Context-Aware Destination and Time-To-Destination Prediction Using Machine Learning

Athanasios Tsiligkaridis*, Jing Zhang†, Ioannis Ch. Paschalidis*,
Hiroshi Taguchi‡, Satoko Sakajo‡, Daniel Nikovski†
*Boston University, Boston, MA, USA
{atsili, yannisp}@bu.edu
†Mitsubishi Electric Research Laboratory, Cambridge, MA, USA
{jingzhang, nikovski}@merl.com
‡Mitsubishi Electric Corporation, Tokyo 100-8310, Japan
{Taguchi.Hiroshi@dw.MitsubishiElectric.co.jp, Sakajo.Satoko@ds.MitsubishiElectric.co.jp}

2022 IEEE International Smart Cities Conference (ISC2) | 978-1-6654-8561-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISC255366.2022.9922593

***Abstract*—The rapid adoption of Internet-connected devices (i.e., smart phones, smart cars, etc.) in today's society has given rise to a massive amount of data that can be harnessed by intelligent systems to learn and model the behavior of people. One useful set of such data is movement data, which can readily be obtained via GPS or motion-detection sensors, and which can be used to create models of user movement. One relevant application task based on this type of data is destination prediction, where movement data are used to form highly customized models that can forecast intended user destinations based on partially observed trajectories. In this work, we present a two-stage predictive model for destination prediction and Time-To-Destination (TTD) estimation using movement trajectories and contextual information. Our two-stage approach uses a Transformer-based architecture to predict an intended destination and a regression model to estimate how many steps must be traversed before a destination is reached. We showcase experimental results on various trajectory datasets and show that our proposed approach is able to yield significant destination prediction improvements over previous state-of-the-art methods and can also produce accurate TTD estimates.**

***Index Terms*—Destination prediction, transformer, deep learning, smart city, machine learning.**

I. Introduction and Related Work

The abundance of smart devices in today's society results in a plethora of recorded data which can be used to learn about a given person's behavior and trends. Location information is a useful set of such data, which can be easily obtained via GPS-enabled devices in an indoor/outdoor setting or from proximity sensors. Location data consists of both positioning (e.g., latitude/longitude coordinates, indoor grid locations) and contextual information (e.g., timestamp, ID tag number); this information is of high importance to user-based service technologies (e.g., Google maps, car navigation systems, predictive smart elevator systems), as it can be used to create intelligent models that represent a user and his/her tendencies when either driving or walking. One beneficial use of this information is to create a *destination prediction* model that can be used to predict a person's intended destination given position and/or additional context data. A model of this type can be used to provide quick and input-free routing information based on an early predicted user destination, tailored advertisements for shops and stores within a close proximity of a predicted destination, and more. Having a means of creating an efficient and accurate destination prediction model is expedient to organizations that exploit these models, as they can provide benefits to both the companies and their users, such as improved user happiness and, in turn, a probable increase in profit resulting from a potential increase in users.

The task of destination prediction has been explored in both indoor and outdoor settings, with solutions ranging from model-based to deep learning-based methods as of late.

A Bayesian approach to destination prediction was used to create destination probability distributions and predict potential destinations given partial input trajectories [1]. In [2], a trajectory distribution model using Gaussian Mixture Models was used to model clusters of similar trajectories and then input query trajectories were assigned to appropriate clusters to determine intended destinations.

In [3], [4], the Sub-Trajectory Synthesis algorithm was proposed as a means of data augmentation for the case where not enough trajectories exist to cover all potential input queries; a Markov model was employed to predict potential destination given a query trajectory. Markov models [5], [6] and Hidden Markov Models [7]–[10] have been extensively studied for route and destination prediction problems. These model-based approaches have been observed to falter in the presence of long sequence inputs; deep learning approaches remedy this.

In [11], deep learning methods, such as Multi-Layer Perceptrons (MLP) and Recurrent Neural Networks (RNN), were implemented and compared in the task of destination prediction on a real taxi trajectory dataset. RNNs have been widely used for destination prediction [12]–[14]; Long-Short Term Memory (LSTM) networks, which improve upon the gradient vanishing/exploding limitation of the RNN [15], [16], have also been used successfully in trajectory prediction tasks [17]–[19]. This model has also been used for the task of route and destination prediction using metadata and mapped trajectories [20]; in [21], a hierarchical model using a fusion of an attention mechanism and the LSTM structure was used to predict destinations.

After LSTMs, Transformer-based architectures became prevalent. In [22], a Transformer was used to predict destinations in a contextless datasetting where solely positioning information was used to make informed decisions. Transformer networks have also recently been used for the task of trajectory forecasting [23]. A spatio-temporal based Transformer has also been proposed as a means of predicting the next destination of a taxi driver given geographical information [24].

As an extension to [22], which presents a contextless Transformer for destination prediction, we propose a two-stage context-informed prediction system for accurate destination and Time-To-Destination (TTD) prediction. Estimating how much time will be needed to reach a destination is something that is immensely useful alongside the destination prediction task.

In this paper, we ultimately make the following contributions. First, we present a novel two-stage architecture for destination prediction and TTD estimation. Second, we apply our methods on both simulated indoor movement and real outdoor taxi trajectory datasets and showcase improved performances in destination prediction along with the accurate TTD estimation element, which has not been previously explored in destination prediction applications.

II. Proposed Approach

We propose a two-stage system, as shown in Figure 1, for both *destination prediction* and *Time-To-Destination* (TTD) estimation. In the first stage, positioning and context data are used to predict a destination. In the second stage, the predicted destination obtained from the first stage, along with the current location (most recent position in an input sequence), is used to estimate the amount of *steps* that will need to be taken in order for a user to reach his/her intended destination, given that the user is currently at the most recent position in the input trajectory. We make the assumption that a user moves around a floor, or more generally some indoor/outdoor area, at a near constant speed; with this, if we have an estimate of the amount of *steps* that need to be taken at some known speed, then a time value can be obtained which will represent a person's time to his/her desired destination.

We elect to train a single global model that receives context information as input when predicting destinations, as opposed to training independent models for each context value or group. This helps when context is continuous-valued or the alphabet of discrete context is large, where training independent models would be computationally expensive and infeasible. Additionally, context partitioning to form multiple models over each created context group assumes prior domain knowledge of how to create these groups, and pre-defining groups can introduce bias into the model. By using a global model that takes context into account, we avoid any such induced bias.

In the first stage of our system, we propose a Transformer Encoder Stack (TES) as a destination prediction model that takes as input both position and context data and outputs a probability distribution over all potential destinations in a given space. In the second stage, we propose a regression model as a means of predicting the amount of steps that need to be traversed to reach a desired destination. The best regressor to use in the second stage is data dependent; in our experiments, results are shown for several choices.

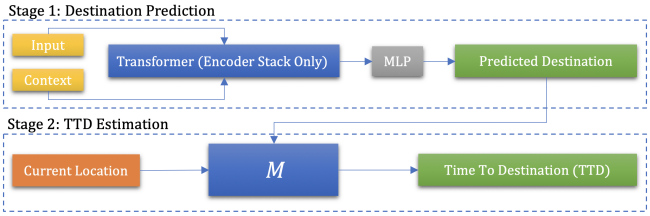

Fig. 1: Our proposed dual stage system for destination prediction using a TES and TTD estimation using a general regression model M.

III. TES Architecture

The original Transformer architecture was proposed in [25] as a sequence-to-sequence model for NLP tasks and was shown to outperform previous state of the art methods (e.g., BiLSTM, RNN, LSTM) in various translation, generation, and understanding tasks [25]–[28]. The Transformer model uses an *attention* mechanism to relate sequence elements; this allows for parallelization in training (and, in turn, more efficient model training) and the dependency modeling of elements in an input sequence without consideration of their distances. The original Transformer is a system comprised of a stack of N encoder and decoder blocks containing both *stacked self-attention* and *fully connected feed forward layer* components.

Our TES is similar to the original Transformer except that there is no decoder present and the original *positional encoding* and *embedding* transformations of the sequence input are concatenated when preparing the input to the encoder stack. In the original model, these are added. The concatenation operation has been observed to yield improved performance in prediction accuracy over all tested datasets. Our proposed TES architecture is shown in Figure 2.

Our proposed mechanism takes as input both a positioning sequence and contextual information, labeled as *Input* and *Context*, respectively, in Figure 2. The positioning sequence passes through an embedding layer which converts the sequence into vectors of length d_{model}. The same sequence also passes through a positional encoding block which identifies information about the ordering of its elements. The outputs of the preceding two blocks are then concatenated with each other, and in turn, concatenated once again with a linearly embedded version of the contextual information. This vector then moves into an encoder stack where in each encoder it first encounters a *Multi-Head Attention* (MHA) block. This MHA block consists of a set of h scaled-dot product attention modules, where each one attempts to focus on a specific pattern of an input. Having multiple of these in parallel, which is what the MHA block is, allows for the TES to focus and learn multiple patterns in the input. Figure 3 shows the

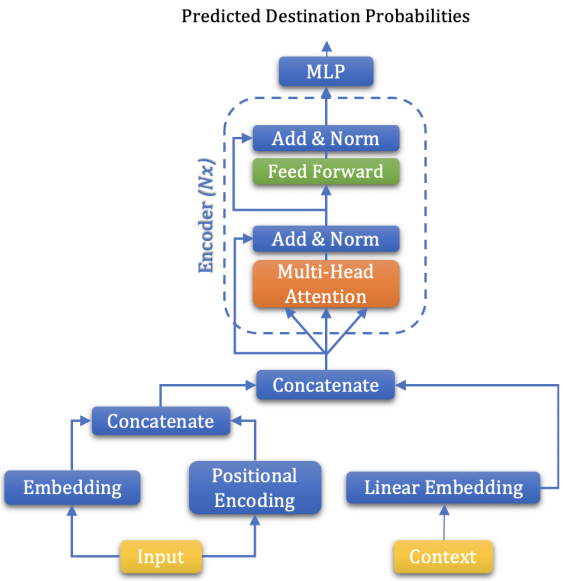

Fig. 2: Stage 1 setup consisting of a Transformer Encoder Stack (TES) along with an external context embedding, which is then fed into the TES via a concatenation operation followed by an MLP which outputs predicted destination probabilities over a grid space.

structure of the scaled-dot product attention module, adapted from [29]. Given a data matrix X, which represents the input to the encoder, three projection matrices, W_Q, W_K, and W_V, are first learned in the training of the TES model. These matrices are then used to form projections of the data matrix which can be thought of as abstractions used to calculate attention; specifically, query $Q = XW_Q$, key $K = XW_K$, and value $V = XW_V$ matrices are obtained. The query and key matrices are multiplied and then scaled by $1/\sqrt{d_k}$, where d_k is the number of columns in the K matrix, in order to have the product matrix of Q and K be unit variance. Next, a SoftMax function is applied to the scaled product of Q and K in order to form an *attention score matrix* whose values will represent probabilities and will be in the interval of $(0, 1)$. This attention score matrix is then used as a weighting matrix and is multiplied with the value matrix V to calculate attention. Ultimately, attention is calculated as: $\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_k})V$. With MHA, multiple attention functions are calculated with different learned projections (W_Q, W_K, W_V). Outside of the MHA block, a Residual connection along with a Layer Normalization (RLN) is then applied which connects the input to the attention block with its output, adds them, and then normalizes. This output then passes through a feed-forward module and another RLN connection. This entire procedure is repeated N times, where N represents the number of encoders. The output of the final encoder then passes through a Multi-Layer Perceptron (MLP), which is used to obtain a probability distribution over the output space. The maximum value over the space will represent our model's predicted destination.

We do not use a decoder stack in our proposed model because we do not predict a multi-element trajectory where the initial masked attention and attention modules in the decoder are needed; we solely attempt to predict a single element, so our problem is cast as a classification task where we associate a partial trajectory with a specific location in a grid space, which can be interpreted as the destination. The decoder stack is advantageous in problems where symbol-by-symbol decoding is necessary, such as in trajectory prediction tasks. Using a decoder here might result in incorrect representations of our output, which can diminish performance. Also, using the encoders from the Transformer architecture to form our prediction model has the additional benefit of being able to access the whole input sequence at once when forming representations, unlike other sequential models that do not have this ability (e.g., LSTM, BiLSTM); this speeds up model training, especially on large and complex datasets.

IV. Experimental Results

Datasets: We use the following datasets for our experiments: 1) SimTread artificial indoor human trajectory dataset, and 2) Porto taxi dataset. For each, we present results on the datasettings where context *is* and *is not* available; this is done to show the benefits of using contextual information in the destination prediction task.

Computing Infrastructure: Experiments were performed with one Tesla P100 GPU on a computer with 16GB RAM. All deep learning-based methods were implemented using the PyTorch deep learning framework; all conventional machine learning-based methods were implemented using the scikit-learn library in Python.

Baselines: For each dataset, we present performance results for each of the two stages and compare against a set of different approaches for each stage.

In the first stage, we compare our TES approach, with a single encoder block, with the previous state-of-the-art one-level stacked LSTM and BiLSTM methods. In the artificial dataset, we additionally compare against a baseline Hidden Markov Model (HMM), where sequence data (along with contextual data) is first fed into an HMM which outputs a state sequence that is fed into an MLP classifier to predict a user's intended destination. This HMM approach yields very underwhelming performance for the taxi dataset, so this result is not included in those experiments. We also provide a comparison against the original Transformer approach used in [22].

In the second stage, we compare various regression models such as Linear, Ridge, LASSO, Elastic Net, MLP, Random Forest, Adaboost, and Gradient Boosting to observe how well TTD can be estimated.

Modeling Assumptions: All of our positioning data involves coordinates, so we discretize the coordinate spaces of our data into grid cells for easier data usage. With this, the problem of destination prediction effectively becomes a classification

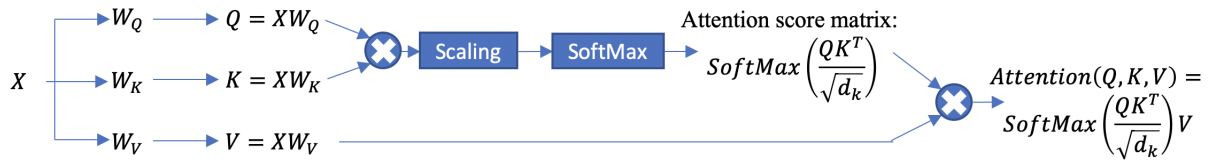

Fig. 3: Visualization of one of the heads in the Multi-Head Attention module in the encoder of our proposed TES mechanism.

problem, where we use input query trajectories that are represented as grid cell index sequences to predict a destination location which is represented by a single grid cell index.

Input discretization is a modeling choice primarily driven by indoor applications, where positions are in most cases discrete (e.g., based on proximity sensors). Prior approaches, such as those mentioned in [14], [20], [21], and [22], also discretize the input space. Treating data as continuous positions remains an interesting problem which can be explored as future work.

Performance Metrics: For the destination prediction stage in the artificial dataset, we look at the true destination probability over various train/test splits as a function of the percentage of the observed test trajectory.

For the taxi dataset, we measure model performance using the Block Distance (BD) metric used in [22], which measures how many grid jumps away an obtained destination D_O is from the true destination D_T; essentially, it is a measure of closeness of a prediction to the true destination. If D_O and D_T are mapped to row and column grid pairs $[x_O, y_O]$ and $[x_T, y_T]$, respectively, then BD is calculated as: $BD|_{O->T} = \max\{|x_T - x_O|, |y_T - y_O|\}$.

In the TTD estimation stage for either dataset, where we attempt to correctly predict the amount of grid steps that need to be traversed from a current location in a sequence to the predicted user destination, we quantify performance using Mean Absolute Error (MAE). This MAE metric quantifies how close our prediction of the number of grid steps needed to reach the true destination is to the true number of grid steps needed to reach the destination while at the most recent (current) location in a movement trajectory.

Implementation Details: For the taxi dataset, our TES method was implemented using a single encoder and the dimension of all model sub-layers and embedding layers were chosen to be $d_{\text{model}} = 128$. The output MLP contains a single hidden layer of size 128. In the MHA module, $h = 8$ parallel scaled dot-product attention layers are used. For each of these, we have $d_k = d_{\text{model}}/h = 16$. A dropout rate of 0.05 is used along with a batch size $B = 128$. We train our model for 200 epochs using a cross entropy loss objective and we use an ADAM optimizer with a learning rate of 0.005. For the SimTread dataset, similar parameters are used but we train our model for 100 epochs, select a smaller batch size of 8, a hidden layer size of 64, and use $d_{\text{model}} = 64$. The original transformer we compare against from [22] uses the same parameters for either dataset though a NOAM optimizer with label smoothing [25] is used in place of the ADAM optimizer.

For the taxi dataset, the LSTM and BiLSTM models we compare against for both datasets are implemented using a single stack, an embedding dimension of 128, a hidden layer dimension of 32, a dropout rate of 0.05, and a batch size $B = 128$. We train the model for 200 epochs using a cross entropy loss objective and an ADAM optimizer with a learning rate of 0.001. For the SimTread dataset, similar parameters are used but we select a smaller batch size of 8 and use smaller embedding and hidden dimensions of 64 and 32, respectively.

For the HMM method we compare against in the SimTread dataset experiment, we use a HMM with 8 states (larger amounts of states were tested but did not yield any improvements in terms of destination prediction accuracy) and a connected MLP with two hidden layers of dimensions $(150, 100)$.

Regarding the TTD prediction tasks on the taxi dataset in the second stage of our proposed system in Figure 1, we test Linear, Lasso, Ridge, and Elastic Net regressors using the default parameters from their scikit-learn implementations. Various amounts of regularization were tested but the default case with the regularization parameter $\alpha = 1.0$ yielded the best MAE metric values. The MLP we implemented contained three hidden layers of sizes $(200, 100, 50)$. The Random Forest regressor we used was defined to have a per-tree maximum depth of 15 along with a maximum number of trees of 1000. The Adaboost regressor we used employed a decision tree base classifier with a per-tree maximum depth of 15, a default learning rate of 1.0, and a maximum number of estimators at which boosting is terminated of 1000. Finally, for the Gradient Boosting regressor, we used 150 boosting stages with a default learning rate of 0.1 and a maximum depth per regression estimator of 15. For the SimTread dataset results, similar parameters were used but a Random Forest maximum tree depth of 8 and a Gradient Boosting stage amount of 50 were employed.

A. *Artificial Indoor Dataset*

We first apply our two stage system to a small artificial indoor dataset. The SimTread[1] simulation software package is used to generate this data; this package creates continuous coordinate data which represent movement trajectories of people in an indoor setting. We create movement patterns on a floor for two time intervals in a day (morning and afternoon). In each time interval, we create 10 trajectories that can end at one of 3 total destinations. The morning and afternoon positional data are visualized in Figure 4 and 5, respectively.

[1]https://www.vectorworks.cn/en/community/partner-community/partner-products/product/simtread/

Stage 2 - TTD Estimation (SimTread)	
Method	MAE (steps)
Linear Regression	10.57
Ridge Regression	10.57
LASSO Regression	10.57
Elastic Net	10.57
Multi-Layer Perceptron (MLP)	5.77
Random Forest	2.24
Adaboost	**1.74**
Gradient Boosting	2.81

TABLE I: MAE results for TTD Estimation using various regression models on the SimTread dataset. We see that Adaboost yields the smallest error when estimating the amount of grid blocks needed to be traversed before a destination is reached.

The entire floors are partitioned into 50^2 blocks and each coordinate trajectory is converted to a sequence of grid indices. In this data, contextual information is a time interval label, which allows us to discern between morning and afternoon periods in a day.

When obtaining performance results for Stage 1, we use the Leave-One-Out Cross-Validation (LOOCV) technique to obtain average performance results for our tested prediction models, since this dataset contains a small amount of trajectories. Specifically, given S total trajectories, we go through S iterations, where in each iteration we use one of the S trajectories for model testing and the remaining $S-1$ paths for model training. Per data split iteration, we obtain a collection of correct destination probabilities as we increase the percentage of our query test trajectory. Then, we average these values over all S iterations to obtain destination prediction performance measures for a given model.

Figure 6 displays correct destination probabilities of the TES, LSTM, BiLSTM, and HMM approaches in the cases where context is and is not included as model input, as a function of the percentage of observed query trajectories. We see that the inclusion of context, which in this setting is the time-interval label (morning or afternoon), yields improvements in prediction probability, as expected. Additionally, we see that our TES approach is able to correctly detect the intended user destination early in the development of the query trajectory, unlike with the other models.

To create the data for stage 2, we first convert each trajectory of a given length L into L pairs of current points and true destinations, which represent the regressor inputs. For every input pair, we then measure the amount of grid jumps needed to reach the destination from the current point; this represents the output of the regression model M. We then use these inputs and outputs to train and test various regression models using an 80%-20% train/test split. Table 1 showcases the MAE values of each tested method; we see that Adaboost yields the smallest error, and in turn the best TTD estimates, for this dataset.

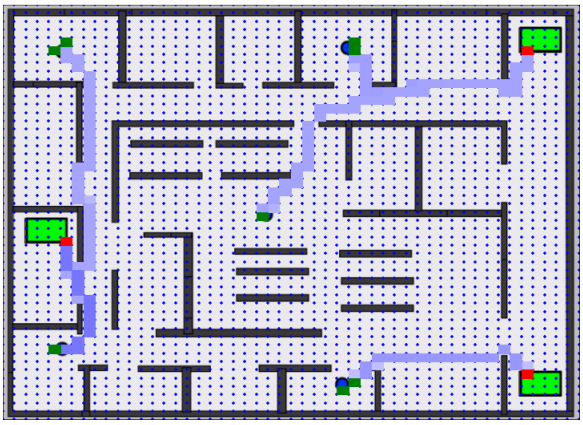

Fig. 4: SimTread data in the first time interval (morning). Trajectory start and end locations are represented by the dark green and bright red blocks on the grid, respectively. The 3 large and bright green blocks represent destinations.

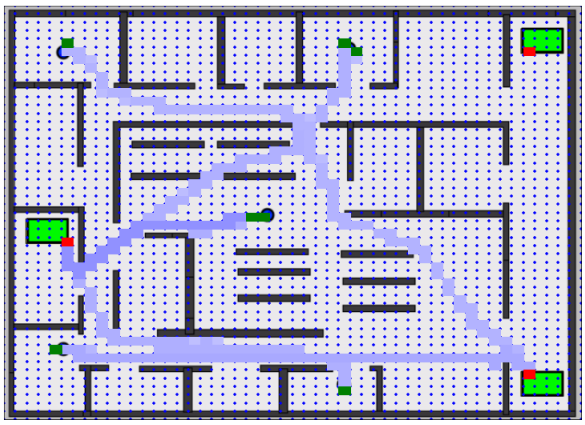

Fig. 5: SimTread data in the second time interval (afternoon). The trajectories in this time interval are different from those in the morning time interval; this is done to accurately model human behavior. People tend to go to different floor destinations, and possibly by using different routes, depending on the time of day.

B. Porto Taxi Dataset

Next, we test our two-stage approach on a real dataset involving taxi trajectories. We use the popular taxi dataset used in a 2015 Kaggle competition[2]. This dataset contains metadata and movement trajectories of 442 taxis in Porto, Portugal, recorded over a whole year.

Figure 7 displays the taxi trajectory data we use for our experiments, where we constrain our data to be within the following latitude and longitude bounds: $[41.135, 41.160]$ and $[-8.600, -8.560]$, respectively. This region represents a neighborhood in Porto which we partition into 25^2 grid cells. Each trajectory in this space is converted from coordinates to grid cell indices, and we also extract timestamps as contextual

[2]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

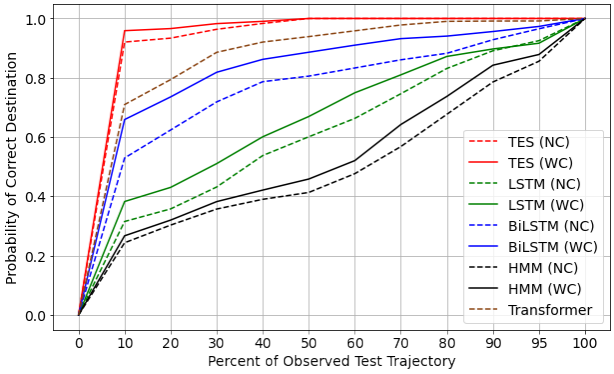

Fig. 6: Correct destination probability over increasing percentages of input test trajectories. Our TES approach outperforms the baseline HMM and previous state-of-the-art (LSTM, BiLSTM, Transformer [22]) methods in both with (WC) and without (NC) context settings. The importance of contextual information is also highlighted through boosts in prediction performance for each approach.

information. We form our training set by extracting 200 taxi trajectories per destination over the top 20 most popular destinations in the region; similarly, we create our testing set by extracting 20 trajectories per destination over the top 20 most popular destinations.

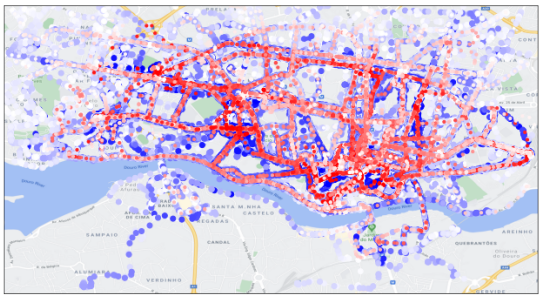

Fig. 7: Visualization of the Porto taxi dataset subset used for training and testing our prediction models. Trajectories used for training and testing are shown in blue and red, respectively. The color gradient from light to solid dark represents the evolution of a trajectory from start to finish, respectively.

In the first stage, we train TES, LSTM, BiLSTM, and Transformer (only without context since [22] does not include a means of handling contextual information) models using full trajectories, and we test on partial trajectories of increasing length. Figure 8 displays the BD performance curves for all tested methods with and without contextual information. We do not display performance curves for the HMM approach as it performed significantly worse than all other deep learning-based approaches. We display BD results for observed trajectory percentages of over 40%, as the obtained metrics are insignificant and too large to be advantageous for all tested methods for smaller percentages in this taxi application. We first observe that the use of contextual information tends to yield smaller BD values over the no-context cases in most methods. Also, our TES approach consistently yields smaller BD values, and in turn more accurate destination estimates early in the development of a test trajectory, over all compared methods, both with and without context.

Stage 2 - TTD Estimation (Taxi)	
Method	MAE (steps)
Linear Regression	4.56
Ridge Regression	4.56
LASSO Regression	4.56
Elastic Net	4.56
Multi-Layer Perceptron (MLP)	3.45
Random Forest	1.99
Adaboost	3.64
Gradient Boosting	**1.96**

TABLE II: MAE results for TTD Estimation using various regression models using the Porto taxi dataset. We see that Gradient Boosting yields the smallest error when estimating the amount of grid blocks needed to be traversed before an intended destination is reached.

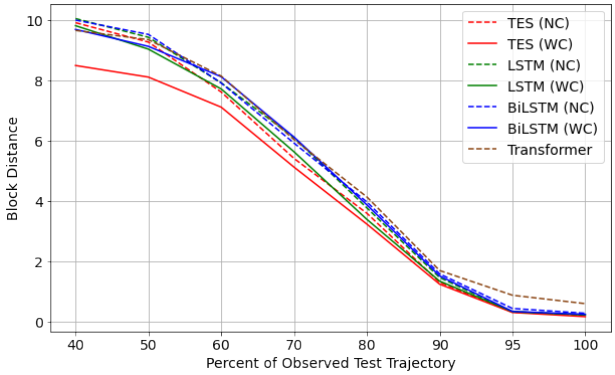

Fig. 8: Block Distance plot over the percentages of observed trajectories. Our TES approach outperforms all other methods in both the with (WC) and without (NC) context cases. Additionally, we observe that the inclusion of contextual data provides prediction benefits early in the development of the trajectories, but as more of the trajectory is seen, context becomes less important.

To create the data for Stage 2, we carry out the same data preparation procedure that was used in the SimTread dataset. Table 2 presents the MAE values of each tested method; we see that the Gradient Boosting regressor yields the smallest TTD estimation error in this setting and provides the best estimate of the number of grid steps that must be traversed to reach an intended destination.

V. Conclusion

In this paper, we proposed a two-stage system for destination prediction and Time-To-Destination estimation using movement data along with additional contextual information. Through experimental results, we demonstrated improved prediction performance over previous state-of-the-art approaches and an ability to accurately estimate the amount of grid units

a user will traverse before reaching the intended destination, which in turn can be used to estimate the amount of time needed to reach a destination under the assumption of a known travel speed. As future work, we can explore an online approach to user movement modeling where movement trajectories are monitored over time, and a continual learning-based model updates itself without full re-training. Additionally, we can consider a multi-task learning approach where both destination and TTD are jointly predicted.

REFERENCES

[1] J. Krumm and E. Horvitz, "Predestination: Inferring destinations from partial trajectories," in *Proceedings of the Eighth International Conference on Ubiquitous Computing*, 2006.

[2] P. C. Besse, B. Guillouet, J. Loubes and F. Royer, "Destination prediction by trajectory distribution-based model," in *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems*, 2018.

[3] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," in *Proceedings of the 2013 IEEE International Conference on Data Engineering*, 2013.

[4] A. Y. Xue, J. Qi, X. Xie, R. Zhang, J. Huang and Y. Li, "Solving the data sparsity problem in destination prediction," *The VLDB Journal*, vol. 24, no. 2, pp. 219–243, 2015.

[5] B. D. Ziebart, A. L. Maas, A. K. Dey and J. A. Bagnell, "Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior," in *Proceedings of the Tenth International Conference on Ubiquitous Computing*, 2008.

[6] N. Ye, Z. Wang, R. Malekian, Q. Lin and R. Wang, "A method for driving route predictions based on hidden markov models," *Mathematical Problems in Engineering*, 2015.

[7] R. Simmons, B. Browning, Y. Zhang and V. Sadekar, "Learning to predict driver route and destination intent," in *Proceedings of the 9th IEEE International Conference on Intelligent Transportation Systems*, 2006.

[8] Y. Lassoued, J. Monteil, Y. Gu, G. Russo, R. Shorten and M. Mevissen, "A hidden markov model for route and destination prediction," in *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems*, 2017.

[9] J. P. Epperlein, J. Monteil, M. Liu, Y. Gu, S. Zhuk and R. Shorten, "Bayesian classifier for route prediction with markov chains," in *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems*, 2018.

[10] F. Zong, Y. Tian, Y. He, J. Tang and J. Lv, "Predicting destinations by a deep learning based approach," *Physica A*, vol. 515, 2019.

[11] A. D. Brebisson, E. Simon, A. Auvolat, P. Vincent and Y. Bengio, "Artificial neural networks applied to taxi destination prediction," in *Proceedings of the International Conference on ECML PKDD*, 2015.

[12] L. Zhang, G. Zhang, Z. Liang and E. F. Ozioko, "Multi-features taxi destination prediction with frequency domain processing," *PLoS One*, vol. 13, no. 3, 2018.

[13] L. Zhang, G. Zhang, Z. Liang, Q. Fan and Y. Li, "Predicting taxi destination by regularized RNN with SDZ," *IEICE Transactions on Information and Systems*, vol. E101.D, no. 8, 2018.

[14] Y. Endo, K. Nishida, H. Toda and H. Sawada, "Predicting destinations from partial trajectories using recurrent neural network," in *Proceedings of the 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2017.

[15] Y. Bengio, P. Frasconi and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *Proceedings of the IEEE International Conference on Neural Networks*, 1993.

[16] R. Pascanu, T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

[17] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[18] Y. Xu, Z. Piao and S. Gao, "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[19] X. Shi, X. Shao, Z. Guo, G. Wu, H. Zhang and R. Shibasaki, "Pedestrian trajectory prediction in extremely crowded scenarios," *Sensors*, 2019.

[20] P. Ebel, I. E. Gol, C. Lingenfelder and A. Vogelsang, "Destination Prediction Based on Partial Trajectory Data," *arXiv e-prints*, Apr 2020.

[21] J. Xu, J. Zhao, R. Zhou, C. Liu, P. Zhao and L. Zhao, "Predicting destinations by a deep learning based approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 2, 2021.

[22] A. Tsiligkaridis, J. Zhang, H. Taguchi and D. Nikovski, "Personalized destination prediction using transformers in a contextless data setting," in *Proceedings of the 2020 International Joint Conference on Neural Networks*, 2020.

[23] F. Giuliari, I. Hasan, M. Cristani and F. Galasso, "Transformer networks for trajectory forecasting," in *Proceedings of the International Conference on Pattern Recognition*, 2020.

[24] Z. U. Abideen, H. Sun, Z. Yang, R. Z. Ahmad, A. Iftekhar and A. Ali, "Deep wide spatial-temporal based transformer networks modeling for the next destination according to the taxi driver behavior prediction," *Applied Sciences*, vol. 11, no. 17, 2021.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

[26] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing," *arXiv e-prints*, Aug 2017.

[27] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, "Improving language understanding by generative pre-training," in *OpenAI*, 2018.

[28] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R.Sepassi, L. Kaiser and N. Shazeer, "Generating wikipedia by summarizing long sequences," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[29] S. Yildirim and M. Asgari-Chenaghlu, *Mastering Transformers*, Packt Publishing, 2021.