Pre-training Contextual Location Embeddings in Personal Trajectories via Efficient Hierarchical Location Representations

Chung Park[1,2], Taesan Kim[1], Junui Hong[1,2], Minsung Choi[1]
, and Jaegul Choo[2]✉

[1] SK Telecom, Seoul, Republic of Korea
{skt.cpark, ktmountain, skt.juhong, ms.choi}@sk.com
[2] Kim Jaechul Graduate School of AI, KAIST, Daejeon, Republic of Korea
{cpark88kr, secondrun3, jchoo}@kaist.ac.kr

Abstract. Pre-training the embedding of a location generated from human mobility data has become a popular method for location based services. In practice, modeling the location embedding is too expensive, due to the large number of locations to be trained in situations with fine-grained resolution or extensive target regions. Previous studies have handled less than ten thousand distinct locations, which is insufficient in the real-world applications. To tackle this problem, we propose a Geo-Tokenizer, designed to efficiently reduce the number of locations to be trained by representing a location as a combination of several grids at different scales. In the Geo-Tokenizer, a grid at a larger scale shares the common set of grids at smaller scales, which is a key factor in reducing the size of the location vocabulary. The sequences of locations preprocessed with the Geo-Tokenizer are utilized by a causal location embedding model to capture the temporal dependencies of locations. This model dynamically calculates the embedding vector of a target location, which varies depending on its trajectory. In addition, to efficiently pre-train the location embedding model, we propose the Hierarchical Auto-regressive Location Model objective to effectively train decomposed locations in the Geo-Tokenizer. We conducted experiments on two real-world user trajectory datasets using our pre-trained location model. The experimental results show that our model significantly improves the performance of downstream tasks with fewer model parameters compared to existing location embedding methods.

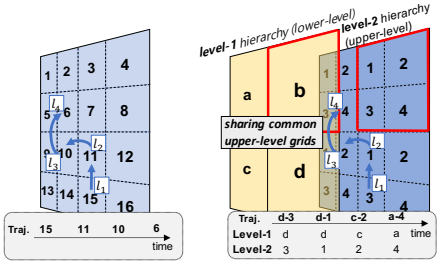
Keywords: Pre-trained Causal Location Embedding · Hierarchical Auto-regressive Location Model · Spatial Hierarchy.

1 Introduction

For modeling human mobility patterns using large-scale mobility data, pre-training location embeddings using a self-supervised objective has advantages, because it allows comprehensive information about locations to be incorporated [7]. The pre-trained location embedding models can also be shared by a

wide range of downstream models, such as those used for next location prediction or transportation mode classification, to improve the prediction performance as well as enhance computation efficiency [16].

Many previous studies have applied language-modeling-based approaches to spatial-temporal datasets [24, 7]. For example, in DeepMove [24], the latent representations of places are trained by applying the skip-gram of word2vec [9] to user trajectories. CTLE [7] is a self-attention based location embedding model that considers a target location's contexts. However, these previous studies still have limitations, as follows: First, the approaches are not scalable to real-world applications, which require numerous locations to be trained. With the fine-grained resolution or extensive target regions, the number of distinct locations, the so-called **location vocabulary**, increases. This deteriorates the quality and efficiency of the pre-trained embedding model because of the heavy embedding layer to be trained. However, previous studies including Geo-Teaser [20], TrajFormer [6], and CTLE [7] train the location embedding model with less than ten thousand locations. Second, locations in a trajectory are often dependent on previously visited locations, meaning that the likelihood of visiting a specific location might be influenced by the locations stayed before [7, 23]. These dependencies can be short-term (e.g., dependencies between consecutive locations) or long-term (e.g., dependencies spanning multiple locations), and they are crucial factors for modeling the context-aware location embedding model. However, previous studies have had difficulty capturing this sequential dependence between locations in their models.



(a) Non-hierarchical (b) 2-levels hierarchies

Fig. 1: An illustration of spatial hierarchies at different scales ($H = 2$). (a) A trajectory with a non-hierarchical case is described. (b) In our hierarchical case, each grid in the level-1 hierarchy shares the common grids set of $1, 2, 3, 4$ in the level-2 hierarchy. For instance, *Grid* 6 in the non-hierarchical case can be represented as (*Grid a*+*Grid* 4) in the hierarchical case.

In order to tackle the discussed problems, we suggest a pre-trained location embedding model to efficiently handle numerous location vocabularies in various real-world applications. First, we devised the **geo-tokenizer embedding layer**, which represents a particular location as a combination of multiple grids at different scales to reduce the number of locations to be trained. In this scheme, a specific location is represented as the combination of the H tokens. For example, as the location is composed of two hierarchies' grids in Figure 1, its final representation is calculated by an element-wise sum of two hierarchies' grid embeddings. Note that in our model, a grid in a lower (i.e., coarser-grained) hi-

erarchy shares the common set of grids in upper (i.e., finer-grained) hierarchies, which is a key factor in reducing the location vocabulary size.

Second, we designed a **causal location embedding model** consisting of the stack of the transformer decoder [14]. The transformer decoder inherently models temporal relationships due to its auto-regressive nature. This allows the model to capture the sequential patterns in the trajectory. Therefore, we dynamically calculate the embedding of a target location considering its temporal order, which varies depending on its trajectory.

Lastly, to pre-train our location embedding model, we modified the Auto-regressive Language Model (ALM) objective introduced in the transformer [14]. Since a grid in the lower hierarchy shares the those of the upper hierarchies in our model, specific two locations with far distance would have same lower-level (e.g., coarser-grained) embeddings despite that they may have different semantics or functionalities. To solve this problem, we devised a **Hierarchical Auto-regressive Location Model (HALM)**. This incorporated information from the lower-level hierarchies into the upper-level hierarchies when implementing ALM tasks to propagate the predicted output of lower-level hierarchies to the upper-level hierarchies. These components are incorporated in our model, as shown in Figure 2. As a result, our location embedding model has relatively fewer parameters to learn and less computational cost than other competitive baselines. In addition, it allows downstream task performance, such as next location prediction or transportation mode classification, to be improved with faster training and inference speed.

2 Preliminaries

A supplementary material (Appendix) with more details about the model, datasets and experiments is available at Github[1].

Definition 1. Trajectory: A trajectory is a sequence of locations where a person stays for a predefined time period [13, 22]. We set each location as a grid shape and l_t as the t-th grid-shaped location. Then, the sequence of visiting locations, denoted as a trajectory, can be defined as follows,

$$s = \{l_0, l_1, \ldots, l_T\} \tag{1}$$

where T is the length of the trajectory s and l_0 is the special token `SOS` which indicates the start of the trajectory. We also denote S as a set of trajectories. We define the **location vocabulary** as the set of locations appearing in the train dataset, and denote it as L. The size of L is the vocabulary size of locations, denoted as $|L|$.

Definition 2. Spatial Hierarchy: Suppose that we set H-levels of **spatial hierarchies** $\{1, 2, ..., H\}$, where the level-h hierarchy consists of grids with sizes of r_h meters. The uppermost hierarchy level H has the smallest scale of r_H, thus the upper-level hierarchy has finer-grained grids than the lower-level.

[1] https://github.com/cpark88/ECML-PKDD2023

Each grid in the level-(h-1) hierarchy is divided into a common grid set in the level-h hierarchy, and the t-th location l_t can be represented with a combination of H grids at different scales. From this spatial hierarchy, the location l_t can be decomposed into the tuple of grids $(l_t^1, l_t^2, ..., l_t^H)$. Therefore, trajectory s consisting of decomposed locations from different hierarchies can be re-defined as follows,

$$s = \{(l_0^1, l_0^2, ..., l_0^H), (l_1^1, l_1^2, ..., l_1^H), \ldots, (l_T^1, l_T^2, ..., l_T^H)\} \tag{2}$$

where l_t^h is a level-h grid at the t-th step and l_0^h is the `SOS` token of the level-h hierarchy. We define the set of all level-h grids appearing in the train dataset as the location vocabulary of level-h hierarchy, and denote it as L^h. The size of L^h is denoted as $|L^h|$. Since $|L| \geq \sum_{h=1}^{H} |L^h|$, using the trajectories of decomposed locations is significantly efficient.

Problem Statement. Pre-training Location Embedding Model for Hierarchically Decomposed Locations: Our goal is to pre-train a location embedding model u to calculate a contextual embedding vector $k(l_t)$ by predicting a next location $l_{t+1} = (l_{t+1}^1, l_{t+1}^2, ..., l_{t+1}^H)$ given its context $s_{<t+1} = \{(l_0^1, l_0^2, ..., l_0^H), (l_1^1, l_1^2, ..., l_1^H), \ldots, (l_t^1, l_t^2, ..., l_t^H)\}$ with H hierarchies. We pre-train our model in a self-supervised manner as shown in Figure 2.

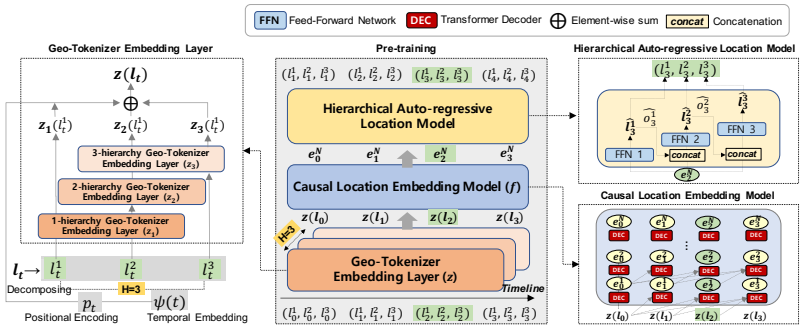


Fig. 2: We display the pre-training process of our model. The model with three level hierarchies(H) case is illustrated. Note that $k(l_2) = f(z(l_0), z(l_1), z(l_2)) = u(s_{<3}) = e_2^N$ as described in Equation 5 and 7.

3 Model

3.1 Geo-tokenizer Embedding Layer

We propose the geo-tokenizer embedding layer, which allows the location embeddings to be trained efficiently with a reduced number of location tokens. By employing spatial hierarchies with different grid sizes, we can potentially

capture varying levels of spatial patterns. We first transform the input sequence into an embedding vector sequence. As shown in Figure 2, we fetch an input latent representation $z(l_t)$ for t-th location l_t from the embedding layer z. We call the embedding layer z the geo-tokenizer embedding layer. The embedding vector $z(l_t)$ can be described as follows,

$$z(l_t) = (\sum_{h=1}^{H} z_h(l_t^h)) + p_t + \psi(t) \tag{3}$$

where z_h is a fully-connected embedding layer of the level-h hierarchy, l_t^h is a grid of the level-h hierarchy at the t-th step, and p_t is the t-th item of the positional encoding (PE) introduced in Transformer [14]. The PE has an important role to capture the relative temporal position in the sequence. In addition, inspired by the previous study [5], we devised the temporal embedding $\psi(t)$. For trajectories, the visiting records have temporal information which may significantly determine predicted locations. $\psi(t)$ is calculated as follows,

$$\psi(t) = \phi(log(r_t)W_d + b_d), \tag{4}$$

where ϕ is a nonlinear activation function (e.g., ReLU), and r_t is an absolute timestamp at the t-th time step such as the real number in Unix Time. W_d is the trainable parameters for linearly transforming $log(r_t)$ and b_d is the bias term. The log transformation is conducted with r_t to effectively cover the wide numerical range of temporal value [5]. The dimension of $\psi(t)$ is equal to that of $z_h(l_t^h)$ and p_t. This procedure generates an input sequence embedding $\{z(l_0), z(l_2), ..., z(l_t)\}$ for the causal location embedding model we will discuss in the next section. Therefore, each embedding layer is represented by a matrix $z_h \in \mathbb{R}^{|L^h| \times W}$, where $|L^h|$ is the size of the vocabulary in the level-h hierarchy, and W is the embedding dimension.

3.2 Causal location embedding model

The context of a target location can be obtained by the sequence of other locations before the target location in a trajectory. From this perspective, we propose a causal location embedding model, which calculates a location's latent representation by considering its contextual neighbors. As shown in Figure 2, given a $(t+1)$-th target location l_{t+1} and its context $s_{<t+1}$, we generate t-th location's final embedding vector $k(l_t)$ by using the casual location embedding model f and the geo-tokenizer embedding layer z, denoted as follows:

$$\begin{aligned} k(l_t) &= f(z(l_0), z(l_1), ..., z(l_t)) \\ &= u(l_0, l_1, ..., l_t), \\ &= u(s_{<t+1}), \end{aligned} \tag{5}$$

where u is our total location embedding model. The embedding vector $k(l_t)$ is t-th item in output vectors' sequence of u. Therefore, the embedding vector of l_t is dynamically generated depending on the context $s_{<t+1}$.

The causal location embedding model f consists of the stack of the transformer decoder [14]. Due to the sequential nature of a trajectory, the model should take into account only the first t items when predicting the (t+1)-th item. This can consider the causal correlations of a target location and its contexts. In addition, compared to the traditional sequential models such as LSTM [4], it has the advantage of the long-term dependency and the parallelization with sequential datasets such as trajectories. Also, unlike previous studies using the transformer encoder structure [7, 10], our model processes location information sequentially and can better handle both short-term and long-term dependencies in the trajectory (See the Appendix A.5).

Specifically, the input sequence embedding $\{z(l_0), z(l_2), ..., z(l_t)\}$ calculated in the geo-tokenizer embedding layer, is then fed into the causal location embedding model f, which is the stack of the transformer decoder. A multi-head self-attention module with a causality mask and a feed-forward network are inherent in each transformer decoder [14]. This process is described as:

$$\begin{aligned} &\{\mathbf{e}_0^{(k)}, \mathbf{e}_1^{(k)}, ..., \mathbf{e}_t^{(k)}\} \\ &= \mathbf{Decoder}(\{\mathbf{e}_0^{(k-1)}, \mathbf{e}_1^{(k-1)}, ..., \mathbf{e}_t^{(k-1)}\}), \\ &\{\mathbf{e}_0^{(0)}, \mathbf{e}_1^{(0)}, ..., \mathbf{e}_t^{(0)}\} = \{z(l_0), z(l_1), ..., z(l_t)\}, \end{aligned} \tag{6}$$

where the **Decoder** represents the transformer decoder. The output sequence of the k-th layer and the input sequence of the (k+1)-th layer are the same as $\{\mathbf{e}_0^{(k)}, \mathbf{e}_1^{(k)}, ..., \mathbf{e}_t^{(k)}\}$. We stack the N transformer decoders in our causal location embedding module f. The t-th item of the N-th transformer decoder is denoted as $\mathbf{e}_t^N$, which is the causal embedding vector of the location l_t. In short, the final output vector of the location l_t in the N stack of the Decoder can be represented as:

$$k(l_t) = \mathbf{e}_t^N. \tag{7}$$

3.3 Pre-training Hierarchical Auto-regressive Location Model

The relationship between target locations and their corresponding contexts should be considered in the location embedding model. For this purpose, we propose the novel variant of the Auto-regressive Language Model (ALM) objective introduced in the transformer [14, 11, 12]. In this paper, since we predict the next location in our pre-trained model, the ALM is rewritten as the Auto-regressive Location Model. The ALM objective encourages the model to predict the next token with its context uni-directionally. In this way, the correlation between the target token and its contexts can be captured in a self-supervised manner. However, since a grid in the lower hierarchy shares the those of the upper hierarchies in our model, specific two locations with far distance would have same upper-level embeddings despite that they may have different semantics. For this reason, we incorporated information from the lower-level hierarchies

into the upper-level hierarchies when implementing ALM tasks to propagate the information of lower-level hierarchies to the upper-level hierarchies. In short, the predictions of upper-level hierarchies are contingent upon the predicted outcomes of lower-level hierarchies. This interdependence between hierarchical levels highlights the significance of integrating information across multiple scales to gain a comprehensive understanding of user trajectories.

As shown in Figure 2, we utilized a decomposed trajectory $s = \{l_0, l_1, \ldots, l_t\} = \{(l_0^1, l_0^2, \ldots, l_0^H), (l_1^1, l_1^2, \ldots, l_1^H), \ldots, (l_t^1, l_t^2, \ldots, l_t^H)\}$ as the input of our location embedding model, and predicted the *shifted* version of the input sequence s. We train our model with multiple training objectives. The ALM objectives of all hierarchies are trained simultaneously. However, each ALM objective has a different task complexity. Actually, since the grid size of the lower-level hierarchies is larger than that of the upper-level hierarchies, the trajectories of the lower-level hierarchies have monotonic patterns. Therefore, the ALM objectives of lower hierarchies are much less demanding to train than the ALM objectives of upper hierarchies, which causes a learning imbalance between tasks. In a multi-task architecture, the learning imbalance between tasks leads to causes the model to memorize a specific task instead of generalizing a pattern of data [1]. To solve this problem, in the ALM objectives, we sequentially incorporate the information from the lower hierarchy into the upper hierarchy. We denote this multi-task objective as Hierarchical ALM (HALM). The next location $l_{t+1} = (l_{t+1}^1, l_{t+1}^2, \ldots, l_{t+1}^H)$ to be predicted in the model consist of the H tokens. For this, we design H fully-connected feed-forward networks to predict the H next tokens using the causal location embedding model's output $\mathbf{e}_t^{(N)}$. First, we predict l_{t+1}^1, the token of level-1 (i.e., the coarsest-grained) hierarchy in the $(t{+}1)$-step, as follows:

$$\widehat{l_{t+1}^1} = FFN_{HLM}^1(\mathbf{e}_t^{(N)}), \tag{8}$$

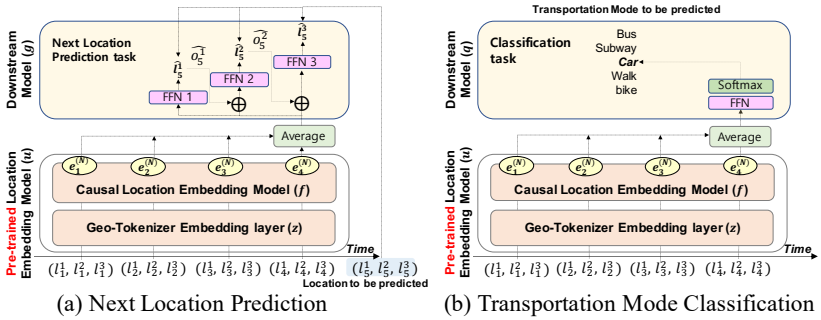
where FFN_{HLM}^1 is the fully-connected feed-forward network of the level-1 hierarchy and $\widehat{l_{t+1}^1}$ is the prediction output for the next location token l_{t+1}^1. In general, the prediction of the token of the level-h hierarchy (i.e., $h{>}1$) in the $(t{+}1)$-step, is sequentially implemented as follows:

$$\begin{aligned} \widehat{l_{t+1}^h} &= FFN_{HLM}^h(\mathbf{e}_t^{(N)} \parallel \widehat{o_{t+1}^{h-1}}), \\ \widehat{o_{t+1}^0} &= \mathbf{0}, \end{aligned} \tag{9}$$

where $\parallel$ is the concatenation operation and $\widehat{o_{t+1}^{h-1}}$ is the one-hot encoding vector from prediction result $\widehat{l_{t+1}^{h-1}}$. FFN_{HLM}^h is composed of two fully-connected layers in this paper. We construct the HALM objective to maximize the prediction accuracy of all of the hierarchies in the next location l_{t+1}. The pre-training objective of the HALM can be described as:

$$O_{HALM} = \underset{\theta}{\operatorname{argmax}} \sum_{h=1}^{H} \sum_{t=0}^{T} log(p(l_{t+1}^h | \widehat{l_{t+1}^h})), \tag{10}$$

where θ denotes the set of all trainable parameters in our model, T is the length of the trajectory, and H is the uppermost level of the hierarchy (i.e., the finest-grained).



(a) Next Location Prediction (b) Transportation Mode Classification

Fig. 3: Illustration of the downstream tasks. (a) Model architecture of the next location prediction task using an FFN layer stacked on the pre-trained location embedding model. (b) Model architecture of transportation mode classification task using one FFN layer stacked on the pre-trained location embedding model.

3.4 Fine-tuning Downstream tasks

Next Location Prediction task We implemented a next location prediction as a downstream task widely used in the location-based service in the real world [19, 7]. The trajectory up to T is used as an input in the downstream model, and the ground-truth is the three decomposed location records in $T+1$ (Figure 3a).

Classification task The model architecture for the transportation mode classification using a fully connected layer stacked on top of the pre-trained location embedding model is described in Figure 3b. A whole trajectory is used as an input in the downstream model, and the output is the transportation mode of the trajectory. See Appendix A.4 for details of above two downstream models.

4 Experiments

Our experiments are designed to answer the following research questions:
(RQ1): How effective is our pre-trained location embedding model compared to the state-of-the-art models in the various downstream tasks?
(RQ2): How do the different components affect the downstream tasks' performance?
(RQ3): What is the effect of the level of hierarchies in our pre-trained model?
(RQ4): How effective is the pre-training of the location embedding model in the self-supervised manner on the downstream tasks?

Table 1: Statistics of datasets

Dataset	Data Type	#Users	#Original Locations (100m)	#Tokenized Locations (100m)				#Traj	Time span
				Total	#Locations level-1 (100km)	#Locations level-2 (1km)	#Locations level-3 (100m)		
Mobile-T	Mobile Signal	0.4M	**79,812**	**6,740**	24	6,616	100	1.3M	7/1,2021-7/31,2021
Geo-Life	GPS	182	**50,003**	**8,476**	183	8,193	100	17,621	4/1,2007-8/31,2012

4.1 Datasets

Mobile-T: This data is a set of user trajectories collected by the base stations of the major cellular network operator, denoted as Mobile-T. As shown in Table 1, the size of location vocabularies at a 100m scale is 79812, which is too large to train for location embeddings. However, using the Geo-tokenizer, the sizes of the location vocabularies in each hierarchy, 100km, 1km, and 100m scale, are 24, 6616, and 100, respectively. This means that the total summation of the size of location vocabularies is 6740, which is less than 79812. Meanwhile, Mobile-T contains the land usage of the last location of a trajectory, associated with the purpose of the trajectory. There are 15 unique land usages of a trajectory, such as Apartment House or Business Facilities.

Geo-Life[21]: We also used the public GPS trajectory dataset, Geo-Life, which was collected with 182 users over a period of five years in Microsoft Research Asia. In the Geo-Life dataset, the trajectories are described as sequences of locations represented as GPS coordinates. Like the Mobile-T, the location record in this dataset was converted into a grid at a 100m scale. In this dataset, the number of distinct decomposed locations with three hierarchies (8476) was less than the number of original locations (50003), as shown in Table 1. The Geo-life dataset contains five unique transportation modes of a trajectory. See Appendix A.1 for details of two datasaets.

4.2 Settings

For both datasets, we assigned pre-train and fine-tune datasets of 80% and 20% of the total dataset. Then, we assigned train, validation, and test datasets of 80%, 10%, and 10% of the fine-tune datasets. We trained fine-tuning (i.e., downstream) models with the train datasets and chose the optimal hyper-parameters with the validation datasets. We set the hierarchy level H as three, and the scales of the level-1, level-2, and level-3 hierarchies were 100km, 1km, and 100m, respectively. We demonstrated the superiority of our pre-trained location model by comparing six location embedding models: (1) SERM [17], (2) HIER [13], (3) DeepMove [24], (4) TALE [15], (5) CTLE [7], and (6) TrajFormer [6]. Including our model, the dimension of the embedding layer and final location embedding vector was set to 256 in all models. We described the model details in Appendix A.2 and the pre-training setting in Appendix A.3.

Table 2: Comparison of Next Location Prediction performance and efficiency with those of previous studies. The top two methods are highlighted in bold and underlined.

Downstream Model		FFN		LSTM					
Metric		Top-1	Top-5	Top-1	Top-5	#Params	#FLOPs	TR-time	Inf-time
Dataset	Pre-trained Model	Acc(%)	Acc(%)	Acc(%)	Acc(%)				
Mobile-T	SERM[17]	8.41±0.11	26.77±0.32	8.23±0.09	25.08±0.35	12.68M	1.31B	647.25	56.00
	HIER[13]	10.09±0.05	30.37±0.15	8.99±0.12	28.75±0.34	18.47M	1.42B	837.39	71.47
	DeepMove[24]	9.05±0.15	30.81±0.19	9.38±0.30	31.75±0.54	49.92M	2.62B	1338.47	116.55
	TALE[15]	9.02±0.14	30.28±0.48	9.43±0.14	29.12±0.42	49.92M	7.85B	3493.91	285.57
	CTLE[7]	10.71±0.24	32.39±1.09	9.31±0.14	25.77±0.46	43.72M	1.71B	642.61	56.60
	TrajFormer[6]	10.45±0.02	30.48±0.06	8.88±0.19	21.47±0.26	40.31M	1.55B	693.37	56.20
	Ours	**11.47±0.13**	**38.41±0.11**	**11.20±0.05**	**40.21±0.08**	**6.90M**	**0.44B**	**400.63**	**43.37**
Geo-Life	SERM[17]	18.35±0.11	29.02±0.18	18.58±0.31	36.46±0.44	12.60M	0.82B	187.12	15.20
	HIER[13]	18.80±0.13	31.33±0.19	18.09±0.16	38.34±0.13	13.67M	0.96B	207.43	16.37
	DeepMove[24]	17.46±0.11	35.72±0.12	18.68±0.19	38.03±0.22	25.60M	1.64B	345.66	30.38
	TALE[15]	17.58±0.13	31.17±0.18	19.04±0.12	38.15±0.14	25.60M	4.92B	678.68	59.59
	CTLE[7]	24.69±0.25	45.12±0.15	21.53±0.38	43.49±0.36	30.39M	1.18B	192.10	16.48
	TrajFormer[6]	27.71±0.97	50.51±0.93	26.34±0.19	51.53±0.30	28.91M	1.06B	223.44	16.97
	Ours	**28.58±0.21**	**60.07±0.31**	**26.99±0.28**	**53.68±0.61**	**7.71M**	**0.48B**	**179.92**	**14.74**

* The number of parameters and FLOPs are derived from only location embedding models, except downstream task models (FFN and LSTM). M and B denote million and billion, respectively. We executed each baseline ten times and recorded the mean and standard deviation of each baseline. TR-time and Inf-time indicate the training time (seconds) per epoch and inference time (seconds) per epoch in the FFN case of the next location prediction model respectively. The training and inference speed were calculated by averaging those of FFN and LSTM with each pre-trained model, using one V100 GPU.

4.3 Experimental Results (RQ1)

Next Location Prediction task The performance of the next location prediction task was assessed using the accuracy of the test dataset. The rate at cutoff k, denoted as **Acc@k**, counts the fraction of cases where the target location is among the top k. We reported this metric as k=1 and k=5. We also evaluated the efficiency of the pre-trained location embedding model by measuring the number of model parameters and operations (FLOPs). A performance comparison for the next location prediction task is shown in Table 2. SERM [17] with the randomly initialized embedding layers did not perform well because this method has difficulty incorporating the context of a trajectory. DeepMove [24] and TALE [15] adopting Skip-gram and CBOW utilize the co-occurrence probabilities of target locations and their contexts, but the contexts they consider were restricted to specific window size. More importantly, these methods train the heavy embedding layers due to the large size of the location vocabulary, which was over 50,000 in both datasets.

Unlike the above previous studies, CTLE [7] and TrajFormer [6] incorporate the multi-functionality of a location via a self-attention module to consider the contexts of trajectories. As a result, they showed significantly better performance than the other baseline models. However, they also had difficulty dealing with the large size of the location vocabulary and needed to train the heavy embedding layers. Our model consistently outperformed other location embedding methods, even with fewer parameters and the number of FLOPs. This can be attributed to the efficient processing of large amounts of location vocabulary using the Geo-tokenizer embedding layer and HALM objective. In addition, our model was faster than other baselines in the training and inference for both datasets (Table 2).

Concurrently, our experimental results demonstrate that, for the next location prediction tasks, adopting a feed-forward network in conjunction with a self-attention-based pre-trained model (e.g., CTLE[7], TrajFormer[6], Ours) proves to be more competitive than utilizing an LSTM-based approach. One potential reason is that the pre-trained model, which is based on a self-attention layer, has already learned to capture long-range dependencies in the input sequence. In this case, adding another layer of sequential processing with LSTM may not provide significant additional benefits.

Classification task The performance of the land usage and transportation mode classification task was assessed using the accuracy, macro-precision, and macro-recall of the test dataset. A performance comparison for these tasks is shown in Table 3. With the fewest parameters, our pre-trained location embedding model showed the best performances among other location embedding models for both tasks, and was faster than other baselines in the training and inference. This indicates the superior quality of our pre-trained location embeddings.

Table 3: Comparison of Land Usage and Transportation Mode Classification task with those of previous studies. The top two methods are highlighted in bold and underlined.

Dataset	Downstream Task	Metric / Pre-trained Model	Accuracy(%)	Precision(%)	Recall(%)	F-1(%)	TR-time	Inf-time
Mobile-T	Land Usage Classification	SERM[17]	79.12±0.02	73.83±0.03	70.35±0.02	70.65±0.02	58.23	5.42
		HIER[13]	79.39±0.05	76.70±0.03	73.31±0.03	74.42±0.02	121.28	6.94
		DeepMove[24]	81.05±0.09	77.95±0.02	73.43±0.03	75.38±0.02	129.09	10.99
		TALE[15]	83.02±0.06	77.09±0.07	73.33±0.10	76.47±0.11	310.95	26.33
		CTLE[7]	87.44±1.29	75.42±2.97	73.31±2.37	73.22±2.04	58.36	4.98
		TrajFormer[6]	73.65±0.83	67.37±4.90	56.24±1.36	59.42±2.08	66.86	6.68
		Ours	**89.47±0.29**	**82.27±1.65**	**84.19±0.94**	**82.80±0.95**	**41.72**	**4.51**
Geo-Life	Transportation Mode Classification	SERM[17]	68.19±0.02	69.13±0.03	69.21±0.03	69.19±0.03	5.26	0.58
		HIER[13]	64.45±0.17	60.56±0.12	64.52±0.21	64.44±0.14	6.44	0.74
		DeepMove[24]	69.81±0.09	71.46±0.02	69.96±0.07	69.95±0.08	9.58	1.28
		TALE[15]	62.88±0.08	70.32±0.09	65.86±0.11	66.53±0.11	18.78	2.51
		CTLE[7]	68.15±1.10	71.36±1.10	73.21±0.84	71.01±1.06	5.30	0.72
		TrajFormer[6]	73.68±1.88	77.37±1.33	76.49±1.95	76.10±1.68	5.88	0.59
		Ours	**81.17±0.40**	**81.58±0.74**	**82.70±0.75**	**81.81±0.32**	**4.34**	**0.46**

* The number of parameters and FLOPs in each pre-trained embedding model is equal to the case of the next location prediction task, as shown in Table 2. We executed each baseline ten times and recorded the mean and standard deviation of each baseline.

4.4 Ablation study

Study on the components (RQ2) We investigated the effectiveness of each component of our pre-trained location embedding model by designing three variants as follows:

(1) Baseline: This model utilizes the original transformer decoder using the ALM objective for pre-training without the Geo-tokenizer embedding layer. This is a simple auto-regressive pre-trained model.

(2) +Geo-tokenizer(GT): This model replaces the embedding layers in the baseline with the Geo-tokenizer embedding layer, which decomposes each location record into the three hierarchical components (100km, 1km, 100m). The

pre-trained model's objective is the basic ALM proposed in the transformer [14]. Therefore, the ALM objectives of the three hierarchies are independent.
(3) +Geo-tokenizer(GT)+HALM: This model uses the Geo-tokenizer fused on the baseline and employs the HALM objective. This is our proposed model.

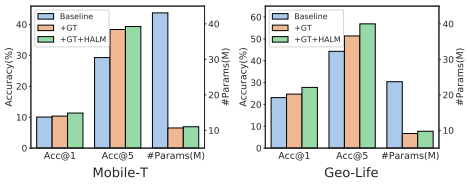


Fig. 4: Comparison of next location prediction performance and efficiency for different combinations of components.

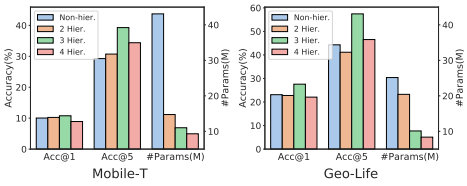


Fig. 5: Next location prediction performance and efficiency comparison of different hierarchy levels.

The comparison of these three variants was conducted with our pre-trained location embedding model on the next location prediction task, shown in Figure 4. The performance was calculated by averaging two downstream models (FFN and LSTM). Compared to the baseline model, the model with the Geo-tokenizer embedding layer showed higher performance in both datasets. In addition, the model combining the HALM objective with the Geo-tokenizer embedding layer outperformed other variants. This means that the learning imbalance caused by location decomposition into multiple hierarchies by the Geo-tokenizer embedding layer was resolved through HALM. The comparison of these three variants was conducted with our pre-trained location embedding model on the classification tasks shown in Figure 6. In the land usage and transportation mode classification tasks, both the Geo-tokenizer embedding layer and HALM can improve the prediction performance over the baseline.

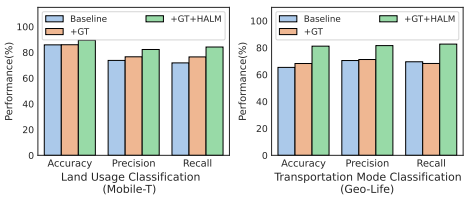


Fig. 6: Comparison of classification performance and efficiency for different combinations of components.

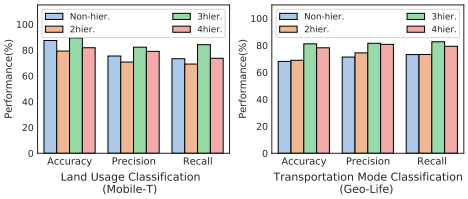


Fig. 7: Comparison of classification performance and efficiency for different hierarchy levels.

Study on the level of hierarchies (RQ3) We studied the effectiveness of the level of hierarchies by comparing three variants in terms of the degree of

hierarchies: **(1) Four** (100km,10km, 1km, and 100m), **(2) Three** (100km, 1km, and 100m), and **(3) Two hierarchies case** (10km and 100m).

The performance was calculated by averaging those of FFN and LSTM. As shown in Figure 5, the three hierarchies case showed the best Acc@*1* and Acc@*5* with relatively few parameters for both datasets on the next location prediction task. In addition, we determined that increasing the hierarchy level did not necessarily improve the next location prediction performance. The larger the hierarchy level(H), the smaller the location vocabulary size, resulting in a smaller model size. If the model size is too small, the performance deteriorates, so it can be seen that setting an appropriate H is essential. We also compared these three variants with the non-hierarchies model on the two classification tasks, as shown in Figure 7. In the both classification tasks, the three hierarchies case showed the best performance with the fewest parameters. It can be seen that the performance of the hierarchical case above a certain level is better than that of the non-hierarchical case with fewer model parameters.

Study on the pre-training (RQ4) Pre-training significantly improved the performance of downstream tasks. We compared the performance of our model in two cases: with pre-training (w/PT) and without pre-training (wo/PT). As shown in Figure 8, the model with the pre-trained backbone showed higher performance in both datasets for the next location prediction task than the wo/PT. In the Geo-Life dataset, the performance gap between the w/PT and wo/PT was relatively small compared to that of Mobile-T. This is because the number of trajectories of Mobile-T is larger than Geo-Life's. In other words, the larger the data, the greater the performance improvement of the downstream task due to pre-training. In the classification task, the w/PT performed significantly better than the wo/PT in terms of accuracy, precision, and recall in both datasets, as shown in Figure 9.

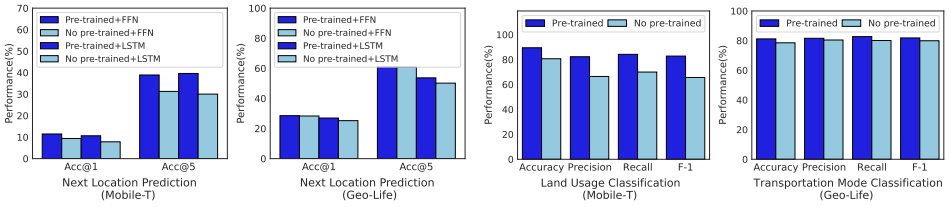


Fig. 8: Effect on the pre-training for the next location prediction task.

Fig. 9: Effect on the pre-training for the classification task.

4.5 Deployed Solution

Our pre-trained model has been implemented in an inter-company marketing tool, designed to predict individuals likely to visit a particular area for location-

based marketing purposes. The deployed solution effectively encompasses entire regions within the author's country by utilizing the next-location prediction model built upon our pre-trained model. More details, including a screenshot of our graphical user interface (GUI) tool, can be found in the Appendix A.6.

5 Related Work

In recent years, pre-training an embedding model with self-supervised objectives has become a common practice in spatial-temporal data mining. For example, DeepMove [24] and TALE [15] implemented skip-gram and CBOW [9], respectively, to model human mobility, and an N-gram model is adopted to learn latent representations of a location [18, 13]. SERM [17] jointly trained the embeddings of user, location, time, and keyword. These location embedding models generated a single latent representation for each location, which indicates they can not discriminate among variable functionalities of a location. To address this problem, previous studies have employed a transformer encoder architecture [14] with Masked Language Model[3] to generate the dynamic embeddings derived along the dissimilar trajectories [10, 7]. Specifically, TrajFormer [6], CTLE [7] and BERTLoc [10] proposed a transformer encoder based location embedding model that dynamically assigns the embedding vector of a target location, varying with the location's trajectory. Nevertheless, previous studies are difficult to be applied in the real world, where the number of locations can be considerably large, or a fine-grained resolution is needed [13]. Previous studies have dealt with at most ten thousand locations to train their representations [7, 13, 20]. This problem can be addressed by reconstructing a location with several grids at different scales and making each grid at a large scale share the grids at a small scale. HIER [13] decomposed a location at several spatial scales to consider the spatial hierarchy in the location embeddings. However, in their approaches, locations in each level of the hierarchy are independently trained, and therefore the number of locations to be embedded is still large. For this reason, we encourage grids in the lower-level hierarchies to share the grid set in the upper-level hierarchy in order to represent a location using relatively small location vocabularies.

6 Conclusions

This paper proposed a contextual location embedding model to efficiently handle numerous location vocabularies in various real-world applications. We represented a particular location as a combination of several grids at different scales to reduce the number of locations to be trained. In addition, to incorporate various location functionalities, our model dynamically calculated the embedding vector of a target location, which varies depending on its trajectory. We employed a variant of the ALM objective, which trains the model with several ALM objectives sequentially. The experimental results demonstrated that our model significantly improved the performance of downstream models with fewer model parameters, compared to the existing location embedding methods.

Acknowledgment This work was supported by the institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2B5B02001913).
The authors would like to thank the AI Service Business Division of SK Telecom for providing GPU cluster support to conduct massive experiments.

Ethical Statement

There are no ethical issues.

References

1. Aksoy, Ç., Ahmetoğlu, A., Güngör, T.: Hierarchical multitask learning approach for bert. arXiv preprint arXiv:2011.04451 (2020)
2. An, S.F.G.C.B., Chee, Y.M.: Poi2vec: Geographical latent representation for predicting future visitors (2017)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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. pp. 4171–4186 (Jun 2019)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
5. Li, Y., Du, N., Bengio, S.: Time-dependent representation for neural event sequence prediction. arXiv preprint arXiv:1708.00065 (2017)
6. Liang, Y., Ouyang, K., Wang, Y., Liu, X., Chen, H., Zhang, J., Zheng, Y., Zimmermann, R.: Trajformer: Efficient trajectory classification with transformers. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 1229–1237 (2022)
7. Lin, Y., Wan, H., Guo, S., Lin, Y.: Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence (2020)
8. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
10. Park, S., Lee, S., Woo, S.S.: Bertloc: duplicate location record detection in a large-scale location dataset. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. pp. 942–951 (2021)
11. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
12. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
13. Shimizu, T., Yabe, T., Tsubouchi, K.: Learning fine grained place embeddings with spatial hierarchy from human mobility trajectories. arXiv preprint arXiv:2002.02058 (2020)

14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
15. Wan, H., Li, F., Guo, S., Cao, Z., Lin, Y.: Learning time-aware distributed representations of locations from spatio-temporal trajectories. In: International Conference on Database Systems for Advanced Applications. pp. 268–272. Springer (2019)
16. Wan, H., Lin, Y., Guo, S., Lin, Y.: Pre-training time-aware location embeddings from spatial-temporal trajectories. IEEE Transactions on Knowledge and Data Engineering (2021)
17. Yao, D., Zhang, C., Huang, J., Bi, J.: Serm: A recurrent model for next location prediction in semantic trajectories. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 2411–2414 (2017)
18. Yao, Z., Fu, Y., Liu, B., Hu, W., Xiong, H.: Representing urban functions through zone embedding with human mobility patterns. In: IJCAI. pp. 3919–3925 (2018)
19. Zhao, P., Luo, A., Liu, Y., Zhuang, F., Xu, J., Li, Z., Sheng, V.S., Zhou, X.: Where to go next: A spatio-temporal gated network for next poi recommendation. IEEE Transactions on Knowledge and Data Engineering (2020)
20. Zhao, S., Zhao, T., King, I., Lyu, M.R.: Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In: Proceedings of the 26th international conference on world wide web companion. pp. 153–162 (2017)
21. Zheng, Y., Xie, X., Ma, W.Y., et al.: Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data Eng. Bull. **33**(2), 32–39 (2010)
22. Zhou, F., Gao, Q., Trajcevski, G., Zhang, K., Zhong, T., Zhang, F.: Trajectory-user linking via variational autoencoder. In: IJCAI. pp. 3212–3218 (2018)
23. Zhou, F., Yue, X., Trajcevski, G., Zhong, T., Zhang, K.: Context-aware variational trajectory encoding and human mobility inference. In: The World Wide Web Conference. pp. 3469–3475 (2019)
24. Zhou, Y., Huang, Y.: Deepmove: Learning place representations through large scale movement data. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 2403–2412. IEEE (2018)

European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) 2023

Title: Pre-training Contextual Location Embeddings in Personal Trajectories via Efficient Hierarchical Location Representations

Chung Park[1], Taesan Kim, Junui Hong, Minsung Choi, and Jaegul Choo[2]

A Appendix

A.1 Dataset Details

We used two real-world datasets: (1) Mobile-T and (2) Geo-Life [21]. In this section, we describe the details of pre-processing in two datasets.

Mobile-T This data is a set of user trajectories collected by the base stations of the major cellular network operator. Each base station provides a signal to the surrounding area and records the user's access to the corresponding area. The average density of base stations in this dataset is about 100m, and therefore we converted the location records in Mobile-T into a grid at a 100m scale. Mobile signaling datasets are more suitable for evaluating the effectiveness of our model because they contain dense trajectories, unlike some public check-in datasets [2, 7]. Since base stations are densely installed, the signaling data is able to represent the user's overall trajectories.

We randomly sampled about 0.4 milion customers who agreed to collect and analyze their information. The Mobile-T dataset consists of mobile signaling data; not all location records indicate a user's visit. Location records simply passed by the user do not imply explicit purposes. To filter out such points, we removed the location records which had an average duration time below five minutes. Then, we calculated the velocity of each location record and denoted *stop* to the location records under 4km/h velocity. A sequence of all locations between successive *stop* records is considered to be the user trajectory. Finally, we derived trajectories from more than ten location records.

Meanwhile, the Mobile-T dataset contains the land usage of the last location of a trajectory. There are 15 unique land usages of a trajectory, which indicates Apartment House (30.34%), Factory (2.07%), Educational Research Facilities (1.53%), Detached House (36.21%), Hotel Facilities (0.50%), Business Facilities (5.38%), Sports Facilities (0.09%), Transportation Facilities (0.41%), Medical Facilities (017%), Automobile related Facilities (0.49%), Residential Neighborhood Facilities/class1 (6.90%), Residential Neighborhood Facilities/class2 (13.90%), Religion Service Facilities (0.14%), Storage Facilities (0.49%), and Shopping Service Facilities (1.06%).

[1] Contact: cpark88kr@gmail.com
[2] Corresponding Author (jchoo@kaist.ac.kr)

Geo-Life [3] The trajectories in this dataset are represented as sequences of locations, each of which contains latitude, longitude, and altitude. GeoLife contains 17,621 trajectories collected by 182 users over a period of five years in Microsoft Research Asia. Among them, trajectories of 73 users have their transportation modes. The GPS trajectories in this dataset were recorded in every 1-5 seconds, and we selected location records of 1-minute increments. In addition, we extracted trajectories from more than ten location records. The way to decompose the location with several hierarchies was the same as that used for the Mobile-T dataset, using latitude and longitude. The Geo-life dataset contains five unique transportation modes of a trajectory, which indicates bus (18.58%), car (21.46%), walk (27.27%), bike (18.58%), and subway (7.13%).

A.2 Pre-trained Location Embedding model Details

We demonstrated the superiority of our pre-trained location embedding model by comparing six distributed embedding models.
(1) SERM [17]: This model is a randomly initialized embedding layer to produce input vectors for downstream task models. The embedding layer consists of the embedding for the location record, the timestamp, and the text information aligned with the location. A specific model (e.g., LSTM) for a downstream task is connected to this embedding layer and trained together. The dimension of the embedding layer was set to 256. We removed the embedding module for the text information in the original SERM, because there is no a text message that describes the user's activity in each GPS record of our datasets.
(2) HIER [13]: The large location vocabulary problem can be solved by reconstructing a location with multiple grids at different scales, and having each large scale grid share the small scale grids. HIER [13] decomposes a location into multiple spatial scales to account for the spatial hierarchy in the location embeddings. In this model, we set the decomposed spatial scales to 100km, 1km, 100m, as in our model.
(3) DeepMove [24]: They applied the Skip-gram of Word2Vec to trajectory data which have a set of origin and destination records. We modified the proposed module to fit our data with dense locations in a trajectory. The skip-gram with negative sampling was used as a training method, and the window size was set to five. The dimension of the location embedding in DeepMove was set to 256.
(4) TALE [15]: The CBOW module of Word2Vec was employed to generate pre-trained vectors of locations. To reduce computational complexity, they used the hierarchical softmax method for training, but we instead employed negative sampling to improve performance. The rest of the parameter settings were identical to the DeepMove.
(5) CTLE [7]: They used the bidirectional transformer encoder architecture with Masked Language Model (MLM) pre-training objective to derive the context-aware location embedding vectors, considering contexts. In this model, six stacks

[3] https://www.microsoft.com/en-us/download/confirmation.aspx?id=52367

of Transformer encoder layers which contained eight attention heads are employed, and the dimension of the embedding layer and final location embedding vector was set to 256.

(6) TrajFormer [6]: They developed the squeezed Transformer Encoder to classify the transportation modes of a trajectory, effectively diminishing the dimensions of keys and values prior to computing the self-attention module. In this model, six stacks of Transformer encoder layers which contained eight attention heads are employed, and the dimension of the embedding layer and final location embedding vector was set to 256. The squeeze rate is set to 1 for the best performance. In our experiments, the sub-path labeling in this model was removed.

Table 4: Hyperparameters of our location pre-trained embedding model on Mobile-T and Geo-Life.

Hyperparameter	**Mobile-T**	**Geo-Life**
Epoch	20	10
Batch size	32	32
Hidden size	256	256
Attention dropout	0.1	0.1
# heads	8	8
# transformer decoder layers	6	6
Max sequence length T	32	32
Hierarchy Level	3	3
Adam ϵ	1e-4	1e-4
Adam (β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)
Weight decay	1e-2	1e-2
# warm-up steps	10000	10000

A.3 Pre-training Details

Table 4 describes the optimal hyperparameters of our location pre-trained embedding model. The max length of an input trajectory was set to 32, and the batch size was 32. Each dimension of the embedding layers in the Geo-tokenizer embedding layer (z) was set to 256, and the dimensions of the final embedding vectors in the pre-trained location embedding models (u) was set to 256. Our model adopted six stacks of transformer dncoder layers which contained eight attention heads. The number of layers in the feed-forward network of HALM is two. The Adam optimizer with a learning rate of 0.001, β_1 of 0.9, and β_2 of 0.999 was used to find the optimal parameters of our model. We trained our pre-training and fine-tuning models with Cross-entropy loss. Our model was trained using one V100 GPU.

A.4 Fine-tuning Downstream model Details

Next Location Prediction task Given a trajectory $s' = \{l_0, l_1, ..., l_T\} = \{(l_0^1, l_0^2, ..., l_0^H)\ , (l_1^1, l_1^2, ..., l_1^H), \ldots, (l_T^1, l_T^2, ..., l_T^H)\}$, the downstream model connected to our pre-trained location model u is a function g to predict the next location $l_{T+1} = (l_{T+1}^1, l_{T+1}^2, ..., l_{T+1}^H)$ as shown in Figure 3a. Similar to the pre-training stage, the fine-tuning for the next location prediction is a multi-task model, which predicts the H next location components of all hierarchies, respectively. We consider the next prediction to be correct when all the hierarchies $(l_{T+1}^1, l_{T+1}^2, ..., l_{T+1}^H)$ are simultaneously correct. Therefore, the function g contains H independent layers to predict the H next location components of H hierarchies. Then, the probability of the next location component in the level-h hierarchy is calculated in the same way as the HALM objective in pre-training. In this paper, we employed two models as the function g: (1) a fully-connected layer and (2) LSTM (Long-Short Term Memory) [4].

(1) FFN: The prediction of the next location l_{T+1}^h is sequentially implemented using a fully-connected feed-forward network g as follows:

$$\begin{aligned} \widehat{l_{T+1}^h} &= g_h(\frac{1}{T}\sum_{t=1}^{T}(\mathbf{e}_t^{(N)}) \parallel \widehat{o_{T+1}^{h-1}}), \\ \widehat{o_{T+1}^0} &= \mathbf{0}, \end{aligned} \tag{11}$$

where $\parallel$ is the concatenation operation, $\widehat{o_{T+1}^{h-1}}$ is the one-hot encoding vector from the prediction result $\widehat{l_{T+1}^{h-1}}$, and g_h is the feed-forward network of the level-h hierarchy, used to predict the next location component l_{T+1}^h. The $\mathbf{e}_t^{(N)}$ is the pre-trained model's output vector corresponding t-th step. In short, this model is a fully-connected feed-forward network, which uses the average of the output vectors of pre-trained location embeddings model and predicts the location of the $T+1$ timestamp. For the experiments, the model g consists of one fully-connected feed-forward layer, and the output of the model g are fed into a softmax layer.

(2) LSTM: The prediction of the next location l_{T+1}^h is sequentially implemented using LSTM layer g as follows:

$$\begin{aligned} \widehat{l_{T+1}^h} &= g_h([\mathbf{e}_t^{(N)} \parallel \widehat{o_{T+1}^{h-1}}]_{t\in\{1:T\}}), \\ \widehat{o_{T+1}^0} &= \mathbf{0}, \end{aligned} \tag{12}$$

where $\widehat{o_{T+1}^{h-1}}$ is the one-hot encoding vector from the prediction result $\widehat{l_{T+1}^{h-1}}$, g_h is the LSTM layer of the level-h hierarchy, and $\mathbf{e}_t^{(N)}$ is the pre-trained model's output vectors in the t-th step. We use the output representation of the last output state of the LSTM to predict the next location component l_{T+1}^h. In short, the sequence of output vectors of the pre-trained location embedding model were sequentially fed into the one LSTM layer and the softmax layer, to predict the location of the $T+1$ timestamp considering temporal correlation.

Classification task In this task, given a trajectory $s' = \{l_0, l_1, ..., l_T\} = \{(l_0^1, l_0^2, ..., l_0^H), (l_1^1, l_1^2, ..., l_1^H), \ldots, (l_T^1, l_T^2, ..., l_T^H)\}$, the downstream model connected to our pre-trained model u is a function q to classify the land usages or transportation modes as shown in Figure 3b. The average of the output vectors of the pre-trained location embeddings model of time ts is fed into the function q. The model q consists of one fully-connected feed-forward layer and the softmax layer.

A.5 Extended Study on the components (RQ2)

As shown in Figure 10, we further investigated the effectiveness of each component of our pre-trained location embedding model by designing four variants as follows:

(1) Baseline: This model utilizes the original transformer decoder using the ALM objective for pre-training without the Geo-tokenizer embedding layer. This is a simple auto-regressive pre-trained model.

(2) +Geo-tokenizer(GT): This model replaces the embedding layers in the baseline with the Geo-tokenizer embedding layer, which decomposes each location record into the three hierarchical components (100km, 1km, 100m). The pre-trained model's objective is the basic ALM proposed in the transformer [14]. Therefore, the ALM objectives of the three hierarchies are independent.

(3) +Geo-tokenizer(GT)+MLM: This model uses the Geo-tokenizer fused on the baseline and employs the Masked Location Model (MLM) objective of the CTLE[7]. This is the pretrained model replacing our causal location embedding model with the stack of the bidirectional transformer encoders.

(4) +Geo-tokenizer(GT)+HALM: This model uses the Geo-tokenizer fused on the baseline and employs the HALM objective. This is our proposed model. This shows that our proposed HALM method is superior to the MLM method.

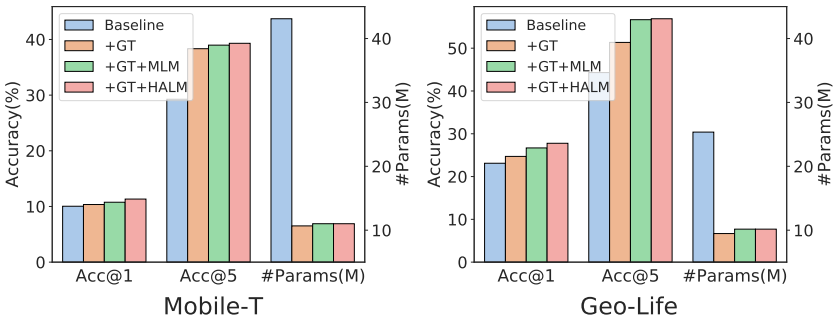


Fig. 10: Comparison of next location prediction performance and efficiency for different combinations of components.

A.6 Our Deployed Solution

Our pre-trained model has been implemented in an inter-company marketing tool, designed to predict individuals likely to visit a particular area for location-based marketing purposes. The deployed solution effectively encompasses entire regions within the author's country by utilizing a next-location prediction model built upon our pre-trained model. Figure 11 shows a screenshot of our custom GUI tool, which extract the list of customers who will move to a specific region given his/her trajectory.

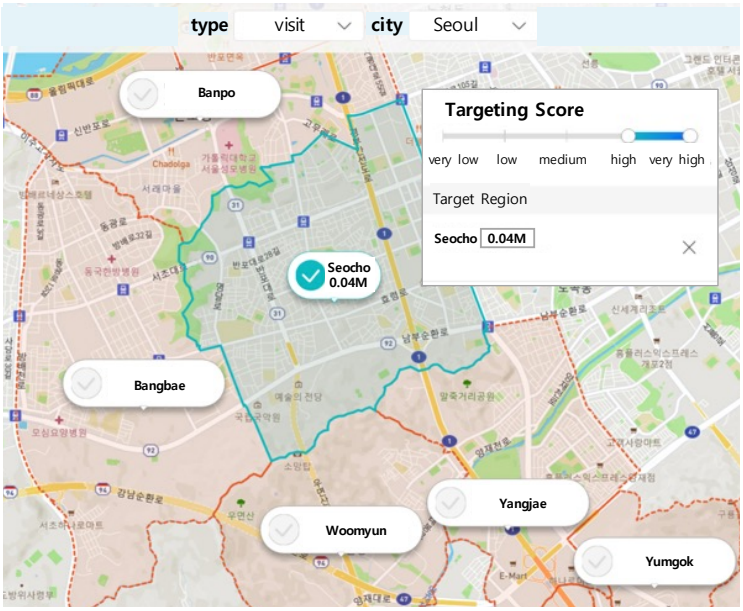


Fig. 11: Illustration of our location-based marketing tool. Seocho refers to a district in Seoul, Korea. In this figure, the estimated number of people expected to visit the Seocho district is 0.04 million. Using this tool, we can identify those who are likely to visit the Seocho district. The abbreviation **M** represents million.

A.7 Qualitative analysis

We also compared our pre-trained model with CTLE by visualizing the trained trajectories' representations ($\sum_{t=1}^{T} e_t^N$) using t-SNE [8] from the test dataset (Mobile-T), as shown in Figure 12. CTLE is the state-of-the-art model for several downstream tasks such as the next location prediction. In the Mobile-T dataset, the land usage of the last location is the purpose of the trajectory (i.e., destination). It can be seen that trajectories' representations trained by

our model tend to push trajectories of different purposes than CTLE. It reflects that representations learned by our model can capture the semantic purpose of the trajectory.

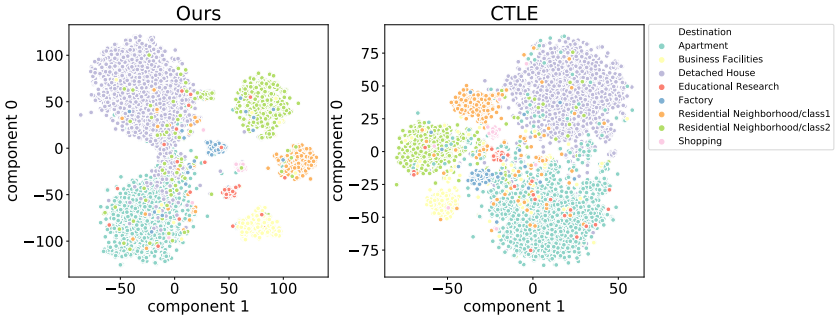


Fig. 12: Visualization of the pre-trained trajectories' representations by our model and CTLE on the Mobile-T dataset.