DEVIEW 2020

Rule의 역습:
NLU에서 머신러닝 기술을 보조/보완할 수 있는 정규표현식 언어 nlu_script

김용희 Clova AI Assistant

CONTENTS

1. 음성 대화 시스템과 NLU

1.1 음성 대화 시스템

Task마다 요구사항과 기술이 천차만별

Domain Experts	QnA	Open Chat	Voice Assistant
e.g. 식당 예약	정보 습득 및 검색	정확도보단 커버리지	명령, 컨트롤, 기능성
Dialog State 관리	Documents & DB	정보보단 흥미	Service API 연결

1.2 스마트 스피커에서 NLU

아이유 노래 틀어줘
상어가족 재생
내일 날씨 어때?
거실 불 켜줘
두부 사라고 메모해
네이버 주가 알려줘
엔화 환율
나훈아 나이 얼마야

사람의 말 → 서비스/기능 mapping

VIBE

쥬니어네이버

네이버 검색

16°
어제보다 1° 낮아요, 구름많음

NAVER 035420 >
295,000 ▲ 3,000 (+1.03%)
1일 3개월 1년 3년 10년 일봉 주봉 월봉
298,000
296,800
295,600
294,400
293,200
292,000
10:00 12:00 14:00

IoT

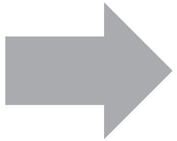
1.2 스마트 스피커에서 NLU

DEVIEW 2020

서비스에서 실용적/현실적인 NLU 문제 정의

- Sentence labels 추출/분류: domain, intent
- Slots 추출 및 정규화

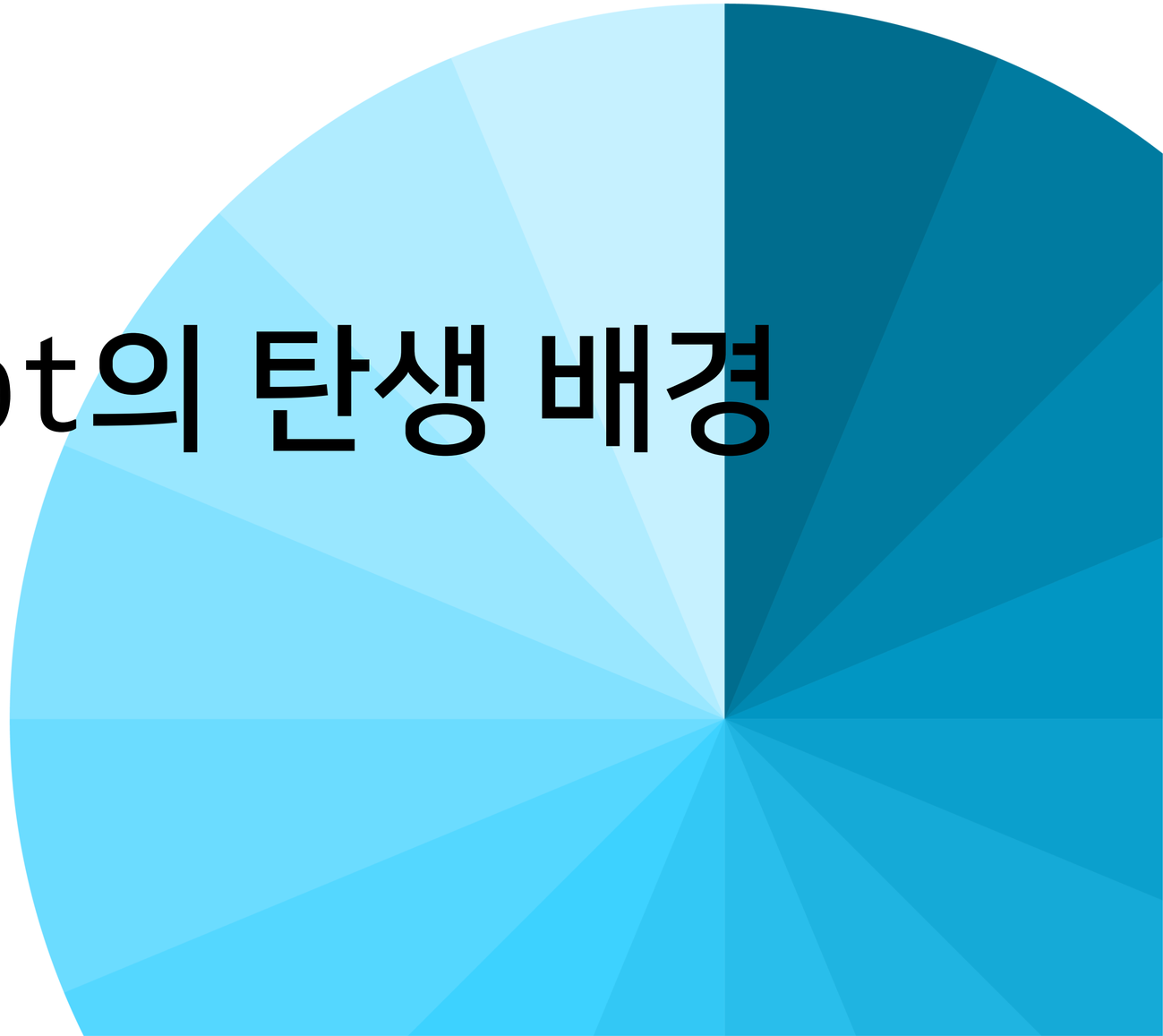
다섯시 반에 티비 꺼줘
5:30에 TV 끄자
5시 30분에 텔레비전 전원 꺼
5시 반에 테레비 끄기
다섯시 삼십분에 티브이 종료해



Sentence Label: control:turnoff

Slots: <datetime hour="5" minute="30">
<device name="TV">

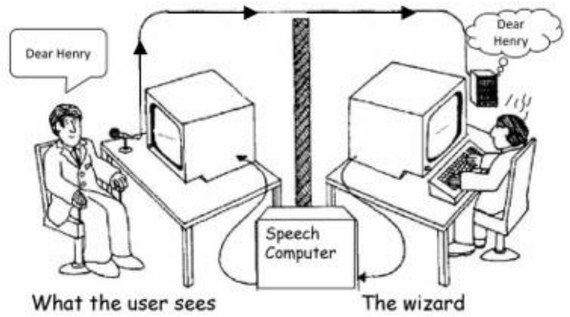
2. nlu_script의 탄생 배경



2.1 NLU 프로토타입은 머신러닝부터

요구사항 수집, Intent category 및 slot types 정의, 말뭉치 제작

Wizard of Oz testing – The listening type writer IBM 1984



User: 갑자기 음악에 대한 대화를 계속 하려니까 힘든데?
그러면은 어: 예를 들어서-
System: 잘 안 들려요.
System: 뭐라고 말씀하셨어요
User: 기다려.
System: 네 주인님.
예를 들어 이런 대화를 ( ) 심쿵해를 부른 가수가 누구야 아니면은
이런:: 가린 부분을 지금은 내가 모르는데, 어떻게 하면 알 수 있을까 이런 시나리오거든요 (…)
User: 심쿵해를 부른 가수가 누구냐
Sytstem: AOA입니다.

```
User: 아이유
System: 가수 아이유의 곡 금요일에 만나요 (Feat. 장이정 of HISTORY) 재생합니다.
User: 그거 말고
System: 곡 금요일에 만나요 (Feat. 장이정 of HISTORY) 어떠세요?
User: 그거 말고 라니깐
System: 곡 좋은 날 어떠세요?
User: ok
System: 가수 ok의 곡 쎄요쎄요 (Narr. 김기욱) 재생합니다.
User: ㅠㅠ
System: 적절한 대화 예제를 찾지 못했습니다. 비슷한 표현으로 다시 말씀해주세요.
```

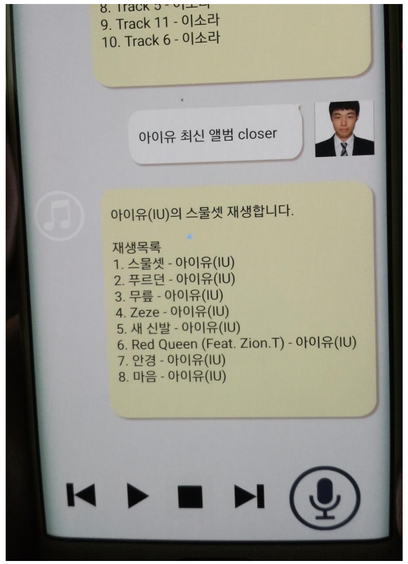
NLU 0.1.4 버전

오류

- 아이유를 SONG 으로 잡는경우
  - 아이유 노래 재생해줘
- 좋은날을 SINGER 로 잡는경우
  - 좋은날 틀어줘

- **8/4 UI & MG define 결과**
  - UI: 13개
  - MG: 22개

#	User Intentions	Main Goal
1	add	album
2	delete	album.name
3	down	app
4	download	artist
5	help	artist.age
6	move	agency
7	play	artist.date



2.2 control 도메인을 구원해준 GF

DEVIEW 2020

“다음 곡”, “소리 꺼” 같은 짧은 발화가 오히려 의도 분류 오류가 많음

- 기분석(using key-value DB) 사용
- Grammatical Framework 도입

```
concrete FoodEng of Food = {
        lincat
                Phrase, Item, Kind, Quality = {s : Str} ;

        lin
                Is item quality = {s = item.s ++ "is" ++ quality.s} ;
                This kind = {s = "this" ++ kind.s} ;
                That kind = {s = "that" ++ kind.s} ;
                QKind quality kind = {s = quality.s ++ kind.s} ;
                Wine = {s = "wine"} ;
                Cheese = {s = "cheese"} ;
                Fish = {s = "fish"} ;
                Very quality = {s = "very" ++ quality.s} ;
                Fresh = {s = "fresh"} ;
                Warm = {s = "warm"} ;
                Italian = {s = "Italian"} ;
                Expensive = {s = "expensive"} ;
                Delicious = {s = "delicious"} ;
                Boring = {s = "boring"} ;
}
```

	종결어미 inflections					명사형 전성어미
	V1	V2	V3	V4	V5	NV
1	고 싶	_아_	줘	어	요	음
2	을 수 있	_어_	줘봐	네	다	기
3	ㄹ 수 있	_해_	줄래	군	라	ㅁ
4			주실래	은데	자	
5			주겠	ㄴ 데	니	
6			주세요	거든	죠	
7			주면 돼	지	주십시오	
8			주면 좋겠			
9			줄 수 있나			
10			줄 수 있을까			
11			봐			

2.3 날짜, 시간, 숫자 정규화의 노하우

DEVIEW 2020

Schema 설계 고생 ㅠㅠ

의외(?)의 발견: rule 기반으로도 99%에 가까운 정규화 성능

2020년 가을

datetime

- year="2020"
- season="FALL"

열흘 동안

duration

- days="10"

3월 2일 동안

duration

- month="3"
- day="2"

4달라

money

- value="4"
- currency="USD"

다음달 마지막주 월요일 오후 3시에

datetime

- month="+1"
- week_in_month="LAST"
- week_day="1"
- part_of_day="PM"
- hour="3"

매일 밤 11시 반 이후에

set

- days="1"
- part_of_day="NIGHT"
- hour="11"
- minute="30"
- mod="AFTER"

2.4 foma fst 발견

DEVIEW 2020

Foma: finite state transducer 오픈소스 라이브러리

- Xerox regex와 호환되는 스크립트 문법 지원

```
foma[0]: def ANIMAL [고 양 이]:CAT ¦ [강 아 지 ¦ 개]:DOG;
defined ANIMAL: 567 bytes. 6 states, 7 arcs, 3 paths.
foma[0]: regex ANIMAL+;
599 bytes. 6 states, 10 arcs, Cyclic.
foma[1]: apply
down  med   up
foma[1]: apply down
apply down> 고양이
CAT
apply down> 고 양이
???
```

sentence label 추출에 적합한가? → X

slot 추출에 적합한가? → X

slot 정규화에 적합한가? → 일부 적용 가능

GF보다 나은가? → 문법은 더 직관적

2.5 nlu_script 프로젝트 시작

DEVIEW 2020

Requirements

- Regex match가 이루어지면 sentence label을 추출해야 한다.
- Slot type and range 정보를 추출해야 한다.
- Slot에 하나 이상의 attribute를 정의하고 정규화할 수 있어야 한다.
- 한국어에 맞게 띄어쓰기는 암묵적으로 허용하되 필요한 경우 명시할 수 있어야 한다.
- 비개발자도 쓸 수 있도록 문법이 직관적이어야 한다.
- 정규식을 재사용할 수 있어야 한다.

...foma로는 견적이 안 나오는데? Transducer를 정규화에 쓰고는 싶지만...

새로운 syntax를 정의하고 parser부터 개발하지 않으면 안 되겠는데.

이게 가능은 한가? 잘 모르겠으니까 일단 개인 프로젝트로...

3. Machine Learning vs. Rule

3.1 머신러닝 기술의 문제점

1. 데이터 구축의 어려움
2. 긴 훈련 시간
3. 검증 및 테스트의 어려움
4. 특수화 비용 문제
5. Out of domain 문제
6. Wildcard slot 훈련 문제

3.1.1 ML Cons. 데이터 구축의 어려움

DEVIEW 2020

- Schema 설계: 통계적 모델의 동작 방식을 모르면 헤매기 십상
- 너무 복잡한 말뭉치 태깅 가이드: 숙련에 최소 2주 이상 소요
- 말뭉치가 많아질수록 수정/관리 비용도 기하급수적으로 증가

오늘 날씨 좋아 → weather로 태깅

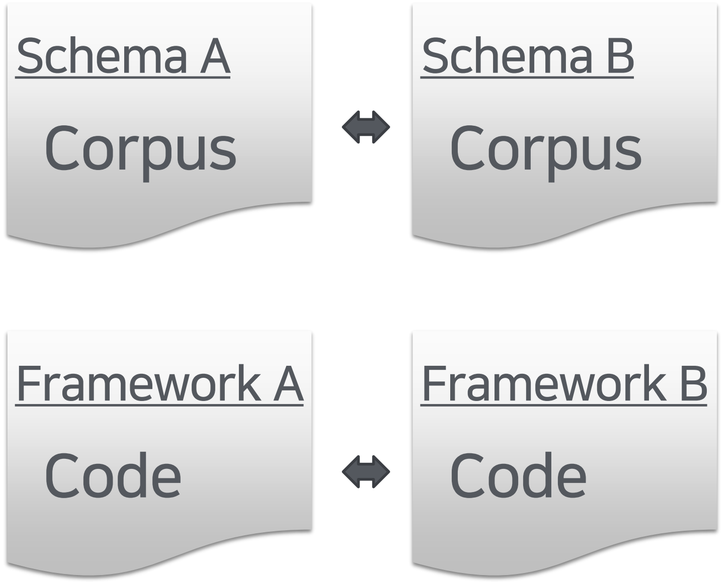
오늘 날씨 좋네 → chat으로 태깅

유재석 나이 → “유재석”을 person으로 태깅

유재석이 부른 노래 → “유재석”을 musician으로 태깅

분당 서울대병원 → “분당 서울대병원“을 하나의 slot으로 태깅

분당 김치찜 → “분당”과 “김치찜”을 별도의 slot으로 태깅



서로 호환이 될까...?

3.1.2 ML Cons. 긴 훈련 시간

DEVIEW 2020

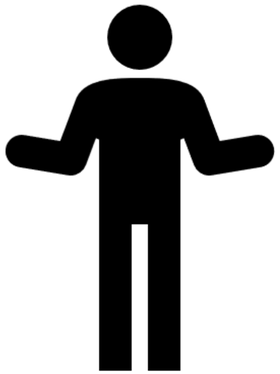
최소 반나절, 길게는 하루에 가까운 시간 소요

- 훈련 실수는 곧 야근과 서비스 출시 지연으로 연결
- 안정적 운영을 위해 배포 주기를 늘림: B2B 요구사항 대응 속도에 문제

3.1.3 ML Cons. 검증 및 테스트의 어려움

랜덤성: 재훈련 뒤 AB test를 하면 다량의 diff 발생

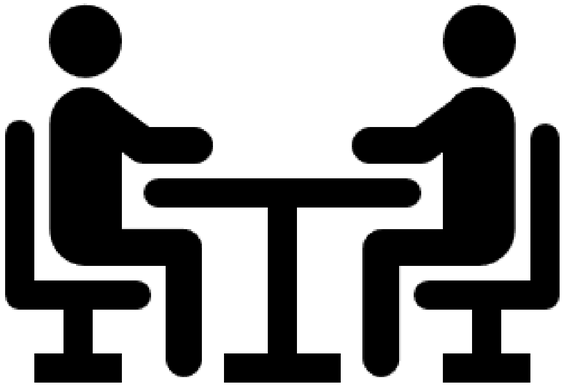
- Diff는 사람이 직접 보면서 좋아진 것, 나빠진 것 비율 측정
- 모델 개선, 말뭉치 추가/변경 등 모든 작업에서 병목
- test case 유지보수 비용이 큼 → 규모 확대가 어려움 → 안정성, 신뢰성 확보 불가



어제부터 “상어가족 틀어줘”가 갑자기 안 돼요

3.1.4 ML Cons. 특수화 비용 문제

이번에 L사에서 출시할
‘C-device’에다
“빵야빵야”라는 명령을
추가해달라고 하던데요.

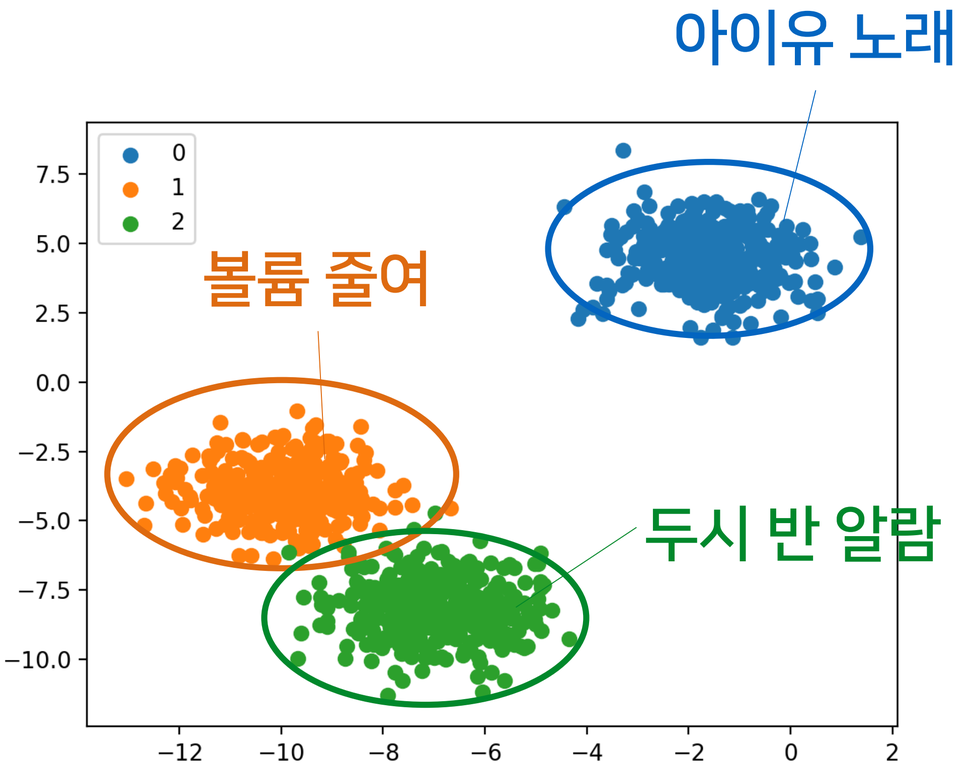


“빵야빵야” 발화에 대한
intent label을 정의해서
말뭉치 추가하고 훈련할게요.
준비 기간은 3주정도...?

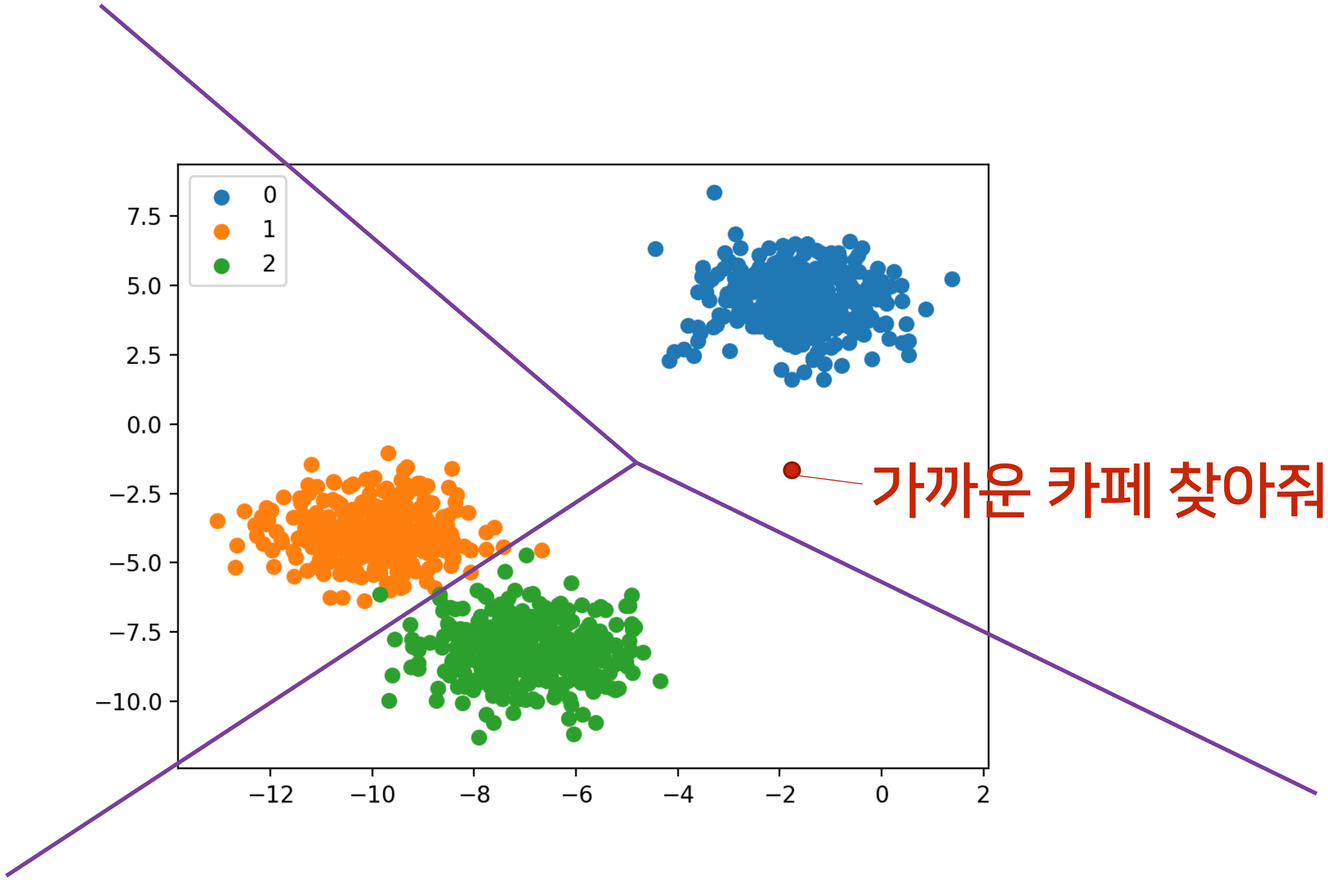
아 근데 다른 intent와
충돌 나서 성능 저하가 보이면
추가 조정이 필요할 수 있어요.

데이터 밸런스 문제, schema 충돌 문제 등등...

3.1.5 ML Cons. Out of domain 문제



이상 (ideal)



현실 (reality)

3.1.6 ML Cons. Wildcard slot 훈련 문제

DEVIEW 2020

메모, 메시지 보내기, 번역 등의 도메인

- 훈련 말뭉치에 넣어도 문제, 안 넣어도 문제, 빠짐없이 다 넣기도 어렵고...

10월 14일 날씨 좋았다고 메모해줘	→ 날씨?
양배추 사기 메모해놔	→ 쇼핑?
핑크퐁 상어가족이라고 메모	→ 음악?
스피커 전원 안 꺼진다고 메모해둬	→ 컨트롤?
7PM의 오후 7시의 메모 재생	→ 메모?

3.2 Rule 기술의 문제점

1. 양 날의 검, Wildcard
2. 유지보수의 어려움
3. No Robustness
4. 중의성, 중의성, 중의성!!!

3.2.1 Rule Cons. 양 날의 검, Wildcard

- Wildcard를 안 쓰면: 처참한 수준의 커버리지 (=low recall)
- Wildcard를 쓰면: 처참한 수준의 신뢰도 (=low precision)

```
if re.match(‘.*날씨.*’, query):
    return ‘weather’
```

검정치마의 날씨 틀어줘
십센치의 10월의 날씨
이별하기 좋은 날씨 재생
네일의 날씨 영업시간

3.2.2 Rule Cons. 유지보수의 어려움

Regex와 전처리/후처리 코드가 뒤섞인 스파게티 코드가 되기 십상

예시: memo slot 어미 정규화

(코드 상태가 많이 [illegible]하여 pseudo code로 대체합니다)

```
morphs = 형태소 분석 결과물
slots = Regex search 결과물

if slots[0]이 “memo” 내용:
    if slots[0]의 어미가 “라고” or “기” or “게”:
        if 어미와 일치하는 형태소가 존재:
            if 어미의 앞의 형태소가 동사 or 접미사:
                slots[0] 정규화 값의 끝에 “기”를 붙인다
        elif 어미가 “기” or “게”:
            return None
```

3.2.3 Rule Cons. No Robustness

- 음성인식 오류: 특히 open-set slot, open-domain 발화
- 오타: 빈도가 높은 오타는 일일이 대응할 수 있다지만...
- Unseen slot: 사용자가 노래 제목을 정확히 모르면? Wildcard를 써볼까?

Pattern: {musician}(의) {song} 틀어줘

- 강산에의 강물을 거슬러 오르는 연어 틀어줘 (원제: 거꾸로 강을 거슬러 오르는 저 힘찬 연어들처럼)
- 핑크퐁 아기상어 틀어줘 (원제: 상어가족)

Pattern: {musician}(의) .* 틀어줘

- 비밀의 화원 틀어줘 (원래 의도: 아이유가 부른 “비밀의 화원“ / 분석 결과: 가수 “비밀”의 노래 “화원“)
- 화사 뮤직비디오 틀어줘 (원래 의도: 뮤직비디오 재생 / 분석 결과: 가수 “화사”의 노래 “뮤직비디오“)

3.2.4 Rule Cons. 중의성, 중의성, 중의성!!!

- Search: 한글 2음절 이하에서 특히 큰 중의성
- Match: 컨텐츠명, 지명 등 open-set slot의 중의성 문제

Datetime patterns
- 엔진<오일> 가격
- 도착<한 시>각 알려줘
- 방<이초>등학교

Pattern: {song} 틀어줘
- 선풍기 틀어줘
- 에어컨 틀어줘
- TV 틀어줘

Pattern: {name}(이) (학교) 소식
- 태풍 소식
- 가을 소식

4. nlu_script 소개 및 시연

4.1 문법 및 기능 소개, 시연

DEVIEW 2020

nlu_script: ko/sent1 Document — Upload Download

```
def 볼륨_A [볼륨 ¦ 소리 ¦ 음량 ¦ 사운드] (크기) ;
def 높여줘_A [올 리다_REQ_A ¦ 높 이다_REQ_A ¦ 키 우다_REQ_A]
    ¦ 업(하다_REQ_A ¦ 시 키다_REQ_A)
    ¦ 위로(하다_REQ_A)
    ¦ [높¦크]게 (하다_REQ_A) ;
def 낮춰줘_A [내 리다_REQ_A ¦ 낮 추다_REQ_A ¦ 줄 이다_REQ_A](줘)
    ¦ 다운(하다_REQ_A ¦ 시 키다_REQ_A)
    ¦ 아래로(하다_REQ_A)
    ¦ [낮¦작]게 (하다_REQ_A) ;

def VOLUME_S <volume value=[1¦2¦3¦일:1¦이:2¦삼:3] (unit=[단계¦레벨]:LEVEL)
    ¦ value=[한:1¦두:2¦세:3] unit=[단계¦레벨]:LEVEL
    ¦ value=[10¦20¦30¦십:10¦이십:20¦삼십:30] unit=[퍼¦프로¦퍼센트¦"%"]:"PERCENT"> ;

sent up.volume 볼륨_A (VOLUME_S (만큼¦씩)) (좀) 높여줘_A ;
sent down.volume 볼륨_A (VOLUME_S (만큼¦씩)) (좀) 낮춰줘_A ;
```

Compile

```
              Parse: 74.522 ms
 Extract SlotPattern: 3.920 ms
Transform Expression: 65.642 ms
          Build FST: 296.849 ms
compile ko/sent/1_script done

볼륨 두단계 올려줘
up.volume:볼륨 <volume value="2" unit="LEVEL">두단계</volume> 올려줘

음량 30프로 키워줘
up.volume:음량 <volume value="30" unit="PERCENT">30프로</volume> 키워줘
```

Input

4.1 문법 및 기능 소개, 시연

DEVIEW 2020

Rule의 단점인 low-recall, 생산성 문제를 상당 부분 해결

- 서비스에 필요한 신뢰도, 안정성은 기본적으로 Rule이 더 뛰어남
- 4~5어절 이내의 발화에 대해서는 ML기술의 훌륭한 대체재/보완재

External resources와 연계 가능: Slot DB, PoS-tags

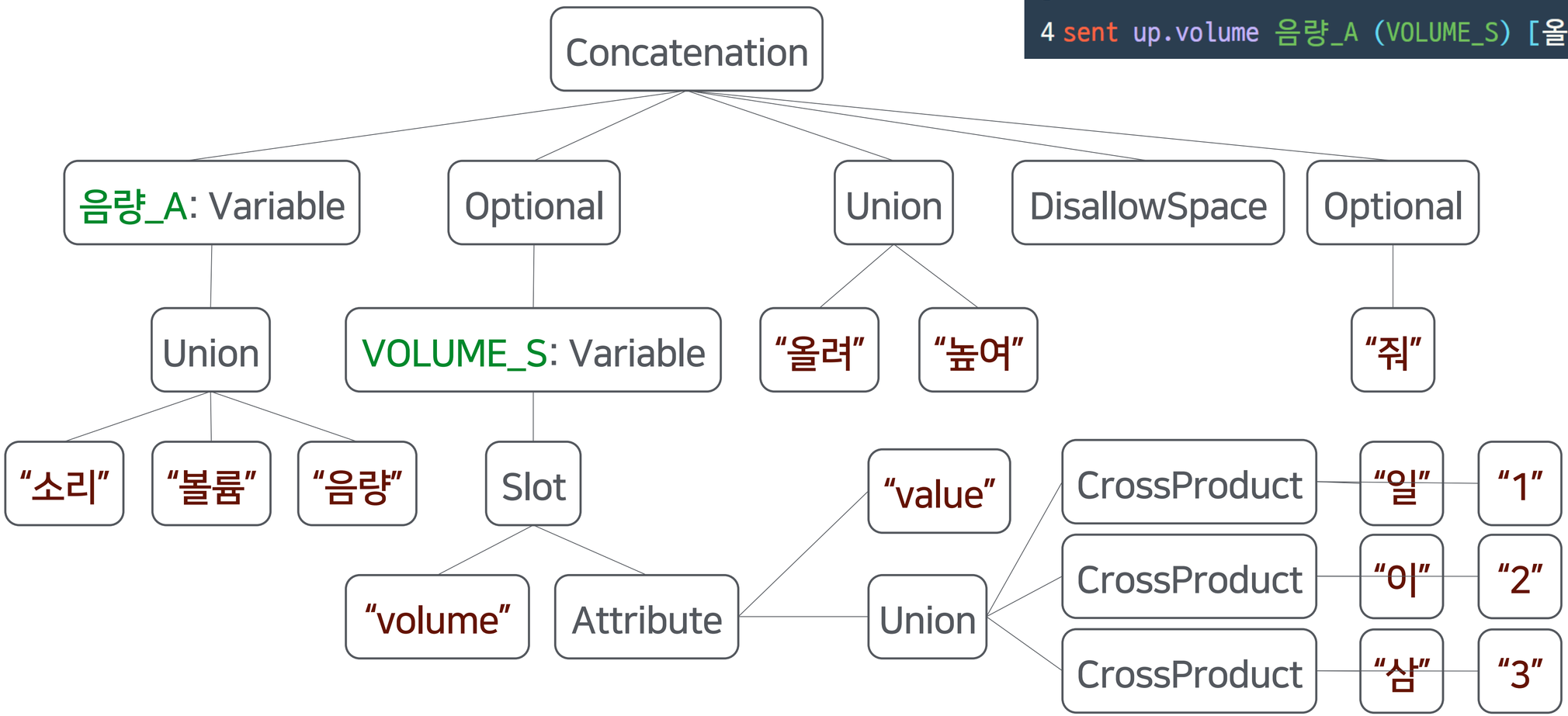
- Regex는 대량의 개체명을 관리하기에 부적합
- Regex와 DB는 lifecycle이 다르다: 관리는 따로, 분석시 연동은 필수
- External resource match를 염두에 두고 regex 설계/구현

4.2 설계 및 구현: Regex Compiler

DEVIEW 2020

Parsing

```
1 def 음량_A 소리¦볼륨¦음량 ;
2 def VOLUME_S <volume value=[일:1¦이:2¦삼:3]> ;
3
4 sent up.volume 음량_A (VOLUME_S) [올려¦높여]::(줘) ;
```

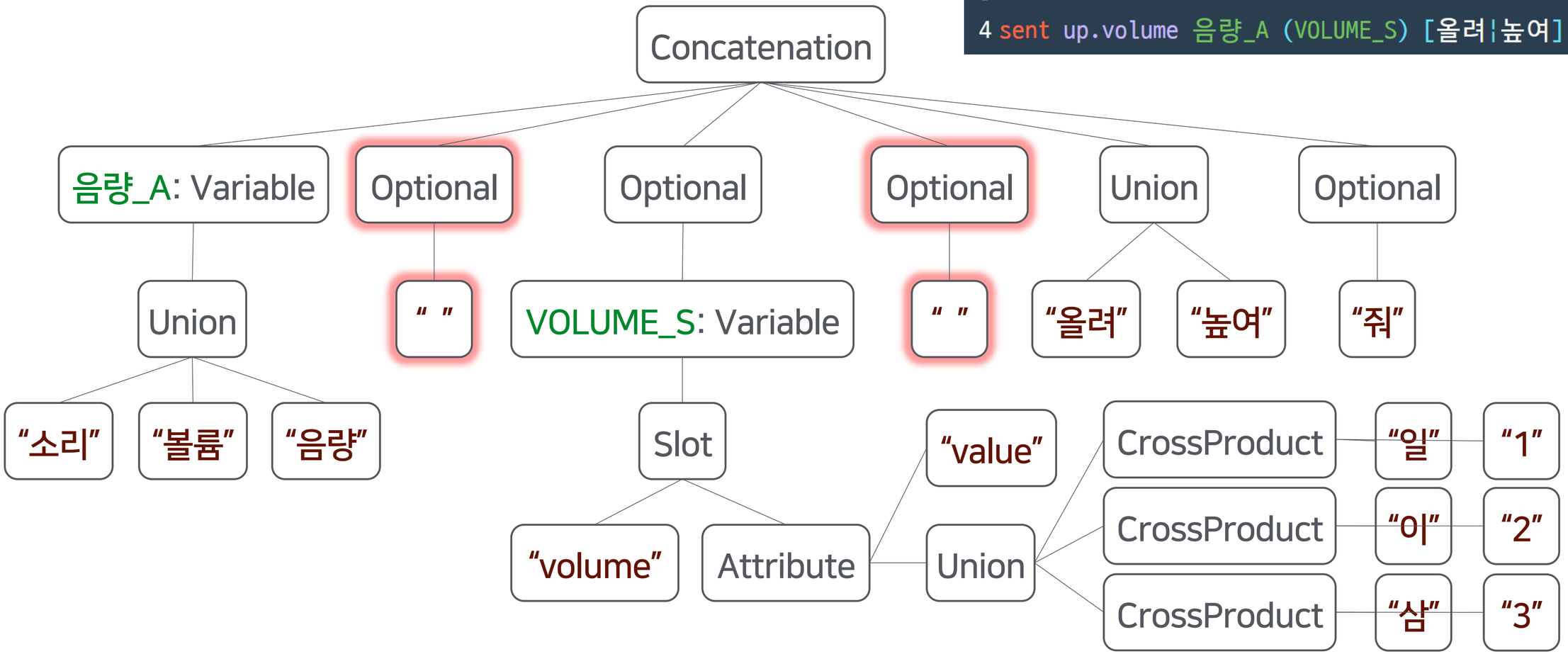


4.2 설계 및 구현: Regex Compiler

DEVIEW 2020

Transform(1)

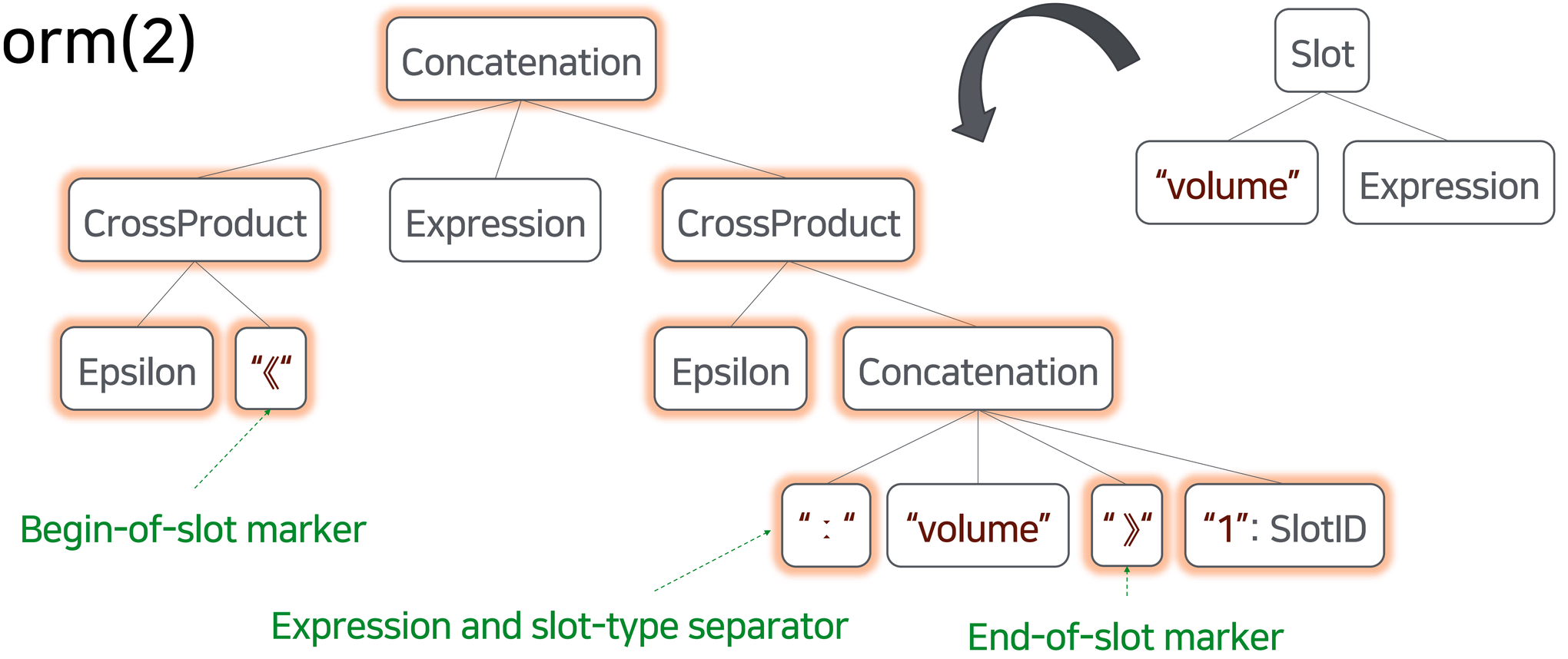
```
1 def 음량_A 소리¦볼륨¦음량 ;
2 def VOLUME_S <volume value=[일:1¦이:2¦삼:3]> ;
3
4 sent up.volume 음량_A (VOLUME_S) [올려¦높여]::(줘) ;
```



4.2 설계 및 구현: Regex Compiler

DEVIEW 2020

Transform(2)



Regex: `<volume 1|2|3>`

Input: “3” → Transformed: `[]:“《 ” [1|2|3] []:“ : volume 》1”` → Output: “《 3 : volume 》1”

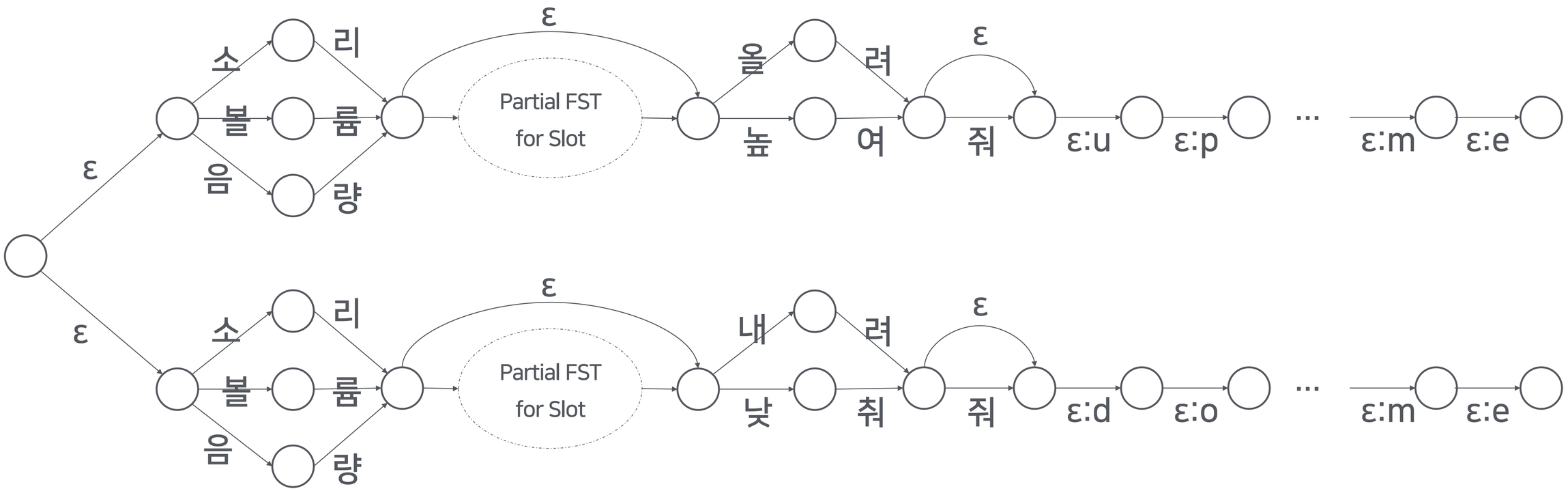
4.2 설계 및 구현: Regex Compiler

DEVIEW 2020

FST construction

- transformed tree는 FST-convertible

```
1 def 음량_A 소리¦볼륨¦음량 ;
2 def VOLUME_S <volume value=[일:1¦이:2¦삼:3]> ;
3
4 sent up.volume 음량_A (VOLUME_S) [올려¦내려]::(줘) ;
5 sent down.volume 음량_A (VOLUME_S) [내려¦낮춰]::(줘) ;
```



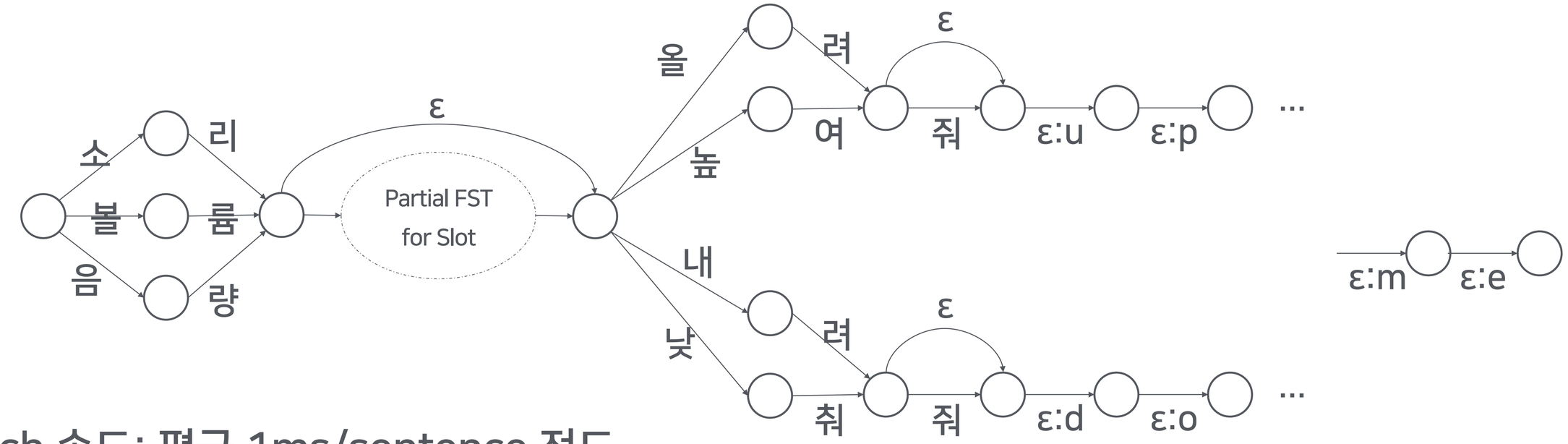
4.2 설계 및 구현: Regex Compiler

DEVIEW 2020

FST minimization

- match/search 속도 증가

```
1 def 음량_A 소리¦볼륨¦음량 ;
2 def VOLUME_S <volume value=[일:1¦이:2¦삼:3]> ;
3
4 sent up.volume 음량_A (VOLUME_S) [올려¦내려]::(줘) ;
5 sent down.volume 음량_A (VOLUME_S) [내려¦낮춰]::(줘) ;
```



match 속도: 평균 1ms/sentence 정도

(조건: import문을 사용한 수백~1천 줄 가량의 nlu_script regex)

4.2 설계 및 구현: match and search

DEVIEW 2020

Input to FST-input, FST match/search, FST-output to Ouptut

```
vector<Result> analyze(query, externalSlots, posTags);
```

Call `analyze` function with arguments:

- query = "텔레비전 켜줘"
- externalSlots = [{
  type: "device",
  beg: 0,
  end: 4,
  attrs: {"name": "TV"}}]
- posTags = []

FST-input: "{device:name=TV}4 켜줘"

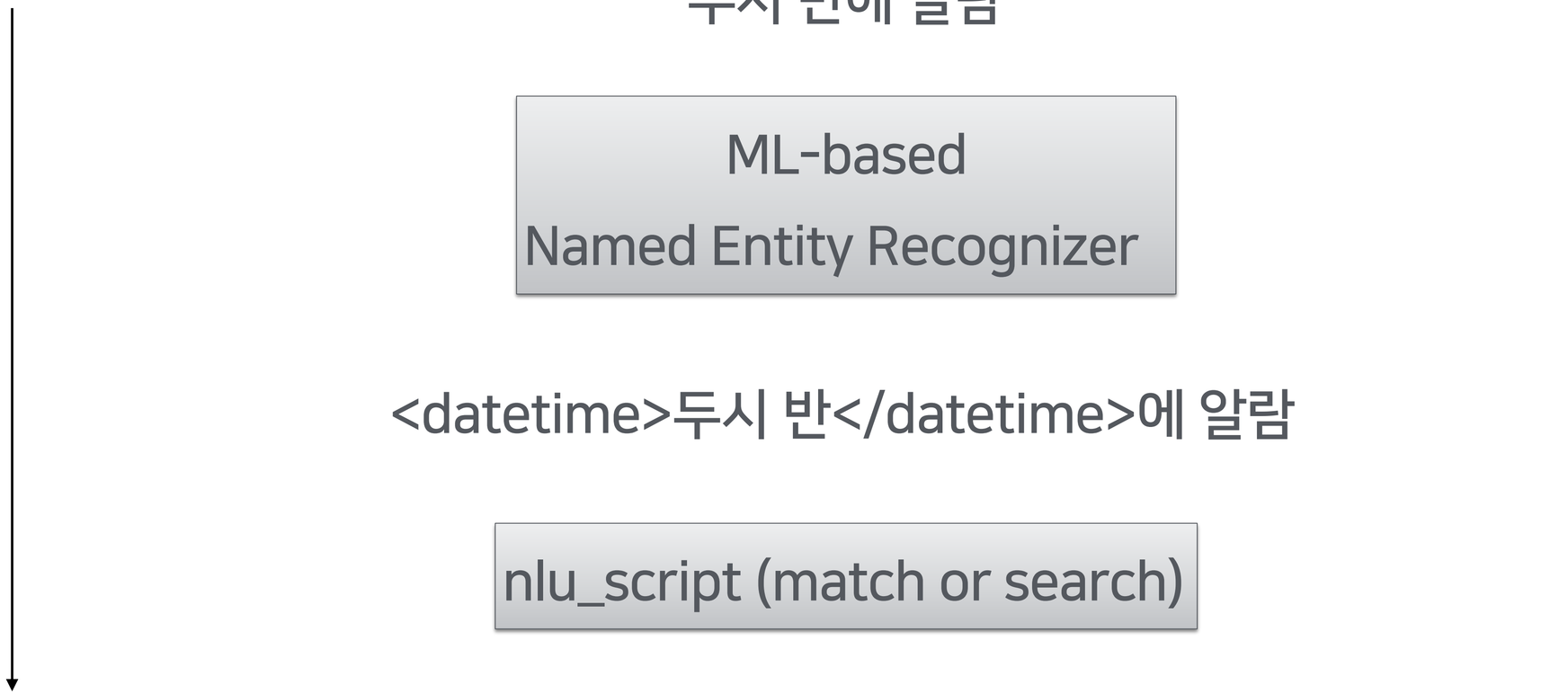
Minimized FST

```
1 from predicate import *
2
3 sent turnon.device {device} (좀) 켜다_REQ_A ;
```

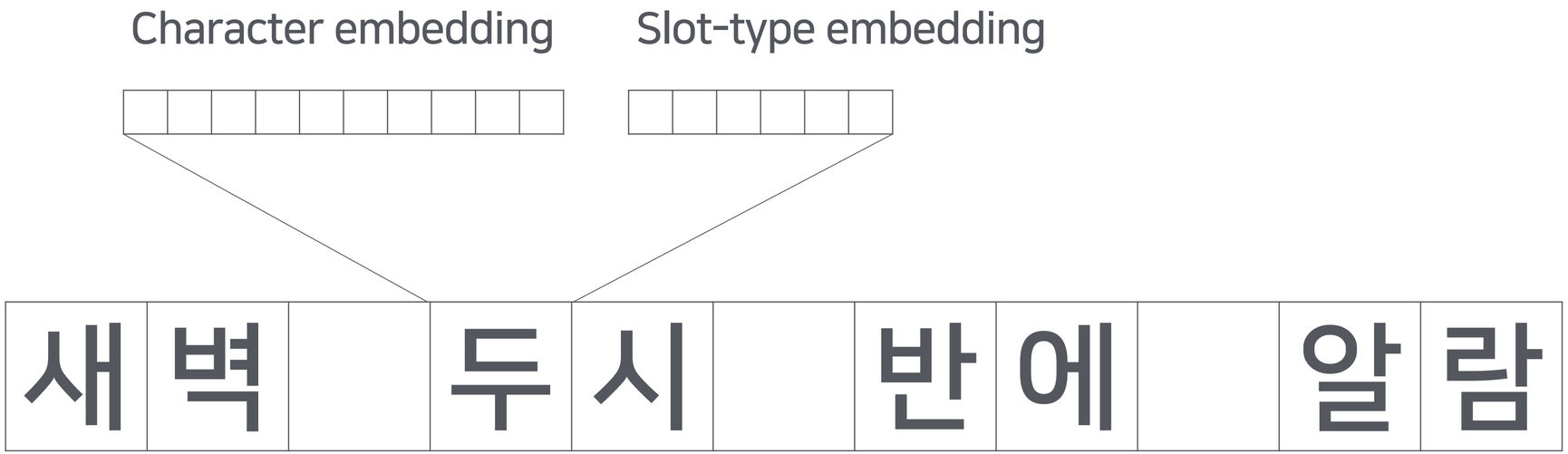
FST-output: "{device:name=TV}4 켜줘:turnon.device"

5. 서비스 활용 사례 정리

5.1 날짜, 시간, 숫자 추출 및 정규화



5.2 Slot을 추출하여 ML-feature로 활용



← datetime (from nlu_script) →

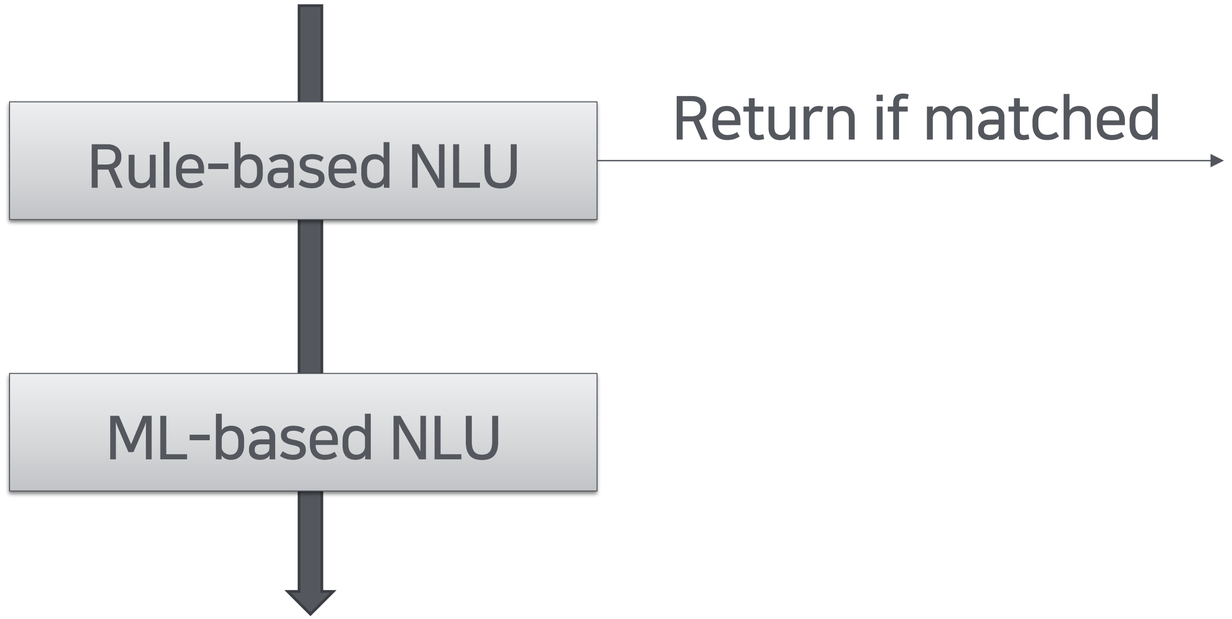
← musician (from DB) →

문제마다 다르지만, 경험적으로 1~3%까지 성능 향상 기대 가능

5.3 Hybrid NLU: Rule and ML

High-precision 필요

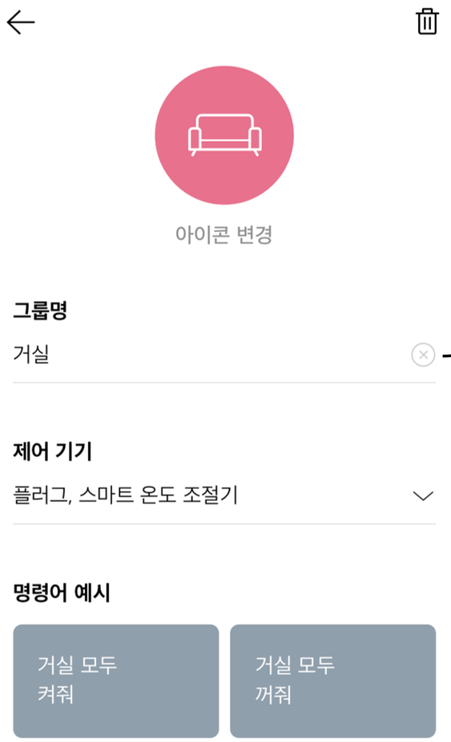
짧은 고빈도 발화 위주



5.4 사용자 정의 기기명/그룹명 분석

DEVIEW 2020

ML로 풀기 어려운 문제 (Wildcard 문제와 비슷한 이유)



{user_device} type external slot

→ NLU input으로 전달

```
1 from predicate import *
2
3 sent turnon.device {user_device}(의¦을¦를) (전원)(좀) [켜다_REQ_A ¦ 시작(하다_REQ_A)] ;
4 sent turnoff.device {user_device}(의¦을¦를) (전원)(좀) [끄다_REQ_A ¦ 종료(하다_REQ_A)] ;
```

2018.6. 국내 스마트 스피커 중 최초로 사용자 정의 기기명 지원

5.5 Wildcard 활용: 메모, 번역, 메시징

DEVIEW 2020

메모 도메인은 어미/조사 정규화까지 nlu_script가 담당

- 당시 A경쟁사의 음성비서는 “양배추 사“ 와 같이 다소 어색한 표현을 기록
- 클로바는 “양배추 사기”로 메모

<memo value=“양배추 사“ ending=“기“>양배추 사라고</memo> 메모해줘

5.6 비개발자도 NLU 개발에 참여

DEVIEW 2020

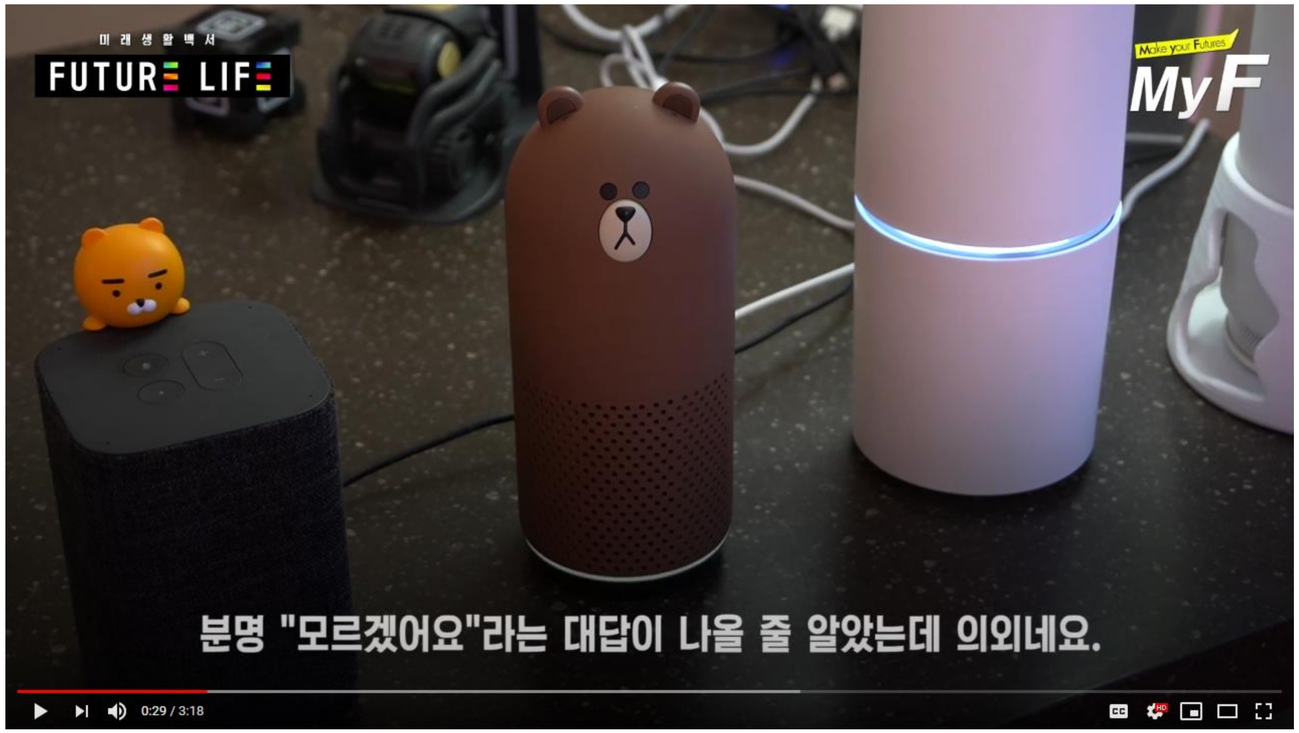
- Schema 설계, 데이터 밸런스/일관성 문제, OOD문제를 걱정할 필요가 없음
- 학습 난이도: 포토샵 정도? 쉽게 쓰려면 쉽게 쓸 수 있는 도구
- NLU팀으로 일이 몰리는 병목현상 완화
- 기획자 및 3rd party의 자유로운 창작이 가능

  Torreta bot builder: `nlu_script` + DM + NLG

5.7 이벤트성 도메인 개발

DEVIEW 2020

월드컵, 광복절 이벤트, 명절 컨텐츠, 코로나19 정보 등



코로나19 정보의 경우는 2020.3. 초반,
국내에서 경쟁사 대비 가장 빠른 대응

(개발해주신 기획자 분들께 감사드립니다)

5.8 특정 기기 전용 명령 추가

Skill 실행에도 invocation name 불필요 (OOD 문제 없음)

- ML-based skills: “클로바, xxx에서 yyyy해줘”
- Rule-based skills (in specific device): “클로바, yyyy해줘”

제휴사의 기기에 클로바를 탑재할 경우 특히 유용

- 클로바 기본 동작을 무시하고 일부 액션 덮어쓰기
- “홈가드”, “XR스쿨“ 등등 기기/서비스 고유의 기능이나 명령어 대응
- 초중등 학습기기, 호텔/아파트, 냉장고, 로봇청소기, …

5.9 NLP 서비스 전반

DEVIEW 2020

Voice assistant가 아니라도,
NLP 기술이 필요한 서비스라면 어디든

Thank You