Tave 챗봇 개발

-Kakao I open builder로 챗봇 만들기-

제4차 산업혁명 연구 동아리

TAVE

Kakao Open Builder를 사용한 챗봇 개발

개발자 : 강기웅, 김예원, 김혜영, 박명진, 안동원

목차

1. 서 론

1-1. 프로젝트 개발 환경 및 개발 목적

1) 프로젝트 개발 목적

기존 Tave는 밴드(Band) 어플을 사용하여, 동아리 정규 세션 장소, 활동사진, 동아리 활동 내역과 출석 점수 등 전반적인 동아리 활동에 대한 관리 및 소식을 전달하는 매개체로 사용했습니다. 또한 구글 드라이브는 Tave에서 스터디나 프로젝트 진행할 때, 필요한 자료 및 보고서, 커리큘럼 등을 올렸습니다.

그러나 Tave에서 진행한 설문조사 결과로 밴드(Band) 어플과 구글 드라이브의 사용이 불편하다고 나왔습니다. 구글 드라이브에서 불편한 점은 스터디 보고서를 제출할 때, 드라이브 주소를 일일이 입력하여 올려야 한다는 것이었고, 밴드(Band)에서 불편한 점은 목차 또는 메뉴가 없어서 글을 찾는데 힘들다는 것이였습니다.

이러한 설문 결과를 듣고, 저희 조는 밴드와 구글 드라이브를 보완할 방법으로 챗봇을 선택했습니다. 대화하듯 질문하면, 인공지능이 대화를 분석하고 그에 맞는 해답을 주는 대화형 메신저 챗봇(Chatter Robot)이 동아리 회원분들이 궁금했던 질문들이나, 필요했던 부분을 충족시킬 수 있다고 생각하였기에 개발하게 되었습니다.

2) 프로젝트 개발 환경

프로젝트 개발 환경은 카카오 오픈 빌더(Kakao I Open Builder)를 사용하였고, 서버는 Goorm IDE를 활용하였으며 언

어는 Python를 활용하여 구축하였습니다.

2. 본 론

2-1. 카카오 선택 이유

챗봇의 활용처를 선택할 때, 가장 많이 고려했던 것은 접근성과 활용성입니다. 다양한 메신저가 존재하지만, Tave 동아리원의 대다수가 카카오톡을 사용하기에 높은 접근성을 지니고, 카카오 자체 내에서 제공하는 오픈 빌더(Open Builder)가 있기에 높은 활용성을 보였기에, 챗봇의 활용처로 카카오를 선택했습니다.

2-2. Intent 추가

간단한 기술은 카카오톡 오픈빌더에서 제공하는 기능으로 구성하였습니다. 챗봇을 사용하는 사용자의 의도(intent)를 응대하는 가장 작은 단위를 블록이라고 하는데, 이 다양한 블록이 모여 시나리오를 이루어지게 됩니다. 저희 조는 시나리오를 여러 개 만들어서, 다양한 상황에 대해서 대응할 수 있게끔 만들었습니다.

시나리오는 크게 TAVE소개, QnA, 서버연동 등의 기능들로 구성했습니다.

TAVE 소개 시나리오에서는 사용자들이 챗봇을 통해 이용할 수 있는 기술, TAVE 동아리 소개, 스터디 및 활동내용에 대해 설명해주는 역할을 합니다.

Q&A는 출결점수 관리 부분이나, TAVE 활동하면서 필요한 역량 등 동아리 활동하면서 궁금했던 내용을 운영진 분들의 답변을 토대로 TAVE 동아리 회원 또는 동아리에 가입을 희망하는 대학생들에게 답변해주는 시나리오입니다.

[Tave 소개 시나리오]

[Q&A 시나리오]

2-3. Goorm IDE, Python 선택 이유

정해진 답이 아닌 실시간으로 정보를 받아오고 그것을 Tave 회원들에게 제공하는 도구로 Groom IDE와 Python을 사용했습니다. 다양한 클라우드 통합 개발 환경이 있지만, 팀원들과 동시 편집과 빠른 커뮤티케이션이 가능한 Groom IDE를 사용했습니다.

개발 언어로는 문법이 간결하고 수많은 라이브러리로 다양한 용도에 맞춰 확장시킬 수 있는 Python을 선택했습니다. 라이브러리 중 오픈 빌더와 서버를 연동하기 위해 Flask를 사용했습니다.

2-4. 서버 연동 후, 스킬 추가

서울에 살고 있는 다수의 회원들을 위해 챗봇에게 날씨를 물어보면 크롤링을 통하여 서

울의 현재 기온을 알려주는 날씨 시나리오와 에어코리아 등 다양한 오픈 API를 활용하여 미세먼지의 농도를 알려주는 시나리오 등 다양한 시나리오를 만들었습니다. 뿐만 아니라, 이미지 보내는 기본 방식으로 랜덤 주사위 시나리오도 추가하였습니다.

```python
@app.route("/dice", methods=["POST"])
def dice():
    ## 주사위 숫자 랜덤으로 제공
    RandomDice = random.randint(1, 6)
    if RandomDice == 1:
        DiceUrl = '주사위 1면 이미지 URL'
    elif RandomDice == 2:
        DiceUrl = '주사위 2면 이미지 URL'
    elif RandomDice == 3:
        DiceUrl = '주사위 3면 이미지 URL'
    elif RandomDice == 4:
        DiceUrl = '주사위 4면 이미지 URL'
    elif RandomDice == 5:
        DiceUrl = '주사위 5면 이미지 URL'
    else:
        DiceUrl = '주사위 6면 이미지 URL'

    ## 오픈 빌더에 이미지 보내는 기본 규칙
    res = {
        "version": "2.0",
        "template": {
            "outputs": [
                {
                    "simpleImage": {
                        "imageUrl": DiceUrl,
                        "altText": "주사위입니다."
                    }
                }
            ]
        }
    }
    return jsonify(res)
```

[주사위 시나리오 코드]

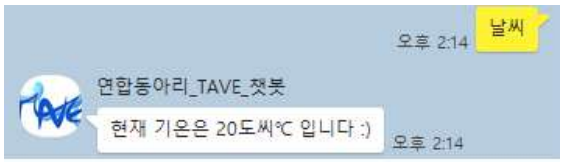
[주사위 실행 결과]

```python
@app.route("/temperature", methods=["POST"])
def temperature():
    ## 네이버에서 현재 기온 크롤링 하여 제공하는 함수
    url = '''https://search.naver.com/search.naver?
    sm=top_hty&fbm=1&ie=utf8&query=서울날씨'''
    ## 크롤링 URL
    req = requests.get(url).text
    soup = BeautifulSoup(req, 'html.parser')
    temp = soup.select_one('.info_temperature').text
    temperature = "현재 기온은 " + temp + "입니다 :)"

    ## 오픈빌더에 메세지 전송을 위한 기본 규칙
    res = {
        "version": "2.0",
        "template": {
            "outputs": [
                {
                    "simpleText": {
                        "text": temperature
                    }
                }
            ]
        }
    }

    return jsonify(res)
```

[서울 날씨 시나리오 코드]



[서울 날씨 실행 결과]

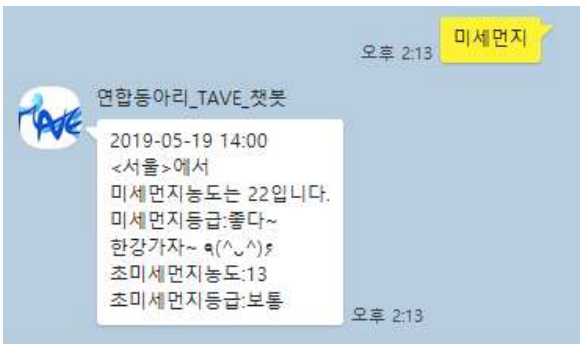
```
@app.route("/dust", methods=["POST"])
def dust(): ## 에어코리아 오픈 api를 통하여 미세먼지 수치

    servicekey = '공공데이터 포탈에서 발급받은 키'
    url='openAPI URL' + servicekey
    response = requests.get(url).text
    soup = BeautifulSoup(response, 'html.parser')
    dust_info = soup.body.pm10grade1h.text

    if dust_info == '1':
        dust_result = '좋다~ 한강가자~ ٩(^‿^)۶'
    elif dust_info == '2':
        dust_result = '보통'
    elif dust_info == '3':
        dust_result = '나빠요:('
    else:
        dust_result = '매우매우매우 나쁘다'

    dust_info = ('{1}\n<{0}>에서 \n미세먼지농도는 {2}입니다.'.
          format('서울', soup.body.datatime.text,
                 soup.body.pm10value24.text,))
    res = {
        "version": "2.0",
        "template": {
            "outputs": [
                {
                    "simpleText": {
                        "text": dust_info
                    }
                }
            ]
        }
    }
    return jsonify(res)
```

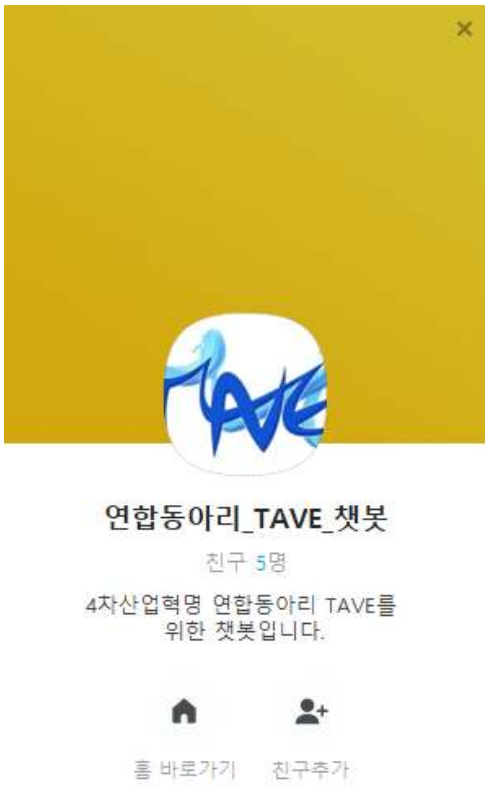
[미세먼지 시나리오 코드]



[미세먼지 실행 결과]

2-5. 배포

[Tave 챗봇 배포]



[플러스 친구와 연동 및 실행]

3. 결 론

3-1 문제점 및 느낀 점

챗봇을 개발하기 전, 구상해왔던 것과 실제로 구현할 수 있는 것은 다르다고 체감했습니다. 챗봇 개발 계획을 세우며, 구상했던 부분은 유저와 챗봇 간의 대화하는 형식을 예상했

으나, 키워드 기반의 답변과 단답형의 답변만이 개발 가능했습니다.

챗봇 개발에 사용했던 Groom IDE에서도 유료가 아닌 무료로 서버연동을 하였기 때문에, 서버를 항상 켜둘 수 없는 문제점과 다수의 컨테이너를 활용하지 못하는 한계가 있었습니다. 가장 아쉬웠던 점은 카카오톡 오픈빌더에 내장된 인공지능은 Diolog Flow등 다른 개발 플랫폼보다 아쉬운 성능을 보여, 회원들의 다양한 질문에 유동적으로 대답하지 못했습니다.

3-2 추후 개발

이번 프로젝트를 통해 개발된 Tave 챗봇은 앞서 말했듯이, 비용문제로 인해 무료로 서버연동을 했습니다. 추후에 더 개발을 더 하게 된다면, 서버를 항시 켜놓지 않아도 회원들이 다양한 기술을 사용할 수 있도록 하고, 지금의 기능 외에 정해진 답변이 아닌 대화형식의 기술, 음악 검색 기능, 사진으로 닮은 연예인 찾기와 같은 기능을 더 추가하고 싶습니다.

4. 참고 자료

<인터넷 자료>

- Dialogflow로 카카오톡 챗봇 만들기 (우종하)
- 카카오 I 오픈빌더로 피자 챗봇 만들기 (우종하)