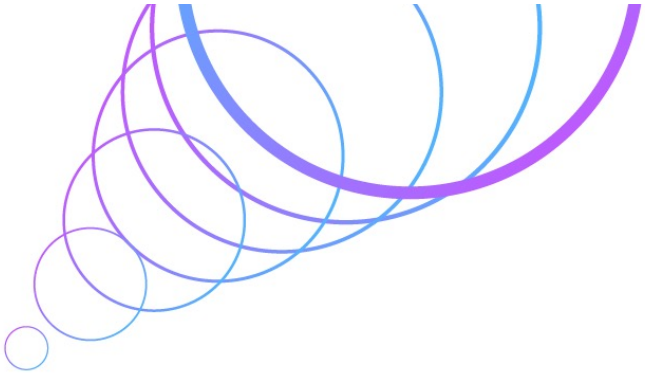


DEVIEW 2021

대규모 자연어 처리 모델 서빙 경험기

김진웅 NAVER, Search CIC
jinwoong.kim@navercorp.com

Serving Food

DEVIEW 2021

음식이 언제 나오는가?
음식이 뜨거운가? 차가운가?
음식의 종류는?
음식의 크기는?
…

https://miro.medium.com/max/1280/1*127vfGSUUEEYsc3xu5_Wyg.jpeg

Serving Model

모델이 **언제 완성**되는지?
모델의 **크기**는 어떠한가?
모델의 **품질**은 적절한가?
모델의 **인퍼런스 속도**는 어떤가?
…

엔지니어링 관점에서의 모델 이해

https://miro.medium.com/max/1280/1*127vfGSUUEEYsc3xu5_Wyg.jpeg

CONTENTS

DEVIEW 2021

1. 대규모 자연어 처리 모델의 등장

1.1 자연어 처리 모델

DEVIEW 2021

자연어 처리(**NLP**)란?

- 우리가 일상에서 사용하는 언어(자연어)를 처리하는 것

자연어 처리 모델을 이용한 분야

- 검색어 추천, 번역, 스팸 분류, 챗봇 등

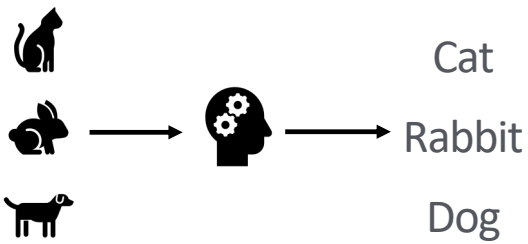
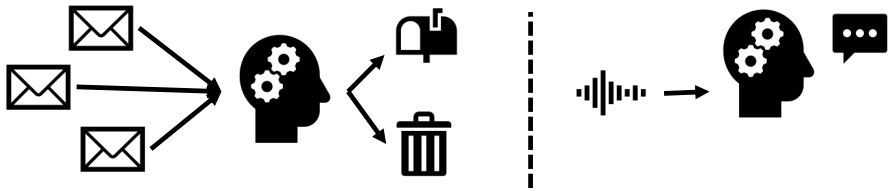


Image Processing



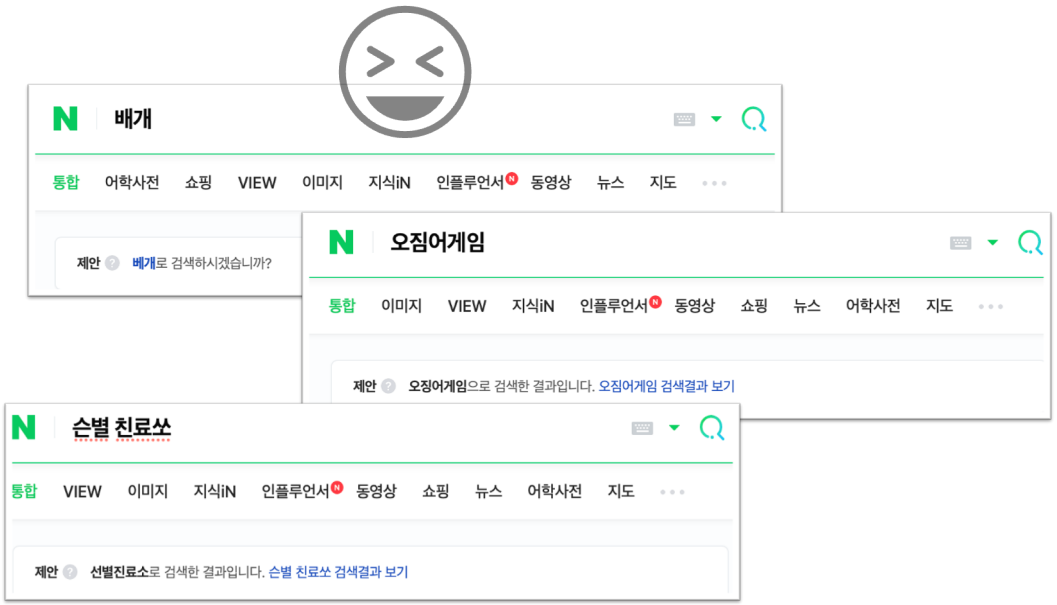
Natural Language Processing

1.2 자연어 처리 in 네이버 검색

DEVIEW 2021

자연어 처리 성능을 올리는 것은 중요 과제 중 하나

- 예) 검색어 제안



null 쿼리의 예

1.3 대규모 자연어 처리 모델의 등장

DEVIEW 2021

2020년 5월에 175B 스케일의 GPT3 등장 (OpenAI)

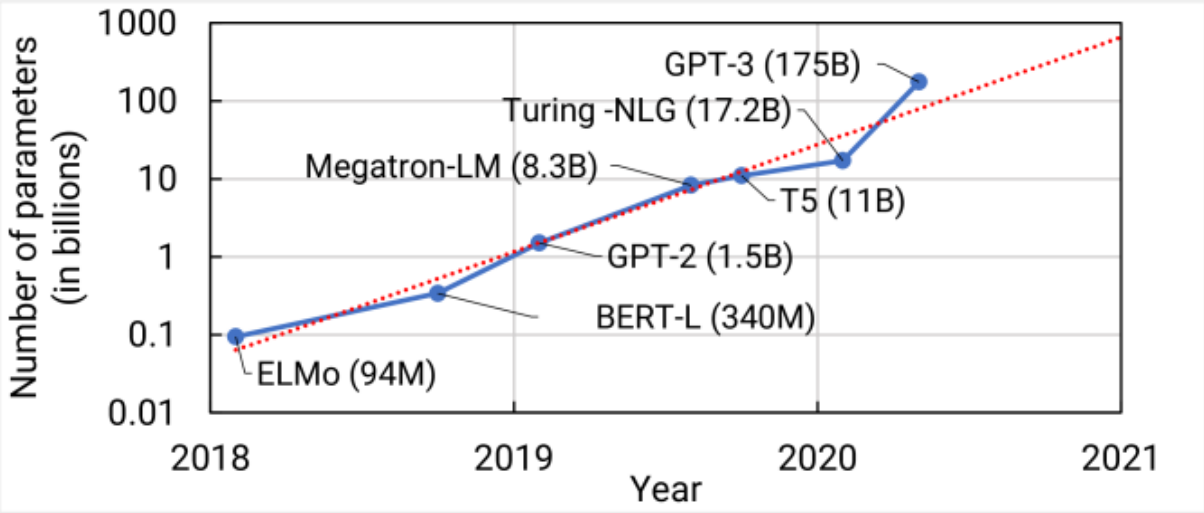


image from https://developer-blogs.nvidia.com/wp-content/uploads/2021/03/NLP_model_size-1-1024x435.png

1.4 GPT3란?

쉽게 말해서, 자연어 생성 모델 입니다

- 예) 챗봇

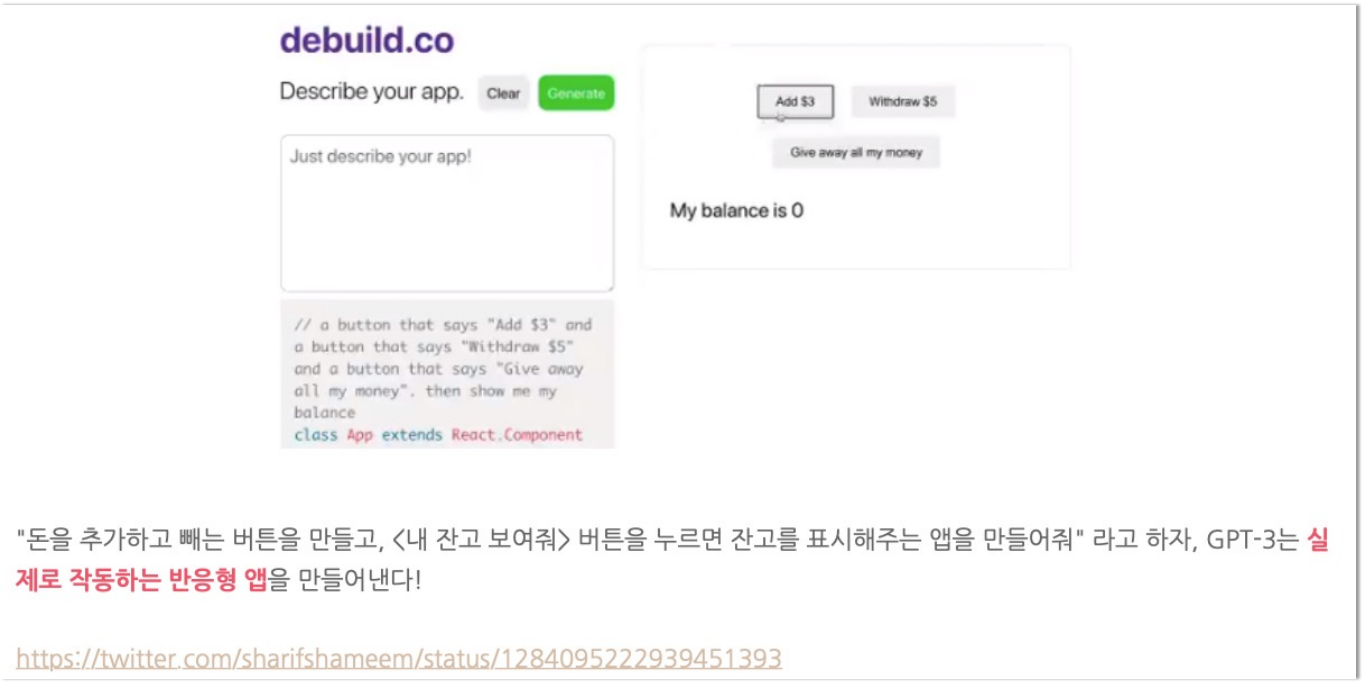
이미 있는 것 아닌가요?

- 맞습니다. 하지만, 엄청 성능이 좋습니다! 엄청 큽니다

- 스케일이 커지면서 기존에 풀 수 없었던 문제들이 풀리기 시작!

이런 것도 가능합니다

1.5.1 반응형 앱 코딩

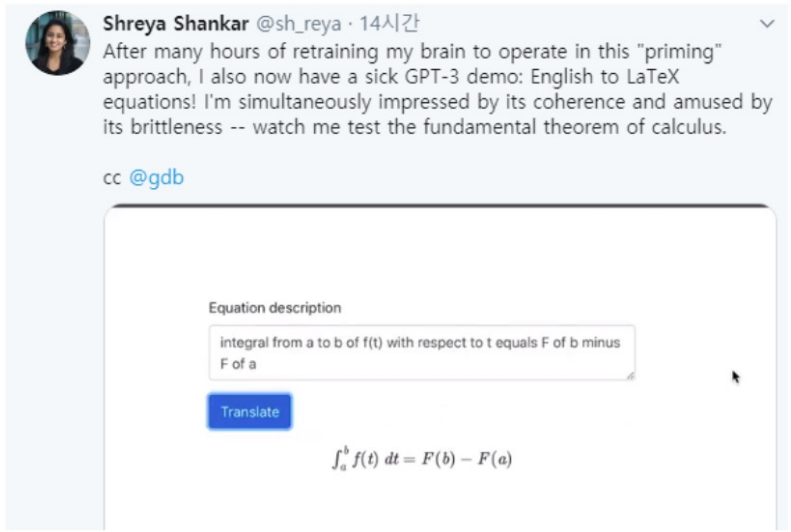


"돈을 추가하고 빼는 버튼을 만들고, <내 잔고 보여줘> 버튼을 누르면 잔고를 표시해주는 앱을 만들어줘" 라고 하자, GPT-3는 **실제로 작동하는 반응형 앱**을 만들어낸다!

https://twitter.com/sharifshameem/status/1284095222939451393

1.5.2 자연어 수식을 LaTeX로 변환

DEVIEW 2021

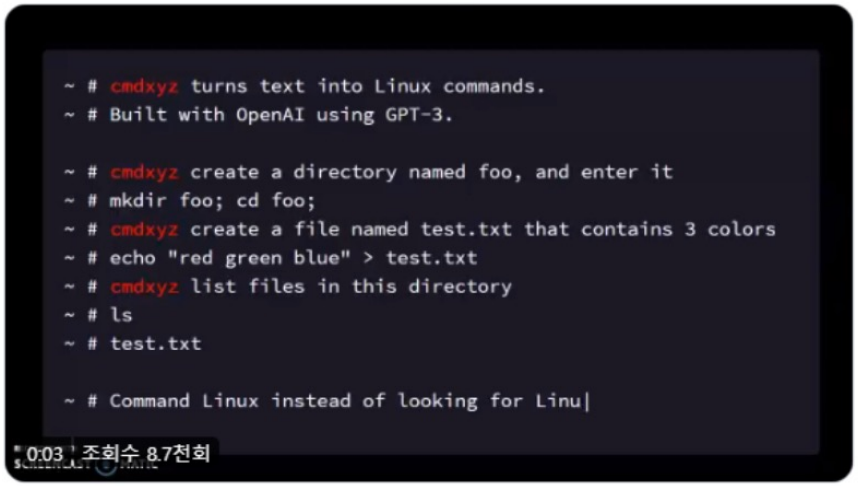


수식에서 처리할 내용을 영어로 입력하자 LaTex 코드가 출력된다. 놀라운 점은 이 데모가 fine-tuning 없이 **GPT-3가 가진 few-shot extrapolation 능력**을 이용해 만든 것이라는 점. ~~아제 LaTex 구글링 없이 논문 쓰는 Codeless 세상이...??~~

https://twitter.com/sh_reya/status/1284746918959239168?s=20&fbclid=IwAR28gSwqae8ZNFxRbTUuv4K6XoQ4wWwLBbnBqIxS0sian21QoZdOE3g3qbQ

1.5.3 리눅스 커널 코딩

DEVIEW 2021

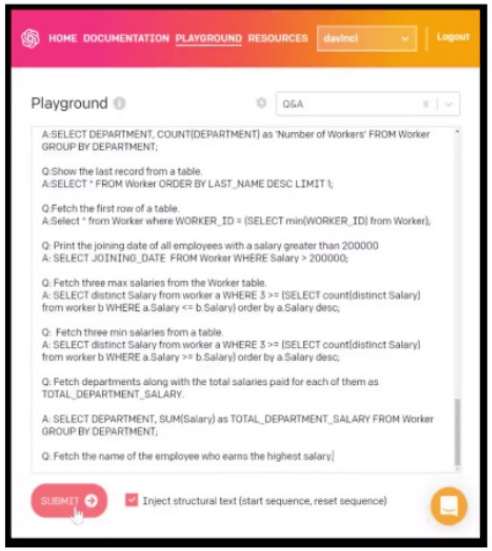


깃헙을 학습한 GPT-2가 파이썬 코딩을 하는 데모에 이어 리눅스 명령어를 만들어내는 GPT-3 데모 발표!

https://twitter.com/super3/status/1284567835386294273

1.5.4 SQL 쿼리 생성기

DEVIEW 2021



few-shot 셋팅으로 SQL 쿼리를 생성하고, **예시로 본적도 없는 SUBSTR과 같은 고난이도 쿼리**도 만들어낼 수 있었다.

https://twitter.com/aquariusacquah/status/1284706786247880705?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1284706786247880705%7Ctwgr%5E&ref_url=https%3A%2F%2Fgpt-3.is%2Fsql%2F

*더 많은 예제 참고 : https://beta.openai.com

1.6 with GPT3

풀기 힘들었던 **기존 문제를 해결**하거나,
혹은 **새로운 형태의 서비스**를 만들 수 있지 않을까?

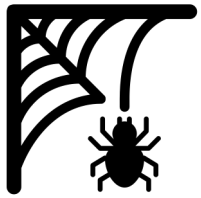
Fixing Problem

New Idea

1.7 But, there’s no free lunch

DEVIEW 2021

With great **Model** there must also come great **headache**



1.8 In this talk,

대규모 자연어 처리 모델(GPT3)을 **실제 서비스에 적용**하며
겪었던 **경험**들, **알게 된 사실**들을 공유하고자 합니다

Focus on,

- 다양한 실험 결과 및 엔지니어 관점 최적화 관련 내용 공유
- NLP 지식이 없어 헤맸던 내용 정리

Not this talk,

- 학습 관련 내용 (HyperClova 논문 참고*)

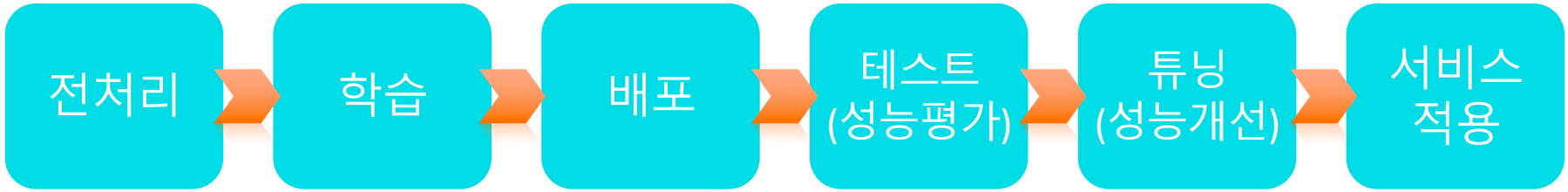
*What Changes Can Large-scale Language Models Bring? Intensive Study on HyperCLOVA: Billions-scale Korean Generative Pretrained Transformers

DEVIEW 2021

2. 자동화 파이프라인 구축

2.1 전처리/배포 자동화

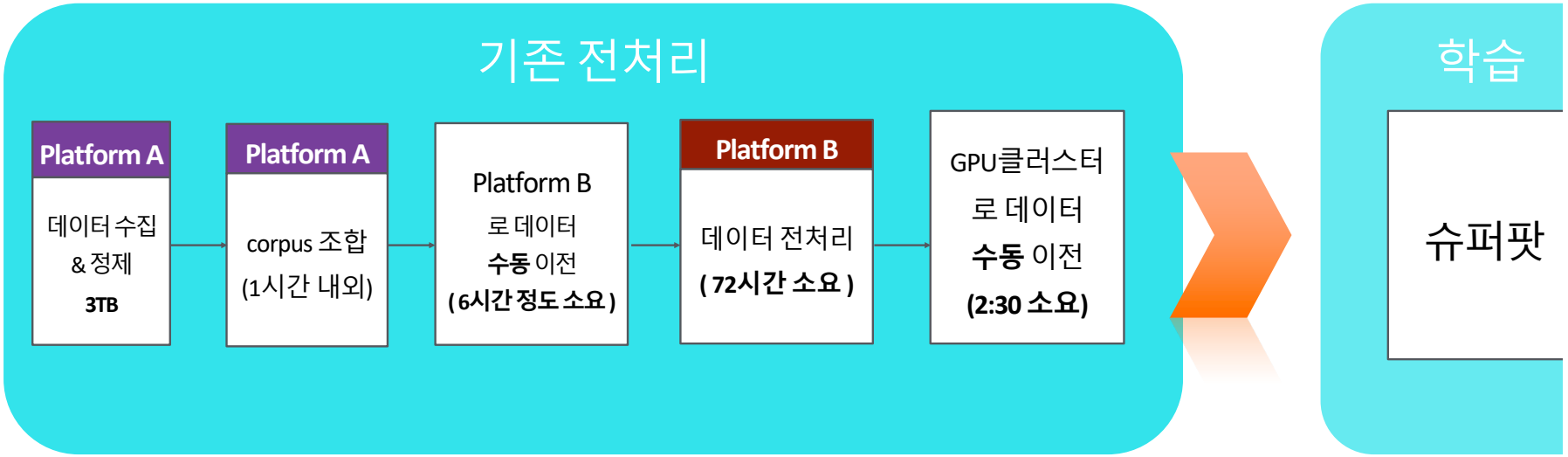
실제 서비스에 적용하기 전까지 적게는 수십, 많게는 수백 번 **반복**
대규모 모델의 경우 전처리와 배포 **자동화가 절실** 하였음



2.2 기존 전처리 방식의 문제

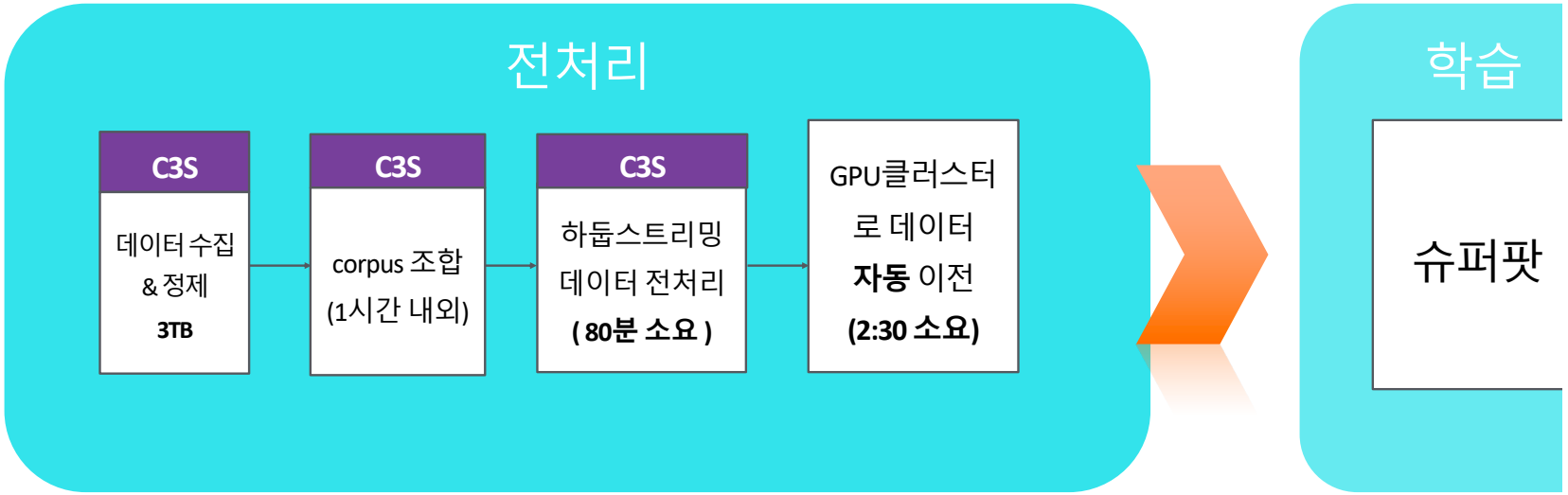
DEVIEW 2021

기존 방식에는 **플랫폼간 이동**과 **수동 작업**이 빈번했음
또한 전처리 최적화가 덜 되어 있어 많은 시간을 필요로 했음



2.3 기존 전처리 방식의 개선

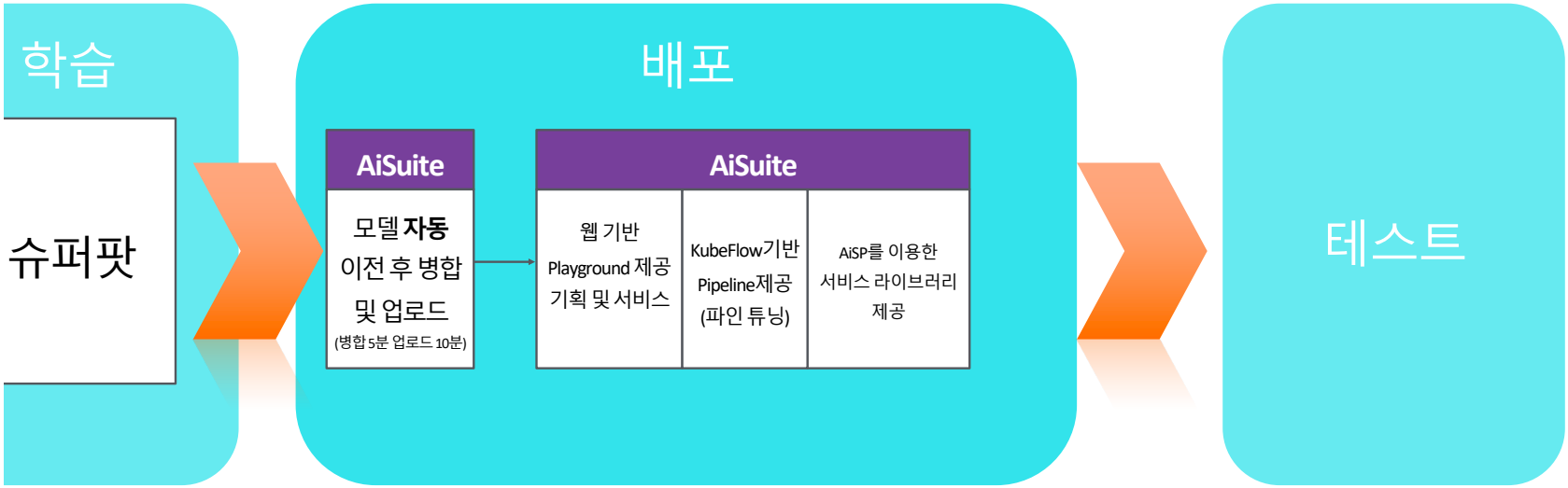
단일 플랫폼(C3S)로 통일하여 **자동화 파이프라인 구축**
C3S의 하둡 스트리밍 병렬화를 이용해 전처리 **시간 58배 단축**



2.4 학습 종료 후 배포

학습 중에도 **3시간 마다 1TB(82B 모델 기준) 모델 저장**
수시로 가져와서 성능 비교 필요하기 때문에 자동화

AiSuite*는 Kubeflow기반의 사내 머신 러닝 플랫폼



*Deview 2021 'AiSuite : Kubeflow를 통해 더 나은 AI 모델 서빙과 MLOps 실현하기 – 한완규, 박정욱' 참고

2.5 MLOps + 최적화

MLOps가 중요하다는 건 모두가 아는 사실

대규모 모델 학습의 경우, 전처리 및 데이터/모델 이동에 많은 시간 소요

자동화 뿐만 아니라 **최적화**가 필수!

자동화

+최적화

DEVIEW 2021

3. 성능 평가

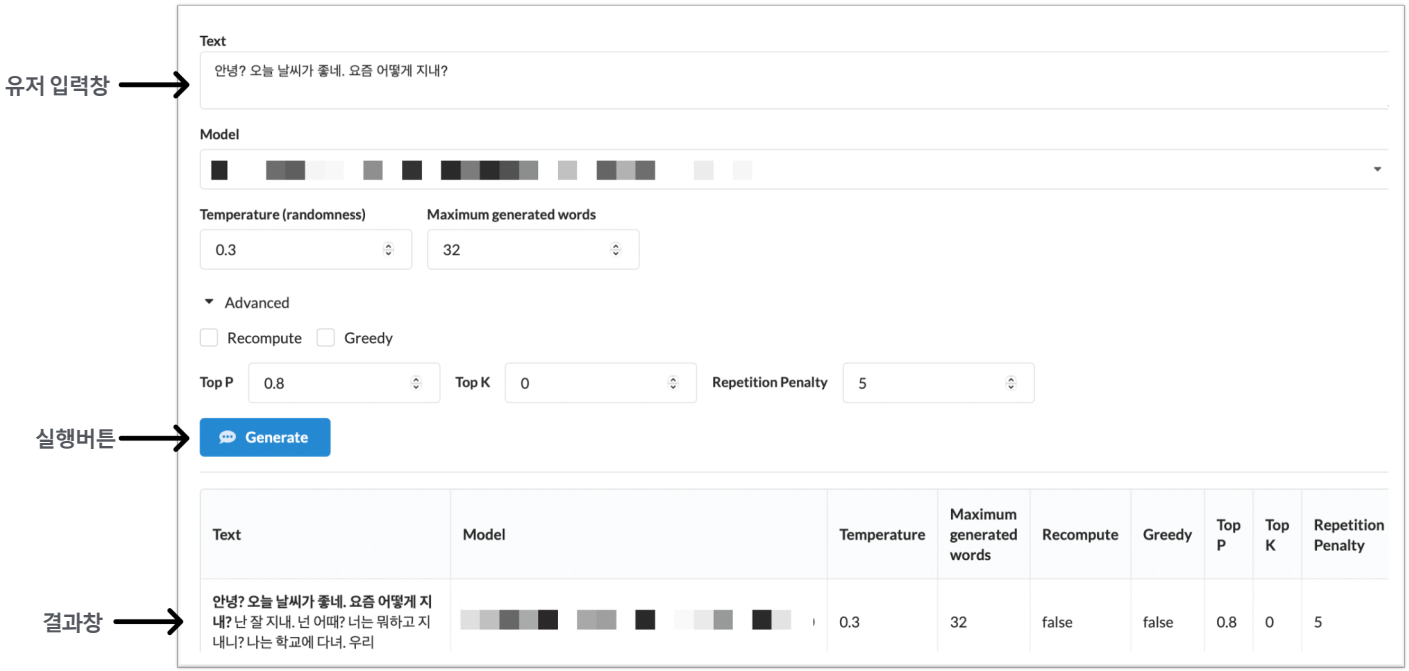
3.1 성능 평가 요소

Performance
- Quality : 서비스 기획자가 직접 평가 셋을 이용해 측정
- Speed : 다양한 모델 사이즈, GPU 및 입력 길이로 실험

3.2 품질 평가를 위한 환경 제공

DEVIEW 2021

학습 시 일반적인 태스크에 대한 품질 평가 완료 (PPL 등)
실제 서비스 적용 시, 유저(기획,개발자)가 자체 평가 셋으로 평가 필요
이를 위해 **REST API 및 Web 인터페이스**로 기획 및 성능 테스트 하도록 플레이그라운드 제공



3.3 인퍼런스 시간 측정

DEVIEW 2021

주요 목적

1) 성능(latency, throughput) 및 메모리를 측정하여 **서비스 구성 시 참고**하기 위해

2) 이후 다양한 사이즈의 모델에 대한 **필요 인프라 자원 및 서비스 스펙 예측**을 위해

3.3.1 실험 요소

DEVIEW 2021

파라미터 — **변경하며 실험**

입력 토큰 길이, 개수 → “밥은 먹었어?”, [“밥은 먹었어?”,”오늘 어땠어?”]

출력 토큰 길이 → 최대 10 단어 생성, 최대 20 단어 생성

GPU타입 → T4, V100

모델 사이즈 → 1B ~ 80B

측정 — **파라미터 변경에 따른 변화량 표기**

단어 생성 시간

GPU 메모리 사용량

회사 내규에 의해 상대 표기 양해 부탁 드립니다.

3.3.2 궁금증

모델 사이즈가 두배가 되면,
  Q. 실행시간이 두배가 될까? → 3.4.7
  Q. GPU 메모리 사용량도 두배가 될까? → 3.4.8
  Q. 품질도 두배가 될까? → 4.3.4
Q. GPU 타입별 성능은 어떻게 다를까? → 4.2.3
…

Q. 입력 쿼리가 길어지면 생성 단어 속도도 느려질까? → 싱글배치
Q. 입력 쿼리가 여러개면 어떻게 될까? → 멀티배치
…

DEVIEW 2021

싱글배치

3.4.1 싱글배치 실험 (1)

입력 : 텍스트를 늘렸을 때, 시간, 메모리

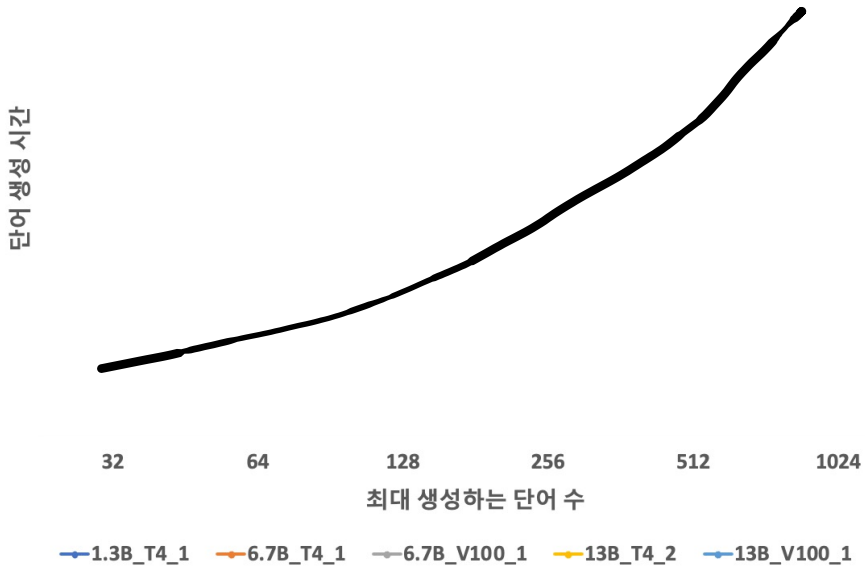
출력 : 최대 생성 단어를 늘렸을 때, 시간, 메모리

모델 : 1.3B, 6.7B, 13B, 39B, 82B

GPU타입 : T4(16GiB), V100(32GiB)

3.4.2 모델 별로 최대 생성 단어 수를 다르게 하여 성능 측정

최대 단어 생성 수가 늘어날수록 수행 속도도 늘어난다고 **예상**

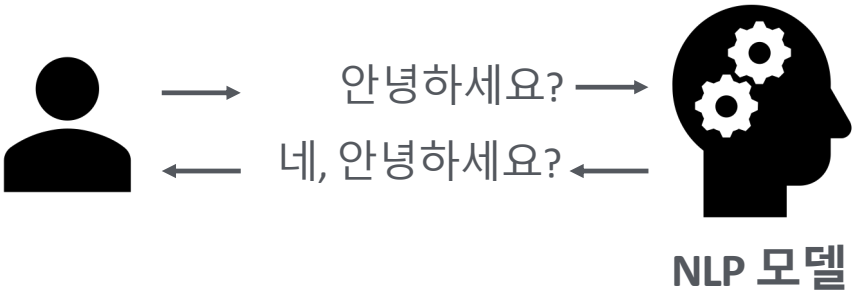


3.4.3 모델 스스로 단어 생성을 멈춘다!

DEVIEW 2021

모델이 스스로 문맥이 종료되었다고 생각하면, EOT (EndOfToken) 토큰을 뱉고 단어 생성을 멈춘다.

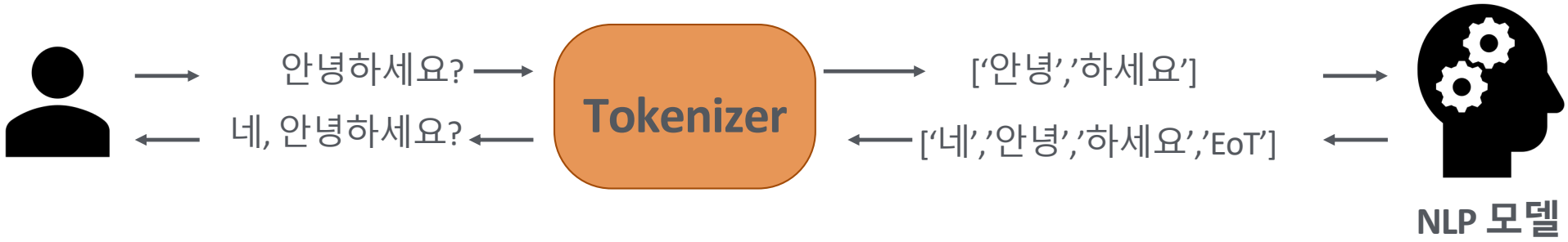
우리 눈에 보이는 모습,



3.4.4 실제 과정 : 토크나이징 필요

DEVIEW 2021

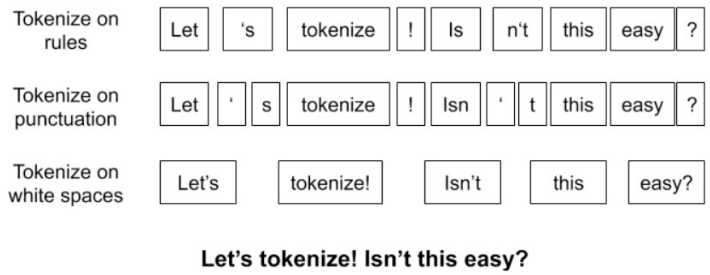
토크나이저 : 텍스트를 토큰 단위로 나누어서 모델에게 한꺼번에 전달
모델 : 입력된 토큰들을 기반으로, 새로운 토큰을 하나씩 생성



3.4.5 토크나이저의 역할

1) Tokenization : 자연어를 특정 기준 단위로 나눔
2) Cleaning : 노이즈 데이터를 제거
3) Normalization : 표현 방법이 다른 단어들을 통합
   ex) 대소문자 통합 등

토크나이저 종류에 따라 방식은 천차만별이며, 이에 따른 모델 성능도 천차만별...



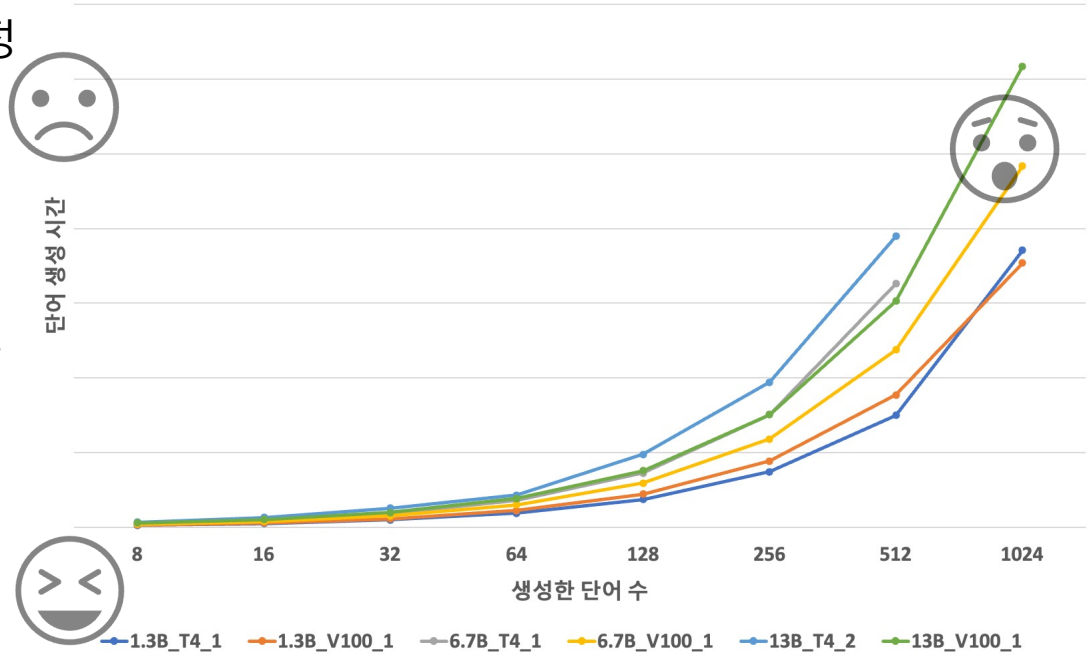
3.4.6 단어 생성에 따른 수행 속도 재측정!

DEVIEW 2021

EOT를 만나도 계속 단어를 생성하도록 수정

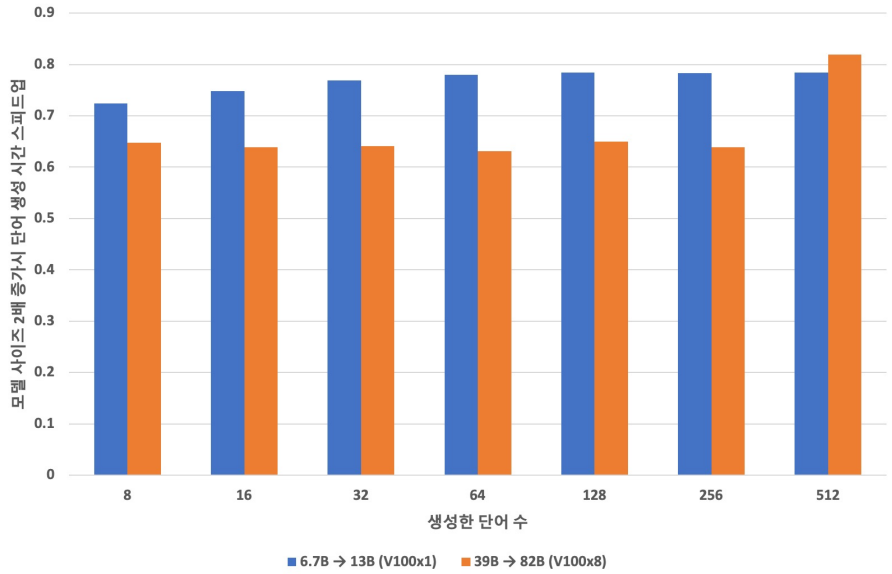
결과적으로 **생성 단어 수에 수행 속도가 비례**하는 모습을 보여줌

흥미로운 점은, 이번엔 실제로 1024 단어를 생성하다 보니 **CUDA OOM이 발생**함



3.4.7 모델 사이즈 증가에 따른 성능 측정

V100 1개) 6.7B 사이즈 모델을 13B로 키웠더니 실행 시간 22~28% 증가
V100 8개) 39B 사이즈 모델을 82B로 키웠더니 실행 시간 20~37% 증가
- GPU간 통신량도 같이 증가해서



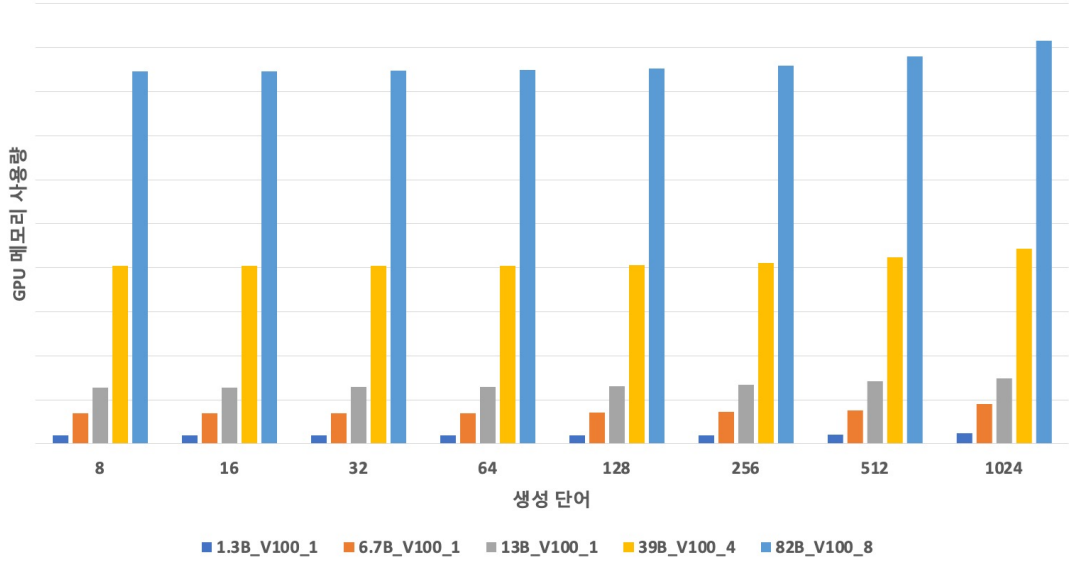
3.4.8 생성 단어 수에 따른 메모리 사용량

DEVIEW 2021

GPU 메모리의 대부분은 모델 로드에 사용

생성 단어 증가당 GPU 메모리 사용량 증가는 약 0.005~0.02% (작은 모델일수록 큼)

- 1.3B 모델의 경우 메모리 사용량 최대 30% 증가, 82B의 경우 8% 증가
- 생성 단어 증가시 늘어나는 메모리 사용량은 엇비슷

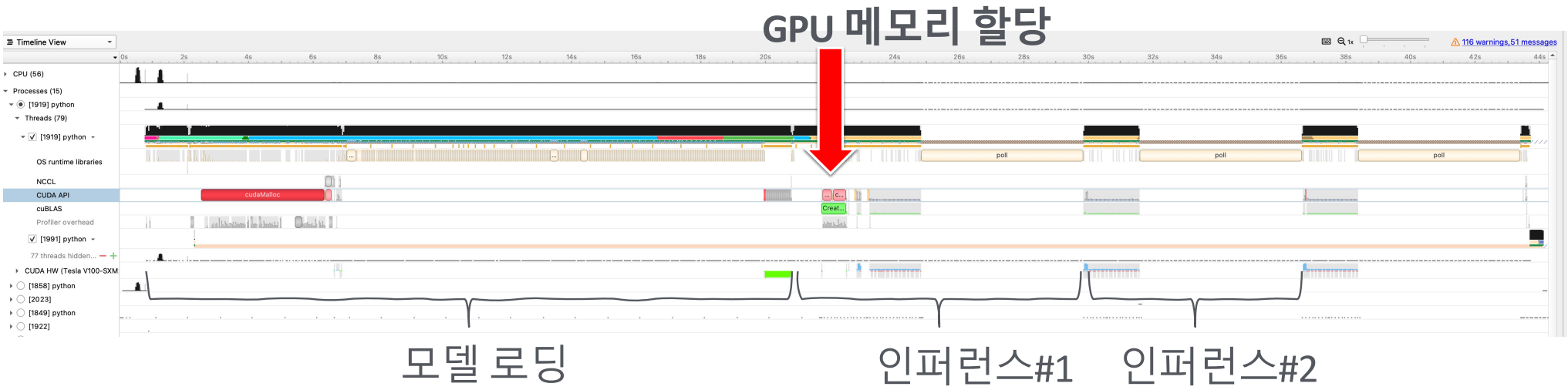


3.4.9 인퍼런스는 워밍업이 필요하다

DEVIEW 2021

처음 인퍼런스 수행 시 **메모리 할당에 많은 시간 소모**

메모리 할당 후 해제를 안하므로, 추후 메모리 할당 오버헤드를 피할 수 있다



3.4.10 싱글배치 실험

DEVIEW 2021

입력 : 텍스트를 늘렸을 때, 시간, 메모리

출력 : 최대 생성 단어를 늘렸을 때, 시간, 메모리

모델 : 1.3B, 6.7B, 13B, 39B, 82B

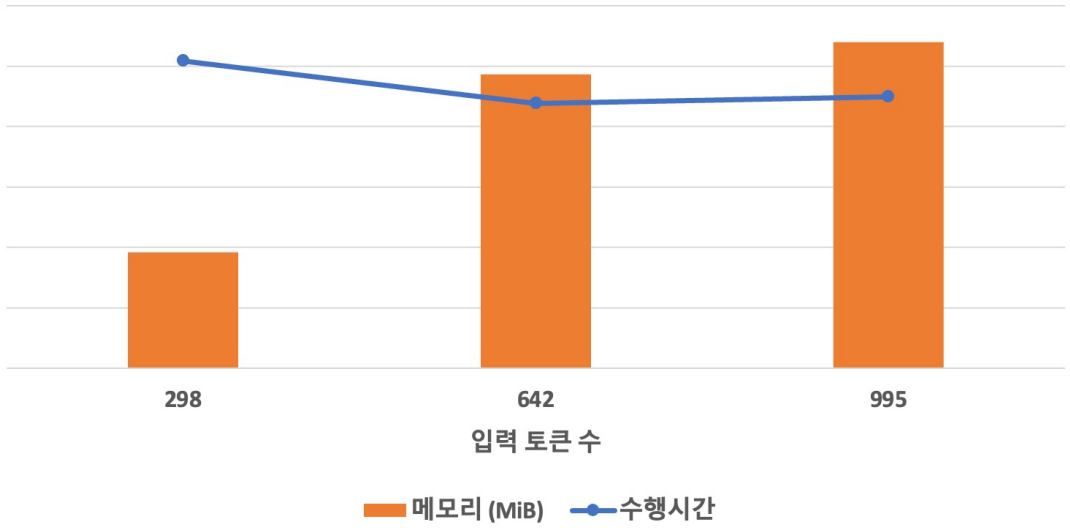
GPU타입 : T4(~16GiB), V100(~32GiB)

3.4.11입력 토큰 수에 따른 성능 측정

DEVIEW 2021

입력 텍스트 길이에 **시간은 비슷, 메모리는 증가**하는 모습을 보여줌

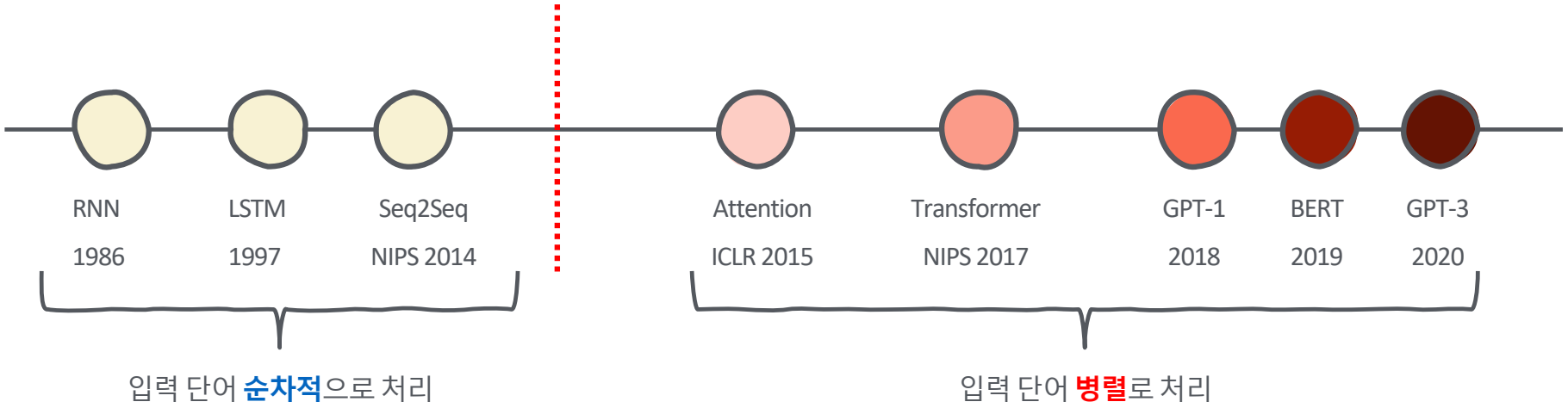
왜 이러는 걸까요?



3.4.12 NLP 모델 계보

DEVIEW 2021

최신 NLP 모델 (Bert, GPT-N) 들은 모두 Transformer 기반
가장 큰 차이는,



멀티배치

3.4.13 멀티배치 실험 요소

입력 : 텍스트를 늘렸을 때, 시간, 메모리

+텍스트를 동시에 두 개 이상 입력,

출력 : 최대 생성 단어를 늘렸을 때, 시간, 메모리

모델 : 1.3B, 6.7B, 13B, 39B, 82B

GPU타입 : T4(~16GiB), V100(~32GiB)

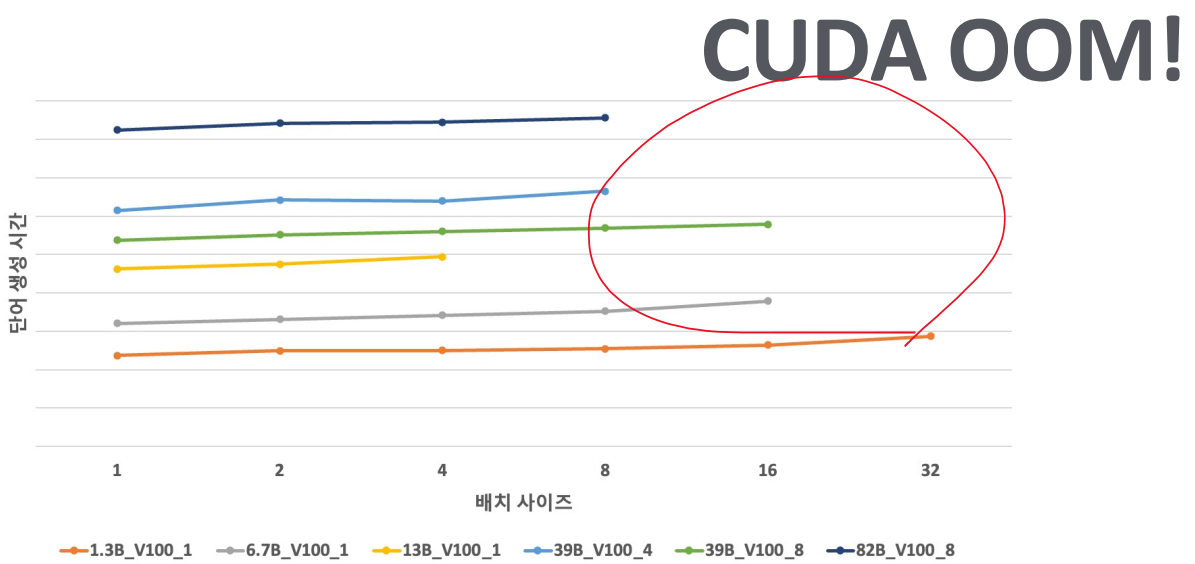
3.4.14 One-shot learning, 128 단어 생성

DEVIEW 2021

입력 토큰 수 103~238 사이로 실험

Latency 저하 거의 없음

메모리 소폭 증가 하였으나 8부터 CUDA OOM발생



3.4.15 Few-shot learning, 32 단어 생성

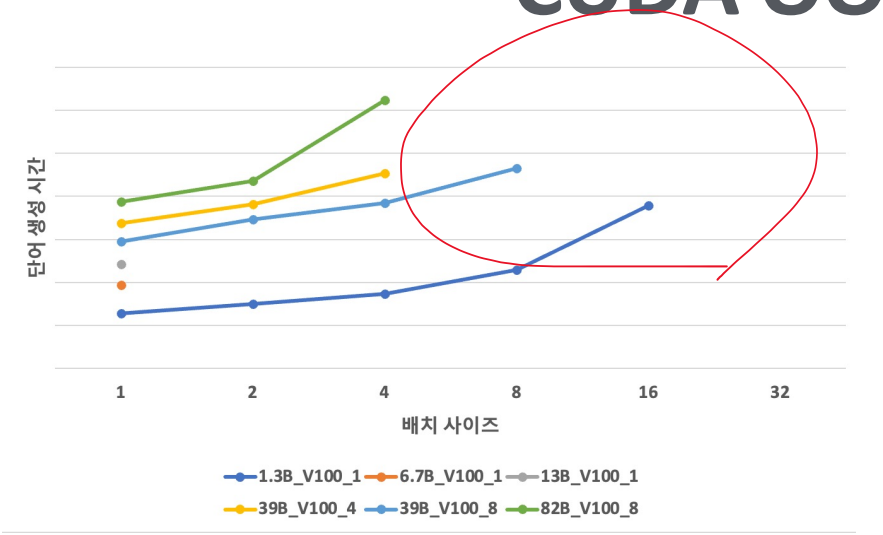
DEVIEW 2021

서비스 타겟으로 실험

품질 위해 예제 최대한 많이 (입력 토큰 956~1024 사이)

Latency도 늘어나지만, 메모리가 문제!

CUDA OOM!



	1.3B_V100_1	6.7B_V100_1	13B_V100_1	39B_V100_4	39B_V100_8	82B_V100_8
0	3	27	25	19	10	20
1	5	31	30	23	16	23
2	6	OOM	OOM	26	20	30
4	10	OOM	OOM	OOM	29	OOM
8	17	OOM	OOM	OOM	OOM	OOM
16	28	OOM	OOM	OOM	OOM	OOM
32	OOM	OOM	OOM	OOM	OOM	OOM

3.4.16 싱글,멀티 배치 요약

DEVIEW 2021

Singlebatch

- 수행 시간은 생성 단어 수에 비례. 조기 종료 조건을 잘 정하는 것은 매우 중요!
- 모델 사이즈가 2배가 된다고 수행속도가 2배가 되지는 않음 (평균1.25배 증가)
- 입력 질의(토큰)를 늘리면 수행 시간은 소폭 상승하나 GPU 메모리는 비례하여 증가

Multibatch

- 퓨샷 개수 N과 배치 사이즈 B에 N*B로 메모리는 비례하여 증가
- 6.7B, 13B 모델의 최대 길이 쿼리의 경우 경우 V100 1개 이용시 메모리 부족으로 최대 배치 사이즈가 1임

DEVIEW 2021

실제 서비스 쿼리를 이용한 성능 검증

3.5 실험 환경

모델 : 한국어 39B

장비 : V100 x 4

실제 서비스에 사용할 테스트 샘플을 받아 실험

	샘플 수	샘플 당 토큰 수
테스트#1	1888	365~410
테스트#2	1007	465~1075

3.5.1 단어 생성에 영향을 주는 요소들

DEVIEW 2021

GPT3로 단어 생성시, 설정 값을 통해 어떠한 단어를 생성할 지 튜닝 할 수 있습니다

Greedy

Sampling

3.5.1 단어 생성에 영향을 주는 요소들

DEVIEW 2021

Greedy : 가장 확률 높은 것 선택

Repetition_penalty : Greedy 사용시 같이 사용. 동의어 반복 제어

Greedy의 경우 가장 기본적인 단어 생성 방법이나, 생성 단어가 길어짐에 따라 **동의어 반복**이나 생성 **문장이 산**으로 가는 경향이 있음

3.5.1 단어 생성에 영향을 주는 요소들

DEVIEW 2021

Sampling

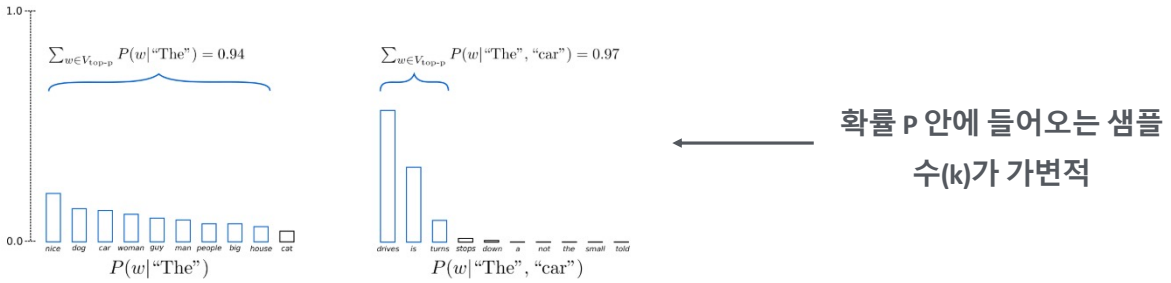
확률 분포에 따른 단어 생성

Temperature : 낮은 확률의 토큰이 지나치게 샘플링 되는 상황 조율. 0에 가까울 수록 greedy 결과와 비슷

Top K : k개를 후보로 두고 샘플링

단점 : 낮은 확률의 단어도 k 개 이내라서 샘플링 될 수 있음

Top P : 확률 P를 넘지 않는 집합 안에서 샘플링



확률 P 안에 들어오는 샘플 수(k)가 가변적

3.6 테스트#1

퓨샷 샘플 수 : 1888

샘플 당 토큰 수 : 365~410

- 테스트#1의 경우 자연스러운 문장 생성(동의어 반복 제어)을 위해, repetiton_penalty를 5로 줌
- 다만, 인퍼런스 성능을 빠르게 하기 위해 temperature 값을 0으로 주었습니다

전달받은 파라미터

파라미터	수치
temperature	0.0
top_p	0.8
out_seq_length	16
repetition_penalty	5
greedy	True
stop_sequences	['\n']

그 외 파라미터

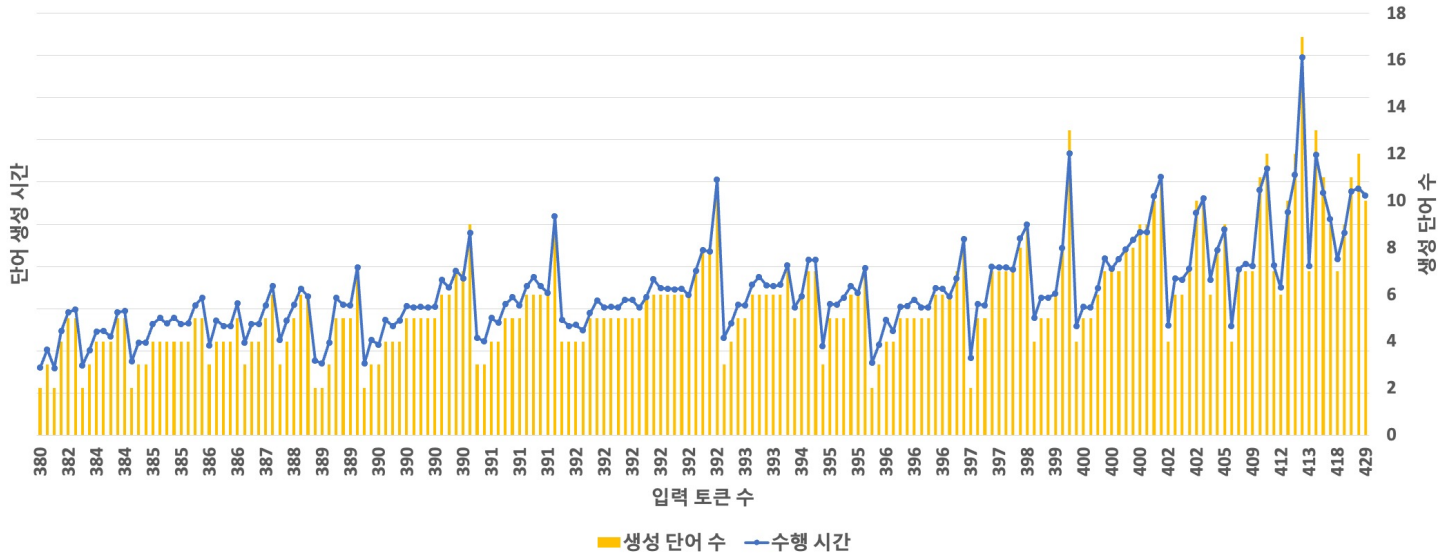
파라미터	수치
recompute	False
top_k	0
start	''
restart	""

테스트#1

3.6.1 싱글 배치 실험

DEVIEW 2021

예상대로 입력 토큰 수(가로)가 아닌, **생성 단어 수(노란 막대)에 수행 시간이 비례**하는 모습 보여줌

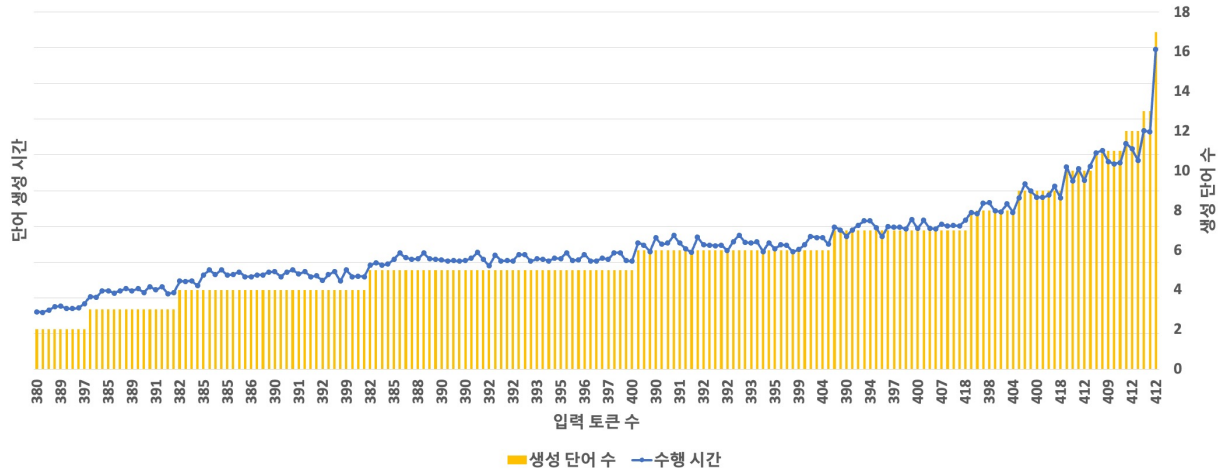


테스트#1

3.6.2 단어 생성 시간에 영향을 미치는 요소 확인

DEVIEW 2021

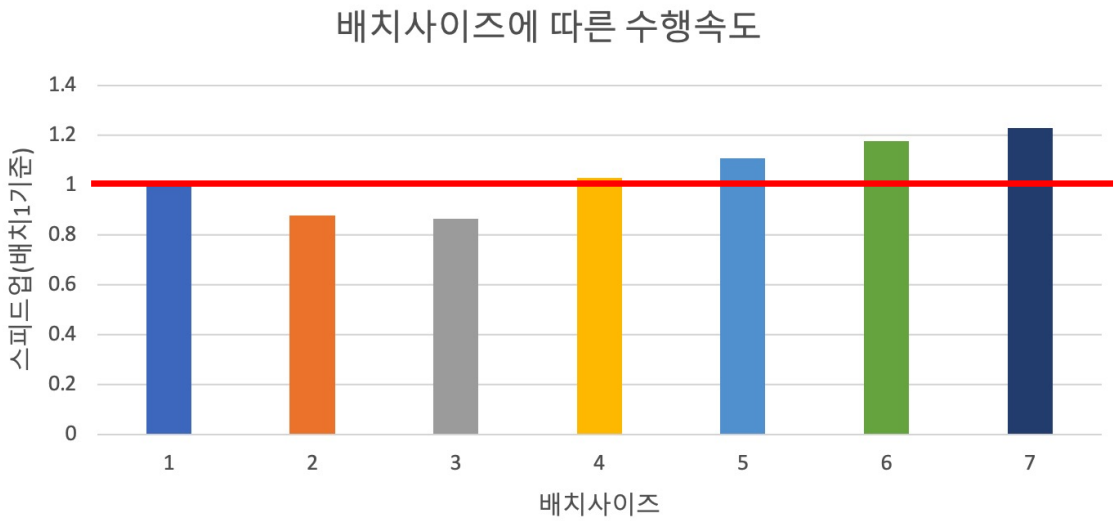
생성 단어 순으로 x 축 정렬하는 경우 더욱 또렷이 보임
단, 생성 단어가 8이하인 경우 다른 오버헤드에 의한 노이즈를 볼 수 있음



테스트#1

3.6.3 멀티배치, 처리량 실험

기존 실험에서처럼 배치사이즈를 늘릴 수록 처리량도 늘어난다고 생각했으나,
예상과 다르게 줄어들거나 소폭만 증가



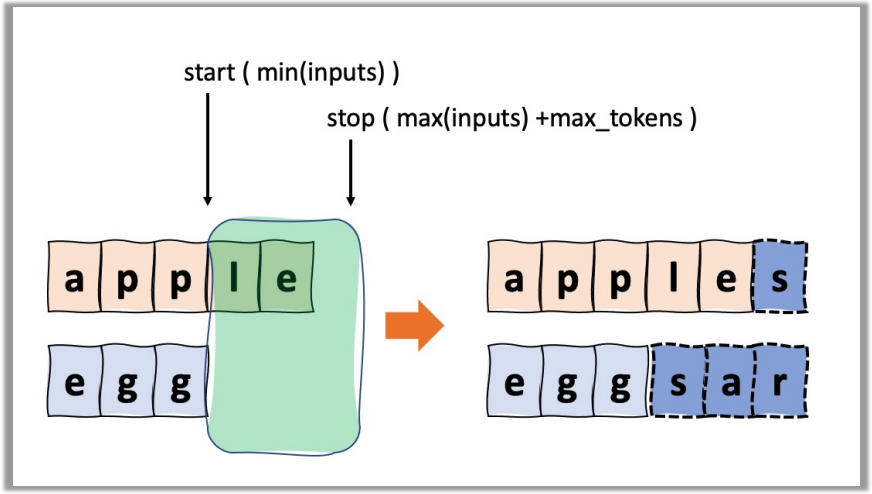
테스트#1

3.6.4 배치 인퍼런스 동작 방식의 이해 필요

트랜스포머의 경우 각각 따로 인퍼런스를 하는게 아니라, 묶은 후 하나씩 생성
하나의 배치 안에 글자수가 다르면 성능 저하를 일으킴

제일 짧은 입력부터 단어 생성해서,
제일 긴 입력 단어 생성이 마무리 될 때까지 진행

예제의 한 글자가 실제론 한 단어 가정
예) 1개의 글자만 생성하도록 수행,
egg의 경우 eggs, apple의 경우 apples가 생성되어야 하나,
eggsar, apples 가 되는 상황



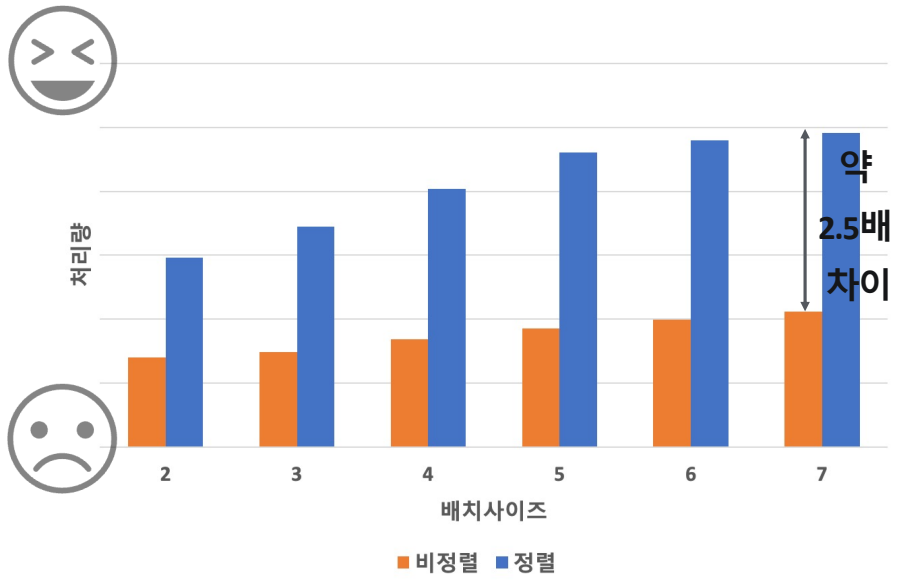
*관련 이슈 : https://github.com/huggingface/transformers/issues/3021

테스트#1

3.6.5 정렬 후 다시 처리량 측정

DEVIEW 2021

입력 길이가 비슷한 쿼리들끼리 클러스터링 후 재측정



3.7 테스트#2

퓨샷 샘플 수 : 1007

샘플 당 토큰 수 : 465~1075

- 샘플당 토큰 수의 차이가 커서, 멀티배치 수행 시 정렬에 따른 성능 편차 심할 것이라 예상
- 테스트#1과 마찬가지로 동의어 반복 제어를 위해 repetition_penalty를 3으로 줌
- out_seq_length (최대 단어 생성 수)가 50이기 때문에 stop_sequence에 따른 성능 차이 심할 것으로 예상

전달받은 파라미터

파라미터	값
temperature	0.0
greedy	False/True (두개 인퍼런스 후 선택)
top_p	0.8
top_k	0
out_seq_length	50
recompute	False
repetition_penalty	3
stop_sequences	['\n']

그 외 파라미터

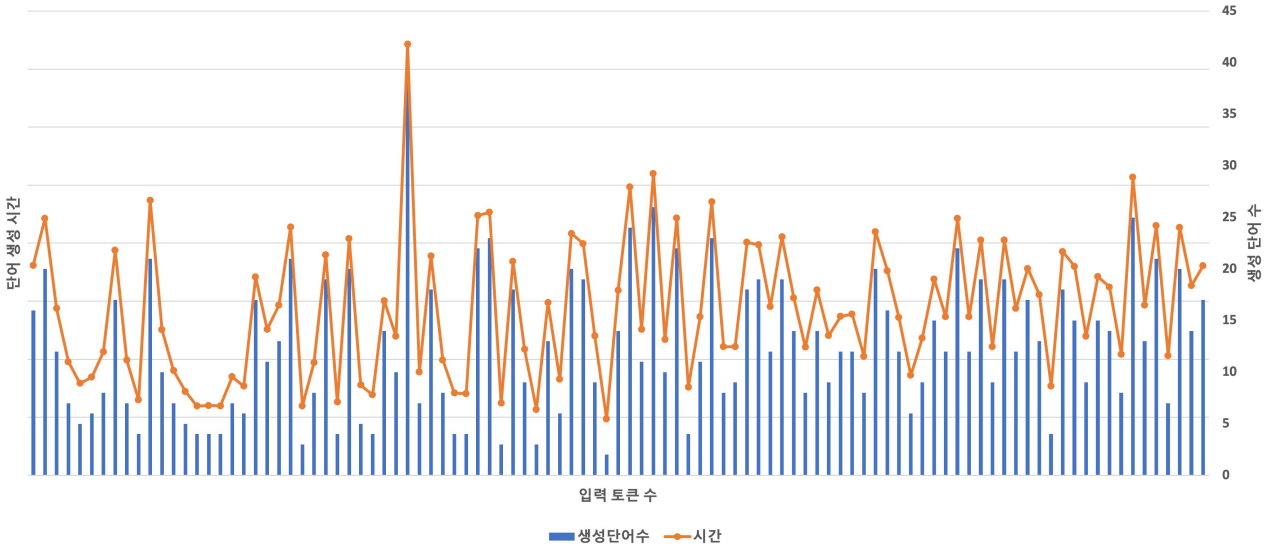
파라미터	값
start	''
restart	""

테스트#2

3.7.1 수행 시간 측정

DEVIEW 2021

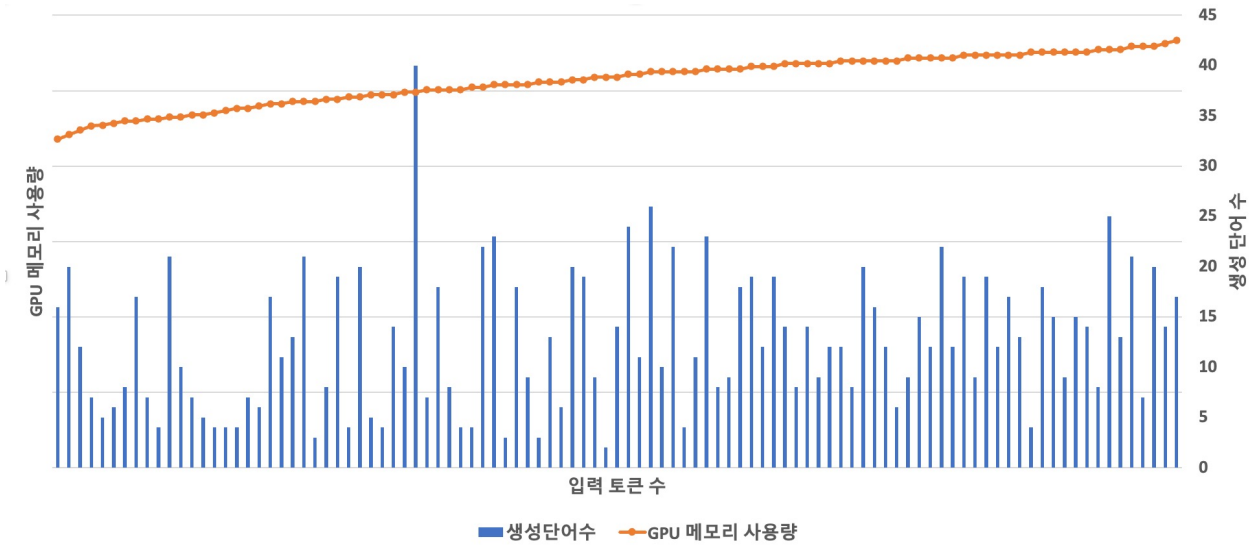
앞서 다른 실험들과 마찬가지로 생성 단어 수에 따른 latency 보여줌



테스트#2

3.7.2 메모리 사용량 측정

생성 단어 수(파란 막대)가 아닌, 입력 토큰 수(가로축 정렬)에 따른 메모리 사용량(주황 라인) 보여줌

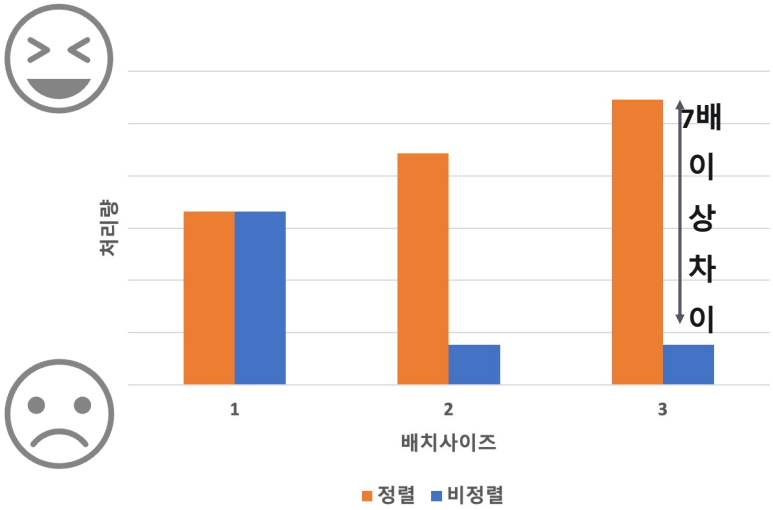


테스트#2

3.7.3 처리량 측정

DEVIEW 2021

입력 간에 토큰 수 차이가 심해서 배치 사이즈 증가시 더더욱
정렬에 따른 성능 편차 심해짐



3.7.4 실제 쿼리 실험 요약

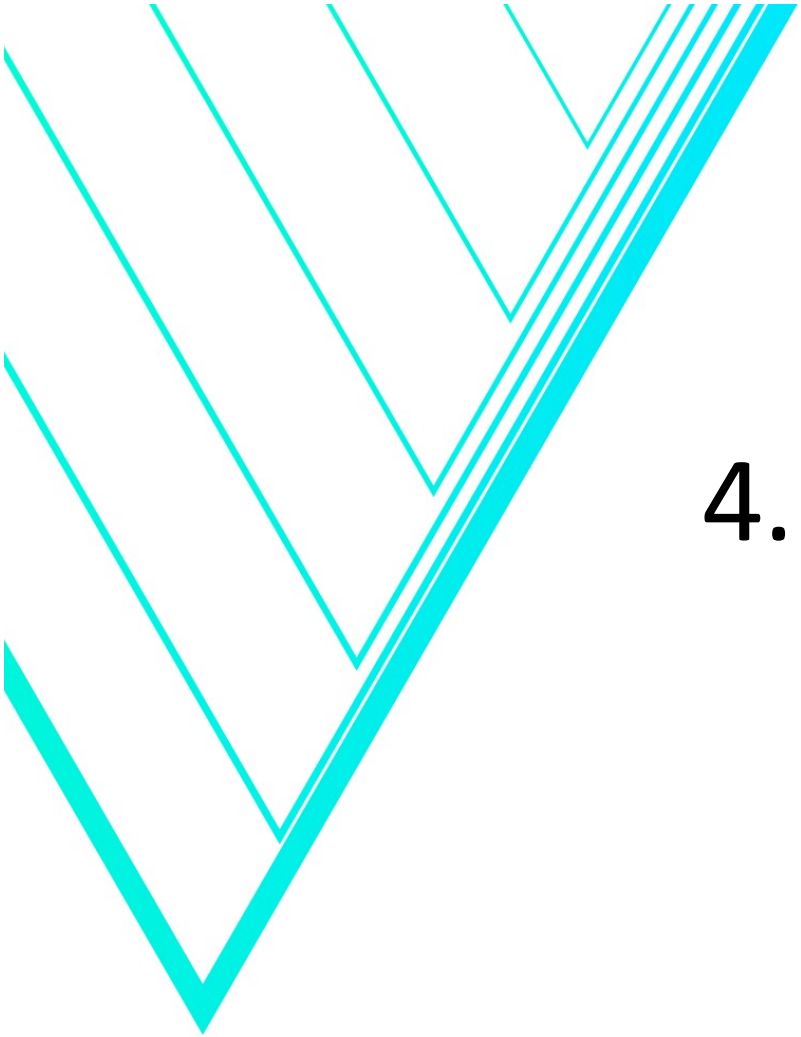
DEVIEW 2021

싱글배치의 경우 예상대로,

- Latency ∝ 실제 단어 생성 수
- GPU 메모리 ∝ (입력 토큰 수*배치 사이즈)

멀티배치의 경우 비슷한 토큰 수의 입력을 클러스터링 하는 것이 필요

- 토큰 수별 길이의 차이가 심하다면 실시간성 서비스 QoS 보장 어려움 있을 것
- 최대 토큰 길이로만 QoS 보장 가능



DEVIEW 2021

4. 성능 개선

4.1 성능 개선 요소

DEVIEW 2021

엔지니어 관점에서의 성능 개선 방안 고찰

- Speed up : SW,HW 개선을 통해 인퍼런스 시간 감소
- Quality up : 파인 튜닝을 통해 단어 생성 품질 향상

DEVIEW 2021

Speed Up

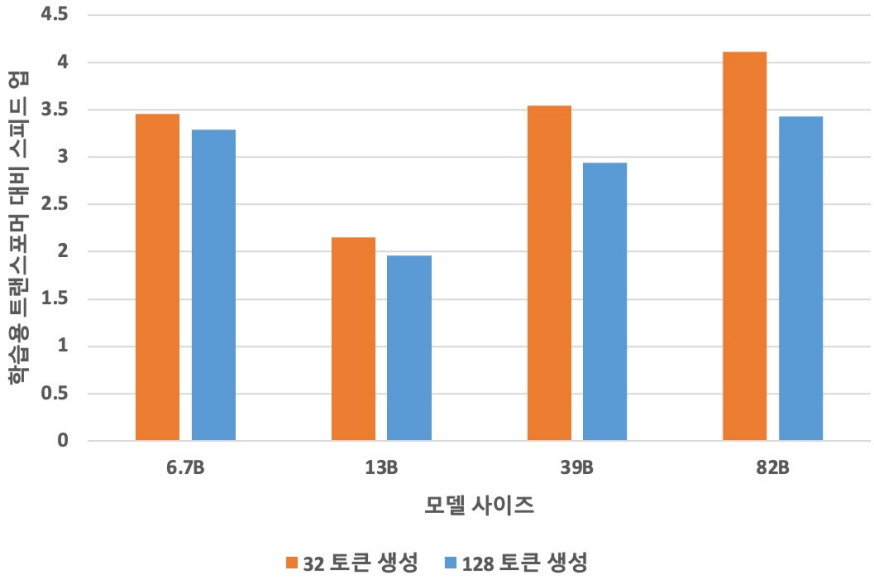
4.2.1 FasterTransformer* 적용

DEVIEW 2021

Nvidia에서 개발한 인퍼런스 전용 트랜스포머

- GPU kernel fusion 등 다양한 GPU 최적화 요소 적용

학습용 트랜스포머 대비 2~4배 성능 향상



https://github.com/NVIDIA/FasterTransformer

4.2.2 새로운 GPU 적용

A100 도입으로 멀티 GPU 환경에서 **1.5 ~ 2배 성능 향상**

- 메모리 대역폭 이랑 NVLink가 성능향상의 중요 요소

GPU 메모리 32GB → 80GB 증가

- 배치 사이즈 2배 이상 증가

GPU 타입 * 개수	39B 모델 스피드 업
V100 * 4	1 (기준)
A100 * 4	1.5
V100 * 8	1.63
A100 * 8	2.34

4.2.3 새로운 GPU 적용 (2)

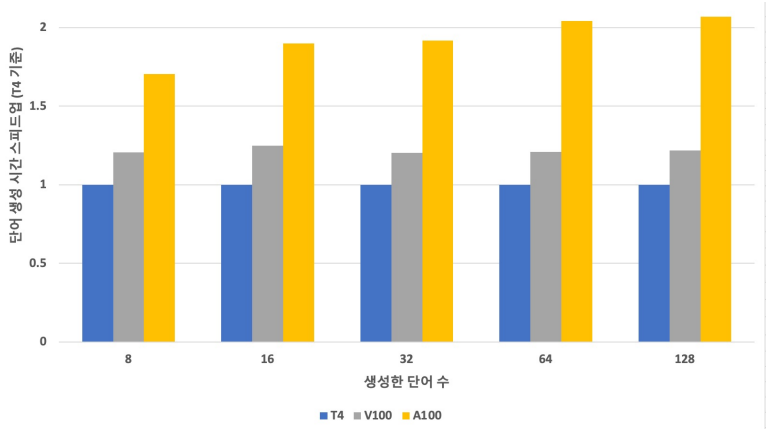
DEVIEW 2021

T4 vs. V100 vs. A100 (모델 사이즈 : 6.7B)

- T4 vs. V100의 경우 20~25% 성능 향상
- T4 vs. A100의 경우 70~106% 성능 향상

T4x2 vs. V100 vs. A100 (모델사이즈 : 13B)

- V100 하나 쓰는 거보다 T4두 개 쓰면 대략 11~22% 느려짐
- A100 은 V100 대비 약 2배 빨랐습니다



DEVIEW 2021

Quality Up

4.3.1 생성 단어 품질을 향상 하는 방법

DEVIEW 2021

파인 튜닝

- 이미 학습된 모델에 새로운 데이터로 추가 학습하는 것
- 전체 학습량보다 현저히 적다고 해도 학습 과정 구현 및 추가적인 자원(GPU, 시간)을 필요로 함

퓨샷 러닝

- 인퍼런스시에 여러 개의 예를 보여줌으로써 추가 학습 없이 보다 나은 결과를 도출해 내는 것

4.3.1 퓨샷러닝

0-shot

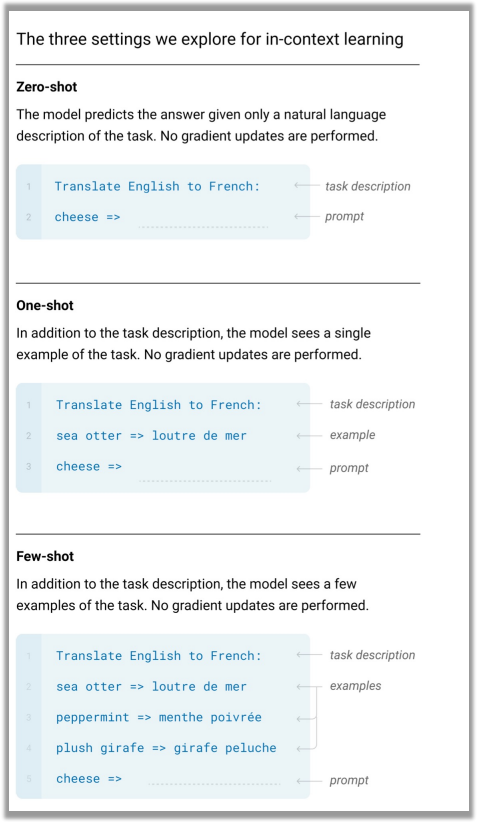
- 예제를 안 보여주는 것

1-shot

- 예제를 하나만 보여주는 것

Few-shot

- 예제를 여러 개 보여 주는 것



The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

4.3.2 퓨샷 러닝의 단점

DEVIEW 2021

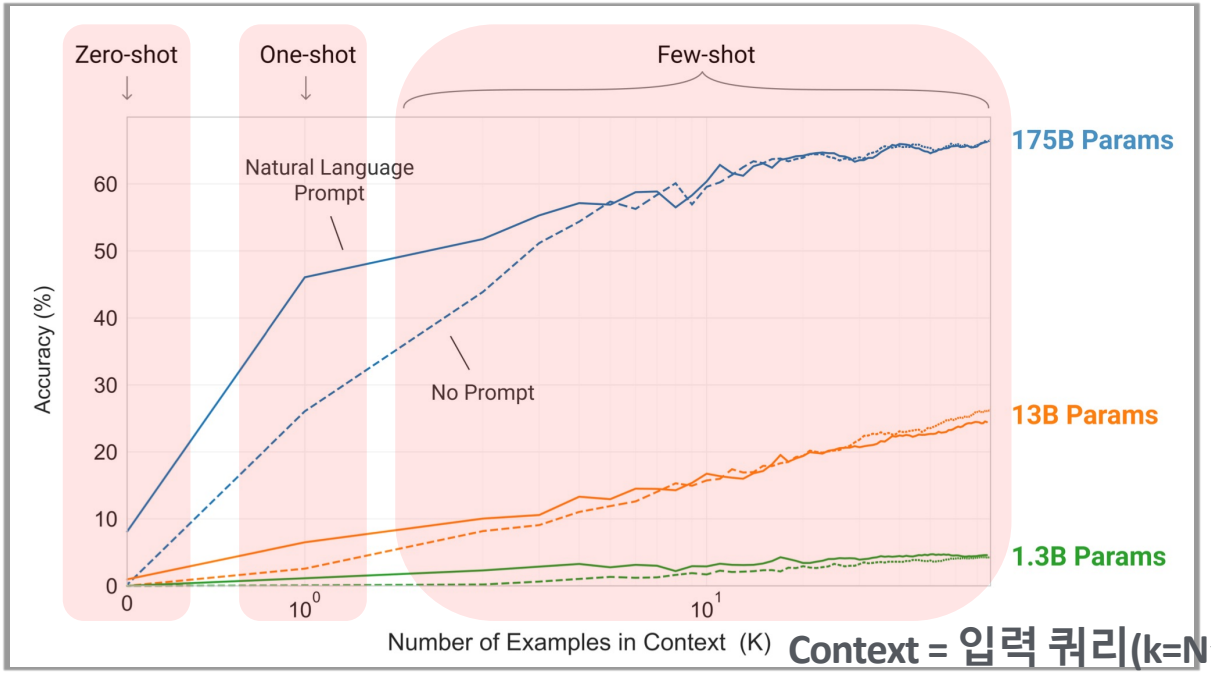
모델이 충분히 크지 않으면 퓨샷의 효과는 낮다

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:      ← task description
2  cheese =>                         ← prompt
```

Natural Language Prompt의 예
-> 테스크에 대한 자연어 설명

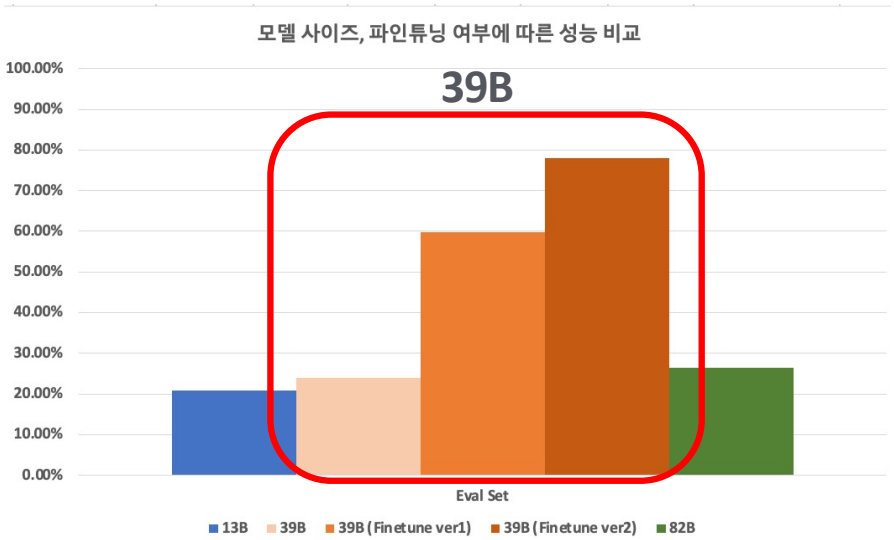


Context = 입력 쿼리(k=N샷)

4.3.3 파인 튜닝에 따른 성능 비교

DEVIEW 2021

자체 서비스 평가 쿼리로 생성 단어 품질 비교, **39B가 82B보다 좋은 결과** 보여줌
서비스에 따라선 파인 튜닝한 작은 모델 사용이 자원 효율, 단어 생성 속도, 품질 모두 유리할 수 있음



4.3.4 파인 튜닝 자동화 및 성능 측정

DEVIEW 2021

하지만, 파인 튜닝은 쉽지 않은 업무
우리는 AiSuite*를 통해 파인 튜닝 자동화하였음
또한 인피니밴드 여부에 따른 성능 차이 및 GPU 종류에 따른 성능 차이도 측정 하였음

GPU	서버 수	총 GPU 수	인피니밴드 유무	Speed Up
V100	2대	16개	X	1 (기준)
A100	2대	16개	X	1.3
V100	2대	16개	O	1.85

*Deview 2021 'AiSuite : Kubeflow를 통해 더 나은 AI 모델 서빙과 MLOps 실현하기 – 한완규, 박정욱' 참고

4.3.5 Bigger is always better?

DEVIEW 2021

필요 자원 : **39B (Good)** < 82B

39B가 인퍼런스에 필요한 GPU **자원 절반**

성능(단어 생성 시간) : **39B (Good)** > 82B

39B가 약 1.2배 빠름

성능(품질) : 39B < **82B (Good)**

82B가 좋으나, 위의 단점을 가릴 정도로 “압도적”으로 좋진 않음

또한, 파인튜닝 진행시 특정 테스크에 대해선 더 좋은 결과 보여줌

DEVIEW 2021

5. 서비스 적용

5.1. we are serving now

서빙중

준비중

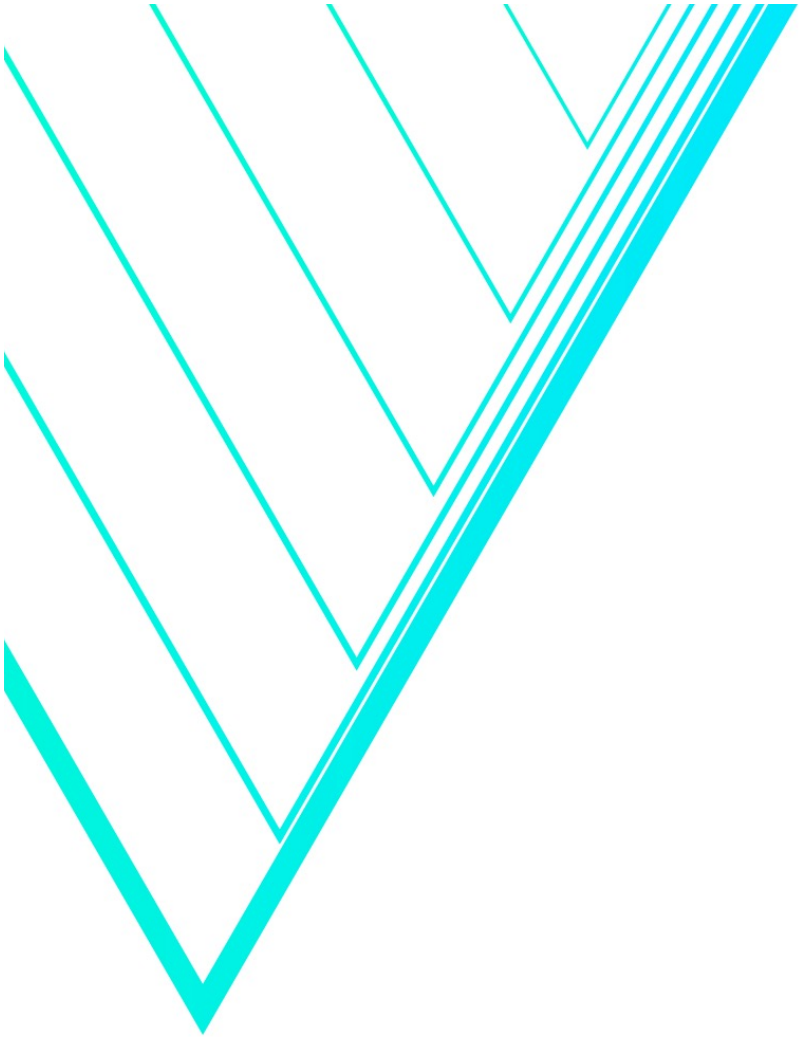
배치성

실시간성

대화형

요약형

제안형



6. 정리

9 lessons we learned (1)

DEVIEW 2021

전처리/배포 파이프라인

1. MLOps는 더 이상 선택이 아닌 **필수**이며, 나아가서 **최적화 필요**

GPT3 Performance Study

2. Latency에 영향을 많이 주는 요소는 생성 단어 수

- 입력 토큰 수와 모델 사이즈에 latency는 sub-linear하게 증가
- 특정 토큰 생성시 조기 종료하는 기법은 latency 감소에 도움 될 수 있음
- 워밍업은 초기 메모리 할당 오버헤드를 감추는데 도움이 됨

3. GPU memory에 영향을 많이 주는 요소는 입력 토큰 수, 배치 사이즈

- 한 배치내 입력 토큰 간의 길이 차이가 심하면 가장 긴 토큰으로 성능 수렴

9 lessons we learned (2)

DEVIEW 2021

GPT3 성능 최적화

5. 인퍼런스 전용 트랜스포머(FasterTransformer) 사용시 2~4배 성능 향상

6. V100 → A100 변경 시 1.5~2배 성능 향상

- 메모리 대역폭 및 NVLink 성능 향상이 많은 영향 줌

7. 파인 튜닝은 때때로 좋은 선택!

정합성 관련

+8. 배치 사이즈, GPU 종류, 트랜스포머에 따라 생성되는 단어가 다를 수 있다

- 내부 matrixMul 연산 및 호출 알고리즘이 달라지기 때문. 품질은 비슷

+9. 모든 인퍼런스는 FP16으로 진행하고 있고 , so far so good!

We’re hiring

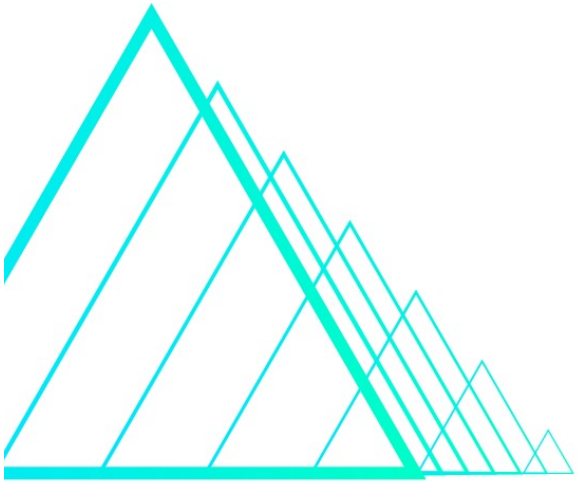
DEVIEW 2021

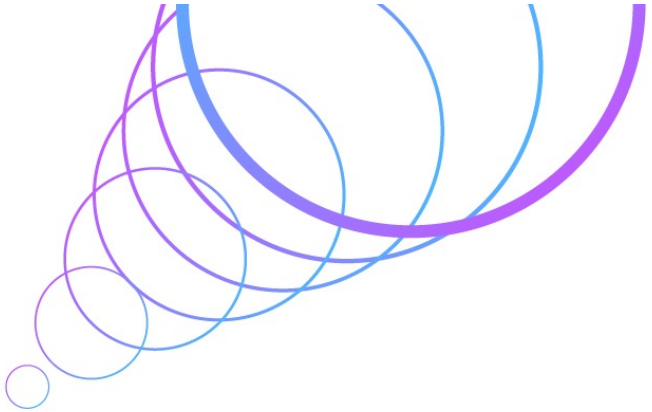
AI & Data Platform 소개

- https://naver-career.gitbook.io/kr/service/search/ai-and-data-platform

경력 소개

- https://d2.naver.com/news/7591059





Thank You