목차

1. 작품 개요

2. 작품 설명

2-1 주요 동작 및 특징

2-2 전체 시스템 구성

2-3 개발 환경

3. 단계별 제작 과정

3-1 각 단계별 구현 방식

3-2 개발 단계 시행착오

3-3 향후 개선방향

4. 사용한 제품 리스트

5. 기타 (개발 코드)

6. 참고 문헌

1. 작품 개요

코로나 19로 인한 대학교 캠퍼스의 비대면화로 인해, 학생들간 소통에 많은 어려움이 발생하였으며 자연스럽게 온라인 대학 커뮤니티를 통해 정보를 얻는 경우가 많아졌다. 본래 선후배나 동기들간 공유되던 정보를 본인 스스로가 온라인상에서 찾아야 하는 상황이다. 하지만, 일부 학생들은 검색을 하기보다는 질문을 올리고 답변을 받는 방식을 선호하기도 한다. 일례로 유명한 대학 온라인 커뮤니티 어플 '에브리타임'에서는 <그림 1.1.>과 같이 다양한 질문들이 올라온다. 때로는 중복된 질문들이 올라와 불필요한 게시글로 도배가 되기도 한다. 예를 들어 <그림1.2.>에서 '프린터'라는 키워드를 검색창에 입력을 했을 때, 아주대 재학생들이 프린터 사용법에 대해 물어보고 있다. 모두 다른 질문 같지만 결국 프린터 실행 프로그램을 설치해야 하는 동일한 문제상황이다. 매일 등록되는 질문 글의 수는 많지만 그 중 대부분이 유사한 질문으로, 반복성 답변이 달리고 있기에 본 팀은 답변의 수가 무수하지 아니하고 한정되어 있다면 '챗봇'을 통해 문제를 해결할 수 있을 것이라 판단하였다.

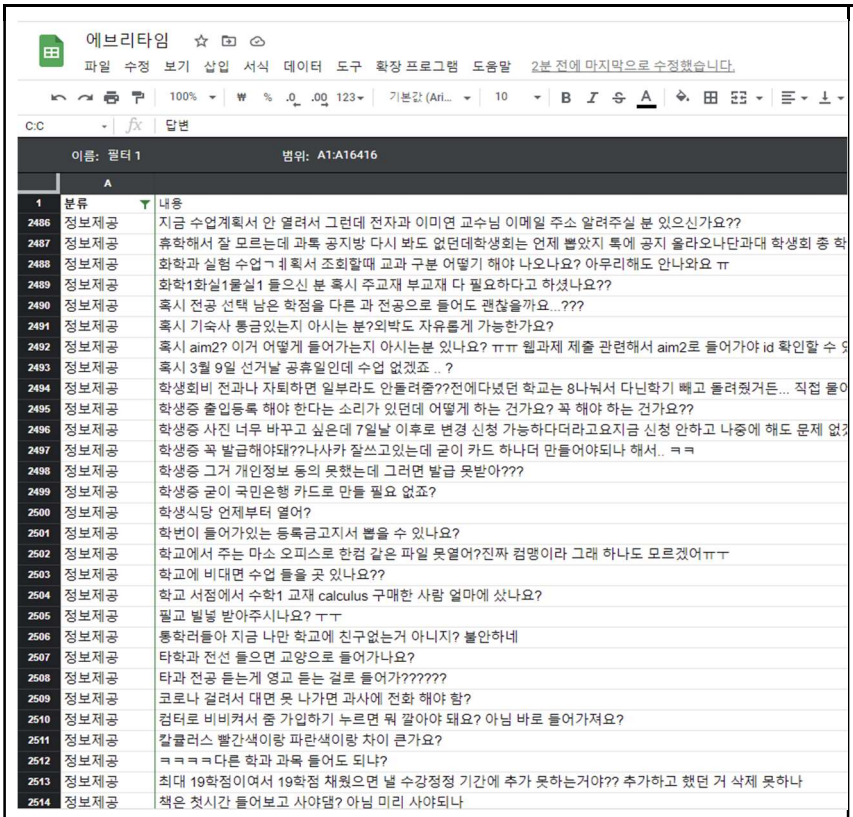


그림 1.1.
에브리타임에 등록된 질문 내역(웹 스크래핑 결과)

그림 1.2. 에브리타임
'프린트' 검색 결과 화면

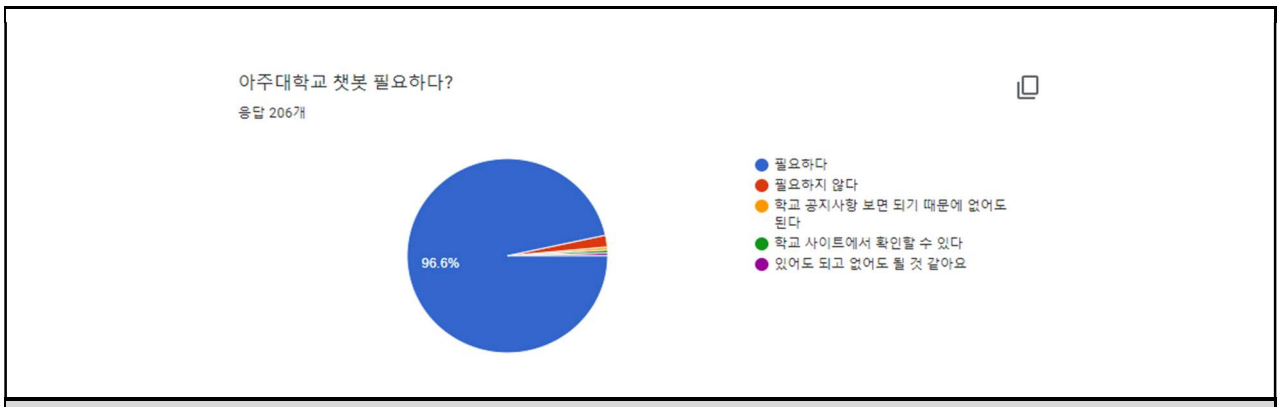


그림 1.3. 신입생 대상 챗봇 서비스 수요조사

따라서 <그림 1.3>과 같이 교내 신입생들이 사용하는 게시판을 통해 본 챗봇이 서비스될 경우 사용 의사에 대한 설문을 206명을 대상으로 설문조사를 실시하였고, 96.6%의 학생들이 아주대학교 챗봇이 필요하다고 언급했다. 1.9%의 학생들이 불필요성을 느꼈고 1.5%의 학생들이 기타사항에 답변을 해주었다. 공통된 의견으로 학교 공지사항이 정보통으로서 제 역할을 하고 있기 때문에 큰 필요성을 느끼지 못한다는 것이었다. 하지만 압도적으로 찬성 측에서 긍정적 평가를 받았기 때문에 챗봇을 제작하기로 하였다.

2. 작품 설명

2-1. 주요 동작 및 특징

Python을 기반으로 PC에서 작동하는 프로그램이지만, 사용자의 편의를 위해 이를 모바일 어플리케이션으로 연동하였다.

사용자가 모바일 어플리케이션을 실행하였을 때, 일반적인 채팅 프로그램의 모습을 하고 있으며 교내 정보에 대한 질문을 하면 이에 챗봇 알고리즘이 적절한 답변을 한다.

문장을 입력했을 때 적절한 답변의 출력까지, 다음과 같은 알고리즘을 가진다.

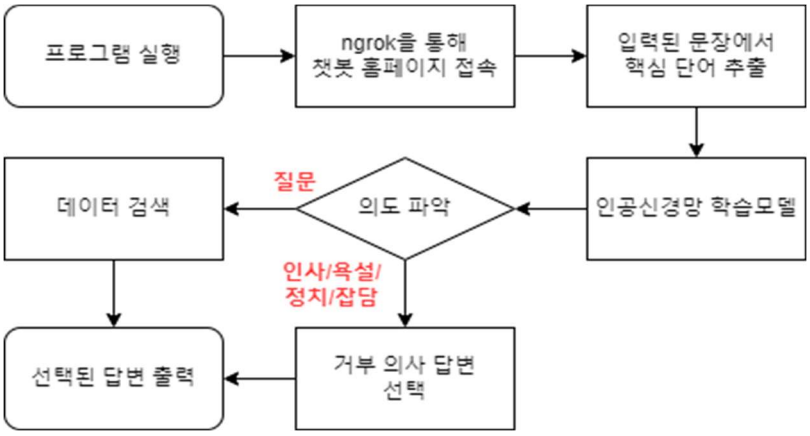


그림 2.1 챗봇 알고리즘(간략화)

프로그램은 어플의 형태를 하고 있으나, 시스템은 개발자의 컴퓨터에서 이루어지며 사용자는 ngrok의 도움을 받아 원격으로 접속해 사용하는 구조이다.

사용자의 입력을 받으면, 문장을 형태소 단위로 분석하고 핵심단어를 추출한다. 핵심단어를 인공신경망 알고리즘에 집어넣음으로써, 해당 문장의 의도를 파악하고 이에 대응한다. 부적절한 입력의 경우 거부의사를, 적절한 질문(정보제공)의 경우 데이터 세트로부터 적절한 답변을 출력한다. 해당 데이터셋은 주기적으로 정보가 업데이트가 된다. 유사도 검사 알고리즘을 통해 가장 적절한 답변이 있다면 이를 출력하도록

설계되었다.

2-2. 전체 시스템 구성

챗봇 알고리즘의 구조는 크게 다음과 같이 나눌 수 있다.

1. 입력 문장의 처리
2. 입력 문장의 의도 분류
3. 의도에 따른 답변 검색
4. 적절한 답변 출력

또한 각 기능을 수행하는데 있어 사용된 주된 개념은 다음과 같다.

1) Convolutional Neural Network(CNN 인공신경망)
2) Natural Language Processing(자연어 처리)
3) Tokenizer(단어의 숫자화를 위한 토크나이저)
4) Cosine Similarity(벡터를 사용한 코사인 유사도)
5) VPN Distribution(로컬 서버 접속 권한 부여)

상기 개념들에 대한 자세한 설명은 아래 3. 단계별 제작 과정에서 설명하도록 한다.

본 서비스는 사용자에게 친숙한 모바일 메신저 형태의 교내 QnA챗봇이다. 특히, 사용자가 입력한 문장의 의도를 파악하기 위해 딥러닝 기술을 사용한다. 사용자가 입력하는 문장의 의도를 대분류로 인사/욕설/정치/잡담/정보 제공 중에서 판단하고, '정보 제공'으로 판단되었을 경우 교내 활동/수업/기숙사/수강신청/학사 문의로 세부 분류를 확인하였다. 이러한 2중 딥러닝을 위해 기본적으로 BigData가 필요하다.
최근 빅데이터와 딥러닝에 대한 관심이 증가함에 따라, 대화 및 말뭉치 데이터가 OpenSource를 통해 무료로 쉽게 구할 수 있다. 우리는 이 중에서 한국어로 된 인사말과 욕설 데이터를 수집하였고, 이외 정치적 문구, 잡담, 교내 정보질문을 판단하기 위해 웹스크래핑으로 데이터를 얻었다. 이렇게 모아진 Data를 원초적 상태의 데이터(Raw Data)로 지칭한다. 해당 데이터를 자연어처리를 통해 핵심단어를 추출하여 정제하였고, 결과 데이터를 딥러닝의 학습에 사용한다. 그 중에서도 질문과 답변형태로 된 데이터는 답변을 검색하기위한 데이터로 따로 구성된다. 딥러닝을 통해 문장의 의도를 파악할 수 있게 되면, 해당 학습의 결과물을 외부 파일(model)로 저장할 수 있다. 이렇게 데이터를 처리하면 정제된 데이터(Processed Data)를 얻을 수 있다.

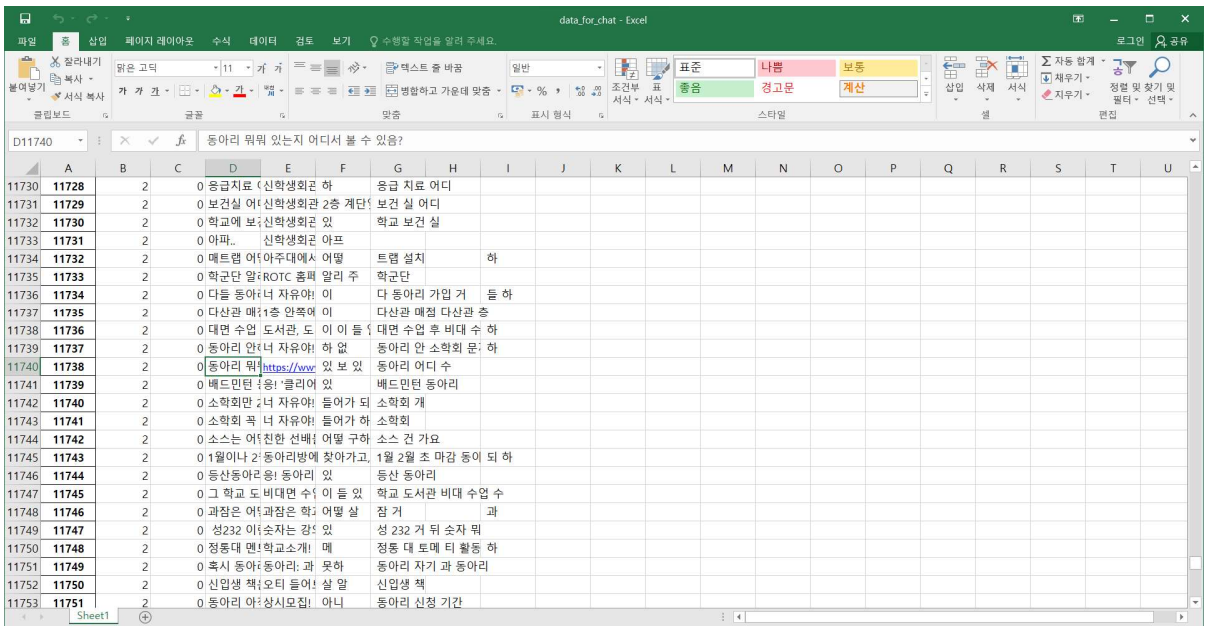
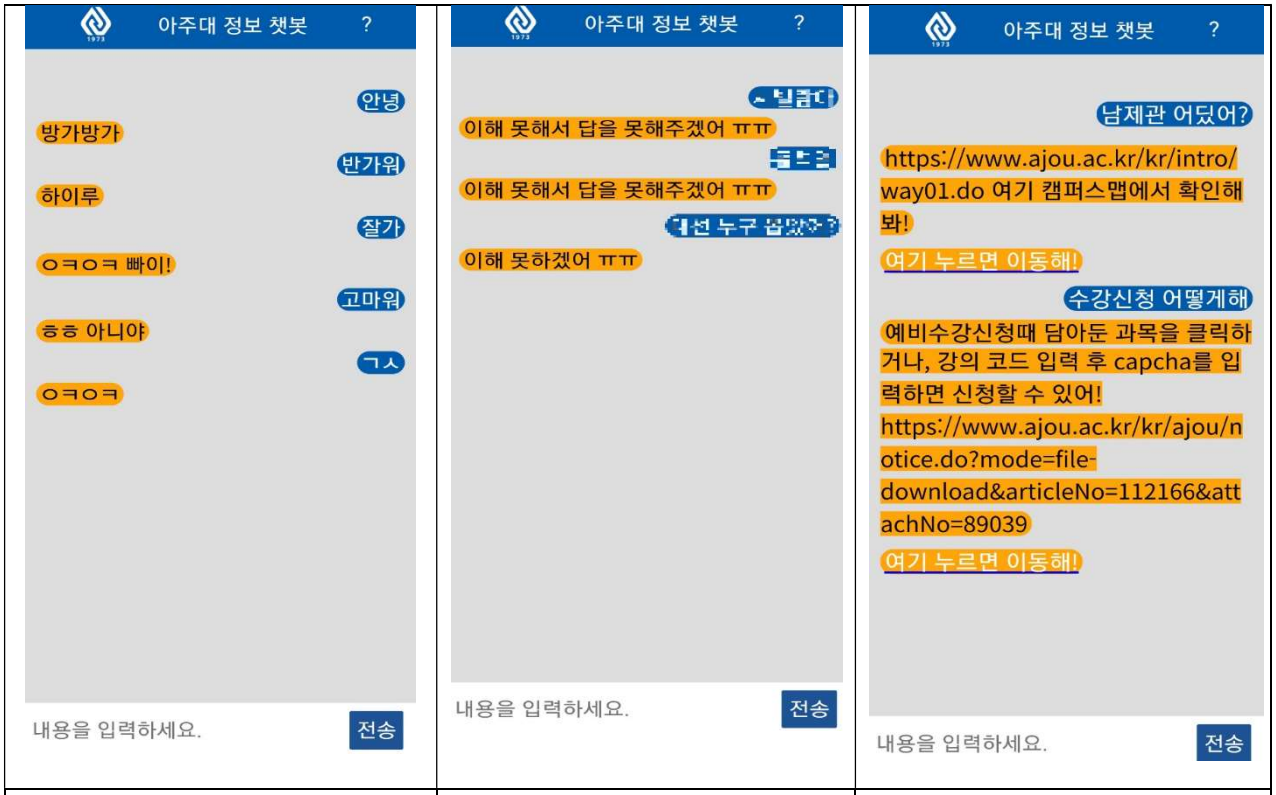


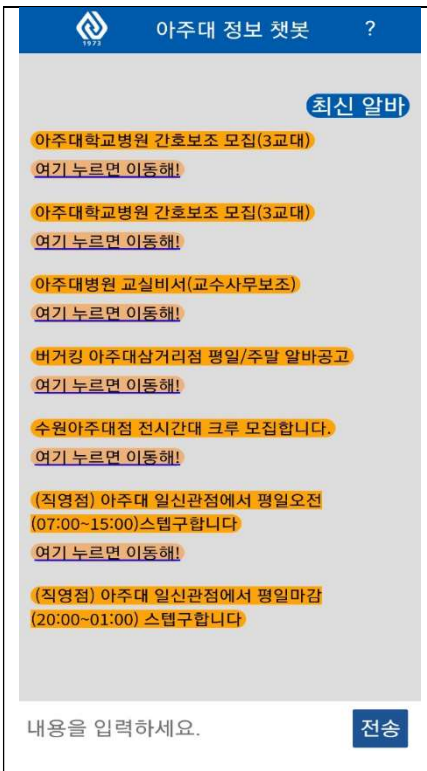
그림 2.2 웹 스크래핑 후 핵심단어 추출을 끝낸 <정제된 데이터>의 일부분

다음은 본격적으로 입력을 받고 처리하는 Chatbot Algorithm의 과정을 설명한다. 사용자가 입력한 문장은 서버를 통해 알고리즘에 입력되며(위 '서버'에 대한 개념은 다음 문단에서 설명한다) 입력은 위에서 저장한 학습 결과물(model)을 통해 판단된다. 결과에 따라, '적절한 질문'이 아닌 경우 상응하는 답변을 서버로 전송해준다. '답변이 요구되는 적절한 질문'의 경우 해당 문장을 '실시간으로 제공되어야 하는지'를 판단한다. 실시간으로 업데이트되는 정보가 중요한 경우, 해당 정보를 제공할 수 있는 웹페이지에서 해당 내용을 추출하여 답변해준다. 여기에는 코로나 확진자 수와 날씨, 교내 식당메뉴 등이 해당한다. 기타 교내 정보와 같이, 한정적이고 최신화 필요성이 비교적 떨어지는 경우 기존 질문과 답변이 있는 데이터에서 해당 질문을 검색한다. 우선 사용자의 입력문장에서 핵심단어를 추출하고, 이를 위에서 만든 정제된 정보(Processed Data)의 내용들과 코사인 유사도를 통해 가장 유사한 문장을 검색한다. 가장 유사한 문장이 유사도가 높다면, 해당 답변을 출력해준다. 유사도가 낮아 원하는 정보가 아닐 것이라 판단되는 경우, 사과의 메시지와 재질문을 유도한다. 이렇게 사용자의 입력에 대해 어떤 답변을 제공할지 결정되면, 이는 서버로 전송되어 사용자에게 전달된다.
서버(Server)는 위 알고리즘이 개발자의 컴퓨터상에서 작동한다면, Chatbot Protocol은 서버를 사용자의 휴대폰과 연결해주는 징검다리 역할을 해준다. 해당 서버를 통해, 개발자의 컴퓨터에서 검은색 명령창(Cmd)을 통해 데이터가 처리되더라도 사용자는 친숙한 모바일 채팅창에서 정보를 입력하고, 서버로 건너간 정보가 답변을 결정하면 다시 서버로부터 사용자의 휴대폰으로 전달되어 대화하듯이 답변이 출력된다.

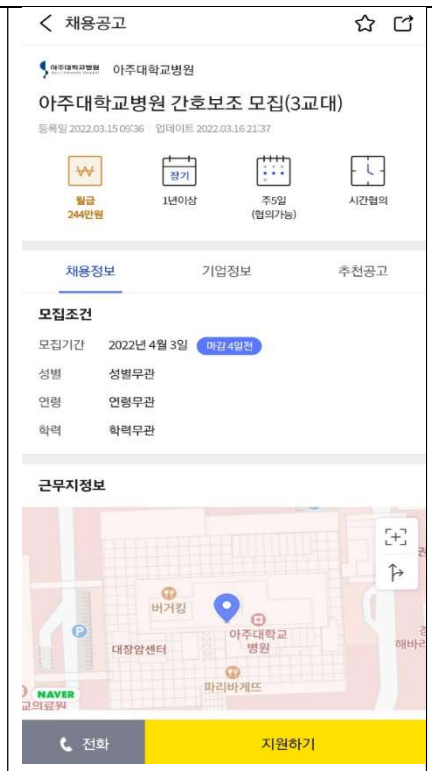
아래는 챗봇이 딥러닝을 통해 입력 문장의 의도를 파악하고 적절한 답변을 출력하는 예시를 나타낸다.



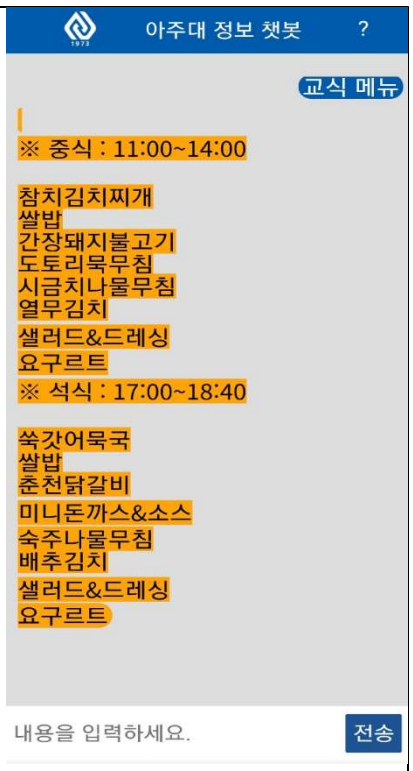
인사말에 대한 답변	욕설, 정치 문장에 대한 답변	교내 정보에 대한 답변



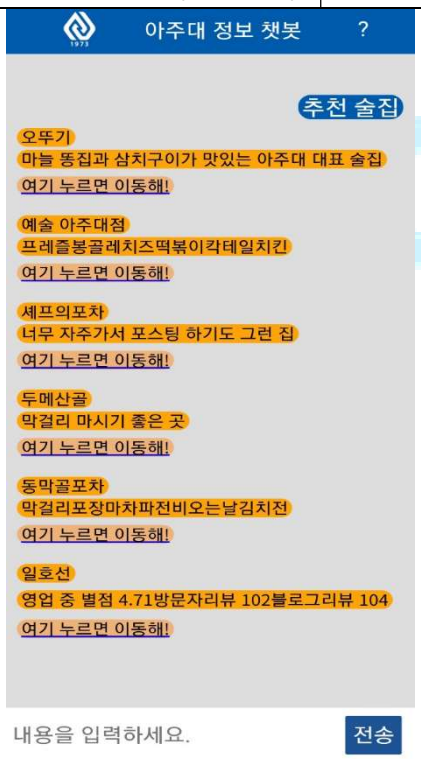
최신 아르바이트 정보(웹크롤링)



<아르바이트> 답변 링크 내용



교내 식당 메뉴 답변(웹크롤링)



주변 맛집 추천 답변(웹크롤링)

<주변 맛집 추천> 링크 내용

그림 2.3 챗봇 각 기능별 구동 화면

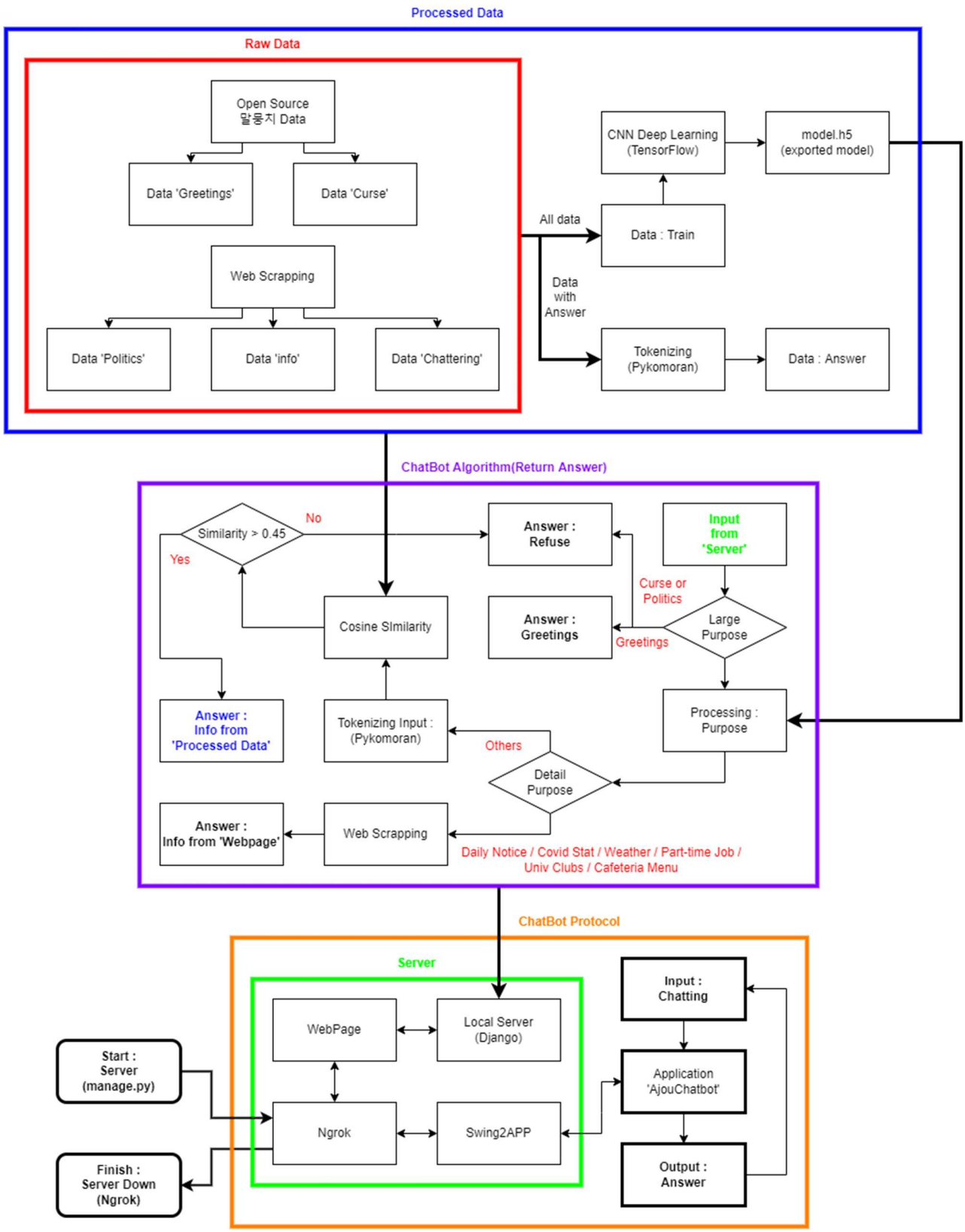


그림2.4. 단계별 제작 과정 플로우차트

본 서비스의 전체 구성 및 알고리즘을 위 사진과 같이 플로우차트로 시각화 하였다.

2-3. 개발 환경(개발 언어, Tool, 사용 시스템 등)

python™ django	python™
Platform : Django	Language : Python
jupyter	HTML CSS JS
Platform : Jupyter	Language : Html, css, javascript
TensorFlow	ngrok
Tool : TensorFlow	Tool : ngrok
VS Code	SWING2APP
Platforml : Visual Studio Code	Platform : Swing2APP

3. 단계별 제작 과정

3-1 각 단계별 구현 방식

세부적인 내용을 설명하는데 있어 챗봇의 각 기능을 단계로 나누기 위해, 위 2-2. 전체 시스템 구성에서 언급된 챗봇 알고리즘의 구조를 재언급한다.

1. 입력 문장의 처리, 2. 입력 문장의 의도 분류, 3. 의도에 따른 답변 검색, 4. 적절한 답변 출력

프로젝트 시작에 앞서 우선적으로 데이터를 수집하였다. 데이터는 사용자의 질문에 적절히 대답하기 위한 유의미한 학교 관련 데이터와, 필터링하기 위한 욕설. 인사말, 잡담, 정치, 사생활 적인 부적절한 데이터를 수집하였다.

우선 욕설 및 인사말은 현재 상용화된 많은 인공지능 프로그램에서 사용되고 있는, '우리말 말뭉치'를 비롯한 오픈소스 말뭉치 데이터를 사용하였고, 정치적/사회적 논란이 있는 언어의 경우 인터넷 뉴스기사를 웹페이지 크롤링 하여 수집하였다

그 후 학교 데이터를 모으기 위해 공지사항 제목, QnA 게시판, 대학 커뮤니티 '에브리타임'을 웹크롤링하였는데, qna게시판과 에브리타임의 경우 홈페이지 로그인이 요구되었기에 Jupyter notebook에서 Selenium을 통해 순차적으로 웹페이지에 접근하며 데이터를 수집하였고, 공지사항은 visual studio code에서 BeautifulSoup을 이용하여 크롤링하였다.

공지사항 제목과 QnA게시판을 통해 수집한 데이터들은 사용자로부터 얻은 입력이 학교 관련 질문인지 아닌지 판단하는데 사용되었으며, 에브리타임의 질문-답변 데이터는 2020-2022년 '새내기 게시판'에서 가져와 신입생들이 자주 하는 질문들을 파악하고 질문에 달린 답변들을 활용하여 답변 데이터 풀을 만들어 엑셀에 저장하였다.

공지사항과 QnA게시판에서 수집한 데이터와 달리 에브리타임 데이터는 정제되지 않은 데이터가 대부분이었기에 전처리 하는 작업이 필요했다. 따라서 수집한 데이터들을 다음과 같은 단계를 거쳐 정제하였다.

1. 불필요한 내용 삭제. 에브리타임에서 얻은 학생들이 직접 작성한 질문들을 불필요한 문장을 삭제한 뒤, 이용하기 쉽게 문장을 재구성하였다.

2. 기준에 따라 분류. 정제되지 않은 초기 데이터를 크게 잡담과 정보제공으로 분류한 뒤, 정보제공 데이터는 '교육과정/과목/수업', '기숙사', '학사문의', '교내활동/생활', '수강신청'이라는 5가지 기준으로 분류하였다. 이는 모델 훈련을 용이하게 하기 위함이다.

3. 문장 구조 통일. 우리 챗봇은 뒤에서 언급될 '코사인 유사도'를 통해 사용자가 입력한 질문과 엑셀의 question 데이터들을 비교하여 가장 유사한 질문을 찾고, 해당 question 데이터에 대응되는 answer data를 출력하는 방식이다. 이때 코사인 유사도는 문장을 토큰화한 뒤 각 단어를 벡터화 하여 유사성을 비교하는 것인데, 만약 "학사서비스 어디서 볼 수 있어?"와 "공지사항 어디서 봐?"라는 두 question 데이터가 있고, 사용자의 질문이 "공지사항 어디서 볼 수 있어?"라면 해당 질문과 비슷한 question 데이터로 전자를 반환한다. 이러한 유사도에서 발생할 수 있는 오류를 해소하기 위해 비슷한 유형인 질문들의 문장 구조를 통일하였다.

위에서 수집한 데이터 외에 에브리타임, 뉴스 등에 없는 문장은 수작업으로 넣어주어 데이터를 저장하였다. 또한 아주대 공지사항, 음식점 추천, 동아리 추천, 공모전, 식당 메뉴, 코로나, 날씨 등의 데이터를 Selenium 및 BeautifulSoup 라이브러리를 통하여 데이터를 실시간으로 가져오게 설정하였다.

이와 같이 수집된 약 12000개의 데이터는 크게 두 가지 목적으로 저장된다.
1. 사용자의 문장 입력 의도 분류
2. 사용자의 질문에 대한 답변 출력
사용자의 입력 문장의 의도가 부적절한 경우, 답변하는 데이터의 양은 소수로 한정되어 있다(예: 인사의 경우엔 인삿말, 욕설/정치의 경우 부정). 반면 입력 의도가 적절한(질문)경우, '질문에 대한 답변'이 필요하므로 각 질문 데이터에 대해 답변 데이터를 연결시켰다.

구조 2번의 <입력 문장의 의도 분류>를 위해, 위에서 수집된 '의도 분류 데이터'를 기반으로 분류 모델을 생성한다. 여기서 딥러닝 모델을 통해, 입력되는 수많은 문장을 각각의 경우의 수가 아닌 소수의 데이터만으로 다양한 경우의 입력문장들을 분석해 자체적으로 의도를 판단할 수 있다. 인공신경망 모델로는 CNN, RNN, LSTM등 다양한 기법이 있으나, 채팅이 단답형 문장이며 빠른 응답이 최우선시 되기에 이 중 CNN(Convolutional Neural Network)기법을 사용해 설계하였다. 해당 CNN 인공신경망 알고리즘을 설명하기위해, 다음 그림과 함께 단계를 나누어 설명하도록 하겠다.

CNN 모델 블록도

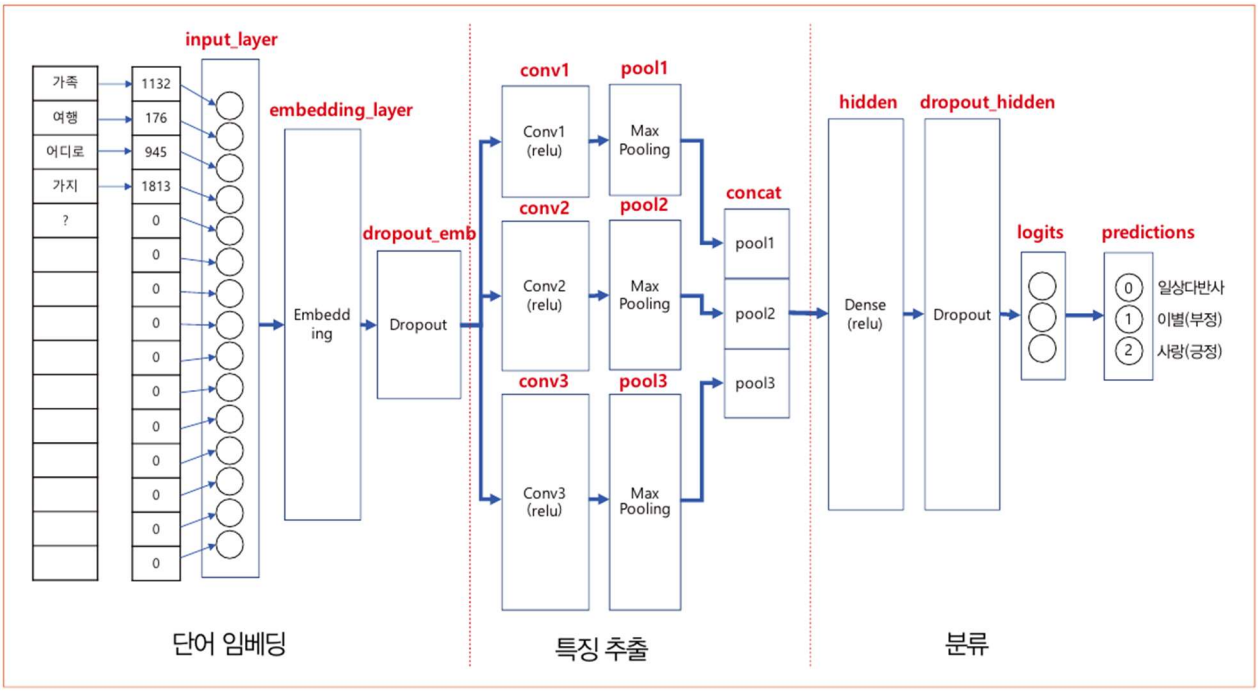


그림 2.2 CNN모델 인공신경망 알고리즘 다이어그램

인공신경망의 처리는 크게 '임베딩(Embedding)', '특징 추출', '분류'로 나뉜다. 임베딩의 경우, 입력된 값이 학습가능한 모양으로 만드는 과정이다. 학습을 위해 입력데이터는 모두 숫자의 형태가 되어야 하며 길이가 동일해야 한다. 임베딩은 크게 4단계로 나뉜다.
1. 문장을 단어로 끊고, 핵심단어를 추출하는 과정이 필요하다. 문장을 보다 정확히 기계학습하기 위해, 문장 데이터들을 형태소별로 분석하였다. 이는 Python의 NLP(자연어 처리)를 위해 개발된 패키지(예 : Komoran, Okt)들을 사용해 해결할 수 있다. 입력 문장에서 명사와 동사등 핵심적인 부분만을 추출하게끔 알고리즘을 작성하였다.

```
from konlpy.tag import Komoran

komoran = Komoran(userdic='./user_dic.tsv')
text = "국제학사 떨어져서 남제관 배정받았어"
pos = komoran.pos(text)
print(pos)
```

executed in 4.26s, finished 13:38:29 2022-03-23

```
[('국제학사', 'NNG'), ('떨어지', 'VV'), ('어서', 'EC'), ('남제관', 'NNG'), ('배정', 'NNG'), ('받', 'VV'), ('았', 'EP'), ('어', 'EC')]
```

그림 2.3 Komoran으로 문장을 형태소로 분리한 모습

2. 각 단어를 중복을 제외하여 저장한다. 이를 사전화(Dictionary)라 한다.
3. 분리된 단어들을 숫자형으로 변환한다. 이를 '토크나이저(Tokenizer)를 통한 시퀀스화(Sequence)'라고 한다. 이는 파이썬에서 제공되는 함수로 구현이 가능하다.
4. 각 문장의 길이가 제각각이므로, 문장의 평균길이보다 조금 큰 값으로 끊는 지점을 정하고 짧은 문장은 나머지 칸에 '0'으로 채워넣어 길이를 맞추었다. 이를 '패딩(Padding)'이라고 한다.

임베딩의 다음은 특징 추출이다. 숫자 처리된 데이터 값을, '합성곱 연산'을 통해 학습한다. 해당 과정에서 데이터들을 한번에 몇개씩 볼 것인지를 설정하여(해당 개수를 '필터'라고 한다) 문장에서 이어지는 단어들의 관계와 문장의 성격을 이해시킬 수 있다. 우리는 크기 3,4,5의 필터를 128개씩, 3회 반복으로 설정하였다. 매 시행마다 나타난 실제 값과의 오차를 확인하여 학습된 정보를 수정하며 진행된다. 여기에는 인공신경망 학습모델로 가장 많이 채용되는 RELU함수를 사용하였으며, 위에서 말한 3가지 크기의 필터 각 128개로 학습을 진행하였다.

마지막은 분류이다. 3번의 반복과정에서, 목표 값의 분류 개수에 맞게끔 학습을 통해 얻어진 특징 데이터를 수집한다. 우리는 분류로 인사/욕설 및 정치/정보제공/잡담의 4가지를 설정하였다. 학습과정에서 모아진 데이터를 바탕으로 각 문장은 해당 4가지 분류에 대한 유사도를 각각 구하게 되고, 유사도 값을 파이썬의 softmax를 비롯한 함수들을 사용해 확률로 변환한다.

이렇게 학습된 결과는 '모델'이라는 이름의 파일로 저장이 되고, 사람으로 치면 '학습한 뇌'를 휴대하는 것과 비슷하게 볼 수 있다. 이제 새로운 문장이 입력되면, 위에서 준비한 뇌, 즉 모델을 불러와 해당 문장의 의도를 파악할 수 있게 된다. 우리는 약 12400개의 데이터 중 70%를 학습에 사용하였고, 나머지 30%를 검사해본 결과 정확도는 99%이상의 매우 성공적인 결과를 얻었다.
여기서 모델, 즉 '뇌'가 학습을 마쳐도 파일로 저장해버린 이상, 더이상 새로운 정보를 받아들일 수 없게 된다. 정보를 추가하려면 기존의 모델을 삭제하고 새롭게 학습과 파일생성의 과정을 거쳐야 한다. 하지만, 본 프로젝트에서는 최신 정보에 민감한 FAQ 챗봇을 제작함에도 해당 방식을 채택하였다. 답변과 달리 질문 의도분류의 경우, 문장의 내용을 섬세하게 인식하지 않고 대화의 주제만을 파악하면 되기에 학습 데이터가 최신화 될 필요성이 낮다. 따라서, 개발과정에서 학습시킨 인공신경망을 외부 파일로 저장하고 이를 사용한다. 이렇게 저장된 파일을 사용하면 프로그램을 실행 시킬 때 마다 학습을 하지 않고 파일을 불러오기만 하면 되므로, 작동시간을 대폭 단축할 수 있다.

구조 3번의 <의도에 따른 답변 검색>은 크게 두가지 형태의 답변을 출력한다. 웹사이트의 정보를 전달하는 방식과, 위의 '질문에 대한 답변 데이터'를 사용하는 방식이다.
실시간으로 정보가 변하거나 해당 자료가 잘 정리된 웹페이지가 있는 경우 이를 파이썬의 Selenium 기능을 통해 웹스크래핑 하여 정보를 출력한다. 이러한 답변은 특정 단어가 포함된 명령어를 통해 인식한다.
이외에는 기존에 저장한 '질문에 대한 답변 데이터'를 사용한다. 해당 경우 입력 받은 문장을 기존 데이터 묶음과 비교가 필요하다. 답변을 검색하는 방법으로 BM25, TF-IDF등 다양한 방식이 사용되나, 우리의 개발목적인 '교내정보전달'에 있어 문서의 검색보다는 단답형 답변이 많기에, '코사인 유사도'가 가장 적합하

다고 판단하고 이를 사용한다. 코사인 유사도의 공식은 다음과 같다 :

$$cos(\Theta) = \frac{A \cdot B}{||A||\ ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

위 공식은 두 벡터 A,B의 유사도를 나타낸다. 두 벡터가 일치한다면, 사잇각은 0도가 될 것이다. 코사인 유사도는 두 벡터에 대해 사잇각을 계산함으로써, 두 벡터가 얼마나 일치하는지를 확인할 수 있다. 이러한 과정은 위 CNN인공신경망의 임베딩 과정과 비슷하게 진행된다. 겉보기에 다른 문장일 지라도, 의미가 비슷하다면 이를 인지할 수 있어야 한다. 이는 형태소 분석을 통한 핵심단어 추출을 통해, 두 문장의 유사도를 판단하기 용이하게 만들 수 있다. 이렇게 추출된 단어를 시퀀스화 시켜 숫자로 변환하면, 숫자의 조합들을 벡터로 간주할 수 있다.
이러한 성질을 사용하여, 문장이 입력되면 1.해당 문장에서 단어를 추출하고 2.이를 시퀀스화(벡터)하여 3. 코사인 유사도를 통해 저장된 답변데이터들과 비교하여 가장 유사한 문장을 검색할 수 있다. 해당과정을 보다 빠르게 처리하기 위해, 우리는 기존의 답변 데이터들을 이미 핵심 단어추출(형태소분석)을 마친 상태로 별도 저장하였고 이를 비교에 사용하였다.
코사인 유사도를 통해 검색된 최고 유사도가 0.45 이상인 경우 해당 질문은 '연관성이 있다'고 판단되어 답변을 출력하고, 그 이하인 경우 '연관성이 낮다'고 판단하여 사과의 메시지와 함께 재질문을 유도하게끔 설계하였다.

이제 챗봇의 마지막단계인, 구조4번 <적절한 답변 출력>이다. 본 서비스는 사용자가 모바일 환경에서 사용하기 위해 어플리케이션 형태로 제공되어야 한다. 모바일과 PC환경 모두 사용 가능한 통일된 시스템 구축을 위해, Python 환경의 프로그램을 웹페이지 형태로 변환이 필요하다.웹페이지 제작에 사용되는 Python, CSS, JAVA, Html은 각기 다른 언어 문법구조를 가진다. 지원하는 프로그램도 상이하나, Django 플랫폼은 위 언어를 모두 지원하여 동일 환경에서 각기 다른 코드의 작업이 가능하다는 점에서 해당 플랫폼을 활용하여 프로그램을 웹페이지화 하였다. 이렇게 하면 실제 실행되는 파일은 Python으로 알고리즘의 동작이 일어나고 사용자는 Html을 통해 친숙한 채팅방 화면으로 조작할 수 있다. 하지만 파이썬과 Html은 다른 언어를 사용하기에, 이 둘을 호환하기 위해 추가적인 작업이 필요하다. 이는 json라이브러리에 있는 jsonresponse를 사용하여 해결 가능하다. json라이브러리는 문자열 데이터의 처리, 가공, 전달에 있어서 다른 컴퓨터 언어들 끼리의 상호작용을 하는 역할을 한다. 그 중 jsonresponse는 python에서 작성된 데이터 뭉치를 html이 인식이 가능한 문자열 형태로 보내준다. 이를 통해 Python에서 주로 사용되는 '리스트'(연속된 데이터들의 배열) 를 비롯하여 처리된 답변 데이터를 html이 인식할 수 있게 변환 및 전송할 수 있다. 하지만 우리가 주로 주고받을 데이터인 답변의 경우 보통 한글 형식이므로 ascii 코드로 깨져 나오는 현상이 발생한다. 이는 json 내부의 'dump 설정 인자' 에서 ensure_ascii를 False로 변경하면 해결할 수 있다.
파이썬 알고리즘에서 선택된 답변 데이터의 형태는 주로 string(문자열)의 형태가 나온다. 이를 html환경으로 넘겨주기 위해 정보(답변)를 '객체화'하여 전달한다. 이는 택배를 보낼때 내용물의 손상을 방지하기 위해 박스로 포장하는 것과 비슷하다. 호환이 안되는 두 언어 간에, 해당 과정을 거쳐 데이터를 정상적으로 교환하게 된다. html에서 문자열을 입력 받으면, bottext를 통해 해당 문자열을 채팅의 말풍선처럼 디자인 하여 출력할 수 있다.

하지만 이렇게 생성된 웹페이지는 개발자의 컴퓨터에서만 접속이 가능하다.(Django의 경우는 local server에서는 문제가 안되지만, 외부 서버에서 접근하게 되면, 기본적으로 local server 내에서 방화벽에 침입하는 것으로 인식하여 차단을 한다). 이를 해결하기 위해 각종 방법을 모색한 결과, 최종적으로 ngrok을 사

용해 외부에서 로컬서버(개발자의 컴퓨터)로 접속하게끔 설정하였다. ngrok은 외부의 접속을 허용하기 위해, 윈도우의 방화벽을 우회하는 VPN기능을 제공한다. 이러한 기능을 통해 개발자가 생성한 웹페이지에 한해, 제한적으로 외부 접속을 허용하게끔 설정하였다.

알고리즘과 서버를 연동한 채팅창을 어플리케이션화 하는 과정의 가장 먼저 제작한 채팅창에 부합하는 제작플랫폼을 선정하는 과정에서 replit, appinventer2, Swing2APP 등의 여러가지 플랫폼을 살펴보았다. 우리가 제작한 채팅창은 Django서버 내에서 대화가 주고 받아지기 때문에, 어플을 제작한다면 다른 기능을 필요로 하는 것이 아니라 그저 채팅창을 보여주는 어플을 제작하면 된다. Swing2APP의 경우, 뷰 전용 어플리케이션 제작 툴이 잘 갖추어져 있다. 또한 그 과정 또한 매우 간단하며 Html, css, javascript 등 여러 언어와 호환 가능하다. 제작된 채팅창을 연결하기에 Swing2APP플랫폼이 가장 적합하다고 판단되어 선정하게 되었다. 어플리케이션화 하기 위해서는 Django를 사용하여 제작한 로컬서버를 외부 사용자가 접근 가능하도록 서버를 호스팅해주어야 한다. 우리는 ngrok툴을 사용하여 서버를 호스팅하였다. 호스팅 된 Url에는 외부 사용자 모두가 접근 및 이용 가능하며, Swing2APP을 통하여 해당 Url에 접근하는 어플리케이션을 제작할 수 있었다. 해당 팀원들처럼 모바일 어플리케이션 개발 경험이 부족한 경우, Swing2APP을 사용하여 코드 작성부터 구글 플레이스토어 등록까지의 절차를 간편화 하여 개발 진입장벽을 크게 낮출 수 있었다. ngrok을 통하여 배포된 고정 URL주소를 제작한 어플리케이션에 연동해주어 다운로드한 사용자들은 모두 제작한 채팅창에 접속 가능하다.

이렇게 개발된 프로그램은, 다양한 모바일 기종에 따라 테스트를 진행함으로써 발생하는 문제점을 확인해가며 수정을 마친 상태로 현재 구글플레이에서 배포 및 다운이 가능하다.

3.2 개발 단계 시행착오

해당 프로젝트를 진행하면서, 각 단계별로 시행착오를 겪으며 계획을 수정하는 방향으로 프로젝트를 진행하였다. 팀원들 모두 c언어/파이썬에 약간의 지식이 있었을 뿐, 역량이 뛰어난 것이 아니었다. 특히 Html/java/css와 같은 모바일 관련 코딩과, 서버 구축을 위한 컴퓨터 네트워크 관련 지식이 전무했기에 밑바닥에서부터 배우며 해당 과제를 수행하였다. 많은 발전이 있었으나 여전히 부족한 실력이기에, 원하는 만큼의 결과를 얻지 못한 것으로 보여 아쉬움이 크게 남는다. 부족한 부분은 이후 학습을 통해 점진적으로 구현할 예정이다.
아래는 개발과정에서 나타난 각종 문제점으로 계획이 수정된 점, 이후 해결방안에 대해 서술하도록 한다.

데이터 수집 단계에서는 최신 공지사항, 동아리 추천, 음식점 추천, 알바 추천, 장학금 추천, 공모전(문학, UCC, 대학생, 과학) 추천 등의 정보를 Selenium과 BeautifulSoup 라이브러리를 통하여 웹스크래핑을 진행하고 그 내용을 챗봇이 내보내는 형태로 진행하였다. BeautifulSoup으로 진행하였을 때에는 Chrome을 따로 키지 않고 url만 따로 설정하여 데이터를 얻을 수 있었다. 하지만 BeautifulSoup 라이브러리로는 얻을 수 없는 것들이 많았다. 대표적으로 음식점 추천은 따로 웹페이지를 키고 '화살표' 버튼을 클릭하여 데이터를 추가적으로 얻어야 원활한 데이터 수집이 가능했다. 이와 같은 경우에는 Selenium으로 웹페이지를 띄워 데이터를 얻었으며 음식점 정보, 네이버 지도 링크 까지 수집하여 챗봇이 내보내는 형태로 설정하였다.
Selenium을 통해 동아리 정보를 얻어내는 과정에서 Chrome의 페이지 크기에 따라 Xpath(html언어에서 원하는 정보가 있는 위치)가 달라지는 것에 어려움이 생겼다. 원래는 options.headless = True를 통하여 웹페이지를 따로 열지는 않고 웹스크래핑을 진행하였다. 하지만 이럴 경우 Chrome이 Full Screen이 아니

기 때문에 동아리 정보의 Xpath가 달라져서 배치가 새롭게 된다. 그렇기 때문에 headless를 설정하지 않고 Full screen으로 만들어준 다음에 따온 Xpath로 웹스크래핑을 진행하였다.
원래는 채팅 문장을 맞춤법 검사한 후 형태소 분석을 진행하였다. 하지만 맞춤법 검사하는 라이브러리는 치명적인 단점이 존재했다. 대부분의 맞춤법 검사(예 : hanspell)은 웹사이트의 맞춤법 검사기에 의뢰해 그 출력 값을 따오는 방향으로 진행된다. 이럴 경우 '텔동' 과 같이 학생들 사이에서 많이 쓰이는 고유명사(건물이름/대학용어)가 다르게 변하는 문제가 발생한다. 따로 딕셔너리 처리가 불가능 하기 때문이다. 이에 맞춤법 검사를 진행하지 않기로 하였으며 맞춤법 검사를 하지 않더라도 정확하게 의도를 분류할 수 있는 것을 확인하였다.

알고리즘의 경우, 사용자가 문장을 입력하는데 있어 오타를 입력할 것을 고려하여 맞춤법 검사 기능을 넣으려 하였으나, 위 데이터 처리단계에서 맞춤법 교정을 하지 못한 것과 동일한 이유로 기능을 추가하지 못하였다. 다음으로 문장 내부 개체명 인식 기능을 구현하지 못하였다. 본래 챗봇에서는, 입력된 문장에서 각 성분의 의미를 인식하여 보다 높은 문장 인식 기능을 사용한다. 하지만, 우리가 제작하는 챗봇의 경우 질문과 답변이 단답형 구조로 되어있어 이러한 과정을 거칠 경우 생략된 성분이 너무 많아 구현이 불가했다. 특히 질문의 맥락이 매번 상이했기에(예:음식 주문의 경우 음식명, 개수, 시간을 인식하면 되지만 대학교 정보는 이러한 정해진 객체가 없다) 해당 기능 구현을 포기할 수 밖에 없었다. 이에 단순히 문장에서 추출된 핵심단어로부터 유사도 검사를 실시하는 방법을 선택했다.
객체인식 코드의 삭제는 질문의 적절한 답변을 선별하는데 있어 정확도를 크게 떨어트리는 요인으로 작용하였다. 때문에, 인식률을 높이기 위해 보다 많은 데이터의 양이 필요했고 이는 작업속도를 저하시키는 원인이 되었다.
위에서 언급된 바와 같이, 답변 데이터의 경우 주기적으로 갱신되어 최신화 될 필요가 있다. 때문에, 우리는 Django 플랫폼을 통해 데이터를 저장하는 웹페이지를 개설하였고 어플리케이션을 실행할 때 마다 최신 데이터를 해당 사이트로부터 다운받는 방식을 시도하였다. 하지만, 위 문제점들을 해결하는 과정에서 너무나 많은 시간이 소요되어 해당 기능을 완성하지 못하였다. 현재 개설된 웹페이지는 답변 데이터를 개별적으로 추가하고 한번에 외부 파일로 저장하는 용도로 사용된다. 이는 데이터를 엑셀을 열어 매번 수정하는 것이 아닌, 사이트에 정보를 갱신해두고 주기적으로 관리자가 Export 버튼을 눌러 데이터 파일을 쉽게 교체할 수 있게 한다. 해당 과제는 올해 여름 중으로 새롭게 시도할 예정이다.
제작된 챗봇은, 컴퓨터의 명령프롬프트창(검은색 명령어 화면)이 아닌 모바일 화면으로 메신저 형태로 제공을 위해 웹페이지에 이식할 필요가 있었다. 해당 과정에서, 팀원들이 Html/css/java 관련 지식 부족으로 인해 UI 개발에 많은 시간이 소요되었다. Python 환경의 프로그램을 모바일 환경으로 제공하기 위해 Django플랫폼에서 작업을 하였으나, 해당 과정에서도 많은 문제가 발생하였다.
우선 Html과 view.py에서의 데이터 상호 작용과 Html에서 데이터의 처리 문제가 있다. 첫 번째의 경우는 Html과 파이썬의 데이터를 주고 받는 형태이다. 원래의 경우는 jsonresponse를 바로 이용하는 방법을 사용하였다. 즉 view.py의 return 값을 jsonresponse(string_data)의 형태로 하였다. 하지만 이 경우를 사용하게 되면, 데이터를 string(문자열)으로 넘기기는 하지만 Html에서는 그 값을 인식하지 못한다(보통의 경우는 undefined라는 출력이 나오게 된다). 이는 Django 내에서의 호환 문제가 있어 인식이 불가한 것으로 유추된다. 이를 해결하기 위한 방안으로는 jsonresponse(string_data)의 형태를 바로 return 하는 것이 아닌 json.dump를 이용하여 한번 더 전처리를 거치는 것이다. 이를 사용하게 되면, 데이터 또한 string 으로 인식하는 것을 볼 수 있었다. 다만 Html에서 인식하는 형태는 string에 ""가 붙어서 나오는 문제가 있었다. 이를 해결한 방안은 javascript의 replace함수를 이용하는 것이다. replace함수의 경우 지정 문자열의 형태(혹은 정규식의 형태)를 인식하여 원하는 문자열로 바꿔주는 역할을 한다. 이를 통해서 챗봇 내에서의 답변에서도 깔끔한 정보를 넘겨 줄 수 있었다.

Django를 통해 웹페이지 형태로 제공되는 서비스를 모바일 환경으로 이식하는 과정에서는 개발 환경 및 디자인 부분에서 많은 문제를 겪었다. 가장 먼저 시도하였던 appinventer2의 경우, 블록코딩 형태로서 채팅창 디자인을 간편하게 제작할 수 있었으나 그 형태가 단조롭고 디자인제작에 있어 한계가 있었기 때문에 사용이 제한된다고 판단하였다.
다음으로 시도하였던 replit의 경우, 알고리즘 제작과정에서 사용된 python파일과의 연동은 Visual studio code가 더 적합하다고 판단하였다. 그래서 제작한 채팅창 템플릿의 미리보기 용도로써 사용할 수 밖에 없었다.
어플을 제작한 후에 해당 어플을 실행시켜 보니 구상했던 화면과 다르게 아이콘의 크기, 배열 등이 모두 엉망이었다. 그래서 어플 상 화면을 비교하면서 replit 플랫폼의 코드를 수정하였다. 해당 플랫폼에서 출력되는 채팅창의 화면은 PC환경에서 배포된 URL에 접근하였을 때 확인할 수 있었던 화면이었기에 어플 상 화면에 보이는 화면과 호환되지 않는 문제가 있었고, 대부분의 사용자가 모바일 어플을 사용할 것이라는 판단하에 모바일 환경에 최적화된 웹페이지로 수정하였다. 모든 기종에서 적용이 가능하게끔 Html코드를 수정하고 어플에 적용시키는 모든 과정을 반복하며, 어플에 접근한 채팅창 화면이 사용자가 보기에 적절하도록 크기와 배열 등을 모두 수정하였다.
마지막으로, 제작한 서비스를 인터넷/어플로 배포하는 단계에서 발생한 문제점이다. 서버사용에 있어서 첫번째 과제는 사용자와 데이터베이스(Django)간 속도이고, 두번째는 여러 사용자가 동시 접속하여 병렬로 문제를 처리하는 것이다. 가장 먼저 사용한 서버의 배포플랫폼은 herok(헤로쿠)이다. herok의 경우는 대여 서버를 이용하여, Django를 연결할 수 있다. herok의 장점은 많은 사용자에 대하여 병렬로 연결이 가능한 서버라는 점이다. 하지만 Django가 Html, python, css등 다양한 언어를 지원하는 반면 herok은 그렇지 못해 호환성 문제가 발생하였다. Django 를 업로드 하는 과정에서, 필요 패키지와 실제 파일에 적용되는 패키지의 호환이 이루어지지 않았고 결국 heroku 를 사용하는 데에는 무리가 있다고 판단 하였다.
다음은 amazon에서 배포하는 대여 서버인 AWS를 사용하고자 하였다. AWS는 합리적인 가격에, 개발자는 서버의 관리부담이 크게 줄어들며 병렬처리가 가능하다. 하지만 기존의 개발환경과의 충돌로 인해 적용되지 못하였다. 이는 팀원들의 개발환경이 노트북이다 보니 컴퓨터의 성능을 보완하기 위해 가상환경에서 프로젝트를 진행하였는데 AWS가 가상환경을 지원하지 않았기 때문이다(정확히는, 페이지의 개설은 가능하나 병렬처리 및 배포가 불가 했다).
결국 최종적으로 ngrok을 채택하였다. ngrok의 경우에는 Django내에서 생성된 웹서버가 개발자의 컴퓨터(local server)에서만 작동이 가능하다는 점을 보완할 수 있다. 해당 컴퓨터 방화벽을 우회하여, 외부에서 접속이 가능하게끔 하는 것이다. 제한적으로 챗봇기능에 한해 개발자의 컴퓨터에 접속이 가능하게끔 하여, 사용자들은 원격 조종하듯이 서버에 접속 및 서비스 이용이 가능하다.

3.3 향후 개선 방향

본 프로젝트는 모바일 / 웹 환경 개발이 익숙하지 않은 팀원으로서 많은 시행착오를 겪을 수 밖에 없었으며, 본래 계획과 달리 개발 방향을 일부 수정하는 등 향후 개선해야할 과제를 일부 남겼다.

첫 번째는 서버의 클라우드화를 시도해야 한다. 현재의 ngrok방식은, 개발자의 컴퓨터에서 프로그램을 명령어로 실행하면 웹페이지가 열리며 이를 외부에서 접속하는 방식이다. 이는 결과적으로 로컬서버로 운영되는 서비스를 의미하며, 개발자가 컴퓨터를 종료할 경우 시스템을 사용할 수 없게 된다. 또한, 하나의 컴퓨터에서 통신과 데이터 처리를 함께 진행하기에 병렬 연산에 취약성을 가진다. 개발된 챗봇의 경우 이동하는 트래픽 양이 많지 않아 현재로서는 문제가 되지 않으나 이후 사용자가 대폭 증가하거나 서비스 범위가 확장되면 문제가 될 수 있다.
이러한 문제를 해결하기 위해 AWS를 사용하면, 서버를 클라우드 방식으로 호스팅 할 수 있다. 하나의 컴

퓨터에서 작동하는 프로그램이 아닌 동일한 시스템을 수많은 포트로 병렬화 하여 서비스를 제공할 수 있으며, 병렬 연산에서 큰 강점을 가진다. 또한 서버가 클라우드로 운영되기에 개발자의 컴퓨터가 종료되어 있어도 서비스 운영이 가능하다. 본래 AWS방식을 시도했으나, 기존의 개발환경과의 충돌문제로 AWS 사용에 맞게끔 환경을 모두 바꾸기에는 주어진 시간이 충분치 못하였다. 따라서, 향후 본 서비스를 본격적으로 배포하기에 앞서 AWS에 맞게끔 환경을 변화시켜야 한다.

두 번째는 데이터의 최신화 과정이다. 본래 정보를 제공하는데 있어, Django를 통해 데이터를 저장할 웹사이트를 개설하고 답변에 필요한 데이터를 저장하였다. 이를 통해 인터넷 커뮤니티 게시판에 글을 쓰는 것처럼, 새롭게 데이터를 추가하고 삭제하는 과정에서 편의성을 가진다. 또한 본 웹사이트가 코드형식으로 작동한다는 점에서 내부의 정보를 파이썬에서 코드를 통해 접근이 가능하다. 따라서 매크로 또는 알고리즘 내부 코드를 통해 주기적으로 신규 정보를 게시판에 업로드하며 관리를 할 수 있다. 해당 기능을 사용해 매번 어플리케이션을 시작하면, 데이터 웹사이트로부터 매번 최신화 된 데이터를 다운받고 실행되게끔 하는 것이 목표였다. 하지만, 개발 언어의 역량 부족과 개발 시간의 부족으로 해당 기능을 구현하지 못하였다. 현재로서는 기존에 준비된 약 12,000개의 데이터와 실시간 웹스크래핑을 통해 사용자에게 정보를 전달하고 있다. 답변이 출력되는데 까지 걸리는 시간과, 내용의 정확성을 따져 서비스 운영에 어려움이 있을 경우 해당 부분을 보완하여 데이터 최신화 프로세스를 개발해야 할 것이다.

세 번째는 채팅 어플리케이션의 UI 개선이다. 현재 사람들이 사용하는 일반적인 메신저(K사)의 제품과 비슷한 채팅화면을 제공하기 위해 노력했으나, 기업에서 제공하는 챗봇 개발툴 없이 직접 처음부터 제작하려다 보니 기능적 한계에 달하였다. 출력되는 메시지는 말풍선 형태로 감싸기보다는 바탕색을 다르게 씌움으로써 입력과 출력을 구분하였고, 대화창 옆에 나타나는 프로필 아이콘 등의 일부 기능이 상실되었다. 이는 내용 전달에 있어 가독성을 높이는 장점도 있지만, 보다 친숙한 디자인 제공이 목표였던 만큼 향후 개선의 의사가 있다. 이외에도 화면의 이동에 있어 부드러운 애니메이션을 삽입하거나, 깔끔한 그래픽을 제공하는 것 역시 현재로서는 CSS언어가 익숙하지 못해 부자연스러운 면이 있으나 해당 부분 역시 보완할 계획이다. 채팅을 작성할 때 키보드가 화면을 가리는 문제, 채팅 답변으로 제공된 링크를 접속하고 나오면 채팅 내역이 삭제되는 문제 및 대화 중 글자 크기가 갑자기 확대되는 일부 문제점을 발견하였으나 해결 방법을 찾지 못한 상황이다.
해당 문제는 Html 언어를 학습하여 향후 점진적으로 개선해야 하는 과제로 남아있다.

4. 사용한 제품 리스트(제품명, 제품 링크 포함)

유료 서비스에 한정하여 작성함.

제품명	개수	가격	제품 링크
ngrok	1 (Account)	$10 (monthly)	https://ngrok.com/

5. 기타(소스코드)

어플 출력을 위한 Html 작성 코드

```
{% load static %}
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>아주대 챗봇</title>
    {% block script %}
    <script>

    function getCookie(name) {
            var cookieValue = null;
            if (document.cookie && document.cookie !== '') {
                var cookies = document.cookie.split(';');
                for (var i = 0; i < cookies.length; i++) {
                    var cookie = jQuery.trim(cookies[i]);
                    if (cookie.substring(0, name.length + 1) === (name + '=')) {
                        cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
                        break;
                    }
                }
            }
            return cookieValue;
        }

        var csrftoken = getCookie('csrftoken');

        var xhr;
    function help(){
        alert("                    ---------사용 가이드----------\n\n 1. 공지사항을 원하면 공지사항이라 적어줘! \n\n 2. 학교 건물 위치를 물어 보고 싶으면 OO관 어딨어? 라고 해줘(ex. 남제관 어딨어?) \n\n 3. 학식/교식 메뉴가 궁금해도 물어봐!!\n\n 4. 날씨/ 코로나 확진자수도 대답해줄게\n\n 5. 오늘/어제/그저께 공지사항이라고 하면 그 기간 공지사항 알려 줄게!!\n\n 6. 괄호 안에 단어중 하나를 골라서 \"추천 OOO\"으로 물어봐\n\n(카페,고기집, 삼겹살, 피자, 김치찌개, 짜장면, 떡볶이, 돼지갈비, 부대찌개, 케이크, 브런치, 닭발, 쭈꾸미, 파스타,샌드위치, 칼국수, 스테이크, 감자탕 , 돈가스, 횟집, 한식, 일식, 중국집) ")
    }

    function sendAsk(){
        chattext = document.getElementById("chattext").value;
        if(chattext == ''){
            document.getElementById("chattext").focus();
            return false;
        }
        addtext = "<div style='margin:30px 0;text-align:right;font-size: 350%;'><span style='padding:3px 10px;background-color:#005BAC;outline-color:#005BAC; border-radius: 2rem;color:#FFFFFF;'>" + chattext + "</span></div>";
        document.getElementById("chatbox").innerHTML += addtext;
```

```
        var mydiv = document.getElementById("chatbox");
        mydiv.scrollTop = document.getElementById('chatbox').scrollHeight;

        var strurl = "chatanswer?questext=" + chattext;
        xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4) {
                var data = xhr.responseText;
                var obj = JSON.parse(data);
                var obj_re = obj.replace(/"/g,"")
                var obj_checked = obj_re.match(/http/)
                var obj_checked_notice = obj_re.match("notice_test")
                var find_word = /[A-Z+a-z+0-9+\`+\~+\@+\#+\$+\%+\^+\&+\*+\(+\)+\-
+\=+\++\\+\{+\}+\[+\]+\;+\_+\:+\<+\>+\.+\?+\/+\s]/g;
                var find_word2 = /[ㄱ-ㅎㅣㅏ-ㅣ|가-힣]/

                if (obj_checked == null && obj_checked_notice == null){
                    bottext = "<div style='margin:15px 0;text-align:left;font-size: 350%;'> <span style='padding:3px
10px;background-color:#FFA409;outline-color = #005BAC; border-radius: 2rem;color:#000000;'>" + obj_re +
"</span></div>";
                    document.getElementById("chatbox").innerHTML += bottext;

                }

                else if (obj_checked != null && obj_checked_notice != null){
                    obj_re = obj_re.replace("notice_test","") //여기는 찾아서 링크랑 내용 주는거
                    var obj_split = obj_re.split(',')
                    for(i=0;i<obj_split.length;i++){

                        if(obj_split[i].match(find_word2) != null){
                            bottext = "<div style='margin:15px 0;text-align:left;font-size: 150%;'> <span
style='padding:3px 10px;background-color:#FFA409;outline-color = #005BAC; border-radius: 2rem;color:#000000;'>"
+ obj_split[i] + "</span></div>";
                            document.getElementById("chatbox").innerHTML += bottext;
                        }
                        else if(obj_split[i].match(find_word2) == null && i>0 &&obj_split[i].match(/http/)!= null){
                            obj_split[i] = obj_split[i].replace("]","")
                            var sent1 = "<div style='margin:15px 0;text-align:left;font-size: 150%;'> <span
style='padding:3px 10px;background-color:#f0b379;outline-color = #005BAC; border-radius: 2rem;color:#000000;'>"
+ "여기 누르면 이동해!" + "</span></div>";
                            var test1 = sent1.link((obj_split[i]))
                            document.getElementById("chatbox").innerHTML += test1;

                            bottext = "<div style='margin:15px 0;text-align:left;font-size: 150%;'> <span
style='padding:3px 10px;background-color:#FFA409;outline-color = #005BAC; border-radius:
2rem;color:#000000;'></span></div>";
                            document.getElementById("chatbox").innerHTML += bottext;
                        }
                    }

                }



                else{

                    var obj_find = obj_re.match(find_word)
                    var obj_join = obj_find.join()
                    var obj_fixed = obj_join.replace(/,/g,"")
                    // var URL_remove = obj_re.replace(find_word,'.')
                    bottext = "<div style='margin:15px 0;text-align:left;font-size: 350%;'> <span style='padding:3px
10px;background-color:#FFA409;outline-color = #005BAC; border-radius: 2rem;color:#000000;'>" + obj_re +
"</span></div>";
                    document.getElementById("chatbox").innerHTML += bottext;

                    var sent = "<div style='margin:15px 0;text-align:left;font-size: 350%;'> <span style='padding:3px
10px;background-color:#FFA409;outline-color = #005BAC; border-radius: 2rem;color:#FFFFFF;'>" + "여기 누르면 이
```

```
동해!" + "</span></div>";
                        var test = sent.link((obj_fixed))
                        document.getElementById("chatbox").innerHTML += test;


                    }
                    var objDiv = document.getElementById("chatbox");
                    objDiv.scrollTop = objDiv.scrollHeight;
                    document.getElementById("chattext").value = "";
                }
            };
            xhr.open("GET", strurl);
            xhr.setRequestHeader("X-CSRFToken", csrftoken);
            xhr.send(null);

        }


        </script>
        {% endblock %}
        <!--send 버튼 누르면 작동되는 곳-->
        {% block styles %}
        <style>
            .chatheader {
                position: fixed;
            left: 0;
            top: 0;
            width: 100%;
            padding:10px 0;
            background-color:#005BAC;
            color: #000;
            text-align:center;
            }
            .chatfooter {
                position: fixed;
                left: 0;
                bottom: 0;
                width: 100%;
                padding: 20px 0;
                background-color:#FFF;
                color: rgb(88, 58, 58);
                text-align: center;
            }

        </style>
        {% endblock %}
</head>

<body style="height:100%;background-color: #DDD;">


        <div class="chatheader">
            <table width ="100%">
                <tr>
                    <td width="15%"></td>
                    <td>   <img src="https://i.ibb.co/svPj5mx/123.png" alt="123" border="0">




</td>

                    <td width="70%" align="left" style="color:white; font-family:arial; font-size: 350%;">       아주대 정보
챗봇 </td>
                    <td width ="15%" align="left" style="color:white; font-family:arial; font-size: 350%;" onclick="help()"
id="helpbtn">?</button></td>
                    <td></td>
                </tr>
```

```
        </table>
    </div>
    <div id="chatbox" style=height:1500px;margin-top:200px;padding:20px;font-size:100%;background-color:#BB;overflow-y:scroll;overflow-x:hiddenl;word-wrap:break-word;max-height:100vh;"></div>
    <div class="chatfooter">
        <table width="100%">
            <tr>
            <td width="80%">
                <input id="chattext" placeholder="  내용을  입력하세요." style="padding:20px 0;width:100%;font-size:350%;border:solid 0px #FFF;">

            </td>
            <td width="20%"><button style="padding:20px 0;width:70%;background:#1e5496;color:white;border:solid 0px #FFF;border-radius:5px 5px 5px 5px;font-size:350%;" onclick="sendAsk()" id="sendbtn">전송</button></td>
            </tr>
        </table>
    </div


</body>
</html>


<script>

    var input = document.getElementById("chattext");

    input.addEventListener("keyup", function(event) {
        if (event.keyCode === 13) {
        document.getElementById("sendbtn").click();
        }
    });

    </script>
```

데이터 수집을 위한 에브리타임 댓글 스크래핑 코드

```
from selenium import webdriver
import time
import pandas as pd

browser = webdriver.Chrome("C:/Users/Bang/Desktop/chat/chromedriver.exe")
browser.get("https://everytime.kr/385892/p/1")

# 로그인
browser.find_element_by_xpath("//*[@id='container']/form/p[1]/input").send_keys("ididid")
browser.find_element_by_xpath("//*[@id='container']/form/p[2]/input").send_keys("password!")
browser.find_element_by_xpath("//*[@id='container']/form/p[3]/input").click()

browser.implicitly_wait(10)
f = open("test_data3.csv",'w',encoding = 'utf-8')

page = 1
while(page<560):
    for row in range(1,21):
            time.sleep(0.1)
            xp = "//*[@id='container']/div[3]/article[{}]/a/p".format(row)
            content = browser.find_element_by_xpath(xp)
            arr=''
            # 질문 추출
            if(content.text.find('?')!=-1):
                # 댓글 개수 이용
                count =   browser.find_element_by_xpath('//*[@id="container"]/div[3]/article[{}]/a/ul/li[2]'.format(row))
```

```
                c = int(count.text)
                time.sleep(0.05)
                con = content.text
                con = con.replace('\n','')
                arr = arr + con +'\t'
                content.click()
                time.sleep(0.2)

                # 댓글 수집
                for i in range(1,c+1):
                    path = "//*[@id='container']/div[3]/article/div/article[{}]/p".format(i)
                    time.sleep(0.1)
                    comment =   browser.find_element_by_xpath(path).text
                    arr = arr + comment+'\t'
                browser.find_element_by_id("goListButton").click()
                arr = arr+'\n'
                f.write(arr)
    page += 1
    browser.get('https://everytime.kr/385892/p/{}'.format(page))
    time.sleep(0.1)
f.close()
```

데이터 전처리 관련 코드

```
import pandas as pd

# 데이터 입력 및 확인
data_real = pd.read_excel('./real_final_data2 (7).xlsx')
data = data_real.fillna("")
data['label'].unique()

# 필요 없는 행 및 index 최신화
data.sort_values('label',inplace=True)
data.reset_index(drop=True, inplace=True)
data.drop(columns=['Unnamed: 0'],inplace=True)

# label 정보 histogram으로 확인
import matplotlib
import matplotlib.pyplot as plt
plt.hist(data['label'])

# data.grouby로 정보제공만 모음
data2 = data.groupby('label').get_group(2)
data4 = data2

# 세부 분류인 label2 에서 결측치를 -1로 대체
for i in range(len(data2['label'])):
    if data2['label2'][i+4516] == "":
        data3 = data2.drop(index=i+4516)
        data4['label2'][i+4516]=-1

# 필요 없는 행 삭제 및 데이터 확인
data.drop(columns=['predict_label','predict_label2','checked_question'],inplace=True)
data

# label2 에서 결측치를 -1로 처리한 data4를 data에 저장
data['label2'] = data4['label2']
data.fillna(-1,inplace=True)

# label2를 오름차순으로 정렬하고 index를 재지정 및 label2의 시작점 확인
data_real = data.sort_values('label2').reset_index(drop=True)
data_real # 마지막 index가 end
```

```
# label2에서 0으로 시작하는 지점의 index를 확인
data_real.groupby('label2').get_group(0) # 첫 시작 index가 start

# 학습을 위해 label2중에서 0~4의 데이터를 받아 data3에 저장
start = 11455
end = 12580+1
data3 = data_real.iloc[start:end]
data3

# label2의 값이 있는 채팅만 형태소 분석한 뒤 data_real에 저장
from konlpy.tag import Komoran
import re

data_real['V']= ''
data_real['N']= ''
data_real['M'] = ''
data_real['X'] = ''
data_real.fillna("",inplace=True)

class Preprocess:
    def __init__(self, userdic = None):
        self.komoran = Komoran(userdic=userdic)
        self.exclusion_tags = [
            'JKS','JKC', 'JKG', 'JKB', 'JKV', 'JKQ', 'JX', 'JC', 'SF', 'SP', 'SS', 'SE', 'SO', 'EP', 'EF', 'EC', 'ETN', 'ETM',
'XSN', 'XSV', 'XSA'
            ]

    def pos(self, sentence):
        return self.komoran.pos(sentence)

    def get_keywords(self, pos, without_tag = False):
        f = lambda x : x in self.exclusion_tags
        word_list = []
        for p in pos:
            if f(p[1]) is False:
                word_list.append(p if without_tag is False else p[0])
        return word_list

p = Preprocess(userdic='./user_dic.tsv')


m = re.compile("^N")
n = re.compile("^V")
o = re.compile("^X")

#len(data['question']

for i in range(end-start):
    sent = data_real['question'][i+start]
    pos = p.pos(sent)
    for k in range(len(pos)):
        if m.search(pos[k][1]) is not None:
            data_real['N'][i+start] = data_real['N'][i+start]+" "+pos[k][0]
        elif n.search(pos[k][1]) is not None:
            data_real['V'][i+start] = data_real['V'][i+start]+" "+pos[k][0]
        elif o.search(pos[k][1]) is not None:
            data_real['X'][i+start] = data_real['X'][i+start]+" "+pos[k][0]
    print(i+1,"/",end-start)

data_real

# 전처리한 데이터를 엑셀에 저장
data_real.to_excel('data_for_chat.xlsx')
```

딥러닝 관련 코드

```
# label(의도 분석 모델)
# 필요한 모듈 임포트
import pandas as pd
import tensorflow as tf
from tensorflow.keras import preprocessing
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Dense, Dropout, Conv1D, GlobalMaxPool1D, concatenate

# 데이터 읽어오기
features = data_real['question'].tolist()
labels = data_real['label'].tolist()

# 단어 인덱스 시퀀스 벡터
corpus = [preprocessing.text.text_to_word_sequence(text) for text in features]

tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(corpus)
sequences = tokenizer.texts_to_sequences(corpus)
word_index = tokenizer.word_index
MAX_SEQ_LEN = 15  # 단어 시퀀스 벡터 크기
padded_seqs = preprocessing.sequence.pad_sequences(sequences, maxlen=MAX_SEQ_LEN, padding='post')

# 학습용, 검증용, 테스트용 데이터셋 생성 ❸
# 학습셋:검증셋:테스트셋 = 7:2:1
ds = tf.data.Dataset.from_tensor_slices((padded_seqs, labels))
ds = ds.shuffle(len(features))
train_size = int(len(padded_seqs) * 0.7)
val_size = int(len(padded_seqs) * 0.2)
test_size = int(len(padded_seqs) * 0.1)
train_ds = ds.take(train_size).batch(20)
val_ds = ds.skip(train_size).take(val_size).batch(20)
test_ds = ds.skip(train_size + val_size).take(test_size).batch(20)

# 하이퍼파라미터 설정
dropout_prob = 0.5
EMB_SIZE = 128
EPOCH = 5
VOCAB_SIZE = len(word_index) + 1  # 전체 단어 수

# CNN 모델 정의
input_layer = Input(shape=(MAX_SEQ_LEN,))
embedding_layer = Embedding(VOCAB_SIZE, EMB_SIZE, input_length=MAX_SEQ_LEN)(input_layer)
dropout_emb = Dropout(rate=dropout_prob)(embedding_layer)

conv1 = Conv1D(filters=128, kernel_size=3, padding='valid', activation=tf.nn.relu)(dropout_emb)
pool1 = GlobalMaxPool1D()(conv1)
conv2 = Conv1D(filters=128, kernel_size=4, padding='valid', activation=tf.nn.relu)(dropout_emb)
pool2 = GlobalMaxPool1D()(conv2)
conv3 = Conv1D(filters=128, kernel_size=5, padding='valid', activation=tf.nn.relu)(dropout_emb)
pool3 = GlobalMaxPool1D()(conv3)

# 3, 4, 5- gram 이후 합치기
concat = concatenate([pool1, pool2, pool3])
hidden = Dense(128, activation=tf.nn.relu)(concat)
dropout_hidden = Dropout(rate=dropout_prob)(hidden)
logits = Dense(4, name='logits')(dropout_hidden)
predictions = Dense(4, activation=tf.nn.softmax)(logits)

# 모델 생성
model = Model(inputs=input_layer, outputs=predictions)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
# 모델 학습
model.fit(train_ds, validation_data=val_ds, epochs=EPOCH, verbose=1)

# 모델 평가(테스트 데이터셋 이용)
loss, accuracy = model.evaluate(test_ds, verbose=1)
print('Accuracy: %f' % (accuracy * 100))
print('loss: %f' % (loss))

# 모델 저장
model.save('cnn_new_model.h5')

————————————————————————————————————————————————————————————————————————————————

# label2 (세부 분류 분석 모델(수강신청, 학교생활 등))
# 필요한 모듈 임포트
import pandas as pd
import tensorflow as tf
from tensorflow.keras import preprocessing
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Dense, Dropout, Conv1D, GlobalMaxPool1D, concatenate

# 데이터 읽어오기
features2 = data3['question'].tolist()
labels2 = data3['label2'].tolist()

# 단어 인덱스 시퀀스 벡터
corpus2 = [preprocessing.text.text_to_word_sequence(text) for text in features2]

tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(corpus2)
sequences2 = tokenizer.texts_to_sequences(corpus2)
word_index = tokenizer.word_index
MAX_SEQ_LEN = 15  # 단어 시퀀스 벡터 크기
padded_seqs2 = preprocessing.sequence.pad_sequences(sequences2, maxlen=MAX_SEQ_LEN, padding='post')

# 학습용, 검증용, 테스트용 데이터셋 생성 ❸
# 학습셋:검증셋:테스트셋 = 7:2:1
ds = tf.data.Dataset.from_tensor_slices((padded_seqs2, labels2))
ds = ds.shuffle(len(features2))
train_size = int(len(padded_seqs2) * 0.7)
val_size = int(len(padded_seqs2) * 0.2)
test_size = int(len(padded_seqs2) * 0.1)
train_ds = ds.take(train_size).batch(1)
val_ds = ds.skip(train_size).take(val_size).batch(1)
test_ds = ds.skip(train_size + val_size).take(test_size).batch(1)

# 하이퍼파라미터 설정
dropout_prob = 0.5
EMB_SIZE = 128
EPOCH = 5
VOCAB_SIZE = len(word_index) + 1  # 전체 단어 수

# CNN 모델 정의
input_layer = Input(shape=(MAX_SEQ_LEN,))
embedding_layer = Embedding(VOCAB_SIZE, EMB_SIZE, input_length=MAX_SEQ_LEN)(input_layer)
dropout_emb = Dropout(rate=dropout_prob)(embedding_layer)

conv1 = Conv1D(filters=128, kernel_size=3, padding='valid', activation=tf.nn.relu)(dropout_emb)
pool1 = GlobalMaxPool1D()(conv1)
conv2 = Conv1D(filters=128, kernel_size=4, padding='valid', activation=tf.nn.relu)(dropout_emb)
pool2 = GlobalMaxPool1D()(conv2)
```

```
conv3 = Conv1D(filters=128, kernel_size=5, padding='valid', activation=tf.nn.relu)(dropout_emb)
pool3 = GlobalMaxPool1D()(conv3)

# 3, 4, 5- gram 이후 합치기
concat = concatenate([pool1, pool2, pool3])
hidden = Dense(128, activation=tf.nn.relu)(concat)
dropout_hidden = Dropout(rate=dropout_prob)(hidden)
logits = Dense(5, name='logits')(dropout_hidden)
predictions = Dense(5, activation=tf.nn.softmax)(logits)

# 모델 생성
model = Model(inputs=input_layer, outputs=predictions)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# 모델 학습
model.fit(train_ds, validation_data=val_ds, epochs=EPOCH, verbose=1)

# 모델 평가(테스트 데이터셋 이용)
loss, accuracy = model.evaluate(test_ds, verbose=1)
print('Accuracy: %f' % (accuracy * 100))
print('loss: %f' % (loss))

# 모델 저장
model.save('cnn_new_label2_model.h5')
```

채팅 답변 파이썬 코드 (1000줄이 넘어서 핵심 코드 및 예시만 따로 넣음)

```
def scholarship():
    scholarship_sum = ["notice_test"]
    for i in range(3):

        url = 'https://www.dreamspon.com/scholarship/list.html?page={}'.format(i+1)

        res = requests.get(url,headers=headers)
        res.raise_for_status() # raise_for_status는 문제가 생기면 바로 에러를 출력
        soup = BeautifulSoup(res.text,"lxml") # res.text를 lxml통해서 beautiful soup 객체로 만듦
        a = soup.find('tbody').find_all("tr")
        for k in range(len(a)):
            # 아래는 제목
            # print(a[0].get_text())
            # # 아래는 링크
            # print('https://www.dreamspon.com'+a[0].find('a')['href'])
            scholarship_sum.append((a[0].get_text()).replace("\n",""))
            scholarship_sum.append('https://www.dreamspon.com'+a[0].find('a')['href'])
    return scholarship_sum

def group(chat):
    # options.headless= False
    browser = webdriver.Chrome(options=options)
    browser.get("https://linkareer.com/list/club")

    browser.maximize_window()

    group_sum = ['notice_test']
    for row in range(5):
        for n in range(4):
            title =
browser.find_element_by_xpath('/html/body/div[1]/div[1]/div[4]/div[4]/div/div[2]/div[1]/div[{}]/div[{}]/div/div[2]/a/h5'.format(r
ow+1,n+1)).text
            d_day =
browser.find_element_by_xpath('/html/body/div[1]/div[1]/div[4]/div[4]/div/div[2]/div[1]/div[{}]/div[{}]/div/div[2]/div/div/h4[1]'.f
```

```
ormat(row+1,n+1)).text
            where =
browser.find_element_by_xpath('/html/body/div[1]/div[1]/div[4]/div[4]/div/div[2]/div[1]/div[{}]/div[{}]/div/div[2]/div/p'.format(r
ow+1,n+1)).text
            link =
browser.find_element_by_xpath('/html/body/div[1]/div[1]/div[4]/div[4]/div/div[2]/div[1]/div[{}]/div[{}]/div/div[2]/a'.format(row
+1,n+1)).get_attribute("href")

            group_sum.append(title)
            group_sum.append(d_day)
            group_sum.append(where)
            group_sum.append(link)

    return group_sum


def chatanswer(request):
    context = {}
    ques = request.GET.get('questext')#html에 있는 strurl지정한 이름에 찾은 다음에 안의 data를 가져온다.
    start_time = time.time()

    chat_input2 = ques
    print(ques)
    number = random.randrange(1,4)

    wevity_web_data = ["웹 공모전","IT 공모전","웹/모바일/IT 공모전","모바일 공모전"]
    if any(format in chat_input2 for format in wevity_web_data):
        wevity_web_data_fun_fun = wevity_web()
        context = wevity_web_data_fun_fun
        middle_time8 = time.time()
        print("최종 시간 : %f"%(middle_time8-start_time))
        return JsonResponse(json.dumps(context,ensure_ascii=False), content_type = "application/json",safe = False)
    else:

        features = data['question'].tolist()
        corpus = [preprocessing.text.text_to_word_sequence(text) for text in features]
        tokenizer = preprocessing.text.Tokenizer()
        tokenizer.fit_on_texts(corpus)

        sent = []
        sent.append(chat_input2)
        chat_input = sent
        sequences2 = tokenizer.texts_to_sequences(chat_input)
        MAX_SEQ_LEN = 15
        padded_seqs = preprocessing.sequence.pad_sequences(sequences2, maxlen=MAX_SEQ_LEN,
padding='post')

        predict = model.predict(padded_seqs)
        predict_class = tf.math.argmax(predict, axis=1)

        if predict_class.numpy() != 2:
            number = random.randrange(1,4)
            if number == 1:
                context = '무슨 말인지 모르겠어 ㅠㅠ'
            elif number == 2:
                context = '이해 못하겠어 ㅠㅠ'
            elif number == 3:
                context = '이해 못해서 답을 못해주겠어 ㅠㅠ'
            middle_time8 = time.time()
            print("최종 시간 : %f"%(middle_time8-start_time))
            return JsonResponse(json.dumps(context,ensure_ascii=False), content_type = "application/json",safe =
False)

        else:
            start = 11455
            end = 12580
            data2 = data.iloc[start:end+1]
```

```
        features = data2['question'].tolist()
        corpus = [preprocessing.text.text_to_word_sequence(text) for text in features]
        tokenizer = preprocessing.text.Tokenizer()
        tokenizer.fit_on_texts(corpus)
        sequences2 = tokenizer.texts_to_sequences(chat_input)
        padded_seqs = preprocessing.sequence.pad_sequences(sequences2, maxlen=MAX_SEQ_LEN,
padding='post')

        predict2 = model2.predict(padded_seqs)
        predict_class2 = tf.math.argmax(predict2, axis=1)

        pos = p.pos(chat_input2)
        bow1 = []

        for i in range(len(pos)):
            if (m.search(pos[i][1])) is not None:
                bow1.append(pos[i][0])
            elif o.search(pos[i][1]) is not None:
                bow1.append(pos[i][0])
            elif n.search(pos[i][1]) is not None:
                bow1.append(pos[i][0])

        r1 = 0
        index=0
        bow2=[]
        bow=[]

        pcm = int(predict_class2.numpy())
        data3 = data2.groupby('label2').get_group(pcm).reset_index(drop=True)
        for i in range(len(data3['question'])):
            bow2 = list(data3['N'][i].split()+data3['V'][i].split())
            bow = bow1 + bow2
            word_dics = []
            for token in bow:
                    if token not in word_dics:
                        word_dics.append(token)
            freq_list1 = make_term_doc_mat(bow1, word_dics)
            freq_list2 = make_term_doc_mat(bow2, word_dics)
            doc1 = np.array(make_vector(freq_list1))
            doc2 = np.array(make_vector(freq_list2))
            r2 = cos_sim(doc1, doc2)

            if r2 > r1 :
                r1 = r2
                index = i
                bow3 = bow2
        if r1 < 0.4 :
            context  = "다르게 말해 줄래? 잘 모르겠어 ㅠ"
            middle_time8 = time.time()
            print("최종 시간 : %f"%(middle_time8-start_time))
            return JsonResponse(json.dumps(context,ensure_ascii=False), content_type = "application/json",safe =
False)

        context =data3['answer'][index]
        middle_time8 = time.time()
        print("최종 시간 : %f"%(middle_time8-start_time))
        return JsonResponse(json.dumps(context,ensure_ascii=False), content_type = "application/json",safe =
False)
```

6. 참고 문헌

(1) 전문 서적

- 조경래,『처음 배우는 딥러닝 챗봇』, 한빛미디어(2020)
- 안드레아스 뮐러, 세라 기아도,『파이썬 라이브러리를 활용한 머신러닝』, 한빛미디어(2021)
- Sumit Raj,『파이썬으로 챗봇 만들기』, 영진닷컴(2020)
- 델립 라오, 브라이언 맥머핸,『파이토치로 배우는 자연어 처리』, 한빛미디어(2021)
- 비슈누 수브라마니안,,『PyTorch로 시작하는 딥러닝』, 에이콘(2019)
- 오렐리앙 제롱,『핸즈온 머신러닝』, 한빛미디어(2021)

(2) 학습 정보 제공 웹사이트

- "Django서버", <파이썬 장고(Django)로 챗봇 만들기[BIPA SORI]>, https://www.youtube.com/watch?v=Ka9kGGfJnCo&t=1952s
- "html, css 기초", <ChatBot using Javascript API>, https://www.youtube.com/watch?v=yXtrlIJzo-A
- "Javascript 기초", <How to create a Chatbot in Javascript[Coding Curry]>, https://www.youtube.com/watch?v=AxjELOePuJg
- "어플제작과정", <웹으로 웹뷰앱을 만들어보자[조코딩 JoCoding]>, https://www.youtube.com/watch?v=GyXLa-0R_S0&t=334s
- "[Django] 게시판 구현 하기", <KyuDevelop>, https://kyuhyuk.kr/article/python/2020/08/14/Django-Board-Write-Post
- "웹스크래핑", <파이썬 코딩 무료 강의 (활용편3) - 웹 크롤링? 웹 스크래핑! 제가 가진 모든 비법을 알려드리겠습니다.>, https://www.youtube.com/watch?v=yQ20jZwDjTE&t=14736s
- "데이터 분석", <파이썬 코딩 무료 강의 (활용편5) - 데이터 분석 및 시각화, 이 영상 하나로 끝내세요>, https://www.youtube.com/watch?v=PjhlUzp_cU0&t=16736s
- "코사인 유사도", <위키백과>, https://ko.wikipedia.org/wiki/%EC%BD%94%EC%82%AC%EC%9D%B8_%EC%9C%A0%EC%82%AC%EB%8F%84
- "Build & Integrate your own custom chatbot to a website (Python & JavaScript)", <Python Engineer>, https://www.youtube.com/watch?v=a37BL0stIuM&t=1527s
- "셀레니움 사용법", <[selenium] 크롤링 selenium 셀레니움 사용법, 명령어 모음>, https://gorokke.tistory.com/8
- "How to create a Messenger Style Chat Bot with JavaScript Tutorial", <Code Palace>, https://www.youtube.com/watch?v=He0xK1x-vnI&ab_channel=CodePalace

(3) 데이터 제공 웹사이트

- “Naver 모바일 뉴스”, <네이버>, https://m.news.naver.com/cluster/main?id=c_202203301440_00000001&sid1=100&oid=023&aid=0003682185
- "교내 공지사항 ", <아주대학교>, https://ajou.ac.kr/kr/ajou/notice.do
- "교내 연락처", <아주대학교>, https://mportal.ajou.ac.kr/system/phone/phone.do
- "학사 Q&A", <아주대학교>, https://www.ajou.ac.kr/kr/bachelor/bamedia-qna.do?auty=2
- "모두의 말뭉치", <국립국어원>, https://corpus.korean.go.kr/main.do#down
- "자유게시판", <아주대학교 에브리타임>, https://everytime.kr/377391

- "새내기 게시판",<아주대학교 에브리타임>, https://everytime.kr/385892
- "가게 추천", <네이버>, https://www.naver.com/
- "공모전", <위비티>, https://www.wevity.com/
- "일반 장학금", <드림스폰>, https://www.dreamspon.com/
- "연합 동아리", <링커리어>, https://linkareer.com/list/club
- "아르바이트", <알바천국>, http://www.alba.co.kr/Main.asp?utm_source=google&utm_medium=paidsearch&utm_campaign=brand&utm_content=pc_cpc&utm_term=%EC%95%8C%EB%B0%94+%EC%B2%9C%EA%B5%AD&gclid=Cj0KCQjw_4-SBhCgARIsAAlegrXFE3G-C56YNvfk1XwXPbDpNRkmA71ccuvAck0S8WvSfabqYtPLK9waAmtbEALw_wcB
- "코로나 확진자수", <네이버검색>, 'https://search.naver.com/search.naver?where=nexearch&sm=top_hty&fbm=1&ie=utf8&query=%EC%BD%94%EB%A1%9C%EB%82%98+%ED%99%95%EC%A7%84%EC%9E%90'
- "날씨", <네이버검색>, 'https://search.naver.com/search.naver?sm=tab_hty.top&where=nexearch&query=%EB%82%A0%EC%94%A8&oquery=%EB%82%A0%EC%94%A8&tqi=hCdp4lprvN8ssfc8iOKsssssol-413115'