2019-2 오픈소스 프로젝트

**방가방가 (서울시 전/월세 예측)**

****

**9조**

201513720 김설매

201714163 김현수

201711309 박현정

201711343 전효진

201711333 이주경

1. **프로젝트 소개 (제안배경 및 개발 목표)**

쉽게 접할 수 있는 부동산 검색 / 계약 어플리케이션들은 허위 매물이 많고 부동산에 대한 지식이 적은 사람들, 특히 직장 근처의 부동산을 구하는 사회 초년생이나 학교 근처의 원룸을 구하는 학생들에게는 난해할 수 정보를 제공한다. 또한 부동산에 직접 방문하기 전에 어플을 통해 사진을 볼 수 있는 시스템을 악용, 부동산 매도인들이 실제 매물의 단점을 감추는 사진만을 업로드하여 매수인들이 시간을 낭비하고, 원하는 매물을 놓치게 되는 경우도 빈번하다.

본 프로젝트에서는 부동산을 계약하고자 하는 사람들이 핵심 조건을 (위치, 면적, 임대형태 등) 입력하는 것만으로도 어느 정도의 예산이 필요할지 예측하는 챗봇 서비스를 제작하는 것을 목표로 하였다.

김설매 조원과 이주경 조원은 챗봇의 인터페이스를, 김현수 조원과 박현정 조원은 사이킷런과 서울시에서 제공하는 데이터를 이용한 머신러닝을, 전효진 조원은 엑소브레인 API를 이용한 챗봇 서비스를 개발하였다.

**2.**   **사용한 오픈 소스 소개**

**2-1. 회귀 분석**

1. scikit-learn (사이킷런)



사이킷런은 파이썬 기반의 머신러닝 라이브러리이다. 분류, 회귀, 클러스터링 등의 기능을 지원하며, 하드웨어 가속을 지원하지는 않지만 입력 데이터가 간단하고 뉴런의 개수가 적은 머신러닝의 경우 텐서플로우보다 적은 자원으로 뛰어난 성능을 보인다.

사이킷런이 지원하는 기능 중 LinearRegression은 종속 변수와 독립변수의 선형 관계를 이용한 예측 알고리즘이고, KNearestNeighbors 알고리즘은 데이터들을 점으로 매핑해 거리가 작은 k개의 점들의 평균을 계산하는 알고리즘이다.

본 프로젝트에서는 종속 변수와 독립 변수의 선형 관계가 뚜렷한 전세 예측에는 Linear Regression 패키지를 사용했다. 월세 예측은 같은 매물이라도 집 주인 마음대로 가격을 책정하기 때문에 평균을 계산하는 의미가 없다고 판단해 KneighborsRegressor 패키지를 이용해 가장 비슷한 매물들의 임대료를 출력했다.

1. GeoPandas

GeoPandas는 파이썬에서 지리정보 데이터를 처리하는 패키지이다. GeoSeries와 GeoDataFrame 기능을 제공하는데, GeoDataFrame의 기능을 사용했다. geopandas를 사용해 구 이름과 좌표 등이 입력되어 있는 서울시 행정구역 데이터 seoul\_municipalities\_geo.json을 읽어와 사용자가 입력한 평수와 건물 유형에 해당하는 서울시 전/월세 지도 이미지를 생성한다.

**2-2. 챗봇**

ETRI의 엑소브레인 프로젝트에서 공개한 언어 분석 API를 이용한다.

언어 분석 API는 HTTP 기반의 REST API 인터페이스로 ETRI에서 제공하는 API Key 인증을 통해 사용 가능한 Open Api이다. 이 모델을 사용하기 위해 우선 공공 인공지능 오픈 API/DATA 서비스 포털(<http://aiopen.etri.re.kr/index.php>)에서 api key를 발급받아야 한다.

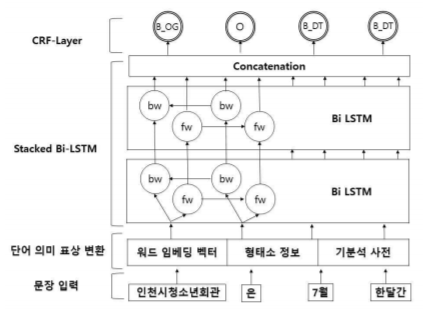
이번 프로젝트는 자연어처리와 임대료 예측 두 단계에 거친 프로그램이다. 임대료 예측에 머신러닝이 쓰이므로 속도와 정확도를 고려할 때 직접 학습을 시켜 개체명을 태깅하는 것보다 api 서버를 이용하는 것이 바람직하다고 판단했다. 따라서 엑소 브레인 api를 이용한 자연어 분석 방법을 채택하게 되었다.

**개체명 인식**

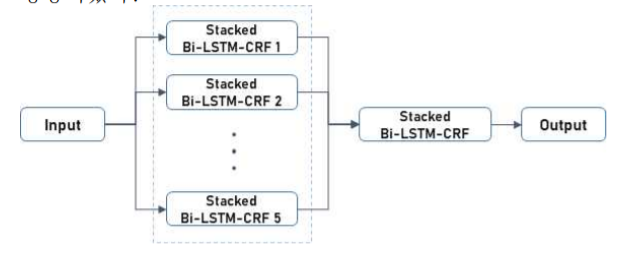
개체명이란 인명, 기관명, 상품명과 같은 고유 명사와, 지명, 주소 등의 공간적 표현과, 날짜, 시간과 같은 시간적 표현 등 문장에서 중요한 역할을 하거나 핵심적인 의미를 지닌 어휘를 말한다. 개체명 인식은 개체명을 문장 내에서 찾고 어떤 의미와 종류인지를 분석하는 방법으로 모든 형태소에 대하여 하나의 태그를 정해주는 시퀀스 라벨링 문제이다. 형태소의 자음, 모음을 이용해 새로운 단어가 등장해도 형태적 유사함으로 단어의 의미를 연관 지을 수 있다. 개체명 태그는 <개체명 태그 세트 및 태깅 말뭉치>를 따른다.

개체명 인식에는 stacked Bi-LSTM-CRF 앙상블 모델을 이용한다.

stacked Bi-LSTM-CRF 앙상블 모델은 기존의 stacked Bi-LSTM-CRF모델에 Bi-LSTM 레이어를 한 층 더 추가해 기존 모델보다 깊은 신경망을 구성한 것이다.



[그림 1-기존 Bi-LSTM-CRF 모델]



[그림2-stacked Bi-LSTM-CRF 앙상블 모델]

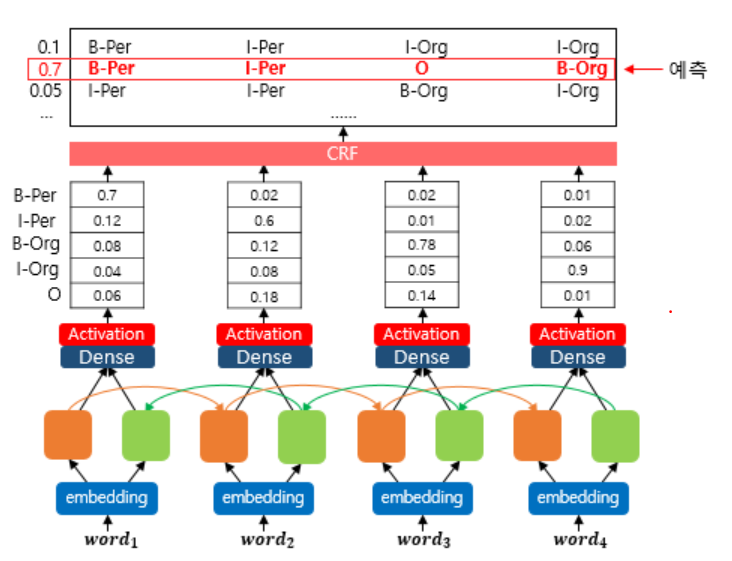
Stacked Bi-LSTM-CRF 모델을 하나의 신경망으로 하여 5개 모델의 결과 값을 합해 다시 한번 Stacked Bi-LSTM-CRF으로 학습, 기존 모델에 비해 2%포인트 정도 더 높은 정확도의 결과를 얻는다.

각각의 신경망에 입력은 그림1과 같이 각 형태소에 대해 아래 3가지 정보로 표현된 156차원의 단어 의미 표상이다.

1. 워드 임베딩 벡터 : 세종 말뭉치, 엑소 브레인 개체명 인식 말뭉치를 이용한 실수형 벡터
2. 형태소 정보 벡터 : 세종 말뭉치 분류를 이용한 one-hot 벡터
3. 기분석 사전 벡터 : 학습 시 3번 이상 같은 개체명으로 주석된 어휘를 표현한 개체명 벡터

3가지의 단어 의미 표상은 Stacked Bi-LSTM 층을 거쳐 CRF 층에 입력된다.

각각의 층에 대한 특징은 아래와 같다.



(사진 출처 : 딥러닝을 이용한 자연어 처리 입문)

**BI-LSTM층**

BI-LSTM모델은 순차적 데이터 활용에 많이 쓰이는 RNN 모델을 개선한 LSTM 모델의 성능을 한번 더 개선한 것이다. LSTM 두개를 함께 학습시켜 이전 시점의 단어 데이터 뿐만 아니라 다음 시점의 단어 데이터도 고려하도록 보완했다. 앞뒤 문맥을 모두 고려할 수 있어 정확한 결과를 얻을 수 있다.

**CRF층**

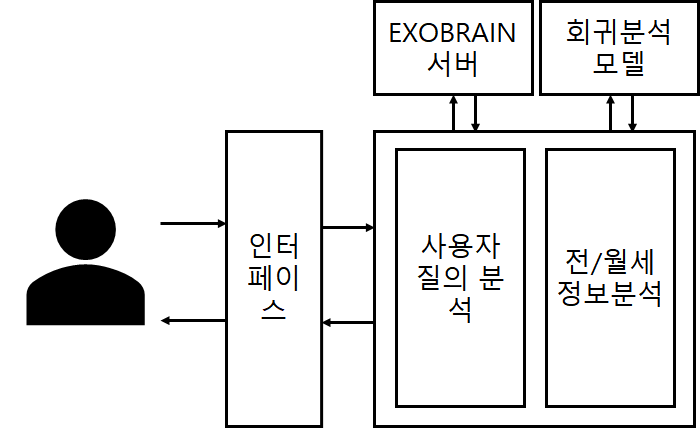
각 단어를 벡터로 입력 받고 출력층 활성화 함수를 통해 개체명을 예측하던 모델에 CRF 층을 추가해 예측 개체명 레이블 사이의 의존성을 고려한다. BIO 표현 방법에 의해 생긴 제약사항을 분석에 반영할 수 있다. BIO 표현은 개체명의 시작 형태소엔 B , 내부 형태소엔 I, 개체명에 속하지 않은 형태소엔 O 태그를 부여하는 방법이다. 이 표현을 사용하면 문장 첫번째 단어에서는 I가 나오지 않고, O-I 패턴은 나오지 않는 등의 규칙이 생긴다. CRF 층을 통해 출력 레이블에 대한 양방향 문맥 반영이 가능하다.

\*논문 인용

<Stacked Bi-LSTM-CRF 앙상블을 사용한 개체명 추출(우수발표논문상), 장윤정, 2018>.

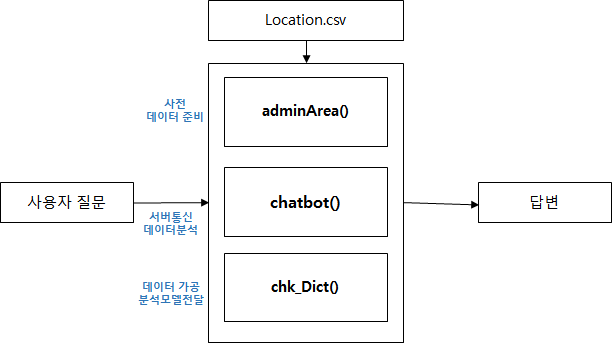
**3.**   **설계 및 개발 내용**

**3-1. 시스템 아키텍처**



**[전체 시스템 구조]**

1. **챗봇 모듈**

****

**[챗봇 작동 과정]**

1. adminArea(): 사전 데이터를 준비하는 과정.

서울특별시 자치구 및 법정동명 데이터인 location.csv 파일을 읽어 25개 자치구 별 법정동 명을 저장하는 guDongDict 딕셔너리를 생성한다. guDongDict으로 사용자가 구를 입력하지 않고 동만 입력한 경우 재입력을 요구할 필요 없이 해당하는 구를 찾아 분석 모델에 전달한다.

자치구 명과 법정동명을 각각 따로 저장하는 guSet, dongSet 집합을 생성한다. 분석 모델에는 사용자 입력에서 구와 동을 분리해 전달해야한다. 사용자 질문 분석 결과로 얻은 주소 개체명이 guSet, dongSet 중 어떤 집합에 속하는지를 판단해 구와 동을 구분한다.

1. chatbot(): 사용자 입력 질의를 분석하고 답변을 추출하는 주요 함수.

다음과 같은 파이썬 딕셔너리 객체를 JSON 형식으로 인코딩하여 HTTP 통신으로 서버에 언어 분석을 요청한다.

 requestJson = {

        "access\_key": 사용자가 발급받은 API KEY,

        "argument": {

            "text": 사용자 질문,

            "analysis\_code": 분석 항목 코드

        }

    }

언어 분석 API는 형태소, 동음이의어, 다의어, 의존구문 분석, 개체명 분석 이 가능하다. 챗봇 모델은 개체명 분석을 주로 사용하므로 analysis\_code를 ner로 설정해 분석을 요청한다.

-서버가 응답한 데이터는 다음과 같은 JSON 형식의 텍스트이다.

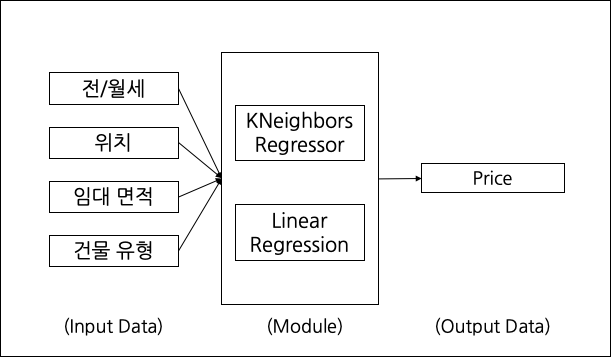
(질문 예시: 광진구 화양동에 있는 오피스텔 15평짜리 월세 가격은 얼마인가요?)

  JSON 객체를 딕셔너리로 디코딩해 분석을 진행한다. 개체명 인식 결과인 “NE”의 value에서 구, 동, 평수 정보를 추출하고, 어휘 의미 분석 결과인 “WSD”의 value를 이용해 전월세, 주거 형태를 추출한다. 추출한 정보는 qDict 딕셔너리에 저장한다. 사용자가 구를 입력하지 않은 경우 adminArea()에서 사전 준비한 구동딕셔너리를 이용해 구를 찾아 qDict 딕셔너리에 추가한다.

1. chk\_Dict(): 분석에 필요한 모든 정보가 딕셔너리에 저장되었는지 확인.

회귀 분석 모델은 ‘자치구 명’, ‘법정동 명’, ’평수’, ’전월세’, ’거주형태(아파트, 오피스텔, 빌라)’ 5가지 정보를 바탕으로 분석을 진행한다. chatBot()에서 사용자 입력을 분석해 생성한 qDict 딕셔너리가 분석에 필요한 모든 정보가 저장되어 있는지 확인하고 누락된 정보가 있을 경우 사용자에게 재입력을 요구한다. qDict 딕셔너리에 모든 정보가 입력된 경우 전세와 월세를 구분해 분석 모델의 분석 함수를 호출, qDict 딕셔너리를 전달한다.

**2)**   **분석 모듈**

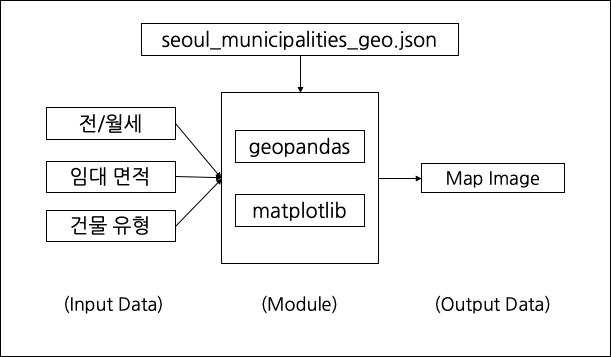
****

**[임대료 예측 과정]**

먼저 모델을 훈련시키는 입력 데이터셋으로는 서울시열린데이터광장에서 제공하는 서울특별시 전월세가 정보(<http://data.seoul.go.kr/dataList/datasetView.do?infId=OA-15549&srvType=S&serviceKind=1&currentPageNo=1>)를 사용하였다. 자치구, 법정동, 건물명, 임대건물명, 전월세구분, 보증금, 임대료, 계약년도 등의 정보가 포함되어 있으며, 이 중 자치구, 법정동, 임대면적, 건물유형, 전월세 구분 속성값들만 추출해 모델 훈련에 사용하였다.

범주형 독립 변수인 위치와 건물 유형 변수를 0과 1로 나타내는 더미변수로 변환해 회귀 모델에 사용할 수 있도록 전처리를 하고, 이 데이터들을 전/월세를 구분해 각각 LinearRegression 모델과 KNeighborsRegressor 모델로 훈련시킨다. 이 때 KNeighborsRegressor는 다중 종속 변수를 지원하지 않으므로 보증금과 월세에 대한 모델을 따로 훈련시킨다.

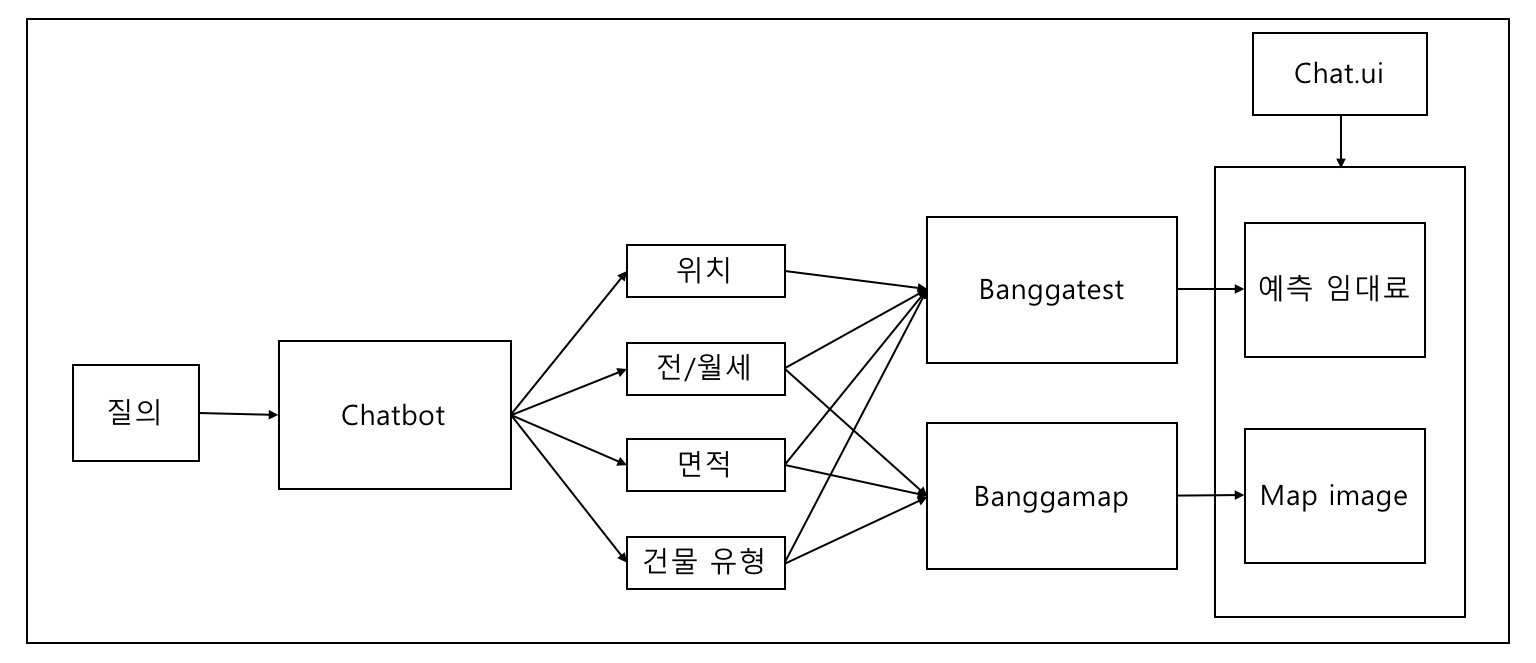
챗봇을 통해 입력 받은 위치, 임대 면적, 건물 유형 정보를 전세인 경우 전세 데이터로 훈련한 LinearRegression 모델에 입력 값으로 넣어 예측 전세 값을 결과값으로 얻고, 월세인 경우 월세 데이터로 훈련한 KNeighborsRegressor 모델에 입력 값으로 넣어준 후 보증금과 월세 값을 결과값으로 얻는다.

****

**[임대료 비교 지도 생성 과정]**

챗봇을 통해 입력 받은 정보 중, 전/월세, 임대 면적, 건물 유형에 해당하는 데이터만을 서울시 전/월세 데이터에서 추출해 seoul\_municipalities\_geo.json 이라는 서울시 행정구의 좌표가 있는 서울시 행정구역 데이터와 병합해 행정구에 따라 임대료가 어느 정도인지를 지도 이미지를 통해 보여준다.

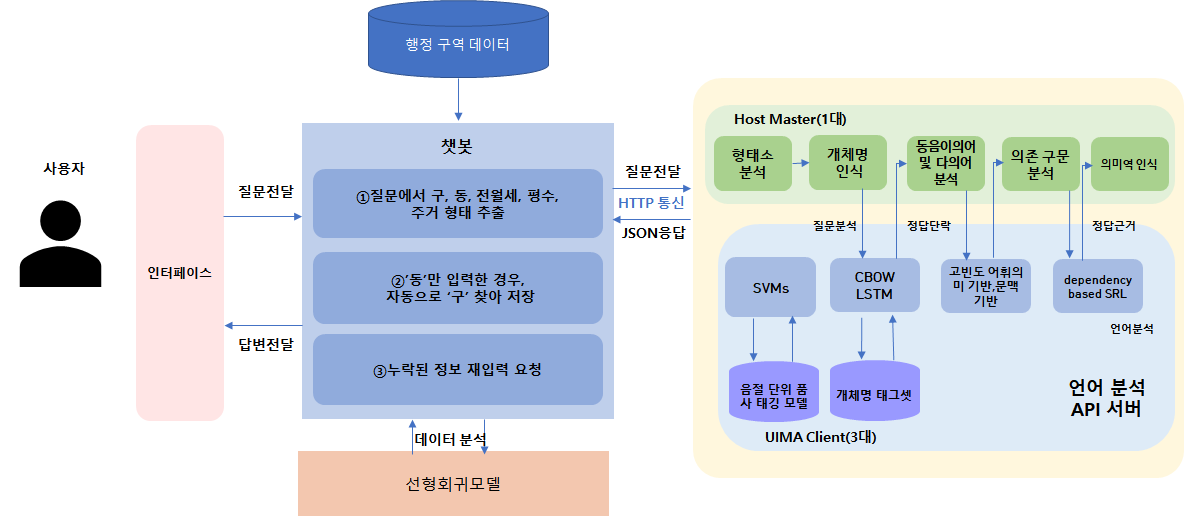
**3)**   **인터페이스 모듈**

****

**[인터페이스 작동 과정]**

qtDesigner를 통해 먼저 ui를 디자인한 후 .ui파일로 저장, 파이썬에서 PyQt의 loadUi를 통해 불러온다. inputTxt에 질의를 작성하고 send 버튼을 누르면 질의 내용을 챗봇 모델에 넣어 전/월세, 위치, 면적, 건물 유형의 딕셔너리를 얻는다. 그 딕셔너리를 임대료 예측 모델에 넣어 예측 값을 얻고, 딕셔너리 중 전/월세, 면적, 건물유형 값을 지도 생성 모델에 넣어 서울시 임대료 지도 이미지를 얻는다. 예측 값과 지도 이미지를 인터페이스에 출력한다.

**3-2. 동작 순서도**

****

1. 사용자가 프로그램을 실행해 질문을 입력한다. 질문 양식은 제한이 없으나 분석에 필요한 ‘자치구 명’, ’법정동 명’, ’전월세’, ‘거주 형태’를 입력해야 한다.

정확한 주소를 몰라 구 정보를 입력하지 않아도 시스템이 찾아 입력하지만 그 외 정보는 미 입력 시 추가 입력을 요청한다.

1. 챗봇은 사용자 질문을 서버로 전송한다. 서버가 질문을 분석해 형태소, 단어 별 태그로 라벨링한 결과를 다시 챗봇으로 전송한다. 챗봇은 이 결과를 분석 모델이 사용할 수 있는 딕셔너리 형태로 가공한다.
2. 분석 모델은 챗봇이 입력한 정보를 바탕으로 예측한 답변과 그래프를 인터페이스로 전달한다.
3. 사용자는 답변과 그래프를 확인한다.

**4.**  **시스템 세팅 및 사용자 가이드**

**4-1. 개발 환경**

시스템 실행 시 다음 베포판과 라이브러리가 필요하다

* python 3.5.6
* urllib3 1.23
* json 2.6
* pandas
* numpy
* scikit-learn
* geopandas
* matplotlib
* pyqt5

**4-2. 사용자 가이드**

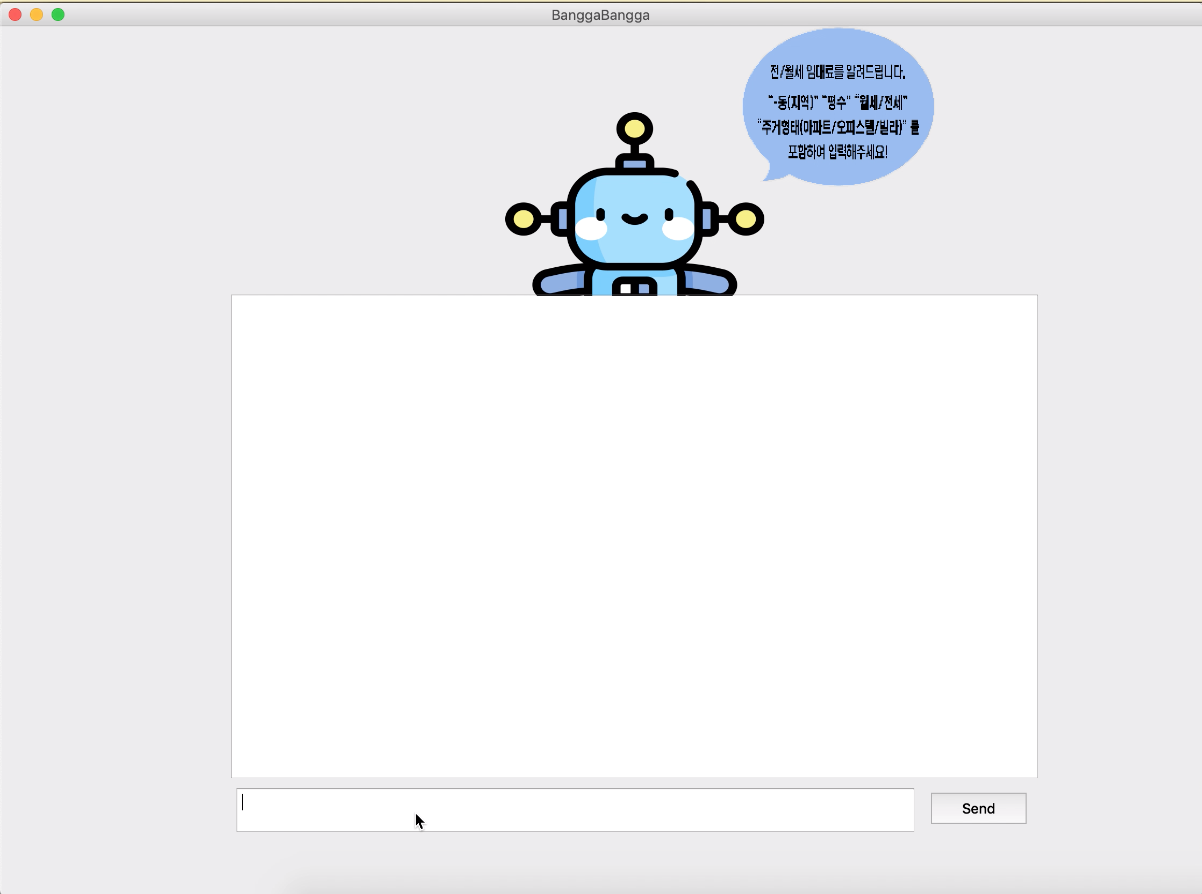
**1)**   **실행 준비**

아래의 파일을 모두 동일 폴더에 저장한다.

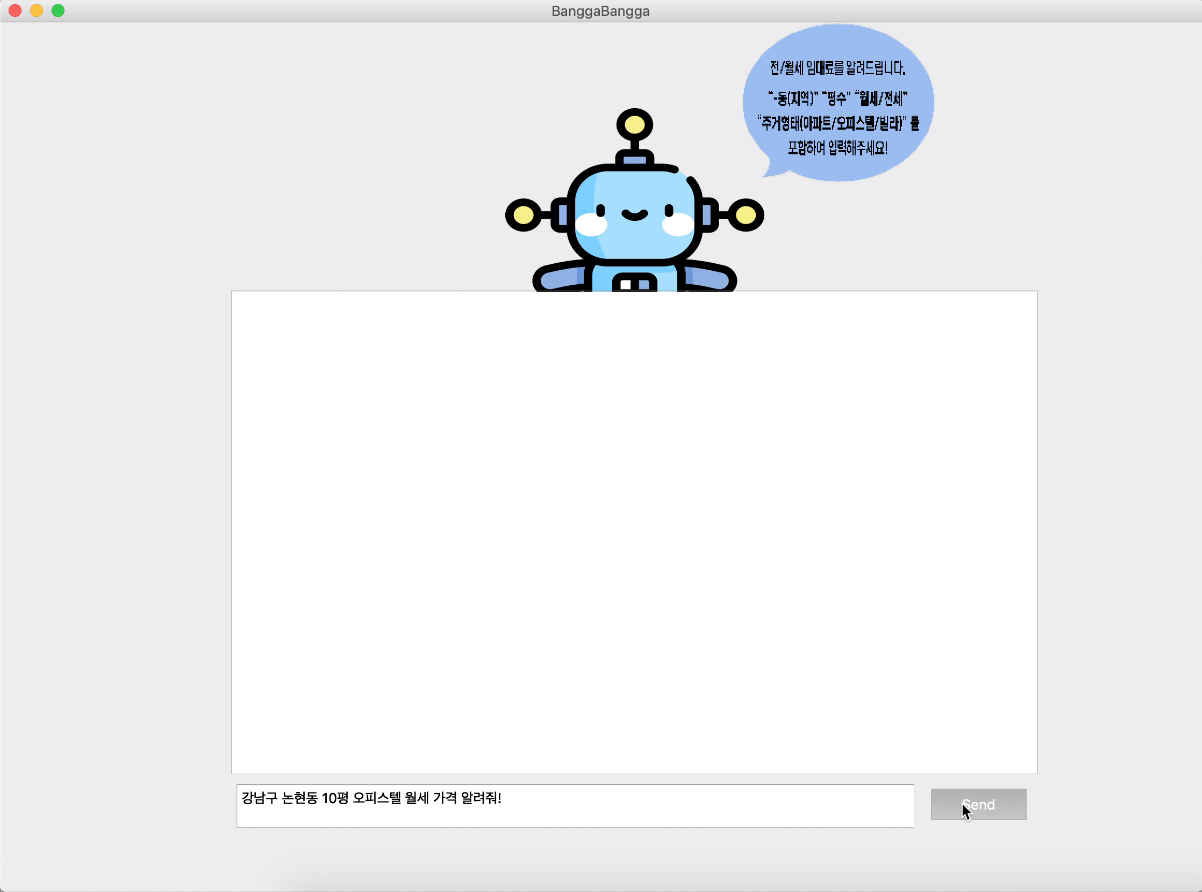
* + - chatbotOS.py
    - banggatest.py
    - banggamap.py
    - chat.py
    - seoul\_municipalities\_geo.json
    - realdata.csv
    - location.xslx
    - chat.ui

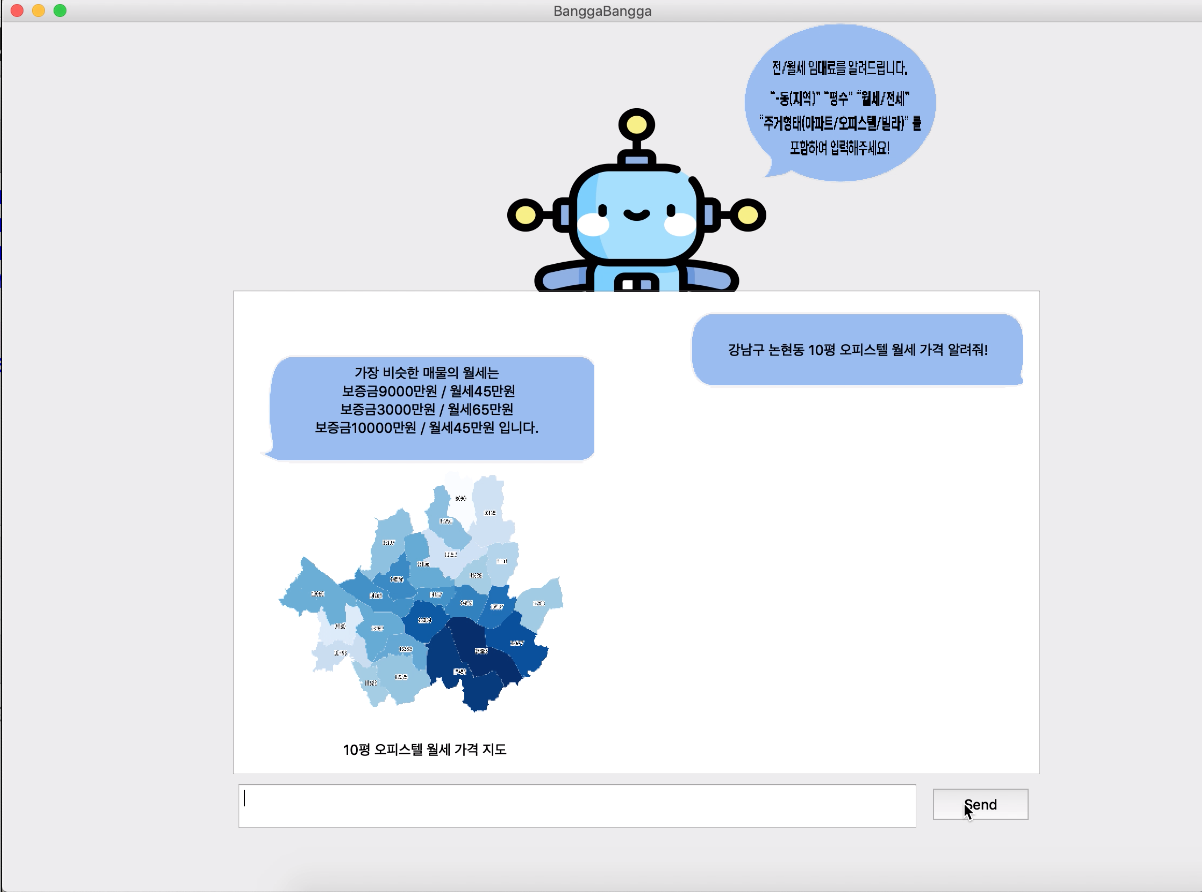
**2)**   **챗봇 실행**

해당 폴더로 이동, 챗봇을 실행한다. (python3 chat.py)

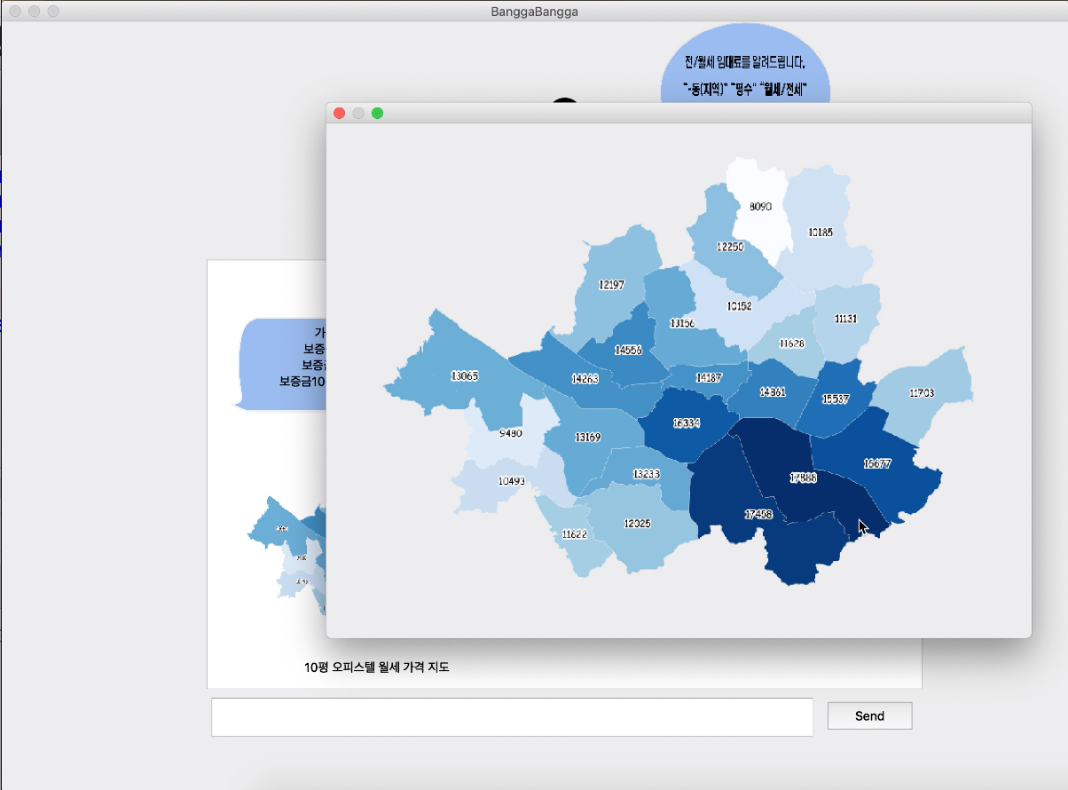


**3)**   **질문 입력창에 ‘원하는 지역, 주거형태, 평수, 전/월세 유형’를 자유로운 형식으로 입력, send버튼을 누른다.**

****

****

**4)**   **답변을 확인한 뒤, 그래프를 누르면 큰 창에서 확인할 수 있다.**

****

**5.**   **개발 후 소감**

1)   김설매

     이번에는 팀원과 협력도 매우 흥미로웠으며 오픈소스에 대한 많은 새로운 지식을 배웠다.생각보다 어려워지만 좋은 경험이었다. 인터페이스가 이론적으로 가장 많은 것을 알고 있다고 생각했는데 실제 구현 했을데 그러지 않다는 생각이들었다. 앞으로도 이에 대해서 더 많은 지식을 배울 것이다.

2)   김현수

사이킷런 사용 경험이 없어 어려웠지만 조원들과 협업하여 새로운 오픈소스에 대해 많은 것을 배울 수 있는 시간이었다. 또한 머신러닝에 대해 이론적으로 잘 알고 있다고 해서 머신러닝을 수행하는 모델을 코드로 직접 구현하는 것까지 잘 할 수 있는 것은 아니라는 생각이 들어 앞으로의 공부 방향을 정하는 데에 큰 도움이 되었다. 무엇보다도 사용자에게 직접 서비스를 제공하는 챗봇 개발에 참여하는 것은 처음이라 좋은 경험이 되었다.

3)   박현정

4)   이주경

5)    전효진

자연어 처리에 관한 전반적인 지식을 익힐 수 있었다. 프로그래밍 능력 외 데이터 전처리를 위해 우리말의 언어적 특징도 잘 알아야 했다. Api를 이용했으므로 서버에서 사용하는 여러 모델과 기술을 공부할 수 있었다. 특히 기존 모델들을 결합하거나 레이어를 추가해 정확도를 향상시키는 것을 보며 자연어 처리 기술의 확장성이 무궁무진함을 깨달았다. 이미지나 영상을 다루는 다른 프로젝트에 비해 결과물의 가시성이 부족한 점이 아쉬웠지만, 단순한 모델 사용이 아닌 일반 사용자가 이용 가능한 인터페이스 구축을 통해 자연어 처리 딥러닝 모델의 상용화 방법까지 고안한 것에 의미를 두고 싶다. 파이썬의 다양한 라이브러리, gui 툴을 공부할 수 있어 많은 것을 배울 수 있었던 프로젝트였다.

**6.**   **프로젝트 소스 코드**

1. **chat.py**

|  |
| --- |
| import sys  from PyQt5.QtCore import \*  from PyQt5.QtWidgets import \*  from PyQt5.uic import loadUi  from chatbotOS import \*  from banggamap import Map  from PyQt5.QtGui import \*  from PIL import Image  class Chat(QMainWindow):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.ui = loadUi('./chat.ui',self)          self.setupUi()          self.setupFunc()          self.returnstr=""          def setupUi(self):          self.setWindowTitle("BanggaBangga")          self.setStyleSheet("#Form {background-image: url(bgimg.png);}");        def setupFunc(self):          self.sendBtn.clicked.connect(self.showQuestion)          self.sendBtn.clicked.connect(self.analysis)          #self.question.clicked.connect(self.showGraph)          self.mapimg.mouseReleaseEvent = self.showmap      def showmap(self, event):          dialog = MyDialog()          dialog.show()          dialog.exec\_()          def showQuestion(self):            self.setStyleSheet("#question {background-image: url('q2.png');}");          q=self.inputTxt.toPlainText()          self.question.setText("\t"+q)        def showAnswer(self,money):          self.answer.setText(str(money)+"입니다. 클릭하면 도표를 보실 수 있습니다.")          self.inputTxt.clear()        def analysis(self):          text=self.inputTxt.toPlainText()          print(text)          qPixmapVar = QPixmap()          d=chk\_dict(chatbot(text))          if d["전월세"]=="전세":              result=jeonse(d["구"]+d['동'],d['평수'][:-1],d['거주 유형'])              Map("전세",int(d['평수'][:-1]),d['거주 유형'])              print(str(result))              print("결과",result)              #self.showAnswer(result)              self.returnstr=result.resultstr              self.setStyleSheet("#question {background-image: url('q2.png');} #answer {background-image: url('a2.png');}");                self.answer.setText(self.returnstr+'\n')              self.labelmap.setText(d['평수'][:-1]+"평 "+d['거주 유형']+" 전세 가격 지도 ")              image = Image.open('seoul\_map.png')                cropImage=image.crop((150,220,1180,880))              resize\_image = cropImage.resize((300,250))              resize\_image.save('seoul\_map2.png')              qPixmapVar.load('seoul\_map2.png')              self.mapimg.setPixmap(qPixmapVar)              self.inputTxt.clear()          else:              result=wolse(d["구"]+d['동'],d['평수'][:-1],d['거주 유형'])              Map("월세",int(d['평수'][:-1]),d['거주 유형'])              #self.showAnswer(result)              self.returnstr=result.resultstr              self.setStyleSheet("#question {background-image: url('q2.png');} #answer {background-image: url('a2.png');}");              self.answer.setText(self.returnstr+'\n')              self.labelmap.setText(d['평수'][:-1]+"평 "+d['거주 유형']+" 월세 가격 지도 ")              image = Image.open('seoul\_map.png')              cropImage=image.crop((150,220,1180,880))              resize\_image = cropImage.resize((300,250))              resize\_image.save('seoul\_map2.png')              qPixmapVar.load('seoul\_map2.png')              self.mapimg.setPixmap(qPixmapVar)              self.inputTxt.clear()  class MyDialog(QDialog):      def \_\_init\_\_(self):          QDialog.\_\_init\_\_(self)            lblmap = QLabel("img")            layout = QVBoxLayout()          layout.addWidget(lblmap)          qPixmapVar = QPixmap()          image = Image.open('seoul\_map.png')          cropImage=image.crop((150,220,1180,880))          resize\_image = cropImage.resize((700,500))          resize\_image.save('seoul\_map3.png')          qPixmapVar.load('seoul\_map3.png')          lblmap.setPixmap(qPixmapVar)            # 다이얼로그에 레이아웃 지정          self.setLayout(layout)  if \_\_name\_\_ =="\_\_main\_\_":      app = QApplication(sys.argv)      mywindow = Chat()      mywindow.show()      app.exec\_() |

1. **banggatest.py (임대료 예측)**

|  |
| --- |
| 전세 예측 |
| class jeonse:      def \_\_init\_\_(self,p1,p2,p3):          self.money=0          self.tlist=[]            df=pd.read\_csv('realdata.csv')          df['구동']=df['자치구명']+df['법정동명']          rent1=df[df['전월세구분']=='전세']            a=rent1[['임대면적']]          b=pd.get\_dummies(rent1['구동'])          self.simple\_rent1=a.join(b)          c=pd.get\_dummies(rent1['임대건물명'])          self.simple\_rent1=self.simple\_rent1.join(c)          self.X\_train, self.X\_test, self.Y\_train, self.Y\_test = train\_test\_split(self.simple\_rent1, rent1['보증금'],random\_state=42)          self.knn=KNeighborsRegressor(n\_neighbors=3)          self.lr=LinearRegression()            self.train()          self.getinput2(p1,p2,p3)          self.predict()        def train(self):          self.lr.fit(self.X\_train,self.Y\_train)          #print(self.lr.score(self.X\_test,self.Y\_test))        def getMoney(self):          return self.money        def predict(self):          for i in self.tlist:              pred=self.lr.predict(i)              pred=pred[0]              if pred>=10000:                  p=str(pred)                  b=p[0]                  m=pred-10000\*int(b)                  print(b+"억"+str(int(m))+"만원")                    self.resultstr=b+"억"+str(int(m))+"만원으로 예측됩니다."                  self.money=b+"억"+str(int(m))+"만원"              else:                  print(str(int(pred))+"만원")                  self.resultstr=str(int(pred))+"만원으로 예측됩니다."                  return str(str(int(pred))+"만원")              print('\n')        def getinput2(self,a,b,c):          self.tlist=[]          zero=np.zeros(len(self.simple\_rent1.columns))          t=pd.DataFrame([zero],columns=self.simple\_rent1.columns)          t['임대면적']=float(b)\*3.305785          t[a]=1.0          if c=='아파트':              t['아파트']=1.0          elif c=='오피스텔':              t['오피스텔']=1.0          else:              t['다세대/연립']=1.0          self.tlist.append(t) |
| 월세 예측 |
| class wolse:      def \_\_init\_\_(self,p1,p2,p3):          df=pd.read\_csv('realdata.csv')          df['구동']=df['자치구명']+df['법정동명']          rent1=df[df['전월세구분']=='준월세']          self.origin=rent1          a=rent1[['임대면적','건축년도']]          a=rent1[['임대면적']]          b=pd.get\_dummies(rent1['구동'])          self.simple\_rent1=a.join(b)          c=pd.get\_dummies(rent1['임대건물명'])          self.simple\_rent1=self.simple\_rent1.join(c)            floor=[]          for f in rent1['층'].values:                if f==-1:                  floor.append(-1)              else:                  floor.append(0)            self.simple\_rent1['floor']=floor          self.X\_train, self.X\_test, self.Y\_train, self.Y\_test = train\_test\_split(self.simple\_rent1, rent1['보증금'],random\_state=42)          self.X\_train2, self.X\_test2, self.Y\_train2, self.Y\_test2 = train\_test\_split(self.simple\_rent1, rent1['임대료'],random\_state=42)            self.knn=KNeighborsRegressor(n\_neighbors=3)          self.knn2=KNeighborsRegressor(n\_neighbors=3)          self.tlist=[]            self.train()            self.getinput2(p1,p2,p3)          self.predict()        def train(self):            self.knn.fit(self.X\_train, self.Y\_train)          self.knn2.fit(self.X\_train2, self.Y\_train2)          def getinput2(self,a,b,c):          self.tlist=[]          zero=np.zeros(len(self.simple\_rent1.columns))          t=pd.DataFrame([zero],columns=self.simple\_rent1.columns)          t['임대면적']=float(b)\*3.305785          t[a]=1.0          if c=='아파트':              t['아파트']=1.0          elif c=='오피스텔':              t['오피스텔']=1.0          else:              t['다세대/연립']=1.0          self.tlist.append(t)        def predict(self):          for i in self.tlist:              ind=self.knn.kneighbors(i,n\_neighbors=3,return\_distance=False)              print(ind[0])              ind2=self.knn2.kneighbors(i,n\_neighbors=3,return\_distance=False)              print(ind2[0])                indices=list(ind[0])              indices2=list(ind2[0])                y\_tr=pd.DataFrame(self.Y\_train)                zlist=list(map(lambda i: y\_tr.iloc[i,:], indices))                z=pd.DataFrame(zlist[1])              rz1=str(int(z.loc['보증금',:]))              y\_tr2=pd.DataFrame(self.Y\_train2)              z2list=list(map(lambda i: y\_tr2.iloc[i,:], indices2))              z2=pd.DataFrame(z2list[1])              rz2=str(int(z2.loc['임대료',:]))              r1="보증금"+rz1+"만원 / "+"월세"+rz2+"만원"                z=pd.DataFrame(zlist[2])              z2=pd.DataFrame(z2list[2])              rz1=str(int(z.loc['보증금',:]))              rz2=str(int(z2.loc['임대료',:]))              r2="보증금"+rz1+"만원 / "+"월세"+rz2+"만원"                z=pd.DataFrame(zlist[0])              z2=pd.DataFrame(z2list[0])              rz1=str(int(z.loc['보증금',:]))              rz2=str(int(z2.loc['임대료',:]))              r3="보증금"+rz1+"만원 / "+"월세"+rz2+"만원"                self.resultstr="가장 비슷한 매물의 월세는\n "+r1+'\n'+r2+'\n'+r3+" 입니다."              return str(self.resultstr)              print('\n') |

1. **banggamap.py (서울시 임대료 지도 생성 )**

|  |
| --- |
| import geopandas as gpd  import matplotlib.pyplot as plt  import pandas as pd  class Map:      def \_\_init\_\_(self,t,a,b):          plt.xkcd()          seoul=gpd.read\_file('seoul\_municipalities\_geo.json')            df=pd.read\_csv('realdata.csv')          t="전세"          if b!="아파트" and b!="오피스텔":              b="다세대/연립"          a=20          print(t)          df=df[['자치구명','전월세구분','임대면적','보증금','임대건물명']]          df=df[df['전월세구분']==t]            print(b)          df=df[df['임대건물명']==b]          print(a)          df=df[df['임대면적']>(a-5)]          df=df[df['임대면적']<(a+5)]            #size=df['임대면적']            sizedf=df[['자치구명','보증금']]            data=sizedf.groupby(['자치구명'], as\_index=False).mean()            data.columns=['SIG\_KOR\_NM','mean']            data\_result=pd.merge(seoul,data,on='SIG\_KOR\_NM')              final\_pic=data\_result.plot( figsize=(18,15),linewidth=0.25, edgecolor='white', column='mean',cmap='Blues')            for index,row in data\_result.iterrows():              xy=row['geometry'].centroid.coords[:]              xytext=row['geometry'].centroid.coords[:]              plt.annotate(int(row['mean']),xy=xy[0],xytext=xytext[0],horizontalalignment='center',verticalalignment='center')              plt.axis('off')          plt.show()              seoul\_fig=final\_pic.get\_figure()          seoul\_fig.savefig('seoul\_map.png',transparent=True) |

1. **chatbotOS.py**

|  |
| --- |
| import urllib3  import json  import pandas as pd  from banggatest import \*  #데이터 전처리  def adminArea():      data\_df=pd.read\_excel("location.xlsx",header=0)      guSet,dongSet=set(data\_df['자치구명']),set(data\_df['법정동명'])      gudong=data\_df[['자치구명','법정동명']]      guDongDict={k:g["법정동명"].tolist() for k,g in gudong.groupby("자치구명")}        return guDongDict,guSet,dongSet  #서버 통신  def chatbot(text):      openApiURL = "http://aiopen.etri.re.kr:8000/WiseNLU"      accessKey= "b0c887cb-30e0-432b-abfd-9b3055485fb4"      analysisCode = "ner"      qDict={}      adminDict,guSet,dongSet=adminArea()      print("질문:",text)      requestJson = {          "access\_key": accessKey,          "argument": {              "text": text,              "analysis\_code": analysisCode          }      }      http = urllib3.PoolManager()      response = http.request(          "POST",          openApiURL,          headers={"Content-Type": "application/json; charset=UTF-8"},          body=json.dumps(requestJson)      )      print("[responseCode] " + str(response.status))      #print("[responBody]")      print(str(response.data,"utf-8"))      result=str(response.data,"utf-8")      #JSON응답을 파이썬 딕셔너리 객체로 디코딩      a=json.loads(result)        ne=a["return\_object"]["sentence"][0]["NE"]      WSD=a["return\_object"]["sentence"][0]["WSD"]      address=[]      quantity=[]      #개체명 분석 결과에서 주소/평수 추출      for i in ne:          if i["type"] in ("LCP\_COUNTY","LCP\_CITY"):              print(i["text"])              address.append(i["text"])            if i["type"] in ("QT\_OTHERS","QT\_SIZE","QT\_LENGTH","QT\_COUNT","QT\_PRICE"):              qDict["평수"]=i["text"]      #어휘 분석 결과에서 전월세, 주거 형태 추출      for i in WSD:          if i["text"] in ("전세","월세"):              qDict["전월세"]=i["text"]          if i["text"] in ("아파트","오피스텔","빌라"):              qDict["거주 유형"]=i["text"]      #구와 동 분리 후 딕셔너리에 각각 저장      for i in address:          print(i)          if i in guSet:              qDict["구"]=i          if i in dongSet:              qDict["동"]=i       #구 미입력 시 자동 찾기      if "동" in qDict.keys() and "구" not in qDict.keys():          for gu, dong in adminDict.items():              for i in dong:                  if i == qDict["동"]:                      qDict["구"]=gu        return qDict  #누락 정보 재입력 요청  def chk\_dict(dic):      keys=["구","동","전월세","평수","거주 유형"]      count=len(keys)      while True:          for i in keys:              if i not in dic.keys():                  dic[i]=input("{}를 입력해주세요.".format(i))              count-=1          if len(dic.keys())==len(keys):              return dic |