Translating Human Mobility Forecasting through Natural Language Generation

Hao Xue
hao.xue@rmit.edu.au
School of Computing Technologies, RMIT University
Melbourne, Victoria, Australia

Flora D. Salim
flora.salim@rmit.edu.au
School of Computing Technologies, RMIT University
Melbourne, Victoria, Australia

Yongli Ren
yongli.ren@rmit.edu.au
School of Computing Technologies, RMIT University
Melbourne, Victoria, Australia

Charles L. A. Clarke
charles.clarke@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

ABSTRACT

Existing human mobility forecasting models follow the standard design of the time-series prediction model which takes a series of numerical values as input to generate a numerical value as a prediction. Although treating this as a regression problem seems straightforward, incorporating various contextual information such as the semantic category information of each Place-of-Interest (POI) is a necessary step, and often the bottleneck, in designing an effective mobility prediction model. As opposed to the typical approach, we treat forecasting as a translation problem and propose a novel forecasting through a language generation pipeline. The paper aims to address the human mobility forecasting problem as a language translation task in a sequence-to-sequence manner. A mobility-to-language template is first introduced to describe the numerical mobility data as natural language sentences. The core intuition of the human mobility forecasting translation task is to convert the input mobility description sentences into a future mobility description from which the prediction target can be obtained. Under this pipeline, a two-branch network, SHIFT (Tran**s**lating **H**uman Mob**i**lity **F**orecas**t**ing), is designed. Specifically, it consists of one main branch for language generation and one auxiliary branch to directly learn mobility patterns. During the training, we develop a momentum mode for better connecting and training the two branches. Extensive experiments on three real-world datasets demonstrate that the proposed SHIFT is effective and presents a new revolutionary approach to forecasting human mobility.

CCS CONCEPTS

• **Computing methodologies** → *Natural language generation*; • **Information systems** → *Data mining*.

KEYWORDS

temporal forecasting, natural language, human mobility prediction

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498387

ACM Reference Format:
Hao Xue, Flora D. Salim, Yongli Ren, and Charles L. A. Clarke. 2022. Translating Human Mobility Forecasting through Natural Language Generation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3488560.3498387

1 INTRODUCTION

Human mobility forecasting such as the next location prediction task [8, 25] and the customer flow prediction task [23] are essential ingredients in many domains including human mobility understanding and smart city applications. During the pandemic, support for contact tracing and crowd management has become of critical importance. In the literature, human mobility prediction is invariably addressed through a time-series forecasting framework. In this forecasting framework, the model takes numerical mobility data (*e.g.*, number of visits of each POI) as input and yields a predicted value for some future time period.

The last few years have seen major advances in deep learning techniques for mobility prediction models. In order to better capture human mobility patterns and predict future mobility, these models consider different types of contextual information beyond historical mobility records. For example, the semantic category associated with points of interest (POIS) are incorporated into human mobility prediction models [3, 7, 9] and external information, such as local weather conditions, day of the week, and time, are incorporated into traffic flow prediction models [24, 32, 35].

This prediction workflow is simplified and summarized in the upper half of Figure 1. Multiple data sources providing diverse information (*e.g.*, POI id, semantic information, and the historical mobility) are first passed through several encoders or embedding layers to extract feature vectors. After the encoding process, the extracted/embedded contextual features are concatenated as the input of a predictor/decoder to yield the prediction (*e.g.*, *10* in the example given in the figure). There are two major limitations of this framework: (a) When there are multiple contexts, the concatenation operation may not be the optimal way of merging different data sources. It may be difficult to learn or capture the latent correlations of multiple contexts when multiple features are appended together. (b) Considering the inherent characteristics of different contexts, several different feature encoders or embedding layers are necessary for a prediction model to learn the influence of these contexts.

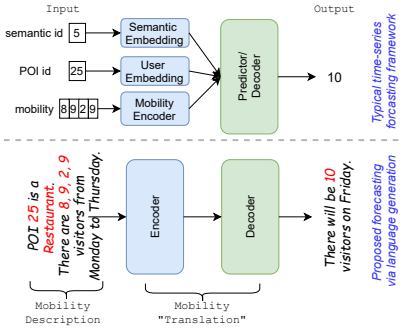


Figure 1: Conceptual illustrations of the typical time-series forecasting framework for human mobility prediction (the upper half) and the proposed forecasting through language generation (the lower half).

These additions may dramatically increase the complexity of the prediction model and makes the model harder to train.

Inspired by the development of natural language processing models, we notice that a neural machine translation structure could be a suitable solution to address the above limitations. Assuming that all types of contextual information and data sources can be described in a natural language sentence, the prediction model then needs only to take the sentence as input without utilizing different encoders or worrying how to combine different contexts. In this paper, we seek to answer the research question: *Can we predict human mobility in a natural language translation manner while maintaining a higher mobility forecasting performance?*

Unlike existing methods for human mobility prediction, we create an unconventional pipeline for mobility translation, essentially translating from historical mobility to future mobility. As illustrated in the lower part of Figure 1, our proposed method for forecasting via a language generation pipeline is a sequence-to-sequence structure. Through a mobility description, both mobility data and other supporting contextual information are transformed into natural language sentences. Then, in a mobility translation step, these descriptive sentences are taken as input and a natural language sentence indicating the prediction is generated as output.

Specifically, in this paper we propose a novel two-branch architecture, SHIFT (Trans**l**ating **H**uman Mob**i**lity **F**orecas**t**ing), for the core mobility translation part under the above human mobility forecasting through language generation pipeline. The architecture consists of a main natural language branch (*NL*) and an auxiliary mobility branch (*Mob*). The *NL* branch is implemented as a sequence-to-sequence structure to "translate" mobility descriptions, whereas the *Mob* branch focuses on learning mobility patterns. The purpose of the auxiliary branch is to further improve the main branch's ability to generate mobility predictions.

In summary, our contributions are three-fold:

- We explore and develop a novel mobility forecasting method through a language generation pipeline. To the best of our knowledge, this is the first work that addresses human mobility prediction task (time-series data forecasting) from the perspective of natural language translation and generation.
- We propose a two-branch network, SHIFT, which has a main branch for language generation and an auxiliary branch for explicitly learning mobility patterns. To connect the two branches, a momentum averaging-based method is also introduced.
- We conduct extensive experiments on three real-world datasets. The results demonstrate the superior performance of our SHIFT and the effectiveness of each of its components.

2 RELATED WORK

For human mobility prediction, existing methods can be categorized into two types: classical methods and deep learning-based methods. Under the classical category, based on ARIMA and Seasonal ARIMA, different methods have been designed [18, 20, 28] for forecasting crowd flow. In addition, the Matrix Factorization is widely applied in next POI recommendation methods including [17, 19, 26].

Deep learning-based methods mostly leverage Recurrent Neural Networks (RNNs) (as well as Long Short Term Memory (LSTM) networks [12] and Gated Recurrent Units (GRU) [4]) for capturing mobility patterns in the observation history sequences. ST-RNN proposed by Liu *et al.* [21] and DeepMove [8] are two popular methods that apply attention mechanisms upon RNN to forecast human mobility. Following this trend, various methods [3, 9, 22, 33] incorporating different context information have been proposed to predict human mobility. Since the introduction of the self-attention-based Transformer architecture [27], it has been applied and achieved great success in many fields such as computer vision [1, 6], audio processing [31], and natural language processing [5, 15]. Recently, based on the effective Transformer, various methods have been designed and introduced for time-series data forecasting [16, 29, 36] and specifically for human mobility prediction [10, 30, 32].

Compared to the above-reviewed human mobility forecasting methods, we reshape the human mobility forecasting task and aim to address the prediction from the perspective of language translation. The proposed method designed for translating human mobility forecasting in this work differs from these existing techniques.

3 METHOD

3.1 Problem Formulation

Assume that there is a set of POIs (place-of-interests) in a city: $\mathbf{U} = \{u_1, u_2, \cdots, u_p\}$. For each POI u, c_u stands for the semantic category information, such as a restaurant or a park. The number of visits in a day t of POI u is represented as x_t^u. The human mobility forecasting problem focused in this work is defined as follows. Given the history record of visiting numbers $\mathbf{X}^u = [x_{t_1}^u, x_{t_2}^u, \cdots, x_{t_{\text{obs}}}^u]$, the goal is to predict the number of visits $\hat{x}_{t_{\text{obs}+1}}^u$ for the next day $t_{\text{obs}+1}$. The ground truth of visiting number is represented as $x_{t_{\text{obs}+1}}^u$ and obs stands for the observation length of the given history visiting record. For simplification, the superscript u (indicating the POI id) is ignored in the rest of the paper.

3.2 Mobility Description

In the proposed forecasting via language generation pipeline, an important step to be addressed is how to describe the mobility data (always available in the numerical format) in natural language. This

Table 1: The template of mobility-to-language description. In the proposed mobility forecasting through language generation pipeline, the input part can be seen as the source sentences and the output part is the destination sentence.

	Description	Template	Example
Input	POI Semantic	Place-of-Interest (POI) $\{u\}$ is a/an $\{c_u\}$.	Place-of-Interest (POI) 81 is a Optical Goods Store.
	Observation Time	From $\{t_1\}$ to $\{t_{\text{obs}}\}$,	From August 26, 2020, Wednesday to August 28, 2020, Friday,
	Mobility Data	there were $\{[x_{t_1}, x_{t_2}, \cdots, x_{t_{\text{obs}}}]\}$ people visiting POI $\{u\}$ on each day.	there were 42, 32, 29 people visiting POI 81 on each day.
	Prediction Target Time	On $\{t_{\text{obs+1}}\}$,	On August 29, 2020, Saturday,
Output	Prediction Results	there will be $\{x_{t_{\text{obs+1}}}\}$ people visiting POI $\{u\}$.	there will be 21 people visiting POI 81.

mobility-to-language transformation provides the source sentences and the destination sentences for the "mobility translator". Given that this work explores how to leverage language models for mobility prediction for the first time, to the best of our knowledge, there is no prior work available for mobility-to-language transformation. Therefore, we first develop a simple yet effective template-based method for mobility description.

Table 1 demonstrates the mechanism of the proposed mobility description method. Generally, there are two parts included: input description generation and output description generation. For the input description, it produces *prompts* that serve as the input natural language sentences of the encoder (blue box in the lower half of Figure 1). As given in the table, the prompt consists of four elements:

- POI Semantic: to give the POI id and describe the semantic category information of the POI;
- Observation Time: to indicate the timestamps of the observation period;
- Mobility Data: to transform the numerical mobility data into natural language, which is the essential part of the prompt;
- Prediction Target: to provide a cue of the prediction target timestamp $t_{\text{obs+1}}$.

By linking all four elements together (the first four rows in Table 1), the entire prompt is then generated.

Similarly, the output description part (used as the ground truth for training and evaluation) handles the targeting sentences which are the expected output of the decoder (green box in the lower half of Figure 1). It has only one sentence and focuses on the prediction goal $x_{t_{\text{obs+1}}}$. One example of the output description is given in the last row of Table 1.

Depending on the available data or the application, other sentences for describing extra information for mobility prediction such as holiday information (*e.g.*, `Tuesday is Boxing Day.`) and weather conditions (*e.g.*, `There were showers on Thursday.`) could also be easily appended in the prompt. For the conventional time-series forecasting frameworks, in order to take various types of extra information into consideration, it is necessary to explicitly design and introduce extra modules or layers such as the external component in [34] and the gating mechanism to fuse external information in [35]. On the contrary, the proposed language generation-based mobility prediction method only needs to update the prompts instead of adding extra layers or tweaking the model architecture. This reflects the flexibility of the proposed forecasting via language generation pipeline.

3.3 Two-Branch Structure

The overall framework of the proposed method is illustrated in Figure 2(c) (Figure 2(a) and Figure 2(b) are two variants of our SHIFT and more details are given in Section 3.3.3). It consists of two branches: (1) Natural Language Branch (*NL*): a branch with the sequence-to-sequence structure, which is the main branch of SHIFT to translate the input prompt to generate output sentences; (2) Auxiliary Mobility Branch (*Mob*): an auxiliary branch to strengthen the ability of SHIFT in learning mobility patterns for forecasting. The details of SHIFT are given in the following sections.

3.3.1 NL Branch. Through mobility description, mobility data $\mathbf{X}$ and other context information (*e.g.*, semantic category c) are transformed as a natural language prompt $\mathbf{S}$. In addition, the prediction target $x_{t_{\text{obs+1}}}$ is also described as a target sentence Y. Following standard natural language processing procedures, tokenization[1] is then applied to the generated prompt sentences.

After the tokenization, the prompt $\mathbf{S}$ is interpreted as a list of tokens $[s_1, s_2, \cdots, s_J]$, where J is the length of the list. Each token (element in the list) belongs to a vocabulary where saves the token mapping of the entire dataset. Similarly, the target sentence Y (*i.e.*, the sentence given in the last row of Table 1) is encoded into $[y_1, y_2, \cdots, y_K]$ and K is the length of the target sentence tokens.

The whole *NL* branch follows the sequence-to-sequence/encoder-decoder structure and the encoding process can be formulated as:

$$e_n^j = \phi_n(s_j; \mathbf{W}_{\phi_n}), \tag{1}$$

$$h_N = f_N(e_n^1, e_n^2, \cdots, e_n^J; \theta_N), \tag{2}$$

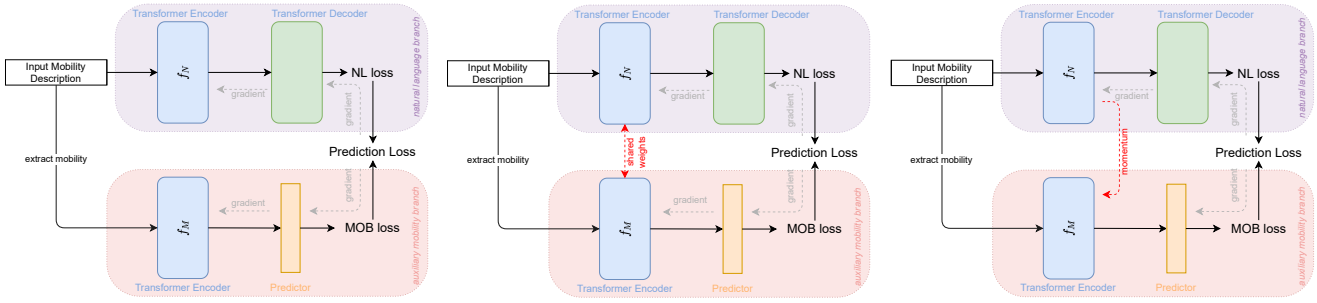
where ϕ_n with weights $\mathbf{W}_{\phi_n}$ is an embedding layer to embed each input token into a d dimension vector $e_n^j \in \mathbb{R}^d$. The encoder $f_N(\cdot)$ with trainable weights θ_N takes embedded vectors to yield a hidden state h_N for the later decoder part. In our SHIFT, Transformer [27] is utilized as the encoder $f_N(\cdot)$.

The decoding part in our *NL* branch generates predicted tokens $[\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K]$ in a autoregressive fashion. Mathematically, the probability of decoding the k-th token $\hat{y}_k$ can be parameterized as:

$$p(\hat{y}_k \mid \hat{y}_{<k}, h_N) = \text{softmax}(f_D(\hat{y}_{<k}, h_N; \theta_D)), \tag{3}$$

where $f_D(\cdot)$ is the decoder in the *NL* branch. After decoding the total K tokens and applying detokenization on decoded tokens, a generated sentence $\hat{Y}$ is then obtained.

[1]Tokenizer provided by HuggingFace is utilized in our implementation: https://huggingface.co/docs/tokenizers/python/latest/.



(a) Basic mode (b) Siamese mode (c) Momentum mode

Figure 2: Conceptual comparison of three modes of connecting two branches in SHIFT (empirical comparisons are presented in Table 4). In SHIFT, the third momentum mode is selected as the default mode.

3.3.2 Mob Branch. Since we are particularly interested in forecasting human mobility (*e.g.*, number of visits of each POI), an auxiliary mobility branch (*Mob* branch) is incorporated into the SHIFT framework. As described in the above section, the *NL* branch is a general sequence-to-sequence architecture for language generation, both mobility data related tokens (*e.g.*, tokens represented the number of visits) and other tokens in the prompt will be treated equally. Therefore, the motivation of introducing this auxiliary branch is to support the main *NL* branch to better learning the mobility pattern.

For the architecture of this *Mob* branch (the lower branch in each sub-figure in Figure 2), it follows the design of typical time-series forecasting framework. The input of this branch is the mobility data $[x_{t_1}, x_{t_2}, \cdots, x_{t_{\text{obs}}}]$ which can be extracted from the input mobility description (prompt) of the *NL* branch or directly taken from the dataset (the raw data before mobility-to-language transformation).

Similar to the *NL* branch, the input of each timestamp $x_t$is first embedded into $e_m^t \in \mathbb{R}^d$ through the embedding layer $\phi_m(\cdot)$:

$$e_m^t = \phi_m(x_t; \mathbf{W}_{\phi_m}) \tag{4}$$

After the embedding, a Transformer-based encoder f_M is used to extract the hidden state h_M:

$$h_M = f_M(e_m^{t_1}, e_m^{t_2}, \cdots, e_m^{t_{\text{obs}}}; \theta_M), \tag{5}$$

where θ_M is the weight matrix of the Transformer encoder in the *Mob* branch. The *Mob* branch prediction $\tilde{x}_{t_{\text{obs+1}}}$ at time step $t^i_{\text{obs+1}}$ is then generated via:

$$\tilde{x}_{t_{\text{obs+1}}} = \text{MLP}(h_M), \tag{6}$$

where $\text{MLP}(\cdot)$ is a multi-layer perceptrons (MLP)-based predictor.

3.3.3 Connecting Two Branches. In this section, we discuss how to connect the *NL* branch and the *Mob* branch in our SHIFT. For our SHIFT, the forecasting performance depends on the main *NL* branch. During the model inference phase, the *Mob* branch will be ignored as the output is in the sentence format. As a consequence, it is more important to learn a better $f_N(\cdot)$ for the *NL* branch. For this purpose and inspired by [11], we introduce a Momentum Mode (as illustrated in Figure 2(c)) to connect two encoders. In more detail, during the training process, only θ_N is updated through back propagation and θ_M is updated via:

$$\theta_M \leftarrow \alpha_m \theta_N + (1 - \alpha_m)\theta_M, \tag{7}$$

where α_m is the momentum factor. Under this mode, the *Mob* branch encoder $f_M(\cdot)$ can be seen as the momentum-based moving average of the *NL* branch encoder $f_N(\cdot)$. Since θ_M is based on θ_N, during the training, the auxiliary *Mob* branch could support the main branch to learn a more powerful $f_N(\cdot)$ in the aspect of encoding mobility data for forecasting.

In addition to the above momentum mode, we also explore and compare the other two ways of connecting the *NL* branch and the *Mob* branch: (i) Basic Mode (Figure 2(a)): this mode is a vanilla mode. There is no interactions between two branches except for the combined loss. (ii) Siamese Mode (Figure 2(b)): the weights of two encoders in two branches are shared during training ($\theta_M = \theta_N$). The comparison of using different modes is given in Section 4.4.

It is worth noting that the final prediction target can be extracted from both the *NL* branch ($\hat{x}_{t_{\text{obs+1}}}$ acquired from the generated sentence $\hat{Y}$) and the *Mob* branch ($\tilde{x}_{t_{\text{obs+1}}}$ in Eq. (6)). Considering that we are interested in performing forecasting through language generation, the overall output of our SHIFT is the generated sentence $\hat{Y}$ from the *NL* branch and $\hat{x}_{t_{\text{obs+1}}}$ embedded in output sentence $\hat{Y}$ is used for evaluation.

3.4 Loss Function

As the *NL* branch is for generating sentences, we use the conventional multi-class cross-entropy loss function (the number of class equals to the total number of tokens in the vocabulary) given by:

$$\mathcal{L}_N = -\sum_{b=1}^{B}\sum_{k=1}^{K} y_k^b \log \hat{y}_k^b, \tag{8}$$

where B is the batch size and the superscript b stands for the b-th training sample in a batch. For the *Mob* branch, it is a basic time-series forecasting branch Thus, we choose the typical mean squared error (MSE) as the loss function:

$$\mathcal{L}_M = \frac{1}{B}\sum_{b=1}^{B} \|\tilde{x}^b_{t_{\text{obs+1}}} - x^b_{t_{\text{obs+1}}}\|^2. \tag{9}$$

As a result, the final loss function of SHIFT is a combination of $\mathcal{L}_N$ and $\mathcal{L}_M$:

$$\mathcal{L} = (1 - \alpha_{loss})\mathcal{L}_N + \alpha_{loss}\mathcal{L}_M, \tag{10}$$

Table 2: Details of three datasets.

	NYC	Dallas	Miami
Collection Start Date	2020-06-15		
Collection End Date	2020-11-08		
Average Visits per Day	17.082	21.520	22.977
Max Number of Visits	246	2746	1550
Total Number of POIs	479	1374	1007
Number of Categories	39	65	51

where α_{loss} is the loss factor to balance the two losses. The impact of setting different α_{loss} is discussed in Section 4.5.2.

4 EXPERIMENTS

4.1 Dataset

We performed extensive experiments on real-world human mobility data presented by SafeGraph's Weekly Patterns[2], which includes visitor and demographic aggregations for POIs in the US. It contains aggregated raw counts (no private information) of visits to POIs from a panel of mobile devices and also provides the semantic category information of each POI. Although SafeGraph provides the data from many cities, we selected data from three major cities with different statistical features (see Table 2 and Figure 6 in the *Supplementary Material*) for building three datasets: New York City (NYC), Dallas, and Miami. Since some POIs only have visiting records for several weeks, we first filter out POIs without complete visiting records during the entire data collection period. The mobility-to-language template introduced in Section 3.2 is then applied to generate natural language sentences to form datasets. Each dataset is randomly divided into the training set (70%), validation set (10%), and testing set (20%). Table 2 shows the statistics (after filtering) of the datasets. Based on the table, it can be seen that three selected datasets have different levels in the total number of POIs, max number of visits, and the number of semantic categories. This ensures the representativeness of our data used for experiments.

4.2 Implementation Details

The hidden dimension d for the Transformer is chosen as 256 for both the main *NL* branch and the auxiliary *Mob* branch. To avoid over-fitting, the dropout rate is set as 0.2. The hyperparameters are set based on the performance of the validation set. The total number of training epochs is 36 with batch size 128 (for the Dallas and Miami) or batch size 64 (for the NYC). The loss factor α_{loss} and the momentum factor α_m are selected as 0.01 and 0.001, respectively. The proposed methods are optimized with Adam optimizer [13] (a 0.0001 initial learning rate with *ReduceLROnPlateau* decay) on a desktop with an NVIDIA GeForce RTX-2080 Ti GPU with PyTorch.

4.3 Prediction Performance

4.3.1 Baselines for Comparison. As comparison, we select 9 methods which are classified into two different categories:

- Time-series forecasting methods: (1) basic linear regression (LR); (2) Gru [4]: Gated Recurrent Units, one of basic RNNs; (3) GruA [2]: Gru with attention mechanism; (4) Transformer [27]: the vanilla Transformer structure. This can considered as a model only using the *Mob* branch of our SHIFT. (5) Reformer [14]: an efficient variant of Transformer; (6) Informer [36]: a state-of-the-art Transformer variant specifically designed for time-series prediction.
- Natural language sequence-to-sequence structure (S2S): (1) using GruA network as the backbone; (2) using Transformer network as the backbone: this can considered as a model only using the *NL* branch of our SHIFT. (3) BART [15]: a recent Transformer-based architecture which has a bidirectional encoder and an autoregressive decoder. It is designed for natural language sequence-to-sequence tasks. Note that the pre-trained weights of this network is not used in the experiments for fair comparison.[3]

For the first category methods, the typical time-series forecasting framework (the upper one in Figure 1) is applied. The proposed forecasting through language generation pipeline (the lower one in Figure 1) is utilized for the methods under the second category.

4.3.2 Evaluation Protocol and Metrics. To evaluate the performance of different methods, we report two widely used metrics for prediction tasks: the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE). For the proposed SHIFT and other S2S-based methods, the direct outputs are sentences (*e.g.*, the last row of Table 1). Thus, $\hat{x}_{t_{\text{obs}+1}}$ is firstly decoded from each output sentence before calculating RMSE and MAE. In the following experiments, we report the average performance (and the standard deviation) of 5 runnings of each method (excluding LR) or configuration.

4.3.3 Performance. Table 3 shows the results of different methods. The average performance across all three datasets is also given in the last two columns of the table. In general, we observe that SHIFT consistently outperforms all baseline techniques in RMSE (12.4% performance gain, compared to the second best) and achieves the second best in average MAE (only about 0.2% worse than the best performer BART). Compared to other methods, SHIFT brings a significant RMSE improvement especially on the Dallas and Miami datasets which are more difficult (due to more POIs and larger range of the number of visits value) to predict. For the MAE metric, our SHIFT is the top performer on Dallas and other top performers are Informer and S2S(BART). Note that although S2S(BART) slightly outperforms our SHIFT on average MAE, the computational cost of S2S(BART) is significantly larger than SHIFT (see Table 6 given in the *Supplementary Material*). These results demonstrate the effectiveness of the proposed SHIFT.

In addition, if we compare methods using the same network architecture, S2S(GruA) leads GruA with an improvement of 11.9% in RMSE and S2S(Transformer) outperforms Transformer by around 5.4% in RMSE. It can be seen that applying the proposed forecasting through language generation pipeline (S2S) is able to boost human mobility forecasting performance and S2S is robust to work with different prediction neural network architectures.

[2]https://docs.safegraph.com/docs/weekly-patterns

[3]The configuration of this model can be accessed through: https://huggingface.co/facebook/bart-base/tree/main.

Table 3: Performance results of different methods on the three datasets. In each row, the best performer is shown in bold and the second best is given in blue.

	NYC		Dallas		Miami		Average	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
LR	9.131	5.639	24.544	6.601	13.081	6.082	15.585	6.107
Gru	7.547 (0.098)	4.550 (0.038)	23.987 (0.262)	5.400 (0.016)	12.125 (0.160)	5.413 (0.026)	14.553	5.121
GruA	7.704 (0.107)	4.464 (0.037)	22.562 (0.433)	5.276 (0.048)	11.465 (0.417)	5.045 (0.107)	13.910	4.928
Transformer	6.714 (0.072)	4.279 (0.058)	18.820 (0.278)	5.166 (0.125)	10.995 (0.181)	5.130 (0.117)	12.176	4.858
Reformer	6.626 (0.061)	4.395 (0.074)	17.392 (0.178)	5.120 (0.037)	10.578 (0.242)	5.117 (0.065)	11.532	4.877
Informer	6.509 (0.073)	**4.248 (0.065)**	19.386 (0.383)	6.717 (0.453)	9.858 (0.171)	5.159 (0.103)	11.918	5.375
S2S(GruA)	6.901 (0.212)	4.290 (0.042)	19.914 (1.259)	5.165 (0.067)	9.964 (0.632)	5.009 (0.055)	12.260	4.821
S2S(Transformer)	6.657 (0.070)	4.286 (0.075)	18.212 (1.422)	5.036 (0.096)	9.672 (0.605)	5.034 (0.105)	11.514	4.785
S2S(BART)	6.645 (0.166)	4.313 (0.232)	18.978 (2.102)	4.968 (0.045)	9.724 (0.307)	**4.834 (0.016)**	11.782	**4.705**
SHIFT	**6.426 (0.067)**	4.274 (0.049)	**15.248 (0.367)**	**4.928 (0.043)**	**8.580 (0.159)**	4.951 (0.028)	**10.085**	4.718

Table 4: The prediction results of the ablation studies on all three datasets, without (w/o) the specifically mentioned branches or using different modes.

	NYC		Dallas		Miami		Average	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
SHIFT (w/o NL branch)	6.714 (0.072)	4.279 (0.058)	18.820 (0.278)	5.166 (0.125)	10.995 (0.181)	5.130 (0.117)	12.176	4.858
SHIFT (w/o Mob branch)	6.657 (0.070)	4.286 (0.075)	18.212 (1.422)	5.036 (0.096)	9.672 (0.605)	5.034 (0.105)	11.514	4.785
SHIFT (basic)	6.493 (0.056)	4.365 (0.057)	18.292 (1.715)	5.037 (0.124)	8.792 (0.254)	5.133 (0.093)	11.192	4.845
SHIFT (siamese)	6.519 (0.077)	4.317 (0.014)	19.311 (1.242)	5.212 (0.091)	8.760 (0.155)	5.020 (0.068)	11.530	4.850
SHIFT	6.426 (0.067)	4.274 (0.049)	15.248 (0.367)	4.928 (0.043)	8.580 (0.159)	4.951 (0.028)	10.085	4.718

4.4 Ablation Study

In this part, we conducted experiments on three datasets with ablation consideration. To evaluate each branch and different connecting modes of SHIFT, the following variants are compared:

- Without the *NL* branch: the *NL* branch of SHIFT is disabled. This variant is the same as Transformer in Table 3.
- Without the *Mob* branch: the *Mob* branch is removed. This variant is the same as S2S(Transformer) in Table 3.
- SHIFT (basic): using Basic mode to connect two branches (details given in Section 3.3.3).
- SHIFT (siamese): using Siamese mode to connect two branches (details given in Section 3.3.3).

The results of these variants and our SHIFT (using the default momentum mode) on the three datasets are given in Table 4. From the table, we can observe that: (1) The proposed SHIFT greatly outperforms the first two variants where only one branch is enabled. It justifies the need of incorporating both branches. (2) The momentum mode shows better performance than the basic and the siamese modes. Specifically, using the siamese mode has the worst performance. It is even worse than SHIFT (w/o Mob branch), which indicates that the auxiliary *Mob* branch has a negative effect on prediction performance under the siamese mode setting. Based on the above results, we conclude that the momentum mode is more suitable for the proposed two-branch SHIFT.

4.5 Impact of Different Settings

4.5.1 Different Prompts. In the proposed forecasting via language generation pipeline, the mobility description is an important factor. We explore the impact of different prompts on mobility forecasting performance. To be specific, two types of prompts are used as the input of our SHIFT:

- Prompt A: This is the default prompt for the proposed method. It contains all the elements listed in Table 1.
- Prompt B: Compared to Prompt A, the sentence used for describing the POI semantic category information (the first row in Table 1) is removed in Prompt B.

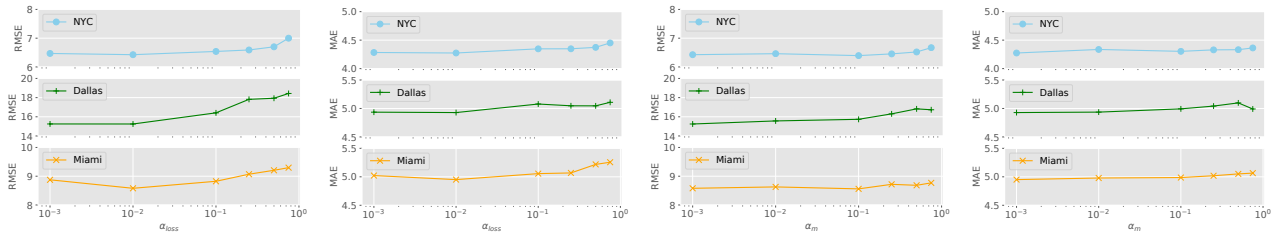
As given in Table 5, using Prompt A consistently produces performance improvements over using Prompt B on all three datasets. This indicates that incorporating external semantic information of POIs is also beneficial to the mobility prediction task under the forecasting via language generation pipeline.

4.5.2 Different Loss Factors. In this experiment, we analysis the impact of the loss factor α_{loss} on the performance of SHIFT by varying α_{loss} from {0.001, 0.01, 0.1, 0.25, 0.5, 0.75}. The average (of 5 runnings) RMSE and MAE of SHIFT with different α_{loss} settings on all three datasets are shown in Figure 3(a) and 3(b), respectively. We can observe that a smaller α_{loss} (< 0.1) leads to a well performance. When a larger α_{loss} is applied, the prediction performance of SHIFT drops considerably. During the training of SHIFT, it can be noticed that $\mathcal{L}_M$ (MSE loss) has a relatively larger value than $\mathcal{L}_N$ (cross-entropy loss). Thus, a smaller α_{loss} could better balance these two loss terms, which results in a better prediction performance.

4.5.3 Different Momentum Factors. In this part, we investigate the impact of the momentum factor by selecting α_m from {0.001, 0.01, 0.1, 0.25, 0.5, 0.75}. The average RMSE and MAE of 5 runnings using

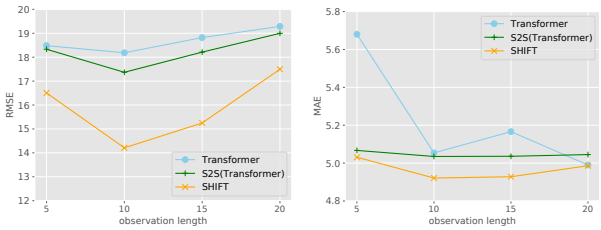
Table 5: The prediction performance of using different Prompts in SHIFT.

	NYC		Dallas		Miami		Average	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Prompt A	6.426 (0.067)	4.274 (0.049)	15.248 (0.367)	4.928 (0.043)	8.580 (0.159)	4.951 (0.028)	10.085	4.718
Prompt B (w/o semantic)	6.496 (0.091)	4.301 (0.047)	16.035 (0.633)	5.032 (0.012)	8.688 (0.223)	4.982 (0.041)	10.406	4.772



(a) Loss Factor: RMSE (b) Loss Factor: MAE (c) Momentum Factor: RMSE (d) Momentum Factor: MAE

Figure 3: The impact of loss factor α_{loss} and momentum factor α_m.



(a) RMSE (b) MAE

Figure 4: The performance of different observation lengths.

different α_m setting is given in Figure 3(c) and 3(d). From these two figures, it can be seen that SHIFT performs reasonably well α_m is small (ranging from 0.001 to 0.1). When the α_m becomes larger, we observe that there is an increasing RMSE for NYC and Dallas, whereas the performance is relatively stable with all α_m values for the Miami dataset. From these results, although a smaller α_m (a slowly updating f_M for the *Mob* branch) is beneficial, the impact of the momentum factor α_m is less than the loss factor α_{loss}.

4.5.4 Different Observation Lengths. In the last part, we evaluate the performance of SHIFT with different observation lengths. Specifically, we compare the performance of Transformer, S2S(Transformer), and SHIFT with the observation length as 5, 10, 15, 20, respectively. Due to the large amount of experiments (3 methods and 4 different observation lengths), we only report the performance (average of 5 runnings) using the challenging Dallas dataset in Figure 4(a) (RMSE) Figure 4(b) (MAE). From the figure, it can be seen that our SHIFT outperforms both Transform and S2S(Transformer) on all different observation lengths. Such an observation demonstrates the superior prediction performance of SHIFT. We can also observe that all three methods have relatively worse results when the observation length is too small (obs = 5) or too large (obs = 20). A smaller observation length leads to fewer available history records to learn mobility patterns, whereas a very large observation length might also increase the difficulty to discover effective mobility patterns for prediction. It suggests that a favorable observation length for SHIFT is roughly one or two weeks.

4.6 Visualization Analysis

In Figure 5, we visualize the attentions (between the input sentence and the output sentence) learned by the Transformer encoder-decoder architecture in SHIFT. The upper half of Figure 5 is the first case and the lower half gives the second case. In each half (each case), the first row is the learned attentions of S2S(Transformer) (SHIFT without the *Mob* branch), whereas the second row illustrates the learned attentions of our SHIFT (both *NL* and *Mob* branches). For each heatmap plot in which a hotter region means a larger attention value, the horizontal axis stands for the input prompt (in the token format) and the vertical axis represents the output sentence tokens. In more detail, `<s>`, `</s>`, Ċ are the sentence starting token, sentence ending token, and padding token, respectively.

4.6.1 Case Analysis 1. The ground truth label of this case is: `There will be 9 people visiting POI 284.` From the upper half of the figure, it can be seen that S2S(Transformer) generates `There will be 4 people visiting POI 284.` and the SHIFT predicts the number of visits as 8. As a comparison, the prediction of only using the *Mob* branch (Transformer method in Table 3, not shown in the figure) is 7.71. From the visualization of S2S(Transformer) and SHIFT, we observe that: (1) When generating the POI id (*e.g.*, 284) in the output sentence, both models would look into not only the POI id given in the prompt but also the semantic information (*e.g.*, higher attention values on the `Office` and `Stationery` tokens). (2) The attentions of S2S(Transformer) has an evener distribution across the input and output tokens, whereas the attentions learned by SHIFT focus more on the mobility data (the `Ġ8` row). Due to the *Mob* branch, SHIFT would particularly learn mobility patterns from the mobility data tokens (*i.e.*, the number of visits values in the input sentence), which leads to high attentions

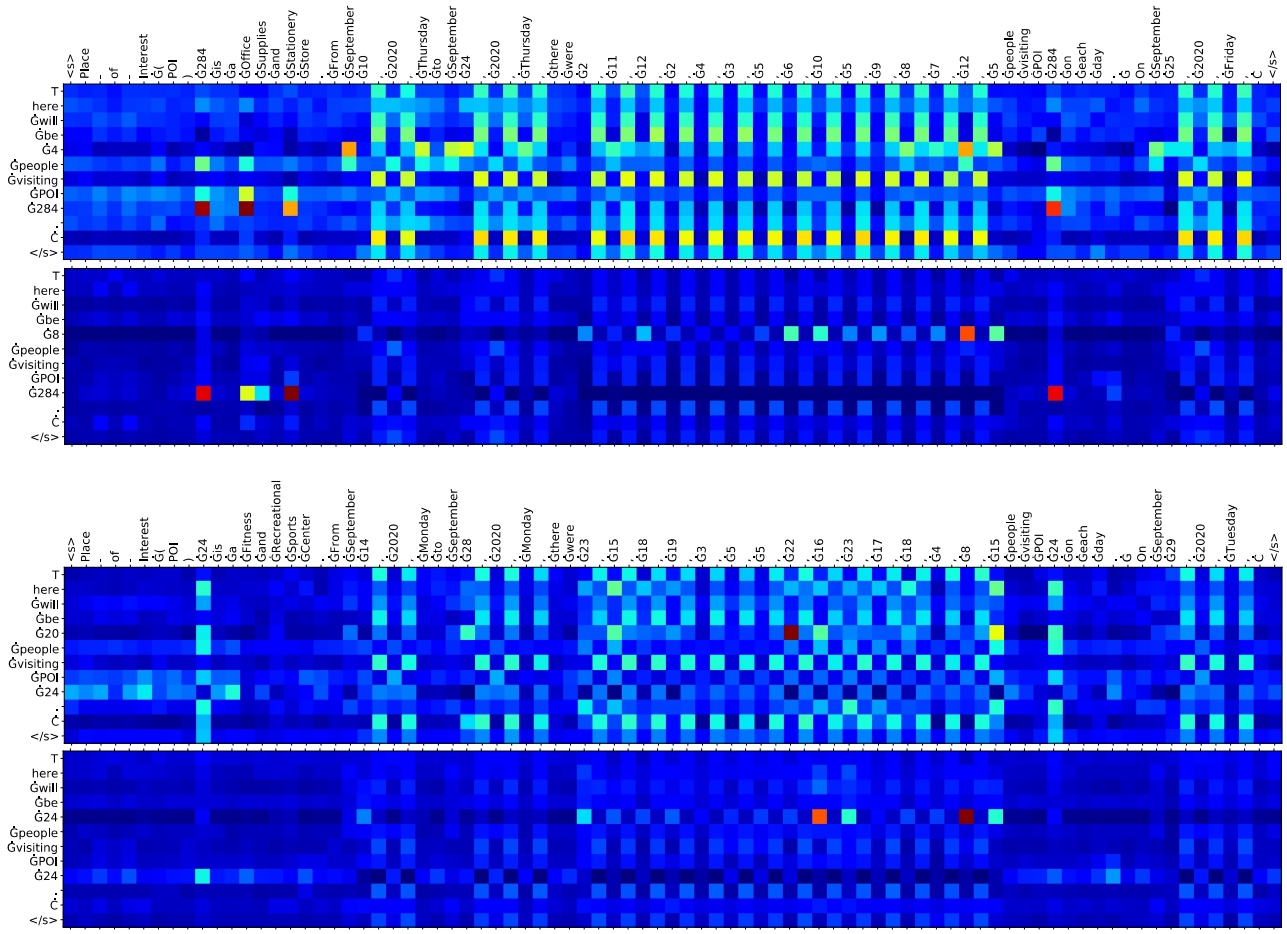


Figure 5: Visualization of two cases (one in the upper half and one in the lower half). For each case, the first row shows the attention of S2S(Transformer) and the attention of SHIFT is presented in the second row. (Better viewed in color.)

on the mobility tokens and a better prediction performance. This result supports our motivation of introducing the *Mob* branch.

4.6.2 Case Analysis 2. The ground truth label of this case is: `There will be 24 people visiting POI 24.` and the Transformer method yields a prediction of 18.41 for this example. As the POI id and the predicted number of visits are the same value (both 24), this case is more difficult. It requires the model to distinguish the same number with different meanings. From the first row of the second case (lower half of Figure 5), it can be noticed that the `Ġ20` row (predicted mobility) has a relatively high attention on the POI id in the input tokens (`Ġ24` columns), while the `Ġ24` row (predicted POI id) has very low attention values on the input POI id tokens. It indicates that the S2S(Transformer) cannot well distinguish the mobility and the POI id in this case. This further results in worse prediction performance. However, with the help of the auxiliary *Mob* branch, SHIFT still can recognize and concentrate on the number of visits (the mobility data part) for this challenging case (the last row of Figure 5). This demonstrates the superior prediction performance of the proposed SHIFT.

5 CONCLUSION

We address the human mobility forecasting problem in a natural language translation manner. Based on our mobility description template, mobility data is transformed into natural language sentences. Through a sequence-to-sequence mobility translation, a sentence indicating the predicted mobility is then generated as output. Furthermore, we have designed a two-branch SHIFT architecture to perform mobility translation. Through extensive experiments, the results illustrate that SHIFT is effective for the human mobility forecasting task. The effectiveness of each branch and the momentum mode in SHIFT are also demonstrated in ablation studies. In the future, we will focus on developing a method for automatic mobility-to-language generation, which aims at providing diverse prompts for the language generation-based mobility prediction.

ACKNOWLEDGMENTS

We would like to acknowledge the support Australian Research Council (ARC) Discovery Project *DP190101485*. We thank SafeGraph for providing free access to the mobility data.

REFERENCES

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691* (2021).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.0473

[3] Yile Chen, Cheng Long, Gao Cong, and Chenliang Li. 2020. Context-aware deep model for joint mobility and time prediction. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 106–114.

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555* (2014).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=YicbFdNTTy

[7] Jie Feng, Yong Li, Zeyu Yang, Qiang Qiu, and Depeng Jin. 2020. Predicting Human Mobility with Semantic Motivation via Multi-task Attentional Recurrent Networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[8] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 world wide web conference*. 1459–1468.

[9] Qing Guo, Zhu Sun, Jie Zhang, and Yin-Leng Theng. 2020. An attentional recurrent neural network for personalized next location recommendation. In *Proceedings of the AAAI Conference on artificial intelligence*, Vol. 34. 83–90.

[10] Sajal Halder, Kwan Hui Lim, Je rey Chan, and Xiuzhen Zhang. 2021. Transformer-based Multi-task Learning for euing Time Aware Next POI Recommendation. (2021).

[11] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[14] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rkgNKkHtvB

[15] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7871–7880.

[16] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems* 32 (2019), 5243–5253.

[17] Xutao Li, Gao Cong, Xiao-Li Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. 2015. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 433–442.

[18] Xiaolong Li, Gang Pan, Zhaohui Wu, Guande Qi, Shijian Li, Daqing Zhang, Wangsheng Zhang, and Zonghui Wang. 2012. Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science* 6, 1 (2012), 111–121.

[19] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 831–840.

[20] Marco Lippi, Matteo Bertini, and Paolo Frasconi. 2013. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems* 14, 2 (2013), 871–882.

[21] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Thirtieth AAAI conference on artificial intelligence*.

[22] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. 2021. STAN: Spatio-Temporal Attention Network for Next Location Recommendation. In *Proceedings of the Web Conference 2021*. 2177–2185.

[23] Shaohui Ma and Robert Fildes. 2020. Forecasting third-party mobile payments with implications for customer flow prediction. *International Journal of Forecasting* 36, 3 (2020), 739–760.

[24] Congcong Miao, Jiajun Fu, Jilong Wang, Heng Yu, Botao Yao, Anqi Zhong, Jie Chen, and Zekun He. 2021. Predicting Crowd Flows via Pyramid Dilated Deeper Spatial-temporal Network. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 806–814.

[25] Congcong Miao, Ziyan Luo, Fengzhu Zeng, and Jilong Wang. 2020. Predicting Human Mobility via Attentive Convolutional Network. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 438–446.

[26] Xingyi Ren, Meina Song, E Haihong, and Junde Song. 2017. Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation. *Neurocomputing* 241 (2017), 38–55.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[28] Billy M Williams and Lester A Hoel. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of transportation engineering* 129, 6 (2003), 664–672.

[29] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *arXiv preprint arXiv:2106.13008* (2021).

[30] Xian Wu, Chao Huang, Chuxu Zhang, and Nitesh V Chawla. 2020. Hierarchically structured transformer networks for fine-grained spatial event forecasting. In *Proceedings of The Web Conference 2020*. 2320–2330.

[31] Hao Xue and Flora D. Salim. 2021. Exploring Self-Supervised Representation Ensembles for COVID-19 Cough Classification. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 1944–1952.

[32] Hao Xue and Flora D. Salim. 2021. TERMCast: Temporal Relation Modeling for Effective Urban Flow Forecasting. In *Advances in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11-14, 2021, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12712)*, Kamal Karlapalem, Hong Cheng, Naren Ramakrishnan, R. K. Agrawal, P. Krishna Reddy, Jaideep Srivastava, and Tanmoy Chakraborty (Eds.). Springer, 741–753. https://doi.org/10.1007/978-3-030-75762-5_58

[33] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudre-Mauroux. 2020. Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States!. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. 2184–2190.

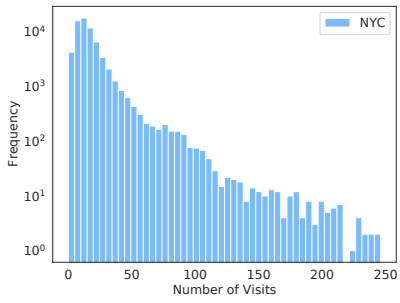
[34] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*.

[35] Junbo Zhang, Yu Zheng, Junkai Sun, and Dekang Qi. 2019. Flow prediction in spatio-temporal networks based on multitask deep learning. *IEEE Transactions on Knowledge and Data Engineering* 32, 3 (2019), 468–478.

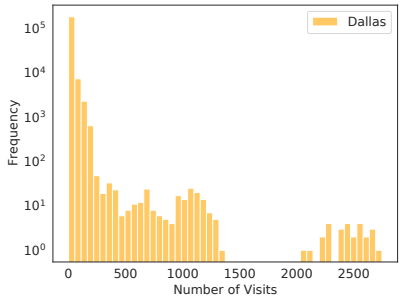
[36] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*.

Table 6: Comparison of computational cost.

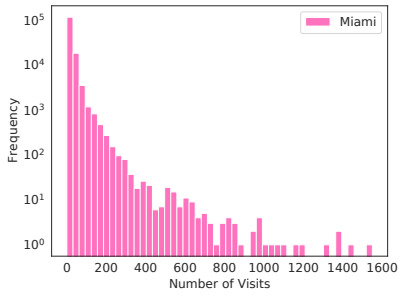
	GPU memory (MB)	# Parameters
Gru	879	0.396×10^6
GruA	911	0.922×10^6
Transformer	1009	3.194×10^6
Reformer	1077	4.342×10^6
Informer	1007	4.410×10^6
S2S(GruA)	1450	3.919×10^6
S2S(Transformer)	1455	4.898×10^6
S2S(BART)	6743	139.42×10^6
SHIFT (Ours)	1469	4.898×10^6



(a) NYC



(b) Dallas



(c) Miami

Figure 6: The distribution of the number of visits in three datasets.

A SUPPLEMENTARY MATERIAL

A.1 Dataset Distribution

In addition to the statistics of three datasets listed in Table 2, the distribution plots of the number of visits are presented in Figure 6. Based on these plots, we can see that the distribution of three cities are different. As shown in the main paper, the proposed SHIFT achieves good forecasting results on all three datasets, which further demonstrates the robustness of our SHIFT.

A.2 Computational Cost

In this section, we analysis the computational cost of SHIFT. Table 6 lists the GPU memory usage (in MB) during training and the number of trainable parameters of each method. These statistics are benchmarked while training each model on the NYC dataset. Generally, the computational cost of language-based models (both S2S and SHIFT) are larger than numerical value-based forecasting models. Also, Transformer-based models require more resources than GRU-based models. These two observations are as expected. Among all language-based methods, the cost of our SHIFT is very close and comparable to S2S(GruA) and S2S(Transformer), whereas the cost of S2S(BART) is significantly larger than others. From the table, we also notice that the number of trainable parameters of SHIFT is almost the same as S2S(Transformer) while SHIFT takes a little bit more GPU memory. Due to the extra *Mob* branch in SHIFT, it takes more memory during training. However, since the *Mob* branch encoder is updated in the momentum mode, this branch does not introduce many trainable parameters.

A.3 Pseudo-code of SHIFT

In Algorithm 1, the pseudo-code of SHIFT training process (using one epoch as example) is presented in the PyTorch-like style.

Algorithm 1 Pseudo-code of training SHIFT (PyTorch-like)

ϕ_n, f_N, f_D: the embedding layer, encoder, and decoder of the *NL* branch
ϕ_m, f_M, MLP: the embedding layer, encoder, and predictor of the *Mob* branch
α_{loss}, α_m: loss factor and momentum factor

$\theta_M = \theta_N$ ▷ Momentum updating initialization
for $(\mathbf{X}, Y, x_{t_{\text{obs}+1}}, \hat{Y})$ in train_data_loader **do** ▷ Loading a batch of training data
  $h_N = f_N(\phi_n(Y))$ ▷ *NL* branch encoding, Eqs. (1) & (2)
  $\hat{Y} = f_D(h_N)$ ▷ *NL* branch decoding, Eq. (3)
  $h_M = f_M(\phi_m(\mathbf{X}))$ ▷ *Mob* branch encoding, Eqs. (4) & (5)
  $\tilde{x}_{t_{\text{obs}+1}} = \text{MLP}(h_M)$ ▷ *Mob* branch prediction, Eq. (6)
  $\mathcal{L}_N = \text{CrossEntropy}(\hat{Y}, Y)$ ▷ *NL* branch loss, Eq. (8)
  $\mathcal{L}_M = \text{MSE}(\tilde{x}_{t_{\text{obs}+1}}, x_{t_{\text{obs}+1}})$ ▷ *Mob* branch loss, Eq. (9)
  $\mathcal{L} = (1-\alpha_{loss})\mathcal{L}_N + \alpha_{loss}\mathcal{L}_M$ ▷ Total loss, Eq. (10)
  $\mathcal{L}$.backward() ▷ Back propagation
  update(SHIFT.params) ▷ Update SHIFT parameters except for θ_M
  $\theta_M \leftarrow \alpha_m\theta_N + (1-\alpha_m)\theta_M$ ▷ Momentum updating *Mob* branch, Eq. (7)
end for