Personalized Destination Prediction Using Transformers in a Contextless Data Setting

Athanasios Tsiligkaridis*
Boston University
atsili@bu.edu

Jing Zhang
Mitsubishi Electric Research Labs
jingzhang@merl.com

Hiroshi Taguchi
Advanced Technology R&D Center
Mitsubishi Electric Corporation
Taguchi.Hiroshi@dw.MitsubishiElectric.co.jp

Daniel Nikovski
Mitsubishi Electric Research Labs
nikovski@merl.com

***Abstract*—Destination prediction is an important task where the primary goal is to correctly predict a user's destination given an input movement trajectory. Intelligent machine learning models that learn from observed movement data and can automatically forecast destinations from partial query trajectories are of high interest as they can provide a plethora of benefits to both creators and consumers in various markets. In this work, we present a novel framework for tackling the problem of destination prediction in a contextless data setting where we solely learn from trajectory coordinate information. We propose a Transformer model to predict destinations from partial trajectories and we demonstrate its use on two datasets from different domains, including a simulated indoor dataset and an outdoor taxi trajectory dataset. Our proposed method improves upon the previous state-of-the-art LSTM and BiLSTM deep learning approaches in terms of accuracy and distance from true destinations.**

***Index Terms*—destination prediction, transformer, deep learning, personalized prediction, probabilistic prediction**

I. Introduction

With the rapid advancement of technology in today's society, there exists an abundance of recorded data that can be harnessed to understand people's behavior, trends, and tendencies. One relevant set of data is location information (e.g., GPS data obtained outside while driving/walking via a cellular device, indoor walking data obtained via proximity sensors). Movement data in the form of trajectories can be used by various indoor or outdoor location-based services (e.g., Google Maps, Apple Maps) to assist a user in her travels through the use of a *destination prediction* intelligence method. Specifically, if such a service can perform destination prediction, then user movement data can be used to predict a person's intended destination; with this, the service can provide tailored advertisements based on the destination and any nearby locations, fastest route information based on early detection of a destination without any user input needed, and more. The task of predicting destinations from input trajectories is very advantageous to companies that can use this information and can result in improved user satisfaction, higher service usage, and increased efficiency in a user's day.

*Work performed while at Mitsubishi Electric Research Labs (MERL).

The problem of destination prediction has been studied in a multitude of settings (e.g., pedestrian, vehicular, misc.); most solutions involve either model-based or deep learning-based approaches, with the latter being much more popular nowadays due to the eminence and strength of deep learning techniques.

A. Model-Based Methods

Conventional model-based methods were some of the first explored and heavily-used solutions to the task of destination prediction and the closely-related trajectory prediction task.

In [1], a system called *Predestination* was proposed that used driver destination history and behavior to predict potential destinations during an ongoing trip. A Bayesian approach was taken to form destination probability distributions over a geographic area, which were then used to determine likely destinations from partial trajectories.

In [2], *PROCAB* was presented for Probabilistically Reasoning from Observed Context-Aware Behavior. A probabilistic model of behavior was formed where behavior was represented as sequential actions in a Markov Decision Process whose weights were learned using Inverse Reinforcement Learning and the principle of maximum entropy. The created model was successfully applied to vehicle route preference modeling (e.g., turn, route, and destination prediction).

A trajectory distribution model was formed for destination prediction in [3]. First, trajectories were clustered based on a similarity measure, and each cluster was then modeled by a 2D Gaussian Mixture Model. For a partial trajectory, destination forecasting was then done by first finding the cluster that the query would belong to, and then using a score metric over the cluster's constituent trajectories to determine a destination estimate.

Markov models have been very popular for route and destination prediction [2], [4] and pedestrian trajectory prediction [5], [6]. The Sub-Trajectory Synthesis algorithm was presented in [7], [8] to deal with the data sparsity issue (the issue of not having enough historical trajectories to cover all possible queries) by decomposing historical paths into sub-trajectories and connecting them into a plethora of synthetic trajectories. A Markov model was then used to leverage historical information

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

and posterior distribution values were then calculated and sorted to find the most likely destinations given an initial query.

Hidden Markov Models (HMM) were used in [9] along with road map knowledge to predict driver intent and destination. HMM were also used in [10], [11] in route and destination prediction problems. A combination of a Markov model and a Prediction by Partial Matching compression method was used in [12] for route and destination prediction.

B. Deep Learning-Based Methods

Deep learning-based methods have gained much traction over recent years due to their ability to work with long sequence inputs. Conventional model-based approaches encounter difficulties when dealing with long sequences of past observed data, as early information tends to dissipate when a long sequence is encountered. This limitation precludes these methods from making the best and most informed decisions in complex scenarios.

Deep learning methods for sequence processing first gained traction for their successes in areas such as natural language processing; they have been adapted since to non-language applications, such as destination/trajectory prediction. Specifically, Recurrent Neural Networks (RNN), which work with variable length time sequence data, have gained popularity since trajectories, which can be thought of as time sequence data, can be predicted via RNNs. Equivalently, RNNs can also be used for destination prediction as an output target can indeed be of length one.

In [13], a Multi-Layer Perceptron (MLP) was used for the problem of taxi destination prediction reformulated as a regression problem where the destination was expressed as a linear combination of mean locations of destination clusters. The proposed MLP method was the first place solution to the 2015 ECML/PKDD discovery challenge on taxi destination prediction. In addition, RNN and Bidirectional (Bi) RNN methods were also tested and performed very well (better than the MLP on a larger and more representative test set than that from the challenge, which the MLP performed the best on).

An RNN model was used for destination prediction in [14] where candidate destinations were obtained from partial trajectories by estimating visiting probabilities through object movement simulation with stochastic sampling. As an improvement, [15] used regularized RNN with surprisal-driven zoneout, which serves to prevent error transmission in an RNN, to solve the destination prediction problem in a vehicular setting. A frequency domain processing approach using Convolutional Neural Networks (CNN) and RNN was also created for taxi destination prediction in [16].

RNNs have also been leveraged for various trajectory prediction problems. In [17], an RNN was used for vehicle trajectory prediction; as an extension, an attention-based RNN model was used to accomplish the same task with improved performance [18]. RNNs were also used in [19] to predict hurricane trajectories.

RNNs are known to suffer from gradient vanishing/exploding issues [20], [21], which hinder them from remembering long input sequences; to deal with this, Long-Short Term Memory (LSTM) [22] was designed to allow for long term dependency retention. LSTM methods are being used for trajectory prediction because of their power to make informed decisions by retaining and using past observed sequence data.

A Social LSTM approach for pedestrian trajectory prediction was presented in [23] where an LSTM network was combined with a social pooling layer to model pedestrian interactions. In [24], an LSTM+MLP architecture was used to model pedestrian affinity and in turn determine location/trajectory displacements; [25] used a stacked LSTM approach for trajectory prediction. A combination of LSTM, CNN, and a Fully-Connected Neural Network (FCNN) was used in [26] to analyze user behavior, position mapping, and external features, respectively, for cyclist destination prediction.

C. Our Approach

We use a Transformer architecture as our destination prediction model by converting our problem to an equivalent sequence transduction problem that a Transformer is suitable for. Specifically, we consider a partial query trajectory as an input sequence and the end point of the full trajectory as the output sequence (of length 1). We choose to use the Transformer as our prediction model due to its dynamic attention mechanism that allows for improved model training speed through parallelization (problematic in RNN/LSTM) and a better handling and retention of long term dependencies (an issue in RNN and also sometimes in LSTM for long sequences) that become integral in transportation problems where long trajectories exist and beginning trajectory information can greatly influence decisions.

In this paper, we make the following contributions. First, we present a novel destination prediction system using the attention-based Transformer architecture. Second, we apply our method to both indoor movement and outdoor taxi trajectory datasets. Third, we demonstrate through numerical experiments that our approach achieves higher true destination confidence in the indoor dataset and higher accuracy and lower distance from the true destination in the outdoor dataset in comparison to two previous state-of-the-art LSTM and Bidirectional LSTM (BiLSTM) deep learning approaches.

II. Transformer Architecture

The Transformer was first introduced in [27] as a sequence transduction system that has been shown to outperform preceding state-of-the-art methods (RNN, LSTM, BiLSTM, etc.) in various tasks such as language translation [27], [28], natural language understanding [29], and document generation [30]. This system is devoid of any recurrent cell units present in RNN models and instead relies on a more effective *attention* module for relating elements in a sequence.

Unlike RNNs that deal with sequences in a sequential manner which can inhibit parallelization within training, the Transformer uses positional encoding and attention modules

which allow for parallelization and, in turn, a more efficient training procedure. In addition, an RNN model might have difficulty in learning dependencies between distant positions in sequences of large lengths, again due to its sequential nature; but, the Transformer's attention mechanism can allow for modeling of dependencies of elements in a sequence without consideration of their distances. This solely attention-based system consists of an encoder-decoder structure as seen in the architecture displayed in Fig. 1.

The system contains a stack of N encoder and decoder blocks in which there exist *stacked self-attention* and *fully connected feed forward layer* components.

Before the encoder, an input sequence first passes through an embedding layer where input tokens are converted to d_{model}-dimensional vectors. (The tokens are discrete elements.) Then, order information about the elements of the input is obtained using a positional encoding block, and is then combined with the embedding vector via a summation operation. This new vector then passes into the encoder where it first encounters a *multi-head attention block*. Attention in the encoder is used as a means of referencing other tokens in an input sequence when attempting to encode a specific token; this multi-head block consists of an array of *scaled-dot product attention* components where an attention function is calculated based on a data matrix X. The data matrix is projected using three different matrices that are learned in training and projected matrices Q, K, and V are obtained which represent queries, keys, and values, respectively. Attention is then calculated as: $\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_k}) \times V$, where d_k is the number of columns of the K matrix. Multi-head attention involves calculating multiple attention functions with different learned projections in order to yield improved representation performance. A Residual connection along with a Layer Normalization (RLN) then exists which connects the input to the attention block with its output, adds them, and then normalizes. The resulting elements then pass through a feedforward sublayer followed by another RLN connection on its way out of the encoder.

Before the decoder, a target output sequence passes through embedding and positional encoding blocks (similar to those regarding the input) before it enters a masked multi-head attention block. This element is similar to that of the encoder except that the output sequence is masked such that it can only refer to preceding output sequence positions. After an RLN layer, another attention block exists which is similar to that of the encoder except that it uses information from the encoder output as additional input in its calculation of attention. RLN, feedforward, and RLN layers comprise the rest of the decoder block.

Outside of the decoder, a linear layer exists that takes the decoder output and creates a *logit* vector over all possible output elements; then, a softmax operation is done to convert the values to output probabilities from which we can extract the maximum value and its index to locate a predicted destination.

When training the Transformer, a KL Divergence loss function is minimized over multiple epochs and a greedy decoding scheme is used as the method for finding the output destination. In our experiments, we trained the Transformer model using parameter values similar to those discussed in both [27], [31] as they empirically yielded the best prediction performance.

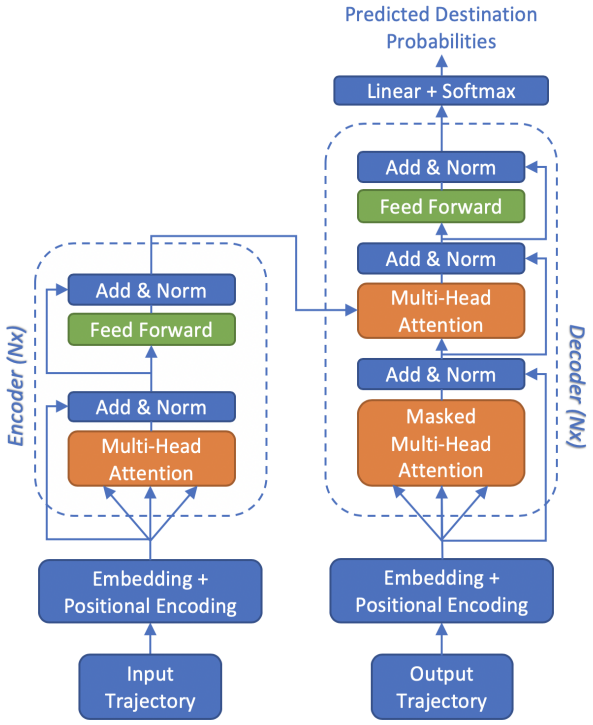

Fig. 1. Transformer architecture [27] comprising an encoder-decoder stack structure used for sequence-to-sequence conversion.

III. Experimental Results

We present numerical results on two complex datasets: A) A small simulated indoor trajectory dataset, and B) A larger taxi trajectory dataset. Other common datasets for destination prediction problems are the TrajNet[1] and the ETH[2]/UCY[3] datasets; these consider small physical spaces (i.e. a room, a city block, etc.) and their respective trajectories are not very lengthy and/or sinuous. The data we consider contain more complex trajectories due to their large lengths and their circuitous nature, as evidenced in Figures 2 and 4.

For our experiments, we consider a minimal data setting where we solely use trajectory coordinates for our predictions (i.e., no contextual information such as timestamps and identification tags). As a possible future extension, contextual information can be incorporated to our prediction model for improved performance. Since all our data involves coordinates, we discretize the coordinate space into grid cells for easier data representation and usage.

The methods we implement are our proposed Transformer architecture with $N = 4$ encoder/decoder blocks along with 4

[1]http://trajnet.stanford.edu
[2]https://data.vision.ee.ethz.ch/cvl/aess/dataset
[3]https://graphics.cs.ucy.ac.cy/research/downloads/crowd-data

level stacked LSTM/BiLSTM models, which are currently the state-of-the-art in the destination/trajectory prediction domain. Through our experiments, we ultimately observe that our Transformer model outperforms the LSTM/BiLSTM architectures in terms of quick destination prediction in addition to destination proximity and accuracy.

A. Simulated Indoor Dataset

We first apply our developed Transformer system to a small simulated indoor movement dataset. For this dataset, we use SimTread simulation software to generate movement trajectories in an indoor setting based on a created floor architecture. We generate 11 user movement trajectories with 4 possible destinations. Since this dataset contains a small amount of trajectories, we use the Leave-One-Out Cross-Validation (LOOCV) technique for obtaining average performance results for our tested prediction models. Specifically, given S total samples (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 iteration, we obtain an array of correct destination probabilities as we increase the percentage of our query test trajectory; we then use these probability arrays over all S iterations to obtain *median* destination prediction performance for a given model.

Fig. 2 displays the data split of an example iteration where we train a prediction model using the 10 blue training trajectories, where their starting locations are depicted with dark blue circles, and test the model on the single red trajectory as its observed length evolves. We discretize the continuous space into a 50×50 grid and convert each trajectory into a sequence of grid locations that serve as input to our models. Overall, this data serves to mimic a single user's movement throughout a floor during some time period of a day where the visited destinations can represent popular real-life indoor destinations such as bathrooms, stairs, elevators, and break rooms.

Fig. 3 displays the correct destination probabilities of the three approaches where the top plot considers the median performances along with nearby quantile behavior while the bottom plot displays the destination probabilities for the specific setting shown in Fig. 2 where the true test trajectory destination probability (destination is the location of the green block where the red test trajectory culminates) is shown as a function of the percentage of the observed test trajectory. In the top section, we display the 40^{th}, 50^{th}, and 60^{th} quantiles of destination probabilities to measure variability in test trajectory complexity. When carrying out the LOOCV approach, we encounter test trajectories of varying lengths and similarities to the training trajectories; with this, the destination probabilities we obtain over some splits will differ and a robust destination prediction model should be able to recognize the differences in the trajectories and adapt its predictions accordingly, as can be seen with a larger spread in prediction probabilities.

The top plot shows that the Transformer can yield an improved destination prediction early in the development of a trajectory, unlike the LSTM and BiLSTM methods that require more observation to make accurate predictions. In addition, the variations in correct class probabilities for the Transformer show improved accuracies over the baseline methods. In the bottom section, we consider one split of the LOOCV approach as shown in Fig. 2. In this case, the test trajectory begins to overlap with some training trajectories early in its development; this overlap provides helpful information regarding a destination and should be recognized and exploited by a prediction model. An ideal model would detect the early overlap and would account for this by increasing the true destination probability at around the 20% mark of the test trajectory. This successful behavior is exclusively observed in our Transformer model.

Fig. 2. Indoor data obtained from the SimTread human movement simulation software and used by our destination prediction models. The indoor space is partitioned into 50^2 blocks; each trajectory is converted to a sequence of grid locations that can be used as input for our decision models and each destination is represented as a single grid index. The pictured training and testing splits are for a specific iteration whose results are displayed in the bottom plot of Fig. 3.

Ultimately, these results show that our Transformer approach is able to learn patterns in this simplistic, low-data setting as evidenced by its increasing destination probabilities, unlike the LSTM and BiLSTM approaches that are unable to detect destinations correctly early in the development of a movement trajectory.

B. Taxi Trajectory Dataset

To further investigate the performance of our Transformer destination prediction approach in large real data scenarios, we apply it to the task of outdoor taxi destination prediction. We focus on the popular taxi trajectory dataset used in a 2015 Kaggle competition[4] which is comprised of full trajectories of 442 taxis in the city of Porto, Portugal, recorded over a complete one-year period starting in July 2013. The whole dataset contains a training set of over a million taxi trajectories with a multitude of features, such as trajectory information (in the form of latitude and longitude coordinate sets), taxi ID, and ride time. For our approach, we desire to make a *personalized*

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

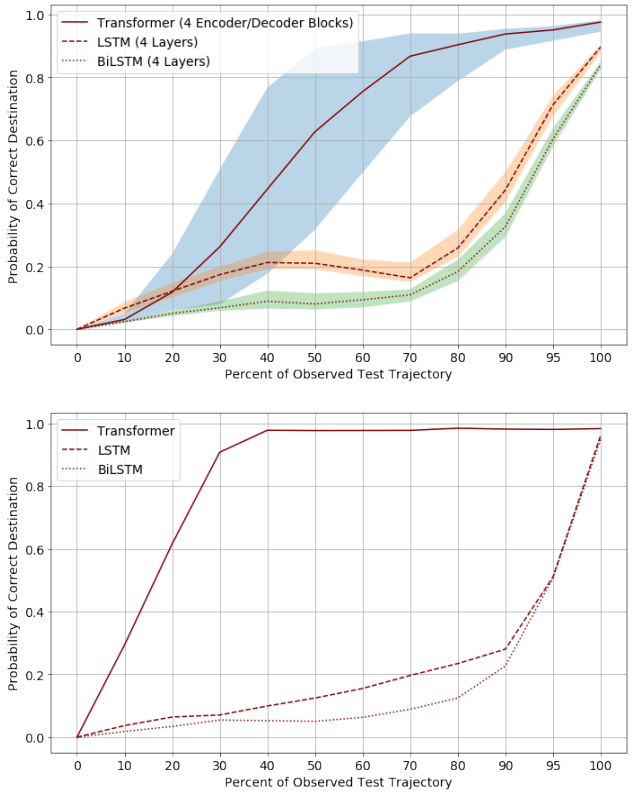

Fig. 3. Evolution of the true destination grid index probability of the Transformer, LSTM, and BiLSTM methods as more of the input test trajectory is observed for the median (top) and test case (bottom) from Fig. 2. The 40^{th} and 60^{th} quantiles of destination probability values are also displayed in the median case to show the robustness of the tested methods in dealing with trajectories of varying characteristics (length, similarity to training set, etc.). Our Transformer is able to accurately detect the true destination early in a trajectory due to its apparent ability to learn in low data settings, unlike with the LSTM and BiLSTM methods.

model for destination prediction, so we must focus on a small subset of the expansive taxi dataset where the data in it can effectively model a single entity's behavior. Specifically, we constrain our data to be within the following latitude and longitude bounds: $[41.135, 41.160]$ and $[-8.600, -8.560]$, respectively. This smaller region concentrates on a specific neighborhood in Porto which we discretize into a 24×24 grid; then, we convert each taxi trajectory of continuous coordinates into a trajectory of grid locations from which we form our training and testing sets.

For our problem, we focus on the case of minimal information, where we solely have access to trajectory information (i.e., no user identification tag or timestamp knowledge), so we form the training and testing sets solely using trajectory data. Our inputs are designed to be partial trajectories, without the destination, whereas the outputs are just the destinations. We form our training set by extracting 100 taxi trajectories per destination over the top 20 most popular destinations in the region; similarly, we create our testing set by extracting 10 trajectories per destination over the top 20 most popular

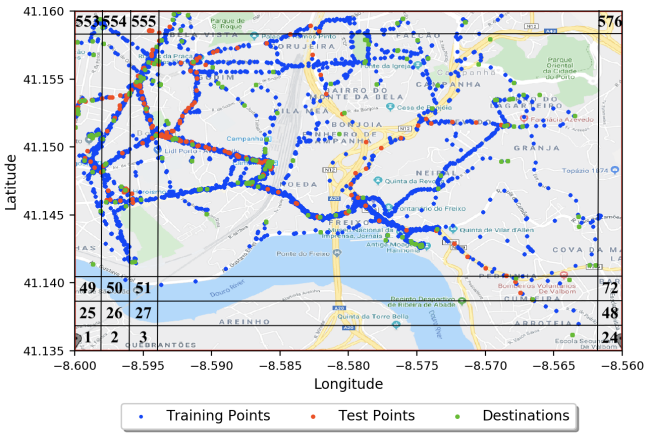

Fig. 4. A visualization of the taxi trajectories that comprise our training and testing sets along with a display of our grid discretization procedure. We partition the Porto neighborhood of interest into 24^2 blocks, which we label and use to convert all the coordinate taxi trajectories into grid index trajectories as the Transformer accepts discrete inputs.

destinations. Fig. 4 displays our trajectory data along with our space discretization scheme, where the non-discretized training and testing trajectories are depicted in blue and red, respectively; the green markers represent non-discretized trajectory destinations.

We train LSTM, BiLSTM, and Transformer models using our custom training set, and then evaluate performance on the test set when query trajectories evolve over time. Specifically, we train on full trajectories and test on partial trajectories of increasing lengths, in order to observe how destination prediction performance changes on average when more of a trajectory is observed. For all tested methods, we search over the entire grid space when attempting to find a destination given a query trajectory. We measure performance using two metrics: 1) Block Distance (BD), and 2) Neighborhood Accuracy (NA).

BD is a metric that 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 can be calculated as: $BD|_{O->T} = \max\{|x_T - x_O|, |y_T - y_O|\}$. NA is also a metric that measures the closeness of a prediction to a true destination; it is an accuracy metric that classifies predictions as correct as long as they are R grid jumps away from the true destination. We obtain these performance metrics by calculating them per inputted partial trajectory (formed by extracting a given percentage of a test point's full trajectory) and averaging all obtained metric values for that given observation percentage.

Fig. 5 displays BD trends of the compared prediction models as the lengths of observed trajectories increase when searching for the top 1 destination. We present results for observed trajectory percentages of over 40%, as the obtained

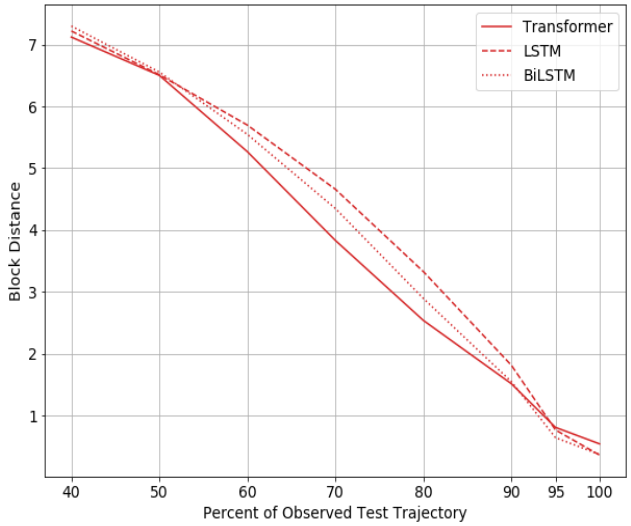

Fig. 5. Block Distance (BD) metric trend as observed trajectories evolve in length when searching for a single destination. We see that the Transformer tends to yield the smallest BD metric over all compared state-of-the-art methods when at least half of the input trajectories are observed.

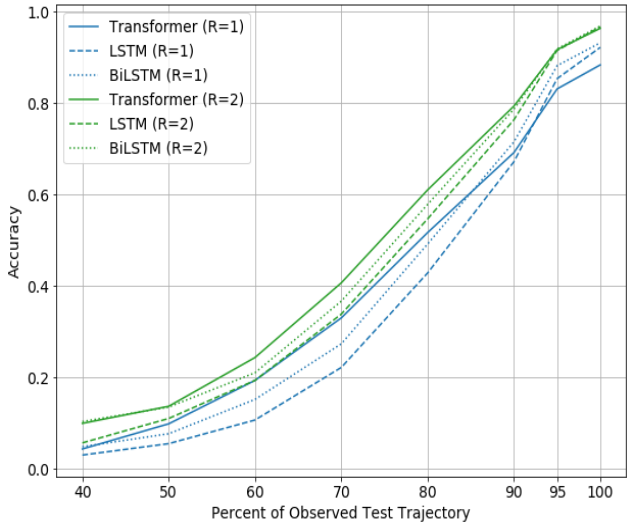

Fig. 6. Behavior of the Neighborhood Accuracy (NA) metric of radius 1 and 2 as observed trajectories evolve in length when searching for a single destination. The Transformer consistently yields larger accuracy values than the state-of-the-art methods for both choices of radius, except when almost the full trajectory has been observed, when all methods tend to perform similarly.

metrics are insignificant for smaller percentages in this taxi application. It is seen that our Transformer consistently outperforms the LSTM/BiLSTM models in terms of BD over much of the trajectory percentages.

Fig. 6 shows NA metric behaviors of the compared models as the lengths of observed trajectories increase when searching for the top 1 destination for both $R = 1$ and $R = 2$ cases. We see that our Transformer yields larger accuracy metrics than the LSTM/BiLSTM models over most of the trajectory percentages, especially when not much of the trajectories has been seen; towards larger percentages, all methods perform quite similarly as the inputs become proximate to the true destinations.

IV. Conclusion

In this paper, we proposed a novel approach to destination prediction based on the Transformer architecture. Numerical experiments showed that our method improved upon the state-of-the-art LSTM/BiLSTM methods' performances in the case of contextless, minimal data for both indoor and outdoor datasets in terms of prediction accuracy metrics.

As future work, we are studying incorporating contextual information into the Transformer architecture, possibly by using a separate context encoder [32] or a simpler two-stage neural network approach that can take as input various data, such as user identification tags and timestamps. Also, a bi-directional Transformer [33] can possibly be used to yield a more robust destination prediction model.

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] 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.

[3] 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.

[4] 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.

[5] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[6] P. Vasishta, D. Vaufreydaz and A. Spalanzani, "Building prior knowledge: A markov based pedestrian prediction model using urban environmental data," in *Proceedings of the 15th International Conference on Control, Automation, Robotics and Vision*, 2018.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] F. D. N. Neto, C. d. S. Baptista and C. E. C. Campelo, "Combining markov model and prediction by partial matching compression technique for route and destination prediction," *Knowledge-Based Systems*, vol. 154, pp. 81–92, 2018.

[13] 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.

[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] 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.

[16] 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.

[17] S. Choi, H. Yeo and J. Kim, "Network-wide vehicle trajectory prediction in urban traffic networks using deep learning," *Transportation Research Record*, vol. 2672, no. 45, pp. 173–184, 2018.

[18] S. Choi, J. Kim and H. Yeo, "Attention-based recurrent neural network for urban vehicle trajectory prediction," *Procedia Computer Science*, vol. 151, pp. 327–334, 2019.

[19] Sheila Alemany, Jonathan Beltran, Adrian Perez, and Sam Ganzfried, "Predicting hurricane trajectories using a recurrent neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 468–475.

[20] 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.

[21] 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.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] 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.

[24] 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.

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

[26] J. Jiang, F. Lin, J. Fan, H. Lv and J. Wu, "A destination prediction network based on spatiotemporal data for bike-sharing," *Complexity*, 2019.

[27] 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.

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

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

[30] 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.

[31] M. Popel and O. Bojar, "Training tips for the transformer model," *The Prague Bulletin of Mathematical Linguistics*, , no. 110, pp. 43–70, April 2018.

[32] J. Zhang, H. Luan, M. Sun, F. Zhai, J. Xu, M. Zhang and Y. Liu, "Improving the transformer translation model with document-level context," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[33] J. Devlin, M. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv e-prints*, Oct 2018.