Location Information Algorithms

A Technical Report Submitted to the

Faculty of the School of Engineering and Applied Science
University of Virginia, Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

Bhaskar Singhvi

Spring 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____Bhaskar Singhvi_____

Location Information Algorithms

A Technical Report for UVa CS Capstone

Bhaskar Singhvi

***Abstract*—As the world becomes increasingly complex, there is a growing need for efficient algorithms that can parse through large amounts of data quickly. In this report, I outline two different scenarios of parsing information about certain locations for which two different approaches were needed. One highlights the how timed geolocation data can be used to understand areas of high interaction within a city. The second attempts to predict energy consumption of a particular area. The purpose of this exploration is to provide additional tools in generating insights about locations.**

I. Introduction

This research was conducted as part of the Capstone Research track of University of Virginia's Capstone sequence for the Bachelor of Science Degree in Computer Science. For these tasks, I was interested in how location data can better inform our understanding of the needs of the community. In this report, I will discuss two tools I implemented through Python 3.8 and software libraries, such as NetworkX and Keras, that attempt to address the two scenarios discussed earlier.

II. Task 1 - Tracking Interactions

A. Task Motivation

The COVID-19 pandemic in 2020 illustrated the importance of the need for data-based tools that can help public officials make more informed decisions. In this case, technology can help by processing large amounts of data to determine where are the places with the most interactions between people and what times. Through this, the intention of the task was to figure out what ares needed more attention to better mitigate a public health threat such as a virus.

B. Task Description

Interactions in a city can be analyzed in many ways, where two specific ways were studied in this task. The first was calculating the total movement occurring in a city across a day.The second method of analysis was determining what parts of the city were most densely populated in any given day. This could be done via using anonymous location data and matching it to points of interest (POI) across a city.

C. Task Method

Both these subtasks were completed by creating algorithms using Python, a powerful language for which there are extensive data processing libraries. For the movement patterns, adjacency matrices were used to record spots in a place visited by people at specific times, such as every hour in a day. Then by using the NetworkX library the matrices could be converted into digraphs from which the mean edit distances could be calculated. By continuously computing the edit distance between graphs and storing the appropriate results, plots can be created of edit distance vs time. The second algorithm concerning the interactions matched all the cell phone location data to the POIs. To make this process faster, a city was split into bounding boxes so that the total search area was reduced by a factor of 10.

D. Task Results

The first algorithm computes mean edit distance for each of the hourly intervals between one hour to four hours and creates separate graphs for each. Fig. 1 shows one of the graphs yielded from the first analysis where the goal was to find mean edit distance across time intervals of four hours.

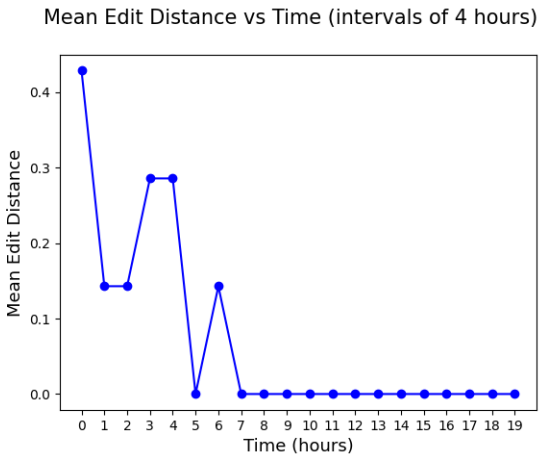

Fig. 1. Graph of mean edit distance with an time intervals of 4 hours.

By using these graphs, one can gauge both small and large trends in movement depending on the time interval used. From Figure 1, it can be deduced that the most changes in interaction occurred between hour 0 and 4 of the data as the highest mean edit distance indicated by the graph is on hour 0.

Next, Table I shows a sample of the output of the interaction algorithm. There's two IDs for every interaction which is matched to a POI identified by its type and unique id.

Unfortunately, the algorithm was not quick enough to run for all of the interaction data provided as search times grow as inputs and POIs increase. However, if optimized, the algorithm can be key in discovering what POIs are most densely packed at a given time in the day on a continuous rolling basis.

TABLE I
RESULT OF INTERACTION ALGORITHM

Index	ID 1	ID 2	POI Name	POI Number
0	"9E5E369..."	"402B0171..."	Restaurant	3
1	"DBAD261D.."	"891210E0..."	Restaurant	9
...				

III. TASK 2 - PREDICTING ENERGY CONSUMPTION

A. Task Motivation

Modelling energy can be used to understand how energy consumption varies by time and therefore can help predict energy demand. This can be particularly useful for areas such as individual buildings to localities as they plan ahead for patterned energy spikes. This information can be used to better allocate resources over time in order to reduce energy inefficiencies. For example, if it can be predicted that the energy demand will dip at certain hours, energy input and production can be reduced during that period of time which ideally could help reduce energy cost.

B. Task Description

For this particular task, the area in question was Rice Hall, a building at UVa. Meters at Rice Hall monitor many aspects of energy consumption, such as Voltage, Power, etc. The goal was to create a model using machine learning that could accurately process total energy consumption data and predict the energy fluctuations using time series data.

C. Task Method

Since the data from Rice Hall included a multitude of data across several files, certain processing needed to be done. This was done through 3 Python programs. One filtered the data relevant to the task such as energy values and timestamps. The next program sorted the data by days separate the data into 2 hour time intervals. The last program combined all the sorted files into one large file that would serve as the input for the machine learning model.

For this task, the Long short-term memory (LSTM) recurring neural network (RNN) architecture was used. The benefit of LSTM is that its architecture is set up to overcome vanishing gradient problem, which is an issue that regular neural networks face where the network fails to persist useful data. It follows that the LSTM architecture can be key for modelling time series data such as energy consumption data.

The Keras machine learning library was selected for this task due to its ease of use and extensive documentation on machine learning tasks. The algorithm implemented the a modified version of Brownlee's tutorial as it gave an in-depth breakdown of a a simple time series model [1]. One issue that was encountered was that LSTM requires extensive hyperparameter tuning. Hyperparameters are needed to adjusted manually in order to control the model's learning process. Doing test trials was a time consuming process so automation was used. The automation was made possible by the Adaptive Experimentation (Ax) Platform, which contains API for running experimental runs and picking the run with the best outcome. For this task, another resource by Lianne and Justin was key in learning hyperparameter tuning with the Ax Platform [2]. By appropriately incorporating the automation process from the aforementioned resource, the following hyperparameters in Table II were found.

TABLE II
HYPERPARAMETER TABLE

Hyperparameter	Value
Learning Rate	0.05407427952920797
Dropout Rate	0.011616500171118609
Neurons per Layer	199
Momentum	0.21804596483707428
Batch Size	128
Activation	'relu'
Optimizer	'sgd'

Then, these hyperparameters were used to create the model's architecture.

D. Task Results

The data was split into one training set, and one test set. which would test the trained model on novel values. The total data set spanned from 0 to 936 hours, and the training was done on hours 0 to 800, and the testing was done on hours 800 to 936. The model was then run for 101 epochs because more epochs yielded marginal reductions in loss rate. The final result on the test data set can be seen in Fig. 2.

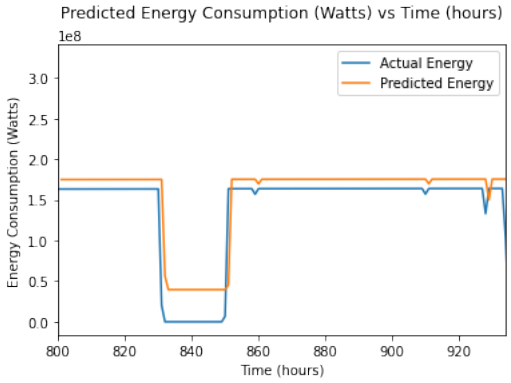

Fig. 2. Output of the ML model on the test data.

The final loss rate in terms of mean squared error was 0.0103. When the Mean Absolute Percentage Error (MAPE) was calculated, the initial result was inconclusive due to the original data having 0s leading to a divide by zero error. To account for this, the 0s were normalized to the value calculated by the model and the resulting MAPE was roughly 18.79. Both these metrics suggest that the model was close to the actual values but not was not completely accurate. This is a desirable outcome because the model needs to be generalizable to energy consumption data beyond the current dataset in order to be useful for future predictions.

Conclusion

This capstone research produced two tools which had different purposes yet were similar in the utility they provide. Both attempt to help gain insights about a particular location where the first can be useful on a fully local scale, while the second task can be beneficial for buildings such as Rice Hall. Further work can make these tools more robust and scalable, however they still illustrate how simple technological tools can helping process large sums of data in powerful ways.

Further Work

With these tasks, it is important to note that there is much work needed in order to be able to implement the tools on the industry level. The first task needs to scale with large amounts of data. This can be done by implementing the algorithm in Spark, a framework that can take advantage of distributed file systems such as Hadoop File System (HDFS). By using Spark's MapReduce, the workload can be parallelized across many nodes and thus the total computation time can be reduced.

For the second task, it is evident that the model predicted the energy consumption on the test data set very well. However it can be the case that the model is overfitting on the data. In order to avoid this possibility, more thorough experimentation can be done to create a model that can be better generalized for this use case. This also requires a consistent and thorough handling of zero values. Lastly, this computation is also very time intensive - some trials can take over 10 minutes. By adding GPU support, computation time can be reduced.

Acknowledgment

This research was conducted under the supervision of Professor Haiying Shen and her graduate assistant Shohaib Mahmud. In undertaking this 3-credit Capstone Research course, I learned about technologies such as Spark, HDFS, and LSTM, and also used ones I was experienced in such as Python.

References

[1] Brownlee, J. (2020, August 28). *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. Machine Learning Mastery*. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

[2] Lianne & Justin (2020, March 15). *Hyperparameter Tuning with Python: Keras Step-by-Step Guide Why and How to use with an example of Keras*. Just into Data. https://www.justintodata.com/hyperparameter-tuning-with-python-keras-guide/