



2024 Summer Seminar

LoftQ | Divergent Token Metrics

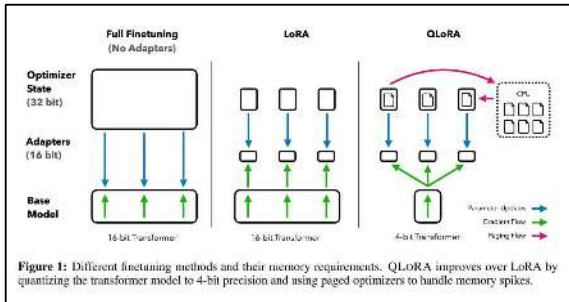
김민혁

Preview



- LIAMA 3 - 8B, 70B
 - 8B 모델 실행에만 최소 16GB RAM
 - Fine-tuning을 위해선 고성능의 GPU 필요

Preview



- QLoRA
 - Fine-tuning에 드는 컴퓨팅 자원을 최소화할 수 있는 기법

Preview



- On-Device AI
 - 모델의 구동 환경을 개인의 디바이스에 이르게끔 최적화
 - 삼성, 애플 등이 자사 제품에 On-Device AI를 적용하기 시작

Preview

원래 모델과 경량화된 모델 간의 격차를 어떻게 좁힐 것인가?

- LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models
- Divergent Token Metrics: Measuring degradation to prune away LLM components – and optimize quantization



LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models

ICLR 2024

김민혁

Goal

- Full fine-tuning과 QLoRA fine-tuning 사이의 성능 차이가 발생함
 - LoFTQ(논문에서 제안하는 새로운 Quantization framework)로 이러한 차이를 최소화

Abstract

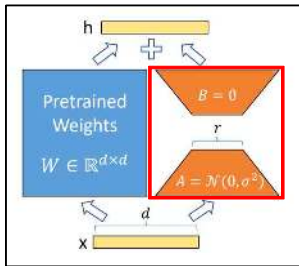
- Downstream task에서 Full fine-tuning과 QLoRA fine-tuning의 성능 차이가 나타남
- 논문에서 제안하는 LoftQ는 LLM을 양자화함과 동시에 적절한 Low-rank 초기화 값을 찾음
 - 이러한 초기화는 Quantized 모델과 Full-precision model 간의 차이를 완화함
 - Quantized 모델의 Downstream tasks 성능을 향상시킴
- 실험 결과 기존의 Quantization 기법의 성능을 능가함
 - 특히 2-bit, 2/4-bit mixed precision에서 성능 향상 폭이 컸음

Background - Quantization

$$X^{\text{INT}} = \text{round} \left((2^N - 1) F(X^{\text{HP}}) \right)$$

- Quantization
 - F는 정규화 함수 (F에 따라 Uniform / Normal Float Quantization으로 구분됨)
 - Normal Float Quantization은 X(가중치)가 중심극한정리에 따라 표준 정규 분포를 따른다고 가정함
- Dequantization은 해당 과정을 역으로 적용

Background - LoRA



- Fine-tuning 비용을 줄이기 위해서 기존 가중치 값은 변경하지 않고, Low-rank Adaptor을 붙이고 해당 부분만 학습하는 방식

Background - LoRA

$$Y = XW + XAB^T$$

- 선형 변환 $Y = XW$ 의 Pre-trained 가중치 행렬 W 에 두 개의 작은 가중치 행렬 A, B 이 붙음
- $A \sim N(0, \sigma^2)$, $B = 0$ 로 초기화되며, Fine-tuning 동안 해당 값을 업데이트

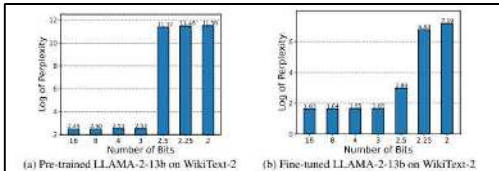
Background - LoRA

$$Y = XW + XAB^{\top}$$


$$Q + AB^{\top}$$

- Quantization + LoRA를 하게 되면, 기존 가중치 행렬 W와의 차이가 발생함
 - Quantization이 된 Q(W)에 이미 W와의 약간의 차이가 생겼고, 이후 A,B를 더하며 성능 저하를 초래함

Introduction



- Pre-trained 모델을 Quantizing할 때 Quantization 기법 자체에 초점을 두는 경우가 많음
 - 따라서 LoRA fine-tuning시 0으로 초기화된 Low rank adapter을 붙이는 경우가 많음
 - 그러나 이러한 방식의 QLoRA는 3-bit 이하로 Quantization을 진행할 때 심각한 성능 저하를 보임

Introduction

- 본 논문에서는 LoftQ(**Lo**RA-**F**ine-**T**uning-aware Quantization)을 제안함
 - Quantization과 Low-rank approximation을 공동으로 진행

Method

$$\min_{Q,A,B} \|W - Q - AB^{\top}\|_F$$

- LoRA Aware Quantization
 - 양자화된 Q와 어댑터 A, B의 초기 값을 공동으로 최적화
- 프로베니우스 Norm 사용
- $W = Q + AB$ 가 되도록 최적화

Method

$$Q_t = q_N(W - A_{t-1}B_{t-1}^\top)$$

- Alternating Optimization - Quantization

$$R_t = \sum_{i=1}^d \sigma_{t,i} u_{t,i} v_{t,i}^\top$$

- Alternating Optimization - SVD

- W - Q 에 대한 잔차 R 에 특이값 분해를 적용

T만큼 반복



Method

$$\begin{aligned} A_t &= [\sqrt{\sigma_{t,1}}u_{t,1}, \dots, \sqrt{\sigma_{t,r}}u_{t,r}] \\ B_t &= [\sqrt{\sigma_{t,1}}v_{t,1}, \dots, \sqrt{\sigma_{t,r}}v_{t,r}]. \end{aligned}$$

- A,B를 통해 잔차 R을 비교적 정확히 계산할 수 있음
 - 이를 통해 원래의 가중치 행렬 W를 비교적 정확히 계산할 수 있음

Method

Algorithm 1 LoftQ

input Pre-trained weight W , target rank r , N -bit quantization function $q_N(\cdot)$, alternating step T

1: Initialize $A_0 \leftarrow 0, B_0 \leftarrow 0$

2: **for** $t = 1$ to T **do**

3: Obtain quantized weight $Q_t \leftarrow q_N(W - A_{t-1}B_{t-1}^\top)$

4: Obtain low-rank approximation $A_t, B_t \leftarrow \text{SVD}(W - Q_t)$ by (9)

5: **end for**

output Q_T, A_T, B_T

Experiments - Settings

- Models
 - DeBERTaV3-base (Encoder ONLY), BART-large (Encoder-Decoder), LLAMA-2 series (Decoder ONLY)
- Quantization Methods
 - Uniform Quantization
 - NF4, 2-bit variant NF2

Experiments - Results

Rank	Method	MNLI m / mm	QNLI Acc	RTE Acc	SST Acc	MRPC Acc	CoLA Matt	QQP Acc	STSB P/S Corr	SQuAD EM/F1	ANLI Acc
-	Full FT	90.5/90.6	94.0	82.0	95.3	89.5/93.3	69.2	92.4/89.8	91.6/91.1	88.5/92.8	59.8
16	LoRA	90.4/90.5	94.6	85.1	95.1	89.9/93.6	69.9	92.0/89.4	91.7/91.1	87.3/93.1	60.2
16	QLoRA	75.4/75.6	82.4	55.9	86.5	73.8/82.8	N.A.	86.8/82.3	83.0/82.8	61.5 / 71.2	N.A.
	LoftQ	84.7/85.1	86.6	61.4	90.2	83.8/88.6	37.4	90.3/86.9	87.1/86.9	81.5/88.6	47.1
32	QLoRA	78.5/78.7	80.4	56.7	86.9	73.8/82.7	N.A.	87.1/82.7	83.6/83.3	64.6/73.8	N.A.
	LoftQ	86.0/86.1	89.9	61.7	92.0	83.6/87.2	47.5	91.0/87.9	87.5/87.0	82.9/89.8	49.0

- DeBERTa V3-base (Encoder ONLY) / NF2 Quantization / GLUE, SQuADv1.1
- N.A.는 모델이 converge할 수 없었던 경우

Experiments - Results

Rank	Method	MNLI m / mm	QNLI Acc	RTE Acc	SST Acc	MRPC Acc	CoLA Matt	QQP Acc	STS P/S Corr	SQuAD Em/F1
-	Full FT	90.5/90.6	94.0	82.0	95.3	89.5/93.3	69.2	92.4/89.8	91.6/91.1	88.5/92.8
16	LoRA	90.4/90.5	94.6	85.1	95.1	89.9/93.6	69.9	92.0/89.4	91.7/91.1	87.3/93.1
16	QLoRA	76.5/76.3	83.8	56.7	86.6	75.7/84.7	N.A.	87.1/82.6	83.5/83.4	69.5/77.6
	LoftQ	87.3/87.1	90.6	61.1	94.0	87.0/90.6	59.1	90.9/88.0	87.9/87.6	84.4/91.2
32	QLoRA	79.9/79.5	83.7	52.8	86.9	76.5/84.5	N.A.	88.6/84.7	84.1/84.0	71.6/80.2
	LoftQ	88.0/88.1	92.2	63.2	94.7	87.5/91.2	60.5	91.3/88.3	89.5/89.2	85.2/91.6

- DeBERTa V3-base (Encoder ONLY) / Uniform Quantization / GLUE, SQuADv1.1

Experiments - Results

Quantization	Rank	Method	XSum	CNN/DailyMail
Full Precision	-	Lead-3	16.30/1.60/11.95	40.42/17.62/36.67
		Full FT	45.14/22.27/37.25	44.16/21.28/40.90
	8	LoRA	43.40/20.20/35.20	44.72/21.58/41.84
	16	LoRA	43.95/20.72/35.68	45.03/21.84/42.15
NF4	8	QLoRA	42.91/19.72/34.82	43.10/20.22/40.06
		LoftQ	44.08/20.72/35.89	43.81/20.95/40.84
	16	QLoRA	43.29/20.05/35.15	43.42/20.62/40.44
		LoftQ	44.51/21.14/36.18	43.96/21.06/40.96
Uniform	8	QLoRA	41.84/18.71/33.74	N.A.
		LoftQ	43.86/20.51/35.69	43.73/20.91/40.77
	16	QLoRA	42.45/19.36/34.38	43.00/20.19/40.02
		LoftQ	44.29/20.90/36.00	43.87/20.99/40.92

- BART-large (Encoder-Decoder) / XSum, CNN/DailyMail

Experiments - Results

Rank	Method	XSum	CNN/DailyMail
8	QLoRA	N.A.	N.A.
	LoftQ	39.63/16.65/31.62	42.24/19.44/29.04
16	QLoRA	N.A.	N.A.
	LoftQ	40.81/17.85/32.80	42.52/19.81/39.51

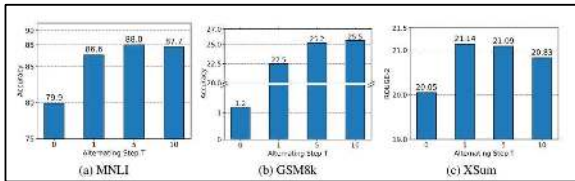
- BART-large (Encoder-Decoder) / NF2 Quantization / XSum, CNN/DailyMail
- LoftQ의 N-bit 양자화에 대한 Robustness

Experiments - Results

Method	Bit	LLAMA-2-7b		LLAMA-2-13b	
		WikiText-2↓	GSM8K↑	WikiText-2↓	GSM8K↑
LoRA	16	5.08	38.5	5.12	48.8
QLoRA	4	5.70	38.2	5.22	48.8
LoftQ	4	5.24	38.0	5.16	49.1
QLoRA	3	5.73	32.1	5.22	40.7
LoftQ	3	5.63	36.2	5.13	45.4
QLoRA	2.5	N.A.	N.A.	19.39	N.A.
LoftQ	2.5	5.78	31.1	5.22	41.1
QLoRA	2.25	N.A.	N.A.	N.A.	N.A.
LoftQ	2.25	6.13	27.5	5.45	38.1
QLoRA	2	N.A.	N.A.	N.A.	N.A.
LoftQ	2	7.85	26.5	7.69	33.4

- LLAMA-2 series (Decoder ONLY) / WikiText-2 and GSM8K

Experiments - Results



- Alternating Optimization의 반복 횟수 T 값에 따른 성능 비교
- Uniform 2-bit DeBERTaV3-base (왼쪽), NF2 2-bit LLAMA-2-13b (중간), NF4 BART-large (오른쪽)
- 일정 반복 횟수가 지나면 성능 저하가 발생함
 - 갭이 작아질수록 Alternating Optimization이 각 단계에서 일관되게 갭을 최소화하는 것이 어려워짐
 - 양자화 과정으로 인한 고유한 오류로 인해 발생

Conclusion

- Pros
 - QLoRA의 성능을 압도하는 Quantization 프레임워크 제안
 - 2bit와 같은 낮은 비트로 구성된 Quantization 환경에서도 Robustness를 보여줌
- Cons
 - Decoder ONLY 모델을 다양화해서 실험해보지 않았음

Appendix A

Method	Rank	MNLI m / mm	QNLI Acc	SST2 Acc
Full FT	-	90.5/90.6	94.0	95.3
QLoRA	32	79.9/79.5	83.8	86.6
LoftQ(SVD First)	32	87.8/87.7	84.9	89.7
LoftQ(Quantization First)	32	88.0/88.1	92.2	94.7



Divergent Token Metrics: Measuring degradation to prune away LLM components – and optimize quantization

NAACL 2024

김민혁

Goal

- Compression(압축)에 적합한 개별 component를 찾기 위해 어떤 Metric을 사용해야 하는가?
 - PPL (Perplexity)
 - DPPL (Divergent Perplexity)
 - SDT (Share of divergent tokens)
 - FDT (First divergent token)

Abstract

- 모델 경량화를 위한 Metrics(Perplexity, Accuracy)는 한계 존재
 - Text Generation Quality를 반영하지 못함
- 논문에서 제안하는 Divergent Token Metrics(DTMs)를 통해 한계를 극복 가능
- SOTA Performance를 유지하며 25%의 Attention Components의 90%를 넘게 Pruning 가능
- 80%가 넘는 파라미터에 대해서 별도의 이상치 처리를 하지 않아도 Naive하게 int8 형태로 변환할 수 있음
 - Compression이 가능한 적절한 파라미터를 찾는게 중요함

Introduction

- Perplexity와 기존 NLP Benchmark는 Diverging performance nuance를 캡처하는데 실패함
 - 불연속적인 텍스트 생성 과정을 무시했기 때문
 - 문법이나 수치 등에서 상당한 차이가 발생할 수 있음

Introduction

Baseline	Albert Einstein was born on March 14, 1879, in Ulm, Germany.	PPL 6.420
Compressed	Albert Einstein was born on March 21, 1879, in Vienna, Austria.	PPL 6.420
	0 1 2 3 4 5 prefix completion	
	FDT 6	

- 중요한 정보를 틀리게 생성했지만 Perplexity로는 이를 반영하지 못함
- 논문에서 제안한 Metrics 중에 하나인 FDT는 이러한 차이점을 식별할 수 있음

Model Divergence Metrics - Basic

$$\begin{aligned}\mathcal{G}(F, y:n, N)_{i+1} \\ = \arg \max_j F(\mathcal{G}(F, y:n, N)_{:i})_{ij}.\end{aligned}$$

- Model F가 있을 때, n 길이의 Prefix가 주어지고 이를 N 길이까지 생성하는 Greedy한 과정
- i+1 에 들어갈 가장 높은 Logit을 가진 j를 구함

Model Divergence Metrics - Perplexity

$$\begin{aligned} \text{NLL}(y, F, n) \\ = -\frac{1}{N-n} \sum_{i=n}^{N-1} \log \mathbb{P}(y_{i+1} | y_i, \dots, y_1), \end{aligned}$$

- Model F가 n+1부터 N까지 생성할 Negative Log-likelihood
- i까지가 주어졌을 때, i+1이 발생할 조건부 확률의 로그 합의 평균을 음수화
 - 해당 값이 적을수록 모델이 안정적으로 결과값을 생성

Model Divergence Metrics - Perplexity

$$\text{PPL}(y, F, n) = \exp(\text{NLL}(y, F, n)).$$

- Negative Log Likelihood 값에 지수함수를 취함
- 논문이 제안한 Metrics와의 비교 지표

Model Divergence Metrics - DPPL

$$\begin{aligned} M_{\text{DPPL}}(F, F', y_{:n}, N) \\ = \text{PPL}(\mathcal{G}(F, y_{:n}, N), F', n) . \end{aligned}$$

- Divergent Perplexity (DPPL)
- Model F의 결과 값을 Ground-Truth 값으로 가정했을 때, 압축된 F'의 PPL

Model Divergence Metrics - SDT

$$\begin{aligned} \text{SDT}(y, F, n) \\ = |\{i \geq n: \arg \max_j F(y)_{ij} \neq y_{i+1}\}|, \end{aligned}$$

- Share of divergent tokens (SDT)
- Model F의 결과 값을 Ground-Truth 값으로 가정했을 때, 압축된 F'의 수정이 필요한 정도

Model Divergence Metrics - FDT

$$\begin{aligned} & \text{FDT}(y, F, n) \\ &= \min\{i \geq n : \arg \max_j F(y)_{i,j} \neq y_{i+1}\} - n, \end{aligned}$$

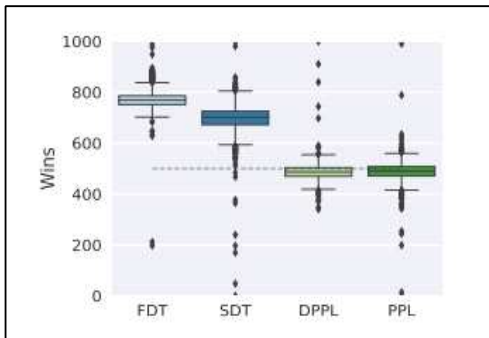
- First divergent tokens (FDT)
- Model F의 결과 값이 Ground Truth와 달라지는 최초의 위치

Model Divergence Metrics

$$\begin{aligned} |\text{PPL}(y, l, 1) - \text{PPL}(y, l', 1)| &< \varepsilon, \\ M_{\text{SDT}}(l, l', y_{:1}, N) &= N. \end{aligned}$$

- 모델 l 과 압축된 l' 의 PPL 값의 차이가 ε 미만으로 작아도 SDT 값이 N 만큼 커질 수 있음
 - (Albert Einstein was born on) March 14
 - (Albert Einstein was born on) April 21

Experiments

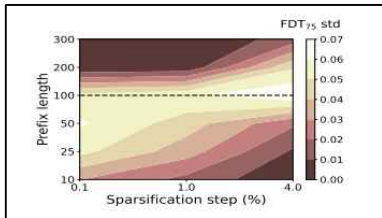


- FDT, SDT를 사용하여 Pruning한 결과 > 기존 평가지표(PPL)를 사용하여 Pruning한 결과

Experiments - Settings

- Models
 - Llama2-7B, 13B
- Evaluation
 - Wikitext2 dataset
 - NLP Benchmarks
- Sparsity
 - 20, 15, 10, 10, 5, 5, 5(%)로 점진적으로 증가시키며 실험

Experiments - Settings



- FDT 값의 분산이 75% 이상의 구간에서 높게 나타남 (성능 비교가 용이함)
- Prefix를 100으로 주었을 때 마찬가지로 분산이 높게 나타남
- 따라서 실험은 해당 값을 고정하고 진행

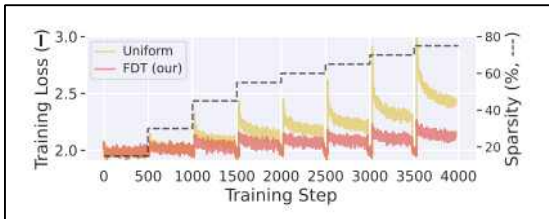
Experiments

- 모델의 Component에 대해 다음과 같은 Metrics를 수행하여 Sparsification과 Quantization의 대상이 되는 Component와 그 비율을 선택
 - FDT
 - DPPL
 - PPL
 - Uniform

Experiments - Results

- A. FDT 기반 모델 경량화 기법의 성능이 PPL 기반 기법보다 뛰어남
- B. FDT기반 Pruning하는 과정에서 각 Layer의 Component별 중요도를 식별하는 것이 수월함

Experiments - Result A



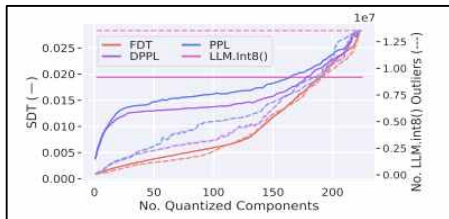
- FDT vs Uniform (Sparsification)
- Uniform 하게 Pruning하는 것보다 FDT를 사용하여 선택된 Component에 개별적으로 Pruning을 적용했을 때 Convergence가 훨씬 빠름

Experiments - Result A

Model	Sparsification		
	FDT ↑	PPL ↓	NLP ↑
Llama2-13B	-	4.884	53.59
~ 60% sparse (unif.)	4.7	9.244	46.32
~ 60% sparse (our)	7.9	6.242	48.89
~ 75% sparse (unif.)	3.5	13.512	41.67
~ 75% sparse (our)	5.5	8.101	46.32
~ 80% sparse (our)	5.2	9.531	45.66

- 사용 Metric 별 성능 비교 (Uniform vs. FDT - Sparsification)
 - 모든 부문에서 FDT가 Uniform을 능가함을 확인할 수 있음
 - 70% 이상의 Sparsification이 진행됐을 때 PPL 점수가 한 자릿수를 달성한 것은 최초임

Experiments - Result A



- Metrics 별 성능 비교 (Quantization)
 - Quantized Components가 많을수록 이상치와 SDT값이 증가
 - 이때, FDT가 PPL, DPPL보다 해당 값이 낮음을 보임

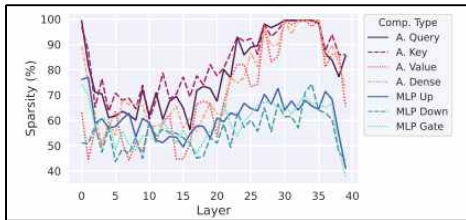
Experiments - Result A

		Quantization		
	Model	FDT $\uparrow$	PPL $\downarrow$	NLP $\uparrow$
	Llama2-7B	-	5.472	50.79
	LLM.int8() <i>all</i>	36.1	5.505	50.81
int8	AbsMax PPL ₁₅₀	46.3	5.500	50.72
	AbsMax DPPL ₁₅₀	54.1	5.490	50.75
	AbsMax FDT ₁₅₀ (our)	71.7	5.489	50.75
	GPTQ <i>all</i>	11.1	5.665	48.34
int4	GPTQ PPL ₁₆	45.0	5.511	49.91
	GPTQ DPPL ₁₆	137.0	5.476	50.02
	GPTQ FDT ₁₆ (our)	205.0	5.475	50.13

- 사용 Metric 별 성능 비교 (Quantization)

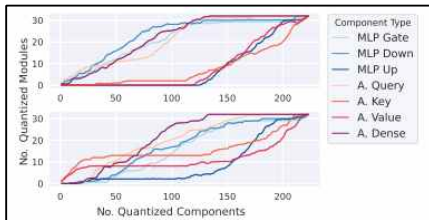
- int 8에서 NLP Benchmark로 평가한 지표를 제외한 나머지 부문에서 FDT가 타 Metrics를 능가함

Experiments - Result B



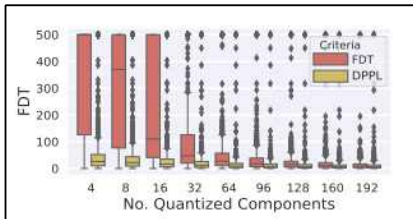
- FDT를 기반으로 하여 평균 75%의 Sparsification을 진행했을 때 각 Component별 Sparsity 정도
- 일부 Layer의 Attention Component가 90% 이상이 Pruning된 것을 확인할 수 있음

Experiments - Result B



- FDT (위) vs PPL (아래) Component Type에 따른 Quantized Module의 갯수
 - FDT를 기반으로 했을 때 Attention의 Key와 Value에 해당하는 Component가 비교적 나중에 Quantization이 이루어진 것을 확인할 수 있음

Experiments - Result B



- GPTQ에서 FDT와 DPPL의 FDT값 비교
 - FDT기반 Quantization의 FDT값 분산이 커서 Component별 중요도에 따른 top-k Component를 파악하기 용이함

Conclusion

- Pros
 - 경량화 모델의 성능을 향상시킴
 - 경량화의 정도를 증가
 - 개별 Component의 중요도를 파악했기 때문
- Cons
 - 다양한 Sparsification 기법을 실험해보지 않았음
 - 영어로 훈련된 모델로만 실험
 - 한국어는 비교적 단어의 배치가 자유로운데, FDT값을 임의로 조정한다면 어떤 결과가 발생할지?

감사합니다.

Q&A

Appendix A

Algorithm 1 Iteration of pruning algorithm

input: $F, step$ $\triangleright$ current model, target sparsity

$fdt_sparse_map \leftarrow \{\}$

for $c_i \in \text{Components}$ **do**

$fdt_sparse_map[c_i] \leftarrow [100,$
 $\mathcal{M}_{FDT_{75}}(F, F^{c_i+step/2}),$
 $\mathcal{M}_{FDT_{75}}(F, F^{c_i+step+step/2}), 0]$

$\triangleright$ FDT values measured for added sparsities of
0, $step/2$, $step + step/2$, 100% to
component c_i on model F . The maximal
FDT value measured is 100.

end for

$f \leftarrow 100$ $\triangleright$ iteratively decrease from maximum

$s \leftarrow 0$ $\triangleright$ track current added sparsity

$comp_sparse_map \leftarrow \{\}$

while $s \leq step$ and $f \geq 0$ **do**

for $c_i \in \text{Components}$ **do**

$comp_sparse_map[c_i] \leftarrow$
 $lin_interpol(fdt_sparse_map[c_i], f)$

end for

$s \leftarrow weighted_mean(comp_sparse_map)$

$f \leftarrow f - 1$

end while

$F' \leftarrow F$ pruned by $comp_sparse_map$

output: F'

Appendix B

