

분류번호 : 2023-2-4105-06

졸업작품 최종보고서

Kakao open builder의 자연어 처리 알고리즘 기반 챗봇

지도교수 : 신 일 훈 교수님

제 출 일 : 2023년 11월 10일

제 작 자 : 18101577 한 태 섭

18101531 이 석 규

18101512 엄 태 훈

서울과학기술대학교 전자IT미디어공학과

목 차

┌ 요 약	i
표 목 차	ii
그림 목 차	ii
└ 사진 목 차	iii
I . 개요 [목표설정]	6
1.1. 제작 동기 및 목적	
1.2. 제작 목표	
1.3. 예상되는 제약 조건	
1.4. 설계 진행 일정	
II . 시스템의 설계 [분석, 합성]	8
2.1. 설계 진행	
2.1.1. 자연어 처리 모델 선정	
2.1.2. 설계 환경 설정	
2.2. 시스템 구성 요약	
III . 시스템의 제작 [제작]	11
3.1. kakao builder	
3.1.1. 시나리오 & 블록 설정	
3.1.2. 엔티티 설정	
3.2. skill data 제작	
3.2.1. web crawling	
3.2.1.1. 표 이미지 만들기	
3.2.2. json	
3.2.3. 랜덤 블록 추천	
IV .시스템의 동작 검증 [시험, 성능평가]	24
4.1. 실행화면 및 기존 챗미와의 비교	
4.2. 데모 결과	
4.3. 피드백	
V . 결론 [결과도출]	29
4.1. 실행화면 및 기존 챗미와의 비교	
4.2. 한계점과 극복방법	
VI . 참고 문헌	30
VIII . 감사의 글	31

요 약

제 목 : kakao openbuilder의 자연어 처리 알고리즘 기반 챗봇

챗gpt의 등장으로 다양한 방법으로 챗봇을 사용하고 있는 지금. 학교의 여러 정보를 간편하게 묻고 답할 수 있는 챗봇의 필요성을 느꼈다. 이에 카카오 오픈빌더에서 제공하는 NLU 기반 모델을 사용하여 챗봇을 만들었다.

파이썬을 기반으로 한 크롤링, 데이터 처리가 주를 이루고 사용자의 가독성을 위해 크롤링한 데이터를 따로 표로 만들어 실시간으로 표를 업데이트 하는 기능을 구현해보았다, 카카오 오픈빌더와 연결할 웹 구현으로는 flask를 사용하였다. 각 블록들이 호출 당했을시 그에 연결된 답변을 보내주는 식으로 코드를 작성하였다.

로컬 서버에서의 웹의 구현은 성공 하였지만 외부 접속이 가능하며 24시간 웹서버가 가동 되어야했기 때문에 naver cloud에서 제공하는 무료 서버를 받아 그 서버 안에서 웹을 구현했으며 외부 IP를 받아 외부에서도 접속이 가능하도록 하였다.

9월까지 생각했던 챗봇의 서비스 구현을 완료한 후 2회에 걸쳐 학생들을 대상으로 데모 테스트를 하였으며 그 결과 생각하지 못한 기능이나 매칭되지 못한 발화들을 패턴에 추가하여 답변율을 약 20퍼센트 상승 시킬 수 있었다.

표 목 차

표 1. 설계 진행 일정	7
---------------------	---

그 림 목 차

그림 1. dialogflow 시나리오 구성 화면	8
그림 2. dialogflow 와 kakao openbuilder 동작 예시	9
그림 3. 서버 구성도	9
그림 4. 시스템 구성도면	10
그림 5. 시나리오-1	11
그림 6. 시나리오-2	12
그림 7. 시나리오-3	12
그림 8. 블록 구성 화면	13
그림 9. 패턴 발화 관리 화면	13
그림 10. 파라미터 관리 화면	14
그림 11. 패턴 발화 관리 화면	14
그림 12. 제공 중인 응답 형태	15
그림 13. 응답 형태 (리스트 형, 텍스트 형)	15
그림 14. 응답 형태 (스킬데이터)	16
그림 15. 사용한 라이브러리 목록	16
그림 16. 라우팅 설정, flask 실행 코드	16
그림 17. 리스트 형태의 응답을 전달하는 코드	17
그림 18. 네이버 날씨 페이지 정보를 크롤링 하는 코드	18
그림 19. 표 생성 및 이미지화 코드-1	19
그림 20. 표 생성 및 이미지화 코드-2	20
그림 21. 부서 연락처 정보를 적은 json 파일	21
그림 22. json 파일에서 필요 정보를 추출하는 코드	21
그림 23. 블록들중 랜덤으로 추천하는 코드-1	22
그림 24. 블록들중 랜덤으로 추천하는 코드-2	23
그림 25. 데모 결과	27
그림 26. 피드백	28

사 진 목 차

사진 1. 인사	24
사진 2. 날씨	24
사진 3. 학식-1	25
사진 4. 학식-2	25
사진 5. 생활관 성과평가	26
사진 6. 건물 위치	26
사진 7. 연락처	27

I. 개 요

1.1. 제작 동기 및 목적

Chat GPT의 등장으로 단답형의 대답을 했었던 기존의 챗봇의 방식을 벗어나 인간과도 같은 대화가 가능한 Ai 기반 대화형 챗봇의 시대가 왔다. 이에 관심이 생겨 작품을 선정하게 되었다.

주제 선정 후 찾아보니 이미 서울과학기술대의 정보를 답변해주는 챗봇이 있다는 사실을 알게 되었다. 하지만 학교 학생들에게 설문조사 결과 기존 챗봇의 사용률이 그다지 높지 않다는 점을 알게 되어 조금더 학생 친화적이고 접근성이 좋게 만들고자 하였다.

1.2. 제작 목표

1. 서울과학기술대의 모든 정보를 제공하여 주기
2. 기존의 방식보다 접근성을 높이기
3. 학생 친화적인 정보를 제공하기
4. 단순한 단답형의 대답이 아닌 대화를 이끌기
5. 최신화가 자동으로 되도록 하기

1.3. 예상되는 제약 조건

데모를 진행할 때 되도록 많은 사람들이 사용하여 줄수록 생각 못한 경우의 수를 받을 수 있어 성능이 향상된다. 하지만 섭외 가능한 사용자들의 양에 한계가 있어 만족할 만큼의 데이터를 받을 수 있을지 미지수였다.

또한 목표가 ‘서울과학기술대의 모든 정보를 제공하는 챗봇’이었다 보니 입력 해야하는 정보의 양이 많기에 답변의 질에 있어선 어느정도 타협을 봐야 할 것이라 보았다.

1.4. 설계 진행 일정

월		3월	4월	5월	6월	7월	8월	9월	10월	11월
계획										
작품구상										
작 품 제 작	모델 선정									
	flask 서버 구현									
	기능 추가									
데모 및 설문조사										
업데이트										
보고서 작성										

표 1. 설계 진행 일정

II. 시스템의 설계[분석, 합성]

2.1. 설계 진행

2.1.1. 자연어 처리 모델 선정

다양한 플랫폼에서 제공하고 있는 챗봇 모델 등은 자연어 처리 기능 중 NLU (natural language understanding) 기능을 사용하며 주어진 텍스트에서 의미를 추출하고 해석하는 작업에 중점을 두는 공통점이 있다. 하지만 챗봇을 만드는 과정에서 각 모델들이 제공하는 서비스가 차이가 있었기에 이를 고려하여 사용할 모델을 선정하였다.

google cloud에서 제공하는 ‘dialogflow’ 와 카카오에서 제공하는 ‘kakao open builder’ 를 모두 사용해 본 결과 느낀 장단점이다.

(1) dialogflow

시나리오를 구성할 때 블록(dialogflow 에선 intent 라 지칭) 밑에 follow-up 블록을 설정하여 문맥을 이어 나갈 수 있다. follow-up 블록들은 상위 블록이 호출되지 않았을 경우 호출할 수 없기에 블록 간의 겹침 현상에 자유롭다.

하지만 학생들이 잘 사용하지 않는 메신저 또는 앱을 따로 만들어서 호환해야 하기에 접근성에 문제가 있다.



그림 1. dialogflow 시나리오 구성화면

(2) kakao openbuilder

시나리오 구성 시 블록의 레벨을 나누는 기능이 없어 완전 대화형 챗봇을 만들기엔 적절치 않으나 응답 방식으로 ‘버튼’을 제공하기에 조금 더 간편히 질문의 의도를 파악해 나갈 수 있다. 무엇보다 학생들이 주로 사용하는 메신저인 ‘카카오톡’과의 호환이 되었기에 kakao openbuilder를 사용하게 되었다.

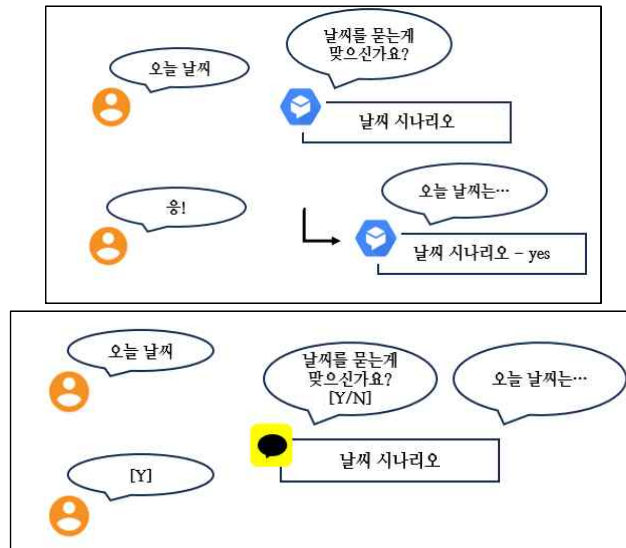


그림 2. dialogflow 와 kakao openbuilder 동작 예시

2.1.2. 설계 환경 설정

질문에 답변이 간단한 경우 kakao openbuilder에서 제공하는 응답 관리 항목을 이용하면 되지만 실시간으로 업데이트 되어야 하는 정보나 방대한 양의 정보들을 답변하기 위해서 답변들을 저장할 서버를 구축하였다.

navercloud에서 제공하는 무료서버를 받은 후 그 서버안에서 파이썬 언어 기반 웹 프레임워크인 flask를 사용하여 웹을 만들어 연동해주었다.



그림 3. 서버 구성도

2.2. 시스템 구성 요약

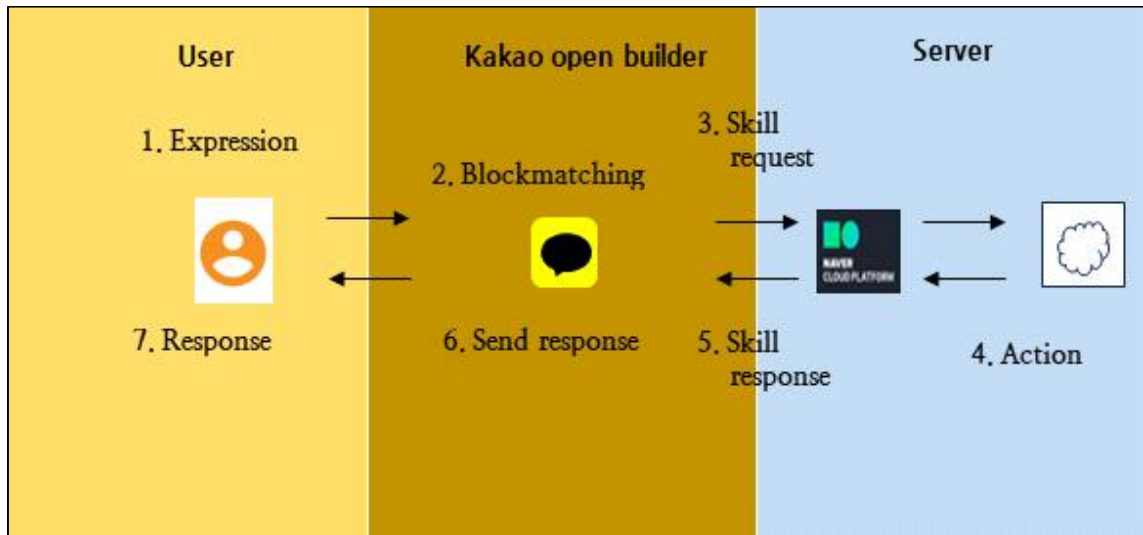


그림 4. 시스템 구성도면

시스템 구성도면이다. 사용자가 질문을 던지면 카카오톡 오픈빌더 api가 자연어 처리를 통해 블록을 매칭 시켜주고 만약 스킬 데이터가 필요한 응답일시 외부 서버에 응답에 필요한 여러 가지 정보 (parameter) 값들을 보내주게 되며 이에 반응하여 외부서버가 보내준 응답을 사용자에게 보여주는 식으로 동작하게 된다.

Ⅲ. 시스템의 제작[합성, 제작]

3.1. kakao builder

3.1.1. 시나리오 & 블록 설정

챗봇의 개발을 위해 카카오 오픈 빌더에서 제공하는 기능을 사용하여 사용자가 경험할 수 있는 큰 틀인 “시나리오”라는 서비스 단위를 작성하고 해당 시나리오를 구체적인 상황, 사용자의 의도에 응대할 수 있는 가장 작은 단위인 “블록” 들로 구성하였다.

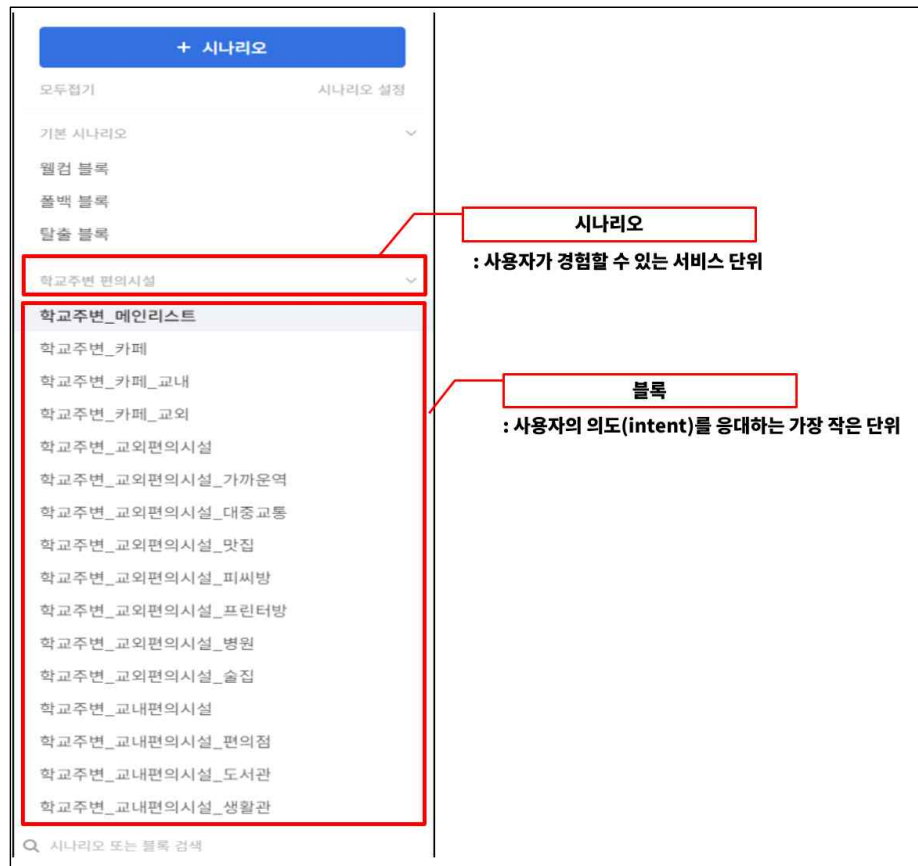


그림 5. 시나리오-1

챗봇의 틀을 잡기 위해 챗미를 토대로 시나리오의 도메인 범위를 결정하고 최종적으로 29개의 시나리오를 만들어 사용자에게 응답을 제공하였다.



그림 6. 시나리오-2

각 시나리오는 해당 시나리오 내부에서 사용자의 구체적인 의도에 맞는 약 800개의 블록들로 구성하였다. 블록은 사용자의 질문의 의도가 넓을수록 상위 블록, 구체적일수록 하위 블록으로 이루어져 있으며 블록간의 유기적인 연결을 통해 사용자의 발화에 보다 유연하게 응답 가능하도록 하였다.

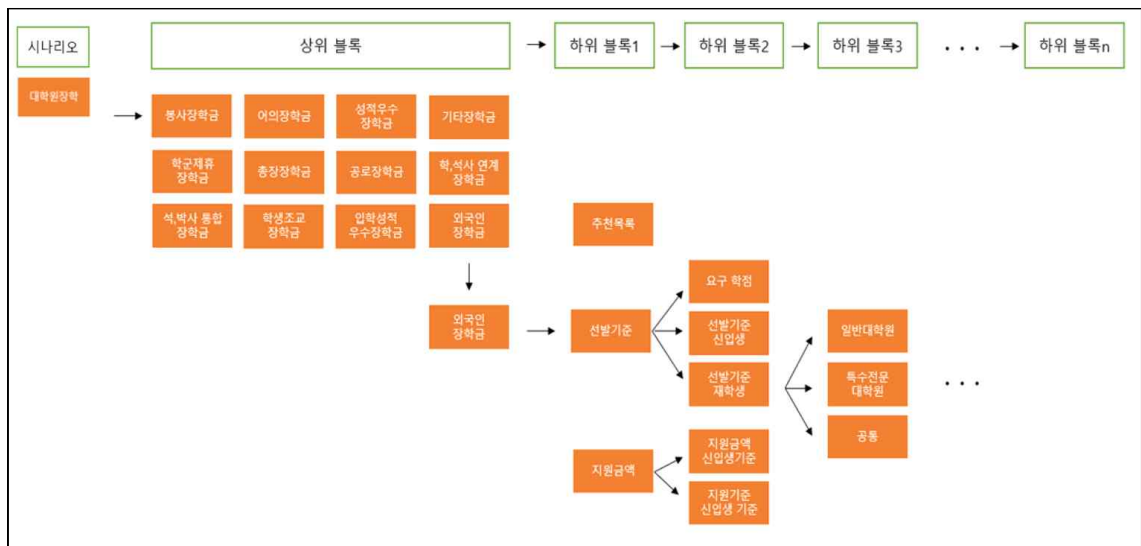


그림 7. 시나리오-3

챗봇의 전체적인 도메인을 결정한 후 카카오 빌더에서 제공하는 기능을 통해 시나리오의 이름을 구성 후 블록들을 추가하였다.



그림 8. 블록 구성 화면

사용자의 질문의 의도가 형성한 블록과 일치할 경우 해당 블록으로 진입하기 위해 사용자의 발화로부터 가져와야하는 파라미터를 정의하고 약 10 ~ 20개의 발화를 학습시켰다. 발화를 학습시키는 경우, 다양한 패턴을 입력할수록 다양한 발화에 응답할 수 있었으며 파라미터의 값들을 정의하여 의도에 맞지 않는 블록으로 진입하는 경우를 방지할 수 있었다.



그림 9. 패턴 발화 관리 화면

사용자가 특정 정보를 제공해야만 챗봇이 응답을 할 수 있는 경우(학과 연락처, 건물 위치 등) 필요한 정보를 필수 파라미터로 설정하여 해당 파라미터 값을 발화로부터 추출하지 못한 경우 되묻기 질문을 통해 정보를 받아오도록 설정하였다.



그림 10. 파라미터 관리 화면

3.1.2 엔티티

엔티티란 자연어처리 과정에서 문맥 안의 특정한 의미를 가진 식별 가능한 대상이나 개념을 의미한다. 엔티티를 설정하여 파라미터의 관리를 용이하게 하고 동의어, 유의어, 오타 등의 챗봇을 사용할 때 빈번히 발생하는 오류들을 다룰 수 있다. 학생들이 실제로 사용하는 용어는 형식적인 용어가 아니라 다양한 줄임말, 신조어 등을 엔티티 처리를 통해 챗봇이 인식하도록 할 수 있다. 본 작품에서는 총 13가지의 개념 및 대상을 엔티티 처리한 후 관리하였다.

나의 엔티티	
나의 엔티티 관리	검색어를 입력하세요
전공_1	공과대학,신소재공학과,기계자동차공학과,안전공학과,기계시스템디자인공학과,건설사...
건물이름_번호	서울과학기술대,대학본부,다산관,창학관,제2창업보육센터,혜성관,청운관,창조관,파워룸...
대학_부서	총장실,비서실장,비서실,교육부총장실,연구기획부총장실,교무학생팀,행정지원팀,실험실...
시작_학식	학식
건물이름_번호_2	제2학생회관,수림학사,성림학사,KB학사,서울테크노파크
와이파이_	와이파이
Webmail	웹메일
연락처_1	전화번호,연락
시작_장학	장학
장학_detail	대학원장학금,교내장학금,국가장학금

그림 11. 패턴 발화 관리 화면

3.1.3 봇 응답

기본적으로 카카오 오픈빌더에서는 사용자에게 다양한 유형의 응답 형식을 제공한다.



그림 12. 제공 중인 응답 형태

응답 형식은 텍스트, 이미지, 카드, 커머스, 리스트 스킴데이터 총 6개의 유형으로 구성되어 있다. 본 작품의 경우 텍스트, 리스트, 스킴데이터 위주로 응답을 구성하였다.

텍스트의 경우, 사용자에게 문자로만 구성된 정보를 제공한다. 총 1000자까지의 문자를 입력할 수 있으며 최대 3개의 버튼을 추가하여 버튼을 통해 다른 블록 혹은 URL 주소로 이동할 수 있다. 버튼을 추가하는 경우 최대 문자수는 400자로 제한된다.

리스트의 경우, 5개의 목록을 사용자에게 제공한다. 각 목록을 통해 다른 블록이나 URL 주소로의 접근이 가능하다.

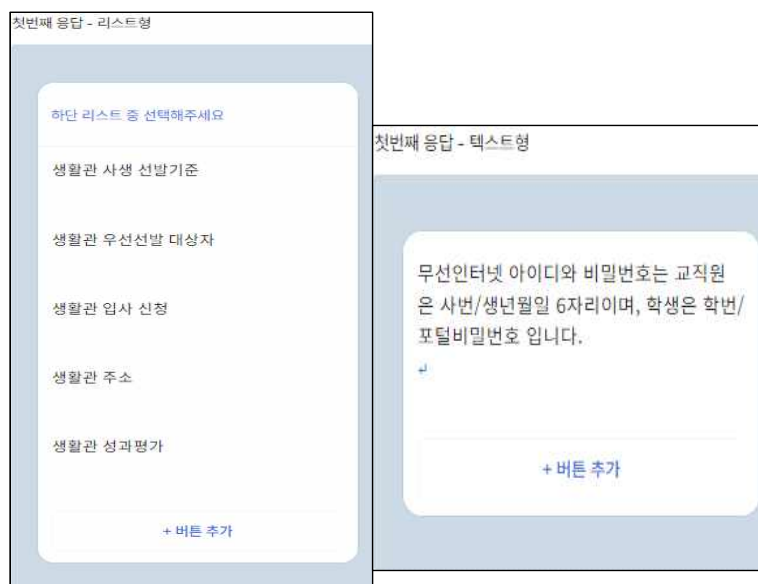


그림 13. 응답 형태 (리스트 형, 텍스트 형)

마지막으로 스킬데이터의 경우, 카카오 빌더에서 제공하는 응답 형식과 챗봇 제작자가 원하는 형식이 다를 경우, 코드로 응답을 구현한 후 카카오 빌더와 연동시켜 자유로운 형식의 응답을 제공할 수 있도록 한다.



그림 14. 응답 형태 (스킬데이터)

3.2. skill data 제작

우선 전체 코드에서 사용한 모듈은 위와 같이 구성되어져 있다.

```
import requests
from flask import Flask, request, jsonify
import json
from datetime import date, datetime, timedelta
import urllib
import ssl
from bs4 import BeautifulSoup
from pprint import pprint
import urllib.parse
from PIL import Image, ImageDraw, ImageFont
from create_table_image import create_table_image
import pandas as pd
import random
```

그림 15. 사용한 라이브러리 목록

main 코드는 웹 프레임워크인 Flask를 이용해 /webhook 엔드포인트에서 POST 요청을 처리하는 webhook 함수로 정의하였다. request 모듈을 통해 POST 요청에서 JSON 데이터를 강제로 파싱하여 사용자의 의도와 발화 정보들을 변수에 저장하고, flask를 실행시킨다.

```
@app.route('/webhook', methods=['GET', 'POST'])
def webhook():
    req = request.get_json(silent=True, force=True)
    user_result = req.get('userRequest')
    action = req.get('action')
    contexts = req.get('contexts')
    user_input = request.json['userRequest']['utterance']
    print(user_input)

#플라스크 실행 시키기
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

그림 16. 라우팅 설정, flask 실행 코드

사용자의 의도에 맞는 특정 블록으로 발화가 진입한 경우 해당 블록의 응답을 if문을 통해 파이썬으로 작성하여 res 변수에 카카오 오픈 빌더의 버전과 플랫폼에서 제공하는 json 파일의 형식에 맞춰 응답을 반환하였다. 아래의 경우 5개의 리스트를 제공하는 응답으로 리스트를 누르면 block의 고유 id를 통해 다른 블록으로 이동하여 사용자가 원하는 정보를 찾아갈 수 있도록 구현한 코드이다.

```
# 장학금
if user_result.get('block')['name'] == '장학금-시작':
    res = {
        "version": "2.0",
        "template": {
            "outputs": [
                {
                    "listCard": {
                        "header": {
                            "title": "궁금한 내용을 선택해 주세요."
                        },
                        "items": [
                            {
                                "title": "Q. 장학금에 대해 알고 싶어요.",
                                "action": "message",
                                "messageText": "Q. 장학금에 대해 알고 싶어요.",
                                "action": "block",
                                "blockId": "646875066ce345751ec251e3",
                                "extra": {
                                    "발화": "1"
                                }
                            },
                            {
                                "title": "Q. 장학금 중 교내성적 추가 장학금은 없나요?",
                                "action": "message",
                                "messageText": "Q. 장학금 중 교내성적 추가 장학금은 없나요?",
                                "action": "block",
                                "blockId": "646875066ce345751ec251e3",
                                "extra": {
                                    "발화": "2"
                                }
                            },
                            {
                                "title": "Q. 장학금 중복으로 못 받나요?",
                                "action": "message",
                                "messageText": "Q. 장학금 중복으로 못 받나요?",
                                "action": "block",
                                "blockId": "646875066ce345751ec251e3",
                                "extra": {
                                    "발화": "3"
                                }
                            },
                            {
                                "title": "Q. 장학금 포기에 대해 알고 싶어요.",
                                "action": "message",
                                "messageText": "Q. 장학금 포기에 대해 알고 싶어요.",
                                "action": "block",
                                "blockId": "646875066ce345751ec251e3",
                                "extra": {
                                    "발화": "4"
                                }
                            },
                            {
                                "title": "Q. 장학금 기준이 어떻게 되나요?",
                                "action": "message",
                                "messageText": "Q. 장학금 기준이 어떻게 되나요?",
                                "action": "block",
                                "blockId": "646875066ce345751ec251e3",
                                "extra": {
                                    "발화": "5"
                                }
                            }
                        ]
                    }
                }
            ]
        }
    }
    return jsonify(res)
```

그림 17. 리스트 형태의 응답을 전달하는 코드

3.2.1. web crawling # 날씨

crawling을 통해 스킴 응답을 구성한 코드이다. 대표적으로 날씨, 학식, 생활관련 정보를 crawling을 통해 구성하였다.

사용자의 발화가 ‘날씨-예’ 블록으로 들어갔을 경우, 현재 날짜 및 시간을 datetime 모듈을 사용하여 설정하고 strftime 함수를 통해 원하는 형식으로 포맷하였다. urllib.request 모듈을 사용해 네이버 날씨 페이지를 열고 BeautifulSoup 모듈을 이용하여 HTML을 파싱하여 원하는 요소를 추출한 후 find 메서드로 원하는 요소를 추출하여 변수에 저장하였다. 이후 카카오 응답 형식에 맞춰 json 형태로 return하는 방식으로 코드를 작성하였다.

```
# 날씨
if user_result.get('block')['name'] == "날씨-예":
    now = datetime.now()
    nowDate = now.strftime('%Y년 %m월 %d일 %시 %분 입니다.')

    ful_text1 = nowDate
    ful_text2 = "오늘의 날씨 정보입니다.\n"

    # 네이버 날씨 크롤링
    context = ssl._create_unverified_context()
    webpage = urllib.request.urlopen('https://search.naver.com/search.naver?set=tab.hty_top&where=newsearch&query=%E4%B3%B5%E6%A8%B9%E8%99%E8%82%A0%E6%94%A8')
    soup = BeautifulSoup(webpage, 'html.parser')
    temps = soup.find('div', 'temperature-text')
    summary = soup.find('p', 'summary')
    ful_text3 = "공률동 " + temps.text.strip()
    ful_text4 = summary.text.strip()

    html = requests.get('https://search.naver.com/search.naver?where=newsearch&sm=top_hty&fbm=1&ie=utf8&query=%E4%B3%B5%E6%A8%B9%E8%99%E8%82%A0%E6%94%A8')
    #pprint(html.text)

    soup = BeautifulSoup(html.text, 'html.parser')

    data1 = soup.find('div', {'class': 'report_card_wrap'})
    #pprint(data1)

    data2 = data1.findAll('li')
    #pprint(data2)

    fine_dust = data2[0].find('span', {'class': 'txt'}).text
    ful_text5 = "미세먼지 농도:", fine_dust

    ultra_fine_dust = data2[1].find('span', {'class': 'txt'}).text
    ful_text6 = "초미세먼지 농도:", ultra_fine_dust

    UV_rays = data2[2].find('span', {'class': 'txt'}).text
    ful_text7 = "자외선:", UV_rays

    sunset = data2[3].find('span', {'class': 'txt'}).text
    ful_text8 = "일몰:", sunset

    answer = ''.join((ful_text1)) + "\n" + ''.join((ful_text2)) + "\n" + ''.join((ful_text3)) + "\n" + ''.join((ful_text4)) + "\n" + ''.join((ful_text5)) + ...
    "\n" + ''.join((ful_text6)) + "\n" + ''.join((ful_text7)) + "\n" + ''.join((ful_text8)) + "\n"

    res = {
        "version": "2.0",
        "template": {
            "outputs": [
                {
                    "simpleText": {
                        "text": answer
                    }
                }
            ]
        }
    }

    return jsonify(res)
```

그림 18. 네이버 날씨 페이지 정보를 크롤링하는 코드

3.2.1.1. 표 이미지 만들기

생활관 성과평과 결과와 인원현황 등의 정보 또한 crawling 방식을 통해 응답을 구현하였는데, 생활관의 정보는 대부분 단순 텍스트가 아닌 다양하게 병합된 셀들로 이루어진 표의 형식으로 정보를 제공하고 있어 단순히 crawling만으로는 사용자에게 적절한 응답을 제공하기 어려워 추출한 정보를 토대로 새롭게 표 이미지를 형성한 후 해당 이미지를 사용자에게 제공하도록 하였다.

crawling 방식과 응답 제공 방식 자체는 비슷하나 추출한 정보를 토대로 표를 구성하고 해당 표를 다시 이미지화하는 과정이 필요했기에 모듈을 작성하였다.

main 코드에서 crawling을 통해 table_data를 가져오면 해당 정보를 Pandas 모듈을 통해 데이터프레임으로 변환하고 데이터 프레임의 행과 열의 개수를 추출하였다. 다음으로 각 셀의 너비와 높이를 적절하게 설정하고 ImageDraw 모듈을 통해 이미지에 그림을 그릴 수 있는 객체를 생성하였다.

```
from PIL import Image, ImageDraw, ImageFont
import os
import pandas as pd

def create_table_image(table_data):
    # DataFrame으로 변환
    table_data = pd.DataFrame(table_data)

    # 테이블의 행과 열 개수 추출
    num_rows = len(table_data)
    num_cols = len(table_data.columns) if num_rows > 0 else 0

    # 셀의 너비와 높이 설정
    cell_width = 120
    cell_height = 40

    # 테이블 이미지의 너비와 높이 계산
    table_width = num_cols * cell_width
    table_height = (num_rows + 1) * cell_height + (len(table_data.columns) - 1) * cell_height # 헤더 행 추가

    # 폰트 경로 설정
    font_path = r"C:\Users\wtjrrb\source\repos\카오톡봇\카오톡봇\font\NanumGothic.ttf'

    # 테이블 이미지 생성
    table_image = Image.new('RGB', (table_width, table_height), color='white')
    draw = ImageDraw.Draw(table_image)

    # 폰트 설정
    font = ImageFont.truetype(font_path, size=10)
```

그림 19. 표 생성 및 이미지화 코드-1

for문을 통해 crawling한 표의 헤더 행을 구분하고 table_data의 행과 열 정보를 순회하면서 draw 모듈을 통해 선과 텍스트를 그려 표를 이미지에 그려 넣었다. rowspan과 colspan이 존재하는 경우 셀들을 병합처리하여 표의 정보가 왜곡되지 않도록 하였다. 최종적으로 그려진 table_image를 반환하는 모듈을 작성하였다.

```
# 테이블 데이터를 이미지에 그리기
for i in range(num_rows + 1): # 헤더 행 추가
    for j in range(num_cols):
        if i == 0: # 헤더 행
            cell_text = str(table_data.columns[j])
        else:
            cell_info = table_data.iat[i - 1, j]
            if isinstance(cell_info, dict):
                cell_text = str(cell_info.get('text', ''))
            else:
                cell_text = str(cell_info)

        # 셀의 좌상단 좌표 계산
        x = j * cell_width
        y = i * cell_height

        # 셀의 경계선 그리기
        draw.rectangle([(x, y), (x + cell_width, y + cell_height)], outline='black')

        # 셀에 텍스트 그리기
        text_width, text_height = draw.textsize(cell_text, font=font)
        text_x = x + (cell_width - text_width) // 2
        text_y = y + (cell_height - text_height) // 2
        draw.text((text_x, text_y), cell_text, font=font, fill='black')

merged_cells = set() # 병합된 셀의 인덱스를 저장할 집합

for i in range(num_rows):
    for j in range(num_cols):
        cell_info = table_data.iat[i, j]
        if isinstance(cell_info, dict):
            rowspan = int(cell_info.get('rowspan', 1))
            colspan = int(cell_info.get('colspan', 1))
        else:
            rowspan = 1
            colspan = 1

        # rowspan과 colspan이 1보다 큰 경우, 해당 셀을 병합합니다.
        if rowspan > 1 or colspan > 1:
            merged_cells.add((i, j)) # 병합된 셀의 인덱스를 저장합니다.

# 병합된 셀을 처리하는 부분
for i, j in merged_cells:
    cell_info = table_data.iat[i, j]

    if isinstance(cell_info, dict):
        cell_text = str(cell_info.get('text', ''))
    else:
        cell_text = str(cell_info)

    # 병합된 셀의 영역을 그리기 위해 좌상단 좌표와 우하단 좌표를 계산합니다.
    start_x = j * cell_width
    start_y = i * cell_height
    end_x = (j + colspan) * cell_width
    end_y = (i + rowspan) * cell_height

    # 병합된 셀 영역에 대한 직사각형을 그립니다.
    draw.rectangle([(start_x, start_y), (end_x, end_y)], fill='white', outline='black')

    # 병합된 셀에 텍스트를 가운데 정렬하여 그립니다.
    text_width, text_height = draw.textsize(cell_text, font=font)
    text_x = start_x + (end_x - start_x - text_width) // 2
    text_y = start_y + (end_y - start_y - text_height) // 2
    draw.text((text_x, text_y), cell_text, font=font, fill='black')

return table_image
```

그림 20. 표 생성 및 이미지화 코드-2

3.2.2. json

연락처 정보나 건물 정보 등 고정된 값이지만 정보의 양이 많은 경우 json 파일을 통해 데이터를 정리하여 관리하였다.

```

1  "contact": {
2
3
4      "출장실": {
5          "name": "출장실",
6          "number": "02-970-6000"
7      },
8
9
10     "비서실장": {
11         "name": "비서실장",
12         "number": "02-940-6003"
13     },
14
15     "비서실": {
16         "name": "비서실",
17         "number": "02-970-6001 or 6002"
18     },
19
20     "교육부출장실": {
21         "name": "교육부출장실",
22         "number": "02-970-9700 or 9701"
23     },
24
25     "연구기획부출장실": {
26         "name": "연구기획부출장실",
27         "number": "02-970-9400 or 9401"
28     },
29
30     "공과대학": {
31         "name": "공과대학",
32         "number": "한교무학생팀 : 02-970-6175 or 6182 or 6183#행정지원팀 : 02-970-6172-4#실업실습지원팀 : 02-970-6179 or 02-970-6579"
33     },
34
35     "교무학생팀": {
36         "name": "교무학생팀",
37         "number": "02-970-6175 or 6182"
38     },
39
40     "행정지원팀": {
41         "name": "행정지원팀",
42         "number": "02-970-6172-4"
43     },
44
45     "실업실습지원팀": {
46         "name": "실업실습지원팀",
47         "number": "02-970-6179 or 6579"
48     },
49
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

그림 21. 부서 연락처 정보를 적은 json 파일

연락처의 경우 if문을 통해 사용자의 발화가 특정 블록에 들어가면 어떤 학과 혹은 부서의 연락처를 사용자가 원하는지 챗봇이 알아야한다. 따라서 req를 통해 사용자의 발화에서 파라미터를 추출하여 해당 파라미터가 json 파일 안에 존재할 경우 데이터를 불러와 사용자에게 연락처를 제공하도록 코드를 구성하였다.

```
# 연락처 블록 화과
if user_result.get('block')['name'] == '연락처_학과':
    user_params = req['action']['params']
    major_name = user_params.get('전공_1')

    print(req)
    with open('연락처.json', 'r', encoding='utf-8') as f:
        data = json.load(f)

        contact = data.get('contact')

    if major_name in data['contact']:
        contact_info = data['contact'][major_name]
        contact_name = contact_info['name']
        contact_number = contact_info['number']

        res = {
            "version": "2.0",
            "template": {
                "outputs": [
                    {
                        "simpleText": {
                            "text": f"{contact_name}의 연락처입니다. 📞({contact_name}) : {contact_number}"
                        }
                    }
                ]
            }
        }

    return jsonify(res)
```

그림 22. json 파일에서 필요 정보를 추출하는 코드

3.2.3. 랜덤 블록 추천

사용자의 발화가 항상 의도에 맞는 블록으로 들어가지는 않는다. 발화 학습이 부족하거나 파라미터나 문맥을 챗봇이 이해하지 못한 경우 fall-back 블록이나 다른 의도의 블록으로 진입할 가능성이 있다. 이러한 경우를 방지하기 위해 사용자에게 챗봇이 제공할 수 있는 시나리오를 추천해줘서 사용자가 원하는 정보에 접근하기 쉽도록 랜덤으로 블록을 추천해주는 코드를 구현했다.

사용자가 챗봇에 처음 진입하거나 인사 블록으로 들어간 경우 해당 코드가 실행된다. 우선 각 시나리오의 최상위 블록 id와 해당 블록의 이름을 변수에 저장하고 block_ids에 모든 정보를 저장했다.

```
# 랜덤 리스트 출력
if user_result.get('block')['name'] == '인사' or user_result.get('block')['name'] == '환영 블록':

    block_id_연락처 = ['65215e1c4a2aa96a2977f465', '연락처 정보']
    block_id_학교주변 = ['652a4ac4da81f579cdd366fa', '학교주변 정보']
    block_id_생활관 = ['65065e22ed24787e55b1b274', '생활관 정보']
    block_id_학식 = ['64621e1b95f0716a5582bb8d', '학식 정보']
    block_id_와이파이 = ['646246e1954e43077f05bc8d', '와이파이 정보']
    block_id_웹메일 = ['64688d40a2844e2dbe389f58', '웹메일 정보']
    block_id_장학 = ['646873426a1cf7451400dc60', '장학 정보']
    block_id_휴학 = ['644f856c82e69401205aacc0', '휴학 정보']
    block_id_복학 = ['644fbffdb002148b56d36ec', '복학 정보']
    block_id_수강신청 = ['64bcd160d3906927d1ddc6bd', '수강신청 정보']
    block_id_전과 = ['65067258443c4c44dc7fc372', '전과 정보']
    block_id_졸업 = ['65100433e1d39518fa282c77', '졸업 정보']
    block_id_학사일정 = ['652a4086648eee21606e220d', '학사일정']
    block_id_학교정보 = ['652a54707a5bab3eee316c96', '학교정보']

    block_ids = [

        block_id_연락처,
        block_id_학교주변,
        block_id_생활관,
        block_id_학식,
        block_id_와이파이,
        block_id_웹메일,
        block_id_장학,
        block_id_휴학,
        block_id_복학,
        block_id_수강신청,
        block_id_전과,
        block_id_졸업,
        block_id_학사일정,
        block_id_학교정보

    ]
```

그림 23. 블록들중 랜덤으로 추천하는 코드-1

다음으로 random 모듈을 통해 block_ids 안에 있는 최상위 블록들 중 랜덤으로 5개를 선택하여 사용자에게 제공하도록 코드를 작성하였다.

```
selected_block_ids = random.sample(block_ids, 5)

block_0 = selected_block_ids[0][0]
block_0_title = selected_block_ids[0][1]

block_1 = selected_block_ids[1][0]
block_1_title = selected_block_ids[1][1]

block_2 = selected_block_ids[2][0]
block_2_title = selected_block_ids[2][1]

block_3 = selected_block_ids[3][0]
block_3_title = selected_block_ids[3][1]

block_4 = selected_block_ids[4][0]
block_4_title = selected_block_ids[4][1]

res = {
    "version": "2.0",
    "template": {
        "outputs": [
            {
                "listCard": {
                    "header": {
                        "title": "안녕하세요! 서울과학기술대학교 챗봇입니다!"
                    },
                    "items": [
                        {
                            "title": block_0_title,
                            "description": "",
                            "action": "block",
                            "blockId": block_0
                        }
                    ]
                }
            }
        ]
    }
}
```

그림 24. 블록들중 랜덤으로 추천하는 코드-2

Ⅳ.시스템의 동작 검증 [시험, 성능평가]

4.1. 실행화면 및 기존 챗미와의 비교

1) 인사말

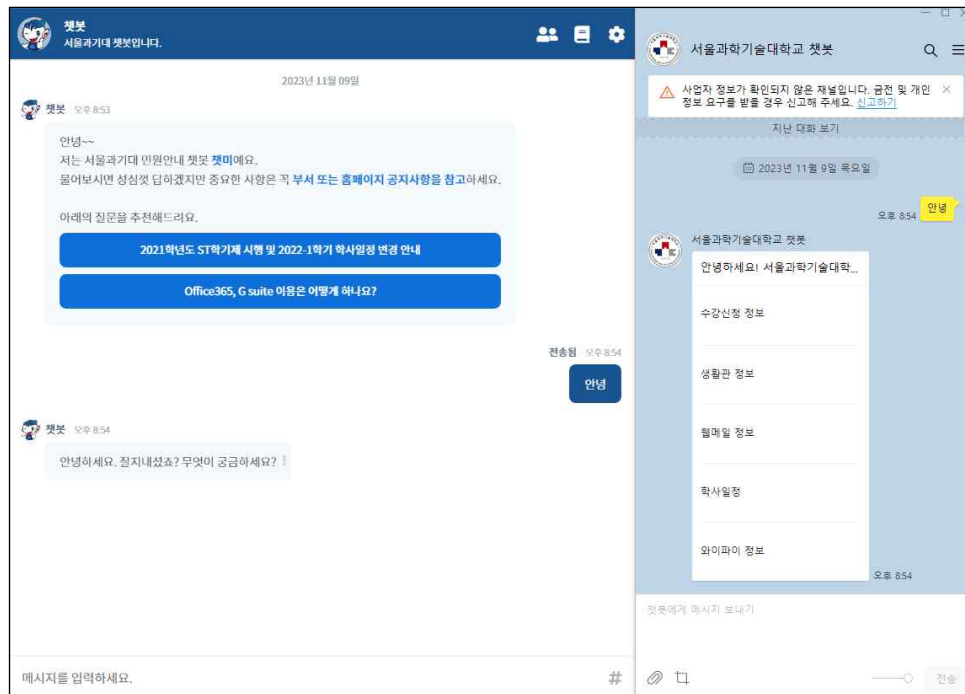


사진 1. 인사

2) 날씨

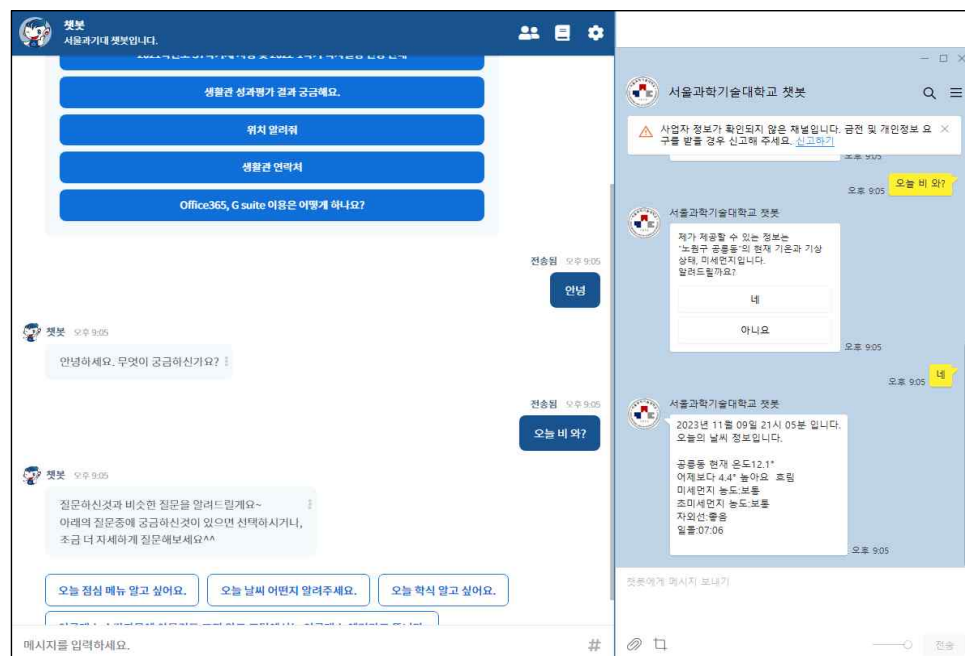


사진 2. 날씨

The image displays two side-by-side screenshots of a mobile application interface.

Left Screenshot:

- Header:** Blue bar with a profile icon, the name "챗봇" (Chatbot), and the text "서울게가네 챗봇입니다." (This is the Seoul Gega Chatbot).
- Chat History:** Shows previous messages from "전송됨" (Sent) at 9:05 AM, including a greeting and a question about login status.
- Current Message:** A light blue bubble asking if the user has logged in yet ("로그인하셨나요?").
- Bottom Bar:** Includes navigation icons and a list of categories: "학생 제1식당" (Student Cafeteria 1), "학생 제2식당" (Student Cafeteria 2), and "테크노파크(STP)" (Technopark). A button labeled "오늘 배워?" (Learn today?) is also visible.

Right Screenshot:

- Header:** Similar header to the left, but with a warning icon and text: "사실자 정보가 확인되지 않은 채널입니다. 금전 및 개인정보 요구를 받을 경우 신 고해 주세요" (This channel may contain false information as real identity is not confirmed. Please report immediately if you receive requests for money or personal info).
- Quiz Content:** A section titled "서울과학기술대학교 챗봇" (Seoul National University Chatbot Quiz) dated Nov 9, 2023. It contains a multiple-choice question about the difference in height between two floors (12.1m vs 4.4m) and a calculation problem involving floor area and volume.
- Options:** Below the question are four selectable options: "학생제2식당" (Student Cafeteria 2), "테크노파크" (Technopark), "KB학사" (KB Classroom), and "수업학사" (Classroom).
- Footer:** A yellow banner asks "오늘 학습 뭐야?" (What's your study today?).

챗봇
서울과학기술대학교 챗봇.

챗봇 오후 9:07

아래의 식당 중에 하나를 선택해주세요. 🍽️

식단표

2023년 11월 9일

<p>점심</p> <p>일반식 5,000원 편당알밥 유부창국,미니돈까스,열무나물,배추김치</p> <p>푸링 special 5,000원 돼지고기감지찌개 뽕밥,멘지깨소,새송이버섯볶음,열무나물,배추김치</p> <p>간단 snack 3,000원~ 라면 계란라면(3,000원), 떡계란라면(3,300원), 치즈계란라면(3,300원), 물만두계란라면(3,300원), 공깃밥(500원)</p>	<p>다른 식당 메뉴 보기</p>
--	---------------------------

서울과학기술대학교 챗봇

⚠️ 사용자 정보가 확인되지 않은 채입니다. 관련 및 개인정보 요구를 받을 경우 신
교학주세요. [링크하기](#)

오후 9:07 **오늘 확실히 봐?**

서울과학기술대학교 챗봇

아래의 식당 중 하나를 선택해주세요.

학생제2식당

테크노파크

KB학사

아래의 식당 중 하나를 선택해주세요.

수집학사

성림학사

오후 9:07

오후 9:07 학생제2식당

서울과학기술대학교 챗봇

2023년 11월 09일
점심
일반식 : 찹쌀알밥유부창국 미니돈까스 열무나물 배추김치 5,000원
푸링 special : 돼지고기감지찌개뽕밥 멘지깨소 새송이버섯볶음 열무나물 배추김치 5,000원
간단 snack : 라면계란라면(3,000원) 떡계란라면(3,300원) 치즈계란라면(3,300원) 물만두계란라면(3,300원) 공깃밥(500원) 3,000원~ 저녁
간단 snack : 닭강정마요덮밥 라면떡볶이마요덮밥(5,000원) 계란라면(3,000원) 치즈계란라면(3,300원) 차창라면(3,500원) 치즈차창라면(3,800원) 공깃밥(500원) 3,000원~

오후 9:07

챗봇에게 메시지 보내기

- 25 -

4) 성과평가

Feat
서울과학기술대Feat입니다.

전송됨 오후 9:08

생활선 성과평가

Feat 오후 9:08

<2019년 1분기 (성원학사) 성과평가 결과>

	실점수	가중치
유용성	10.00	0.4
안전성및내구성	9.1	0.4
서비스	9.8	0.2
평가		
위원	평가점수	평가등급
8명	9.6	A

<2019년 2분기 (성원학사) 성과평가 결과>

	실점수	가중치
유용성	10.00	0.4
안전성및내구성	9.6	0.4
서비스	9.4	0.2
평가		
위원	평가점수	평가등급
5명	9.6	A

<2019년 3분기 (성원학사) 성과평가 결과>

	실점수	가중치
유용성	9.7	0.4
안전성및내구성	8.8	0.4

Feat
서울과학기술대Feat입니다.

전송됨 오후 9:08

생활선 성과평가

Feat 오후 9:08

<2019년 1분기 (성원학사) 성과평가 결과>

	실점수	가중치
유용성	10.00	0.4
안전성및내구성	9.1	0.4
서비스	9.8	0.2
평가		
위원	평가점수	평가등급
8명	9.6	A

<2019년 2분기 (성원학사) 성과평가 결과>

	실점수	가중치
유용성	10.00	0.4
안전성및내구성	9.6	0.4
서비스	9.4	0.2
평가		
위원	평가점수	평가등급
5명	9.6	A

<2019년 3분기 (성원학사) 성과평가 결과>

	실점수	가중치
유용성	9.7	0.4
안전성및내구성	8.8	0.4

사진 5. 생활관 성과평가

5) 위치

세울과기대 세넷입니다.

자비주	8.8	0.2
평가		
위원	평가점수	평가등급
9명	9.1	A

전송됨 오후 9:08

장학관 위치가 어디야?

서울과학기술대학교 세넷

사자자 정보가 확인되지 않은 채널입니다. 금전 및 개인정보 요구를 받을 경우 신 >

고객 주석으로 [신고하기](#)

오후 9:07

장학관 성과평가

서울과학기술대학교 세넷

사자자 정보가 확인되지 않은 채널입니다. 금전 및 개인정보 요구를 받을 경우 신 >

고객 주석으로 [신고하기](#)

오후 9:08

장학관 위치가 어디야?

서울과학기술대학교 세넷

사자자 정보가 확인되지 않은 채널입니다. 금전 및 개인정보 요구를 받을 경우 신 >

고객 주석으로 [신고하기](#)

오후 9:08

장학관 위치가 어디야?

세넷 오후 9:08

3. 장학관

본 지도 보기

국민은행ATM
서울산업대 본

Google

한국지도 지도 데이터 ©2023 TMMap Mobility 약관

- 정보통신대학
- 에너지바이오대학

#

메시지를 입력하세요.

사진 6. 건물 위치

6) 연락처

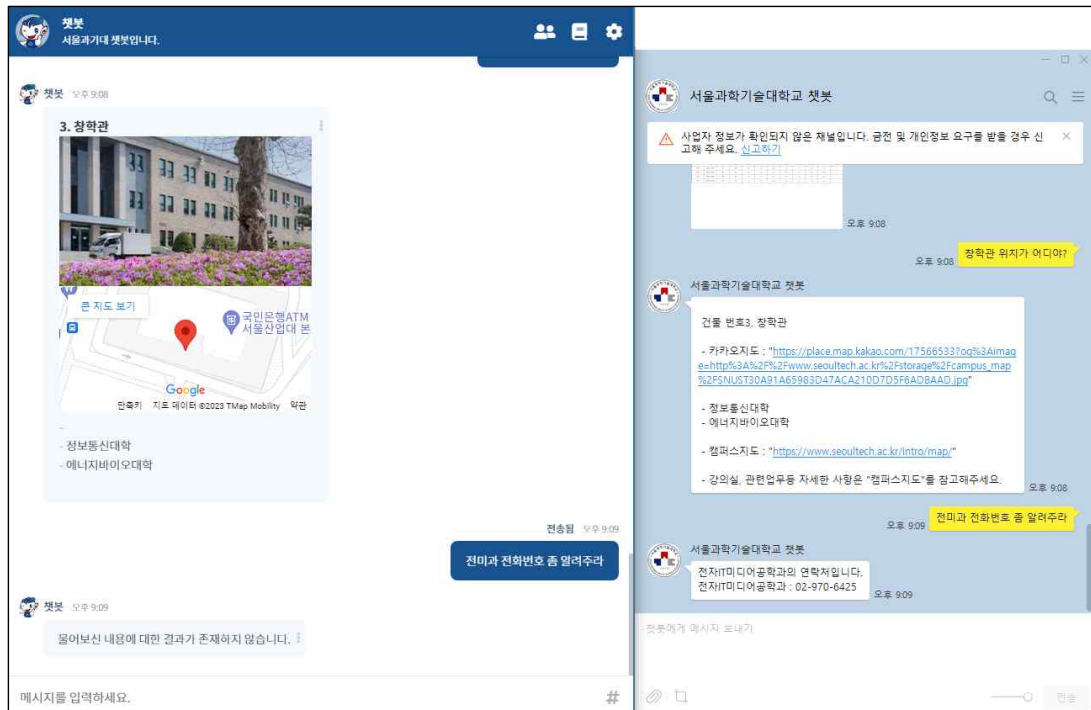


사진 7. 연락처

4.2. 데모 결과

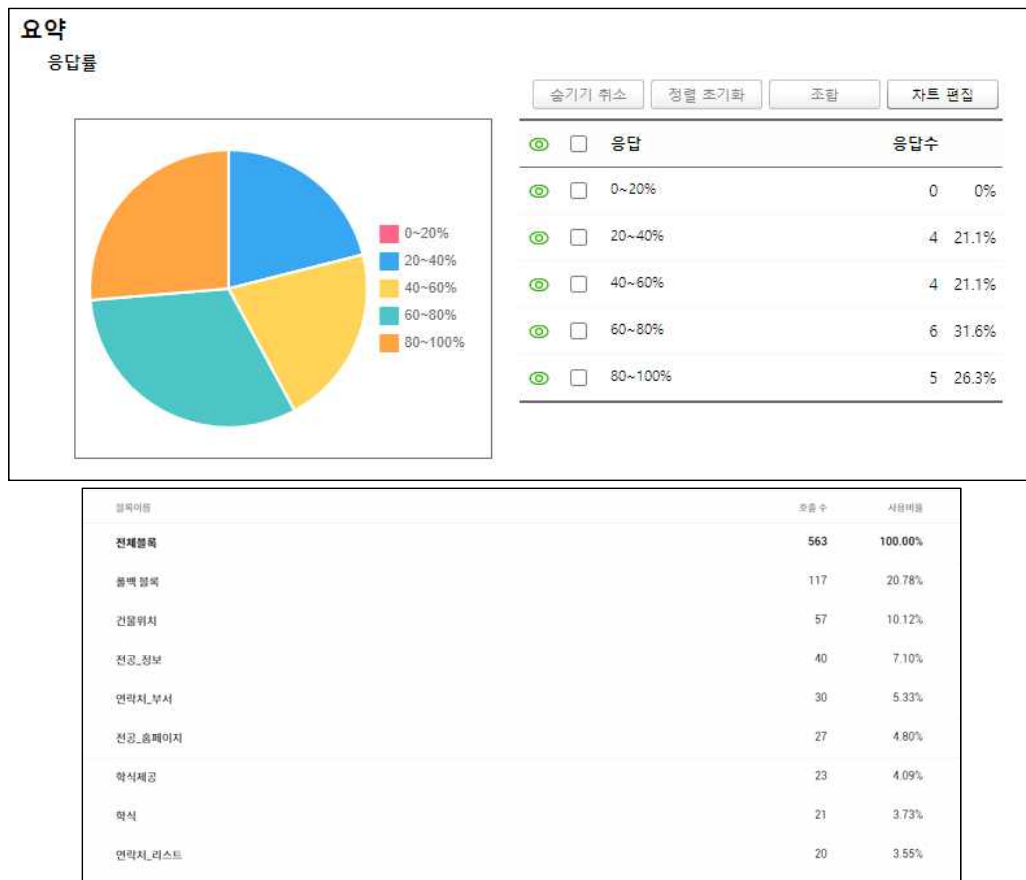


그림 25. 데모 결과

4.3 피드백

학교 교가	교가	2023.10.15	원료
태훈이형	만든이	2023.10.15	원료
서울 과기대 행정실 전화번호	연락처_리스트	2023.10.15	원료
화학과 연락처	연락처_리스트	2023.10.15	원료
전미과 연락처	연락처_리스트	2023.10.15	원료
퇴학	자퇴	2023.10.14	원료
자퇴는 어떻게 해?	자퇴	2023.10.14	원료

그림 26. 피드백

데모를 진행하거나 실제 사용자들이 입력한 발화가 의도에 맞는 블록이 존재하지만 fall-back 블록으로 진입한 경우 해당 발화를 직접 올바른 블록으로 연결시켜 응답률을 높이는 형태로 챗봇을 지속적으로 개선한다.

V. 결론 [결과도출]

5.1 챗미와의 비교 분석

기존에 목표로 하였던 챗미보다 높은 접근성을 제공하는 챗봇을 만드는 점 있어서는 상당히 만족스러운 결과를 얻을 수 있었다. 카카오 플랫폼 기반의 챗봇이다보니 홍보 없이도 미미하지만 사용자의 증가를 확인할 수 있었고 실제 학생들이 사용하는 줄임말이나 신조어들을 엔티티에 추가하여 챗미에서는 나오지 않은 응답들이 본 작품에서는 출력되는 것을 확인할 수 있었다.

또한 업데이트 주기에도 큰 차이가 있었다. 챗미에서 현재 제공하는 정보는 챗미 제작 후 업데이트 되지 않고 있다는 것을 확인하였다. 대표적으로 생활관 평가나 인원 현황들의 정보를 챗미는 2019년의 자료를 적은 이미지를 그대로 내보냈지만 본 작품은 실시간으로 정보를 크롤링해 이미지를 새로 만들어 최신의 정보를 사용자에게 제공할 수 있도록 하였다.

5.2 한계점과 극복 방법

기존에 목표로 했던 chat-gpt와 같이 문맥을 파악하여 사용자와 마치 사람처럼 대화 하는듯한 챗봇은 구현할 수 없었으며 응답 형식이 스킴 데이터를 사용하였음에도 불구하고 상당히 제한적이였다는 점은 상당히 아쉬운 부분이다.

데모 테스트 후 설문조사에서 챗봇의 답변은 되었지만 만족스러운 답변이 아니었다는 내용이 많았던 것으로 보아 어느 정도 타협했던 대답의 질을 높일 필요도 있어 보였다.

사용도중 포기했던 dialogflow의 문맥처리 기능을 사용하여 기존의 버튼형태의 추천식이 아닌 대화 형식으로 간다면 해결할 수 있지만 그만큼 하나로 합쳐버린 상황을 나눠서 하나하나 블록을 만들어야 하기에 쉽지는 않을 것으로 보인다.

VI. 참고 문헌

- [1] <https://mustzee.tistory.com/entry/Urllib-%EB%84%A4%EC%9D%B4%EB%B2%84-%EB%82%A0%EC%94%A8-%ED%81%AC%EB%A1%A4%EB%A7%81-2022-%EA%B0%9C%EC%A0%95%ED%8C%90-%EC%98%88%EC%A0%9C>
- [2] <https://vmpo.tistory.com/94>
- [3] <https://i.kakao.com/docs/skill-response-format#intent>
- [4] <https://m.blog.naver.com/nieah914/221609709142>

VIII. 감사의 글

본 작품의 연구는 서울과학기술대학교 전자IT미디어공학과에서 이루어졌으며 설계부터 완성에 이르기까지 많은 분들의 도움을 받았습니다. 먼저 주제 선정부터 이론의 정립 등 작품의 완성이 이루어지기까지 학문적 지도는 물론 모든 일을 깊은 관심으로 지도해 주신 신일훈 교수님께 감사의 말씀을 드립니다. 또한 1, 2차 데모 테스트를 진행하여 주고 자신의 과제처럼 설문조사를 참여해주신 서울과학기술대 학우분들과 그밖의 친구들에게 감사 인사를 드립니다.

졸업 작품 경험을 통해 앞으로 있을 더욱 힘든 일도 해쳐나갈 자신감을 얻게 되었습니다. 더욱 나아진 모습으로 졸업하여 학교를 빛낼 수 있는 인재가 되도록 노력하겠습니다.

4년이상의 시간동안 스쳐간 학교의 모든 사람들에게 감사의 말씀을 전합니다.

2023년 11월 10일

한 태 섭

이 석 규

엄 태 훈