1. 사용한 라이브러리와 알고리즘

사용한 라이브러리는 Deep Learning 라이브러리인 **Keras**를 사용하였고, 모델은 RNN(Recurrent Neural Network)을 확장한 **LSTM(Long Short-Term Memory Models)**를 사용함.

지하철 데이터는 2012년 ~ 2018년 데이터를 활용 하였고, 우리가 쓰고자 하는 데이터 셋이 존재하지 않아서, 날짜 별, 시간 별 역 승 하차 데이터를 모은 총합해 놓은 데이터(데이터 제공 Site : <https://data.seoul.go.kr/dataList/datasetView.do?infId=OA-12921&srvType=F&serviceKind=1&currentPageNo=1>) 토대로, **요일 별, 시간 별 데이터로 승 하차 데이터를 Excel로 따로 가공하여 CSV 파일로 저장**하였고, 이 데이터로 토대로 현재 요일과, 시간 별로 하여 데이터를 Import 시켜 LSTM 모델을 하여 학습함.

학습 횟수는 한 역당 10번으로 하였습니다. 처음에는 20번으로 Test 했는데 10번이나 20번이나 데이터 결과 값은 차이가 없는데, Delay 부분에서 약간 오래 걸리기도 하기도 하여서, 10번으로 학습 횟수를 정함.

BackEnd Server랑 소켓 통신을 하기 위해서는 SocketIO\_Client를 활용 하였으며, 사용자가 역 이름을 BackEnd Server로 보내면, BackEnd Server는 Deep Learning Server에 역 이름을 받아서 학습을 한 후, 학습 결과를 역 이름, 사용자 이름, 학습 결과 데이터를 딕셔너리로 저장을 한 후, Requests 라이브러리르 활용하여 딕셔너리 형식으로 BackEnd Server에 전송함.

2. 개발과정

|  |  |
| --- | --- |
| 주차 | 내용 |
| 1주차 | 조원들과 프로젝트 주제 선정 & 아이디어 도출  (어떤 기술을 사용 할 것인지, 개발 도구 선정, 데이터 수집 등) |
| 2주차 | 주제에 타당한 지하철 데이터 수집, 가공 |
| 3주차 | Keras LSTM 모델 예제 Test(주가 예측), 수집한 데이터 LSTM 모델 Test |
| 4주차 | BackEnd Server 소켓 통신 개발 |
| 5주차 | 최종 Test |

3. Code

|  |
| --- |
| import pandas as pd  import numpy as np  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import LSTM, Dropout, Dense, Activation  import time  from sklearn.preprocessing import MinMaxScaler  from datetime import datetime  from socketIO\_client\_nexus import SocketIO, LoggingNamespace  import requests  global name, state  result = 0  #칼럼들  now = datetime.now()  days = ['월요일','화요일','수요일','목요일','금요일','토요일','일요일']  time\_list = ['05~06','06~07','07~08','08~09','09~10','10~11','11~12','12~13','13~14','14~15','15~16','16~17','17~18','18~19','19~20','20~21','21~22','22~23','23~24','24~05']  train\_dict = {}  #프로그램 동작 경과시간 체크하기 위해 사용  start\_time = time.time()  #요일  days\_index = time.localtime().tm\_wday  days\_result = days[days\_index]  #월  month = now.month  #시간  times = now.hour  train = []  def time\_value(times):  if times >=5:  value = times - 5  else:  value = 19  return value  time\_index = int(time\_value(times))  time\_result = time\_list[time\_index]  def socket\_result():  # Listen  global state  socketIO = SocketIO('210.119.107.159', 9928, LoggingNamespace)  state = ""  print('시작!')  while True:  print("값을 받는 중...")  socketIO.emit('chat', "hello")  socketIO.on('train', on\_response)  socketIO.wait(2)  print("받은 값은 ", state, "입니다.")  if state != "":  params = training()  URL = 'http://210.119.107.159:9928/set\_page'  requests.post(URL, params).text  print("전송 성공")  state = ""  def on\_response(\*args):  global name, state  temp = list(args)  name,state = str(temp[0]).split(",")  #print(state)  def create\_dataset(signal\_data, look\_back=1):  x\_arr, y\_arr = [], []  for i in range(len(signal\_data) - look\_back):  x\_arr.append(signal\_data[i:(i + look\_back), 0])  y\_arr.append(signal\_data[i + look\_back, 0])  x\_arr = np.array(x\_arr)  x\_arr = np.reshape(x\_arr, (x\_arr.shape[0], x\_arr.shape[1], 1))  return x\_arr, np.array(y\_arr)  def training():  # 전철역 데이터 가지고 오기  with open("E:\\학교\\4학년\\1학기\\안드로이드\\Dataset\\train\_name.csv", "r") as f:  s = f.read() + '\n' # 데이터 읽을 시 문자로 읽게 설정 안하면 리스트 형식으로 읽어옴.... 이것때매 몇 시간 삽질한겨 -\_-  train = s.split('\n')  data = pd.read\_csv('E:\\학교\\4학년\\1학기\\안드로이드\\Dataset\\All\_Data.csv', encoding='CP949')  print("역 :", state)  print("월 : ", month)  print("요일 : ", days\_result)  data\_value = (data['역명'] == state) & (data['월'] == month) & (data['요일'] == days\_result)  data\_result = data[data\_value]  print(data\_result)  # 데이터 전처리  scaler = MinMaxScaler(feature\_range=(0, 1))  # 행렬 다시 설정  time\_value = str(time\_result)  temp = data\_result[time\_value].values.reshape(-1, 1)  trade\_count = scaler.fit\_transform(temp)  # 데이터 분리  # 훈련  train = trade\_count[0:int(len(trade\_count) \* 0.5)]  # 검증  val = trade\_count[int(len(trade\_count) \* 0.5): int(len(trade\_count) \* 0.75)]  # 시험  test = trade\_count[int(len(trade\_count) \* 0.75): -1]  x\_train, y\_train = create\_dataset(train, 1)  x\_val, y\_val = create\_dataset(val, 1)  x\_test, y\_test = create\_dataset(test, 1)  # #학습 모델 구성  batch\_size = 22  model = Sequential()  model.add(LSTM(50, return\_sequences=True, input\_shape=(1, 1)))  model.add(LSTM(64, return\_sequences=False))  # output  model.add(Dense(1, activation='linear'))  # 손실  model.compile(loss='mse', optimizer='rmsprop')  model.summary()  model.fit(x\_train, y\_train, validation\_data=(x\_val, y\_val), batch\_size=batch\_size, epochs=10)  predictions = model.predict(x\_test, batch\_size)  real\_predictions = scaler.inverse\_transform(predictions) # 0~1의 값으로 정규화된 값을 원래의 크기로 되돌린다.  print(state, "의", month, "월", days\_result, "의", time\_result, "시간의 전철 복잡도는 : ", real\_predictions[-1],  "입니다.") # 예측한 건수를 출력한다.  result = real\_predictions[-1]  params = {'user\_name': name, "train\_names": state, "train\_result": result}  return params  #소켓통신  socket\_result() |