로, 모델의 최적 성능을 확보하기 위한 추가적인 연구가 필요함을 시사한다.

향후 연구에서는 다양한 보정 데이터셋의 활용과 함께 하이퍼파라미터 최적화를 진행하여 더욱 세밀하고 정교한 모델 최적화 방법을 탐구하고, 2:4 Sparsity 패턴과 4:8 Sparsity 패턴을 하드웨어적으로 최적화하여 실질적인 학습 및 추론 속도를 향상시켜 속도 대비 성능의 추이를 확인해 볼 수 있을 것이다.

참고문헌

[1] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, “A simple and effective pruning approach for large language models,” 2023.

[7] https://github.com/enlipleai

[8] https://huggingface.co/datasets/beomi/KoAlpaca-v1.1a

[9] https://github.com/Beomi/KoAlpaca

[2] 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, 2017.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[4] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," 2023.

[5] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.

[6] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[7] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," 2023.

[8] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," 2023.

[9] Y. Gu, L. Dong, F. Wei, and M. Huang, "Knowledge distillation of large language models," 2023.

[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[11] E. Frantar and D. Alistarh, "Sparsegpt: Massive language models can be accurately pruned in one-shot," 2023.

[12] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," 2015.

[13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016.

[14] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2017.

[15] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu, and P. Micikevicius, "Accelerating sparse deep neural networks," 2021.

[16] D. Kim, M. Jang, D. S. Kwon, and E. Davis, "Kobest: Korean balanced evaluation of significant tasks," 2022.