ELEC088 23/24 SNS REPORT

SN: 23096834
Programme of study: Telecoms
SN: 23038138
Programme of study: Telecoms
SN: 23143118
Programme of study: Telecoms
SN: 23157164
Programme of study: Internet Engineering
SN: 23218481
Programme of study: Internet Engineering
SN: 23226874
Programme of study: Telecoms

ABSTRACT

This project creates the implementation of an Oracle chatbot for predicting the weather (4 different features included) and the number of bicycle hires in London for the next seven days. First, data is collected on the internet and preprocessed before machine learning is started. In the merged dataset, five features are selected to be predicted, including maximum temperature, minimum temperature, wind speed, precipitation and the number of bicycle hires. In addition, prediction is done by using time series forecasting model (Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM)) with machine learning. Mean Absolute Error (MAE) is used to measure the prediction error of a prediction model over a set of observations. Finally, Transmission Control Protocol (TCP) is used to achieve the operation of the Oracle chatbot which means the prediction model was accessible from a server, the client could request information from the server and get a prediction response back. All codes have been implemented in Python programming language and a GitHub URL is included in this report.[1]

***Index Terms*—** Weather Forecast, Bicycle Hires Numbers Forecast, GRU, LSTM, TCP, Chatbot

1. INTRODUCTION

Machine learning as a subfield of Artificial Intelligence gives computers the ability to learn without being explicitly programmed. It is behind many sophisticated services in today's world such as chatbots, predictive texts and suggestions to users when online. In line with this, prediction of the weather is the most common application in daily life, as it can provide information about the future weather and reminds people to bring or not to bring an umbrella or how much thickness of clothes to wear when going out. Meanwhile, the prediction of the number of bicycle hires is done based on weather conditions. While recognizing that bicycle hires can be influenced by some other factors such as random events in the city, weather has a constant relationship and therefore was the factor considered.

We have divided this project into three sections. The first section is data collection and pre-processing. A large number of datasets about weather and bicycle hire numbers were found on the website and saved, to select the complete and reliable dataset sources. At the same time, data types, missing values in each column, feature selection and other aspects need to be pre-processed likewise before moving on to the machine learning part.

In the second section, which is the machine learning part, the time series forecasting model (GRU and LSTM) is used to train the model for predicting weather and the number of bike hires. Both models are based on neural networks and have a strong ability to process sequential data, capturing long-term dependencies in the data. GRU and LSTM are variants of Recurrent Neural Networks (RNNs), both of which were created to solve the problem of gradient vanishing and gradient explosion encountered by traditional RNNs when dealing with long sequential data. Compared with GRU, LSTM has a more complex model structure with more parameters and provides finer-grained control of in-

[1] The code is provided https://github.com/Flowey0622/ELEC0088-SNS-assignment and GitHub project: ELEC0088-SNS-assignment.

formation flow, which also means that LSTM has a higher computational cost than GRU.

The final stage is the integration part, where the Oracle will run on the server side and contact the client user through the terminal. The socket interface was used to provide network functionality to send and receive data in the network. TCP socket that provides a reliable and connection-oriented service was chosen. The main task performed was to make the developed model available through a server. The client could request information about the weather and the number of bicycle hires from the server and get a response back. At the same time, the group decided to use Google Cloud Platform, a cloud program, as the server. Ease of use and the existing Dialogflow structure are practical in this choice. The goal of utilizing Dialogflow with Google Cloud Platform is to provide a dynamic service that can respond to users' natural language requests quickly and accurately.

2. DESCRIPTION OF DATASETS

2.1 DATA COLLECTION

Relevant data in line with the project objective was searched on the Internet. Much effort was employed in searching for and collecting data to secure good quality data because incorrect or outdated data can lead to wrong outcomes and irrelevant predictions. Among the goals was also to have data large and diverse enough to cover sufficiently well the characteristics of the underlying task, so that the model can learn the task well. Data consistency was also key in the selection of datasets, there was much interest in having time series data.

Finally, the first dataset selected was collected from the website called Visual Crossing, where weather conditions for different areas in different periods can be downloaded. The project was set up in the context of London, so a CSV file of weather conditions in London was downloaded and saved from January 1st 2000 to January 31st 2024 which contained 30 different dimensions or features. The second dataset was downloaded from the official website of Transport for London (TLS), which only contains information about bicycle rental from July 30th 2010 to January 31st 2024 in London.

Two datasets have been displayed in the datasets folder on GitHub: London 2000-01-01 to 2024-01-31.csv and tfl-daily-cycle-hires.csv.

2.2 DATA PREPARATION

At this stage, data was cleaned and pre-processed to make it ready for training. In such a process, several checks were conducted and subsequent improvements were made where necessary. There are some main activities in the data preparation part.

First, it is worthwhile to determine which feature needs to be predicted. Therefore, the percentage of missing data in each parameter was checked. This was to identify the extent of null values in the dataset. The goal is not to pass entries with null values to training. Furthermore, the columns with a high percentage of null values were preferred to be removed from the dataset. After discussion, five features containing maximum temperature, minimum temperature, windspeed, precipitation and the number of bicycle hires want to be predicted by machine learning.

Secondly, two datasets are loaded into this program and the weather conditions from July 30th 2010 to January 31st 2024 should be filtered to fit the size of the dataset downloaded from TFL as the two datasets have different time ranges. One thing to note here is that the Date parameter has been converted from an object type to a datetime type as the index of the dataset.

Next, since the dataset about weather conditions has some columns with poor correlation or columns with only textual descriptions and no numerical values, these are preferred to be removed as this facilitates machine learning later on. After these processes, the two datasets were merged into a single dataset.

Then, the data type needs to be checked where the number of bicycle hires should be converted from an object type to a float type. Furthermore, for the remaining columns, missing data were filled up. Previous day values were used in filling up because there is a high correlation of weather conditions between consecutive days. Meanwhile, the entries for each year will be checked in case the data is not consecutive days.

Finally, the data for the five features were analyzed and their graphs were plotted as shown in Fig. 1.

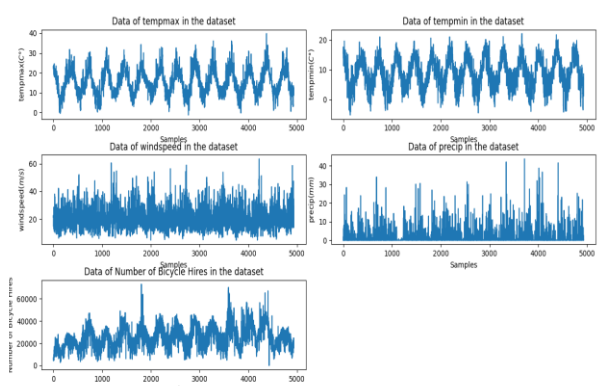

Fig. 1. Plot of five features values

At this point, the collected data was ready to be passed for the next step.

3. METHODOLOGY

3.1 NEURAL NETWORK ARCHITECTURES

As stated in the introduction section, GRU and LSTM models were used to predict the weather and the number of bike hires in this project. However, before training the models by using these methods, each input time-series data needs to have a comparable scale by normalization. Following that, the ratio of the training set (70%), validation set (15%), and test set (15%) were specified according to the input data dimension, the training model architecture, and the case of the application [1]. Based on that, the size of the training set, validation set, and test set were computed accordingly. To enhance the accuracy of prediction, the correlations between different features i.e., maximum temperature, wind speed, precipitation, etc., and the target feature were extracted to generate the correlation matrix, then, only the features that have relatively high correlations with the target feature were retained to complete the following training and prediction processes. In order to train the model more conveniently and efficiently in the later stage, the data in the training set, validation set, and test set were all organized into pairs, and each pair contains the data for the past 30 days and one day in the next 7 days, which means that the past 30 days data will be used by the model to forecast the features for any day in the next 7 days and the prediction result will be tested by the latter in the preceding sentence, respectively. After that, the GRU model was built, trained, and tested which is shown in Fig. 2.

```
# ##=========== build and train the model ===========##
# Train and evaluate a stacked GRU model using dropout regularization
model = Sequential()
model.add(layers.GRU(32,
                     dropout=0.1,
                     recurrent_dropout=0.5,
                     return_sequences=True,
                     input_shape=(None, filtered_data.shape[-1])))
model.add(layers.GRU(16,
                     activation='relu',
                     dropout=0.1,
                     recurrent_dropout=0.5))
model.add(layers.Dense(1))
model.summary()

# compile and train the model
model.compile(optimizer=RMSprop(learning_rate=0.001),
              loss='mae')
history = model.fit(x_train, y_train,
                    epochs=25,
                    batch_size=128,
                    validation_data=(x_val, y_val))

# save the model
filename = f'GRU_Model_Day{date+1}_{feature}.h5'
model.save('../Models_Trained/' + filename)

# save the training history
train_history.at[date, feature] = history

# ##================ test the model ================##
score = model.evaluate(x_test, y_test, verbose=0)
test_mae.at[date, feature] = score*deltas[feature]
```

Fig. 2. Code snippets for building, training, and testing the GRU model

In this project, the model is a regressor, hence, only one neuron is in the output layer [2], and due to the fact that no layers will be shared in this model, the sequential Application Programme Interface (API) was chosen [3]. According to the network capacity requirement of this project, one input layer, one hidden layer, and one output layer were built to produce the GRU model [4]. The unit number in each layer was determined by the data complexity and the trade-off between model performance and the presence of the overfitting issue. To overcome the overfitting problem in this project, dropout regularization is used which can disrupt accidental relationships within the training set by zeroing out the units of input randomly [5]. Additionally, in Keras which is a simple and flexible API for deep learning, two arguments about dropout are required in the recurrent layer, one provides the dropout rate of the layer input units (dropout) and another one provides the dropout rate for the recurrent units (recurrent_dropout) [5]. The dropout rate in the input and hidden layer were chosen based on the trade-off between avoiding the model under-learning issues and preventing the minimal effect of the model [6]. To improve the capability of the recurrent network, the recurrent layer stacking technique was used (return_sequence = True) in the input layer, which means that, instead of returning their output from the final timestep, the entire output sequences need to be returned by all the intermediate layers [4]. After specifying the input shape in the input layer, the activation function needs to be added to the hidden layer to resolve challenging issues and transfer, analyze, and evaluate data by using deep learning algorithms [7]. The rectified linear units (relu) function was selected as the activation function in this project due to its high computational and convergence speed [8]. Since this project belongs to the regression problem, no activation function was added to the last Dense layer of the model [9].

After the structure of the neural network is defined, the model needs to be compiled. The appropriate optimizer was selected based on the project target, the dataset properties, and the training structure [10], to minimize the loss function of the model which will be explained later. The Root Mean Squared Propagation (RMSprop) algorithm was chosen as the optimizer in this project since it provides a stable learning process and fewer hyperparameters to tune which makes the algorithm more practical to use [11]. The learning rate within the optimizer is an extremely significant hyperparameter for the process of neural network training, which was decided based on the trade-off between the model reliability and the optimization speed of the model [12]. Apart from that, the loss function requires to be added in the model compiling process to show the performance of the neural network model [13]. The MAE function was selected as the loss function in this project since the outliers may exist in the dataset and the MAE loss function has a good capability to handle this problem due to its basic algorithm [14]. Then, the model was trained by using the training set, and the number of epochs was decided based on the model complexity and the dataset size [15]; the number of batch sizes was determined according to the trade-off between the prediction accuracy of the model and the model training time [16]. Finally, the performance of the prediction was obtained by testing the model after saving the model for providing the prediction result in the later stage.

In addition, the LSTM model was built, trained, and tested likewise which is shown in Fig. 3.

```
# ##=========== build and train the model ===========##
# Train and evaluate a stacked LSTM model using dropout regularization
model = Sequential()
model.add(layers.LSTM(32,
                      dropout=0.1,
                      recurrent_dropout=0.5,
                      return_sequences=True,
                      input_shape=(None, filtered_data.shape[-1])))
model.add(layers.LSTM(16,
                      activation='relu',
                      dropout=0.1,
                      recurrent_dropout=0.5))
model.add(layers.Dense(1))
model.summary()

# compile and train the model
model.compile(optimizer=RMSprop(learning_rate=0.001),
              loss='mae')
history = model.fit(x_train, y_train,
                    epochs=25,
                    batch_size=128,
                    validation_data=(x_val, y_val))

# save the model
filename = f'LSTM_Model_Day{date+1}_{feature}.h5'
model.save('../Models_Trained/' + filename)

# save the training history
train_history.at[date, feature] = history

# ##================ test the model ================##
score = model.evaluate(x_test, y_test, verbose=0)
test_mae.at[date, feature] = score*deltas[feature]
```

Fig. 3. Code snippets for building, training, and testing the LSTM model

Compared with the completion process of the GRU model, the same datasets, procedures, and values of all the hyper-parameters were applied to the LSTM model except for changing the GRU layer into the LSTM layer.

3.2 BRIDGE BETWEEN THE TRAINED MODELS AND THE CLIENT / SERVER

After loading the trained model saved in the previous step, another program is written which integrates a class named MyModel that looks up the data for the first 30 days of the day of the client's query, predicts the features for the next 7 days of the day, and figures out the corresponding trained models based on the client questions in the loaded models to provide the prediction results. That means it can be activated based on the reference date provided by the client, thus ensuring personalized and contextually relevant predictions.

Specifically, it is divided into two parts, the first one is model initialization. When a new client connects, the server asks for the current date to use as a reference point for predictions. This date initializes the MyModel instance, customizing its prediction to fit the specific situation of the client. The other one is model Prediction. The server queries the ML model for temperature, windspeed, and precipitation forecasts for the weather conditions prediction. While for bike hire predictions, it requests the estimated number of bike hires. These queries are made for the specified day, according to the client's requests.

3.3 CLIENT-SERVER ARCHITECTURE

In the client-server model, the server remains active for incoming connections from clients. Once a client establishes a connection, the server engages in a conversational interface processing requests and delivering predictions based on the inputs received.

The Oracle bot employs TCP for its server architecture, primarily due to TCP's reliability and its capability to transmit data in a sequential order. This ensures that messages exchanged between the client and server are delivered accurately and in the correct sequence, which is crucial for maintaining the conversational flow and integrity of the predictive assistance provided by Oracle.

The server is built utilizing Python's socket and threading libraries, with the OracleBotThread Class managing each client connection in distinct threads for parallel communications with numerous clients, allowing for scalable and concurrent client handling. Various Python functions encapsulate the server's capabilities including client connection establishment, handling the conversation, machine learning integration, and prediction tasks. While The client interface also built using Python's socket library, allows users to connect to the server to ask about the weather and the number of bike hires, and receive responses.

The server processes client sentences using a combination of keyword identification and regular expressions, enabling it to understand and respond to client requests. For the former, it identifies keywords related to the type of prediction (weather or bike hires). The server scans client messages for keywords related to the requested prediction type, such as ”weather” or ”bike hires.” This method allows the server to understand what the client is asking for. Furthermore, it also extracts the day for which the prediction is requested using regular expressions. The server uses regular expressions to identify phrases like ”day X” where X represents a number, in order to pinpoint the exact day a prediction is sought for. This method offers adaptability in processing requests.

3.4 GOOGLE CLOUD

A server is needed to provide external access to the project. Google Cloud was used due to its easy-to-use server structure and a suitable infrastructure to create a dynamic Chatbot system.

The Dialogflow structure in Google Cloud can receive messages from users, process them as necessary, and present the processed results to users. Because it uses NLP, it becomes easier to perceive messages coming from the user.

In the created server-client structure, the user asks his question via Chatbot. The questions asked are received and processed by the predetermined Dialogflow structure. The processed data generates an answer and presents it to the user. All of this happens on the server side. The Dialogflow structure is expected to add flexibility to the user's questions and the answers to be produced.

Google App Engine and Flask features were also used when creating Dialogflow on Google Cloud. The primary purpose is to connect to the server by creating an API and getting the processed result. Flask is a web application framework often used to create web APIs. App Engine also offers secure HTTPS connections.

The web framework of choice for creating REST APIs in our project is Flask. It is an application that facilitates communication between Python programmes and web servers. This situation guarantees that the project can process HTTP requests and is available online. As a result, a Flask application called 'app.py' was launched and configured to process Dialogflow queries. Its contents are related to the server-functioning Python code. It facilitates the production of dynamic responses. When the user queries the weather, the Flask application can run a function that performs the weather forecast and returns this information to the user in response.

Dialogflow has structures such as intents/entities/responses. These are predetermined contents to produce content according to the user's speech and to reach the necessary predictions. The project's scope was prepared considering the date, weather, and bicycle hire. Fig. 4 shows a part of app.py is related to Dialogflow intents. The system: With Dialogflow integration, we want to understand the queries made by users in natural language and produce appropriate answers to these queries by associating them with pre-prepared concepts.

```
if intent_name == 'WeatherIntent':
    date = req.get('queryResult').get('parameters').get('date')

    weather_data = oracle_bot_thread.predict_weather(date)

    fulfillment_text = (f"For {date}, the forecast is: "
                        f"high of {weather_data['high_temp']:.1f}°C, "
                        f"low of {weather_data['low_temp']:.1f}°C, "
                        f"wind speed at {weather_data['wind_speed']:.1f} km/h,

                        f"and precipitation levels around
```

Fig. 4. The part of app.py

In addition, the Google Cloud Build structure was used to access the codes stored in GitHub. Changes are detected using the trigger here, and the system is constantly updated. As shown in Fig. 5.

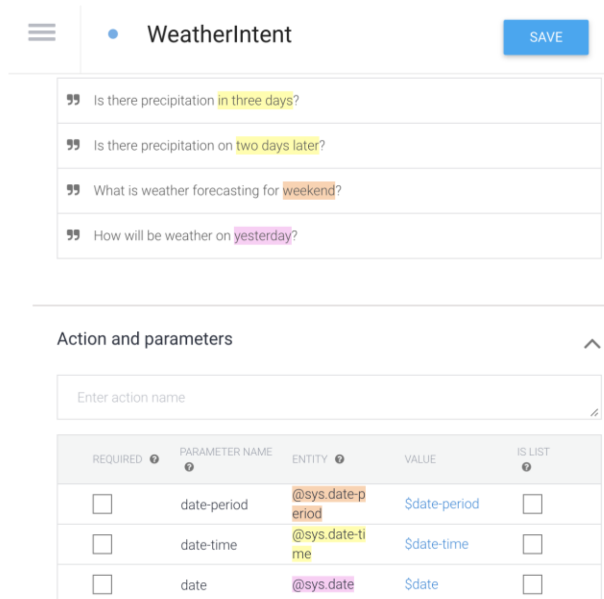

Fig. 5. An Example of Dialogflow Intents

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1 NEURAL NETWORK SECTION

To check the usefulness of the neural network models provided in the previous section i.e., the GRU and LSTM model, the method based on common sense as the reference model was offered according to the principle that the feature values will be the same as they are now for the next 24 hours, to make the simple prediction [17].

The values of five features in the next 7 days were predicted by the three models listed above, and the MAE results as the accuracy prediction scores for all of them are shown in Fig. 6, 7, 8, 9, and 10 respectively.

Date	tempmax		
	reference	GRU	LSTM
1	1.8675	2.0681	2.1955
2	2.4111	2.4042	3.1251
3	2.7398	2.8416	2.5834
4	2.9179	2.6616	2.7943
5	3.1215	2.9316	2.7240
6	3.2323	2.7595	2.8174
7	3.3979	2.8703	2.8754

Fig. 6. The MAE for the maximum temperature prediction in °C

Date	tempmin		
	reference	GRU	LSTM
1	1.8560	1.7053	1.9860
2	2.3842	2.1899	2.2067
3	2.5614	2.2438	2.3445
4	2.7338	2.3203	2.7532
5	2.9113	2.4132	2.6324
6	2.9329	2.4088	2.4199
7	2.9196	2.5482	2.4842

Fig. 7. The MAE for the minimum temperature prediction in °C

Date	windspeed		
	reference	GRU	LSTM
1	5.4713	5.1312	5.2381
2	6.8123	5.6581	5.9081
3	7.3961	5.5739	5.6141
4	7.6501	5.5850	5.6689
5	7.5662	5.5761	5.6453
6	7.4285	5.5793	5.6374
7	7.4140	5.6541	5.6427

Date	precip		
	reference	GRU	LSTM
1	2.3040	1.5623	1.5743
2	2.4086	1.5566	1.5458
3	2.4104	1.5918	1.5516
4	2.4132	1.5559	1.5618
5	2.5490	1.5780	1.5665
6	2.5381	1.5688	1.5604
7	2.5239	1.5706	1.5716

Fig. 8. The MAE for the wind speed prediction in m / s

Fig. 9. The MAE for the precipitation prediction in mm

Date	Number of Bicycle Hires		
	reference	GRU	LSTM
1	4219.940	4647.714	4640.461
2	5342.396	4491.454	4426.313
3	5832.640	4332.369	4497.432
4	5806.306	4432.900	4811.090
5	5524.948	4529.137	4934.056
6	4995.684	4647.183	4727.532
7	4578.321	4986.796	5121.011

Fig. 10. The MAE for the number of bicycle hires prediction

It shows that the majority of MAE values for the reference model prediction are higher than those for the neural network model (GRU and LSTM) prediction, which validates the robustness of the neural network models. In addition, most MAE values for the GRU model prediction are lower than those for the LSTM model prediction, this is due to the lower sensitivity for the hyperparameter choices in the GRU model, and fewer parameters need to be tuned due to its simplified architecture, which could result in the higher prediction performance for the GRU model [18]. It is obvious that the MAE values for the number of bicycle hires prediction are larger than those for the other four features predictions since the bicycle hires number is not only dependent on the weather factors in reality e.g. daily traffic flow, which leads to the inaccurate prediction result for the number of bicycle hires. Apart from the MAE values for the reference model prediction, most MAE values for the 7th-day prediction are higher than those for the previous days' predictions, which is common sense, but the more probable reason is the lack of actual data from Day 1 to Day 6 (the data for these days need to be predicted) to forecast the features, thus cause the inaccurate result for the 7th-day prediction.

Moreover, four training process plots for the GRU model and the LSTM model were chosen respectively, which are displayed in Fig. 11 to Fig. 18.

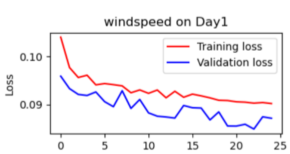

Fig. 11. Training process for predicting the wind speed on Day 1 (GRU model)

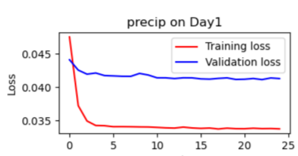

Fig. 12. Training process for predicting the precipitation on Day 1 (GRU model)

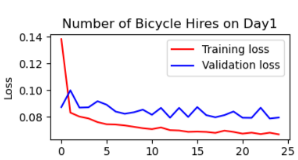

Fig. 13. Training process for predicting the number of bicycle hires on Day 1 (GRU model)

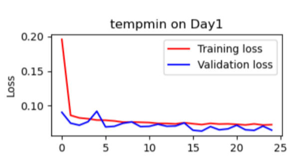

Fig. 14. Training process for predicting the minimum temperature on Day 1 (GRU model)

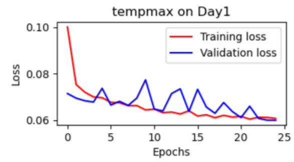

Fig. 15. Training process for predicting the maximum temperature on Day 1 (LSTM model)

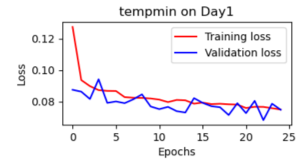

Fig. 16. Training process for predicting the minimum temperature on Day 1 (LSTM model)

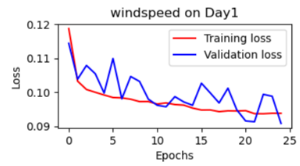

Fig. 17. Training process for predicting the wind speed on Day 1 (LSTM model)

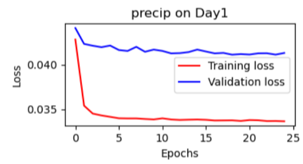

Fig. 18. Training process for predicting the precipitation on Day 1 (LSTM model)

From the figures, it is apparent that the overfitting issue of the model is avoided in this project since the curves for validation loss and training loss do not diverge dramatically when the epochs number rises. Additionally, the validation loss and training loss decrease when the epochs number increases, which follows the correct trend in theory. However, it is worth mentioning that the training loss tends to stop dropping and remain constant after a few epochs in each plot, which shows that the models have achieved their best performance.

4.2 INTERACTIONS AND SERVER RESPONSES

This demonstrates how the server is equipped to interpret sentences in natural language, identify important details, and leverage the machine learning model to deliver predictions that are aware of the context. Fig. 19 and Fig. 20 show two examples on the client side, respectively

In the example illustrated in Fig. 19, the client sets a specific date to be 2023-08-22 and can predict weather conditions and the number of bicycle hires for the seven days following this date. For the client request weather prediction, when the client asks “How is the weather for tomorrow?”, the server identifies the keyword” weather” and extracts” tomorrow” as the day number. It then generates a weather forecast “for day 1 using the machine learning model. Therefore, for day 1, the forecast is high at 22°C, low at 15.2°C, wind speed at 18.4 km/h, and precipitation levels around 0.0mm”. in addition, for the client request number of bike hires predictions, when the client asks “Can you estimate the number of bike hires on day 2?”, the server recognizes” bike hires” and extracts ”2” as the day. It predicts the number of bike hires for day 2 based on the model. So, for Day 2, the expected number of bike hires is approximately 32119.”. Similarly in the first example the weather is predicted for the fifth day and we can end the conversation with “goodbye”.

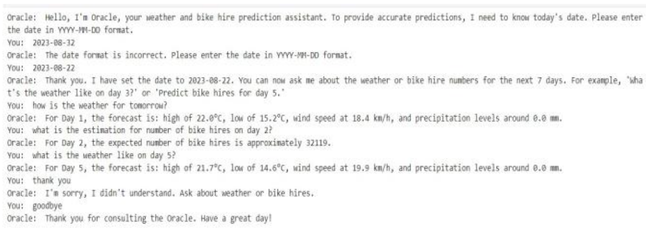

Fig. 19. Example 1 from the client side

5. CONCLUSION

In conclusion, both GRU and LSTM models show a strong capability and reach their optimum behaviour to predict the weather and the number of bicycle hires for the next 7 days. However, the GRU model could be more suitable for this project according to the smaller MAE values for its prediction, although the LSTM model has a higher learning ability theoretically due to its complex structure. If time is permitted, additional data on factors that may affect the number of bicycle hires can be found and added to the original dataset to obtain more accurate prediction results.

Regarding the client-server part, the server uses a combination of keyword recognition and regular expressions to process client sentences that can answer the status of the weather and the number of bike rentals seven days after a specified date. In the future, keywords will be added, more specific feature information will be recognized, and clients can ask more specific questions and be answered by the server.

With Google Cloud, there is a Cloud Build structure which is used to access Google Cloud's GitHub repository. After access, a trigger was created to set when this should run. With this 'cloudbuild. yaml' file in the repository, it specifies what to do when Google Cloud Build is triggered. In this section, accessing the GitHub repository gives successful results. However, the Deployment part could not be carried out successfully. Although the parts of setting up Google Cloud, using the App Engine structure, creating Dialogflow, accessing GitHub, and creating a flask compatible with the server file and Dialogflow were completed, the result images could not be added because the deployment part was not successful. Therefore, the fulfilment could not be accessed. In the future, the deployment part will hopefully be addressed.

6. REFERENCES

[1] Baheti, P. (2023) Train test validation split: How to & best practices [2023], V7. Available at: https://www.v7labs.com/blog/train-validation-test-set#:~:text=In%20general%2C%20putting%2080%25%20of,dimension%20of%20the%20data%2C%20etc. (Accessed: 26 March 2024).

[2] Krishnan, S. (2022) How do determine the number of layers and neurons in the hidden layer?, Medium. Available at: https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3 (Accessed: 27 March 2024).

[3] Swain, S. (2021) Understanding sequential vs functional API in Keras, Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/07/understanding-sequential-vs-functional-api-in-keras/ (Accessed: 27 March 2024).

[4] Chollet, F. (2021) '6.3.7 Stacking recurrent layers', in Deep learning with python. Shelter Island: Manning Publications, pp. 217–218.

[5] Chollet, F. (2021) '6.3.6 Using recurrent dropout to fight over-fitting', in Deep learning with python. Shelter Island: Manning Publications, pp. 216–217.

[6] Vijay, U. (2019) How does dropout help to avoid overfitting in neural networks?, Medium. Available at: https://medium.com/@upendravijay2/how-does-dropout-help-to-avoid-overfitting-in-neural-networks-91b90fd86b20 (Accessed: 27 March 2024).

[7] Buhl, N. (2023) Activation functions in neural networks: With 15 examples, Encord. Available at: https://encord.com/blog/activation-functions-neural-networks/#:~:text=Activation%20functions%20play%20a%20crucial,and%20numerous%20non%2Dlinear%20variants. (Accessed: 27 March 2024).

[8] Thakur, A. (2020) Relu vs. sigmoid function in deep neural networks, W&B. Available at: https://wandb.ai/ayush-thakur/dl-question-bank/reports/ReLU-vs-Sigmoid-Function-in-Deep-Neural-Networks--VmlldzoyMDk0MzI (Accessed: 27 March 2024).

[9] Chollet, F. (2021) '6.3.4 A basic machine-learning approach', in Deep learning with python. Shelter Island: Manning Publications, p. 213.

[10] Adam, W. (2020) Which optimizer should I use for my ML project?, Data Curation for Machine Learning. Available at: https://www.lightly.ai/post/which-optimizer-should-i-use-for-my-machine-learning-project#:~:text=You%20choose%20a%20simple%20stochastic,memory%20to%20store%20the%20state. (Accessed: 27 March 2024).

[11] Aboze, J. (2022) What is rmsprop, Deepchecks. Available at: https://deepchecks.com/glossary/rmsprop/#:~:text=RMSprop%20has%20fewer%20hyperparameters%20than,performance%20on%20non%2Dconvex%20problems. (Accessed: 27 March 2024).

[12] Surmenok, P. (2021) Estimating an optimal learning rate for a deep neural network, Medium. Available at: https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0 (Accessed: 27 March 2024).

[13] Oppermann, A. (2022) How loss functions work in neural networks and Deep Learning, Built In. Available at: https://builtin.com/machine-learning/loss-functions (Accessed: 27 March 2024).

[14] Acharya, N. (2023) Choosing between mean squared error (MSE) and mean absolute error (MAE) in regression: A deep dive, Medium. Available at: https://medium.com/@nirajan.acharya666/choosing-between-mean-squared-error-mse-and-mean-absolute-error-mae-in-regression-a-deep-dive-c16b4eeee603 (Accessed: 27 March 2024).

[15] Rsvmukhesh (2023) Determining the number of epochs, Medium. Available at: https://medium.com/@rsvmukhesh/determining-the-number-of-epochs-d8b3526d8d06 (Accessed: 27 March 2024).

[16] Thakur, A. (2020) What's the optimal batch size to train a neural network?, W&B. Available at: https://wandb.ai/ayush-thakur/dl-question-bank/reports/What-s-the-Optimal-Batch-Size-to-Train-a-Neural-Network---VmlldzoyMDkyNDU (Accessed: 27 March 2024).

[17] Chollet, F. (2021) '6.3.3 A common-sense, non-machine-learning baseline', in Deep learning with python. Shelter Island: Manning Publications, p. 212.

[18] Srivatsavaya, P. (2023) LSTM vs gru, Medium. Available at: https://medium.com/@prudhviraju.srivatsavaya/lstm-vs-gru-c1209b8ecb5a (Accessed: 28 March 2024).