Generation of Synthetic Trajectory Microdata from Language Models

Alberto Blanco-Justicia(✉), Najeeb Moharram Jebreel, Jesús A. Manjón, and Josep Domingo-Ferrer

Department of Computer Engineering and Mathematics,
Universitat Rovira i Virgili, Av. Paisos Catalans 26,
43007 Tarragona, Catalonia
{alberto.blanco,najeeb.jebreel,jesus.manjon,josep.domingo}@urv.cat

Abstract. Releasing and sharing mobility data, and specifically trajectories, is necessary for many applications, from infrastructure planning to epidemiology. Yet, trajectories are highly sensitive data, because the points visited by an individual can be identifying and also confidential. Hence, trajectories must be anonymized before releasing or sharing them. While most contributions to the trajectory anonymization literature take statistical approaches, deep learning is increasingly being used. We observe that natural language sentences and trajectories share a sequential nature that can be exploited in similar ways. In this paper, we present preliminary work on generating synthetic trajectories using machine learning models typically used for natural language processing. Our empirical results attest to the quality of the generated synthetic trajectories. Furthermore, our methods allow discovering natural neighborhoods based on trajectories.

Keywords: Privacy · Synthetic data generation · Mobility data

1 Introduction

Personal mobility data in their simplest form are data about individuals that include their locations at specific times. Sources of real-time raw individual location data include, but are not limited to, cell towers, Wi-Fi access points, RFID tag readers, location-based services, or credit card payments. Historical location data, in the form of data sets in which each of the records corresponds to an individual and includes her location data for some time periods, are referred to as trajectory microdata sets. Such trajectory microdata sets are often interesting to transport authorities, operators, and other stakeholders to evaluate and improve their services, the state of the traffic, etc. Recently, due to the COVID-19 pandemic, the health authorities have also become interested in mobility data to predict the spread of infectious diseases.

The above landscape motivates the need to share or even publicly release mobility data. Sharing is occasionally done at an aggregate level (*e.g.*, heat maps),

J. Domingo-Ferrer and M. Laurent (Eds.): PSD 2022, LNCS 13463, pp. 172–187, 2022.
https://doi.org/10.1007/978-3-031-13945-1_13

rather than at an individual level. Whichever the specific type of mobility data, sharing them entails a potential privacy risk. Mobility data are highly unique and regular. *Unicity* refers to the data of different individuals being easily differentiable, particularly at some specific locations. The starting and ending locations of an individual's trajectories are often their home and work locations which, again, are highly unique and can lead to reidentification. In [4] it is shown that individual full trajectories can be uniquely recovered with the knowledge of only 2 locations, and knowledge of 4 locations can lead to full reidentification of 95% of individuals in data set containing trajectories for 1.5 million individuals. The *regularity* of trajectories implies that each individual's data follows periodic patterns. Namely, individuals tend to follow the same trajectories during workdays–home to work and back to home.

These features may allow attackers with publicly available data or background knowledge about an individual (such as place of work) to infer sensitive information about that individual, including health status, religious beliefs, social relationships, sexual preferences, etc.

Our interest in this paper is trajectory microdata. A trajectory is a list of spatio-temporal points visited by a mobile object. A trajectory microdata set contains a set of trajectories, where each trajectory normally corresponds to a different individual. The points in each trajectory are both quasi-identifiers and confidential information: indeed, some locations can be very identifying (*e.g.* the trajectory origin can be the individual's home) and other locations can be very confidential (*e.g.* if the individual visited a hospital, a church or a brothel). Thus, anonymizing trajectories is not easy. Several anonymization mechanisms have been proposed, but most of them do not provide solid privacy guarantees or distort the data too much [9]. An alternative to releasing anonymized trajectories is to generate synthetic trajectories that do not correspond to any specific real trajectory [18].

Contribution and Plan of this Paper

We propose to leverage deep learning models used in natural language processing, and in particular for next-word prediction, to generate synthetic trajectory data. A key idea in the proposal is that road networks impose a context to people's movements, and so there is semantics connected to the transition from one trajectory point to the next. Capturing this semantics is something that modern language models based on deep learning have shown to excel at.

Section 2 reviews related work on trajectory data protection and synthetic generation, including methods based on deep learning. Section 3 describes our mechanism for synthetic trajectory data generation. Section 4 shows the results of our experimental evaluation. Finally, Sect. 5 concludes the paper and highlights ideas for future work.

2 Related Work

2.1 Sequential Models for Trajectory Prediction

Since trajectories have the same sequential nature as natural language sentences, sequence models used for next-word prediction have also been extended to next location prediction task. [2,21] use recurrent neural networks (RNNs) [19] and their variations in the next location prediction task. RNNs have shown superior performance due to their ability to capture trajectory data's latent spatial and temporal features. [25] utilise the bidirectional long-short-term memory (BiLSTM) model and the similarity-based Markov model (SMM) to predict the individuals' next locations while maintaining the semantic and spatial patterns of the individuals' trajectories.

2.2 Privacy-Preserving Tajectory Data Publishing

Existing methods for privacy-preserving trajectory publishing can be divided into statistical methods and deep learning (DL)-based methods.

Statistical methods rely on one of the following principles [9,13]: (i) suppression by removing points of trajectories that can identify individuals; (ii) generalization by making the trajectories indistinguishable via grouping them into larger ranges; (iii) distortion by using differential privacy (DP) to ensure that the presence of a record in a data set leaks a controlled amount of information; and (iv) perturbation by using techniques like location merging, clustering, or generating virtual trajectories.

Most of the proposed works in the literature adopt one or more of the techniques above to release privacy-preserving trajectory data. For instance, NWA [1] anonymizes trajectories following a two-step procedure: 1) building clusters of at least k similar trajectories, and 2) anonymizing trajectories in each cluster to produce k-anonymous trajectory data. GLOVE [12] adopts a different procedure with two steps as well: 1) computing trajectory-wise merge costs and 2) iteratively building clusters by merging two trajectories with the smallest cost until satisfying k-anonymity. [6] use microaggregation clustering to group trajectories according to their similarity and then replace them with group representatives. [7] group similar trajectories and remove some of them to ensure k-anonymity. KTL [22] adapts both $l-$diversity and $t-$closeness to trajectory data to counter attacks facilitated by k-anonymity (e.g., attribute linkage). [3,14] adopt DP-based methods to release distorted trajectories. [14] merge coexistent points from different trajectories using a partitioning procedure based on the exponential DP mechanism, whereas [3] propose a mechanism for perturbing semantic trajectories that satisfies ϵ-local DP.

However, statistical methods generally do not provide a proper trade-off between the utility and privacy of their published trajectory data [24]. Non-DP methods, to some extent, maintain the utility of the published data, but they are vulnerable to several privacy attacks (e.g., attribute linkage and background knowledge attacks). Although l-diversity and t-closeness methods offer

better protection against privacy attacks, they can have a negative impact on the utility [16]. On the other hand, DP-based methods attempt to make the presence or absence of any single record unnoticeable from the protected output, which makes such methods ill-suited to protect microdata (records corresponding to individual subjects) without (i) severely reducing the data utility or (ii) significantly degrading the privacy guarantee being offered [5].

DL-based methods aim to generate synthetic trajectories that can realistically reproduce the patterns of individuals' mobility [18]. The intuition is that the generated synthetic data come from the same distribution of real trajectories (thereby preserving utility). At the same time, they do not correspond to real trajectories (thereby preserving privacy).

Existing DL methods leverage sequence natural language processing (NLP) models, such as RNNs [19], or generative models, such as generative adversarial networks (GANs) [11], to approximate the distribution of the real trajectory data and then sample synthetic trajectories from that distribution.

[10,17] exploit the ability of RNNs to model problems over sequential data having long-term temporal dependencies. Like training a next-word prediction model, they train a next location prediction model using the real trajectory data as training data. Then, they construct a synthetic trajectory by starting at some arbitrary location and iteratively feeding the current output trajectory sequence as input to the next step in the trained model.

GANs [11] set up a game between two neural networks: the generator G and the discriminator D. G's goal is to generate "synthetic" data classified as "real" by D, whereas D's goal is to correctly distinguish between real and synthetic data and provide feedback to G to improve the realism of the generated data. trajGAN [24] consists of a generator G which generates a dense representation of synthetic trajectories from a random input vector z and a discriminator D, which classifies input trajectory samples as "real" or "fake". To capture contextual and hidden mobility patterns and generate more realistic trajectories, trajGAN [24] uses RNNs to create dense representations of trajectories. SVAE [15] builds its generator G based on an LSTM and a Variational Autoencoder (VAE) to combine the ability of LSTMs to process sequential data with the ability of VAEs to construct a latent space that captures key features of the training data. MoveSim [8] uses a self-attention-based sequential model as a generator to capture the temporal transitions in human mobility. In addition, the discriminator uses a mobility regularity-aware loss to distinguish real from synthetic trajectories. [23] propose a two-stage GAN method (TSG) to generate fine-grained and plausible trajectories. In the first stage, trajectories are transformed into a discrete grid representation and passed as input for a generative model to learn the general pattern. In the second stage, inside each grid, an encoder-decoder generator is used to extract road information from the map image and then embed it into two parallel LSTMs to generate trajectory sequences.

Although DL-based methods have shown promising performance in generating high-utility synthetic trajectories, privacy issues are likely to arise due to overfitting the trained models on the original training data [18]. Consequently,

a synthetic trajectory may resemble a real one and give an attacker the chance to use this information for re-identification. In our proposed work, we adopt a DL-based method to generate plausible synthetic trajectories and also mitigate the above-mentioned privacy risk by integrating a randomization mechanism during the synthetic trajectory generation phase.

3 Synthetic Trajectory Generation Method

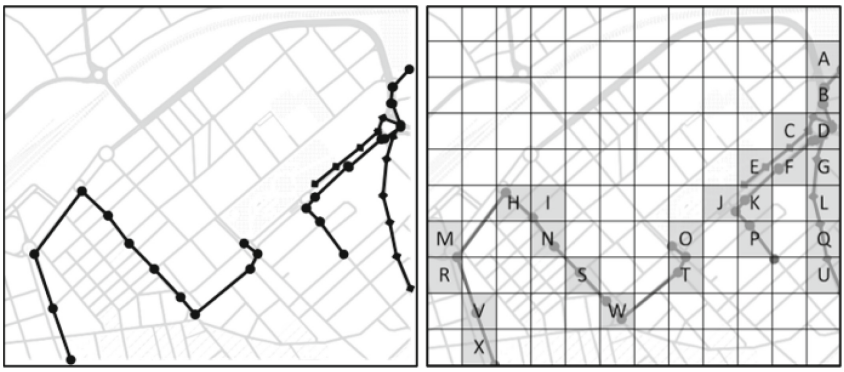
This section presents our proposed mechanism for synthetic trajectory data generation. We first explain our approach to preprocess the original data, to convert them from lists of spatio-temporal points into sequences of labels. Then, we describe the BiLSTM neural network architecture, which we use to train a next-point prediction model. Finally, we present the data generation process, in which we use the randomization of next-point predictions to limit or prevent the release of trajectories or subtrajectories present in the original data.

3.1 Data Preprocessing

The first step is to preprocess the trajectory data so that they are amenable to be used as training data for a natural language processing model. Trajectory microdata contain spatio-temporal points (id, x, y, t), where id is a trajectory identifier, (x, y) is a latitude-longitude location and t is the time at which the location was visited by the moving object. Analogously, NLP models take sequences of tokens representing words (or parts of words) and punctuation. In our preprocessing, we first define a bounding box around the area of interest and discard outlying points. For example, when dealing with trajectory data in a given city, trajectories that depart from the city to a far away area are not of great interest.

Then, we build a grid of an arbitrary resolution within the bounding box. The more resolution, the better accuracy we can obtain from further analysis of the generated data, but also the more resources we will need in order to train the generator model. The grid resolution also has an effect on the privacy properties of the generated data. Continuing with the NLP example, the more resolution, the bigger the dictionary of words that our model has to deal with. Next, we assign each of the points in the data set to cells in the grid and label cells using an invertible encoding function (such as alphabetic labeling or a number computed as row $\times$ number of columns + column). Once each of the points is encoded as a label, we discard all grid labels that do not appear in the data set, so as to reduce the dictionary size, and recode the labels to the range $[0 \ldots \#labels - 1]$. At this point, each of the trajectories is a sequence of labels, similar to what sentences are.

In addition, we compute and store the distribution of trajectory lengths and discard trajectories with outlying lengths, again to save training resources. This length distribution will later be used during the trajectory generation process.



(a) Original trajectory data (b) Tessellation and cell labeling

Fig. 1. From trajectories to sequences of labels

Finally, we obtain the training (and validation) data by extracting n-grams from the trajectories, using a growing window, and taking the last point in each n-gram as the label for next-word prediction.

3.2 Next-Point Prediction Model

After preprocessing the training data, we use the bidirectional long short-term memory (BiLSTM) model to solve the trajectory next-point prediction task. BiLSTM's main advantage over the other sequence models is that its input flows from the past to the future and vice versa, making BiLSTMs a powerful tool for modeling the sequential dependencies between trajectory points. Since the presence of an individual at a specific location is usually influenced by the previous and next locations the individual visits, BiLSTM is expected to capture those local contextual patterns. Moreover, training BiLSTM on the trajectory data of many individuals' visited points within a limited geographic area is expected to capture the global pattern of the individuals' mobility in that area. Therefore, BiLSTM is expected to predict the individuals' next points more accurately than other models.

Figure 2 shows the architecture of the proposed BiLSTM-based model for the next-point prediction.

The first layer of the model takes a processed trajectory as input, which is represented as sequence of points $(p_{t-1}, p_t, \ldots, p_{t+k})$, where p_t is the point an individual visited as time t. The embedding layer then maps each point in the processed trajectory to a multidimensional vector with a predefined length so the BiLSTM units can process it. Then, the BiLSTM units run the obtained embeddings in two ways, one from past to future and one from future to past, to preserve information from both past and future. Finally, the softmax layer uses the output vectors of the BiLSTM units to produce a vector of probabilities $\in \mathcal{R}^L$ of the next point, where L is the total number of labels.

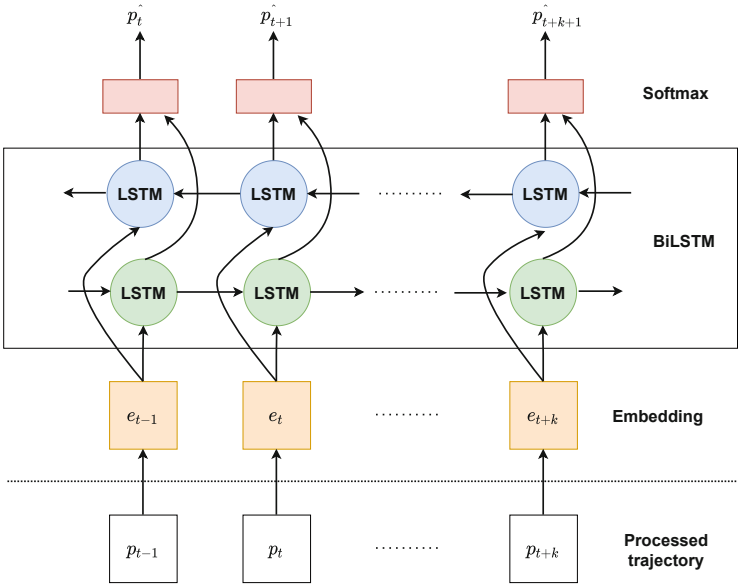


Fig. 2. Proposed BiLSTM-based model for next-point prediction

3.3 Synthetic Data Generation

The generation process starts by choosing the trajectory lengths (number of points per trajectory) from the original length distribution obtained during the preprocessing of the training data. Then, we generate batches of synthetic trajectories of the same length to leverage on the parallelization capabilities of DL models.

To generate each of the synthetic trajectories, we start by drawing a random location label from the dictionary and feeding it to the trained model to predict the next point. Then, we append the predicted point to the trajectory and feed it again to the model. We repeat this process until we obtain trajectories of the desired length.

One potential issue with this approach is that the model learns a 1-to-1 representation or mapping of the training data. This is especially (but not only) possible when ML models overfit the training data. In these cases, the synthesized trajectories are likely to mimic trajectories or sections of trajectories from the training data (which would be akin to sampling them from the training data), and thus not be truly synthetic. The solution we propose in this case is to collect the top-k predictions of the next point and choose one of them uniformly at random. In this case, even if the model is a 1-to-1 mapping of the original data, the probability of a trajectory or sub-trajectory being equal to one in the training data falls to $(1/k)^{length}$. Even so, if the data were big and diverse enough to contain any single possible trajectory in a grided area, all synthetic data generated from them would necessarily be a sample of the original data.

The above described top-k fix can still have issues if the distribution of probabilities for the top k next-point candidates is close to uniform, or if there is a clear peak for the first candidate and very similar probabilities for the rest (which is often a symptom of overfitting). In such cases, the quality of the generated data can decrease, introducing artifacts in the trajectories, such as very long transitions from one point to the following one. The k parameter has to be adequately tuned to avoid these issues. In our experiments, we took $k = 3$.

4 Experimental Analysis

We run experiments that generate synthetic trajectory data out of two public mobility data sets, namely the Cabspotting data set and the GeoLife data set.

4.1 Data Sets and Preprocessing

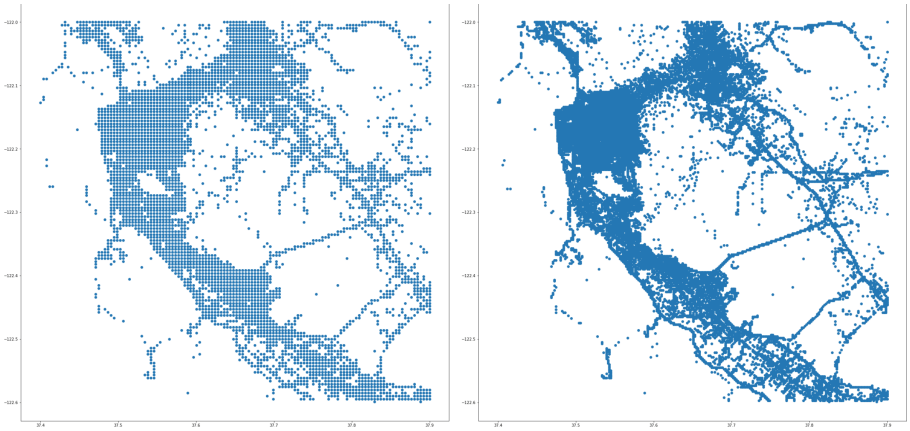
The Cabspotting data set [20] contains trajectories of occupied taxi cabs in San Francisco and the surrounding areas in California. The data set contains trajectories of nearly 500 taxis collected over 30 days. The trajectories consist of points containing each a GPS location, a timestamp and an identifier. In our preprocessing, we just keep points with longitudes between $-122.6°$ and $-122.0°$ and latitudes between $37.0°$ and $37.4°$. These points define an area of $2,938.38$ km^2 around the San Francisco Bay area. Next, we define two grids with different resolutions to generate two different data sets:

San Francisco 128: This is a grid of 128×128 cells, which results in $16,384$ cells, of which $4,944$ are visited at least once. These will become the labels in our location dictionary for the first data set. They are shown in Fig. 3a.

San Francisco 256: This is a grid of 256×256 cells, or a total $65,536$ cells, of which $12,393$ are visited at least once. These are the labels for the second data set. Figure 3b shows the unique locations in the dictionary.

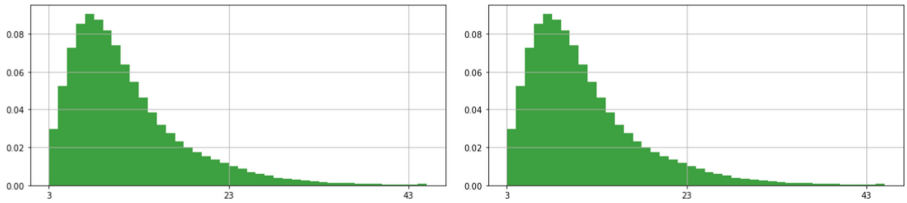
In both cases, we keep only trajectories with a length between 3 and 45 locations, which result in $437,335$ distinct trajectories. Figure 4 shows the distribution of the lengths of trajectories in the San Francisco 128 and 256 data sets.

The GeoLife data set [26–28] is a trajectory microdata set collected by Microsoft Research Asia during the Geolife project. The data collection process was carried out by 182 users during a period of over 3 years (2007–2012). Each of the trajectories consists of a sequence of timestamped locations, collected with varying sampling rates of one point every 1–5 s or every 5–10 m. The data set collects different mobility modes and activities. Again, we keep only points within longitudes $115.9904°$ and $116.8558°$ and latitudes $39.67329°$ and $40.22726°$, which yields an area of $4,548.78$ km^2 around the city of Beijing. We generate two data sets by defining two different grids of spatial resolutions 128×128 and 256×256:



(a) San Francisco 128 (b) San Francisco 256

Fig. 3. Unique locations in the San Francisco data set for resolutions 128 and 256



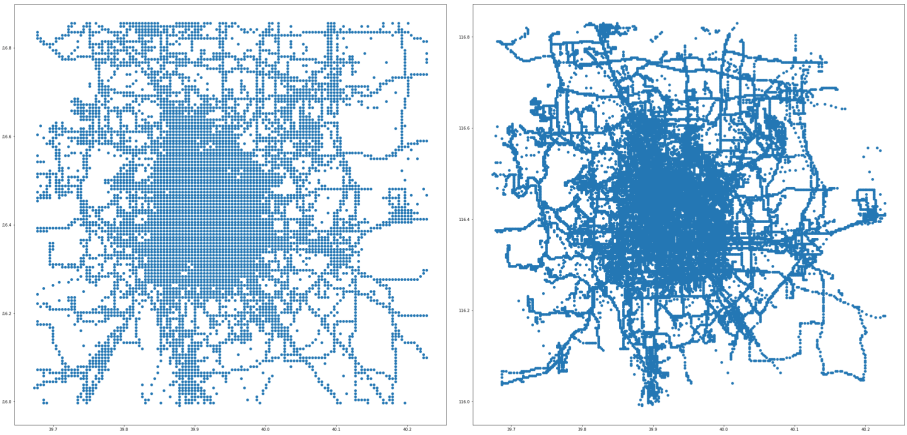
(a) San Francisco 128 (b) San Francisco 256

Fig. 4. Distribution of the lengths of trajectories in the San Francisco 128 and San Francisco 256 data sets

Beijing 128: The first data set is generates by tessellating the given area using a grid of 128×128 cells, which results in $16,384$ cells, of which $7,079$ are visited at least once. Thus, the location dictionary of the Beijing 128 data set consists of those $7,079$ unique locations. Figure 5a shows these unique locations in the dictionary. We keep trajectories with a number of points between 3 and 70, which results in a total $36,827$ distinct trajectories.

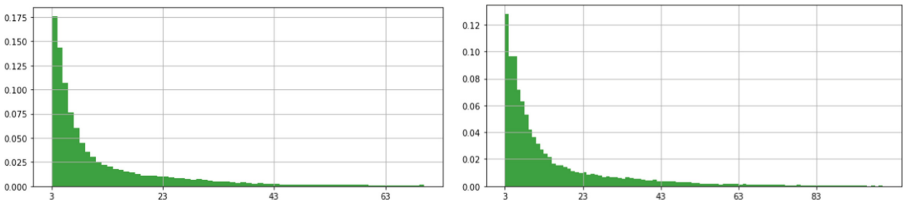
Beijing 256: The second grid consists of 256×256 cells, or a total $65,536$ cells, of which $15,831$ are visited at least once. These are the labels for the Beijing 256 data set. In this case, we keep $43,741$ trajectories, which have lengths between 3 and 100 points. Figure 5b shows the unique locations in the dictionary.

Figure 6 shows the distribution of the lengths of trajectories in the Beijing 128 and 256 data sets.



(a) Beijing 128 (b) Beijing 256

Fig. 5. Unique locations in the Beijing data set for resolutions 128 and 256



(a) Beijing 128 (b) Beijing 256

Fig. 6. Distribution of the lengths of trajectories in the Beijing 128 and Beijing 256 data sets

In all four data sets, we completed the preprocessing phase by generating the n-grams from the trajectories and keeping the last point in each of the n-grams as the classification label for the next-point prediction models.

4.2 Model Training

For the **San Francisco 128** data set, we trained a BiLSTM consisting of an embedding layer of input dimension $4,944$ and output dimension 64, followed by two BiLSTM layers of 256 and 128 units, respectively, and a dense layer of 128 units. The output layer consisted of $4,944$ units with the softmax activation function. Dropout layers were included before and after the second BiLSTM layer, with a dropout rate of 20%. The model was trained for 250 epochs with a batch size of 250 and a validation split of 20%, using the Adam optimizer on the sparse categorical crossentropy loss function. The model obtained an accuracy of 48.53% and a validation accuracy of 48.7%.

In the case of the **San Francisco 256** data, we used a similar architecture and training process, except for the embedding layer's input dimension and the number of units of the output layer, which were set to 12,393, according to the dictionary size. The accuracy of the model was 33.86%, while its validation accuracy was 34.3%.

For the **Beijing 128** and **Beijing 256** data sets, we increased the BiLSTM layer sizes to 512 and 256, respectively. The embedding layer's dimensions and output layer sizes were set to 7,079 for Beijing 128 and 15,831 for Beijing 256, according to the sizes of their label dictionaries. The model trained on Beijing 128 obtained an accuracy of 79.3% and a validation accuracy of 62.29%. The one trained on Beijing 256 obtained a 75.40% accuracy and a 47.81% validation accuracy. These two models show overfitting to the training data, partly because there are many fewer trajectories in these two data sets than in the San Francisco data sets with respect to the number of location labels (7,079 labels for 36,827 trajectories in Beijing 128, and 15,831 labels for 43,741 trajectories in Beijing 256).

4.3 Results of Data Generation

Finally, we generated 20 synthetic data sets for each of the 4 training data sets (San Francisco 128 and 256, and Beijing 128 and 256) both using the top-1 prediction and our randomized strategy, that is, choosing the next point randomly among the top-k predictions, for $k = 3$. Each of the synthetic data sets consisted of 400 synthetic trajectories. For comparison, we also sampled the 4 training data sets, again drawing 20 samples of 400 trajectories for each of them. Figures 7 and 8 show examples of the sampled and generated data sets.

As mentioned before, both Beijing synthetic data sets show some artifacts in the shape of long jumps across points. This is partly the effect of being smaller data sets and also the effect of overfitting.

In order to assess the quality of the generated data, we compared the original samples with the synthetic data sets, according to the following metrics:

- d_{SL}, computed as the mean of the sum of the distances (in km) between all consecutive locations over all trajectories.
- Δr, defined as the average distance (in km) between any two consecutive points over all trajectories.
- d_{max}, computed as the mean maximum distance (in km) between any two consecutive locations over all trajectories.
- $\#locs$, obtained as the mean number of distinct locations visited in each trajectory.
- The mean number of visits per unique location $\#V/loc$ for all trajectories.

Table 1 shows the results for each of the data sets, averaged over the 20 different samples drawn or generated for each of them.

In the case of the San Francisco synthetic data sets, the results show smaller values for d_{SL}, Δr, and d_{max} under the top-1 and top-3 predictions than in

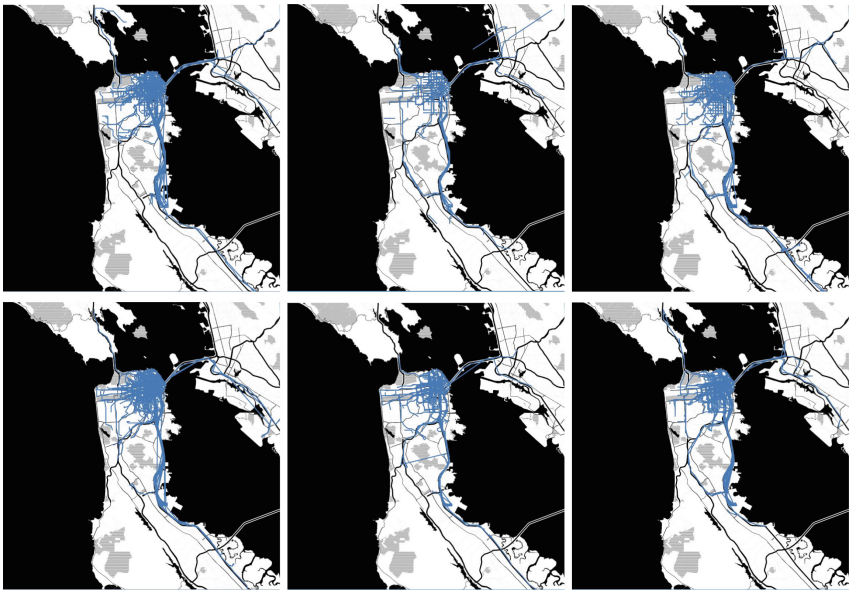


Fig. 7. Examples of San Francisco trajectories. Top figures use a 128×128 grid and bottom figures use a 256×256 grid. Figures in the left are samples from the original data set, figures in the middle show the results of using top-1 predictions, while figures in the right use random top-3 predictions.

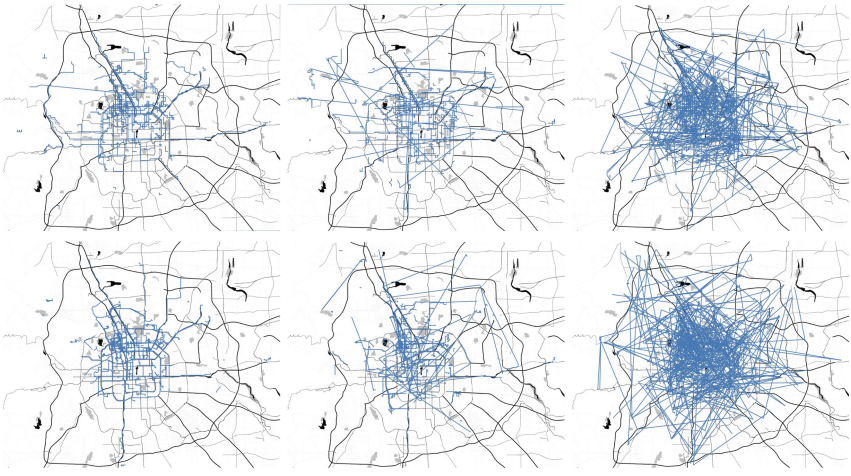


Fig. 8. Examples of Beijing trajectories. Top figures use a 128×128 grid and bottom figures use a 256×256 grid. Figures in the left are samples from the original data set, figures in the middle show the results of using top-1 predictions, while figures in the right use random top-3 predictions.

Table 1. Quality metrics for the generated data

City	Resolution	Method	d_{SL}	Δr	d_{max}	$\#locs$	$\#V/loc$
San Francisco	128	sample	5.94	0.78	1.16	8.40	6.87
		top-1	4.50	0.60	0.84	8.30	8.84
		top-3	4.96	0.67	1.09	8.13	7.20
	256	sample	5.73	0.64	1.11	9.83	3.78
		top-1	3.94	0.45	0.81	9.57	4.81
		top-3	4.70	0.54	1.16	9.52	4.11
Beijing	128	sample	5.91	0.56	0.71	10.37	3.43
		top-1	6.26	0.62	1.31	7.83	4.01
		top-3	14.5	1.45	4.90	8.38	3.41
	256	sample	4.79	0.35	0.48	13.4	2.47
		top-1	5.81	0.45	1.82	9.47	3.00
		top-3	21.47	1.69	6.71	10.93	2.47

the sampled trajectories. This indicates that the synthetic trajectories tend to be slightly shorter than the original ones, especially under the top-1 prediction. The $\#locs$ metric shows similar results among the sampled and synthetic data sets, while the $\#V/loc$ metric show slightly higher values for the synthetic data. This, together with the shorter trajectories, seem to indicate that the trajectories are more concentrated in the synthetic data than in the training data sets. This might be caused by the sometimes limited capability of RNNs and BiLSTMs to capture long-range relationships. ML models based on transformers seem to capture these relationships better, and we plan to conduct experiments in this regard in the future.

Regarding the Beijing data, the results reveal the effects of the artifacts described above, reflected by values for d_{SL}, Δr, and d_{max} under top-1 and top-3 that are much higher than those in the sampled data (especially in the top-3 case). The number of locations, and the number of visits per location, however, do not show such big deviations.

4.4 Additional Remarks on the Experimental Work

One point of independent interest appears when analyzing the vector embeddings learnt by the models, especially in the case of the San Francisco data set. Figure 9 shows a clustering of the vector embeddings projected on the San Francisco map together with a neighborhood division of the same city[1]. While not exactly aligned, the clustering shows three big areas in the west and three in the east, surrounding a central area with several divisions, similar to that in the neighborhood map.

[1] A SF local's guide to the neighborhoods of San Francisco. https://sfgal.com/sf-locals-guide-to-neighborhoods-of-san-francisco/.

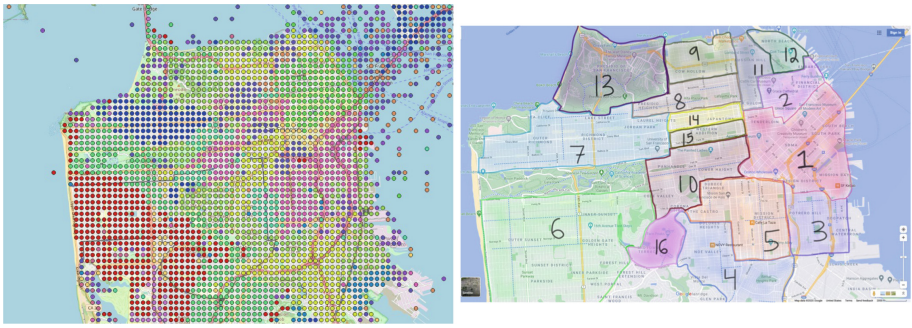


Fig. 9. Clustering of embeddings compared to a map of neighborhoods

Such results suggest that analyzing the vector embeddings resulting from trajectory data sets might be of use to other applications based on city areas, possibly together with additional information. One possible example of application would be the prediction of house prices in different areas of a city.

5 Conclusions and Future Work

In this paper, we have shown preliminary work on the generation of synthetic trajectory microdata using machine learning models typically used for natural language processing and time series. We have shown the potential of such approaches and proposed a strategy to limit the possible data leakages from the data generation. An independent result arising from the study of the vector embeddings learnt during the training process might be of interest in other related areas.

As future work, we plan to replicate the experiments using modern architectures for NLP based on transformers, which have shown a higher performance in NLP tasks than BiLSTMs, especially regarding long-term relationships between concepts. Additionally, we plan to study the effects of differential privacy in the synthetic data, using different approaches such as DP-SGD and PATE. Other potential research directions involve including the time dimension in the generated data and replicating the process in distributed or federated learning scenarios.

Acknowledgements. This research was funded by the European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab"), and the Government of Catalonia (ICREA Acadèmia Prize to J. Domingo-Ferrer, FI grant to N. Jebreel). The authors are with the UNESCO Chair in Data Privacy, but the views in this paper are their own and are not necessarily shared by UNESCO.

References

1. Abul, O., Bonchi, F., Nanni, M.: Never walk alone: Uncertainty for anonymity in moving objects databases. In: 2008 IEEE 24th International Conference on Data Engineering, pp. 376–385. IEEE, 7 April 2008

2. Al-Molegi, A,. Jabreel, M., Ghaleb, B.: STF-RNN: space time features-based recurrent neural network for predicting people next location. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7. IEEE, 6 December 2006
3. Cunningham, T., Cormode, G., Ferhatosmanoglu, H., Srivastava, D.: Real-world trajectory sharing with local differential privacy. arXiv preprint arXiv:2108.02084. 4 August 2021
4. De Montjoye, Y.-A., Hidalgo, C.A., Verleysen, M., Blondel, V.D.: Unique in the crowd: the privacy bounds of human mobility. Sci. Rep. **3**(1), 1–5 (2013)
5. Domingo-Ferrer, J., Sáinches, D., Blanco-Justicia, A. The limits of differential privacy (and its misuse in data release and machine learning). Commun. ACM **64**(7), 33–35 (2021)
6. Domingo-Ferrer, J., Trujillo-Rasua, R.: Microaggregation- and permutation-based anonymization of movement data. Inf. Sci. **15**(208), 55–80 (2012)
7. Dong, Y., Pi, D.: Novel privacy-preserving algorithm based on frequent path for trajectory data publishing. Knowl. Based Syst. **15**(148), 55–65 (2018)
8. Feng, J., Yang, Z., Xu, F., Yu, H., Wang, M,. Li, Y.: Learning to simulate human mobility. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3426–3433. 25 August 2020
9. Fiore, W., et al.: Privacy in trajectory micro-data publishing: a survey. Trans. Data Privacy **13**, 91–149 (2020)
10. Gao, Q., Zhou, F., Zhang, K., Trajcevski, G., Luo, X., Zhang, F.: Identifying human mobility via trajectory embeddings. In: IJCAI, vol. 17, pp. 1689–1695, 19 August 2017
11. Goodfellow, I., et al.: Generative adversarial nets. Adv. Neural Inf. Process. Syst. **27** (2014)
12. Gramaglia, M., Fiore, M.: Hiding mobile traffic fingerprints with glove. In: Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, pp. 1–13, 1 December 2015
13. Guerra-Balboa, P., Pascual, A.M., Parra-Arnau, J,. Forné, J.: Strufe. Anonymizing trajectory data: limitations and opportunities (2022)
14. Hua, J., Gao, Y., Zhong, S.: Differentially private publication of general time-serial trajectory data. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 549–557, IEEE, 26 April 2015
15. Huang, D., et al.: A variational autoencoder based generative model of urban human mobility. In: 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 425–430. IEEE, 28 March 2019
16. Jin, F., Hua, W., Francia, M., Chao, P., Orlowska, M., Zhou, X.: A survey and experimental study on privacy-preserving trajectory data publishing. TechRxiv (2021)
17. Kulkarni, V., Garbinato, B.: Generating synthetic mobility traffic using RNNs. In: Proceedings of the 1st Workshop on Artificial Intelligence and Deep Learning for Geographic Knowledge Discovery, pp. 1–4, 7 November 2017
18. Luca, M., Barlacchi, G., Lepri, B., Pappalardo, L.: A survey on deep learning for human mobility. ACM Comput. Surv. (CSUR) **55**(1), 1–44 (2021)
19. Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S.: Recurrent neural network based language model. In Interspeech **2**(3), 1045–1048 (2010)
20. Piorkowski, M, Sarafijanovic-Djukic, N., Grossglauser, M.: CRAWDAD data set EPFL/mobility (v. 2009–02-24). Traceset: cab, downloaded from February 2009
21. Rossi, A., Barlacchi, G., Bianchini, M., Lepri, B.: Modelling taxi drivers' behaviour for the next destination prediction. IEEE Trans. Intell. Transp. Syst. **21**(7), 2980–2989 (2019)

22. Tu, Z., Zhao, K., Xu, F., Li, Y., Su, L., Jin, D.: Protecting trajectory from semantic attack considering k-anonymity, l-diversity, and t-loseness. IEEE Trans. Netw. Serv. Manag. **16**(1), 264–78 (2018)
23. Wang, X., Liu, X., Lu, Z., Yang, H.: Large scale GPS trajectory generation using map based on two stage GAN. J. Data Sci. **19**(1), 126–41 (2021)
24. Xi, L., Hanzhou, C., Clio, A.: trajGANs: using generative adversarial networks for geo-privacy protection of trajectory data. Vision paper (2018)
25. Xu, M., Han, J.: Next location recommendation based on semantic-behavior prediction. In: Proceedings of the 2020 5th International Conference on Big Data and Computing, pp. 65–73, 28 May 2020
26. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.-Y.: Understanding mobility based on GPS data. In: Proceedings of ACM Conference on Ubiquitous Computing (UbiComp 2008), Seoul, Korea, pp. 312–321. ACM Press (2008)
27. Zheng, Y., Xie, X., Ma, W.-Y.: GeoLife: a collaborative social networking service among User, location and trajectory. IEEE Data Eng. Bull. **33**(2), 32–40 (2010)
28. Zheng, Y., Zhang, L., Xie, X., Ma, W-.Y.: Mining interesting locations and travel sequences from GPS trajectories. In: Proceedings of International conference on World Wild Web (WWW 2009), Madrid, Spain, pp. 791–800. ACM Press (2009)