Module Exam

Module code and Name	DE4-SIOT Sensing & IoT
Student CID	01049103
Assessment date	10th Jan 2019

Presentation URL (publicly accessible link):
https://www.youtube.com/watch?v=yshwWDmwJr8

Code & Data (publicly accessible link):

Coursework 1: Sensing

Introduction

In today's world of social-media savvy individuals, several conclusion can be drawn from the content of their posts, tweets or hashtags. It has also been shown that moods and behaviours are heavily influenced by local daylight levels (1). By combining information gathered from a social media API and locally recorded light levels, several conclusions could potentially be drawn.

Prior work in this field has focussed on correlating weather with tweet sentiment. Tweether, a real time weather and tweet visualisation tool does just this. It collects live tweet data and weather from APIs to create visualisations of the information.

The project scope was based on the geographic location that was available during the project execution time. In this case, the location was the my home, situated in North West London, near to Hampstead Heath. This is a well-known, popular location for dog walkers (2). The incidence of the word 'dog' used across social media could be tracked in correlation with daylight levels. This had potential to uncover not only dog walking patterns but also when people are interacting with their pets most. By using daylight level, it combined both time of day and local weather conditions into a single metric. On top of this, basic sentiment analysis was completed to provide greater insight into the pattern of data.

The collected data could be used by various groups such as advertisers, local shops or residents for a variety of purposes including timed marketing, anticipating customer surges or finding a social time to walk their dog respectively.

Objectives

This project had four main objectives:

1. Collect light level using a sensor and wirelessly transmit this data to the my personal computer for further analysis. This would be continuous time-series data collected in real time.
2. Collect social media posts in the local area that use the word 'dog'. Again, this would be collected and stored in real time.
3. Analyse the sentiment of each tweet
4. Complete data analysis on these data sources together to highlight patterns
5. Create an online platform to visualise the collected data and analysis for potential users

Planning

In order to keep the project on track, a Gantt chart was used as shown in Figure 1. This accounted for the stages of design including development, testing and delivery.

Figure 1 - Project Gantt Chart

Data Sources and Sensing Set-Up

Development

At the top level, a Raspberry Pi was selected over Arduino due to its full computational capacity. This allowed for easy data storage and wireless upload without need for additional components.

When looking at collecting light level data from the local area surrounding my house, the options were split between

1. Existing data collection services including APIs
2. Setting up a new collection sensor

For the purposes of furthering my own breadth of knowledge and skills in this area, option two was selected. This is a completely different method used to the social media technique used, giving variety to the project.

There were a range of light sensor types to choose from including light dependent resistors, photovoltaic and photodiode. These sensors all have different properties including measured wavelength range and intensity measurements.

To collect data from social media, there were three main options; Twitter, Facebook and Instagram. These are the most widely used social media platforms in the UK excluding YouTube, which doesn't fit the requirements of this project (3). From these, Instagram was eliminated due to its restrictive rate limits (4). Twitter was selected over Facebook due to the way in which it used by a wide audience to share primarily text-based information.

Selected Data Sources

To measure light intensity, sensor BH1750 was selected for its low cost, easy accessibility and suitability to 'obtain the ambient light data 'using a photo diode 'with approximately human eye response (5). It was able to capture light level at high resolution to the nearest lux.

On the social media side, Twitter was used to search for the keyword 'dog' within a 15 km radius of the Raspberry Pi location. This radius was selected as a local area that would likely have the same weather at one time, and therefore light intensity.

Light Level Sensing Set-Up

The light sensor itself was mounted inside a small box and sealed to the window using tape, as shown in Figure 2. This method helped to isolate the light sensor from light inside the room or the effects of the curtain being opened or closed. After testing, it was found that turning on and off the lights inside changed the reading by 0 lx, while opening and closing the curtains changed the reading by 3 lx. This was deemed an acceptable fluctuation as it is 0.4 % of the average daylight reading.

Figure 2 - Light sensor set up

The sensor communicated with the Raspberry Pi (RPi) using the I^2C bus on port 1. The sensor itself had several resolution levels. High resolution was selected, giving results to the nearest lux. This was important for collecting results at dawn and dusk where the light levels would increase or decrease respectively very quickly. Figure 3 shows the poor response of the low resolution option at lower light levels. High resolution mode has a longer measurement time, although this is unimportant for this application.

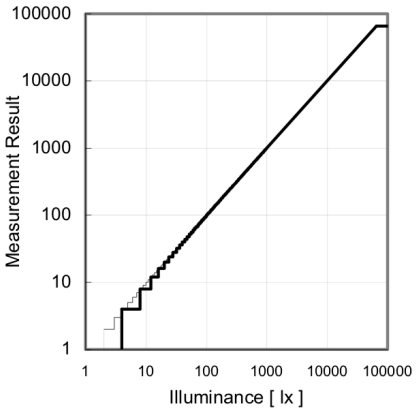


Figure 3 - High resolution (grey) and low resolution (black) modes of the light sensor (5)

Twitter Sensing Set-Up
Before setting up the Twitter sensing platform, a developer account was made and an app created. This allowed keys, secret keys and access tokens to be generated for the app. These were then authenticated at the beginning of the programme script.

Data collection and Storage Process

Data Collection
Both the Twitter and Light Intensity were sampled at a rate of 1/300 Hz, equivalent to one sample every 5 minutes. This rate was selected to be suitable for both data sources.

Light intensity at dawn and dusk changes significantly within the space of an hour. It was found that collecting a data point every 5 minutes gave sufficient precision to interpolate the missing data. Furthermore, the light intensity was sampled every 10 seconds for the 5 minute interval and then averaged. This helped to reduce the effect of anomalies, such as a bright car driving past. These intermediate measurements were only stored for the 5 minutes that they were relevant, and then deleted. This script was adapted from an existing program by Matt Hawkins (6).

Tweepy, a Python wrapper for the Twitter API was used to collect the tweets. After some preliminary testing, it was found that there are usually under 10 new Tweets every five minutes. This rate is well below the twitter rate limit of 15 requests per 15 minutes. The script searches for new tweets (using the `sinceID` argument) with the keyword ‘dog’ within a 15km radius of the home, with geographical coordinates -----, ------.

The sentiment of each Tweet was recorded as either positive, negative or neutral. This analysis was completed by an existing Python based sentiment analysis library, TextBlob. The implementation of this library was adapted from the GeeksforGeeks sentiment analysis script (7). The TextBlob sentiment analysis works by comparing the tweet with a model of labelled data created from movie reviews.

Communication Protocols
To communicate with the RPi, a secure shell (SSH) was used. Through use of Remote.it, the RPi could be controlled remotely. When the RPi was set up, a Weaved service was installed. Screen was then used to run multiple terminals at once and conveniently switch between them. This was incredibly useful when running two scripts at once, as it allowed the terminal to be detached from without interrupting the script.

Storage
Every five minutes, the data was stored in a CSV file, with each new entry forming a new row. There were two separate files, one for the light data and one for Twitter. Every 12 hours, a new CSV file was created and the data was stored there for the following 12 hours. This prevented any problems that could have arisen from overwriting or spoiling the earlier data.

To prevent data loss, these CSVs were backed up online every 24 hours. This was done using Rclone, a command line program to sync files and directories. The Box variant of this software was selected to copy all CSVs to a folder in a Box account. This allowed the data to be viewed from any computer at any time. A bash script written by Claude Pageau (8) was adapted to copy the ‘Data’ folder, stored locally on the RPi to a ‘SIOT’ folder on the my Box account. By copying files rather than syncing the folder, no data is ever overwritten. When a file is copied twice, it is stored as a new ‘version’ on Box. A cronjob was then created to run the bash script every 24 hours.

Development

Before using Box to back up data, several other options were trialled. Firstly, it was attempted to use the Box API directly. This proved overly complex for what was required, so Google Drive was trialled. PyDrive was used to automatically backup the files. The problem with this method was storage size; the my Google Drive was almost full. This is when RClone was trialled with Box, and worked well as a smooth way of backing up the information to a location with unlimited storage.

End-to-end System

A system diagram for the entire platform can be seen in Figure 4.

The system is operated by a Raspberry Pi model 3B+ connected to WiFi. This computer ran two Python scripts continuously for 19 days. One script collected the light sensor information, while the other collected Twitter information. These scripts both wrote to CSVs in a 'Data' folder stored locally on the RPi. As stated earlier, these CSVs were then uploaded to Box using RClone.

To use this data for the next stage of the project, it was downloaded to a folder stored locally on the my laptop.

The scripts were interrupted twice during the course of the data collection, leaving short gaps of data. These gaps are sufficiently short that interpolation can be used.

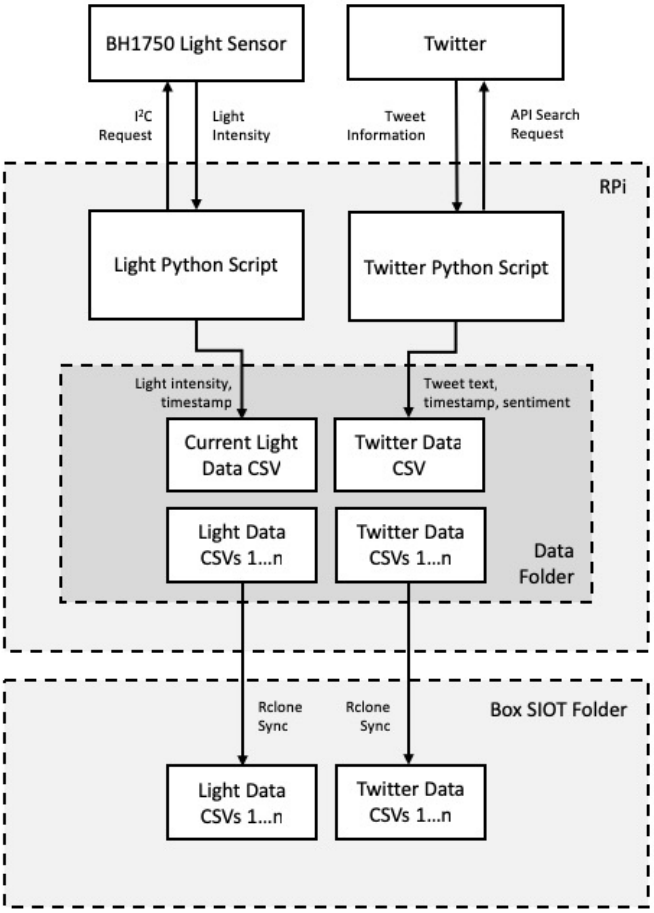


Figure 4 - Sensing system diagram

Basic Time Series Data Analysis

The data collected was recorded on very different scales, making them difficult to compare directly. The two datasets were normalised, as shown in Figure 5. Comparing the Light Level and Twitter count data, it can be seen that the peaks of the light level lags slightly behind the twitter count peaks. The Tweet sentiment score behaves erratically without a clear visual pattern.

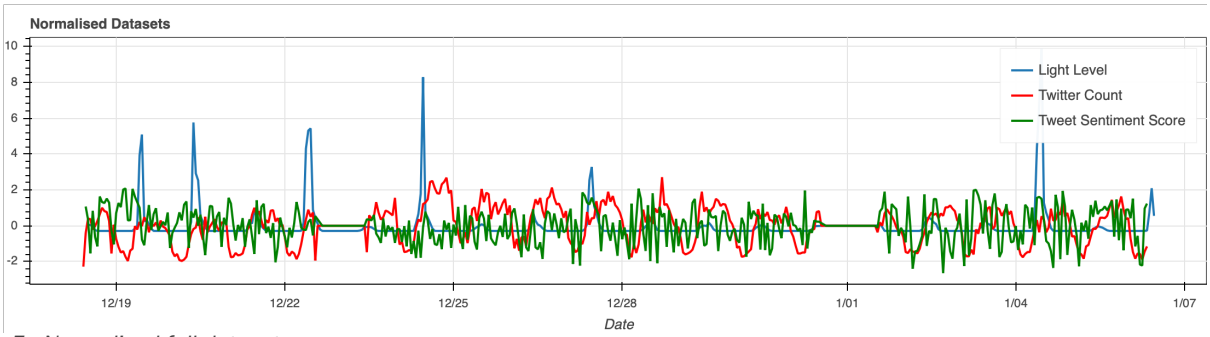


Figure 5 - Normalised full datasets

Coursework 2: Internet of Things

Data Platform

Structure

A web application called 'LightDogs' was aimed at Pet Product Companies. By correlating light levels and tweets using the word 'Dog', LightDogs could predict at what time marketing would be most effective for a business. This makes the assumption that if more people are Tweeting about dogs, more people are currently receptive to advertisements.

LightDogs was built using the Flask framework. This allowed the site to be based on Python logic, with HTML markup to create the webpages. After creating an instance of the Flask class, a `route()` decorator tells Flask what URL should trigger each function. Every function in the routes script corresponds with one webpage.

The data was originally stored in an SQL using SQLAlchemy, a Python SQL toolkit. However, it was found that during data processing the data needed to revert back to a Pandas DataFrame, meaning there wasn't much value in having the SQL at all. In the final LightDogs version, two Pandas DataFrames (one for Light, one for Twitter) were instantiated in the `__init__` file, allowing access from all other scripts. The helpers.py script was used to format the data into dataframes, while keeping the `__init__` script clean.

The entire app was structured as shown in Figures 6 and 7. When `main.py` is run, the app is run on a local port.

```
.
└── LightDogs
    ├── dashboards.py
    ├── __init__.py
    ├── data
    │   ├── l_out.csv
    │   └── t_out.csv
    ├── data_preprocess.py
    ├── helpers.py
    ├── main.py
    ├── routes.py
    ├── sentiment.py
    ├── static
    │   ├── Logo.jpg
    │   ├── css
    │   │   ├── bootstrap-theme.min.css
    │   │   ├── bootstrap.min.css
    │   │   ├── home.css
    │   │   ├── jqcloud.min.css
    │   │   └── main.css
    │   └── js
    │       ├── bootstrap.min.js
    │       ├── jqcloud.min.js
    │       └── script.js
    └── templates
        ├── about.html
        ├── analysis.html
        ├── home.html
        ├── index.html
        ├── insights.html
        ├── prediction.html
        ├── result.html
        └── timeline.html
```

Figure 6 - Project Structure

Data Visualisation: Graphs

Bokeh visualisations were created and embedded into the web app using the `components()` function. This function returns a `<script>`, containing the plot data and a `<div>` tag that the plot will be loaded into. These tags are then loaded into HTML templates. A variety of line and scatter plots were used depending on the data being displayed. The `dashboards.py` script completes this work and returns the `<script>` and `<div>` tags to `routes.py`.

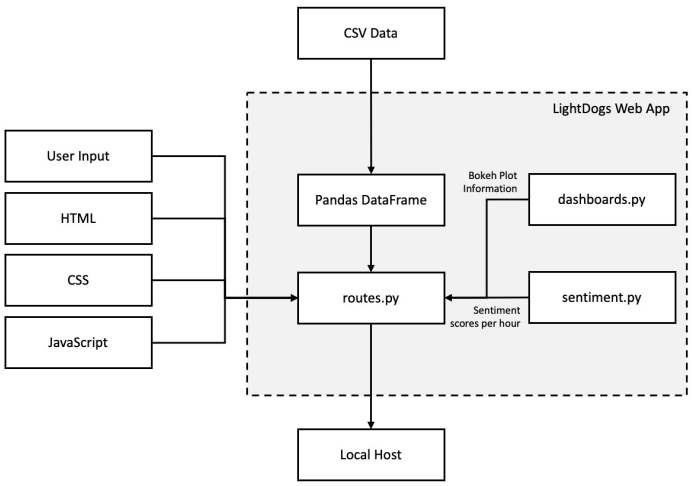


Figure 7 - LightDogs web app system diagram

Data Interaction: GUI

Bootstrap was used to improve the User Interface (UI) and User Experience (UX) of the web app. Custom CSS was then used to overwrite some of the Bootstrap standard UI, changing the look and feel of the LightDogs web app, as shown in Figure 8. A master template called 'index' was used to format the basis of every page, providing a navigation bar at the top. This HTML file was then inherited by every other HTML file. Every further page simply inserted it's content into the `block content` container of the index file.

It was important that users could interact with the Bokeh data visualisations for their individual purpose. The tools section of each Bokeh plot were altered to allow for panning, wheel zooming, box zooming, resetting and downloading the plots. Some plots were also linked to behave simultaneously, such as the Timeline Plots. A hover tool was included to allow for more specific data analysis. An interactive legend was included, allowing the user to switch between visualised datasets. These features were all demonstrated in the project video.

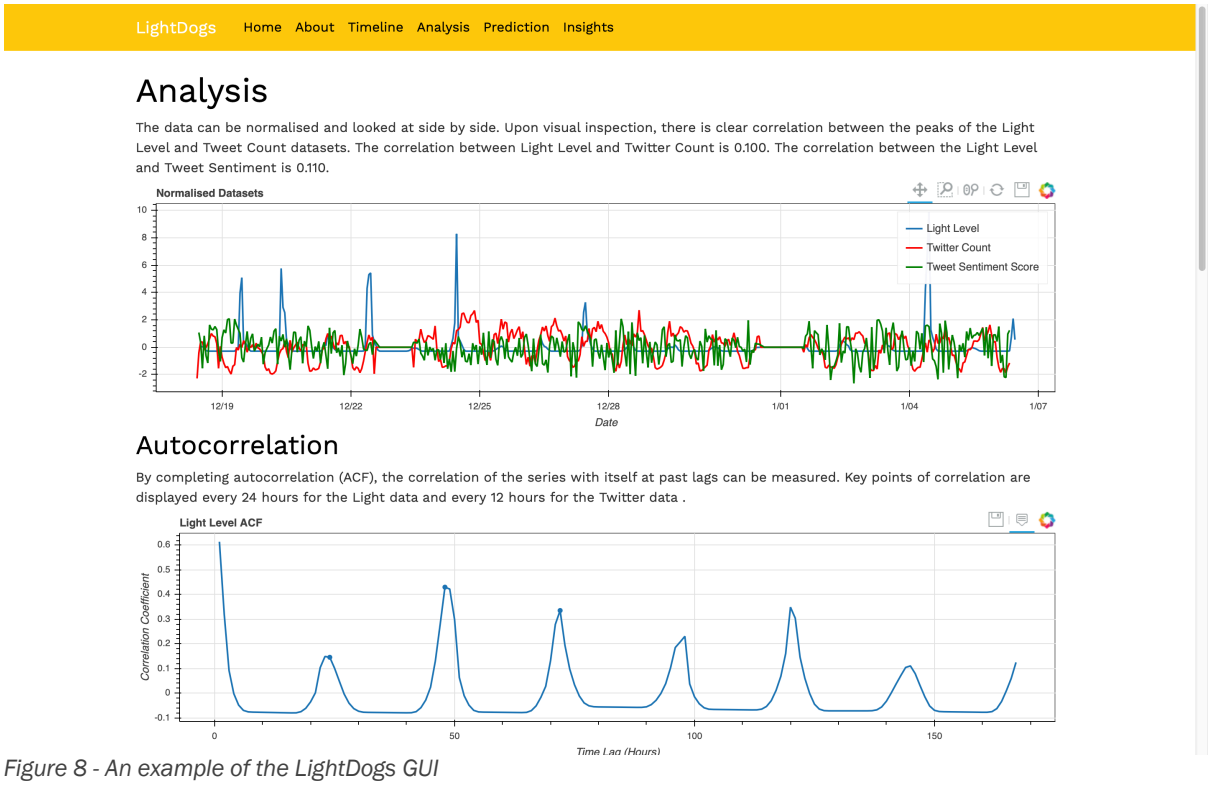


Figure 8 - An example of the LightDogs GUI

Data Visualisation: Word Cloud

A word cloud was also added to visualise the most common words used in every tweet. The more common the word, the larger and more centrally it is shown in the word cloud. As anticipated, the word 'dog' is most used. The word cloud worked by first searching through all tweets, and calculating the frequency of every word used, excluding words shorter than three letters and stop words, such as 'am', 'is' or 'are'. This data is then written to a JSON file. In the HTML for the page, a JQuery script calls the LightDogs endpoint, gets the data and builds a word cloud. A JavaScript file is then called, in which the LightDogs endpoint `word_cloud` is called and visualises the word cloud using JQCloud. This method was adapted from Prateek Jain's Tutorial on using JQCloud with Flask (9). The styling of this word cloud was customised to match the LightDogs UI, as shown in Figure 9.

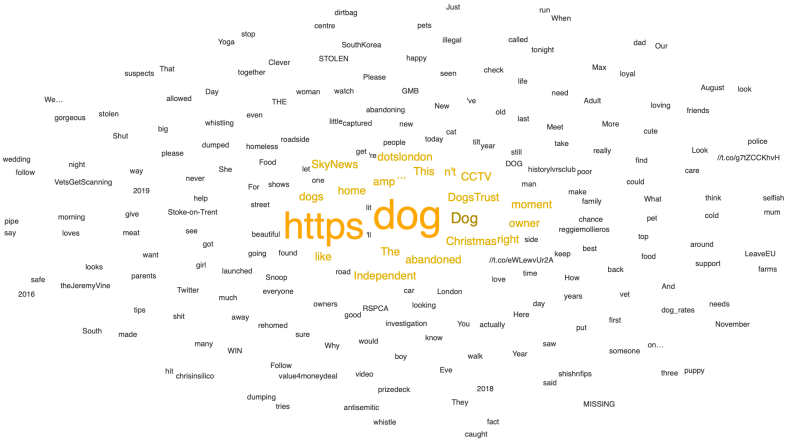


Figure 9 - Word cloud of most commonly tweeted words

Data Actuation

LightDogs allows users to find out if their selected time to send out marketing materials will be effective. On the 'Prediction' page, a date and time can be inputted, returning a statement telling the user if their selected time is good or not. This has been built using a web form with a JQuery date selector tool, as shown in Figure 10. This ensures that the inputted date is in the correct format. As this Web App was only a prototype, the results were based on a simple time filter, checking if the inputted time is between 7 am and 10 pm, the timings found to have most Tweet activity. In the future, more data could be collected over several months, and a statistical model could be built. By modelling the relationship between tweets and daylight level, as well as daylight level and time of day, a prediction could be made.

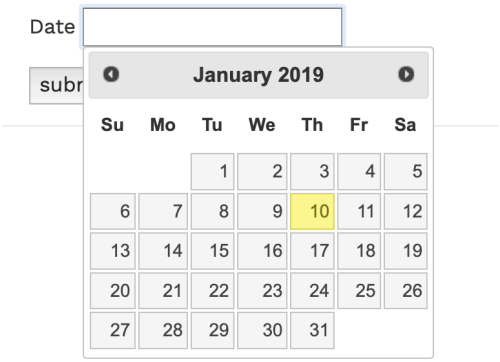


Figure 10 - JQuery datetime selector

Data Analytics, Inferences and Insights

All plots shown here have been directly downloaded from the LightDogs web App. The code was developed in a Jupyter Notebook and then transferred to the main project.

Pre-processing

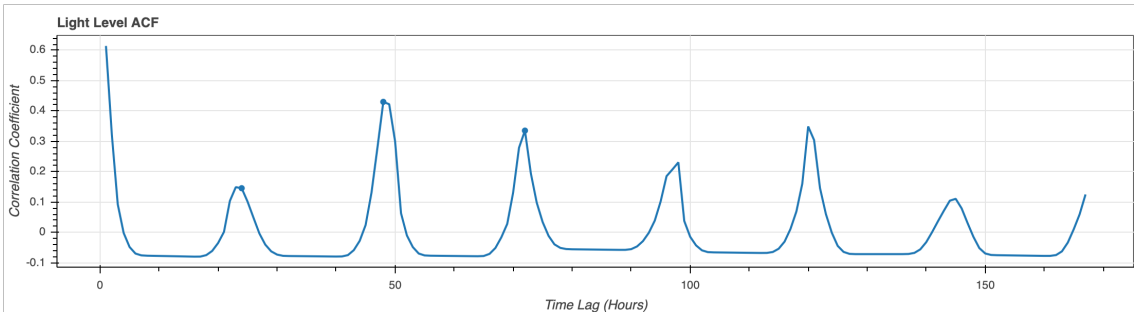
The data collected from Coursework 1 spanned 19 days, from 19th December 2018 to 6th January 2019. There was a gap in the Twitter data on the 24th December. This was caused by an error in the script collecting Tweets, preventing the script from running if no new tweets are found. This problem was rectified and a further 8 days of continuous data was collected. Another gap caused by a loss of WiFi connection paused the project on 1st January. These gaps were filled using an imputation method; all missing values were filled with the mean value of that dataset. After all of the data was collected, the numerous CSVs were combined into one for all of the Light data and one for all of the Twitter data by running the `data_preprocess.py` script. The twitter data was downsampled into one hour bins, showing how many tweets were found each hour.

The sentiment of each tweet was already recorded during Coursework 1. The script `sentiment.py` then calculated a 'sentiment score' for every hour of collected data using Equation 1. This equation assigns a score of 1 to every positive tweet, -1 to every negative tweet and 0 to every neutral tweet. These scores were then returned to `routes.py`.

$$sentiment\ score = \frac{(1)n_{positive} + (-1)n_{negative} + (0)n_{neutral}}{n_{positive} + n_{negative} + n_{neutral}} \quad (1)$$

Autocorrelation (ACF)

ACF correlates the time series data with itself at past lags, This is shown in Figure 11. As expected, the Light data is highly correlated every 24 hours while the Tweet Incidence and Sentiment data follows a 12 hour correlation peak cycle.



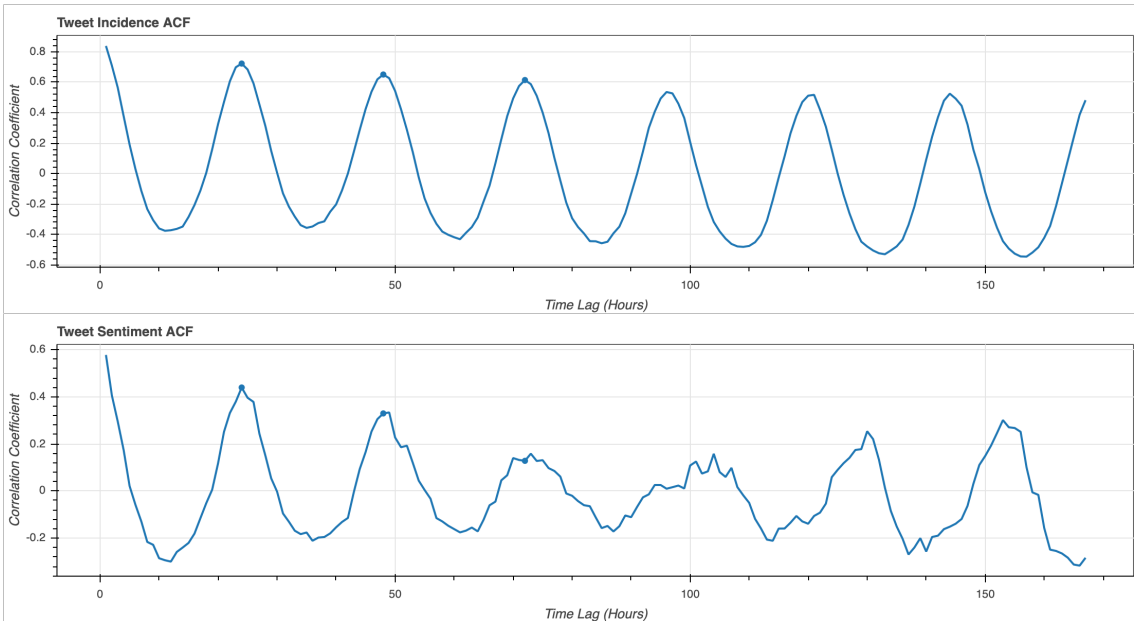


Figure 11 - ACF of each dataset

Seasonality

The data series could be broken down into their components; trend, seasonality and noise. In order to do this, the Light data was downsampled into one hour bins. The results can be seen in Figure 12.

Figure 12 - Trend, seasonality and noise of the datasets

Correlation

To measure the correlation between these datsets, the Pandas `corr` function was used to compute pairwise correlation. The results are shown in Table 1.

Table 1 - Correlation coefficients for each dataset

Dataset 1	Dataset 2	Correlation Coefficient
Light Level	Tweet Count	0.100
Light Level	Tweet Sentiment	0.110

Discussion

From the analysis completed, several conclusions can be drawn from the collected data. Upon visual inspection of the normalised datasets (Figure 5) , it can be seen that the Light Level and Tweet Incidence peaks are correlated, with the daylight peak lagging just behind the tweet incidence peak. This indicates that people tweet more in daylight hours and dusk. The correlation scores (Table 1) are both low, indicating that there is not a strong correlation between the datasets, despite this feature. After autocorrelation (Figure 11) was completed, it can be seen that the Light Levels operate on a 24 hour cycle, as expected the tweet incidence and sentiment follows a 12 hour cycle of peaks and troughs. Looking at the seasonality of the data (Figure 12) showed that the number of Tweets peaked on Christmas day while the sentiment of these tweets was lowest just before the 25th of December. The seasonal data shows the same patterns as were found through ACF. Finally, the word cloud (Figure 9) as anticipated, displayed 'dog' as the most common word. The next most common was 'https', showing that a large proportion of the tweets included a hyperlink. The next most common words included 'abandoned', 'CCTV' and 'DogsTrust', showing that a large volume of the tweets were in relation to lost dogs or the prevention of such an incident.

Conclusions

Improvements

In the future, this project could be improved through three main measures:

1. Flask, although an excellent method to quickly launch Web Apps, does not have the same functionality as creating an app in JavaScript. If the project were to be rolled out on a larger scale with increased functionality, the framework should switch to Javascript.
2. Data should be collected from a wider catchment area to improve the accuracy based on location.
3. The question of 'correlation is not causation' needs to be further analysed, Light and Twitter data alone are not enough to draw concrete conclusions.

Evaluation

Overall the project achieved what it set out to do; two datasets from different sources were collected and compared via an online platform. The variety of techniques and processes used allowed my knowledge to be greatly expanded.

Avenues for future work and potential impact

The project was aimed at pet product companies, but only collected data with the word 'dog'. In the future, this could be expanded to account for a variety of other pets. The project could also be creatively adapted to fit a different market all together. By correlating more specific tweets about dog walking and daylight level, conclusions could be drawn about when most people are walking their dogs. A platform could then be used to show dog owners when other dog walkers are out, creating a kind of dog walking social media platform.

References

1. Julie Taylor. *Can Rainy Days Really Get You Down?* Available from: https://www.webmd.com/balance/features/can-rainy-days-really-get-you-down#1 [Accessed 10th January 2019]
2. City of London. *Dog Walking*. Available from: https://www.cityoflondon.gov.uk/things-to-do/green-spaces/hampstead-heath/events-and-activities/Pages/dog-walking.aspx [Accessed 10th January 2019]
3. Social Media. *Most Popular Social Networks in the UK*. Available from: https://social-media.co.uk/list-popular-social-networking-websites [Accessed 10th January 2019]
4. Facebook for Developers. *Overview*. Available from: https://developers.facebook.com/docs/instagram-api/overview/ [Accessed 10th January 2019]
5. Rohm Semiconductor. *Digital 16bit Serial Output Type Ambient Light Sensor IC*. ROHM Co; 2014. Available from: https://www.mouser.com/ds/2/348/bh1750fvi-e-186247.pdf [Accessed 10th January 2019]
6. Matt Hawkins. *BH1750*. [Code] Available from: https://github.com/seblucas/i2c2mqtt/blob/master/bh1750.py [Accessed 10th January 2019]
7. Geeks for Geeks. *Twitter Sentiment Analysis using Python.* Available from: Accessed 10th January 2019] https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/
8. Claude Pageau. *rclone4pi*. [Code] Available from: https://github.com/pageauc/rclone4pi/wiki [Accessed 10th January 2019]
9. Prateek Jain. *Visualise News Word Cloud using Python, Flask and JQCLoud*. Available from: https://www.codementor.io/prateekkrjain/visualize-news-word-cloud-using-python-flask-and-jqcloud-8i3w57cfb [Accessed 10th January 2019]