08 August 2021

Research project IoT

Haithem BEN ABDELAZIZ

Expert embedded systems

Introduction

The Internet of connected objects represents the exchange of information and data from real world devices with the Internet network.
Considered the third evolution of the Internet, dubbed Web 3.0, has a universal character to designate connected objects for various uses, in the field of e-health, home automation.

The huge amount of data produced by connected objects requires storage space,processing speed and often bandwidth for streaming audio or video data. Forsome, the ideal solution to these problems is cloud computing.

Google and IBM are the main providers of IoT services, they offer platforms for data storage, visualization and analysis.

The objective of the work is to create an application in the field of connected objects which consists to simulate a network of buses and stations, considering that each bus has a sensor which transmitted its position to the cloud, each bus station receives the position of the buses then calculates the time to arrive from each bus.

The second part of this work is to apply one of the machine learning algorithms and predict bus wait times.

IOT short description

Fundamental components of an IoT system:

1) Sensors/Devices: Sensors or devices are a key component that helps you to collect live data from the surrounding environment. All this data may have various levels of complexities. It could be a simple temperature monitoring sensor, or it may be in the form of the video feed.

A device may have various types of sensors which performs multiple tasks apart from sensing. Example, A mobile phone is a device which has multiple sensors like GPS, camera but your smartphone is not able to sense these things.

2) Connectivity: All the collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud using various mediums of communications. These communication mediums include mobile or satellite networks, Bluetooth, WI-FI, WAN, etc.

3) Data Processing: Once that data is collected, and it gets to the cloud, the software performs processing on the gathered data. This process can be just checking the temperature, reading on devices like AC or heaters. However, it can sometimes also be very complex like identifying objects, using computer vision on video.

4) User Interface: The information needs to be available to the end-user in some way which can be achieved by triggering alarms on their phones or sending them notification through email or text message. The user sometimes might need an interface which actively checks their IoT system. For example, the user has a camera installed in his home. He wants to access video recording and all the feeds with the help of a web server.

Program structure

To analyze the problem, we created 2 class Bus.py, Station.py, a file utils.py for the utility functions and main.py for the main function.
The bus_predicition.py file contains analyzes the data collected by the stations, the weather wait time, then it predicts the wait time with a machine learning algorithm.

Class diagram

Bus class:

The bus class contains 6 attributes and 6 methods:
Id: a unique identifier of integer type for each bus.
Img: an image that represents the bus on the GUI.
Stop: a Boolean variable, true if the bus has stopped.

Temperature: refers to the temperature of the real-type bus, generated in a random fashion.
X_pos: an integer that represents the position of the bus in the GUI relative to the X axis.
Y_pos: an integer that represents the position of the bus in the GUI relative to the Y axis.
setPosition (): a method that is used to initialize the position of the bus.
getPosition (): a method used to get the position of the bus.
sendPosition (): a method that is used to send the position of the bus to the cloud.
setTemperature (): a method used to set the temperature of the bus.
getTemperature (): a method used to retrieve the temperature of the bus.
sendTemperature (): a method used to send the temperature from the bus to the cloud.

Class station:

The bus class contains 7 attributes and 5 methods:
Id: a unique identifier of type integer for each station.
Img: an image that represents the bus station on the GUI.
logger: a log file that saves the data received and the calculated waiting time.
Schedule_bus: list type variable
X_pos: an integer that represents the position of the bus in the GUI relative to the X axis.
Y_pos: an integer that represents the position of the bus in the GUI relative to the Y axis.
setPosition (): a method that is used to initialize the position of the bus.
getPosition (): a method used to get the position of the bus.
sendPosition (): a method that is used to send the position of the bus to the cloud.
setTemperature (): a method used to set the temperature of the bus.
getTemperature (): a method used to retrieve the temperature of the bus.
sendTemperature (): a method used to send the temperature from the bus to the cloud.

Random speed generation

The buses move forward by a single pixel each t = time.sleep (uniform (T_MIN, T_MAX)).

Threads using

In our application, we used the `` threads '' which are execution units stand-alone that can perform tasks, in parallel with other threads, such as for displaying each bus on the graphical interface, and sending data, and calculations made by each station.
thread_bus_1 = Thread (target = running_bus_1, args = (bus_1, False))
thread_bus_1.start ()
thread_send_data = Thread (target = send_data, args = (bus_1, bus_2, bus_3))
thread_send_data.start ()

Which programming language

We have opted for the Python language, which is a very powerful programming language, moreover in addition used to develop applications quickly and at the same time efficient. He is very used in the scientific world for its airy syntax and fast execution speed.

This language is also excellent for the creation of prototypes because its simplicity allows to implement a project as quickly as you think about it.

We used several libraries, which are mentioned in the requirements.txt file, and installable with the following command :

pip install -r requirements.txt

Pygame: For the graphical interface
Ibmiotf: Client libraries and connection examples to IBM Watson IoT.
matplotlib: Library to plot and visualize data in the form of graphs.
numpy: Library intended to handle multidimensional matrices or arrays.
paho-mqtt: Open source implementation of MQTT messaging protocol.

MQTT messaging

MQTT is the primary protocol used by terminals and applications to communicate with IBM WatsonTM IoT Platform. MQTT is a publishing and messaging transport protocol subscription designed to optimize real-time data exchange between the sensor and the mobile terminals.
MQTT runs over TCP / IP, and although it is possible to encode directly for TCP / IP, you can also choose to use a library that handles protocol details for you MQTT.
A wide variety of MQTT client libraries are available. IBM contributes to development and support of several client libraries, including those that are not available at the following sites:
There are also size limitations for message content on Watson IoT Platform. Messages can contain any valid string, however, JSON ("json") formats, text ("text") and binary ("bin") are the most commonly used.

The following table shows the message content restrictions for the different types.

Format de contenu	**Instructions pour des cas d'utilisation spécifiques**
JSON	JSON is the standard format for Watson IoT Platform. If you plan to use Watson IoT dashboards, tables and maps and analytics functions Integrated platforms, make sure the message content format conforms to correctly formatted JSON text.
Texte	Use valid UTF-8 character encoding.
Binaire	No restriction.

Important:
The maximum message content size on Watson IoT Platform is 131,072 bytes. Messages with content greater than the limit are rejected. The client during connection is disconnected and a message is displayed in the diagnostic logs

Automatic learning

Pretreatment

First, we collected the data in an *activity.log* file which contains activity log of our application in the form of data separated by - 02/01 / 18- 20:16:36 - 2018 - 131 - 80 - {'2019': 1000.0, '2020': 1000.0, '2021': 1000.0}.
Then we deleted the duplicate lines and generated a *train.txt* file, by the function *delete_duplicate ()*.
the text_to_csv () function allowed us to select all the data (rows) that correspond to the 2018 bus and generate a file in csv train.csv format which will be used in the prediction of the bus waiting time.

Prediction

Concerning the prediction part we have implemented a predict () function which applies the Bayesian regression to predict 2018 bus wait time.
The result is shown in graph format below.
Figure 2 comparison of waiting times (actual, predicted)

Conclusion

During this work we were able to put into practice our theoretical knowledge of the field of
connected objects, using IBM technologies such as the bluemix platform, the MQTT protocol
and Python as a programming language, thus doing an analysis of the data generated by the
Machine Learning techniques by applying the Bayesian regression algorithm to predict
the waiting time for buses.