

Task 3 Open Advanced Tasks

Problem Statement

Following definitions by Hong et al. (in Appendix), we aim to design an algorithm that, given an input of a user_id and his fixed-length (99 in our model) sequence of consecutive staypoints, together with corresponding sequences of location_id, arrival_time, duration stayed, location area (area of convex hull), location centre and Place Of Interest (POI) context, will output a prediction of the next location to be visited. For a sample of inputs, our output will be a sample_size x 22179 probability matrix, where the $(i,j)^{th}$ element gives the probability that the next location of data i is location j. next_location_id is used as the ground truth class labels.

Data Preprocessing

Cleaning

We removed rows where $x,y = \pm 999$ as this means missing data. Then we multiplied x and y by 500 as 1 grid unit = 500m, then multiplied t by 30 to convert it to minutes before combining with d to form a proper DateTime (assuming UTC timezone & that d = 0 starts from 1st Jan 2024). We then filtered out the last half of the time period and users as the dataset was too large. We finally obtained 2175462 positionfixes.

Parameter Settings

Staypoints Generation

In Trackintel, we need to set $\text{gap_threshold} \geq \text{temporal_resolution}$ = minimum duration between any 2 positionfixes = 30 min, otherwise no staypoints will be generated. Based on Trackintel, a set with only 1 positionfix is allowed to be a staypoint. To avoid this, we used $\text{time_threshold} = \text{temporal_resolution} + 1 > \text{temporal_resolution}$ & $\text{gap_threshold} = 2 * \text{temporal_resolution} - 1 < 2 * \text{temporal_resolution}$ (i.e. do not allow tracking gaps). We set $\text{distance_threshold} = \text{spatial_resolution}$ = minimum distance between any 2 positionfixes = 500m so that for each staypoint, its positionfixes have identical geometries. For our data with poor spatial resolution, this is ideal as 1 positionfix covers $500^2 = 250000\text{m}^2$ of area which is already too big so we cannot allow for any more discrepancy in positionfixes' geometries in a staypoint. Trackintel only supports haversine distance which is for latitude-longitude coordinates but we are using flat cartesian coordinates for geometry, so we did monkey patching to use the euclidean distance function. We finally obtained 171105 staypoints.

Locations Generation

We used DBSCAN clustering as it accommodates non-spherical clusters of staypoints. We used a euclidean distance function as explained. We set num_samples (minPts for DBSCAN) to 1 so that every staypoint will be a core point and thus belong to exactly 1 location and there will be no outliers, which makes sense since every staypoint, even if isolated, is at least a location in itself. If $\text{epsilon} < \text{spatial_resolution}$ then every individual

staypoint is a unique location, which is not very meaningful, so we need to avoid this by ensuring that $\epsilon \geq \text{spatial_resolution}$. Also $\text{min_intercluster_distance} = (\epsilon / \text{spatial_resolution} + 1) * \text{spatial_resolution}$, so we set this minimum separation distance between locations to 1000m, then set $\epsilon = \text{min_intercluster_distance} - \text{spatial_resolution}$. We finally obtained 22179 locations

Preparation of Input Data

POI Context Vector Generation (Hong et al., 2023)

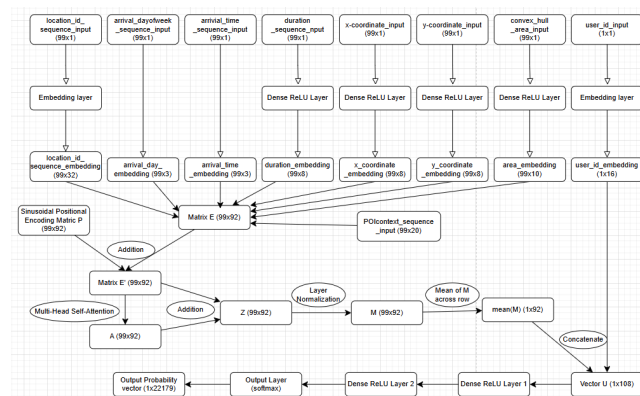
We first set the vicinity_radius to 1000m. Then for each location, compute the POI vector, where element j of the vector gives the count of POIs of category j within the vicinity_radius of the location's centre. We then treat each location as a document and each POI category as a word, so each POI vector can then be viewed as a Bag-Of-Words (BOW) vector. We stack them as row vectors of BOW matrix. We then fit and transform this BOW matrix using Latent Dirichlet Allocation (LDA) algorithm to obtain topic distribution matrix, whose $(i,j)^{\text{th}}$ element gives probability of topic j occurring in location i .

Obtaining Historical Sequences

Afterwards, we also calculated the convex hull area of each location. Then we filter out positionfixes which do not belong to any staypoint, then merge additional information from staypoints using staypoint_id, and from locations using location_id. We then sort this complete dataframe by user_id and tracked_at, then group by user_id, and for each user, obtain his full ordered sequence of staypoints and relevant information. Arrival timestamp was split into day & hour to enable the model to identify weekly and daily patterns respectively. We then filter out users whose sequence length < 100 as we want our input sequence length = 99 (excluding ground truth). If on average each user has about 10 staypoints/day, then duration span of the input sequence is about 10 days, which is reasonable as up to the previous 7 days of historical data is relevant in next location prediction (Hong et al., 2023). For each user, we then obtain all possible 100-length subsequences of consecutive elements from his full sequences, then remove the last element from each sequence as it is used as ground truth (next_location_id)

Model Architecture

Flow Chart for Single Data Input



Referring to our flow chart, in generation of embeddings from inputs, embedding layers were used for categorical features while dense layers with ReLU (Rectified Linear Unit) activation function were used for numeric features (except arrival timestamp). Our embedding row vector of arrival timestamp (formula in Appendix) represents both linear and cyclic (using sinusoidal functions scaled to its period (24 for hour, 7 for day)) nature of time. As POI context vector sequence already sufficiently represents land use context information, we will not further embed it and simply take it as it is. All concatenations were done along the column axis to preserve separate feature dimensions. We used sinusoidal positional encoding to get positional_encoding_matrix P (formula in Appendix). For the Multi-Head Self-Attention (MHSA) module (formulas in Appendix), we set key_dim such that A and E' will have the same dimensions. We calculated $Z = A + E'$ so as to retain information from E', then we obtain M by layer-normalisation (formula in Appendix) of Z to prevent values from growing too large as they propagate through layers. The single vector mean(M) is representative of the entire sequence information. Final output layer uses softmax activation as it is producing a probability vector for Multiclass Classification. (model details in Appendix)

Model Explanation

Rationale of LDA

The rationale of using LDA is that we can set the dimension of the POI context vector, which is the number of topics present among the locations and POI categories, to be smaller than the number of POI categories since we can expect them to have some common topics, e.g. "Food". This enables us to not only reduce the dimension of POI vector, but also capture similarities between POIs using common topics which will be beneficial for the model to infer the land use context more effectively.

Rationale of Positional Encoding

Getting $E' = E + P$ is essential in Transformers (Vaswani et al., 2017) because it provides a sense of order of sequence to the input data, which is inherently missing in the Transformer architecture due to its non-sequential design. For the i^{th} staypoint, row i of P gives the embedding row vector for the number i.

Rationale of MHSA

Each row i of E' is now an embedding row vector for the i^{th} staypoint's entire information (except user_id). Each row i of the attention output A is thus an attention (relevance) weighted sum of the value (V) information of all rows of E', with each row contributing proportionally based on its relevance to the row of E'. Therefore, this self-attention mechanism captures both short- and long-range dependencies regardless of sequence length. With multiple heads, MHSA can learn richer, multi-perspective representations of the sequence in parallel as each head has its own set of learnable parameters.

Model Compilation

Metric & Loss Function

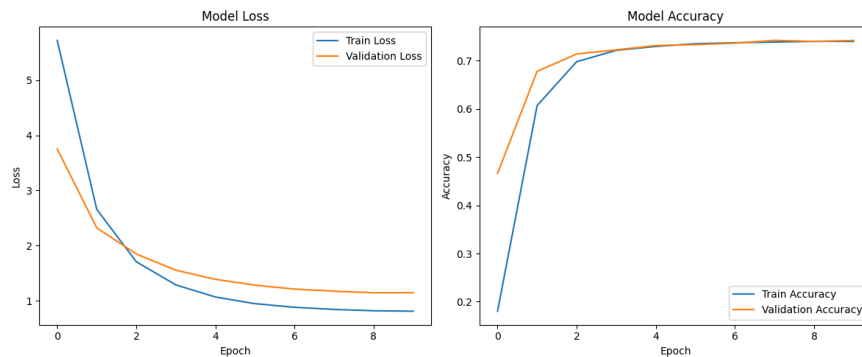
Sparse_categorical_crossentropy (SCCE) was used as loss function (formula in Appendix) as we are using true class labels as ground truth and they are not one-hot encoded. The metric to be reported during training is accuracy (same as top-1 accuracy).

Optimiser

Adaptive Moment Estimation (ADAM) optimiser (equations in Appendix) was used as it is more effective than Stochastic Gradient Descent (SGD) (formula in Appendix). To update each learnable parameter (θ), instead of using gradient at current iteration step t (g_t), ADAM uses (bias-corrected) moving average of past g values (\hat{m}_t), which smooths noisy updates (u_t) due to fluctuating g , since now $u_t = -\alpha_t \hat{m}_t$. Moreover, instead of using a constant learning rate α , at each t , it divides α by square-root of (bias-corrected) moving average of past g^2 values to obtain dynamically adjusted learning rate α_t , so overall those θ with larger $|g|$ (and thus larger \hat{m}_t) will generally have smaller α_t and vice-versa, so $|u_t| = |\alpha_t \hat{m}_t|$ will be balanced in magnitude across all θ and all t , which enables better performance while avoiding extensive hyper-parameter tuning.

Model Evaluation

Learning Curve



We used 60:20:20 train:validation:test split, with 10 epochs, and batch size = 32 to prevent memory overload. The learning curve shows a generally steady increase in accuracy and generally steady decrease in loss, with later epochs having less fluctuations, indicating that our model was able to learn effectively from the data. Also similar values for both the train and validation set mean that there is no overfitting on the train set.

Evaluation Metrics

Mean Reciprocal Rank (MRR)

We sort each row of the predicted probability matrix in descending order, then find the rank of the true class label, then calculate MRR (Hong et al., 2023) (formula in Appendix). Higher value means better, max score = 1

Top-k Accuracy

For $k = 1, 5, 10$, we also calculate the percentage of data inputs whose true `next_location_id` was among the k locations with highest predicted probabilities it

Evaluation Results

	Mean Reciprocal Rank	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy
Train	0.8514834	0.7491299	0.9815505	0.99698144
Validation	0.83921164	0.7421135	0.96038795	0.97703934
Test	0.8381379	0.74122274	0.95875496	0.9763466

Notably, across all the data including the completely unseen test set, our algorithm is able to achieve top-5 accuracy $> 95\%$ and MRR score > 0.83 , meaning it has excellent performance in predicting the next location, i.e. taking only its top 5 predictions for next location of a user almost always contains the correct answer.

Limitations

Due to limited computational time and space, we demonstrated our algorithm's results using a subset (25%) of "kumamoto_challengedata_cityD.csv" data set only, but its results can be simply reproduced on any positionfixes data accompanied with appropriately formatted POI data (geometry, POI category, count). We were also not given mapping to real-world latitude-longitude coordinates so we have to train the model to fit each specific region, otherwise we would have been able to train the model simultaneously on multi-regional data





Conclusion

We successfully implemented a context-aware next-location prediction neural network algorithm. It is based on most effective MHSA transformer architecture and utilises Hong et al's innovative POI land-use context representation using LDA but is largely simplified to concisely use core input features, which optimises the balance between model complexity and accuracy. On unseen test sets, it is able to achieve excellent evaluation results, considering its simplicity. Its predictive performance is also higher than those reported in similar current literature, although this could be biased due to small regional data. For future extension, we can train and test our algorithm on global data using latitude-longitude coordinates to verify if it is truly effective.

References

- Hong, Y., Zhang, Y., Schindler, K., Raubal, M., & The Author(s). (2023). Context-aware multi-head self-attentional neural network model for next location prediction. In *Transportation Research Part C* (Vol. 156, p. 104315) [Journal-article]. <https://doi.org/10.1016/j.trc.2023.104315>
- Vaswani, A., Shazeer, N., Google Brain, Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., University of Toronto, Kaiser, Ł., Google Research, & Polosukhin, I. (2017). *Attention is all you need* [Conference-proceeding]. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Contributions

Chong Geng Yang	Task 2.1	
Ezra Koh See Hwa	Task 2.2	
Jace Aung Kaung Kaung	Task 3	
Jason Lim Jiasheng	Task 3	

Appendix

Preliminary Definitions

A positionfix (track point) is a recorded position of a user, it records the raw time (tracked_at), spatial coordinates (geometry) and the user_id. A staypoint is a set of more than 1 positionfixes such that they are all within distance_threshold of the earliest positionfix and also the duration between earliest and latest positionfixes is more than time_threshold. Intuitively it is a region that the user stays around for some time. It has a user_id, geometry which is centroid of the geometries of its positionfixes, arrival time (started_at) and departure

(finished_at) time. A location is a cluster of staypoints, its centre is centroid of the staypoints and its extent is the convex hull of the staypoints. Each positionfix also has staypoint_id indicating the staypoint it belongs to (null if it does not belong to any). Every positionfix belongs to at most 1 staypoint. Each staypoint also has a location_id indicating its cluster label

Model Details

Parameter settings

We set the embedding dimensions, as well as the following parameters

- sequence_length = 99
- Number of attention heads for MHSA module = 4
- key_dim for MHSA = $92/4 = 23$
- total number of unique users considered = 3000
- Total number of unique locations = 22179

Inputs & Outputs

Each input to our model will have the following features: user_id_input (dimension 1 x 1), location_id_sequence_input, arrival_dayofweek_sequence_input, arrival_time_sequence_input, duration_sequence_input, x_coordinate_input, y_coordinate_input, convex_hull_area_input, POIcontext_sequence_input (dimension 99 x 20). Other than user_id_input and context_sequence_input, the rest are vectors with dimension 99 x 1. For a sample of inputs, our output will be a sample_size x 22179 probability matrix, where the $(i,j)^{th}$ element gives the probability that the next location of data i is location j. next_location_id is used as the ground truth class labels.

Dimensions of Embeddings & Intermediates

For a single data input,

- dimension of user_id embedding = 1 x 16
- dimension of location_id embedding = 99 x 32
- dimension for each location spatial coordinate embedding = 99 x 8
- dimension for staypoint arrival time embedding = 99 x 3
- dimension for staypoint arrival day embedding = 99 x 3
- dimension for staypoint duration embedding = 99 x 8
- dimension for a location's area embedding = 99 x 10
- dimension of POI context "embedding" = 99 x 20
- Dimension of P, E, E', A, Z, M = 99 x 92
- Dimension of mean(M) = 1 x 92
- Dimension of U = 1 x 108

Embedding Row Vector For Time

$$\left[\sin\left(\frac{2\pi t}{T}\right), \cos\left(\frac{2\pi t}{T}\right), \frac{t}{T} \right]$$

where t = time, T = period

Sinusoidal Positional Encoding Formula

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

PE is the positional_encoding_matrix (P in our model).

MHSA Equations

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \quad Q_i = QW_i^Q, \quad K_i = KW_i^K, \quad V_i = VW_i^V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O, \quad \text{head}_i = \text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

$$\begin{array}{l} W_Q \in \mathbb{R}^{D_{model} \times D_{key}} \\ W_K \in \mathbb{R}^{D_{model} \times D_{key}} \\ W_V \in \mathbb{R}^{D_{model} \times D_{value}} \end{array}, \quad \begin{array}{l} W_i^Q \in \mathbb{R}^{D_{key} \times D_{key}} \\ W_i^K \in \mathbb{R}^{D_{key} \times D_{key}} \\ W_i^V \in \mathbb{R}^{D_{value} \times D_{value}} \end{array}$$

$$W^O \text{ has dimensions } \mathbb{R}^{(h \cdot D_{value}) \times D_{model}}$$

Where h is number of attention heads (4 in our model), X is the MHSA input (E' in our model), $D_{model} = 92$ in our model, $D_{key} = d_k = D_{value} = \text{key_dim} = 23$ in our model. All the W matrices are learnable parameters.

Layer Normalisation Formula

1. Calculate the mean for each row:

$$\mu_i = \frac{1}{d} \sum_{j=1}^d Z_{i,j}$$

2. Calculate the standard deviation for each row:

$$\sigma_i = \sqrt{\frac{1}{d} \sum_{j=1}^d (Z_{i,j} - \mu_i)^2}$$

3. Apply learnable scaling and shifting:

$$M_{i,j} = \gamma_j \cdot \frac{Z_{i,j} - \mu_i}{\sigma_i} + \beta_j$$

SCCE Formula

$$\text{Loss} = -\frac{1}{N} \sum_{i=1}^N \log(p_{i,y_i})$$

Where:

- N is the number of samples in the batch.
- y_i is the true class label (integer) for sample i .
- p_{i,y_i} is the predicted probability for the true class y_i of sample i .

ADAM Optimizer For Single Learnable Parameter θ

Equations

1. Initialization:

- Set the time step $t = 0$.
- Initialize first moment $m = 0$ (moving average of the gradient).
- Initialize second moment $v = 0$ (moving average of the squared gradient).
- Set the learning rate α (commonly 0.001).
- Set the decay rates β_1 and β_2 (typically $\beta_1 = 0.9$ and $\beta_2 = 0.999$).
- Set a small constant ϵ (typically 10^{-8}) for numerical stability.

2. Update at Each Time Step t :

- Increment the time step: $t = t + 1$.

3. Compute the Gradient of the loss function with respect to θ :

$$g_t = \frac{\partial \text{Loss}}{\partial \theta_t}$$

4. Update Biased First Moment Estimate (moving average of the gradient):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

5. Update Biased Second Moment Estimate (moving average of the squared gradient):

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

6. Compute Bias-Corrected First Moment:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

7. Compute Bias-Corrected Second Moment:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

8. Update the Parameter:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Explanation of Each Term

- g_t : Gradient of the loss function with respect to the parameter θ at time step t .
- m_t : Biased first moment estimate, which is an exponential moving average of past gradients.
- v_t : Biased second moment estimate, which is an exponential moving average of past squared gradients.
- \hat{m}_t : Bias-corrected first moment estimate.
- \hat{v}_t : Bias-corrected second moment estimate.
- α : Learning rate.
- β_1 and β_2 : Decay rates for the first and second moments.
- ϵ : Small constant for numerical stability, to prevent division by zero in the parameter update.

Rationale for Bias Correction

Since m and v are initialised to 0, with β meant to be set close to 1, they will be biased towards 0.

1. Initial Steps (Small t):

- For small t , β_1^t and β_2^t are close to 0. This makes $1 - \beta_1^t$ and $1 - \beta_2^t$ close to 1, which means the correction factors $\frac{1}{1-\beta_1^t}$ and $\frac{1}{1-\beta_2^t}$ are large.
- These large correction factors compensate for the fact that m_t and v_t are initially small, effectively "boosting" their values to approximate what they would be if they had been accumulating data from the start (rather than starting from zero).

2. As t Increases:

- As t grows, β_1^t and β_2^t approach 1, so $1 - \beta_1^t$ and $1 - \beta_2^t$ approach 0.
- When t is large, the correction factors $\frac{1}{1-\beta_1^t}$ and $\frac{1}{1-\beta_2^t}$ approach infinity as well, meaning that the bias correction has a negligible effect once the moving averages are fully "warmed up" and have accumulated enough data.

SGD Formula For Single Learnable Parameter θ

$$\theta_{t+1} = \theta_t - \alpha \cdot g_t$$

MRR Formula

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^N \frac{1}{\text{rank}_i}$$