GEORGIA INSTITUTE OF TECHNOLOGY

UNDERGRADUATE THESIS

Trajectory Prediction for Nonuniform Geospatial Mobile Device Data

Author:
Sara BRANHAM

Supervisor:
Dr. Mark DAVENPORT

A thesis submitted in fulfillment of the requirements for the Degree Bachelor of Science

in the

Center for Machine Learning
College of Computing

July 12, 2020

Abstract

Sara BRANHAM

Trajectory Prediction for Nonuniform Geospatial Mobile Device Data

GPS collection has become increasingly popular with the rise of mobile devices and applications. This data is collected for an incredibly wide range of reasons, including applications like Google Maps providing directions, weather applications providing weather predictions, Uber or Lyft providing transportation, or service providers seeking to better understand their users. While GPS data has many uses in our society, there exist enormous obstacles surrounding long term data collection, namely how to acquire uniformly sampled device data. We propose using Transformers and GRUs with added Attention to extract long term habits for individual users in the context of nonuniform GPS data. These models are traditionally used for Neural Machine Translation (NMT), so they are well equipped for nonuniform problem spaces.

Introduction

Mobile device location tracking has become increasingly common in recent years. The rise of applications like Uber, Find My Friends, and Google Maps has resulted in the collection of human mobility data that is unprecedented in scope and quality. A natural question that arises with an increase in geospatial data is how to understand, analyze, and use the information to predict future behavior. However, data quality is one of the primary issues that emerges when considering mobile devices; it is exceedingly difficult to extract perfectly sampled, error-free GPS data. This complexity makes it challenging for applications to effectively utilize geospatial data.

Some of the earliest solutions to the problem trajectory prediction lie in different contexts, but they helped generate models that paved the way for mobile device GPS tracking. One of the earliest areas where trajectory projection was implemented was in Air Traffic Control systems. Originally these systems relied on physical models that made a variety of assumptions about how the underlying physics of aircraft motion behaved. There has been a recent shift, however, towards data-driven models that instead rely on a variety of measurements to predict aircraft behavior [2, 5, 8]. The transition to data-driven models also makes sense when considering trajectory prediction in the context of mobile devices. Aircrafts have flight plans, constant communication with surrounding airports, auxiliary weather information, and much more, whereas with mobile devices it is difficult to collect that much information and it is important to consider countless other factors. For instance, a user may want to avoid traffic or take a more scenic route. The differences in the type of data necessary to make accurate predictions between these contexts facilitated the need for new models to be considered.

While mobile device trajectory prediction is not a new issue, very few existing studies address the core question of how to process nonuniform, sparsely sampled datasets to make meaningful conclusions. Many of the datasets used for mobile device trajectory predictions were experimentally gathered, meaning authors develop models that depend on having a near-perfect sampling rate and minimal error. This experimental accuracy can only be guaranteed if users agree to have their location sampled every few seconds for weeks, months, or years, at a time [15, 4, 7]. Studies that explicitly claim to address sparse trajectory data often analyze users on a mass scale, by looking at traffic flows to interpolate paths, or by taking uniform sparse samples, for example looking at data sampled every 5 minutes instead of every 3 seconds [10, 9, 4]. This clearly leaves a huge gap in the field of trajectory prediction because there are no well established solutions for dealing with nonuniform sparsely sampled data.

One promising technique that does address the issue of nonuniform sparsely sampled trajectory data borrows from sequential Natural Language Processing (NLP) models. Yang Li and José Moura have laid the groundwork for using Transformers to predict taxi demand in New York with sparse geospatial datasets [12]. Because the Transformer architecture is better at capturing long-range temporal relationships, when compared to other sequential models - namely Recurrent Neural Networks (RNNs) and Long-Short-Term Memory (LSTM) Networks - we believe this model will generalize to our research question of processing sparse nonuniform GPS data for individual users [1, 17]. Transformers, although a relatively new architecture, have made a profound impact in the scope of NLP and there is reason to believe these improvements are also applicable to time-series analysis [18]. We hypothesize that this architecture will not fall into the common pitfalls of standard trajectory models when processing nonuniform data because it treats all node-to-node distances as constant, whereas alternative models take a non-constant form of distance into account, whether temporal or geometric [15, 4, 7]. This means that while an RNN or an LSTM may struggle to find patterns between behaviors that are far apart, for example a user taking a specific route to work every other week, a Transformer would have much more success because it doesn't acknowledge that these actions are weeks apart. If the Transformer is successful, it will have profound implications on geospatial data collection practices, because companies will not need as much data to make as accurate predictions.

Background

In order to understand the nature of using geospatial data to track and predict individual trajectories, it is important to evaluate the question in its original context, aircraft trajectory prediction. For example, in 1999 Chatterji combined a Kalman filter, a Proportional Integral Derivative (PID) controller, and a least-squares low pass to generate a short-term trajectory prediction that relied on aircraft angle, speed, and weather conditions [5]. While this approach was cutting edge for the time, Kalman filters only work for linear models and the need for more robust models paved the way for data-driven approaches to take over. Fernández, et al, developed Data-driven Aircraft Trajectory Prediction Research (DART), which used a combination of individual and collaborative machine learning approaches to generate predictions that were independent of traditional, flawed, kinematic assumptions [8]. Similarly, Ayhan and Samet utilized Hidden Markov Models (HMMs) to incorporate historical trajectory data and weather analyses into aircraft trajectory predictions [2]. This transition to data-driven models meant that trajectory predictions no longer relied on physical assumptions and resulted in more accurate predictions for Air Traffic Management systems, consequently improving safety, saving fuel, and decreasing travel time [8, 2]. While aircraft trajectory models are foundational, they often have ideal data collection, which features samples minutes apart and features like GPS coordinates, speed, plane heading, weather conditions, and potentially much more [8]. Conversely, mobile device data typically consists of GPS coordinates, velocity, although measuring velocity is often difficult, and very rarely weather.

To combat the challenges that arise with mobile device data collection, a common tactic is to use experimental device data, which can take a variety of forms, for example, users agreeing to download specific apps or wear a device that records their location for an amount of time [4]. For instance, Newson, Paul, and Krumm, as well as Barnett and Onnela, defined trajectory prediction approaches that excel in experimental circumstances [15, 4]. Newson, Paul, and Krumm used HMMs to develop an underlying strategy for predicting what state transitions make sense for the users before considering potentially error-prone data points. However, the accuracy of this algorithm drops by over 20% once the samples grow to be more than 3 minutes apart [15]. Similarly, Barnett and Onnela used a dataset where an application tracked their users for 2 weeks with a sampling frequency of 1 second. While their approach was successful when extrapolating up to a sampling frequency of 10 minutes, it is still dependent on samples being a uniform distance apart, which is difficult to guarantee [4]. These approaches work quite well under the circumstances that they were designed, but are not transferable when it comes to messier nonuniform datasets, like the ones we are dealing with.

Other authors have used clustering-based methodologies to predict trajectories by using datasets not centered on car travel for an individual user. For example, Nikhil and Morris generated a Convolutional Neural Network (CNN) that modeled pedestrian trajectories. However, they focused on short term interpolation and collaborative components of trajectories when pedestrians interact with one another, whereas our problem space is concerned with long term trajectory predictions and does not incorporate collaboration into the model [16]. Mao, et al developed a 3 pronged approach to track hurricane trajectory data by utilizing Density-based spatial clustering of applications with noise (DBSCAN). This study produced an effective preprocessing approach that partitions and then aligns sub trajectories, making them easier to process [14]. While their preprocessing technique is effective and generalizable to processing our own geospatial data, clustering with DBSCAN is dependent on having densely sampled data to extract routes. This assumption is not always the case in nonuniform datasets, as we may have a densely sampled beginning and end of a path, and need to infer what happened in the middle instead of treating them as separate. These studies are helpful to understand the mathematical foundations of trajectory prediction, although the approaches are not transferable to our problem space.

There are also some methodologies that rest on assumptions that do not necessarily apply to our users. The paper by Eisner et al, for instance, outlines a way to partition trajectories into piecewise paths and perform a variation of Dijkstra's shortest path, but the authors also assume that people will take the shortest path between two points [7]. Realistically a number of factors may impact why someone may not take the optimal path on their way home from work, which means this assumption would lead us to wrong conclusions about their routes. Another approach by Ahmed, et al uses GPS snippets to reconstruct trajectories in major cities, but they relied heavily on having accurate speed measurements to help their interpolation [10]. Conversely, our dataset does not have reliable speed measurements, so we can only extract conclusions from GPS updates.

Our solution to this issue is heavily defined by our dataset and because of this, we plan to take a new approach by reworking sequential models traditionally used in NLP, in combination with preprocessing techniques that have already been defined. Yang Li and José Moura have already used Transformers to predict taxi demand in New York by using geospatial datasets [12, 18]. This architecture will help us learn and extract long term spatial dependencies, and therefore reconstruct sparse paths that users take. Other authors have used alternative sequential models for trajectory predictions, although those were primarily focused on smaller-scale predictions, for example, what a user would do over the course of a few seconds [1, 17]. Additionally, an LSTM approach would fail to capture long term relationships as effectively as the Transformer, and we want to analyze spatial dependencies on the scale of months at a time. We hope to emulate the work of Li and Moura and incorporate preprocessing techniques similar to Binh Han, who executed a framework to mine complicated mobile device trajectories [9]. Her dissertation details a way to combine piecewise clusters of trajectory fragments with traffic flows to generate most likely paths. Our approach will differ from this because instead of using information from other users to interpolate a route, we will instead rely on the historical data of a unique user. By pulling techniques from these three authors, we hope to combine historical trajectory prediction algorithms with sequential models that are newly applied to the field of time-series predictions.

Overall our work is combining elements from previous algorithms that have been heavily researched and developed. The primary differentiator is the nature of our data, which is non-uniformly sampled, and collected over a period of 3 months. This difference facilitates the need to generate a new approach that is not reliant on uniform sampling. Should we succeed, there is a wide variety of commercial applications that will benefit from needing to gather less information to generate useful conclusions about users' behavior.

Methodology

Preprocessing:

We began by selecting a user with clearly discernible behaviors. For instance, the path of User 010 is featured in Figure 1 and it spreads throughout China, whereas in Figure 2 we can see that User 000 has pretty consistent habits. Thus User 000 is a good candidate to train our model on.

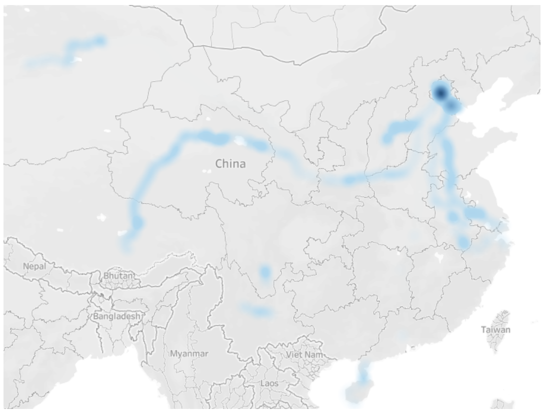

FIGURE 1: User ID 010

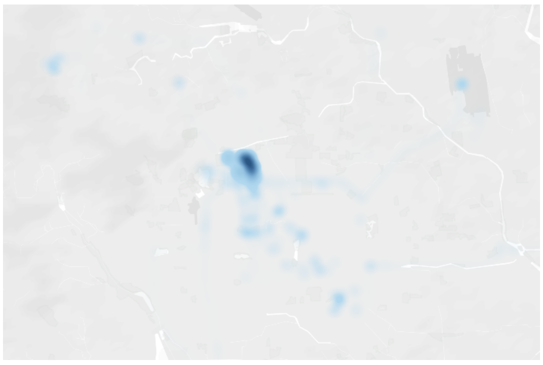

FIGURE 2: User ID 000

We then proceeded to isolate individual days where User 000 took similar routes to build our train and test sets.

Our second course of action was finding a way to localize and subdivide our routes. We decided on the gridding procedure featured in Figure 3, where the limits of the rectangle were the maximum latitude and longitude and minimum latitude and longitude of a given sequence. We were then able to use a 2D distance metric in our loss function.

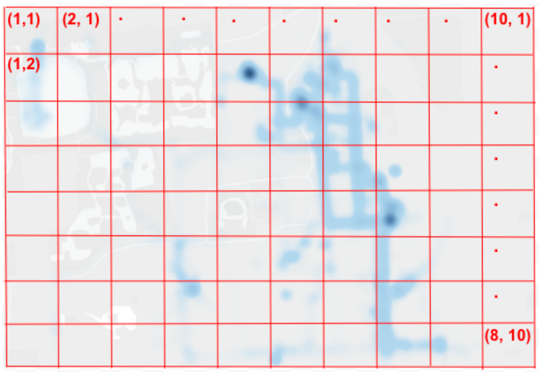

FIGURE 3: Second Gridding Approach

Finally, we needed to pad our sequences to standardize our sequence length. We are using the GeoLife dataset as an example of uniformly collected GPS data. Microsoft collected the GPS information of 182 users in China over the span of 3 years, with a sampling frequency of 3-5 seconds [19, 20]. However, this dataset is separated into "trajectories" for each individual user, and there are a total of 17,621 trajectories. Once we isolated our user and the specific days we wanted to train on, we standardized the sequence lengths by padding matrices filled with -1. These matrices were later ignored through the help of a masking layer in each of our models.

Attention:

Encoder-decoder architectures and attention mechanisms were developed in the context of Neural Machine Translation (NMT), where the encoder would extract and condense helpful information about the input sentence into a context vector, which the decoder would use to generate the transformed output - in many cases a translated sentence. Cho et al. incorporated an additive attention mechanism to the decoder so that the decoder could adaptively decode information, which meant there was no more need to fit an encoded sentence into a fixed length vector [3].

Luong et al. expanded the attention mechanism by proposing global and local attention [13]. Global attention considers every hidden state in the encoder before deriving the context vector, whereas local attention only considers a small window of hidden states. In global attention the context vector, $\mathbf{c_t}$, represents a weighted average of the alignment scores, which compare the target hidden state $\mathbf{h_t}$ to the source hidden state $\bar{\mathbf{h}}_\mathbf{s}$. Conversely in the local attention model, $\mathbf{c_t}$ is a weighted average over the set hidden states within the chosen window. These values are then combined into an attentional hidden state, $\tilde{\mathbf{h}}_\mathbf{t}$, that is input into the following time step, where $\tilde{\mathbf{h}}_\mathbf{t}$ is:

$$\tilde{\mathbf{h}}_\mathbf{t} = \tanh \mathbf{W_c}[\mathbf{c_t}; \mathbf{h_t}] \tag{1}$$

Vaswani et al. adapted attention mechanisms further to develop the Transformer, namely self attention and multi-head attention. [18]. Self attention computes a representation of a sequence by analyzing different positions within one sequence. The authors introduce dot product attention, featured in equation 2.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt(n)} V\right) \tag{2}$$

Where (K, V) is the encoded representation of the input, presented as a key-value pair, and Q is achieved as a result of the decoder compressing the previous output into a query.

Multi-head attention behaves similarly, although it performs the self attention calculations in parallel by projecting the queries, keys, and values across different dimensions and then concatenating these values.

$$\mathrm{MHA}(Q, K, V) = \mathrm{concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h])W^O \tag{3}$$

$$\mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{4}$$

where W_i^Q, W_i^K, W_i^V, and W^O are learned projection matrices.

Model #1: GRU with Attention

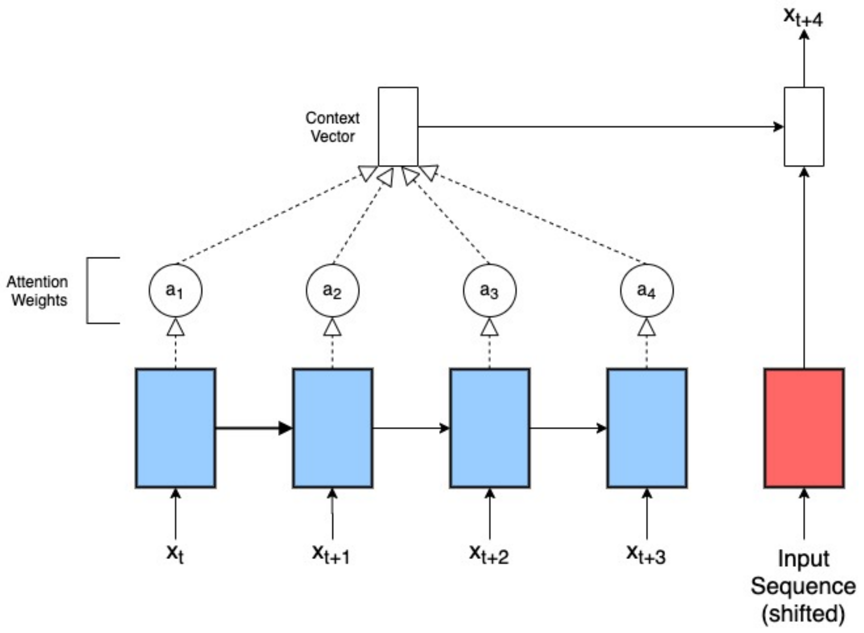

FIGURE 4: GRU with Attention Architecture

Our first model is an encoder-decoder architecture with Gated Recurrent Unit (GRU) and global attention. GRUs are regarded as a simplified LSTM, which makes them quicker to train, and they are better at retaining long term memory when compared to RNNs. We have included global attention layers to attend over every layer in the encoder, similar to the work of Luong, et al [13]. We preprocess our data by developing an $N \times N$ grid for each individual user and then transforming the latitude and longitude into the grid system, which reduces prediction that arises with error with raw latitude and longitude values. We then feed an encoding of the grid value and the datetime into the encoder for every step in our sequence. The encoder utilizes global attention to generate an attentional hidden state, which is featured in equation 1, and calculate a context vector that is then input into our decoder. The decoder takes the context vector from the encoder as well as a timestep, t, that we manually assign as an input. The decoder then outputs one grid location that is t steps in the future from where our input sequence ended. It is important to note that at this stage our decoder is assuming a uniform sampling, although we plan to modify that in later stages of our workflow. Our hypothesized solution will be to interpolate our nonuniform sequence into a uniform sequence by adding null entries at even timesteps, and then incorporating a masking into our encoder that ignores any null values during runtime.

Model #2: Transformer

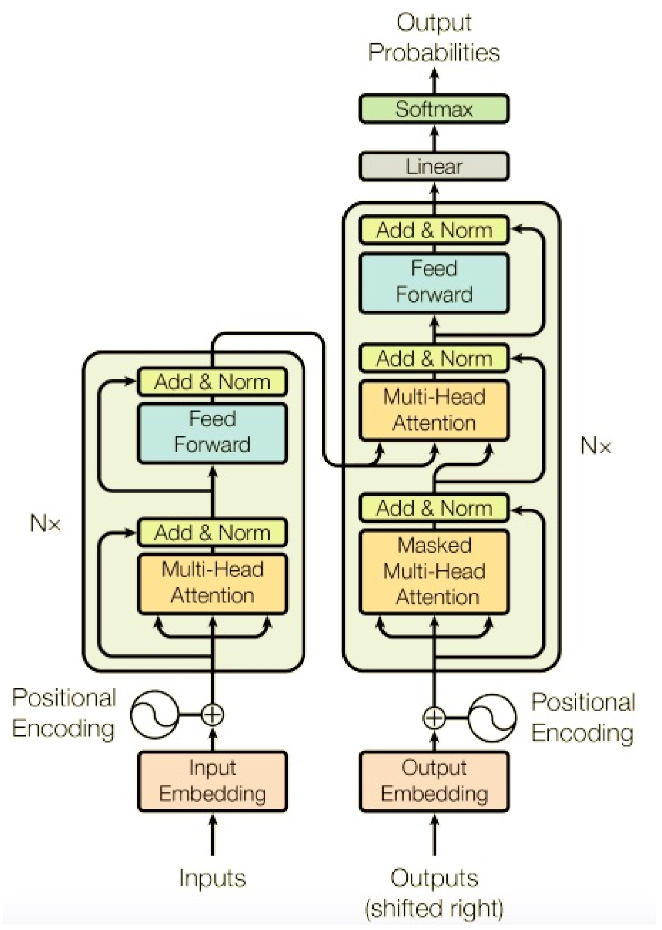

FIGURE 5: Transformer Architecture (Source: Vaswani et. al [18])

To implement our Transformer we are following the architecture by Vaswani et al., which is featured in Figure 5 [18]. The Transformer also utilizes an encoder decoder architecture with added residual connections and attention layers. We use the same preprocessing technique and input a sequence of grid locations concatenated with datetime values into our model to develop our input embedding. We also input a positional encoding so that the Transformer can keep track of relative positions as it attends over our sequence. Our positional encodings are combined with our input as we move towards the encoder. Before normalizing each time we re-input out initial sequence, which forms the residual connection. We do this to avoid losing the relationships from our initial sequence over the course of a very deep network. The encoder applies multi-head attention to the input sequence, and then passes the sequence through a feed forward network. The feed forward network applies two linear transformations and a ReLU activation. Then the decoder takes the shifted sequence, another positional encoding, as well as our encoder output as an input. The shifted sequence is the same as our input sequence, although it won't consider the first t timesteps - this is to prevent our model from simply memorizing the decoder input and instead generate a prediction. Then the decoder performs multi-head attention over the shifted decoder input as well as the output of the encoder before going through the feed forward and outputting the future trajectory of our user.

Experimental Approach

The goal of our project is to show that models almost exclusively used in NLP problems can be applied in time series contexts and outperform the models traditionally seen in time series forecasting. In order to prove this, we have set up multiple iterations, where each iteration runs the same models over different datasets. The first iteration utilizes the Microsoft GeoLife dataset as-is [19, 20]. We will then run our two benchmark models, the RNN and LSTM, over the GeoLife dataset. This is to establish baseline results using industry-standard models. We then migrate to our newer models - the Transformer and the GRU with Attention. After processing the dataset we will determine the accuracy of each model by calculating the Mean Absolute Error (MAE) between our predicted location and the ground truth. This serves as a euclidean distance measurement between our predicted and actual GPS locations. The second iteration is to repeat running these 4 models over the GeoLife dataset, however, we will modify the dataset to have a sparse uniform sampling. This means data points will be 5-10 minutes apart instead of 3-5 seconds. Our third iteration is similar, but now we non-uniformly sample the GeoLife data by taking points anywhere from 2 minutes to 5 hours apart. Finally, we repeat the same process but with our Mogean dataset. This dataset features individualized GPS data aggregated across multiple apps on the individual's mobile device. The Mogean dataset is our example of a standard commercial dataset, where traditional models like the RNN and LSTM fail, so if the Transformer and GRU succeed then we know we have developed something that surpasses the industry standard for GPS forecasting.

Results

I had intended on using tensorboard to track the fluctuations in the mean absolute error across the different models and different datasets. Unfortunately, due to delays caused by COVID-19 as well as a main water line breaking on Georgia Tech's campus that resulted in the loss of pressure in the cooling system for our data center, none of this was possible.

In lieu of results I will detail my issues with PACE and the lab GPU. Initially I planned on training using the PACE Clusters at Georgia Tech to quickly train my models. I immediately had issues defining a custom conda environment and found that I did not have write access to any existing environments. Once I was able to define my environment I learned that there was a 5GB cap on any files in my home directory, which was largely taken up by TensorFlow and meant that my environment could not include any of the other Python libraries I was reliant on. After discovering a workaround for defining an environment that had the appropriate capacity, I quickly hit the 5GB capacity while training my model. I tried changing the batch size and was unable to sufficiently decrease the memory I needed to successfully train any of my models. The PACE support team advised me to use a strategy similar to what allowed me to increase the capacity of my environment, however, after moving my jupyter notebook to a new location I lost access to my custom environment and was unable to find sufficient documentation for a workaround. By this point the PACE support team was inundated with issues regarding the cooling breakdown and they were no longer able to guide me.

After my roadblocks with PACE I decided to use the NVIDIA Quadro RTX 5000 GPU that's available as apart of Dr. Davenport's lab. I was able to quickly establish a conda environment and set up port forwarding through an ECE Server to access my jupyter notebook. However, yet again, I ran into many issues with space on the machine in addition to network connectivity. Over the course of several weeks I was never able to completely train a model due to network timeouts and space limitations. I experimented with changing the batch size, changing the number of epochs, and using keras' Model Checkpoint callback to train iteratively, however, on the few occasions I was able to complete training on a model - the mean absolute error was never lower than 30%. Throughout each of these attempts I would wait several hours for training to complete, only for it to have failed or lost connectivity. Whenever the GPU ran out of memory it was also very unpredictable how quickly it would be up and running again, ranging from 18 hours to 1 hour.

GRU Results

While this has been a frustrating process, we were able to achieve some results prior to developing our coordinate system preprocessing technique (featured in Figure 3. When we were training using a flawed distance metric, our GRU completed a run using the data from User 000 for April thru July. When training on PACE the model received a training accuracy of 0.9663 and a test accuracy of 0.2642. This led us to the conclusion that we were vastly overfitting our model, which, when considering the training conditions, did make sense. We were only training on one person, and only inputting one day at a time, which meant we were effectively overlooking many of the nuances that apply on the day to day scale. For example, the model wouldn't recognize that the same route taken across multiple days but at different start times was, in fact, the same route. We predicted our test accuracy would increase as we considered a few different factors:

- Training over multiple days
- Considering multiple users with similar habits
- Shifting to a coordinate gridding system

Training over more time would give us a broader view of the individual habits for our selected users. Extracting information from users with similar habits would help our model become better at generalizing information, as well as give the model more data to train on. Finally, updating our distance metric would eliminate the error where our loss function would incorrectly penalize points that were close together.

Results for Transformer

We were also able to train a custom implementation of the Transformer on our raw data for the same User and time frame. The Transformer was significantly faster than the LSTM and it received a training and test accuracy of 0.993. We acknowledge that this score seems high, and we think this has to do with the timestep of the prediction. The GRU predicts points x timesteps in the future, which can mean a few minutes to a few hours. On the contrary, the Transformer we implemented only predicts 1 timestep in the future, which meant it was only predicting 3 seconds into the future for our first dataset. At a scale of 3 seconds a very appropriate prediction is to assume the user doesn't move and the latitude and longitude will remain the same, which is actually what our model ended up doing. We think transitioning to a different framework, ideally one implemented by a library, will help reduce some of the confusion surrounding the architecture and speed up our progress with developing this model.

Conclusion and Future Work

Thus far the models and methodology we have proposed seem promising. Our results show that the GRU and the Transformer are capable of processing this data, so our primary points of failure are when dealing with computational resources and when debugging our models to ensure we're predicting exactly what we intend to. Beyond the action items stated in the results, we are looking to incorporate dynamic time warping (DTW), a technique to standardize our data across time. This would help our accuracy scores because if a user took the same path at 8:00 am on Monday, and 10:00 am on Tuesday, DTW would normalize the paths so that the start time wasn't taken into account.

We are also considering shifting to convolutional attention in place of standard self-attention. Li et al. introduce convolutional self-attention to break the memory bottleneck that arises when training Transformers on large sequences. Additionally convolutional self-attention is proposed to work better in a time series context [11].

Finally, we are evaluating establishing a concentric distance metric instead of a coordinate based approach. If we gridded our data in a way that's similar to what's depicted in Figure 6, we'd eliminate the need for a 2D loss function and hopefully speed up our computation, leading to fewer issues in the realm of computational resources.

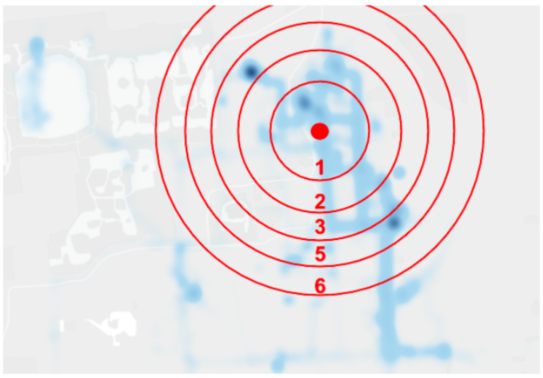

FIGURE 6: Concentric Coordinate System

As we research further into the field we have yet to discover any studies that claim to accomplish the same goals as us. We firmly believe there's a massive need for algorithms that deal with nonuniform GPS forecasting. Our results, combined with the results recent studies, indicate that different attention mechanisms are becoming more and more prominent in time series contexts - for instance the time series Transformer being developed by Max Cohen [6]. The success of the GRU with Attention or the Transformer would indicate that mobile GPS collection could be equally as successful with vastly less data. These results would also imply that smaller corporations, like Mogean, would be able to compete with larger corporations, like Google and Uber, because barriers to GPS forecasting would be significantly reduced.

Bibliography

[1] Florent Altché and Arnaud de La Fortelle. "An LSTM Network for Highway Trajectory Prediction". In: *CoRR* abs/1801.07962 (2018). arXiv: 1801.07962. URL: http://arxiv.org/abs/1801.07962.

[2] Samet Ayhan and Hanan Samet. "Aircraft Trajectory Prediction Made Easy with Predictive Analytics". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 21–30. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939694. URL: http://doi.acm.org/10.1145/2939672.2939694.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2015).

[4] Ian Barnett and Jukka-Pekka Onnela. *Inferring Mobility Measures from GPS Traces with Missing Data*. 2016. arXiv: 1606.06328 [stat.ME].

[5] Gano Chatterji. "Short-term Trajectory Prediction Methods". In: *Guidance, Navigation, and Control Conference and Exhibit*. DOI: 10.2514/6.1999-4233. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.1999-4233. URL: https://arc.aiaa.org/doi/abs/10.2514/6.1999-4233.

[6] Max J Cohen. *Transformers for Time Series*. 2020. URL: https://timeseriestransformer.readthedocs.io/en/latest/README.html.

[7] Jochen Eisner et al. "Algorithms for Matching and Predicting Trajectories". In: Oct. 2011, pp. 84–95. DOI: 10.1137/1.9781611972917.9.

[8] Esther Calvo Fernández et al. "DART : A Machine-Learning Approach to Trajectory Prediction and Demand-Capacity Balancing". In: 2017.

[9] Bin Han. *Mining Mobile Object Trajectories: Frameworks and Algorithms*. 2013.

[10] Mu Li, Amr Ahmed, and Alexander J. Smola. "Inferring Movement Trajectories from GPS Snippets". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. Shanghai, China: ACM, 2015, pp. 325–334. ISBN: 978-1-4503-3317-7. DOI: 10.1145/2684822.2685313. URL: http://doi.acm.org/10.1145/2684822.2685313.

[11] Shiyang Li et al. "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting". In: *CoRR* abs/1907.00235 (2019). arXiv: 1907.00235. URL: http://arxiv.org/abs/1907.00235.

[12] Yang Li and José M. F. Moura. *Forecaster: A Graph Transformer for Forecasting Spatial and Time-Dependent Data*. 2019. arXiv: 1909.04019 [cs.LG].

[13] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *CoRR* abs/1508.04025 (2015). arXiv: 1508.04025. URL: http://arxiv.org/abs/1508.04025.

[14] Yingchi Mao et al. “An Adaptive Trajectory Clustering Method Based on Grid and Density in Mobile Pattern Analysis”. In: *Sensors* 17 (Sept. 2017), p. 2013. DOI: 10.3390/s17092013.

[15] Paul Newson and John Krumm. “Hidden Markov Map Matching Through Noise and Sparseness”. In: *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2009), November 4-6, Seattle, WA*. 2009, pp. 336–343. URL: https://www.microsoft.com/en-us/research/publication/hidden-markov-map-matching-noise-sparseness/.

[16] Nishant Nikhil and Brendan Tran Morris. *Convolutional Neural Network for Trajectory Prediction*. 2018. arXiv: 1809.00696 [cs.CV].

[17] SeongHyeon Park et al. “Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture”. In: *CoRR* abs/1802.06338 (2018). arXiv: 1802.06338. URL: http://arxiv.org/abs/1802.06338.

[18] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[19] Yu Zheng, Xing Xie, and Wei-Ying Ma. “Mining Interesting Locations and Travel Sequences From GPS Trajectories”. In: *Proceedings of International conference on World Wide Web 2009*. WWW 2009. 2009. URL: https://www.microsoft.com/en-us/research/publication/mining-interesting-locations-and-travel-sequences-from-gps-trajectories/.

[20] Yu Zheng, Xing Xie, and Wei-Ying Ma. “Understanding Mobility Based on GPS Data”. In: *Proceedings of the 10th ACM conference on Ubiquitous Computing (Ubicomp 2008)*. 2008. URL: https://www.microsoft.com/en-us/research/publication/understanding-mobility-based-on-gps-data/.