DEVIEW 2020

서비스로 보는 소프트웨어 2.0 : 챗봇 ML 파이프라인 구축

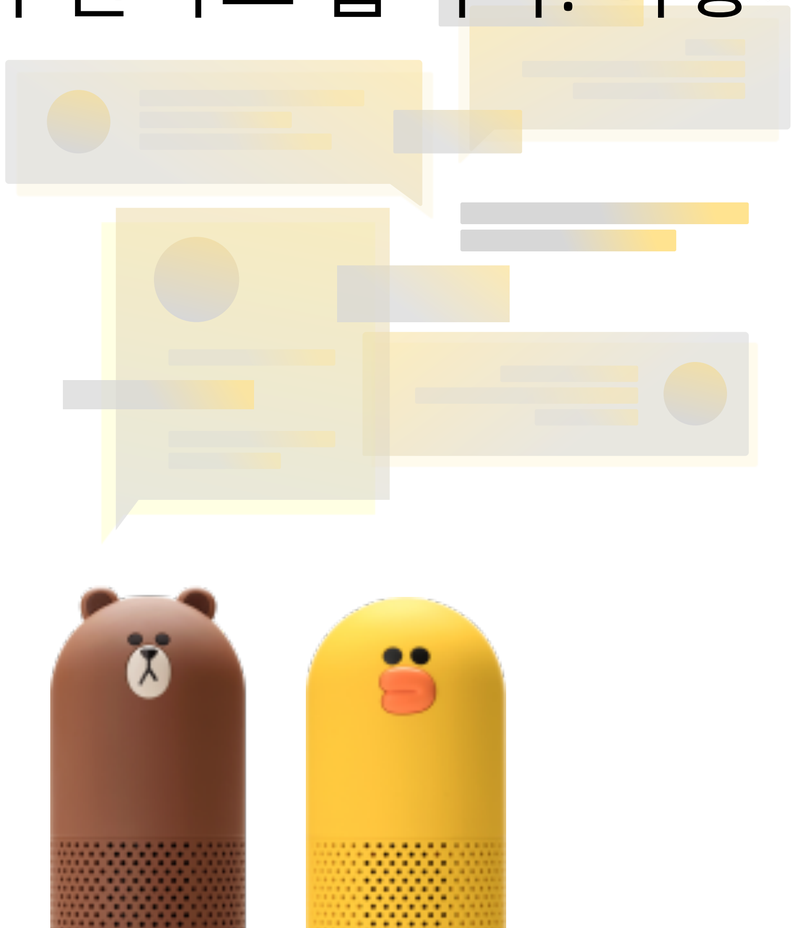
윤성빈 Conversation Engineering

CONTENTS

1. ML 파이프라인, 맨땅부터 시작하기

1.1 맨땅의 ML 파이프라인 (1/4)

[가정] 저는 엔지니어입니다. 100만 건 대화 데이터를 주고 서비스를 시작해야 한다고 합니다. 어떻게 해야 할까요?



1.1 맨땅의 ML 파이프라인 (1/4)

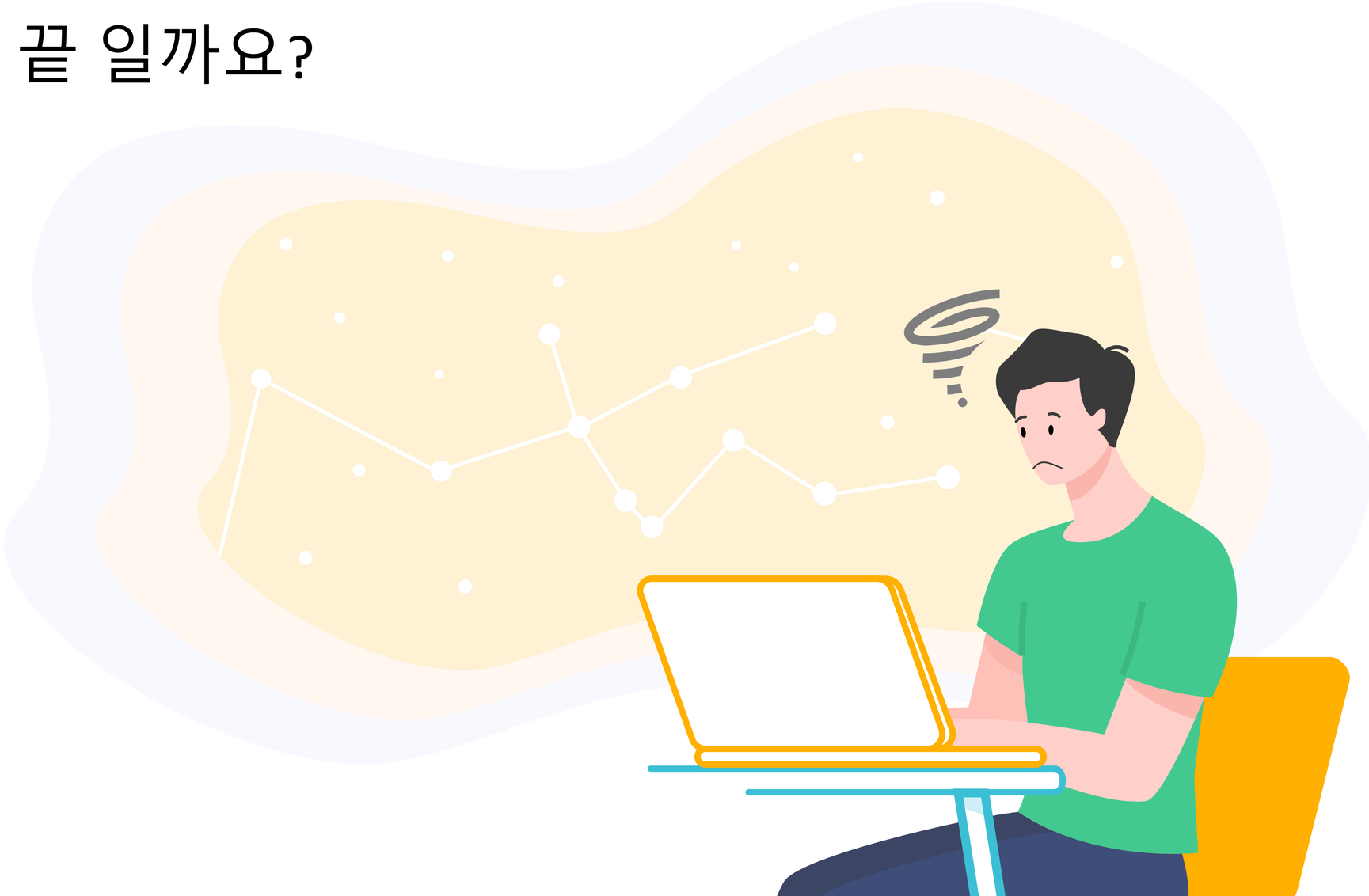
DEVIEW 2020

일단 필요한 기능을 개발합니다.

- 데이터 수집
- 데이터 가공
- 모델 학습
- 서빙 엔진
- 검증

1.1 맨땅의 ML 파이프라인 (2/4)

여기서 끝 일까요?



1.1 맨땅의 ML 파이프라인 (2/4)

DEVIEW 2020

서비스 배포 관리

- CI/CD 구축
- 서비스 버전이 기록
- 설정에 대한 정보들이 기록

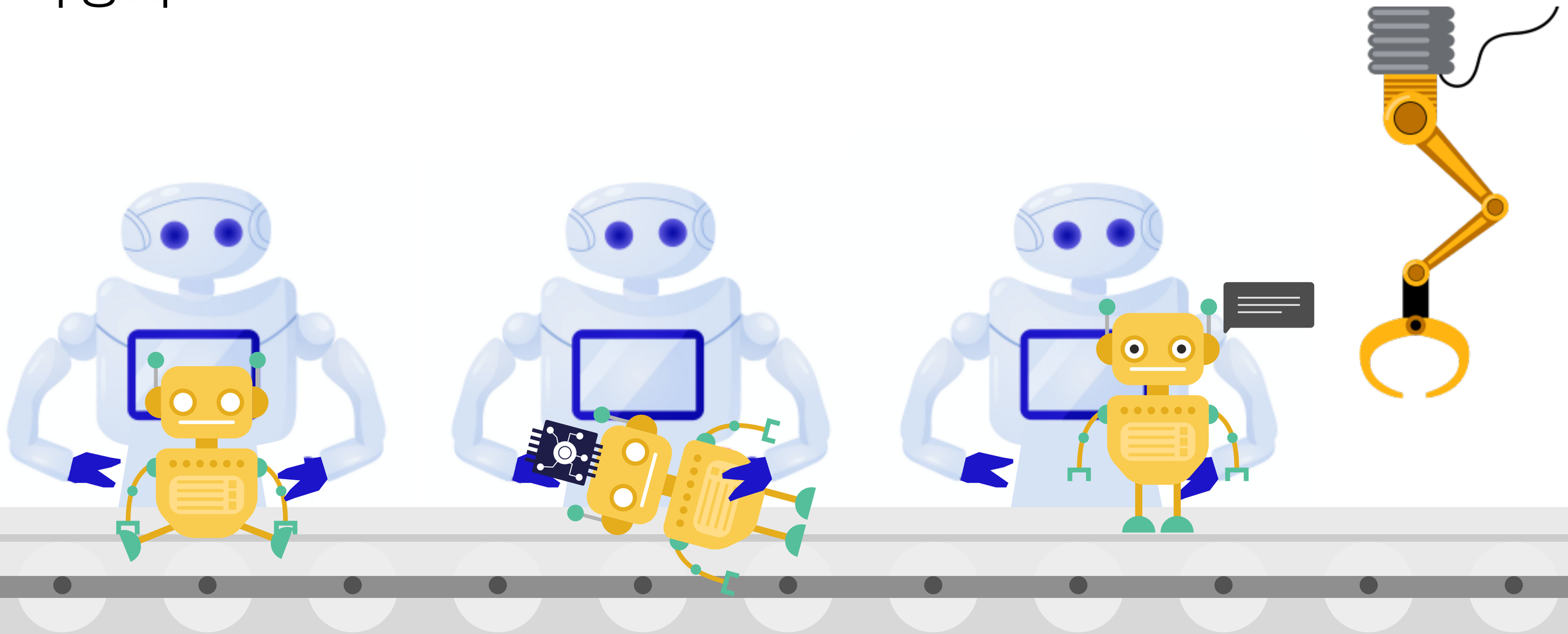
도메인의 확장

- 위의 작업을 N번 사람 반복

1.1 맨땅의 ML 파이프라인 (3/4)

DEVIEW 2020

자동화



1.1 맨땅의 ML 파이프라인 (3/4)

DEVIEW 2020

무엇이 자동화가 되어야 할까요?

- 새로운 데이터가 들어오면 학습이 자동으로 진행
- 학습이 완료되면 서비스가 자동으로 배포
- 배포된 버전이 문제가 없는지 확인

1.1 맨땅의 ML 파이프라인 (4/4)

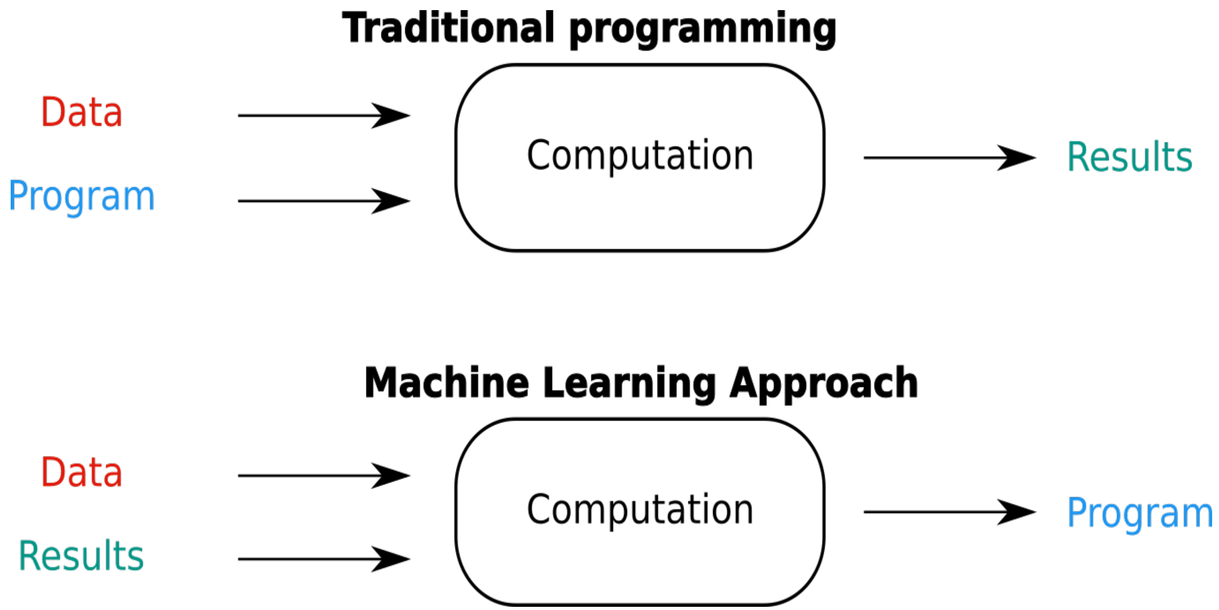
DEVIEW 2020

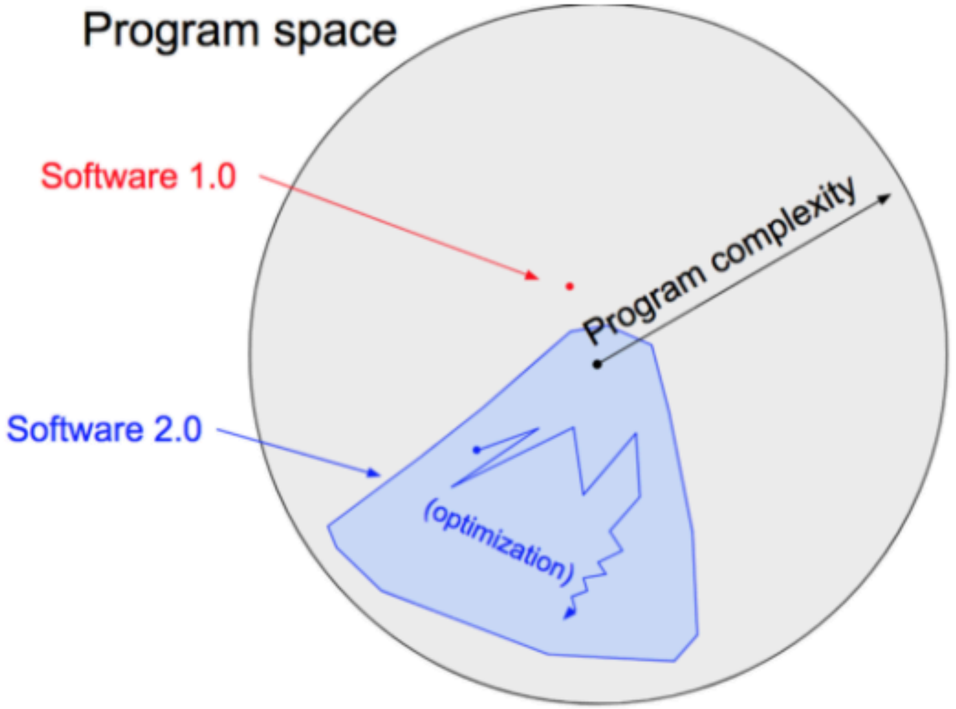
그러면 우리는 ML 파이프라인을 완성한 것일까요?

- 데이터는 어디에 저장해야 하나요?
- 성능을 올리기 위해서 무엇을 추가해야 할까요?
- 모든 요구 사항이 해결 되었을까요?

2. ML 파이프라인, 실제 구축기

2.1 Software 2.0 이란





https://medium.com/@karpathy/software-2-0-a64152b37c35

2.2 구축에 앞서 (1/3)

DEVIEW 2020

ML 파이프라인의 방향성

2.2 구축에 앞서 (2/3)

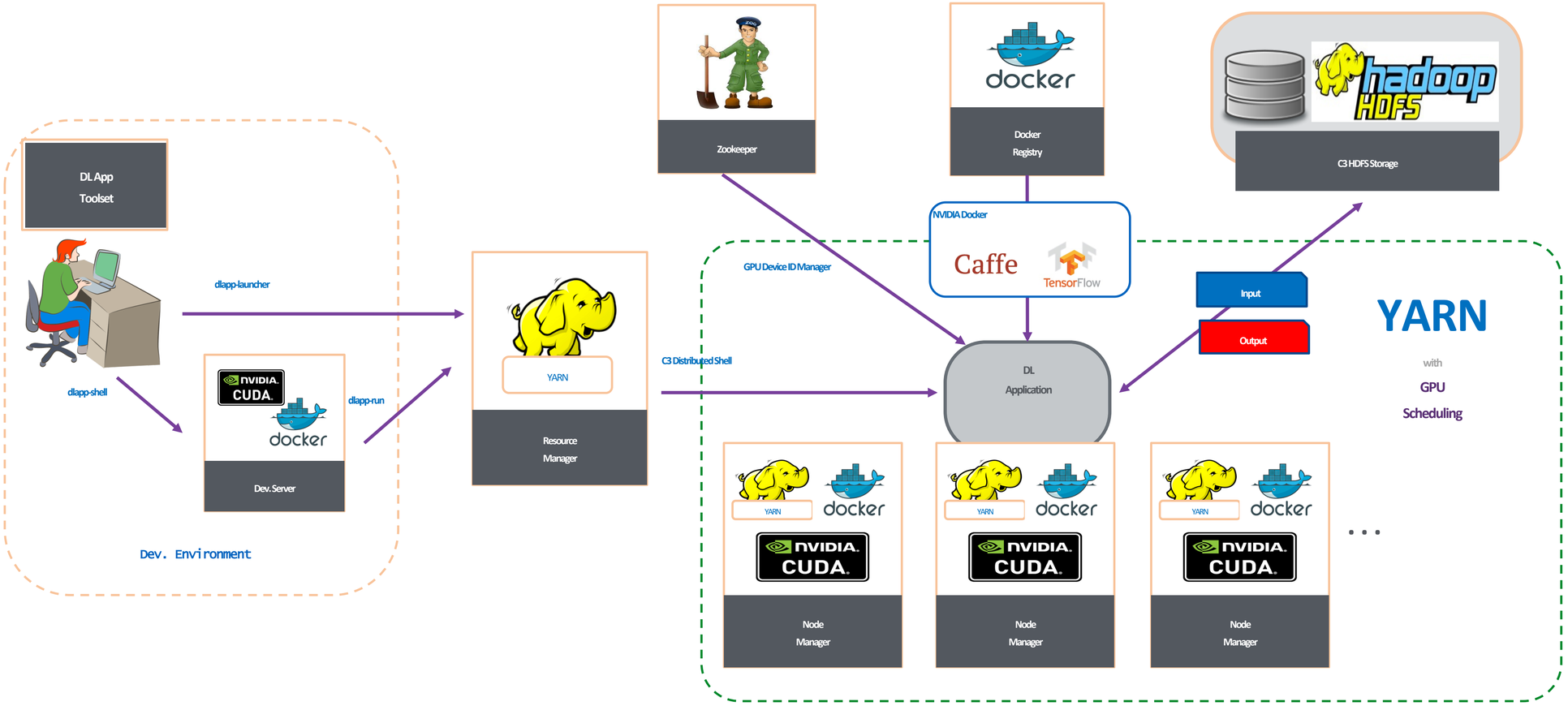
DEVIEW 2020

Spark & Scala

- Spark은 비정형 데이터를 분산 처리로 빠르게 처리
- Spark은 Scala로 만들어졌고 python 보다 빠름
- Data 조작하기에는 functional programing언어가 최고
- 팀내 모든 프로젝트가 Scala

2.2 구축에 앞서 (3/3)

무엇이 준비가 되어 있었나?

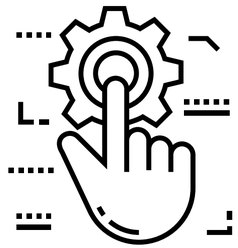


https://deview.kr/2017/schedule/190

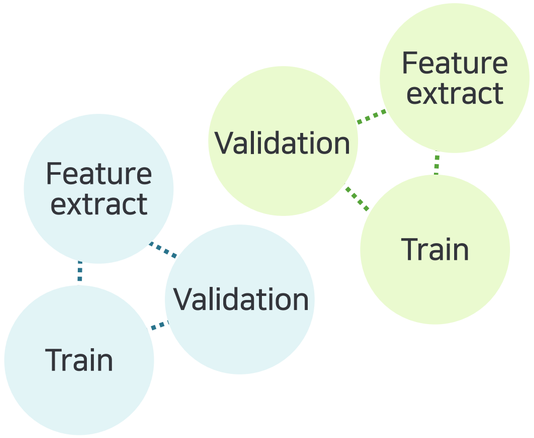
2.3 ML 파이프라인의 변천사

DEVIEW 2020

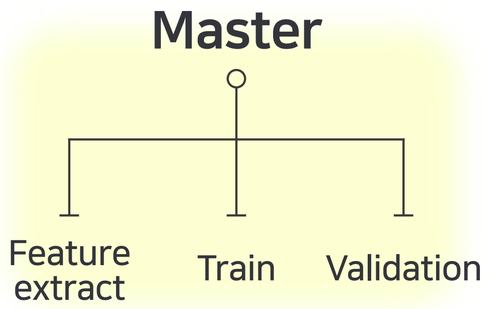
Manual



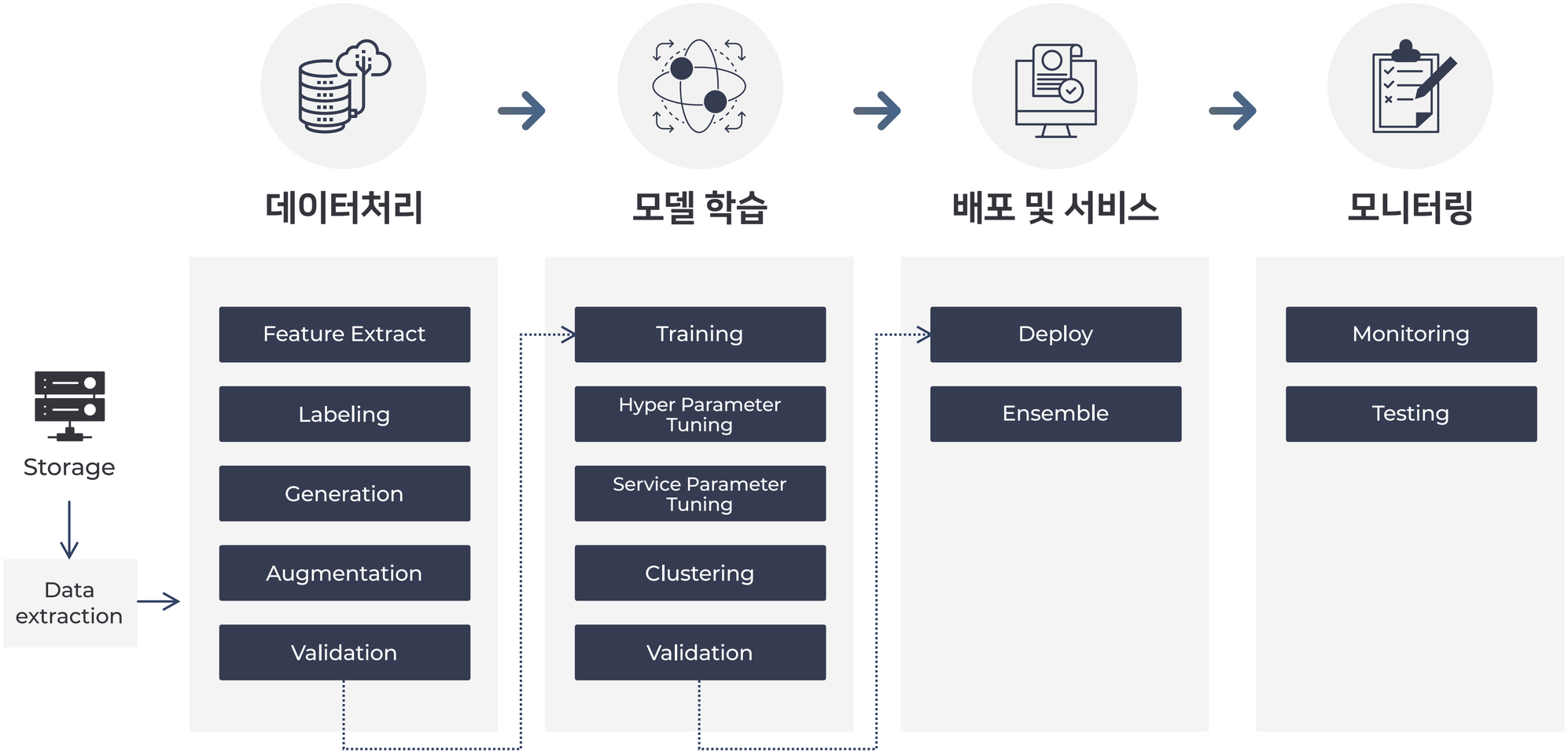
Automation



Modularization

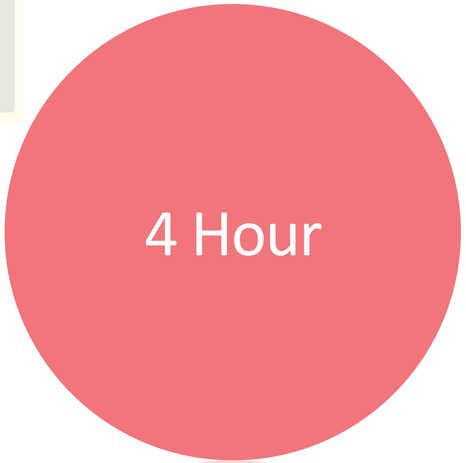


2.4 ML 파이프라인 아키텍처

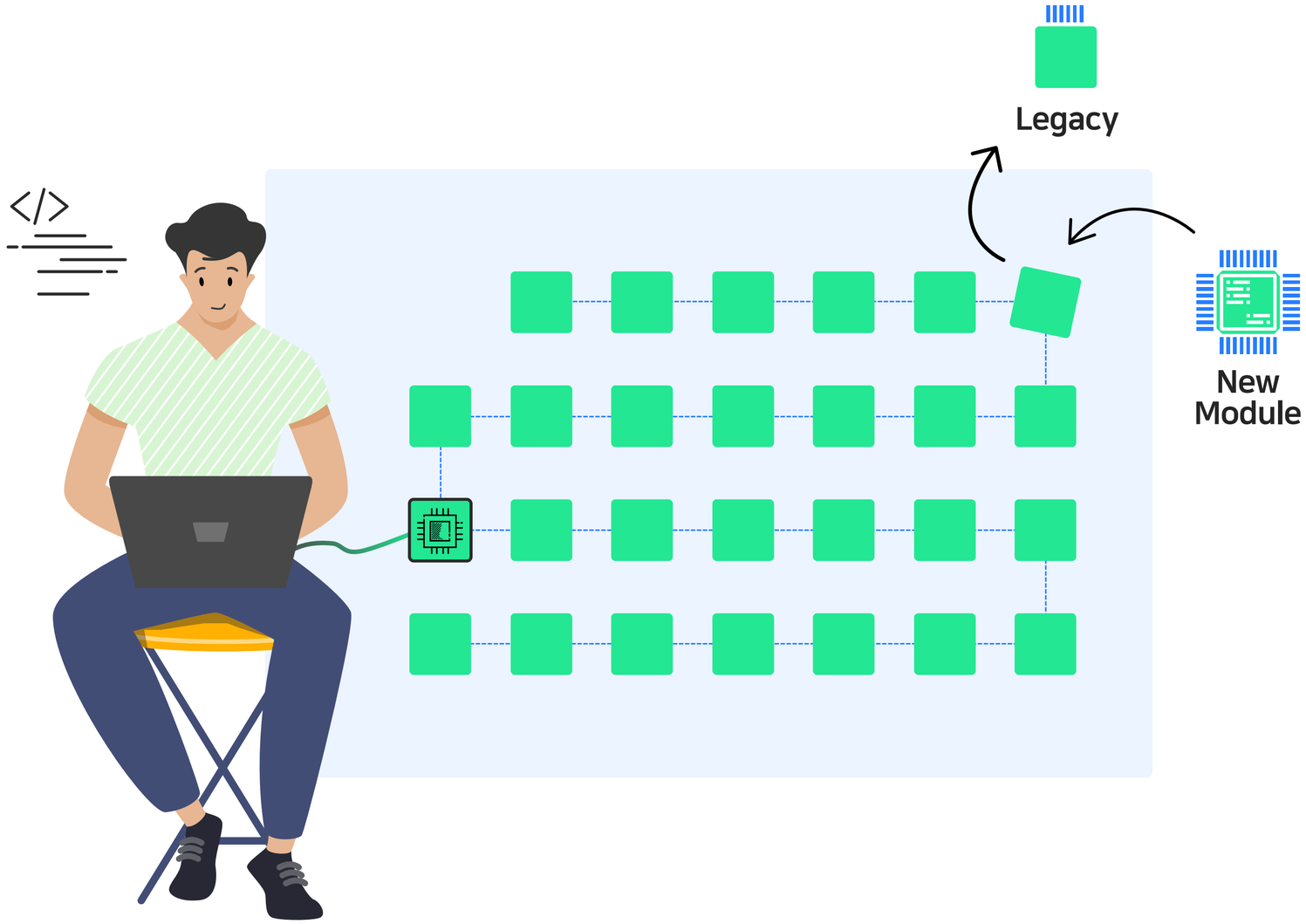


2.5 모듈화를 통해 얻은 것 (1/4)

- Resource를 많이 사용하는 모듈에 집중적으로 자원을 할당 할 수 있었다.
- 모듈화를 통한 단위 테스트는 우리의 병목이 어디인지 알게 해주었다.



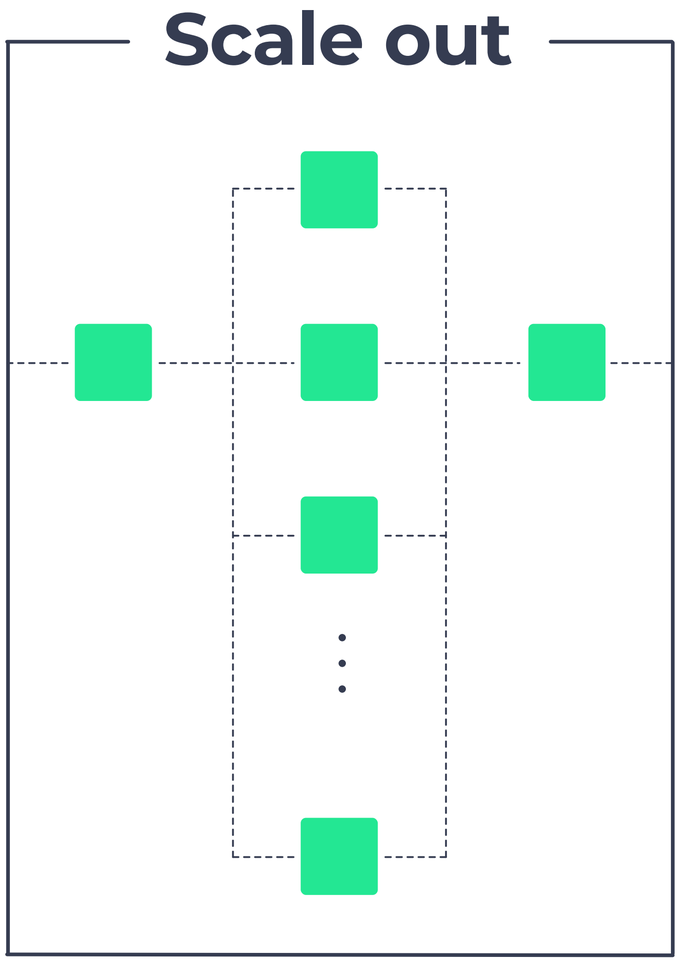
2.5 모듈화를 통해 얻은 것 (2/4)

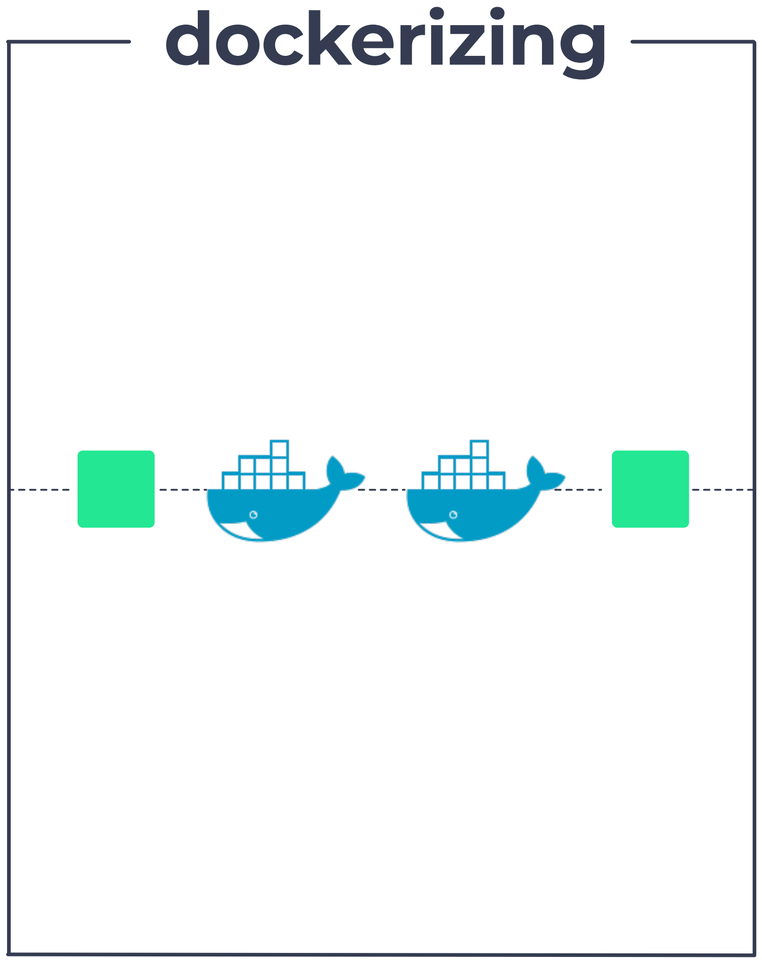


2.5 모듈화를 통해 얻은 것 (3/4)

연구	서비스 개발	기획(제품)
데이터 검증 모델 추가 모델 검증	모듈 단위의 테스트 서비스의 안정성	전체 서비스 완성도

2.5 모듈화를 통해 얻은 것 (4/4)



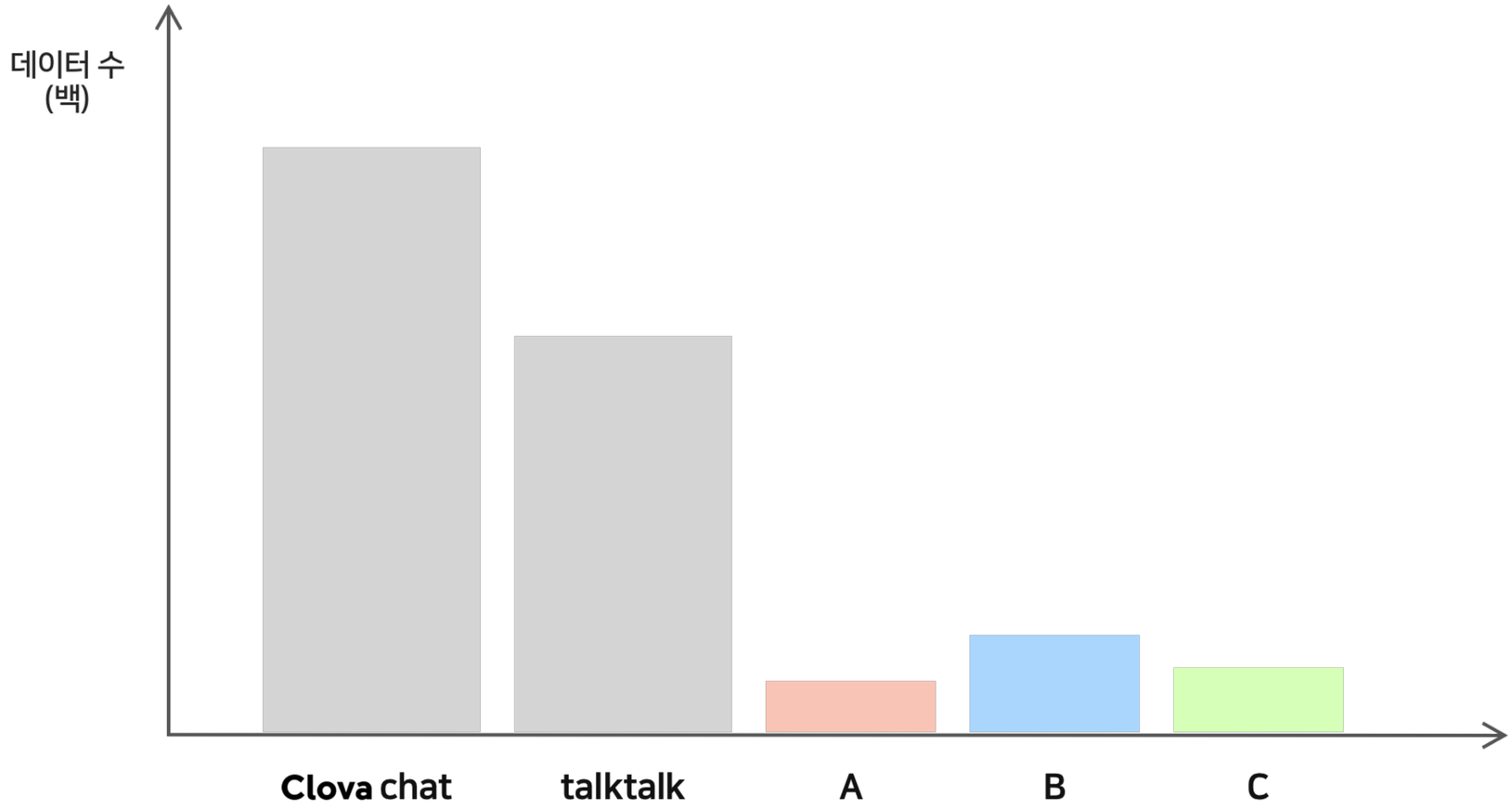


2.6 데이터의 변화 – 비정형 빅데이터

그래서 비정형 빅데이터가 들어온다면,

- 분석 할 수 있을까
- 서비스 될 수 있을까
- 무엇을 생각 해야 할까

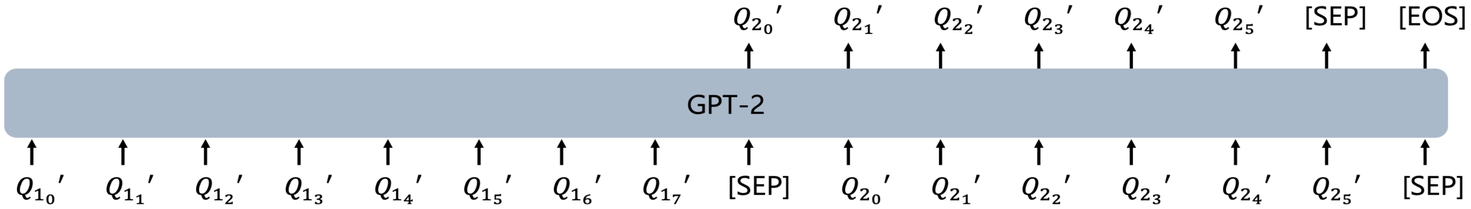
2.6 데이터의 변화 – 스몰 데이터 (1/2)

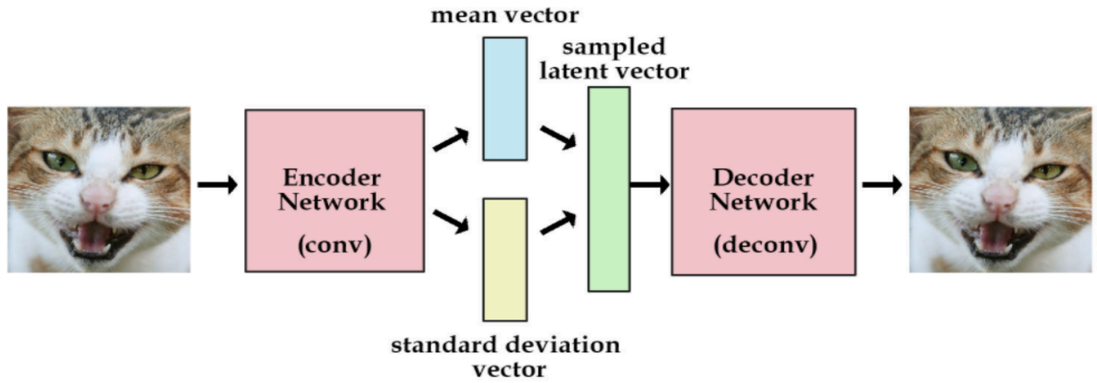


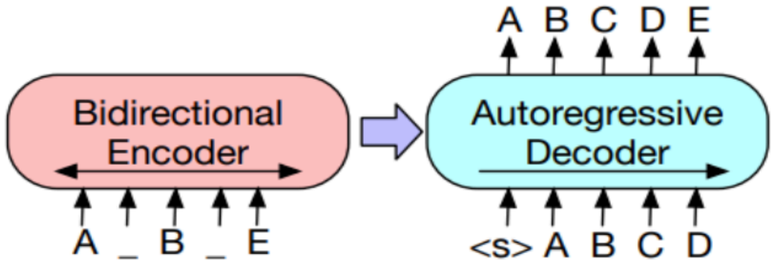
2.6 데이터의 변화 – 스몰 데이터 (2/2)

기존	확장
점심 먹었니?	점심 뭐 먹었어? 점심은 먹었니? 점심 먹었어요?

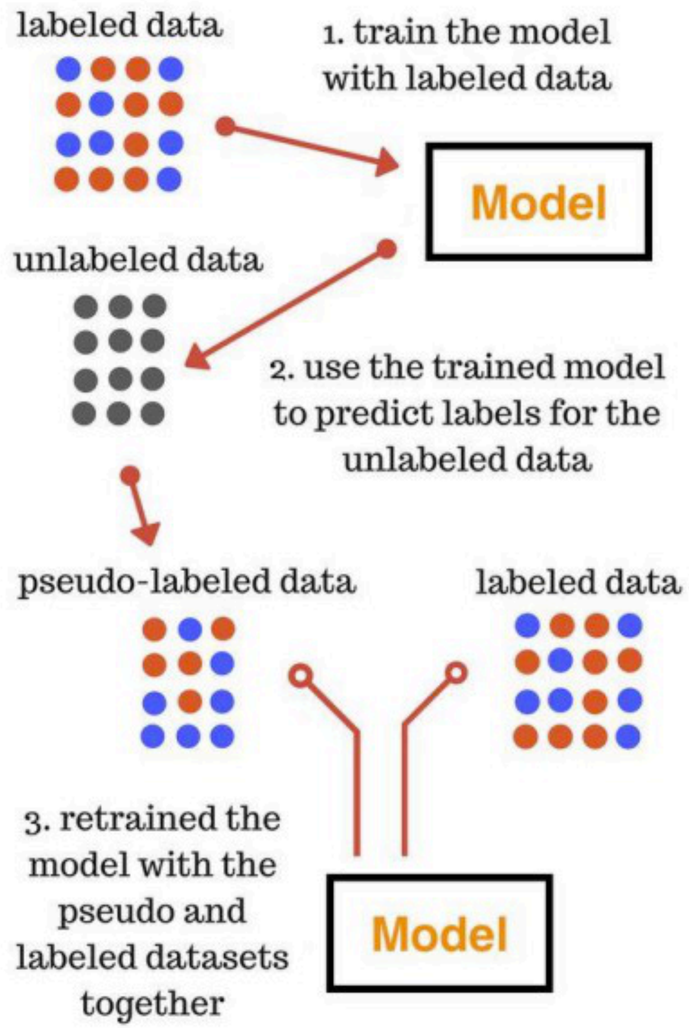
2.7 스몰 데이터 해결 (1/3)





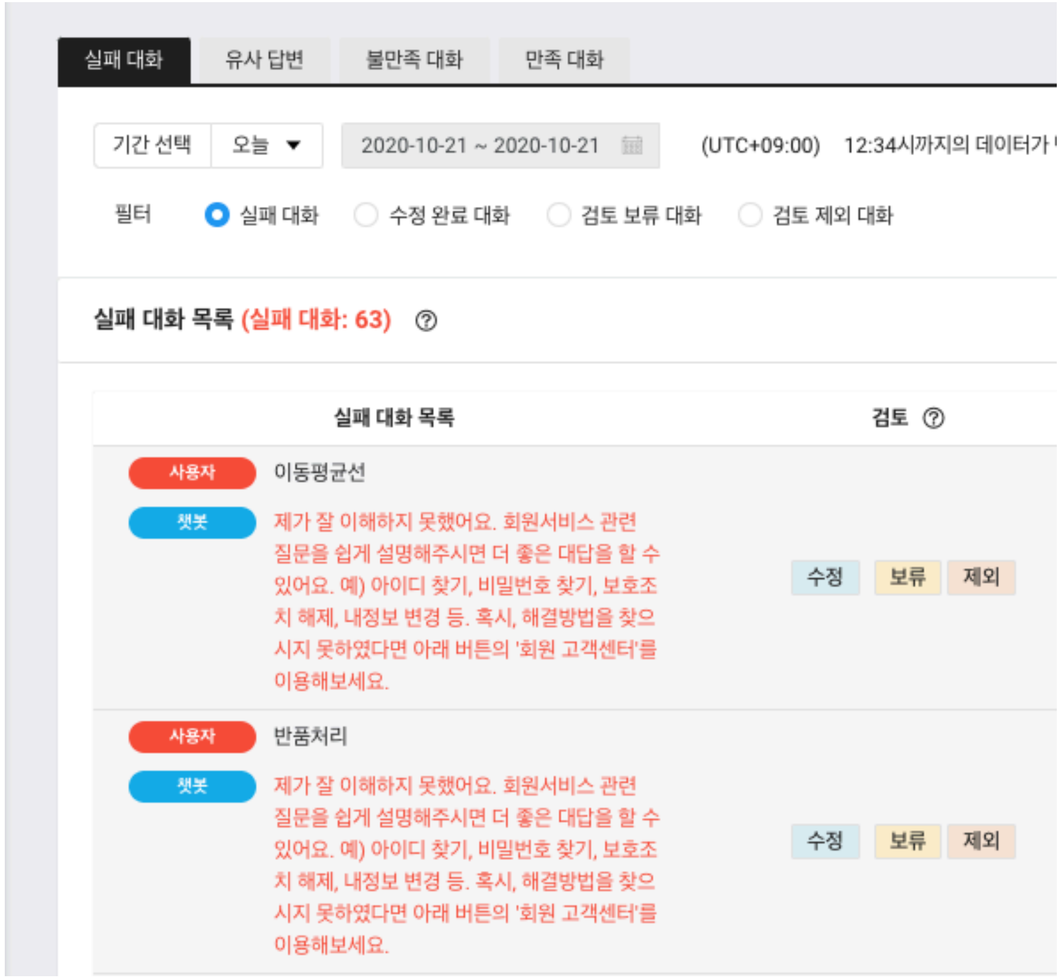


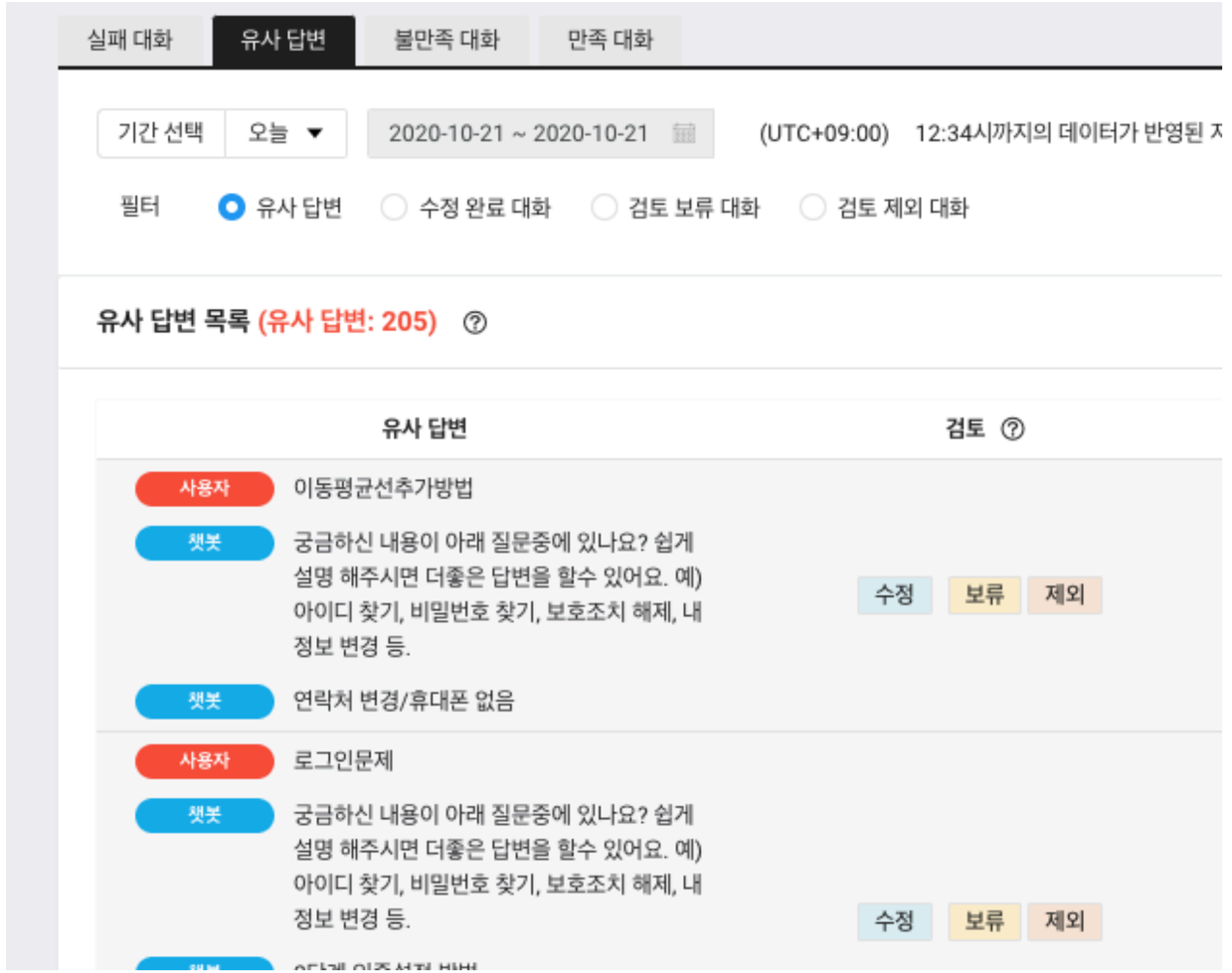
2.7 스몰 데이터 해결 (2/3)



2.7 스몰 데이터 해결 (3/3)

재학습 – 실패 대화 & 유사 답변



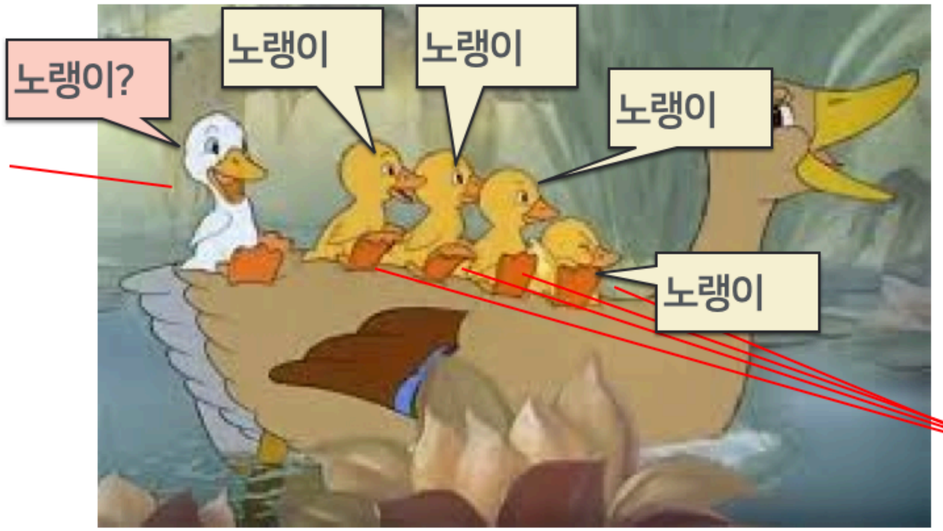


2.8 데이터 노이즈 해결 (1/2)

DEVIEW 2020

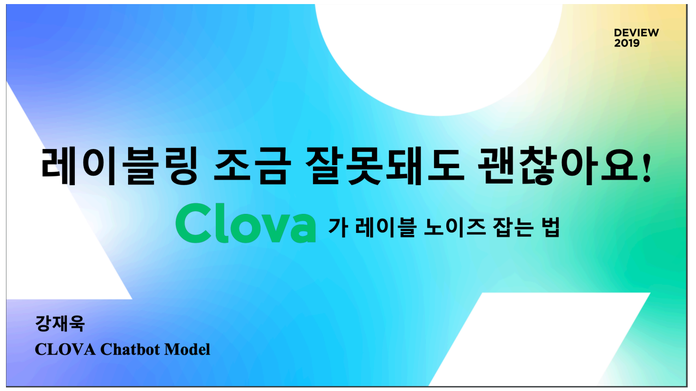
Labeled as the Same Class

The Mislabeled (Label Noise)

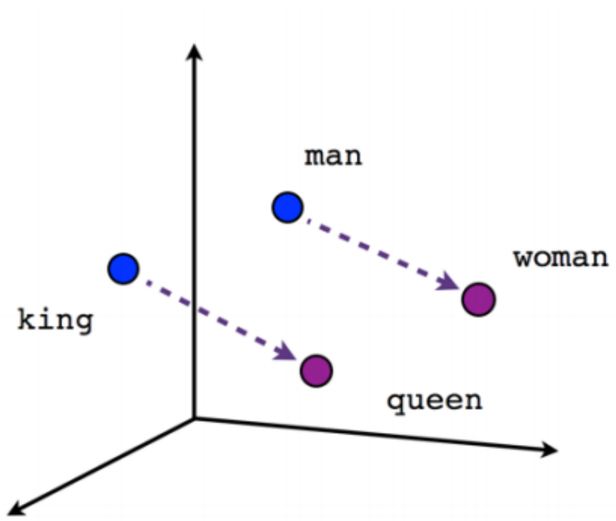


The Well-labeled

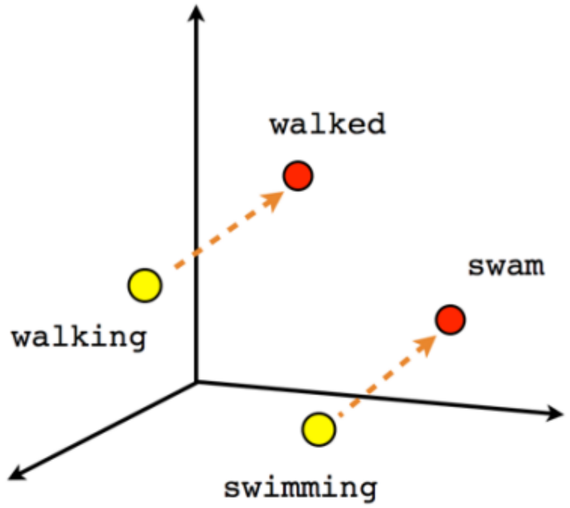
2.8 데이터 노이즈 해결 (2/2)



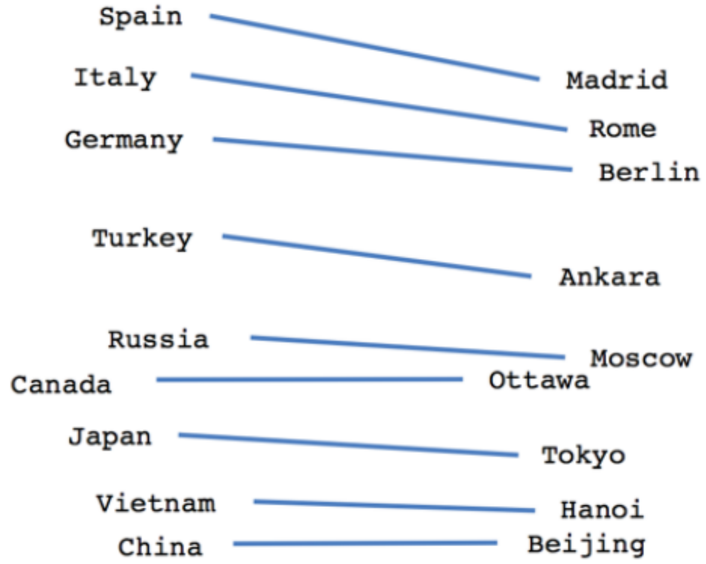
2.9 품질 좋은 임베딩



Male-Female

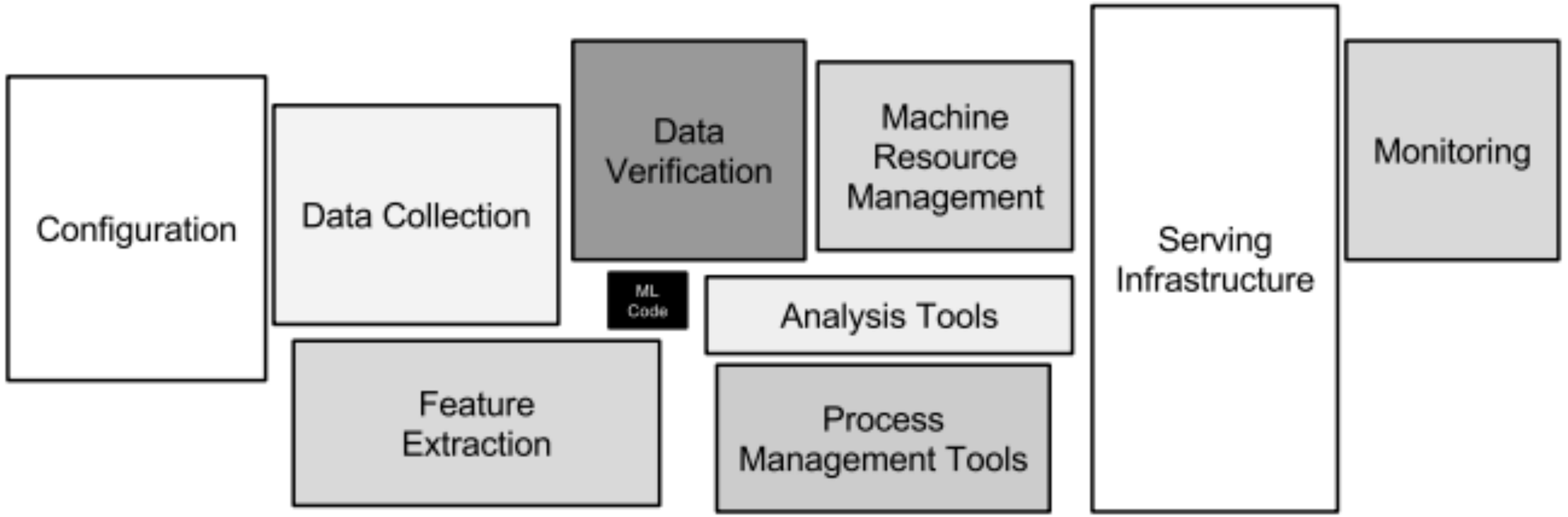


Verb tense



Country-Capital

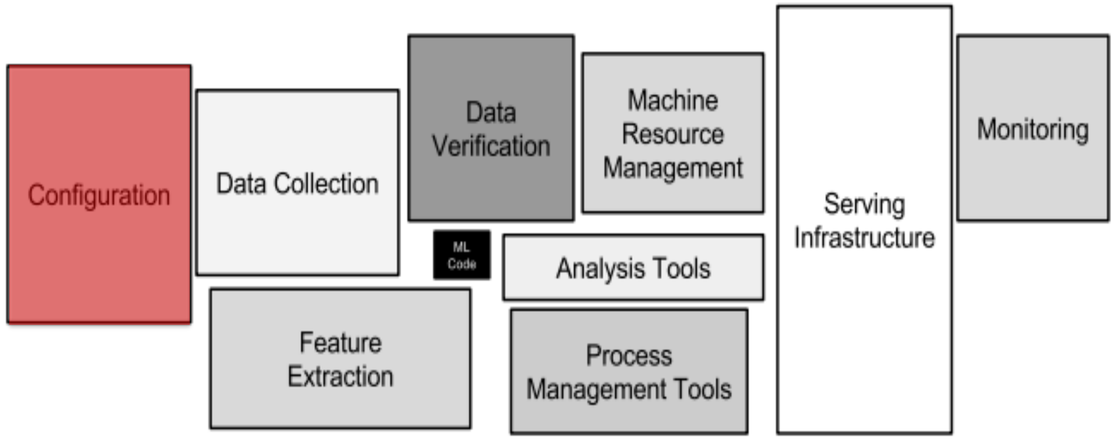
2.10 ML 시스템의 기술적 한계



Sculley, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Advances in neural information processing systems* (pp. 2503-2511).

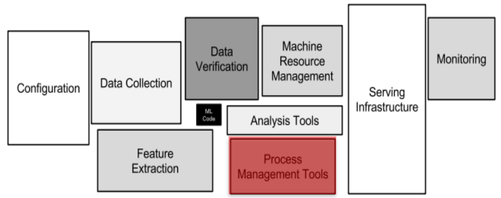
2.11 확장의 유연함

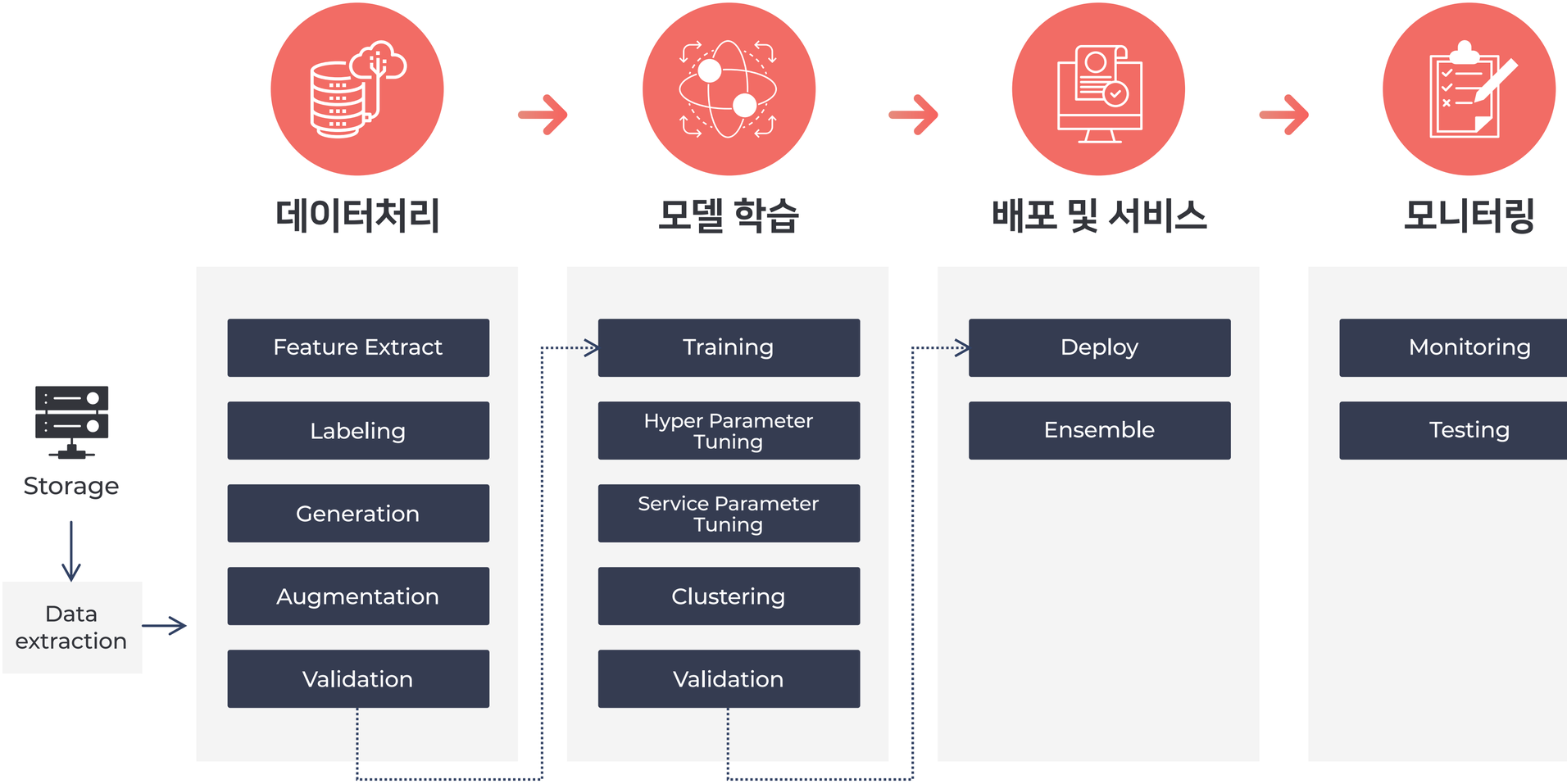
많은 부분이 config로 제어를 할 수 있어야한다.



언어	추가
한국어	한국어 기본 파라메터 추가
일본어	일본어 기본 파라메터 추가
영어	영어 기본 파라메터 추가
중국어(번체)	중국어(번체) 기본 파라메터 추가
중국어(간체)	중국어(간체) 기본 파라메터 추가
타이완(번체)	타이완(번체) 기본 파라메터 추가
태국어	태국어 기본 파라메터 추가
베트남어	베트남어 기본 파라메터 추가

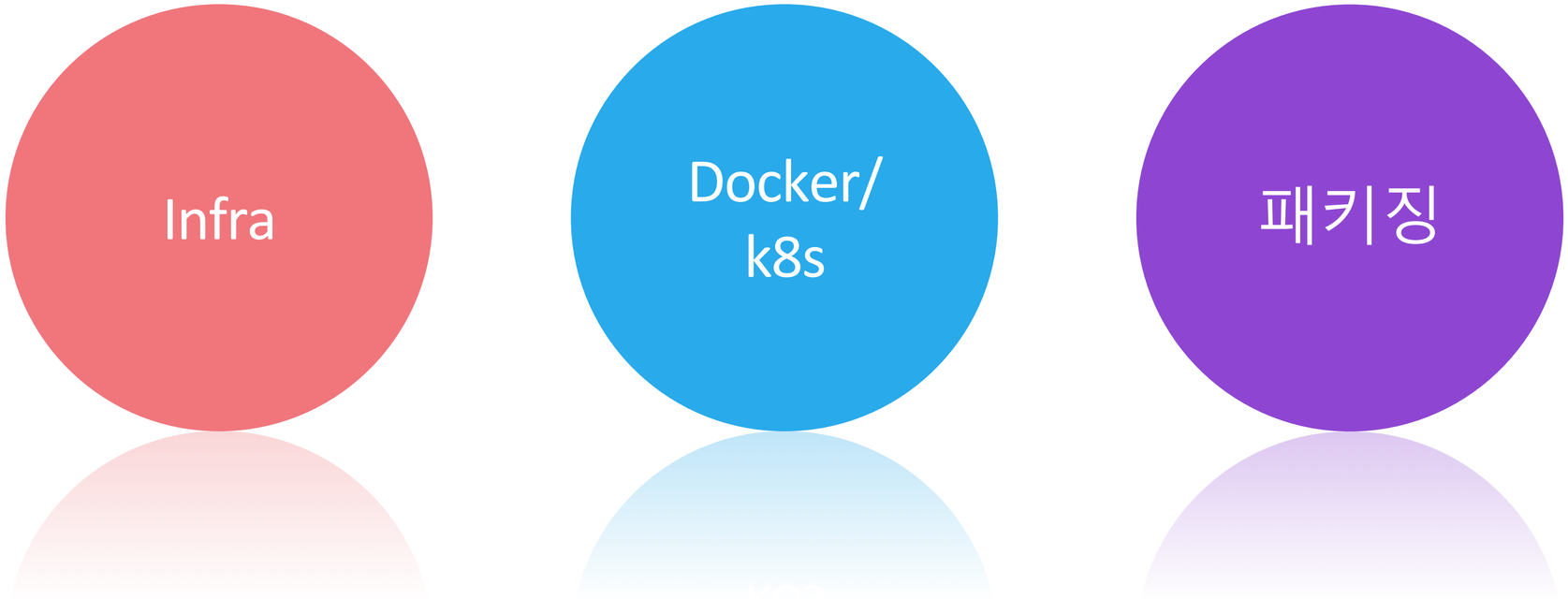
2.12 현재 파이프라인


Configuration
Data Collection
Data Verification
Machine Resource Management
Monitoring
Serving Infrastructure
Analysis Tools
Feature Extraction
Process Management Tools


데이터처리
모델 학습
배포 및 서비스
모니터링
Storage
Data extraction
Feature Extract
Labeling
Generation
Augmentation
Validation
Training
Hyper Parameter Tuning
Service Parameter Tuning
Clustering
Validation
Deploy
Ensemble
Monitoring
Testing

2.13 환경의 확장

B2B를 진행하면서



3. ML 파이프라인, Spark 삽질기

3.1 들어가며

4 Hour

10 Minute

3.2 속도와 효율의 tradeoff

DEVIEW 2020

한정된 자원 속에서

- 최대의 설정으로 1개 -> 자원의 낭비
- 최소의 설정으로 N개 -> OOM 발생

한정된 자원으로 최적화를 이루기 위한 설정을 이야기해보자.

3.3 용어

Driver Program – SparkContext를 생성하는 메인 프로세스

Executor – Application에서 요청한 Task를 수행하는 프로세스

Job – Application에서 Spark에 요청하는 일련의 작업 단위

Task – Executor에서 수행되는 최소 작업 단위

3.4 튜닝 1 – Executor core & num

DEVIEW 2020

```
$spark-submit
 --class Mlpipeline.class
 --executor-memory 10G
 --executor-cores 5
 --num-executors 10
 --jars mlpipeline.jar
```

동시성

- Core 수 = Task 처리량
- Core ⬆ 오버헤드 ⬆

주의)

Q) 16core 머신에 16core를 할당하면 돌아 갈까?

A) X , Driver Program이 실행되어야 한다.

3.4 튜닝 1 – Executor core & num

DEVIEW 2020

```
$spark-submit
 --class Mlpipeline.class
 --executor-memory 10G
 --executor-cores 5
 --num-executors 10
 --jars mlpipeline.jar
```

병렬성

- executor 수 = 동시에 처리되는 job
- 서버 한대의 스펙과, executor core 수에 의해 계산됨

Total Core / executor core = executor num

3.4 튜닝 2 - Dynamic Module

spark-default.conf

`spark.dynamicAllocation.enabled`	false	Whether to use dynamic resource allocation, which scales the number of executors registered with this application up and down based on the workload. For more detail, see the description here.	1.2.0
`spark.dynamicAllocation.initialExecutors`	`spark.dynamicAllocation.minExecutors`	Initial number of executors to run if dynamic allocation is enabled. If `--num-executors` (or `spark.executor.instances`) is set and larger than this value, it will be used as the initial number of executors.	1.3.0
`spark.dynamicAllocation.maxExecutors`	infinity	Upper bound for the number of executors if dynamic allocation is enabled.	1.2.0
`spark.dynamicAllocation.minExecutors`	0	Lower bound for the number of executors if dynamic allocation is enabled.	1.2.0

```
conf.set("spark.scheduler.mode", "FAIR")
```

3.4 튜닝 3 - cluster & local

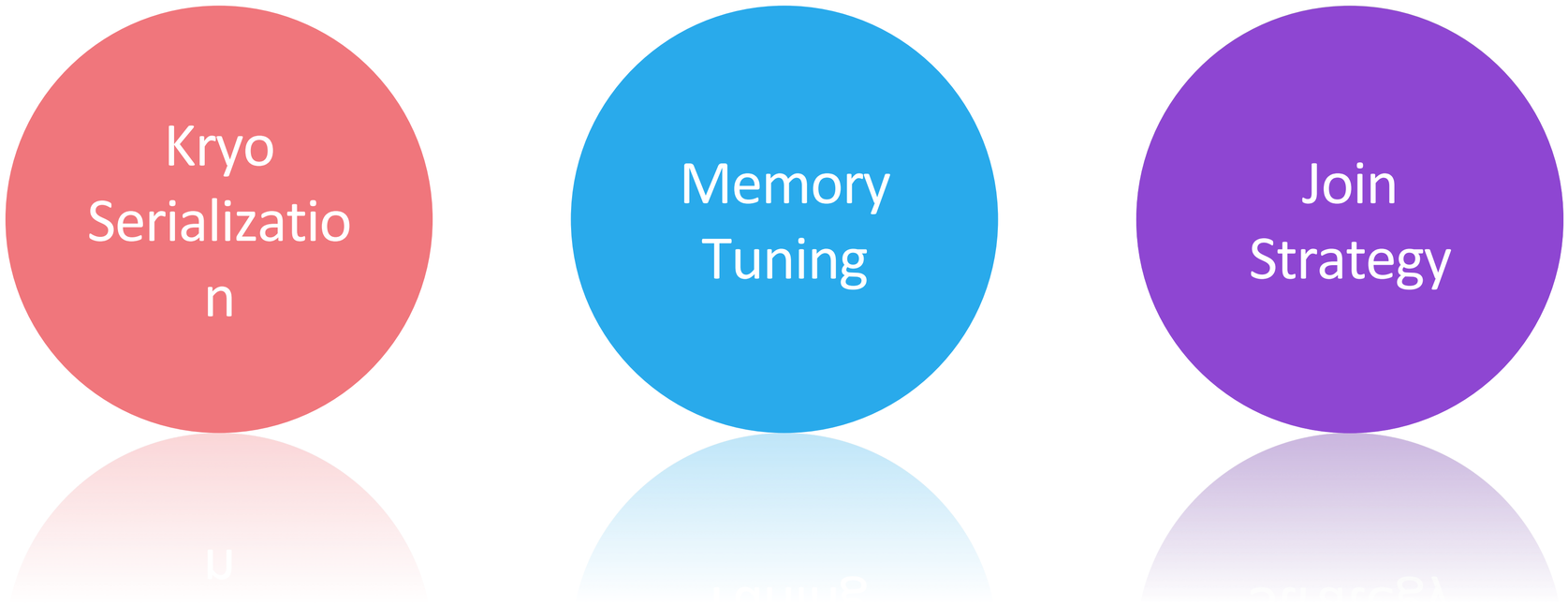
DEVIEW 2020

모든 작업을 cluster에서 실행 해야 하나?

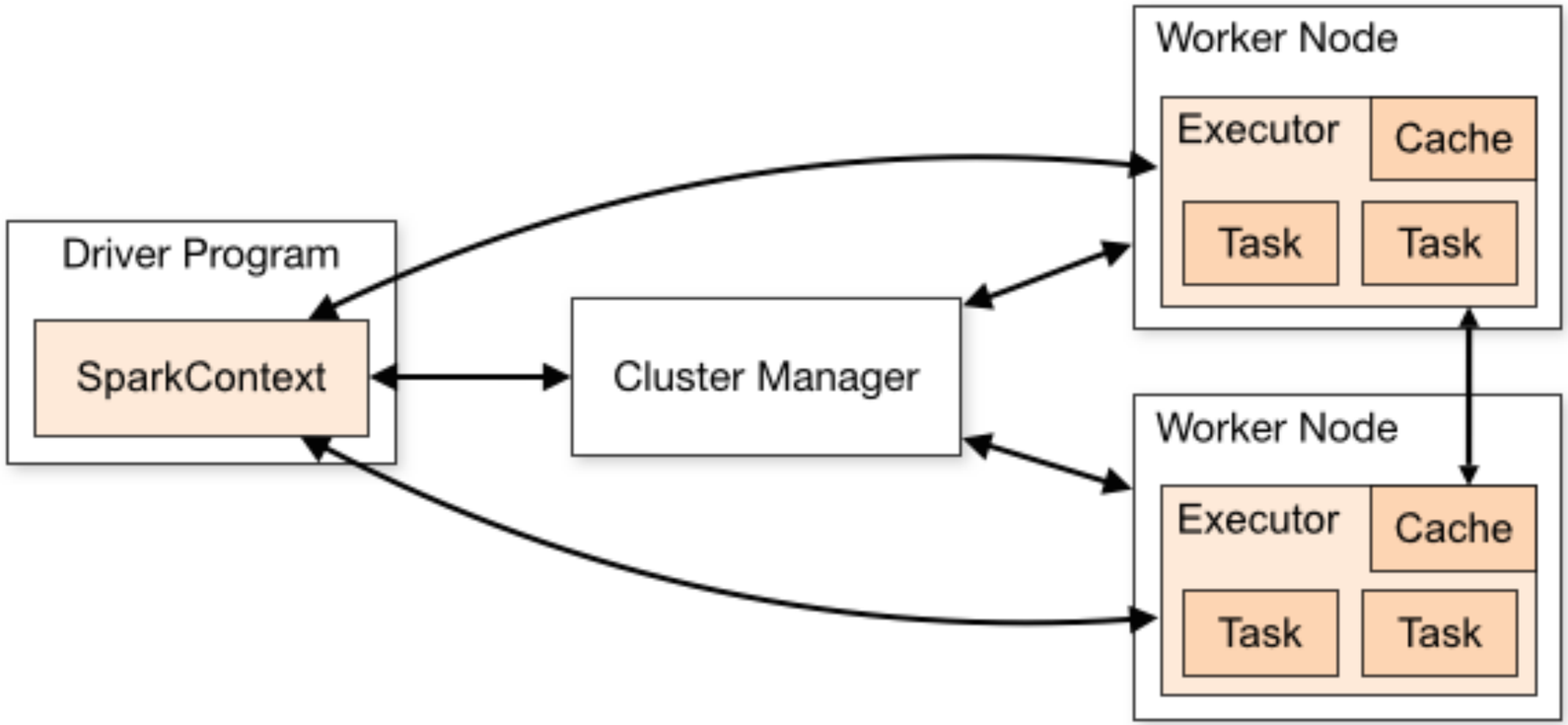
- Spark session overhead
- 리소스 할당 = 8 s
- 동작하는 환경에 따라 선택적

3.5 그 밖의 튜닝

더 생각 해 볼 수 있는 방법



3.6 병목 (1/2)


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task


Error
Out of memory.
OK

3.6 병목 (2/2)

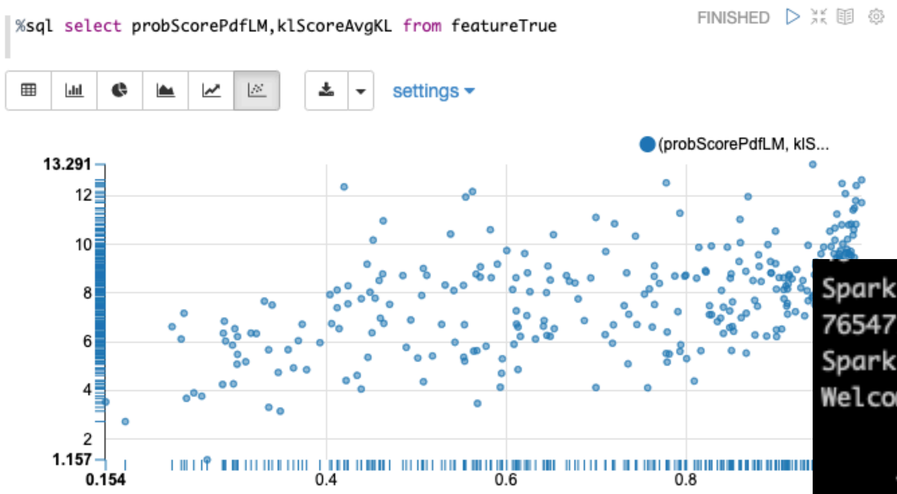
DEVIEW 2020

JNI를 사용한다면

- Resource 초기화 확인
- 환경 변수가 executor에서도 적용되고 있는지 확인
- Driver – executor network 비용을 줄이기 위해서 mapPartition을 사용

3.7 튜닝 3 – use spark and debug

DEVIEW 2020

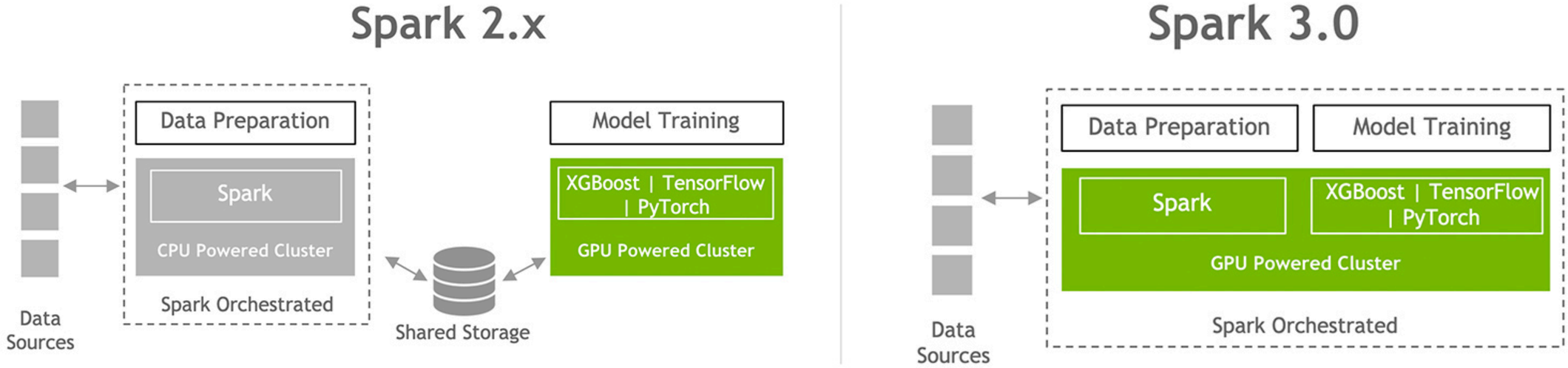



```
Spark context available as 'sc' (master = local[*], app id = local-1603285
765471).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.0
      /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0
_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Show 20 entries

App ID	App Name	Started	Completed	Duration
local-1603285209622	balooRemote	2020-10-21 13:00:08	2020-10-21 13:00:18	9 s
application_1595826668059_19143453	balooRemote	2020-10-21 12:58:20	2020-10-21 12:59:40	1.3 min
local-1603284973529	balooRemote	2020-10-21 12:56:12	2020-10-21 12:57:08	56 s
application_1595826668059_19142912	balooRemote	2020-10-21 12:53:39	2020-10-21 12:56:08	2.5 min
local-1603284792210	balooRemote	2020-10-21 12:53:11	2020-10-21 12:53:31	20 s
local-1603284778449	balooRemote	2020-10-21 12:52:57	2020-10-21 12:53:08	10 s
local-1603283974810	balooRemote	2020-10-21 12:39:34	2020-10-21 12:40:31	57 s
application_1595826668059_19140201	balooRemote	2020-10-21 12:37:05	2020-10-21 12:39:29	2.4 min
local-1603283799581	balooRemote	2020-10-21 12:36:38	2020-10-21 12:36:56	18 s
local-1603283781065	balooRemote	2020-10-21 12:36:20	2020-10-21 12:36:29	9 s

3.8 spark 3.0 GPU (1/3)

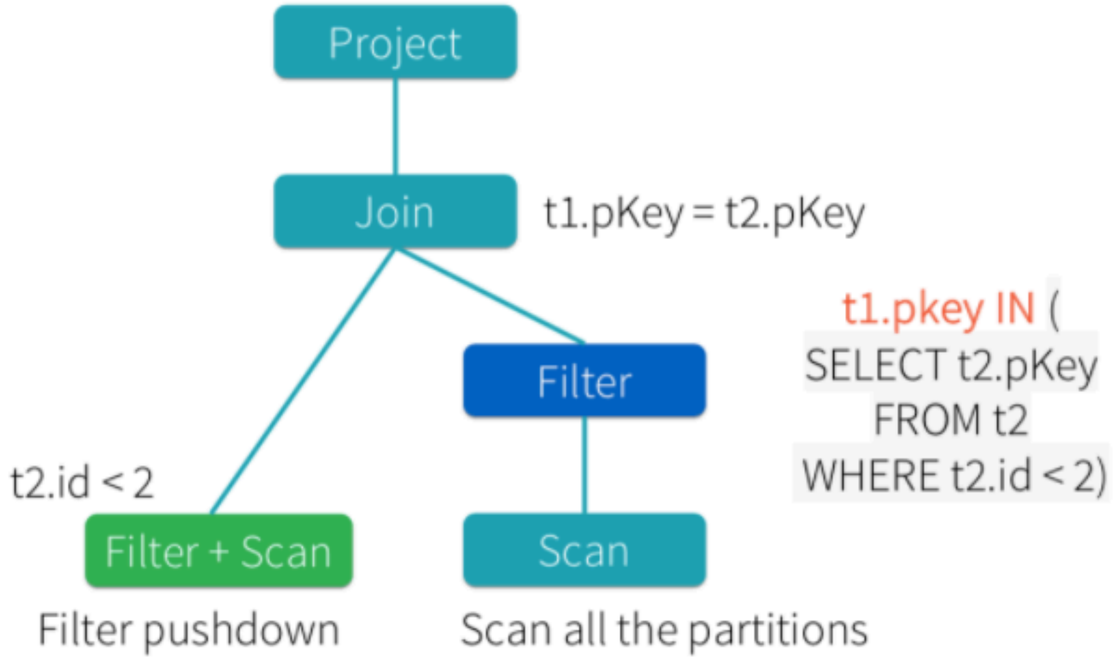


spark.executor.resource.gpu.amount =1

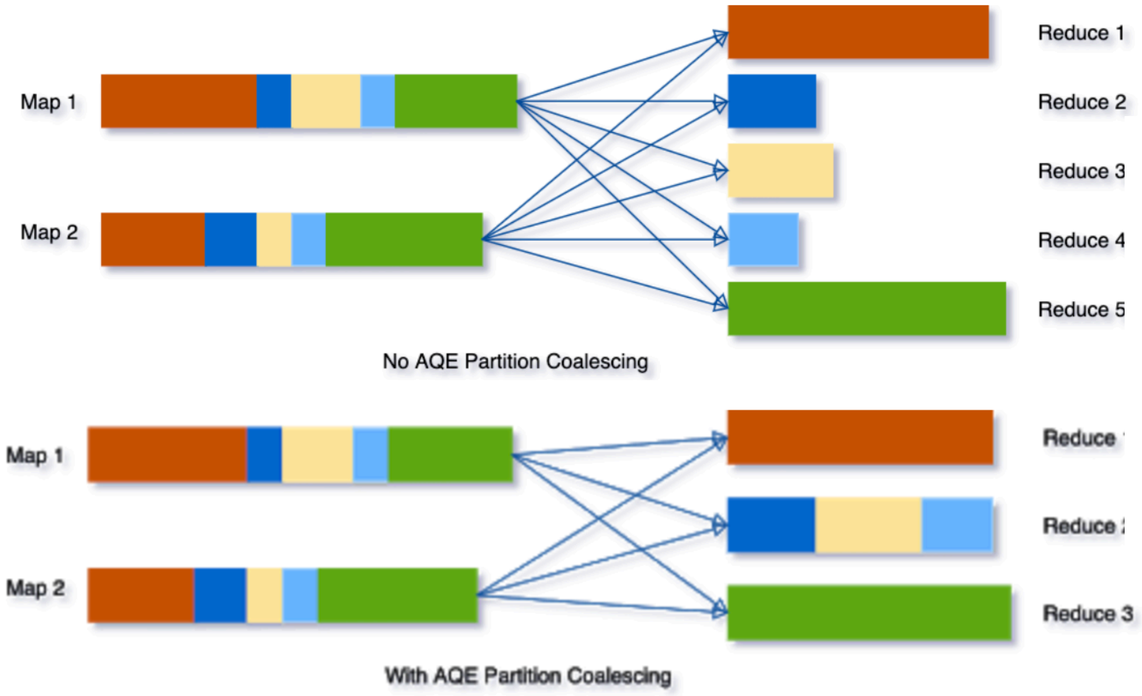
spark.task.resource.gpu.amount = 1

3.8 spark 3.0 DPP (2/3)


```
SELECT t1.id, t2.pKey
FROM   t1
  JOIN t2
    ON t1.pKey = t2.pKey
    AND t2.id < 2
```



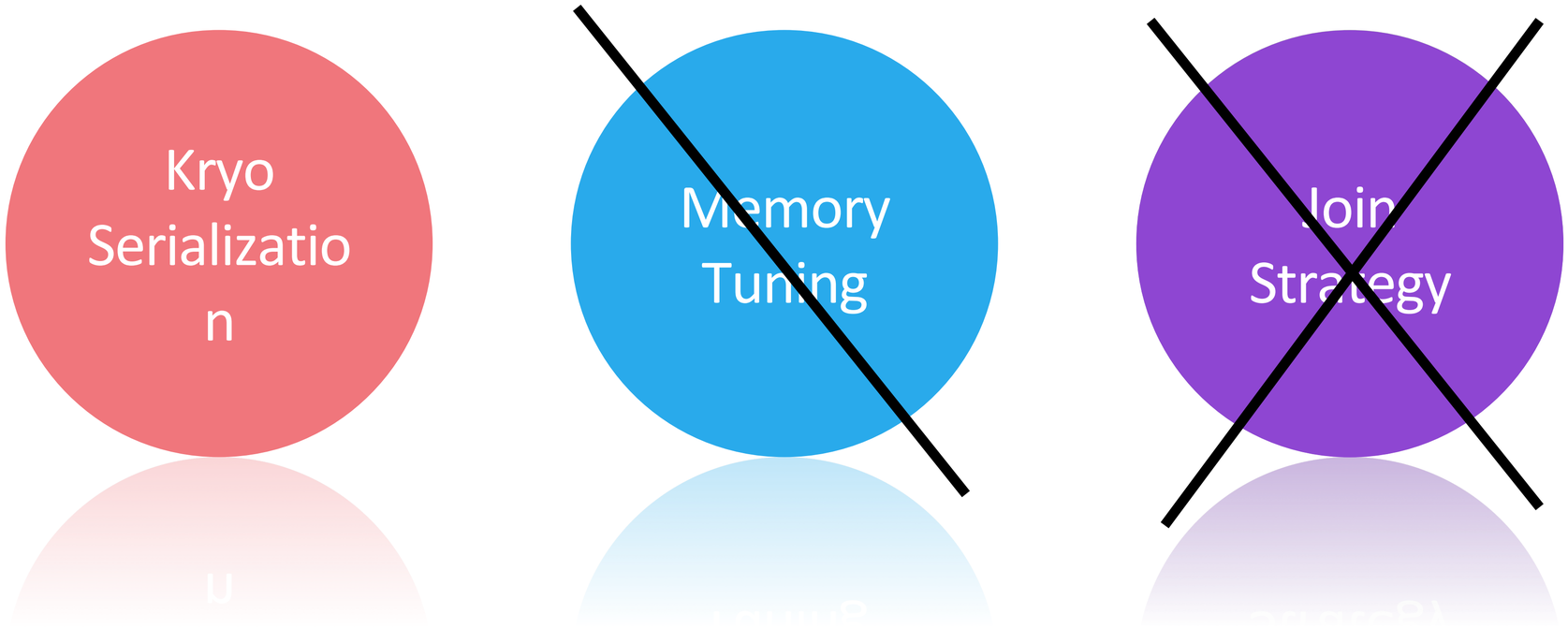
3.8 spark 3.0 AQE (3/3)



3.9 그 밖의 튜닝

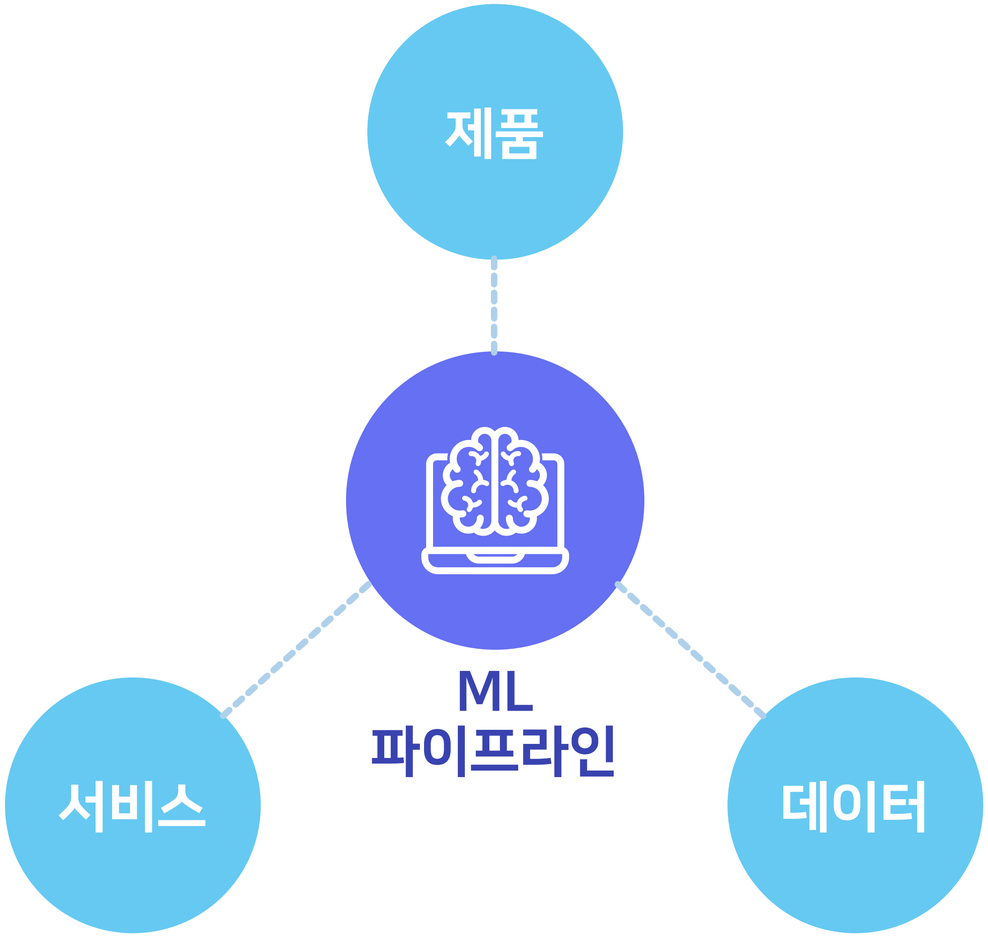
DEVIEW 2020

더 생각 해 볼 수 있는 방법



4. 정리

4.1 지속적인 발전

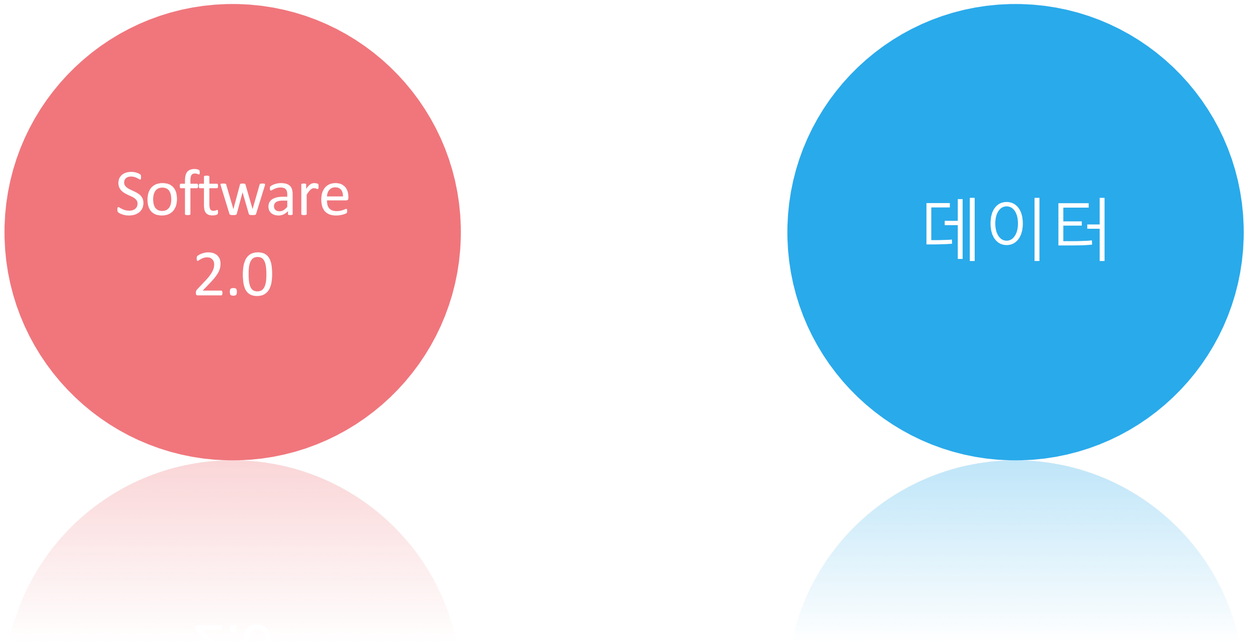


4.2 ML 파이프라인 방향성

1. 데이터 변화에 대응
2. 재현이 가능한지
3. 서비스 관점에서 설명가능한지
4. 한 명을 위한 파이프 라인이 아니고 모두를 위한 파이프라인
5. 챗봇 도메인에 대한 파이프라인

4.3 완벽한 ML 파이프라인

완벽한 ML 파이프 라인은 없다.
유연한 파이프 라인은 있다.
중요한 건 데이터의 흐름이다.



4.4 아직 남은 과제

오픈 챗팅 병렬성 지원

TF-TRT을 통한 모듈 고도화

전처리 GPU 고도화

Q & A

Thank You