Reproducing PREDICTING FLOOR LEVEL FOR 911 CALLS WITH NEURAL NETWORKS AND SMARTPHONE SENSOR DATA

Pierre He
Department of Computer Science
McGill University
pierre.he@mail.mcgill.ca

Maxence Hull
Department of Computer Science
McGill University
maxence.hull@mail.mcgill.ca

1 Paper Summary

1.1 Motivation

The paper proposes an approach to predict floor levels. The motivation of this work is to have a faster and more efficient response to a 911 call. With an algorithm that can efficiently predict floor levels just based on data collected by a smartphone, it is easy to see that it would greatly help officers responding to a victim's call. Especially in cities with buildings with numerous floors.

1.2 Proposed approach

1.2.1 Creating a dataset

In order to correctly predict the floor level of a device, one must create a dataset to train some machine learning models and ultimately tests its system. Since there is no existing dataset with that information, the authors made one. They created a smartphone application and collected data throughout New-York City streets and buildings. More precisely, they collected the followings metrics: barometric pressure, GPS vertical accuracy, GPS horizontal accuracy, GPS Speed, RSSI level and magnetic field. Each trial has been performed following the same procedure: start outside the building, then walk into it and rise n floors. When entering a building, he notifies the application that he is indoor. At the end of the experiment, it manually entered the final floor level. These trials have been performed on many different buildings (different heights, floors configuration ...) Overall, there are 96 trials, each of them containing about 120 measurements, one per second.

1.2.2 Indoor/outdoor classification

When a floor prediction is requested by a user, the first step is to find the moment when the user first entered the building. This information is important since it will be used to predict user current floor using barometric pressure difference. To resolve this prediction problem, authors used many different machine learning models: an LSTM, a feed-forward neural network, an SVM model, a random forest model and a logistic regression model. In order to make valid predictions, authors used a time window of length d around a given measurement. For example, to make a prediction at time t with $d = 3$, the model input will be a concatenation of three measurements x_{t-1}, x_t and x_{t+1}. The output will be 0 if outside, 1 if inside.

1.2.3 Detecting an indoor/outdoor transition

In the end, the previous models output a series of 0 and 1's (one for each measure). One must be able to find an indoor-outdoor transition even if predictions are not 100% accurate. It is the goal of the transition detector: classify a subsequence s of predictions as a transition or not. To do this,

Submitted as course project for IFT6135 H2018 Representation Learning

authors used two vector masks: $V_1 = [1, 1, 1, 1, 0, 0, 0, 0]$ and $V_2 = [0, 0, 0, 0, 1, 1, 1, 1]$. Convolving them around the vector of prediction for each possible sub-sequence and using the Jaccard distance, authors were able to find indoor/outdoor transitions. Jaccard distance measures dissimilarity between two vectors: if the dissimilarity between a sub-sequence s and one of the two vector masks exceeds a threshold, this subsequence is tagged as a transition.

$$Jaccard(s_j, V_i) = \frac{|s_i \cap V_j|}{|s_i| + |V_j| - |s_i \cap V_j|}$$

Then the transition indexes are set to be the mean of the first index of consecutive tagged subsequences. (It seems strange but it does find the correct transition indexes).

1.2.4 Estimating an absolute floor level

The final step is to determine the vertical offset between the last transition ($t_{transition}$) and the current device location, in meters. Retrieving the last transition is straightforward given the previous algorithm. The algorithm will select the minimal barometric pressure as p_0 within a 15 seconds window around $t_{transition}$ due to GPS latency. The device vertical height is calculated using the following formula:

$$m_\Delta = 44330(1 - (\frac{p_{current}}{p_0})^{\frac{1}{5.255}})$$

To get the absolute floor level, m_Δ is divided by the height of one floor (carpet to carpet). Authors used an average measure if this information is unknown ($floor\_height_{average} = 3.63$). To get a more precise measurement for every building, they proposed a clustering algorithm. Since they only tested this algorithm in a single building as a proof of concept, they used some prior knowledge to get final results (for example, 3.8 meters for the Rockefeller Center). We didn't reproduce the clustering algorithm.

2 Experiments

2.1 Motivation for the experiments

Using an LSTM seems a good choice of model for such a problem as we are dealing with time-series data. He also compares it to multiple different baselines algorithm to see whether it is a good choice or not.
The author used a grid search method to select the different hyperparemeters for the different models such as learning rate, number of neurons in the layers, dropout rate etc... For each parameter configuration, he performed around a 100 random restarts and saved the best model overall. For each model, we used his best parameters.

2.2 Reproducing the main results

The results from the paper were not reproducible (at least not the 100% accuracy sometimes claimed by the author). In this section we will present the results we obtained and compare them to the results from the paper. We will explain where the difference in performance comes from in the last section. The author made available his code as well as the different datasets (training set and validation set) he collected.
First, let's reproduce the first part of the paper: the I/O classification problem. For the LSTM, authors used a three layers architecture:

- a first LSTM layer with 5 neurons and a dropout rate of 0.2
- a second LSTM layer with 50 neurons and a dropout rate of 0.2
- a final feed-forward layer with a single neuron and sigmoid as an activation function

Concerning the feedforward neural network, it is composed of:

- A layer of 15 neurons and ReLU as an activation function
- A batch normalization layer

- A layer of 18 neurons with a dropout rate of 0.7 and ReLU as an activation function
- A batch normalization layer
- A layer with a single neuron and sigmoid as an activation function

All models were trained using a batch size of 128, binary cross-entropy as the cost function and Adam as the optimizer.

	Paper results		Our results	
Model	Validation Accuracy	Test Accuracy	Validation Accuracy	Test Accuracy
LSTM	0.923	0.903	0.952	0.897
FFW	0.954	0.903	0.941	0.895
SVM	0.956	0.876	0.947	0.855
RF	0.974	0.845	0.997	0.830
LR	0.921	0.676	0.90	0.718

For the absolute floor level prediction, we mostly used the code published by the authors (available through GitHub).

	Paper results			Our results		
Model	Exact Floor	± 1 floor	> 1 floor	Exact Floor	± 1 floor	> 1 floor
LSTM	1 | 0.65	0 | 0.33	0 | 0.016	1 | 0.63	0 | 0.36	0 | 0
FFW	1 | 0.65	0 | 0.33	0 | 0.016	1 | 0.63	0 | 0.36	0 | 0
SVM	1 | 0.65	0 | 0.33	0 | 0.016	0.97 | 0.62	0.015 | 0.36	0.015 | 0.02
RF	1 | 0.65	0 | 0.33	0 | 0.016	0.97 | 0.63	0.03 | 0.36	0 | 0
LR	1 | 0.65	0 | 0.33	0 |0.016	0.98 | 0.63	0 | 0.36	0 | 0

For each result, the left side represents the accuracy with building knowledge, the right side without.

We will explain in the next section the reasons why we didn't achieve the same results as in the paper. But overall, the results are very satisfactory as even without building knowledge, the algorithm achieves a precision of $\pm$ 1 floor 99% of the time.

3 Discussion

3.1 Non reproducibility of the results

The first thing that seemed strange was that despite very different accuracies with the different I/O (in out) models, the latter part showed exactly the same performances for all the different models (except the HMM, but we didn't test it). When using the code he made available, we obtained also the exact same results as presented in his paper. So we are pretty sure the code he made available is the one he used to evaluate the final performances he put into his paper.
However, we found a mistake in the Jaccard distance he uses. In the paper, he describes a Jaccard 'similarity'. But the Jaccard he uses (from the scipy library) is a distance measure: the higher the distance, the higher the dissimilarity in number of bits between two vectors. Moreover, through a hyperparameter search, he sets the threshold for the dissimilarity at 0.4. Any subsequence with a Jaccard dissimilarity of above 0.4 with any of the two masks (see section 1.2.3) is tagged as a match. The final transition index is the mean of the first index of every consecutive matched subsequence. However, we found empirically (and it is easy to prove) that any random sequence of 1 and 0 will have a Jaccard distance of at least 0.5 with one of the two masks. (e.g. it can be 0.5 and 0.5, or 0.6 and 0.4. It follows this pattern, the sum is always 1, and the increment is 0.1). So setting a minimum threshold of 0.4 makes the algorithm find matches at every position of the total sequence. Then when it does the mean of all the indexes, it always outputs the middle index of the sequence (actually, it is middle - 5).
He got such good results because luckily for him, his dataset has been collected in such a way. For almost all the data sequences collected, the middle of the sequence is an outdoor data point. So taking it as a transition point to then look for the reference ground pressure would yield a correct reference pressure. So the following steps would also yield a correct answer. For the few test examples where the middle of the sequence is an indoor point, this point is never far from the last outdoor point. So when looking in a 15 second window before the predicted I/O transition, the device was most

probably still at ground level (it takes time for the user collecting the data to start going up the floors...). So once again, in this case the algorithms despite not having the desired behavior outputs a correct p_0.

All in all, because of the structure of the dataset, with a wrong algorithm he ends up with excellent results. However, as shown in the last section, we still got near perfect results with a corrected algorithm. To correct it, we set the Jaccard maximum dissimilarity threshold at 0.4. (Before it was a minimum threshold). With this, the algorithm finds the correct I/O transition point.

3.2 Ground level pressure search window size

To obtain m_Δ once he found the transition point from outdoor to indoor in the data sequence, he needs the ground level pressure p_0. p_0 is set to be the lowest pressure measured in a 15 second window around the transition point (15 data point window in the sequence). This 15 seconds window is meant to take into account the 15 second lag that the smartphone's GPS system needs to accurately recalculate his precision once it enters a building. However this raises two questions.
The first one is the size of this "look-around" window. His whole dataset consists of sequences of data that are on average of length 100 to 150. A 15 second window is not a negligible length. So the lowest pressure measured in this 15 seconds window is very likely to be a ground pressure in this case.
The second is why even incorporate this into the system. The whole point of the network in the first step is to learn when in the sequence the user entered a building despite non obvious measurements. Taking the lowest pressure in a window around the predicted transition point gets rid of the need of a very accurate algorithm in the first step.

3.3 Adding a deterministic baseline

It would have been insightful to see if we can achieve good results with a completely deterministic algorithm and include it in the paper's result as a baseline. From visual inspection of the dataset, just based on the GPS horizontal accuracy, it seems easy to determine when the transition from indoor to outdoor occurred. While outdoor, the GPS horizontal accuracy is around 30. While indoor, it is around 200 and it takes roughly 10 seconds for this accuracy to increase from 30 to 200. This is the lag mentioned before. In this case, an algorithm that would look for the first point where the accuracy reaches 200 then takes the lowest pressure up to 15 seconds before this point (cf section 3.2) could probably find the correct ground pressure reference p_0. Once it found p_0, we just need to follow the next steps described in the paper, which have nothing to do with machine learning, to determine the correct floor from which a call has been issued.

3.4 Feature selection

As discussed before, having 100% accuracy in the first step of the system might not be necessary to ensure almost perfect performance (with building knowledge) in the later steps. We tried using a subset of the features proposed by the paper. Getting rid of the magnetic field in the features only reduced the accuracy of the I/O network by less than 1%. We did not test the full system on these reduced features but for the reasons explained before, this might not affect the overall performance in the slightest.

3.5 Shortcomings of the dataset

In his dataset, all of the different sequences of data are similar. They begin with data collected outdoor (sequence of 0s). Then the person gets into a building and go up the floors (sequence of 1s). There is no data from odd behavior. It would have been useful to include someone that repeatedly enters then leaves a building. Moreover, as described before, despite having an algorithm that doesn't behave as intended, it still manages to find the correct ground level pressure. This comes from the fact that the middle of each data sequence usually corresponds to an outdoor point (ground level) or is an indoor point that happens not long after the person entered the building (so still at ground level). Only one or two sequences include a very long sequence of 1s. So there is an underlying pattern in his dataset that doesn't necessarily reflects a real situation application. On the contrary, a real case situation would probably be someone who stayed in the building for a long period of time already (maybe 20 minutes

or even hours) before issuing a call to 911. His dataset only consists of sequences of 2-3 minutes of data collection usually, the maximum being 4 minutes (1 measure every second). However, we do understand that the author built the dataset himself and that a dataset that reflects real case situations would have been too time consuming to collect.

3.6 Performance discussion

Overall, the algorithm achieves great accuracy. Without building knowledge, it has a precision of $\pm$ 1 floor 99% of the time on the test set. And with building knowledge, the baselines (SVM, Random Forest and Logistic Regression) achieves perfect accuracy 99% of the time and the Neural Network algorithms (LSTM and basic Feed Forward MLP) achieves 100% accuracy. This raises the question of whether there is a need to use a complicated neural network that requires more hyperparameter tuning to solve the problem. A simple SVM that requires almost no memory to store the model (feature space of dimension 5) and very low computational requirements for prediction seems enough. If one wants to deploy this system on phones, this solution appears to be more suitable.

4 Conclusion

Overall, the reasoning behind this paper is interesting and could lead to great performance. However, some technical mistakes prevent us to entirely be sure that the solution could work in the real world. Testing the solution again with an enhanced dataset (more variety in the samples) seems to be a necessary first step before any further improvements. One can also wonder if a complex machine learning algorithm such as an LSTM is necessary for the first task: much simpler models works almost as well. However, as the authors mentioned in their conclusion, this solution is only a first step to solve this problem: ideally, a single model could directly predict the floor level. In that case, an LSTM (which is capable of learning long-term dependencies) seems like a reasonable choice.