Trajectory Completion via Context-Guided Neural Filtering and Encoding

Di Yao[1*], Fangda Guo[1*], Zhenyu Wen[2], Yu Guo[3], Peng Cheng[4], Yangyuan He[5], and Jingping Bi[1+]

[1] CAS Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences
[2] Zhejiang University of Technology
[3] Beijing Normal University
[4] East China Normal University
[5] China University of Petroleum at Karamay
`bjp@ict.ac.cn`

Abstract. Trajectories have been massively collected in a wide range of domains and play a critical role in data-driven task support. However, trajectories are often highly sparse and incomplete, which has become a key bottleneck that limits the applicability of trajectory analysis techniques. While many existing sequential models are seemingly applicable to the trajectory completion problem, they often suffer severely from data sparsity and irregularity and yield poor performance in practice. We propose an effective method, named TrajCom, for completing sparse and irregular trajectories. To address data sparsity, TrajCom leverages rich context information to filter a set of reference trajectories that correlate strongly with the target incomplete trajectory. Then, TrajCom learns time-aware encodings of these trajectories by a newly proposed time-aware recurrent unit. Moreover, a popularity-weighted attention mechanism is proposed to complete the missing locations. Extensive experiments on four datasets show that TrajCom outperforms competitive baselines with up to 25% relative improvements.

1 Introduction

Trajectories, defined as timestamped sequences of locations, arise in a wide spectrum of domains and enable data-driven task support therein. To name a few, trajectory data analytics has shown to be critical to applications ranging from smart transportation [18] and urban planning [1], to robotics design [16] and self-driving cars [19]

Despite the flourish of trajectory data, one key but often overlooked bottleneck that limits the applicability of existing trajectory analysis techniques is the *incompleteness* of trajectory data. Instead of being sampled at a regular and high rate, raw trajectory data are often sparse and incomplete. For example, taxi trajectories in ride-sharing services can easily suffer from unstable GPS signals

* equal contribution, + corresponding author

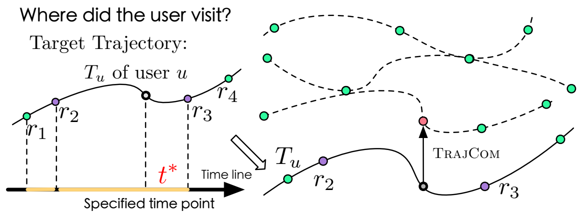


Fig. 1. Given a target trajectory of user u and a specified time point t^*, the trajectory completion problem aims at estimating the location that u is most likely to visit at t^*. To remedy the data sparsity and irregularity, TrajCom is proposed. As shown, the the dotted lines are the reference trajectories. *(Best viewed in color).*

and hardware malfunction, resulting in many missing records in the trajectory database; social network users mostly share their locations at an extremely low frequency, leading to highly sparse trajectories where the time gap varies from hours to days. Trajectory completion, which aims at estimating the "missing" locations at specific time points in a target trajectory, has become a critical step prior to trajectory analysis. Figure 1 shows an example of trajectory completion. Suppose we have a target trajectory T_u of user u where his/her locations between consecutive records are missing. Given a time point t^*, trajectory completion aims at estimating the location of the user at t^* (the red point in the right part).

However, completing missing trajectories is non-trivial due to the following two challenges:

- **Sparsity of trajectories.** Finding proper candidate locations that the user may visit at t^* is key to estimate the missing location accurately. This is challenging when the target trajectory is sparse. Sparse trajectory leads to the large time gaps between consecutive records and decreases the effectiveness of spatio-temporal constrain in inferring missing points. For example, if the time gap between r_2 and r_3 is several hours in Figure 1, the candidate locations will cover a broad range of space, which makes it hard to estimate the correct location.
- **Irregularity of time gaps.** To complete trajectories effectively, we should be able to learn user mobility model in continuous time from user trajectories which have irregular time gaps between consecutive records. It is challenging because the influence between consecutive records are not identical. For example, in the left part of Figure 1, the time gap between r_1 and r_2 and that between r_2 and r_3 is different and thus the influence of r_1 on r_2 is different from that of r_2 on r_3. It is expected that a trajectory completion method is able to capture the difference. In addition, we should also consider the gaps of t^* from the previous point and subsequent point when estimating locations.

Although a number of existing sequential models [2, 21, 31, 28, 4, 15, 17] can be potentially applied to the trajectory completion problem, but they cannot address the aforementioned challenges. These models fall into two categories:

single-sequence models and multi-sequence models. Single-sequence models[2, 21, 17] operate on one single timestamped sequence and estimate missing values by capturing the regularity in that sequence. Example models include linear interpolation and autoregressive models. It seems possible to directly apply single-sequence models on the target trajectory. But these methods are designed for dense trajectories on road network and are not suitable for other types of trajectories. Their performance will be poor when the target trajectory is *sparse and short*. The case is even worse when the time gaps vary. This is because the target trajectory alone does not provide sufficient information for learning reliable single-sequence models. Multi-sequence methods [31, 28, 4, 15, 11, 9, 22] learn sequential models (*e.g.*, hidden Markov model, recurrent neural network) from a collection of sequences. Most of these methods are originally designed to make predictions or recommendations on the target sequence and can be easily adapted to trajectory completion task. However, these methods often assume all the records are sampled uniformly without considering the time gap *irregularity* between consecutive records. Furthermore, they use all the sequences but ignore the fact that many sequences are irrelevant to the target trajectory. Another issue of adapting these methods for trajectory completion is that they use only the preceding sub-trajectory of the target point t^* for prediction, but not the sub-trajectory after t^*. As a result, these models fail to emphasize local patterns around the query point and often yield suboptimal completion accuracy.

Motivated by the lack of effective techniques for completing sparse and irregular trajectories, we propose a new neural trajectory completion method. Our method, named TrajCom, employs a filtering-and-encoding pipeline. In the first step, it features a **context-guided neural filtering** module to select a set of reference trajectories. Instead of relying on the target trajectory alone or feeding in all trajectories indiscriminately, TrajCom filters a set of reference trajectories that highly correlate with the target trajectory. Then, TrajCom performs **context-aware record encoding** of records in both the target trajectory and reference trajectories. To learn contextual transition regularities, we design a *time-aware* recurrent unit, which explicitly models time gap information to learn time-aware transitions for estimating missing locations. To estimate the missing location, a popularity-weighted attention mechanism is proposed. It can not only attend to the most important contexts, but also provide interpretable explanations that rationalize its completions.

We summarize the contributions of this paper as follows:

– We propose a context-guided neural filtering mechanism for trajectory completion. Different from existing works, the neural filtering module selects a set of reference trajectories to alleviate data sparsity.
– We propose a time-aware gated recurrent unit (tGRU) to explicitly model time gap information to handle data irregularity. Moreover, we also design a popularity-weighted attention mechanism for inferring missing locations.
– We conduct extensive experiments on four real-world datasets, which contain both check-in trajectories and human movement trajectories. The exper-

imental results show that TrajCom outperforms state-of-the-art trajectory completion methods with up to 25% improvements.

2 Related Work

We provide an overview of existing studies related to TrajCom. These works can be classified into three orthogonal aspects: trajectory completion, location prediction and POI recommendation, and methods for modeling varied time gap.

Trajectory Completion. Various techniques have been proposed to complete trajectory data. They can be broadly categorized into two groups: single-sequence models and multi-sequence models. Methods in the first group are interpolation/regression-based methods [2, 21, 17] which complete the missing location by capturing the regularity in target trajectory itself. This kind of methods are poorly preformed in our task because the target trajectory alone can lack evidence for estimating reliable location in such sparse and irregular data. The other group of methods is multi-sequence models. Methods in this group can be further divided in two lines. The first line is road-based methods which relies on external road information to complete trajectory. [13] [13] exploit the structural regularity in large GPS trace data to infer the missing locations without information about the underlying road network. [24] [24] propose a method to calibrate trajectory data based on the geometric characters and road map. [10] [10] and [8] [8] present efficient models to map the trace into road maps and complete the missing locations. However, these methods can only complete dense GPS traces in road networks, and are not suitable for sparse trajectories. Another line is clustering based method which completes the trajectory based on the clustering result. To our knowledge, there is only one work falling in this line. [23] [23] first cluster trajectories into several clusters and complete the target trajectory with trajectories in the same cluster. Unfortunately, this work is designed for dense trajectories and the performance will be poor when the target trajectory is sparse.

Location Prediction & POI Recommendation. The methods for location prediction and POI recommendation are not proposed for completing trajectory but can be adapted to this task. These methods can be roughly categorized into two groups: location prediction methods and POI recommendation methods. For the first group, [29] [29] and [30] [30] propose to predict the next location by mining the frequent sequencial patterns of locations. In recent years, [15] [15], [28] [28] and [14] [14] design several RNN structures to capture the sequential influence and predict the future locations of users. Many different methods are proposed for POI recommendation, such as metric embedding[5], collaborative retrieval[32], matrix factorization[6], factorization machine[12], and neural network[27][33]. However, these methods are incapable of capturing the uneven time gaps between consecutive records and all these methods fail to utilize the subsequent information of the target time to complete a location for the target time.

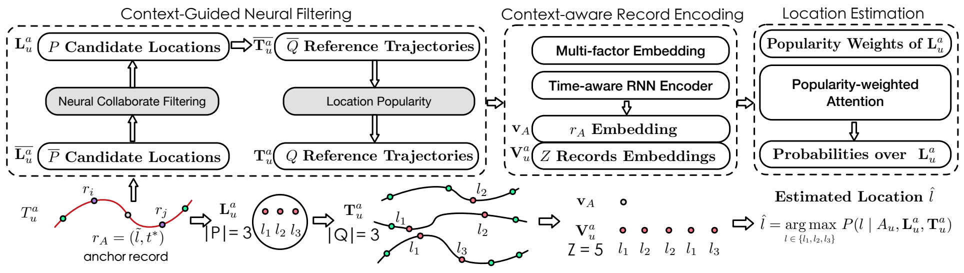


Fig. 2. Illustration of the workflow for our proposed TrajCom.

3 Methodology

In this section, we first formulate the problem of trajectory completion in Section 3.1. Then, three key modules of TrajCom are elaborated in Section 3.2 - 3.4 respectively.

3.1 Problem Formalization

We consider a set of M users $\mathscr{U} = \{u_1, \ldots, u_M\}$ and a set of N locations $\mathscr{L} = \{l_1, \ldots, l_N\}$. Each user u has a chronologically ordered sequence of records $S_u = [r_1, \ldots, r_t]$, wherein each record $r_i = (l_i, t_i)$ is a tuple containing a location $l_i \in \mathscr{L}$ and the time stamp value t_i. Based on a location sequence, we define trajectories as follows:

Definition 1 (**Trajectory**): *Given a sequence of records S_u of user u and a time gap threshold $\delta > 0$, we consider a valid trajectory to be any sub-sequence $T_u = [r_i, r_{i+1}, \ldots, r_{i+k}]$ in S_u which satisfies the following constraints:* ***(1)*** *$\forall j, 1 < j \leq k : t_j - t_{j-1} \leq \delta$; and* ***(2)*** *there exists no longer sub-sequence in S_u which contains T_u and satisfies (1).*

Based on Definition 1, each user record sequence can be partitioned to construct a trajectory set $\mathscr{T}_u = \{T_u^1, T_u^2, \ldots\}$. We combine the trajectory sets from all M users to form a global trajectory set $\mathscr{T}$. The goal of trajectory completion is to estimate the location $\hat{l}$ for a user u based on a target trajectory $T_u^a \in \mathscr{T}_u$ and a specified time point $t^* \in (t_i, t_{i+k})$, where t_i and t_{i+k} are the first and last time stamp of trajectory T_u^a, respectively. Formally, $\hat{l} = \underset{l \in \mathscr{L}}{\arg\max}\, \mathrm{P}(l \mid T_u^a, t^*, \mathscr{T})$.

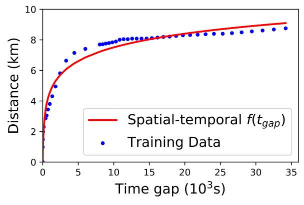
3.2 Context-guided neural filtering

To address the issue of data sparsity when completing T_u^a, we propose a context-guided neural filtering technique which retrieves a pool of candidate locations and a set of reference trajectories from the historical data, that are relevant to the given target trajectory and time point. It contains two steps: (1) filtering candidate locations , and (2) retrieving reference trajectories.

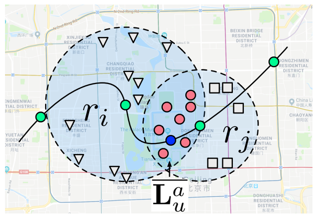
Filtering Candidate Locations. We leverage two types of context information, *i.e.* spatio-temporal constraints and user preference, to filter a pool of candidates from all possible locations for completing.

Spatio-temporal constraint. We note that any user's movement is constrained by the spatio-temporal contexts, which means a user's current location is bounded by the time elapsed since he/she has left the last location. For example, we observe that a user u is currently at location A. Within the next 10 minutes, it would be unlikely for u to move too far away from A. From our data analysis, we find that the relationship between the distance of two consecutive records in a trajectory and its time gap can be modeled by a logarithmic function, if the time gap is not too large (see Figure 3(a)). As such, we model the spatio-temporal constraint as follows: $f(x) = \alpha \cdot \log(\beta \cdot x + 1)$ where the input x is the time gap value, and the function value $f(x)$ is the maximum distance between any pair of records with such a time gap value. The parameters, α and β highly dataset dependent, and thus we learn them on the training data.

To obtain the parameters of $f(\cdot)$, we first calculate the time gap and distance between each consecutive pair of records in the given trajectory dataset. Next, we compute the 95% quantiles of distances for different time gap orders and use them to fit $f(\cdot)$. For example, we plot the quantiles and the spatio-temporal constraint function $f(x)$ using a Foursquare (US) dataset in Figure 3(a).



(a)



(b)

Fig. 3. Illustration of initial candidate locations. (a) Distance Statistics. (b) Initial Candidate Locations.

Given T_u^a and t^*, we first construct an **anchor record** $r_A = (\tilde{l}, t^*)$, where $\tilde{l}$ is a pseudo location. By complimenting T_u^a with r_A, we get the **anchor trajectory** $A_u = [r_1, \ldots, r_i, \ r_A, \ r_j, \ldots, r_K]$.

As shown in Figure 3(b), we first utilize the spatio-temporal constraint function $f(\cdot)$ to obtain the initial candidate locations as follows:

Definition 2 (**Candidate Locations**): *Given the preceding record* $r_i = (l_i, t_i)$, *the succeeding record* $r_j = (l_j, t_j)$, *and a specific spatio-temporal constraint function* $f(\cdot)$, *we consider the initial candidate locations to be a subset of* $\overline{P}$ *locations* $\overline{\mathbf{L}_u^a} = \{l_1, \ldots, l_{\overline{P}}\} \subset \mathscr{L}$, s.t. $\forall\, l_p \in \overline{\mathbf{L}_u^a}$: $\mathrm{dis}(l_i, l_p) < f(t^* - t_i)$ *and* $\mathrm{dis}(l_j, l_p) < f(t_j - t^*)$, *where* $\mathrm{dis}(\cdot,\cdot)$ *is calculated using the Haversine formula.*

User Preference. We note that generating the set of candidate locations based on Definition 2 could potentially result in a vast number of locations, especially when the time gap between t_i and t_j is sufficiently large, and this may become a computational bottleneck. As such, we propose using Neural Collaborative Filtering (**NCF**) [7] for incorporating the user preference in order to obtain a user-specific subset of P candidate locations $\mathbf{L}_u^a$ from $\overline{\mathbf{L}_u^a}$. In other words, we filter away those locations that user u is unlikely to visit. Specifically, we construct a binary user-location matrix $\mathbf{B} \in \mathbb{R}^{M\times N}$, whereby the entry $b_{u,l} \in \{0,1\}$ indicates whether user u has visited location l. NCF decomposes $\mathbf{B}$ into a user latent matrix $\mathbf{E}_u^{\mathbf{NCF}} \in \mathbb{R}^{M\times d_{\mathbf{NCF}}}$ and a location latent matrix $\mathbf{E}_l^{\mathbf{NCF}} \in \mathbb{R}^{N\times d_{\mathbf{NCF}}}$ via the collaborative filtering approach. Specifically, the output of NCF is calculated by a multilayer perceptron(**MLP**) with a logistic function: $\hat{b}_{u,l} = \text{logistic}(\text{MLP}(\mathbf{e}_{\mathrm{l}}^{\mathbf{NCF}}, \mathbf{e}_{\mathrm{u}}^{\mathbf{NCF}}))$ The parameters in NCF can be learned using back-propagation with the cross-entropy loss, and we can easily obtain a probability distribution $\mathbf{P}_{\overline{\mathbf{L}}}$ over the $\overline{P}$ initial candidate locations. The set of user-specific candidate locations $\mathbf{L}_u^a$ is generated by ranking and selecting the top-P locations based on $\mathbf{P}_{\overline{\mathbf{L}}}$.

Retrieving Reference Trajectories. For a given target trajectory and a specified time point, we can obtain the initial set of reference trajectories as follows:

Definition 3 (**Reference Trajectories**): *Reference trajectories is a subset of $\overline{Q}$ trajectories $\overline{\mathbf{T}_u^a} \subset \mathscr{T}$, whereby each trajectory in $\overline{\mathbf{T}_u^a}$ passes through at least one location in $\mathbf{L}_u^a$.*

However, $\overline{\mathbf{T}_u^a}$ can be dominated by the popular locations in $\mathbf{L}_u^a$ which tend to have many trajectories passing through them. Instead of simply taking all trajectories passing through as the reference trajectories, we propose to select at most γ trajectories to form the reference set. By doing so, for all P locations, we obtain a smaller and balanced subset of Q reference trajectories $\mathbf{T}_u^a$.

3.3 Context-aware record encoding

To learn the representation of records within each trajectory, we use a multi-factor embedding layer and a bi-directional RNN layer with time-aware GRU (tGRU), for solving the irregularity.

Multi-factor Embedding Layer. For an arbitrary trajectory $T_u = [r_1, ..., r_K]$ for some user $u \in \mathscr{U}$, we designed a multi-factor embedding layer to capture the information for each record $r_k = (l_k, t_k)$. The user u and location l_k are represented using one-hot encodings. Similarly, the temporal information t_k is represented using a one-hot encoding with 48 possible values. We can obtain the multi-factor embedding for record r_k as follows:

$$\mathbf{e}_k^u = u\cdot\mathbf{E}_u; \quad \mathbf{e}_k^l = l_k\cdot\mathbf{E}_l; \quad \mathbf{e}_k^t = t_k\cdot\mathbf{E}_t;$$
$$\mathbf{e}_k = [\mathbf{e}_k^u;\ \mathbf{e}_k^l;\ \mathbf{e}_k^t]$$

where $[\cdot\,;\cdot]$ is the concatenation operator, and $\mathbf{E}_u \in \mathbb{R}^{M\times d_u}$, $\mathbf{E}_l \in \mathbb{R}^{(N+1)\times d_l}$, $\mathbf{E}_t \in \mathbb{R}^{48\times d_t}$ are the embedding matrices for the user, spatial and temporal ~~information,~~ respectively.

We consider 24 hourly intervals for both weekdays and weekends.

There is an additional entry for $\mathbf{E}_l$, due to the placeholder location $\tilde{l}$ used in the construction of the anchor record r_A.

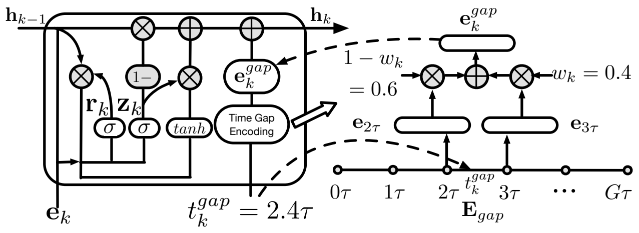


Fig. 4. Architecture of tGRU. E.g. if $t_k^{gap} = 2.4\tau$, the time gap embedding is the linear combination of $\mathbf{e}_{2\tau}$ and $\mathbf{e}_{3\tau}$, with the weight $w_k = 0.4$.

Time-aware GRU (tGRU). Unlike other forms of sequential data such as video or stock, time gap between consecutive records in trajectory data can vary from minutes to hours. Traditional RNNs cannot model this unique characteristic in trajectory data and fail to perform well due to the *varying time gap information* in trajectory completion. In order to modulate the influence of a record towards its preceding or succeeding records within a trajectory w.r.t. the time gap between each consecutive pair of records, we propose a novel time-aware GRU (tGRU).

Time Gap Encoding. Recall that in Definition 1, we specified a constraint such that the temporal difference between any consecutive pair of records in a trajectory is no larger than a predefined threshold δ. As such, to encode time gaps, we segment the time period $[0, \delta]$ into G equal-length periods with duration $\tau = \delta/G$, and obtain $G + 1$ segment points. The time gap value for a record r_k is defined as $t_k^{gap} = t_k - t_{k-1}$, *i.e.* the temporal difference between record r_k and its preceding record. To encode t_k^{gap}, we perform the following steps:

$$
\begin{aligned}
&i_k^p = \lfloor t_k^{gap}/\tau \rfloor; \quad i_k^s = \lceil t_k^{gap}/\tau \rceil; \\
&w_k = t_k^{gap}/\tau - i_k^p; \\
&\mathbf{e}_k^{gap} = (1 - w_k) \cdot \mathbf{i}_k^p \cdot \mathbf{E}_{gap} + w_k \cdot \mathbf{i}_k^s \cdot \mathbf{E}_{gap}
\end{aligned}
$$

where i_k^p and i_k^s are the indexes for its preceding and succeeding segment points in $[0, \delta]$. $\mathbf{i}_k^p$ and $\mathbf{i}_k^s$ are one-hot encodings of i_k^p and i_k^s. $\mathbf{E}_{gap} \in \mathbb{R}^{(G+1)\times d_{gap}}$ is the embedding matrix for the $G + 1$ segment points. An arbitrary time gap t_k^{gap} should lie in between two segment points, and thus we use a weighted sum of the embeddings of its preceding and succeeding segment points to compute the encoding of t_k^{gap}. Figure 4 illustrates the idea of time gap encoding in tGRU.

Operations in tGRU. To effectively fuse the multi-factor embedding $\mathbf{e}_k$ and the time gap encoding $\mathbf{e}_k^{gap}$, we rely on our proposed tGRU. At each time step k, w.r.t. the k-th record r_k in a trajectory, tGRU performs the following operations

The time gap value for the first record in any trajectory is always zero, i.e. $t_1^{gap} = 0$.

to generate the corresponding hidden state h_k:

$$\begin{aligned}
(\mathbf{r}_k,\ \mathbf{z}_k)^\intercal &= \sigma\left(\mathbf{W}_I \cdot \mathbf{e}_k + \mathbf{U}_I \cdot \mathbf{h}_{k-1} + \mathbf{b}\right) && (1)\\
\mathbf{c}_k &= \tanh\left(\mathbf{W}_\mathrm{C} \cdot \mathbf{e}_\mathrm{k} + \mathbf{U}_\mathrm{C} \cdot \mathbf{r}_\mathrm{k} \cdot \mathbf{h}_{\mathrm{k}-1}\right) && (2)\\
\mathbf{h}'_k &= \mathbf{z}_k \cdot \mathbf{c}_k + (1 - \mathbf{z}_k) \cdot \mathbf{h}_{k-1} && (3)\\
\mathbf{h}_k &= \mathbf{h}'_k + \mathbf{e}_k^{gap} && (4)
\end{aligned}$$

where $\mathbf{W}_I$, $\mathbf{U}_I$, $\mathbf{W}_C$, $\mathbf{U}_C$ and $\mathbf{b}$ are the learnable parameters of tGRU, while $\sigma(\cdot)$ and $\tanh(\cdot)$ are the sigmoid and hyperbolic tangent activation functions. The key difference to GRU lies in Equation 4, which modulates the influence of preceding records in a trajectory based on the encoded time gap information. The final learned representation for a record r_k, *i.e.* $\mathbf{v}_k$, can be obtained by applying two RNNs in a bi-directional setting as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{h}_k} &= \overrightarrow{\mathrm{RNN}}(\mathbf{e}_k,\ \overrightarrow{t_k^{gap}});\ \overleftarrow{\mathbf{h}_k} = \overleftarrow{\mathrm{RNN}}(\mathbf{e}_k,\ \overleftarrow{t_k^{gap}}) && (5)\\
\mathbf{v}_k &= [\overrightarrow{\mathbf{h}_k};\ \overleftarrow{\mathbf{h}_k}] && (6)
\end{aligned}$$

where $\overrightarrow{\mathrm{RNN}}$ and $\overleftarrow{\mathrm{RNN}}$ are the RNNs equipped with tGRU in the forward and backward directions; $\overrightarrow{t_k^{gap}}$ and $\overleftarrow{t_k^{gap}}$ are the time gap values in forward and backward sequences. Assuming that the hidden state size of tGRU is h, the learned representation $\mathbf{v}_k$ for a record r_k will be a $2h$-dimensional vector which contains all related information from the reference trajectory.

3.4 Location Estimation and Optimization

Location Estimation To complement the context-guided neural filtering module, we propose a **popularity-weighted attention** network which takes the record embeddings as input and estimates the probability over candidate locations. Inspired by [25], we find that employing *multi-head* attention can enhance the completion accuracy due to its capability to attend to information from different embedding subspaces.

Specifically, after applying the bi-directional RNN to encode all trajectories in $\mathbf{T}_u^a$ and A_u, we can obtain the learned representations of records within them. For estimating $\hat{l}$, we are interested in records whose locations are in the pool of P candidate locations $\mathbf{L}_u^a$. We refer to all such records as reference records, and correspondingly, the collection of Z reference record embeddings as $\mathbf{V}_u^a = [\mathbf{v}_1, ..., \mathbf{v}_Z]$. For each record embedding $\mathbf{v}_z \in \mathbf{V}_u^a$, we denote its corresponding location as l_z. As for the anchor record r_A for which we would like to estimate the location $\hat{l}$, we denote its learned representation as $\mathbf{v}_A$.

When estimating the location $\hat{l}$, we adjust the attention weights for records belonging to each location $l \in \mathbf{L}_u^a$ w.r.t. its popularity. The popularity-based weight for any location l can be derived as follows:

$$g(l) = \begin{cases} 1, & n_l \leq \gamma \\ min\{n_l/\gamma,\ 3\}, & n_l > \gamma \end{cases} \tag{7}$$

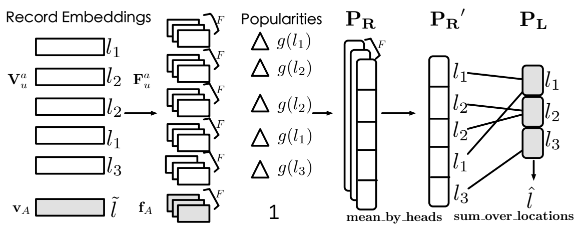


Fig. 5. Popularity-weighted Attention. We reuse the candidate locations shown in Figure 2 to illustrate the operations for this module.

where n_l is the number of trajectories passing through location l, and γ is the maximum number of reference trajectories sampled for each location in $\mathbf{L}_u^a$. These popularity-based location weights are taken into consideration when estimating $\hat{l}$.

We design a popularity-weighted attention with multi-head mechanism to estimate $\hat{l}$. Taking $\mathbf{v}_A \in \mathbb{R}^{1\times 2h}$ as the query and $\mathbf{V}_u^a \in \mathbb{R}^{Z\times 2h}$ as the values, it first performs a multi-head operation. Assuming that we have F different 'heads', the model performs F different linear projections for each record representation. In other words, $\mathbf{v}_A$ gets projected to $\mathbf{f}_A \in \mathbb{R}^{F\times 1\times(2h/F)}$, and similarly, the Z record embeddings $\mathbf{V}_u^a$ now becomes $\mathbf{F}_u^a \in \mathbb{R}^{F\times Z\times(2h/F)}$. To obtain a final probability distribution over the P candidate locations, we first compute the popularity-weighted attention score by $s_z = g(l_z)\cdot(\mathbf{f}_A \otimes \mathbf{F}_u^a[z])$ and perform multi-head attention by the following steps:

$$\mathbf{s} = [s_1, \ldots, s_Z] \tag{8}$$
$$\mathbf{P_R} = \varphi([s_1, \ldots, s_Z]) \tag{9}$$
$$\mathbf{P'_R} = \mathbf{mean\_by\_heads}(\mathbf{P_R}) \tag{10}$$
$$\mathbf{P_L} = \mathbf{sum\_over\_locations}(\mathbf{P'_R}) \tag{11}$$

where $\mathbf{F}_u^a[z]$ is the F heads tensor of the z-th record embedding, $g(l_z)$ is the popularity weight of l_z, $\otimes$ is the batch matrix multiplication operation, $\mathbf{P_R} \in \mathbb{R}^{Z\times F}$ are F probability distributions of heads; $\mathbf{P'_R} \in \mathbb{R}^{Z}$ is the overall probability over the Z records; and $\mathbf{P_L} \in \mathbb{R}^{P}$ is the visiting probability over $\mathbf{L}_u^a$. Figure 5 provides a simple illustration of our proposed popularity-weighted multi-head attention, and our model selects the location with the highest probability in $\mathbf{P_L}$ as the final estimated location $\hat{l}$.

Model Optimization Given a training sample $(T_u^a, t^*) \rightarrow l^*$, the loss function of TrajCom contains two parts: (1) location filtering loss J_s, which encourages l^* to be selected by NCF; (2) location estimation loss J_m, which ensures l^* has the highest probability in $\mathbf{P_L}$. We use the cross-entropy loss to optimize these two parts. Formally, J_s and J_m are defined as follows:

$$J_s = -\log(\mathbf{P}_{\overline{\mathbf{L}}}[l^*]) - \sum_{l \in \overline{\mathbf{L}_u^a} \setminus l^*} \log(1 - \mathbf{P}_{\overline{\mathbf{L}}}[l]) + \frac{\lambda_1}{2}||\Theta_F||^2$$

$$J_m = -\log(\mathbf{P}_{\mathbf{L}}[l^*]) - \sum_{l \in \mathbf{L}_u^a \setminus l^*} \log(1 - \mathbf{P}_{\mathbf{L}}[l]) + \frac{\lambda_2}{2}||\Theta_E||^2$$

$\mathbf{P}_{\overline{\mathbf{L}}}$ is the probability distribution of $\overline{\mathbf{L}_u^a}$ generated by NCF model. $\mathbf{P}_{\mathbf{L}}$ is the probability distribution over $\mathbf{L}_u^a$ from Equation 11; λ_1 and λ_2 are parameters that used for controlling the regularization terms. We use Θ_F and Θ_E to denote all trainable parameters in the NCF and the RNN encoder, respectively.

In order to train all parameters of TrajCom in an end-to-end manner, we propose a parameter learning algorithm. For each training sample, we first check whether the ground truth location l^* is in the top-P locations of the NCF. If l^* is in $\mathbf{L}_u^a$, we only minimize J_m. Otherwise, we minimize J_s and update Θ_F, up to the point whereby either $l^* \in \mathbf{L}_u^a$ has been satisfied or the maximum number of iterations η has been reached (lines 4-8). For optimization, we use two RMSProp optimizers for these two losses. *Back Propagation Through Time*[26] algorithm is employed to learn the parameters in Θ_E.

4 Experiments

Firstly, in Section 4.1, we introduce the datasets used in our experiments. Next, we describe the experimental settings in Section 4.2, which include the data setup, baseline methods, and evaluation metrics. The performance comparison and model ablation studies are detailed in Sections 4.3 and 4.4, respectively.

4.1 Datasets

We evaluate our proposed TrajCom against several baseline methods using four publicly available real-world trajectory datasets, which can be categorized into two groups. The first group consists of three check-in datasets from location-based social networks: **(1)** Foursquare in Tokyo (**TKY**), **(2)** Foursquare in United States (**US**) [3], and **(3) GeoTweets**, a geotagged tweet dataset in Los Angeles [28]. For the second group, we use **Geolife** [34], which is a human movement trajectory dataset in Beijing. Geolife contains human movement-based trajectories with different transportation modes, and we use the top-3 transportation modes, namely *Walking*, *Biking*, and traveling by *Car*. The trajectories in these modes are 4k, 3k and 1.5k, respectively.

Table 1. Statistical information of datasets

Dataset	$\#_{record}$	$\#_{user}$	$\#_{loc}$	$\#_{traj}$
Foursquare (TKY)	573,703	1,083	38,333	10k
Foursquare (US)	3,564,144	50,813	501,900	126k
GeoTweets (LA)	1,188,405	3,401	67,210	13k
Geolife (BJ)	5,431,867	69	40,000	8.5k

4.2 Experiment Settings

Data Setup To generate incomplete trajectories for our experiments, we randomly removed records from the original trajectories. Specifically, for each trajectory with length L, we randomly remove $\lfloor L/2 \rfloor$ records, and treat the timestamp in these records as the target time point for location estimation. Consequently, we have $\lfloor L/2 \rfloor$ data samples for each input trajectory. In each dataset, we randomly choose 70% of the trajectories for each user to construct the training set, and utilize the remaining 5% and 25% for validation and testing, respectively.

Compared Methods As TrajCom does not rely on any external information(such as road network), we restrict our comparison to baseline methods which are not dependent on such information. Under this setting, we compare TrajCom with a total of seven baseline methods, which can be grouped into four categories:1 *Naïve Methods.* **Nearest Locations** (**NL**) chooses the nearest neighbors based on the midpoint of the users previous locations. **Most Frequent Location** (**MFL**) selects the most frequently visited locations for user u. (2)**Cluster** [23] uses K-means on locations and performs context mapping between cluster centroids and trajectories. (3)*Location Prediction Methods.* **ST-RNN** [15] is an RNN-based method to predict the next location. **SERM** [28] utilizes embedding techniques to jointly model the influence of different factors. (4)*POI Recommendation Methods:* **BPR** [20] is a MF-based recommendation method for implicit feedback datasets. **RankGeoFM** [12] is a method for POI recommendation.

Evaluation Metrics We choose two commonly-used metrics to evaluate the performance. The first one is Hit Ratio @k (HR@k), which examines whether the predicted location l^* appears in the top-k estimated locations. The second metric is distance error ϵ, based on the minimum geographical distance between l^* and the top-5 estimated locations calculated using the Haversine formula.

Table 2. Hyperparameter values in context-guided neural filtering module.

Dataset	α	β	γ	P	Acc. of NCF
Foursquare (TKY)	1.97	5.68	12	200	0.673
Foursquare (US)	1.39	33.09	10	200	0.684
GeoTweets (LA)	1.34	22.45	60	50	0.932
	Distance (km)		γ	P	Acc. of NCF
Geolife(BJ)-Walk	1.814		25	—	0.907
Geolife(BJ)-Bike	2.489		23	—	0.932
Geolife(BJ)-Car	4.227		51	—	0.948

Hyperparameter Settings For the hyperparameters in context-guided neural filtering, we fit α and β based on the approach described in Section 3.2. The settings are shown in Table 2. In the encoding module, we use the same number of factors for all the embedding matrices, i.e. $d_u = d_l = d_t = d_{\mathbf{NCF}} = 50$. For the time gap information, we partition the time gap $[0, \delta]$ into $G = 20$ segments, and we use $d_{gap} = 128$ to encode each segment point. We set $h = 128$ for tGRU, and the number of 'heads' in the attention network is set as $F = 8$. Lastly, we set the maximum update threshold as $\eta = 20$, and both the learning rates lr_s and lr_m as 0.01.

4.3 Performance Comparison

Tables 3 and 4 show the performance comparison of TrajCom against the baseline methods. TrajCom outperforms the baselines significantly on most of the datasets. Next, we explain the results and analyze the reasons on the two groups of datasets *i.e.* check-in and human movement, separately.

Table 3. Performance comparison for different methods on check-in datasets (Best result is indicated in bold).

	Foursquare (TKY)			Foursquare (US)			GeoTweets (LA)			
Method	HR@1	HR@10	HR@20	HR@1	HR@10	HR@20	HR@1	HR@10	HR@20	ϵ (km)
NL	0.0094	0.0242	0.0692	0.0097	0.0180	0.0637	0.3270	0.4368	0.4539	4.132
MFL	0.0231	0.0732	0.1421	0.0220	0.0736	0.1401	0.3480	0.4582	0.4772	3.721
Cluster	0.0161	0.0330	0.0760	0.0140	0.0308	0.0723	0.3482	0.4690	0.4921	3.980
ST-RNN	0.0676	0.1400	0.2668	0.0641	0.1358	0.2528	0.4328	0.6096	0.6502	2.498
SERM	0.0716	0.1653	0.2962	0.0688	0.1593	0.2873	0.4591	0.6267	0.6930	2.292
BPR	0.0537	0.1280	0.2657	0.0542	0.1234	0.2607	0.3719	0.5117	0.6893	2.599
RankGeoFM	0.0680	0.1409	0.2730	0.0618	0.1354	0.2702	0.3950	0.5506	0.7122	2.512
TrajCom	**0.0984**	**0.2231**	**0.3219**	**0.0880**	**0.2131**	**0.3169**	**0.5670**	**0.7161**	**0.8614**	**2.130**

Table 4. Performance comparison for different methods on Geolife dataset (Best result is indicated in bold).

	Geolife(BJ)-Walk				Geolife(BJ)-Bike				Geolife(BJ)-Car			
Method	HR@1	HR@10	HR@20	ϵ (km)	HR@1	HR@10	HR@20	ϵ (km)	HR@1	HR@10	HR@20	ϵ (km)
NL	0.7054	0.8541	0.8894	0.687	0.6297	0.7802	0.8597	0.708	0.4231	0.6211	0.7200	1.364
MFL	0.1910	0.2539	0.3321	2.936	0.2020	0.2736	0.3012	2.960	0.1839	0.2317	0.2875	2.985
Cluster	**0.7376**	**0.8600**	**0.8981**	**0.645**	0.6941	0.8308	0.8582	0.671	0.6236	0.7731	0.8129	0.937
ST-RNN	0.6176	0.7300	0.8268	1.780	0.6423	0.7382	0.8518	1.703	0.5928	0.6423	0.7129	1.916
SERM	0.5716	0.6653	0.7962	1.845	0.5608	0.6529	0.7896	1.803	0.5133	0.5942	0.6537	2.035
BPR	0.2571	0.3879	0.5630	2.763	0.2542	0.3960	0.5873	2.937	0.2197	0.3518	0.5140	2.649
RankGeoFM	0.2761	0.4279	0.5930	2.459	0.3189	0.4747	0.6239	2.402	0.2732	0.4180	0.6074	2.304
TrajCom	0.6372	0.8169	0.8543	0.680	**0.7140**	**0.8553**	**0.8961**	**0.648**	**0.6705**	**0.7931**	**0.8511**	**0.893**

Check-in Datasets. As shown in Table 3, TrajCom significantly outperforms all baseline methods on check-in datasets, i.e. Foursquare (TKY), Foursquare (US), and GeoTweets. For example, in the Foursquare (US) dataset, TrajCom achieves over 25% improvement on HR@1 compared with the best baseline method. We note that Naïve methods (**NL** and **MFL**) perform worst in general, as user check-ins are extremely sparse and the nearest first principle cannot capture the complex surrounding information. **Cluster** performs slightly better than **NL**, but still fails to generate meaningful results for completion due to the sparsity of check-in data. Location prediction methods (**ST-RNN** and **SERM**) outperform POI recommendation methods (**BPR** and **RankGeoFM**) as they consider the sequential information. However, location prediction methods do not consider the subsequent records within the trajectory, and pales by comparison to TrajCom. Additionally, TrajCom outperforms existing RNN-based methods (**ST-RNN** and **SERM**) due to its ability to model the crucial time gap information using tGRU.

For TrajCom and all baseline methods, the performance is better on GeoTweets compared to the other two Foursquare datasets. This can be attributed to the fact that the locations in Foursquare are POIs, which are rather sparse and unevenly distributed within the geographical space, in contrast with the grid-based approach used in GeoTweets.

Geolife Dataset. We show the comparison results for the Geolife dataset in Table 4. As an example, we consider the transportation mode of traveling by car, *i.e. Geolife-Car.* Unsurprisingly, the strongest baseline for this particular dataset is the clustering-based method **Cluster**, as Geolife trajectories are rather dense and the human movement are constrained by the road network. This can be captured via the clustering of historical trajectories.

Notably, TrajCom outperforms all baselines, due to its ability to jointly model multiple influence factors.

In terms of different transportation modes, we find that TrajCom fails to outperform the naïve Nearest Locations (**NL**) method for Geolife-Walk. As we subsample trajectories in Geolife for this task, the time gap between each pair of records may be insufficient for most users to move to a different cell with a reasonable walking speed. Due to such an unintended constraint, **NL** performs surprisingly well for this transportation mode.

4.4 Ablation Studies

We study the impact of our proposed tGRU and the multi-head attention mechanism. TrajCom-vanilla uses a vanilla attention network and the standard GRU. TrajCom-pwAtten improves TrajCom-vanilla with a popularity-weighted attention network, and TrajCom-tGRU improves TrajCom-vanilla by replacing standard GRU with tGRU. Due to the space limit, we only present the results of Foursquare (US) in Table 5.

Table 5. Results of ablations on Foursquare dataset.

Method	HR@1	HR@10	HR@20
TrajCom-vanilla	0.0710	0.1601	0.2321
TrajCom-pwAtten	0.0751	0.1621	0.2539
TrajCom-tGRU	0.0813	0.1869	0.2980
TrajCom	0.0880	0.2131	0.3169

We observe that both tGRU and popularity-weighted attention contribute significantly to the overall model performance. By including tGRU which models the vital time gap information for trajectory completion, TrajCom-tGRU improves HR@1 of TrajCom-vanilla form 0.0710 to 0.0813. By adopting the popularity-weighted multi-head attention, TrajCom improves HR@1 of TrajCom-tGRU form 0.0813 to 0.0870, as it is capable of attending to information from different embedding subspaces. By integrating both components, TrajCom further boosts the performance by 24% in terms of HR@1 compared to TrajCom-vanilla. Similar results can be observed in the other datasets.

5 Conclusions

We proposed a neural filtering-and-encoding method TrajCom for completing sparse and irregular trajectories. The novelty of TrajCom lies in two aspects: (1) A context-guided neural filtering module that identifies a set of reference trajectories which are strongly correlated with the target trajectory.

(2) a time-aware neural encoding module which learns recurrent representations for records in reference trajectories and the target trajectory for inferring the missing location. Our experiments on four real-world datasets have demonstrated the efficacy of TrajCom compared with state-of-the-art baselines.

References

1. Bao, J., He, T., Ruan, S., Li, Y., Zheng, Y.: Planning bike lanes based on sharing-bikes' trajectories. In: SIGKDD'2017. pp. 1377–1386 (2017)
2. Chandler, J., Obermaier, H., Joy, K.I.: Interpolation-based pathline tracing in particle-based flow visualization. IEEE Trans. Vis. Comput. Graph. 21(1), 68–80 (2015)
3. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: SIGKDD'2011. pp. 1082–1090 (2011)
4. Feng, J., Li, Y., Zhang, C., Sun, F., Meng, F., Guo, A., Jin, D.: Deepmove: Predicting human mobility with attentional recurrent networks. In: WWW'2018. pp. 1459–1468 (2018)
5. Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y.M., Yuan, Q.: Personalized ranking metric embedding for next new POI recommendation. In: IJCAI'2015. pp. 2069–2075 (2015)
6. Han, P., Shang, S., Sun, A., Zhao, P., Zheng, K., Kalnis, P.: AUC-MF: point of interest recommendation with AUC maximization. In: ICDE'2019. pp. 1558–1561 (2019)
7. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: WWW'2017. pp. 173–182 (2017)
8. Hu, G., Shao, J., Liu, F., Wang, Y., Shen, H.T.: If-matching: Towards accurate map-matching with information fusion. In: ICDE'2017. pp. 9–10 (2017)
9. Kong, D., Wu, F.: HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction. In: IJCAI'2018. pp. 2341–2347 (2018)
10. Li, M., Ahmed, A., Smola, A.J.: Inferring movement trajectories from GPS snippets. In: WSDM'2015. pp. 325–334 (2015)
11. Li, R., Shen, Y., Zhu, Y.: Next point-of-interest recommendation with temporal and multi-level context attention. In: ICDM'2018. pp. 1110–1115 (2018)
12. Li, X., Cong, G., Li, X., Pham, T.N., Krishnaswamy, S.: Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In: SIGIR' 2015. pp. 433–442 (2015)
13. Li, Y., Li, Y., Gunopulos, D., Guibas, L.J.: Knowledge-based trajectory completion from sparse GPS samples. In: SIGSPATIAL'2016. pp. 33:1–33:10 (2016)
14. Liao, D., Liu, W., Zhong, Y., Li, J., Wang, G.: Predicting activity and location with multi-task context aware recurrent neural network. In: IJCAI2018. pp. 3435–3441 (2018)
15. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: A recurrent model with spatial and temporal contexts. In: AAAI'2016. pp. 194–200 (2016)

16. Liu, Y., Cong, M., Dong, H., Liu, D.: Reinforcement learning and ega-based trajectory planning for dual robots. I. J. Robotics and Automation 33(4) (2018)
17. Long, J.A.: Kinematic interpolation of movement data. International Journal of Geographical Information Science 30(5), 854–868 (2016)
18. Meng, C., Yi, X., Su, L., Gao, J., Zheng, Y.: City-wide traffic volume inference with loop detector data and taxi trajectories. In: SIGSPATIAL'2017. pp. 1:1–1:10 (2017)
19. Pek, C., Althoff, M.: Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization. In: ITSC'2018. pp. 1447–1454 (2018)
20. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: bayesian personalized ranking from implicit feedback. In: UAI'2009. pp. 452–461 (2009)
21. Serrano, M.E., Godoy, S.A., Montoya, L.Q., Scaglia, G.J.E.: Interpolation based controller for trajectory tracking in mobile robots. Journal of Intelligent and Robotic Systems 86(3-4), 569–581 (2017)
22. Shen, T., Zhou, T., Long, G., Jiang, J., Wang, S., Zhang, C.: Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. In: IJCAI'2018, Sweden. pp. 4345–4352 (2018)
23. Silva, F.A., Celes, C., Boukerche, A., Ruiz, L.B., Loureiro, A.A.F.: Filling the gaps of vehicular mobility traces. In: MSWiM'15. pp. 47–54 (2015)
24. Su, H., Zheng, K., Huang, J., Wang, H., Zhou, X.: Calibrating trajectory data for spatio-temporal similarity analysis. VLDB'2015 24(1), 93–116 (2015)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS'2017. pp. 6000–6010 (2017)
26. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10), 1550–1560 (1990)
27. Yang, C., Bai, L., Zhang, C., Yuan, Q., Han, J.: Bridging collaborative filtering and semi-supervised learning: A neural approach for POI recommendation. In: SIGKDD'2017. pp. 1245–1254 (2017)
28. Yao, D., Zhang, C., Huang, J., Bi, J.: SERM: A recurrent model for next location prediction in semantic trajectories. In: CIKM'2017. pp. 2411–2414 (2017)
29. Yavas, G., Katsaros, D., Ulusoy, Ö., Manolopoulos, Y.: A data mining approach for location prediction in mobile environments. Data Knowl. Eng. 54(2), 121–146 (2005)
30. Ying, J.J., Lee, W., Weng, T., Tseng, V.S.: Semantic trajectory mining for location prediction. In: ACM-GIS'2011. pp. 34–43 (2011)
31. Zhang, C., Zhang, K., Yuan, Q., Zhang, L., Hanratty, T., Han, J.: Gmove: Group-level mobility modeling using geo-tagged social media. In: SIGKDD'2016. pp. 1305–1314 (2016)
32. Zhang, W., Wang, J.: Location and time aware social collaborative retrieval for new successive point-of-interest recommendation. In: CIKM'2015. pp. 1221–1230 (2015)
33. Zhao, P., Zhu, H., Liu, Y., Xu, J., Li, Z., Zhuang, F., Sheng, V.S., Zhou, X.: Where to go next: A spatio-temporal gated network for next POI recommendation. In: AAAI'2019. pp. 5877–5884 (2019)
34. Zheng, Y., Wang, L., Zhang, R., Xie, X., Ma, W.: Geolife: Managing and understanding your past life over maps. In: MDM'2008. pp. 211–212 (2008)