Lowest-cost Gas Station Inform Service – NUGU OIL

Jang Seung Hwi
Department of Information Systems
College of Engineering,
Hanyang University
Seoul, Rep. of Korea
Email : seunghwi5545@gmail.com

Kim Ga Eun
Department of Information Systems
College of Engineering,
Hanyang University
Seoul, Rep. of Korea
Email : kimga-eun@naver.com

Choi Yun
Department of Information Systems
College of Engineering,
Hanyang University
Seoul, Rep. of Korea
Email : yunchoi22@gmail.com

Lee Gi Min
Department of Information Systems
College of Engineering,
Hanyang University
Seoul, Rep. of Korea
Email : syjmlove18@gmail.com

Abstract - How can we find the lowest-cost gas station near our current location? Dad's experience? It's someone opinion? Our team has developed a service that uses SKT NUGU speakers to provide an information of the lowest-cost gas stations based on your current location.

I. INTRODUCTION

The NUGU oil service is designed for people who want to cut oil prices even a little. Oil prices always change. In the midst of the change, it is good to go to the gas station you know, but we wanted to recommend the gas stations in order of the lowest diesel or gasoline price.

When the user asks the NUGU speaker about the lowest price gas station, the data is imported from the server and forwarded to the user. The following is the conversation flow between the user and the NUGU when using the NUGU oil service.

User: Ari, find me the lowest-priced gas station.
NUGU: Do you want diesel or gasoline? tell me the number.
User : Number 2
NUGU: I'll tell you three gas stations within a 5-kilometer radius. Hyundai Oilbank Co., Ltd. directly manages Topyeong Gas Station, 1,387 won, Blue Gas Station, 1,399 won, Hyundai Oilbank Co., Ltd. direct Topyeong Gas Station, 1,427 won.

Here are the items our team would like to serve:

- lowest gas stations recommendation based on your current location
- Provide information on the price of diesel or gasoline at each gas station
- Collect prices from 2007 gasoline prices to the present and provide forecast price information for next week

Research on any related software

1) SK Telecom T map

Launched in 2002, the T Map service is the top mobility platform in Korea, with approximately 12.5 million monthly active users (MAU). T Map provides the navigation service based on extensive real-time traffic information and precise pattern information. It provides a high level of convenience and safety based on the AI service, Nugu. Also, SK Telecom provides mobility services such as T Map Public Transportation, T Map Taxi, and T Map Parking based on the T Map API, making inroads into the global market through partnerships with companies such as Grab and global automobile makers.

2) OILNow

OilNow provides information on the nearest and cheapest gas station to the driver. In partnership with Korea National Oil Corporation, it collects information on gas stations nationwide in real time and calculates and provides personalized recommendation information through big data analysis algorithms. Based on the experience of using apps optimized for individuals, the company has achieved a cumulative 70,000 downloads in just one year since its release.

3) Opinet

Based on the data reported by the oil business operators, the company provides various oil price information such as the current selling price of gas stations and auto charging stations across the country, and the average supply price of oil refineries and dealerships, and Opinet started the smartphone app service in 2010 to make it easier to use the oil price information.

II. REQUIREMENT

1. AI Speaker

A. Ask and Answer user's request about gas station
   i. Set expected utterance that user's intent asking about nearest and cheapest gas station

ii. Process user's current location in backend proxy based on entity
iii. Get gas station name and each price. NUGU speaker will tell the result of information

B. Ask and Answer user's request about gasoline predicted price
i. Set expected utterance that user's intent asking about gasoline predicted price
ii. Get the predicted price from backend proxy. NUGU speaker will tell the result of price

III. DEVELOPMENT ENVIROMENT

1. *Which software devepment platform*

Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

2. *Which programming lanuage and why*

Python 3.8

Python 3.8.0 is the newest major release of the Python programming language, and it contains many new features and optimizations.

3. *Tools*

A. Oil Price Information API : The API provided by the Korea National Oil Corporation provides information on gas stations nationwide and average oil prices such as gas station sales price, gas station location, and additional services.

B. Google Geolocation API : The API returns the location and accuracy radius based on information about the user's base stations and WiFi nodes.

C. Facebook prophet : The Prophet algorithm is a time series prediction library created by Facebook that can be used as R and Python.

D. REST API : tands for Representative State Transfer, which means sending and receiving status based on the name of the resource.

4. *Task distribution*

Kim Ga Eun	Backend, NUGU Play Builder
Jang Seung Hwi	Backend
Choi Yun	Machine Learning
Lee Gi Min	Backend

IV. ARCHITECTURE DESIGN & IMPLEMENTATION

1. *Overall architecture*

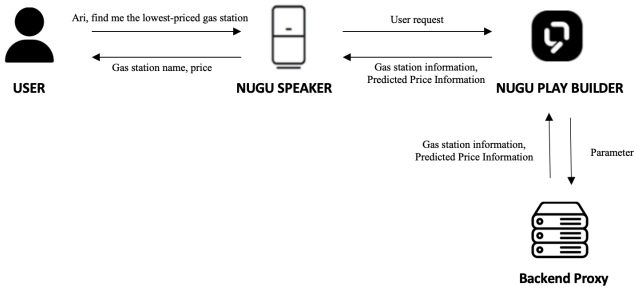


Fig 1

When the user asks the NUGU speaker about the lowest price gas station, the data is imported from the NUGU Play Builder via NUGU Speaker. NUGU Play Builder pass the information to Backend proxy about parameter. The following Fig 1 is showing connection between user and Backend proxy.

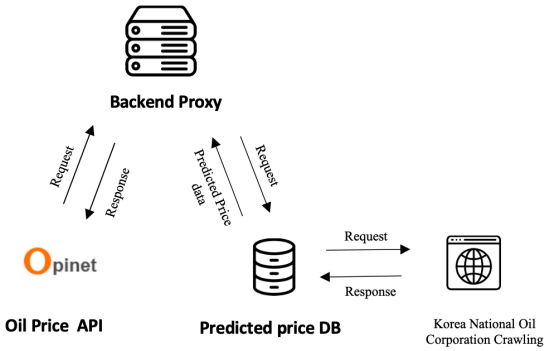


Fig 2

Backend proxy is connect with 2 part. One is Oil pcice API from Korea National Oil Corporation and other is predicted price DB. This DB is store our predicted price over two month.

2. *Directory organization*

i. Function part

location()	Find user's Current Location with the Geolocation API
trans(lat,lng)	Converting from WGS84 format to TM128 format for Opinet
ask_oil_type	Convert to "prodcd" code for Opinet
browse(x-poing, y-poing, oil-type)	arranged in the order of lowest prices
content()	provide up to 3 results
action(c)	Put result values in the list of dictionaries
make_response(result_list)	Returns to match answer format in NUGU PlayBuilder

i. location()

Enter Google Geolocation API url and api key for 'url' and 'data' will have code on what information to request location tracking. Because nuguil tracks user locations based on ip, 'considerIp': True' was added.

In 'result variable', information obtained from geolocation api is converted into json format and put into 'a variable'. The values that we need here are 'a['lat'] and 'a['location']['lng'] and they correspond to y-coordinates and x- These values are put into 'lat variable' and 'lng variable' and 'return lat,lng' respectively.

ii. trans(lat,lng)

We used the Pyproj package, which contains functions related to coordinate transformation. The lat,lng values of WGS84 format received as parameters were converted to TM128 format and returned in x_point and y_point variables.

iii. ask_oil_type

The function takes the information received as post request from the nugu speaker as a factor and converts it to the "prodcd" code you must enter when using the Opinet api.

iv. browse(x-poing, y-poing, oil-type)

This function takes x_point, y_point, and oil_type as factors using the Opinet api and puts them back into the request factor to get information from the gas stations arranged in the order of the cheapest gas stations within a radius of 5 km from the corresponding location.

type of gas service users want to search for the lowest-priced gas stations. This information is contained in a parameter named 'SELECT' or 'OIL_TYPE' and only one parameter is sent to the request body according to the action set by the nugu playbuilder. Check if there is a corresponding key value through the condition of the if statement and put the value in 'ans variable'. When adding oiltype to the browse function, the value of the ans variable must be converted to "No. 1" or "No. 2" in order to use the Ask_oil_type(ans) function, which converts the oiltype into the related code. Therefore, if the value of 'data['action']['parameters']['OIL_TYPE']['value'] is "volatile oil", then "no. 1" or "no. 2" is converted.

Next, the location information obtained by the location() function is initialized in 'a,b variable', and the values of a,b of WGS84 type are converted to TM128 format as a trans (a,b) function.

After initializing the list of gas stations that were obtained as a browse (a,b,ask_oil_type(ans) function into 'oil_list', convert the value of 'oil_list' into an action function.

After processing the information according to the response body format with the make_response(result) function, add the value entered by select or the value entered by oiltype to the response body.

iii. NUGU Paly Developer part

A. Create user ultterance Model

Intent refers to a user's intention, and the function of handling this intention is action. For example, to ask about the weather, "Ariya, let me know the weather" is Intent, and the answer is

v. content()

The function counts how many gas stations are in the global type oil_list, which contains the list of gas stations as the return value of the brows function, and puts the names and prices of up to three gas stations in the global type data_num variable, respectively.

vi. action(c)

The action(c) function is a function that returns a value of c in a dictionary whose key value is keys=['number','title','cost'] by taking the factor C, which is the return of the content function.

vii. make_response(result_list)

The make_response(result_list) function is a function that receives a list of dictionaries from the return value of the action function, receives a post request from the nugu speaker, and returns the dictionaries tailored to the type of response block data required by the nugu speaker. The return["output"]["COUNT"] contains the total number of gas stations parsed, and the return["output"]["STATION_INFORMATION"] contains the information of the gas stations parsed. Take the form of a string containing information from up to three gas stations, such as~~oil stations,~won,~oil stations,~won"

ii. Flask part

First of all, 'data = request.The request body, which has been received through get_json()', contains information on what

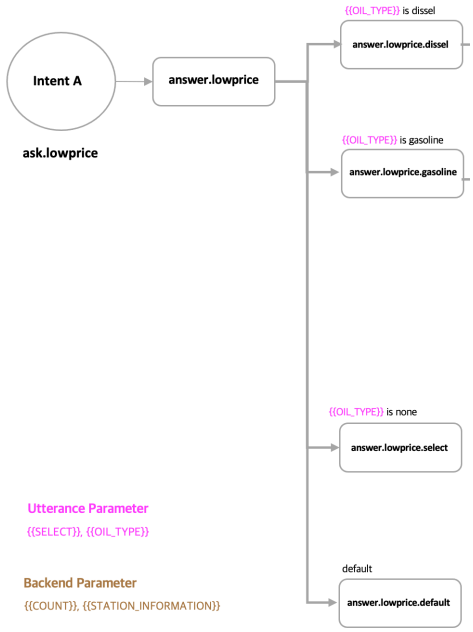
"It's sunny today" is Action. The Intent defined in NUGU oil is as follows:

ask.lowprice	This intent receives the type of oil (hard/volatile oil) entity, which is a user ignited, and sends it to the Backend Proxy server
ask.price	This intent receives forecast prices of diesel and gasoline from Backend Proxy server.
ask.select	Intent used when asking again to find out if the user is asking if it is diesel or gasoline

B. Create Action

Action means to process Intent when it is actually analyzed through the NLU engine. As we saw in 'Let me know the weather', we can create one action to handle Intent, and we can write a response to this action.

NUGU Play Builder's speech processing is as follows. If the user mentions whether it is diesel or gasoline, the name and price of the gas station brought directly from Backend Proxy are linked, and if the user did not mention it in advance, it is written in a structure that asks once more whether it is diesel or gasoline.



Fif 3

i. Prediction part

A. Prediction of Oil Prices through Time Series Analysis

In the case of the oil price information API, it only provides diesel and gasoline prices for a week, so it set a period on the website of the Korea National Oil Corporation and downloaded price information. Our goal is to forecast oil prices for the next four weeks. To achieve this goal, we will use the FBprofile library on Facebook to predict time series data.

The Profile algorithm is a time series prediction library created by Facebook that can be used as R and Python. There are various time series models such as Arima, but we chose Prophet not only because the usage of the propet is simple but also has high accuracy and can handle irregular cycles. However, since Prophet does not disclose internal algorithms, it is not known how they actually work within the code.

```
In [5]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from fbprophet import Prophet
```

```
In [7]: df = pd.read_csv("/Users/choiyun/Downloads/sampler.csv",engine='python',encoding='CP949')
        df.tail()
```

Out[7]:

	ds	y
235	2020-05-01	1585.8
236	2020-05-02	1577.0
237	2020-05-03	1574.9
238	2020-05-04	1580.6
239	2020-06-01	1588.5

The oil price information received through Opinet is retrieved through the read_csv function. After the creation of an instance through a propet call, the fit function passes over the data frame of the csv file from the Opinet. Invoke make_future_dataframe, a function that creates a data frame to hold predicted data. This created a dataframe to predict the data for the next four weeks, and saved the variable name as future. The predict method assigns a predictive value named yhat to each row in the future. We

```
In [8]: model=Prophet()
        model.fit(df)
        future=model.make_future_dataframe(periods=4,freq='w')
        forecast=model.predict(future)
```

```
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to ov
```

```
In [9]: result=forecast[['ds','yhat']]
        result.to_csv("/Users/choiyun/Downloads/result.csv", index=False)
        result.tail()
```

Out[9]:

	ds	yhat
239	2020-06-01	1717.390143
240	2020-06-07	1737.888957
241	2020-06-14	1754.893891
242	2020-06-21	1754.476336
243	2020-06-28	1733.447534

will store only the actual predicted yhat value in 'result' among the values stored in the forecast.
Finally, using the to_csv function of the pandas library, we saved the forecast results in the form of a csv file.

B. Measurement of performance through cross-validation

Cross-validation is required to prevent overfitting of predictive algorithms and reduce model variability. In general, for cross-validation, follow the following sequence.

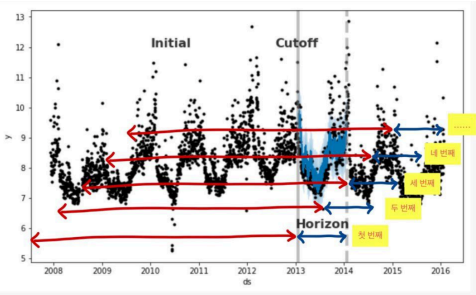
1) Divide the data into trainings and test sets.
2) Test the model and record the test performance.
3) Perform the above tasks with different partitions at each stage of cross-validation.
4) Final performance is derived by averaging the test performance at each stage.

The fbprofile library provides a function ('cross_validation') that automatically executes this cross-validation process for the convenience of users.

To this end, we are done by selecting the cut-off point from the past data (the data we have given the computer), and using the data only for each cut-off point to learn the model. The predicted values are then compared with the actual values.

This cross-validation procedure can be performed automatically for various historical cutoff points using the 'cross_validation' function. The resultant value of the 'cross_validation' function is a data frame that has the actual value y and the non-sampling forecast yhat for the simulated predicted date and cutoff date respectively.

Below is an example of a graph to help explain.



←——> : training data(initial) ←——> : test data(horizon)
cutoff point는 initial 과 horizon을 구분해 주는 날짜

In particular, predictions are made at all points observed between the cut-off and the cut-off + horizon (point painted blue).Now this dataframe is used to calculate the error between yhat and y.

This is a process of cross-verification using 'cross_validation' function that is based on what was previously explained. (*However, when setting 'initial'), it should be long enough to include both seasonal and periodicity.*) We gave 60 weeks of 'initial' and 20 weeks of predicted future.

In [10]:
```
from fbprophet.diagnostics import cross_validation
df_cv = cross_validation(model, initial='60 w', period='10 w', horizon = '20 w')
df_cv[['ds','yhat','y','cutoff']].head()
```

```
INFO:fbprophet:Making 16 forecasts with cutoffs between 2017-02-27 00:00:00 and 2
HBox(children=(FloatProgress(value=0.0, max=16.0), HTML(value='')))
```

Out[10]:

	ds	yhat	y	cutoff
0	2017-03-01	1878.603309	1842.9	2017-02-27
1	2017-03-02	1880.034963	1841.5	2017-02-27
2	2017-03-03	1887.795075	1837.1	2017-02-27
3	2017-03-04	1894.831045	1830.6	2017-02-27
4	2017-03-05	1900.593434	1822.4	2017-02-27

The performance_metrics package can be used to calculate useful statistics (mse, rmse, mae, map, mdape, coverage) for performance forecasting.

In [11]:
```
from fbprophet.diagnostics import performance_metrics
df_p = performance_metrics(df_cv)
df_p.head()
```

Out[11]:

	horizon	mse	rmse	mae	mape	mdape	coverage
0	15 days	3285.566658	57.319863	40.942065	0.022940	0.015428	0.354839
1	16 days	3364.747430	58.006443	42.088405	0.023597	0.015428	0.354839
2	17 days	3373.555199	58.082314	41.102252	0.023051	0.014008	0.419355
3	18 days	3257.199395	57.071879	38.765277	0.021682	0.012563	0.483871
4	19 days	3241.443325	56.933675	37.936349	0.021175	0.007799	0.467742

'Plot_cross_validation_metric' can visualize various useful statistics (MSE, RMSE, MAE, MAPE, and MAPE) that were previously derived through 'performance_metrics'.

The following graph shows the mean square root error (RMSE) and the mean absolute ratio error (MAPE).

In [12]:
```
from fbprophet.plot import plot_cross_validation_metric
fig = plot_cross_validation_metric(df_cv, metric='mape')
```