SVD에 기반한 모델 경량화를 통한 문서 그라운딩된 응답 생성

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

Lightweight Language Models based on SVD for Document-Grounded Response Generation

Geom Lee[○1], Dea-ryong Seo[2], Dong-Hyeon Jeon[3], In-ho Kang[4], Seung-Hoon Na[*5]
Jeonbuk National University[○1,*5], NAVER Corporation[2,3,4]

요 약

문서 기반 대화 시스템은 크게 질문으로부터 문서를 검색하는 과정과 응답 텍스트를 생성하는 과정으로 나뉜다. 이러한 대화 시스템의 응답 생성 과정에 디코더 기반 LLM을 사용하기 위해서 사전 학습된 LLM을 미세 조정한다면 많은 메모리, 연산 자원이 소모된다. 본 연구에서는 SVD에 기반한 LLM의 경량화를 시도한다. 사전 학습된 polyglot-ko 모델의 행렬을 SVD로 분해한 뒤, full-fine-tuning 해보고, LoRA를 붙여서 미세 조정 해본 뒤, 원본 모델을 미세 조정한 것과 점수를 비교하고, 정성평가를 수행하여 경량화된 모델의 응답 생성 성능을 평가한다. 문서 기반 대화를 위한 한국어 대화 데이터셋인 KoDoc2Dial에 대하여 평가한다.

주제어: 문서 기반 대화 시스템, 응답 생성, 특이값 분해, 저차원 분해

1. 서론

텍스트 생성을 위해 사용되는 디코더 모델은 큰 관심을 받고 있다. 모델의 파라미터 수가 많아질수록 성능이 좋아져서 시간이 지날수록 더 큰 모델들이 출시되고 있다. 다양한 챗봇 서비스들에 적용하기 위해 이러한 큰 LLM들이 연구되고 있으며, 문서 기반 대화 시스템은 이런 서비스들을 구현하는 데 있어 필수가 되는 연구 분야이다.

문서 기반 대화 시스템은 크게 두 단계로 사용자의 질문과 대화 기록으로부터 적절한 문서를 찾아내는 문서 검색 과정, 질문과 대화 기록, 찾아낸 문서로부터 알맞은 응답을 생성하는 응답 생성 과정으로 나뉠 수 있다. 응답 생성 과정에 사용할 LLM은 문서 - 대화 기록 - 질문을 입력받아 정답을 생성해 내는 과정에 대한 학습이 필요하다. 이런 응답 생성 과정에 매우 많은 지식을 학습한, 성능이 좋은 LLM들을 사용할 수 있으면 좋겠으나, 그러한 모델들은 파라미터 수가 너무 많아서 GPU 간 분산을 사용해야 하는 등의 문제가 존재할 수밖에 없다. 그렇다고 하여 최근에 비교적 작은 크기의 LLM인데도 성능이 뛰어나다고 하여 널리 퍼진 LLaMA, Polyglot-ko 등의 모델들을 사용하기에도 다음과 같은 문제들이 존재한다.

성능이 좋은 LLM들을 사용하기 위해서는 VRAM이 큰 GPU가 필요할 뿐만 아니라, LLM을 미세 조정하는 과정에도 굉장히 많은 메모리, 연산 자원을 사용한다. 다양한 방법을 통해 큰 모델을 미세 조정하는 것과 동일한 결과를 더 효율적으로 얻을 수 있도록 하는 방법들에 대한 연구가 지속되고 있다.

본 연구에서는 모델 파라미터 수를 줄이기 위해 다음과 같은 과정을 진행한다. 특이값 분해(Singular Value Decomposition, SVD)를 사용하여 사전 학습된 LLM의 각 가중치 행렬을 low rank 행렬로 분해한다. 사전 학습된 LLM과 분해된 LLM 각각을 대화를 위한 응답 생성 task에 대해 미세 조정 하여 분해 전후 성능을 비교해본다. 또한, 학습 과정에서의 효율성 증가 및 low rank 행렬이 표현할 수 있는 범위 이상의 정보를 학습할 수 있을지를 알아보기 위해 LoRA를 추가하여 성능을 비교한다.

본 논문의 기여는 다음과 같다.

- 경량화를 위한 특이값 분해 적용.
- low rank 구조의 모델에 LoRA 적용.
- 특이값 분해가 응답 생성을 위한 LLM 미세 조정 결과에 미치는 영향 확인.

2. 관련 연구

2.1 거대 언어 모델

언어 모델은 파라미터가 많을수록 성능이 높아지고, 또 더욱 복잡하고 정교한 기능을 수행할 수 있다고 여겨진다. 대규모 언어모델인 GPT3[1]는 1,750억 개의 파라미터를 가지는 언어모델로, 무작위 글짓기, 간단한 수준의 사칙연산, 번역, 주어진 문장에 따른 간단한 웹 코딩, 간단한 대화 등의 다양한 작업이 가능하다. 이 GPT3에서 약간 변경된 GPT3.5는 현재 chatGPT의 무료 버전에 포함되어 서비스되고 있다. PALM[2]과 같이 5,300억 개의 파라미터를 가진 더 큰 모델들도 존재한다.

2.2 모델 경량화

언어 모델들의 파라미터 수가 커짐에 따라서 GPU 메모리가 부족해지는 것, 학습에 걸리는 시간이 증가하는 것, 추론에 걸리는 시간이 증가하는 것들은 피할 수 없는 문제이다. 그럼에도 크기가 큰 모델의 성능이 작은 모델에 비해 더 좋으므로, 큰 모델의 성능을 작은 모델에서 근사하기 위한 다양한 방식들이 연구되어 왔다. 대표적으로 사용되는 방법중 하나는 Knowledge Distillation이 있다. 파라미터 수가 많은 teacher 모델에서 출력은 일반적으로 특정 레이블에 대한 하나의 확률값만을 나타내지만, 이를 확률값들의 분포 형태로 변형하여, 파라미터 수가 적은 student 모델의 학습에 사용하는 방법이다. 이를 위해 작은 모델을 초기화하는 과정에서, DistilBERT[3]에서는 행렬 연산을 수행하는 과정에서, 행렬의 크기보다는 연산하는 행렬의 개수가 속도에 더 큰 영향을 주므로, 레이어 수를 줄이는 방향으로 작은 모델의 구조를 결정하고, teacher 모델에서 일부 레이어를 추출하여 작은 모델의 레이어를 초기화하는데 사용하였다. 이후, finetuning 과정에서 DistilBERT[3]에서는 원핫이 아니라 teacher model의 logit에 대해 temperature scaling이 포함된 softmax를 수행한 뒤 student 모델의 logit에도 동일한 처리를 수행하여 크로스 엔트로피 loss를 계산하는 방식인 dilstillation loss을 제안한다. 이 loss와 데이터의 정답에 대한 student 모델의 loss 별도로 계산되는 다른 loss 세 개를 합쳐서 미세 조정을 수행하였을 때, 큰 모델과 성능이 유사한 작은 모델을 얻을 수 있었으며, 기존 BERT에 비해 40% 감소된 모델 크기와, 60%빨라진 연산 속도로, 97%의 성능을 유지하는 모델을 얻었다고 한다.

모델을 미세 조정 하는 과정에서 GPU 연산이 많이 필요하기 때문에, 사전 학습된 언어모델의 파라미터를 전부 다 미세 조정하지않고, 별도의 무작위 초기화된 파라미터를 갖는 adapter를 연산 과정에 추가하고, 모델 파라미터는 freeze 하여 학습하는 다양한 Parameter-Efficient Fine-Tuning (PEFT) 방법론들이 연구되어왔다[4]. LoRA[5]의 경우, 모델이 학습하는 중에 가중치 행렬 W의 변화가 '내재적 순위'가 낮다는 가설을 세우고, 이를 바탕으로 저순위 적응(Low Rank Approximation, LoRA) 접근법을 제안했다. 모델의 가중치 행렬 W에 병행하는 low rank 행렬 A, B 두 개를 추가하여, task에 대한 완전 미세 조정 시 W가 가지게 될 특정 task에 적응하기 위한 변화량 ΔW를 근사할 수 있게 한다.

$$h = W_0x + \Delta Wx = W_0x + BAx \tag{1}$$

위 수식에서 ΔW가 BA 에 의해 표현될 수 있을 만큼 low rank기 때문에, 즉, 변화량은 low rank 구조의 행렬 두 개를 행렬 곱한 것과 동일한 역할을 하도록 취급할 수 있으므로, 역전파 과정에서 모델의 가중치 행렬 W에 대한 gradient 계산이나 업데이트를 하지 않고 LoRA만 업데이트하는 구조를 제안했다.

모델의 파라미터와 activation을 더 작은 크기의 bit로 표현하여 메모리 차지하는 양, 메모리 접근 속도, 연산 속도를 높이는 quantization 또한 LLM을 학습하는 데에 있어 자주 사용된다.[6]

그 외에 모델에서 중요도가 낮은 뉴런을 제거하여 직접적인 모델 크기의 감소를 추구하는 방식인 Pruning 등이 주로 연구되어 왔다.

QLoRA[7] 같은 경우는 메모리 사용량을 크게 줄이는 효율적인 미세 조정 방식이다. 4-bit로 양자화된 모델을 가져와서 LoRA 기법을 활용해 미세 조정을 진행하는 방법으로, 32bit로 표현되던 값을 4bit로 감소하는 데에 정보량 손실이 크므로, 이를 보정하기 위해 NF4(4-bit NormalFloat) + 이중 양자화(Double Quantized)를 제안하였고, GPU의 VRAM 크기를 넘는 경우를 대비하여 페이지드 옵티마이저(Paged Optimizer) 등을 제안하여 메모리 소모를 크게 줄이는 학습 방식을 제안하였다.

2.3 Low Rank Factorization

Attention과 FFNN의 연산은 모두 행렬 곱셈이다. 복잡하게 모델 구조를 재설계하는 방식에 비해 고전적이고 쉽게 접근할 수 있는 모델 경량화 방식인 Low Rank Factorization에 대한 연구가 진행되어 왔다. 사전 학습된 모델의 파라미터인 행렬을 분해함으로써 얻으려는 이점은 수학적 방식을 통해 weight matrix를 low rank factorization하여 근사함으로, 파라미터 수에 이득을 보는 것이다.

DRONE[8]에서는 입력 X가 W를 거쳐 나온 weight-vector 곱 결과는 low rank인 경우가 많다는 점을 통해 $WX = UV^TX$를 만족할 수 있는 UV^T를 구하기 위해 주어진 task data에 대한 수학적인 연산을 수행하여 UV^T로 Low rank 구조를 초기화한다.

FWSVD[9]에서는 W가 특정 task에 대해 가지는 각 요소별 중요도를 구하기 위해 경험적 Fisher Information을 사용한다. 원본 모델을 task에 대해 미세 조정한 후, task의 train 데이터 각 샘플에 대한 gradient를 구하여 제곱한 뒤 평균을 내어 W에 대한 경험적 Fisher information matrix I를 얻고, I를 출력 차원에 대해 요소별 덧셈을 하여 입력차원 벡터만 남긴 뒤, 요소별 제곱근의 대각행렬I'을 W에 대한 중요도로 삼아 $I'W$를 SVD로 분해하여 low rank 구조를 만들고, I'^{-1}을 곱하여 제거하는 방식으로 모델을 초기화한 뒤 다시 task에 대해 미세 조정을 수행한다. 논문에서는 인코더 모델들을 대상으로 수행하여, 동일한 압축률에서 단순 SVD보다 더 나은 성능을 얻었다.

LightFormer[10]에서는 low rank factorization을 했을 때와

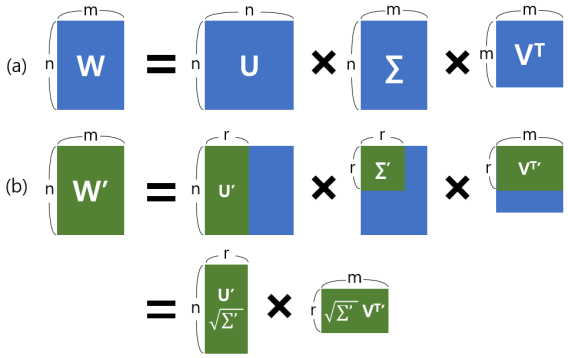

그림 1. (a)는 일반적인 SVD 방식. (b)는 W를 SVD로 분해하여 얻은 U, Σ, V^T 행렬에서 일부 특이값만을 취하여 W'을 만들 수 있는 Low Rank 행렬을 얻는 과정.

동일한 압축률을 얻고 싶지만, rank 값은 늘려서 원본 weight에 대한 재구성 오차를 더 줄이기 위해 weight sharing을 적용하기도 한다. task에 대해 원본 transformer를 미세 조정하고, rank 값을 결정한 뒤, SVD로 분해하고, 인코더의 일부 레이어들을 한 그룹으로 묶어서 그룹 내에서 파라미터를 공유하여, 25%로 압축한 모델을 사용하여 번역 task에서 원본 transformer와 유사한 성능을 달성하였다.

그러나 대부분의 선행 연구에서 weight는 low rank가 아닌 full rank인 점, SVD로 분해할 경우, 특이값의 크기와 성능 간에 비례관계가 존재하지 않다는 점의 이유를 들어 weight 자체를 분해하여 모델의 성능 하락이 존재할 수밖에 없기 때문에, weight만을 분해하는 것은 크게 선호하지 않는것으로 생각된다. DRONE[8], FWSVD[9], LightFormer[10] 모두 특정 task에 대한 finetuning을 진행한 후 SVD를 하거나 task의 데이터에 대해 종속적인 연산 과정을 거친다.

본 연구에서는 단순 SVD를 텍스트 생성을 위한 디코더 모델에 적용하여, 그 성능을 확인해본다. 또한, low rank 구조에 의한 제한이 존재하는지, 존재한다면 full finetuning 대신 LoRA를 통해서 개선할 수 있을지 비교해본다.

3. SVD

SVD는 선형대수학에서 행렬을 세 개의 다른 행렬로 분해하는 방법이다. SVD로 분해한 뒤 크기가 큰 특이값을 원하는 만큼, r개 선택하여 low rank를 얻는 방법은 Frobenius norm 측면에서 원본 W와의 차이를 최소화할 수 있는 방식이다. 이는 다음과 같은 수식으로 표현된다.

$$W_{n\times m} = U_{n\times n}\Sigma_{n\times m}V^T_{m\times m} \tag{2}$$

$$\begin{aligned} U'_{n\times r} &= U_{n\times n}[:, :r] \\ \Sigma'_{r\times r} &= \Sigma_{n\times m}[:r, :r] \\ (V^T)'_{r\times m} &= V^T_{r\times m}[:r, :] \end{aligned} \tag{3}$$

$$W_{n\times m} \approx W'_{n\times m} = U'_{n\times r}\sqrt{\Sigma'_{r\times r}}\sqrt{\Sigma'_{r\times r}}(V^T)'_{r\times m} \tag{4}$$

그림 1 에서 이 수식을 그림으로 표현하였다.

Low rank factorization을 위해서 SVD를 사용할 경우, 제시되는 문제점들은 다음과 같다. 첫 번째로, 이상적으로는 크기가 작은 특이값을 잘라낼 때 성능 저하가 적을 것으로 생각되지만, 특이값 상위 일부 rank를 취하여 행렬을 근사하는 것은 모델의 성능과의 연관 관계를 찾을 수는 없는 것으로 생각된다. FWSVD[9]에서는 제거되는 특이값이 모델의 성능에 어떤 영향 주는지 확인하였다. 전체 특이값 대각 행렬 Σ 에서 특이값들을 크기별로 10개의 그룹으로 묶어서 각각 제거하고 모델의 성능을 측정한 결과, 가장 작은 특이값 묶음을 제거하는 것이 그보다 큰 특이값 묶음을 제거하는 것보다 더 큰 성능 하락을 가져왔다고 제시하였다. 두 번째로, SVD를 통해 경량화된 low rank 모델이 원본 모델로부터 어떤 점에 손실을 입었는지 알 수 없다.

4. 실험

4.1 데이터 셋

한국어 문서 기반 대화 시스템의 학습을 위한 KoDoc2Dial[11] 데이터 셋을 사용하였다. Doc2Dial은 문서에 기반한 대화 데이터 셋으로, 하나의 문서가 주어지면 두 사람이 문서의 내용과 관련된 발화를 번갈아 가며 제시한다. KoDoc2Dial은 이러한 Doc2Dial 데이터 셋을 기반으로 한국어로 번역하여 5개의 도메인에 대한 문서 487개, 대화 4,922개 로부터 학습을 위한 데이터 13,737개, 평가를 위해 2,678개의 데이터를 준비하였다. 입력 데이터의 구성은 다음과 같다.

"당신은 주어지는 문서를 보고, 이전 대화 기록을 참조하여, 질문에 답변해야 합니다. 질문은 명백한 지식을 물어볼 수도 있고, 뚜렷한 정답이 없는 선호 대상 추천이나 공감을 바라는 질문일 수도 있습니다. 주어지는 문서는 다음 양식을 따릅니다. ...(중략)... 당신은 ⟨답변 내용⟩을 작성해야 합니다. 이제 문서를 보고, 다음 사용자와 에이전트의 대화 기록을 참조하여, 질문에 대한 ⟨답변 내용⟩을 잘 작성하세요. [문서] {passage} [대화 기록] {dialogue} [질문] {question} [답변]"

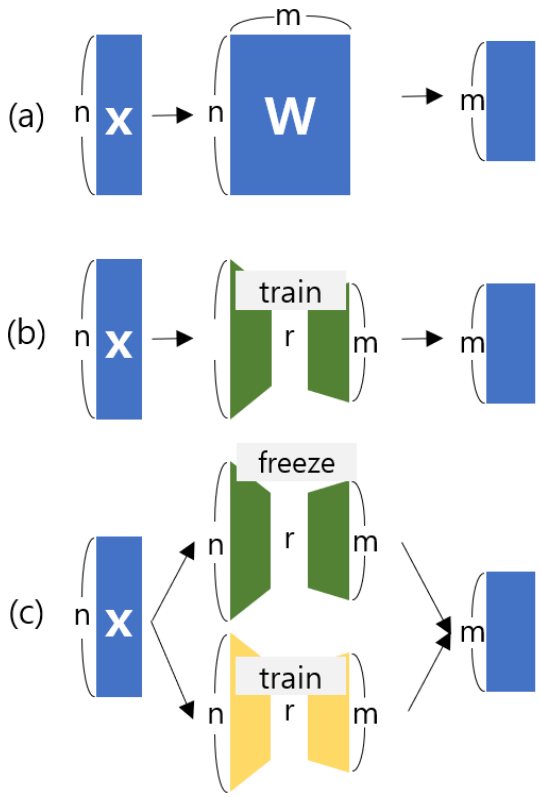

그림 2. (a)는 아무런 변화 없이 W를 사용하는 것. (b)는 SVD로 분해 후 train 하는 것. (c)는 SVD로 분해 후 low rank로 표현된 W' 부분은 freeze 하고, LoRA 부분만 학습 하는 것.

4.2 실험 설정

실험에 사용할 사전 학습 모델로 Polyglot-ko 를 사용하였다. 다양한 비교를 위해 1.3B과 3.8B의 두 가지 크기의 모델을 선택하였다. 각각 **1.3B**, **3.8B**로 표기한다. 경량 학습 방법론으로 SVD와 LoRA 방식을 적용한다. 각각 원본 모델 뒤에 **-S**, **-L**를 붙여 표기한다. 이 설정들을 조합하여 두 규모 각각 원본, SVD 적용, SVD와 LoRA적용의 3가지 설정으로 총 6가지 모델을 학습한다.

각 실험의 학습은 동일한 값으로 진행하였다.

epoch: 3, batch size: 64, gradient clipping: 1.0, learning rate: 3e-4 learning rate scheduler: warmup with linear, warmup: 0.3 epoch, weight decay: 0.02

모델에서 SVD로 분해한 행렬은 다음과 같다. self attention의 query_key_value와 dense, feed forward network의 dense_h_to_4h와 dense_4h_to_h 를 분해하였고, rank는 다음과 같이 정하였다. 원본 모델이 3.8B인 경우: 768, 원본 모델이 1.3B인 경우: 512

LoRA가 붙는 경우 LoRA의 하이퍼파라미터는 다음과 같다. rank: 32, α: 32, dropout: 0.1

LoRA가 붙을 때에는 그림 2의 (c) 처럼 행렬에 대한 연산이 진행된다.

매 0.5 epoch마다 평가하였다.

4.3 비교 대상

FWSVD[9]에서는 BERT 모델을 SVD로 분해 한 후 미세조정해도 GLUE[12] task들에 대한 평균 성능이 80.1로 원본 BERT-base가 낸 점수인 84.1에 비교해 심각한 성능 하락으로 보이지는 않았다. 이에 문서 기반 대화 시스템의 응답 생성 과정에서도 SVD만 사용한 뒤 미세 조정했을 때 성능을 확인해 본다.

또한 LLM을 사용하는 방식 중 범용성을 증가시키는 대표적인 방식인 LoRA를 함께 적용하여 결과가 달라지는지 실험해 본다.

4.4 평가 방법

4.4.1 정량 평가

프롬프트, 문서, 질문을 입력하고, 답변만을 생성하도록 하였고, 생성된 답변 부분과 정답 텍스트 간의 F1, Meteor, Rouge-L, SacreBLEU 점수를 구하였다. 각 성능 지표는 [0, 1] 범위로 측정되고 100을 곱하여 표시하였다.

4.4.2 정성 평가

각 모델들이 질문에 대해 생성한 텍스트를 평가하기 위해 무작위로 30개의 샘플을 선택하여 정성 평가를 수행하였다. 다음 세 개의 기준에 따라 분류하였다.

- 좋음: 생성이 문법에 맞는 문장이고, 제시된 레이블(정답)과 같은 말이거나, 정답이 아니더라도 질문에 적절한 응답을 하여 도움이 되는 경우.
- 나쁨: 생성이 문법에 맞는 문장이지만, 정답이 아니고 질문에 도움도 안되는 경우.
- 판단불가: 생성이 문법에 맞는 문장이 아니고, 정답 여부를 판별할 수 없는 경우.

5. 실험 결과

평가 결과는 표 1과 같다.

표 2는 한 예제의 각 모델 별 생성 결과와 정성 평가 기록을 나타낸 것이다.

정성평가 결과, 아무런 변화를 주지 않고 full 미세 조정한 **3.8B**이 좋음 20개로 가장 높은 성능을 보였고, 그 다음으로 변화를 주지 않은 1.3B 모델을 full 미세 조정한 **1.3B**과, 1.3B 모델을 SVD를 사용하여 rank 512로 분해하여 0.5B가 된 모델을 full 미세 조정 한 **1.3B-S**이 좋음 17개로 두 번째로 좋은 성능을 보였다. 그 다음으로 3.8B 모델을 SVD로 분해한 후 LoRA를 붙여 학습한 **3.8B-S-L**이 좋음 15개로 좋은 성능을

표 1. 각 실험별 답변 생성 정량 평가 및 정성 평가 결과

모델	학습 설정		규모(B)		정량 평가					정성 평가		
구분	SVD	LoRA	전	후	F1	Meteor	Rouge-L	SacreBLEU	Total	좋음	나쁨	판단불가
3.8B	X	X	3.8	3.8	17.68	18.07	11.47	8.78	56.00	20	8	2
1.3B	X	X	1.3	1.3	17.04	17.33	10.39	8.13	52.89	17	12	1
1.3B-S	O	X	1.3	0.5	14.47	15.05	**9.28**	**6.38**	44.43	17	11	2
3.8B-S-L	O	O	3.8	1.4	**14.90**	**15.17**	8.37	5.99	**45.18**	15	11	4
1.3B-S-L	O	O	1.3	0.5	12.58	13.44	8.45	5.19	39.66	11	14	5
3.8B-S	O	X	3.8	1.4	7.68	8.98	5.33	2.10	24.09	7	13	10
1.3B-P-FS	-	X	1.3	0.5	9.27	10.22	7.36	2.35	29.19	-	-	-
1.3B-P-S	O	X	1.3	0.5	10.21	11.70	5.50	2.65	30.05	-	-	-

보였다. 1.3B 모델을 SVD로 분해한 뒤 LoRA를 붙여 학습한 **1.3B-S-L**은 좋음 11개로, 동일한 조건에서 LoRA를 붙이지 않고 학습한 **1.3B-S**보다 좋음 6개 만큼 떨어지는 성능을 보였다. 마지막으로, 3.8B 모델을 SVD를 사용하여 rank 768로 분해하여 1.4B가 된 **3.8B-S**은 정상적인 문장을 생성해 내지 못했다.

정량 평가 결과와 정성 평가 결과 간에 어느정도 연관성이 존재하기는 한 것으로 보인다. 그러나, 정량 평가에서 **1.3B**와 **1.3B-S**를 비교했을 때, **1.3B-S**가 더 좋지 않은 점수를 보였지만, 정성 평가 결과, 큰 차이가 없는 것으로 보인다. 그러나, 동일한 조건이라고 볼 수 있는 **3.8B**과 **3.8B-S**를 비교했을 때, **3.8B-S**는 학습 자체가 잘 되지 않는 모습을 보였고, 생성 결과 또한 같은 토큰을 반복하여 생성하는 등, 언어 모델로써의 기능을 제대로 하지 못하는 모습을 보였다.

학습이 잘 된 것으로 생각되는 **3.8B-S-L**과 원본 **1.3B**을 비교했을 때, **3.8B-S-L**가 정량 평가와 정성 평가 모두에서 더 낮은 성능을 보였다.

또한, FWSVD 논문에서 제시한 방법대로 사전학습된 언어 모델을 full finetuning - FWSVD - full finetuning 한 것(**1.3B-P-FS**)과, full finetuning - SVD - full finetuning 한 것(**1.3B-P-S**)을 추가로 실험하였으나, **1.3B-S** 보다 낮은 성능을 보였다. 또한, 추가로 실험한 두 모델 모두 대다수가 판단 불가능한 텍스트를 생성하였다.

6. 결론

LLM을 경량화하기 위해 SVD 방식을 사용하여 Attention과 FFNN를 분해하였다. 실험 결과, SVD 방식을 사용하는 것보다 비슷한 크기의 모델을 사용하는 것이 좋을 것으로 생각된다. 그러나, 0.5B 모델의 경우에 분해하기 전의 성능과 비슷한 정성평가 결과를 내는 것을 고려하면, 반드시 안 좋아진다고 말할 수는 없을 것으로 생각된다. 실험 결과, LoRA를 붙이는 것을 비교해 보았을 때, **3.8B-S**에서는 학습이 아예 되지 않았으나, 3.8B-S-L 에서는 LoRA를 붙여서 판단불가인 텍스트 생성이 줄어드는 것을 보아, 낮아진 rank 수에 따라 학습할 수 있는 성능의 상한이 있거나, 언어 능력이 상실되는 것으로 추측할 수 있고, LoRA 부분이 언어능력을 보정해 준다고 말할 수 있을 것으로 생각된다.

FWSVD 논문에서 제안한 방식을 사용하여 모델을 경량화 할 경우, FWSVD가 적용되는 대상 모델이 fisher information을 구하려는 task에 대해 학습이 된 상태여야 하는 것으로 생각된다. 그러나, **1.3B-P-S**와 **1.3B-P-FS**의 점수가 비슷하다는 점과, svd 전에 full finetuning을 한 **1.3B-P-S**보다, 하지 않은 **1.3B-S** 의 성능이 더 높은 것을 고려하였을 때, 단순 SVD를 하는것 대신 FWSVD를 수행하는 것의 이점을 찾을 수 없었다.

7. 향후 연구

SVD방식을 통해 원본 행렬과의 Frobenius norm을 최소화하는 low rank 행렬을 구하여 원본 행렬을 근사하는 방식은 low rank 행렬을 초기화 하는데에 있어 고려해볼만한 방법이다. 그러나, 경량화를 통해 원본 행렬에서 손실된 값이 어떠한 정보를 나타내는지는 알 수 없는 부분이다. 향후 이러한 부분에 대해 비교 및 평가를 하기 위해 SVD로 분해된 구조를 추가로 사전학습을 수행해보고, 대화 시스템을 위한 미세 조정을 수행하여 답변이 잘 생성될 수 있는지 비교해 볼 예정이다.

본 연구를 통해 원본과 정성평가 결과가 유사한 **1.3B-S** 모델을 얻을 수 있었다. 이제, 이 모델에 대해 quantization과 같은 추가적인 경량화 방식을 적용해보며 추가로 압축을 시도해보고, 성능이나 생성된 텍스트의 품질이 어느 정도 떨어지는지 확인해 볼 예정이다.

참고문헌

표 2. 실험 설정별 생성 예시: 데이터 중 레이블이 비정상적인 데이터의 생성 예시:

문서: 직장으로 돌아간다면 대부분의 사람들과 마찬가지로 장애 혜택을 받으며 살기보다는 일을 하고 싶을 것입니다. 근로 능력을 테스트하는 동안 현금 혜택과 Medicare를 유지하는 데 도움이 되는 특별 규정이 있습니다. 우리는 이러한 규칙을 "근로 인센티브"라고 부릅니다. 사회 보장 근로 인센티브에 대한 자세한 내용은 장애인 동안 일하기: 우리가 도울 수 있는 방법을 읽으십시오.		
대화 기록	없음	
질문	사람이 직장으로 돌아가면 어떻게 됩니까?	정성 평가
정답	다시 일하러 갔어?	비정상
3.8B	대부분의 사람들과 마찬가지로 장애 혜택을 받으며 살기보다는 일을 하고 싶을 것입니다.	좋음
1.3B	대부분의 사람들은 장애 혜택을 받기보다 일을 하고 싶어합니다.	좋음
1.3B-S	대부분의 사람들은 장애 혜택을 받을 자격이 있을 수 있습니다.	판단불가
3.8B-S-L	대부분의 사람들과 마찬가지로 장애 혜택을 받으며 살기보다는 일을 하고 싶을 것입니다.	좋음
1.3B-S-L	근무 능력을 테스트하는 동안 현금 혜택과 Medicare를 유지하는 데 도움이 되는 특별 규정이 있습니다.	좋음
3.8B-S	귀하는 사람이 일하는 동안 일하기 위해 일하는 것이 중요합니다.	판단불가

[1] T. B. Brown *et al.*, "Language models are few-shot learners," 2020.

[2] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways," 2022.

[3] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2020.

[4] V. Lialin, V. Deshpande, and A. Rumshisky, "Scaling down to scale up: A guide to parameter-efficient fine-tuning," 2023.

[5] E. J. Hu *et al.*, "Lora: Low-rank adaptation of large language models," 2021.

[6] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017.

[7] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[8] P. Chen *et al.*, "Drone: Data-aware low-rank compression for large nlp models," *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., Vol. 34, pp. 29 321–29 334, 2021. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/f56de5ef149cf0aedcc8f4797031e229-Paper.pdf

[9] Y.-C. Hsu, T. Hua, S. Chang, Q. Lou, Y. Shen, and H. Jin, "Language model compression with weighted low-rank factorization," 2022.

[10] X. Lv, P. Zhang, S. Li, G. Gan, and Y. Sun, "LightFormer: Light-weight transformer using SVD-based weight transfer and parameter sharing," *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10 323–10 335, Jul. 2023. [Online]. Available: https://aclanthology.org/2023.findings-acl.656

[11] 김보은, 이도행, 장영진, 황금하, 권오욱, and 김학수, "Kodoc2dial: 문서 기반 대화를 위한 한국어 대화 데이터셋," *한국정보과학회 2022 한국컴퓨터종합학술대회 논문집*, pp. 458–460, 2022.

[12] A. Wang *et al.*, "Glue: A multi-task benchmark and analysis platform for natural language understanding," 2019.