How Do Your Neighbors Disclose Your Information: Social-Aware Time Series Imputation

Zongtao Liu
Zhejiang University
tomstream@zju.edu.cn

Yang Yang*
Zhejiang University
yangya@zju.edu.cn

Wei Huang
Zhejiang University
hw313@zju.edu.cn

Zhongyi Tang
State Grid Zhejiang Jinyun Power Supply Co. Ltd.
tangzhongyi0123@163.com

Ning Li
State Grid Zhejiang Jinyun Power Supply Co. Ltd.
shiepning@163.com

Fei Wu
Zhejiang University
wufei@cs.zju.edu.cn

ABSTRACT

Different time series is measured in almost all fields including biology, economics and sociology. A common challenge for using such data is the imputation of the missing values with reasonable ones. Most of existing approaches to data imputation assume that individual's observations are independent to each other, which is rarely the case in real-world. In this paper, we study the social-aware time series imputation problem. Given a social network that represents social relations between individuals, we propose a sequential encoder-decoder-based framework and build a connection between the missing observations and the social context. In particular, the proposed model employs the attention mechanism to incorporate social context and temporal context into the imputation task. Experimental results based on two real-world datasets demonstrate that our approach outperforms 11 different baseline methods.

CCS CONCEPTS

• **Mathematics of computing** → **Time series analysis**; • **Information systems** → **Social networks**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

time series, missing data imputation, social networks

ACM Reference Format:
Zongtao Liu, Yang Yang*, Wei Huang, Zhongyi Tang, Ning Li, and Fei Wu. 2019. How Do Your Neighbors Disclose Your Information: Social-Aware Time Series Imputation. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313714

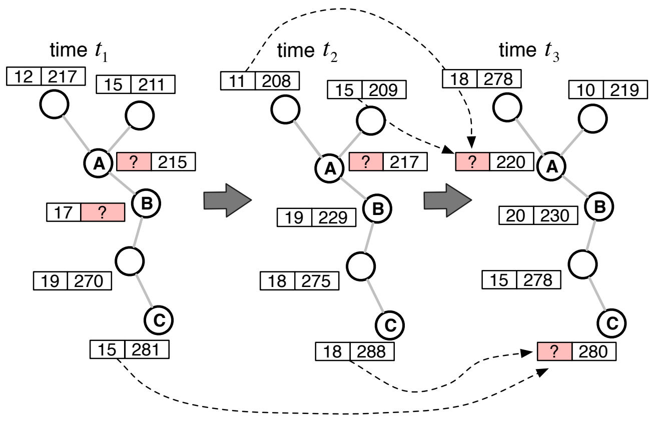


Figure 1: An illustration of the social-aware time series imputation problem. We are given a network where each vertex denotes an individual, and we create an edge between two individuals if the distance between their living places is less than a threshold. Each person has a multivariate observation at every timestamp (two-dimensional in our case), with some missing values (colored in red and marked as "?"), to indicate her electricity consumption. The goal is to impute the missing values by building surrounding and temporal influences.

1 INTRODUCTION

Time-series modeling has been extensively studied and applied in a broad range of applications, such as financial marketing [51], bioinformation [19, 22], and wearable sensors [33]. Among these studies, one fundamental issue is that time series data often inevitably carries missing observations due to various reasons, such as anomaly events, security issues, communication errors, privacy, and so on. Missing observations cause serious damages to the analysis and model results based on them, and thus it is necessary to impute these values.

On the other hand, the social network, a natural way of representing personal relationships, has turned out to be valuable and has recently attracted considerable research efforts [14, 30]. Time series data also plays an important role in social network analysis, as individuals in a social network generate a huge amount of data, which evolves over time. Meanwhile, homophily, a widely recognized organizing principle of social networks, suggests that people with common behavior patterns or similar interests are more likely

*Corresponding author: Yang Yang, yangya@zju.edu.cn.

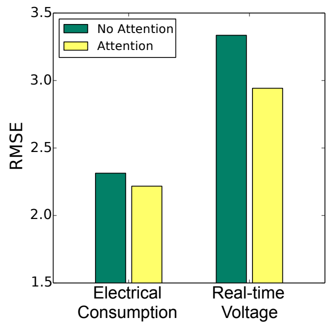


Figure 2: The performance of the proposed method for time series imputation. Two methods are shown here: one considers social and temporal information in the context of the attention mechanism, and another ignores this information. Root Mean Square Error (RMSE) is considered in the figure (the smaller the better).

to connect with each other. Therefore, individuals' missing observations can be guided by their neighbors' data, which is the basis of our work.

To address the problem of time series imputation, various approaches have been explored in the past few decades. For example, one practical method is to replace the missing values with the mean value of observations at adjacent timestamps. Similar methods include smoothing [38], interpolation [1, 45], and spline [36], which are simple and practical when data evolves smoothly. However, these methods can not capture the complex patterns of a time series; therefore, they fail to perform imputation. Recently, deep neural networks-based methods have been employed for data imputation [13, 15, 50]. These methods benefit from their strong prediction ability as well as their ability to capture temporal dependencies. However, to the best of our knowledge, there is a lack of research on the incorporation of *social information* and *deep models* to better perform time series imputation, which we believe is a new promising direction.

In this paper, we study the problem of social-aware time series imputation, which is nontrivial and gives rise to several challenges. Figure 1 demonstrates an example. In particular, given a social network, in which each vertex indicates an individual, there is an undirected edge between two individuals if they live close together (i.e., the distance between their living places is less than a certain predefined threshold). We monitor the electricity consumption of each individual, which is represented by two-dimensional vector, composed of daily power and voltage. At time t_3, individual A's daily power value is missing. To complete the missing value, traditional methods consider A's previous power value at the timestamps t_1 and t_2, which, however, are also missing. Fortunately, observations of A's neighbors provide us with a clue for the missing value. For instance, it is natural to assume that people live close share similar electricity consumption patterns to some extent. We call this "surrounding influence". Intuitively, each of individual v's neighbors has different surrounding influence on v's observations. The first challenge in this study is therefore how to identity the surrounding influence and model the connection between the missing observations and social context of an individual.

Returning to our example, the missing observations of individual C at time t_3 are affected by C's observations at t_1 and t_2. We call this "temporal influence", which has different effects for different timestamps. For instance, if some of C's family leave for a trip during t_2, and return at t_3, the electricity usage of C's family at t_1 shall be closer to that at t_3 and has a larger influence than that at t_2. How to quantify and differentiate temporal influence at different timestamps is the second challenge in this work.

Another factor that influences data imputation in our example is the time intervals between t_1, t_2, and t_3. Most existing time-series imputation methods assume that the elapsed time between the elements of a time series is equal, which may not hold true in practice. How to handle irregular time intervals is our third challenge.

To address the above mentioned challenges, we propose an novel model based on a deep encoder-decoder architecture. It utilizes the attention mechanism to incorporate surrounding influence and temporal influence in the time series imputation task. As Figure 2 shows, the experimental results show that with the help of surrounding influence and temporal influence, we can obtain a clear improvement in imputation performance. Furthermore, we apply Time-Aware Long Short-Term Memory (LSTM), which acts as the basic component of our model for handling the time irregularity issue. See details in Section 3.

We further test the proposed model on two real-world datasets. We compare 11 different state-of-the-art baseline methods and find that our model consistently outperforms others. More detailed results can be found in Section 4.

Accordingly, our contributions are as follows:

- We propose a novel time series imputation framework that takes both surrounding influence and temporal influence into consideration.
- We apply a variant LSTM as the basic component of our model for handling irregular time intervals.
- We construct sufficient experiments under various missing conditions based on two real-world datasets. Experimental results exhibit our method's advantage over eleven baseline methods.

Organization. The remainder of this paper is organized as follows. We first formulate the problem and give the necessary definitions. We then introduce the proposed model, including the model structure, the learning algorithm, and its applications. Subsequently, we present the experimental results. Finally, we review some relevant research and conclude the work.

2 PROBLEM DEFINITION

Let $G =< V, E >$ be a social network, where V is the set of users, and each edge $e_{ij} \in E$ indicates that $v_i \in V$ has a social relationship with v_j. For each user v, her behavior data is measured by a sensor periodically, denoted as $X = \{x_1, x_2, ..., x_T\} \in \mathbb{R}^{T \times D}$. For each timestamp $t \in \{1, 2, ..., T\}$, x_t indicates the t-th observations with D channels (or variables), and x_t^d represents the t-th observation in the

d-th channel. In the real world, many observations are missing or erroneous. To mark the missing data in the time series, we introduce an index m_t for missing ones at timestamp t, where $m_t = 0$ if x_t is not observed, otherwise $m_t = 1$. We also maintain an observed sequence of X as $X' = \{x_{s_1}, x_{s_2}, ..., x_{s_L}\}$, where s_l represents the timestamp of l-th observations, and the corresponding interval sequence Δ, where

$$\delta_l = \begin{cases} 1, & l = 1 \\ s_l - s_{l-1}, & l \neq 1 \end{cases} \tag{1}$$

In this work, we are interested in the missing data imputation task of time series data from users in social networks, where we impute the missing data given the social network G, time series dataset $\mathcal{D} = \{X_n, M_n\}_{n=1}^{N}$, where $X_n = \{x_1^{(n)}, ..., x_T^{(n)}\}$, $M_n = \{m_1^{(n)}, ..., m_T^{(n)}\}$.

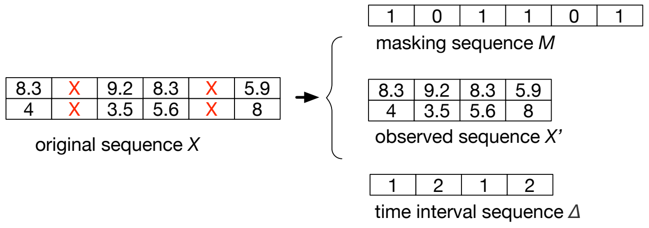


Figure 3: An example of the whole sequence X, masking M, observed sequence X', and time intervals Δ.

3 OUR APPROACH

Recently, recurrent sequences generated by data through an attention mechanism have shown very good performance in multiple tasks, such as neural translation [10, 28], speech recognition [6], and image caption generation [27]. To some extent, our task can also be viewed as a recurrent sequence generation task that generates the complete sequence X^* using the original sequences X from a user v and her social network G. We design an attention mechanism considering two types of information, i.e., surrounding influence and temporal influence, to learn a sequence model for the imputation task. To handle the time irregularity due to missing data, we use a variant LSTM cell considering different time intervals.

In this section, we describe the details for our proposed model **S**ocial-Aware **T**ime Series **I**mputation (STI). Figure 4 shows the architecture of our proposed model.

3.1 Social Attention Network

In order to capture the surrounding influence patterns for imputation, we leverage the attention mechanism to align the useful information from users with similar behavioral patterns. We also shows our solution to handle the time irregularity in the sequences with missing values. This part of our proposed model is called Social Attention Network.

LSTM and variant LSTM. To extract recurrent series patterns from the original sequences with non-uniformed time intervals, we first review a variant LSTM cell. The LSTM [21] recently shows its good properties in modeling series data with recursive relations. For instance, it can avoid the vanishing gradient problem by introducing the gate mechanism. Given the input x_t, the output of hidden state h_t is computed as below:

$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g) \tag{2}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_f) \tag{5}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t \tag{6}$$
$$h_t = o_t \cdot tanh(c_t) \tag{7}$$

where $h_t, c_t \in \mathbb{R}^H$, H is the cell size, $\sigma(\cdot)$ is the logistic sigmoid function, and i, f, o, and g represent *input gate,forget gate*, *output gate* and *cell state* respectively. The three gates control the data flow through the cell. In particular, the input gate i regulates the extent to which a new value data is fed into the cell, the forget gate f regulates the extent to which the history is forgotten, and the output gate o decides the extent to which the value is used to compute the output activation.

By introducing the gate structure, the LSTM unit is capable of handling long-term dependency, but the structure has implicitly assumed that the elapsed time between the elements of a sequence is uniformly distributed. Hence, the time irregularity in a sequence is not considered as being part of its structure, while it often occurs in the sequences with missing elements, that is, the interval sequence Δ is non-uniform.

To capture the patterns with time irregularity using LSTM, many studies have proposed modifications to original LSTM to incorporate the elapsed time into the standard LSTM. One method involves imputing data to maintain the regular time gaps. Based on this idea, some studies impute data by sampling from the observed elements, and many studies have also explored imputing data that is learned in the optimization process. However, the imputation method can have a serious influence on performance as the imputed data might not reflect reality well. Another way of handling the time irregularity is to modify the structure of LSTM to utilize the time gaps. Pham et al. [34] multiplies the output of the forget gate by $g(\delta)$, i.e., $f_t = g(\delta_t) \cdot f_t$. Another way of considering the time irregularity is to adjust the memory cell by applying the short-term memory discount. The mathematic expressions of such an adjusted memory are as follows:

$$c_t^s = tanh(W_d c_{t-1} + b_d) \tag{8}$$
$$\hat{c}_t^s = c_{t-1}^s \cdot g(\delta) \tag{9}$$
$$c_{t-1}^l = c_{t-1} - c_{t-1}^s \tag{10}$$
$$c_{t-1}^* = c_{t-1}^l + \hat{c}_t^s \tag{11}$$
$$\tilde{c} = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{12}$$
$$c_t = f_t \cdot c_{t-1}^* + i_t \cdot \tilde{c} \tag{13}$$

Firstly, the short-term memory component (c_t^s) is computed by a network. This decomposition process is data-driven, and its parameters are updated at the mean time with the rest of the network parameters. Secondly, the short-term memory is discounted by a weight factor $g(\delta)$ to obtain the discounted short-term memory ($\hat{c}_t^s$). Here,

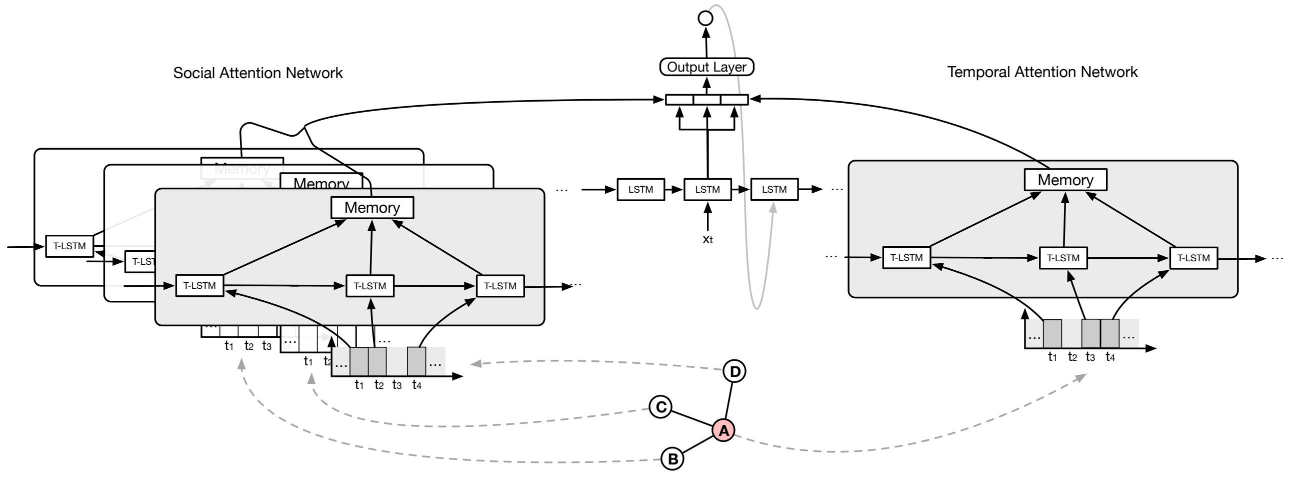


Figure 4: An illustration of our model, Social-Aware Time Series Imputation (STI). Our model mainly consists of two components: Social Attention Network (left) and Temporal Attention Network (right). In the social attention network, we first compute the fix sized memory representations of neighbors' temporal data, and then use the concatenation with the current hidden states of the user to generate the social context vector. In the temporal attention network, we directly feed users' memory representation with the current hidden states to generate the temporal context vector. The concatenation of current hidden states, social context vector and temporal context vector is used to predict the target missing elements.

$g(\cdot)$ is a heuristic decaying function which is monotonically non-increasing. Different types of $g(\cdot)$ can be chosen according to specific application domains. For instance, $g(\delta) = 1/\delta$ is appropriate for datasets with small elapsed times, and $g(\delta_t) = 1/log(e+\delta)$ is preferable for datasets with large elapsed times [34]. We tried several functions $g(\cdot)$ but did not observe a drastic difference in the experimental results; however, $g(\delta) = 1/\delta$ performed slightly better that other functions. Thus, we choose the decaying function $g(\delta) = 1/\delta$ in this work. Finally, to obtain the adjusted memory, the complementary subspace of the long-term memory ($c^l_{t-1} = c_{t-1} - c^S_{t-1}$) is combined with the discounted short-term memory. This variant is proposed in [7], named **T**ime-Aware LSTM (T-LSTM).

Social Context Representation. Attention-based RNNs have been successfully applied in many tasks. These models iteratively feed their inputs by selecting relevant contents at each step. We introduce our extensions to this mechanism to extract useful patterns from social context.

For each user v, we have a group of sequences $\{\hat{X}_{(1)}, \hat{X}_{(2)}, ..., \hat{X}_{(P)}\}$ collected from her neighbors $\hat{v} = \{\hat{v}_{(1)}, \hat{v}_{(2)}, ..., \hat{v}_{(P)}\}$. We extract their observed sequences $\hat{M} = \{\hat{M}_{(1)}, \hat{M}_{(2)}, ..., \hat{M}_{(P)}\}$ and time interval sequences $\hat{\Delta} = \{\hat{\Delta}_{(1)}, \hat{\Delta}_{(2)}, ..., \hat{\Delta}_{(P)}\}$. For each neighbor $\hat{v}_{(p)}$, a T-LSTM network described in Section 3.1 is used to encode its original series data, which takes observed sequence $\hat{X}'_{(p)}$ and interval sequence $\hat{\Delta}_{(p)}$ as input and produces a sequence of hidden states $\hat{h}_{(p)} = \{\hat{h}_{1(p)}, ..., \hat{h}_{s(p)}\}$ by the means of following iterative process:

$$h_{s(p)}, c_{s(p)} = T - LSTM(x'_{s(p)}, \delta_{s(p)}, h_{s-1(p)}, c_{s-1(p)}) \tag{14}$$

It is important to note that the network parameters of T-LSTM are shared for all neighbors.

The decoder is a standard LSTM network that reconstructs the complete sequence X^*. Given the recurrent states h^*_s, we can calculate the social context vector $\hat{a}$. The context vector, also known as attention vector, is a weighted average of the source states. In the studies, Bahdanau et al. [5] and Luong et al. [28] proposed context-based models that calculate the context vector by comparing encoder and decoder at each step. Such computations are expensive. To leverage the attention mechanism without sacrificing too much efficiency, a memory-based attention model was explored in [10]. In the current work, we choose the memory-based attention method to efficiently calculate the social context vector.

In the memory-based attention method, a fixed-size memory representation is updated during the encoding. The memory representation is matrix $C \in \mathbb{R}^{K \times H}$, where K is the number of temporal context vectors and a hyper-parameter in this model, and H is the size of the cell states. C is a linear combination of the encoder states, weighted by a score vector $\alpha^s \in \mathbb{R}^K$:

$$C_k = \sum_{s=0}^{|S|} \alpha_{sk} h_s \tag{15}$$

$$\alpha_{sk} = softmax(W_\alpha h_s \cdot l_s) \tag{16}$$

where $W_\alpha \in \mathbb{R}^K$ is a parameter matrix, and l_s is a vector of position-encodings that prevents the learned context from being symmetric. In particular, the position-encodings are used to force the first few context vectors $C_1, C_2...$ to focus on the start of the sequence and the last few vectors $...C_{k-1}, C_k$ to focus on the end of the sequence. Similar to network parameters of T-LSTM, W_α is shared for all neighbors. We adapt the formula proposed in [10] to obtain l_s:

$$L_{ks} = (1 - \frac{k}{K})(1 - \frac{s}{S}) + \frac{k}{K}\frac{s}{S} \tag{17}$$

where, $k \in \{1, 2, ..., K\}$ is the context vector index, and S is the maximum sequence length across all source sequences. Hence, given the social context, we can compute their memory representations $\hat{C} = \{\hat{C_{(1)}}, \hat{C}_{(2)}, ..., \hat{C}_{(P)}\}$. We define the concatenation of these matrix as $\tilde{C}$, sized $KP \times H$.

In the decoding step, we set the social context vector $\hat{a}$, in each step, by the means of linear combination of the rows in $\tilde{C}$ weighted by a score vector $\hat{\beta}$.

$$\hat{a} = \sum_{i=0}^{K} \hat{\beta}_i \tilde{C}_i \tag{18}$$

$$\hat{\beta} = softmax(W_{\hat{\beta}} h^*) \tag{19}$$

where h^* represents the current hidden states of the decoder, and $W_{\hat{\beta}}$ is the parameter matrix. In each step of the decoding, the current state h_t^* is computed by feeding the previous state h_{t-1}^*, the previous cell state c_{t-1}^* , and previous output x_{t-1}^* to decoding unit:

$$h_t^*, c_t^* = LSTM(x_{t-1}^*, h_{t-1}^*, c_{t-1}^*) \tag{20}$$

3.2 Temporal Attention Network

In the social attention network, we extract social contextual information from user v's social network. Another important factor that should be considered for imputation is temporal influence, that is, how a user's historical and future behaviors can relate to her current state. Similar to the social attention network, we use a memory-based attention network to capture such influence. We name this part of our proposed model Temporal Attention Network.

Given a specific user, we extract her observed sequence X' and interval sequence Δ. Like extracting memory representation before, we also compute its memory matrix C. More specifically, we first feed X' and Δ into another T-LSTM:

$$h_s, c_s = T - LSTM(x_s', \delta_s, h_{s-1}, c_{s-1}) \tag{21}$$

Hence, we get their hidden states $h = \{h_1, ..., h_s\}$, then we calculate memory representation C by (15).

In the decoding phase, we similarly compute the temporal context vector a. A slight difference is that its input is the memory representation of a single user, rather than the concatenation of memory representations from her neighbors:

$$a = \sum_{i=0}^{K} \beta_i C_i \tag{22}$$

$$\beta = softmax(W_\beta h^*) \tag{23}$$

3.3 Learning and Imputation

We concatenate the hidden representation h_t^*, social context vector $\hat{a_t}$, and temporal context vector a_t , which preserves the information from current states, social context and temporal context for predicting the targets. We then feed the concatenation to a fully connected layer and get the final prediction value.

$$x_t^* = \phi(h_t^*, \hat{a}_t, a_t) \tag{24}$$

where, $\phi(\cdot)$ is a non-linear function that maps the concatenation to the observations.

The decoding phase aims to generate the whole sequence and to minimize potential error in estimating the missing values. However, we cannot compute the observation that is truly missing in the sequence. Instead, we only calculate the error between the estimates and the actual observations. Thus, the total loss/error for the entire dataset $\mathcal{D}$ is the partial mean squared error (the $l2$-regularization term is omitted in the following expression):

$$\mathcal{L}(X^N, X^{*N}) = \sum_{n=1}^{N} [\sum_{t=1}^{T} \sum_{d=1}^{D} m_t^{(n)} \times (x_t^{d(n)} - x_t^{*d(n)})^2)] \tag{25}$$

To train this model, we process social context $\hat{X}$ to social attention network and original sequence X to temporal attention network, keeping track of each output and the latest hidden state. The first hidden states h_0^* of decoder is given by the last hidden states of the temporal context networks . To make the model converge faster and avoid over-fitting, we adopt a mixed strategy to feed the input data into the decoder. For some observations, we use the real target values as each next input if they are not missing; otherwise, we use the decoder's estimates as the next input. For the missing elements, we directly give the decoder's estimates. The fraction of samples trained using the real target values is denoted as γ. In the section of experiment, we set $\gamma = 0.9$.

After the model is well-trained, we can use it to impute the missing elements. Like in the learning phase, we process the social context $\hat{X}$ and original sequence X to the encoder and obtain the candidate context vector $\tilde{C}$ and C. For the decoder, we give the original values when data is observed; otherwise, we give estimates in order to use as much information as possible from the original sequences.

Algorithm 1 training procedure

```
while not converged do
    draw a mini-batch of sequences $\mathbf{X}^{(n)}$ and their corresponding social context sequence sets $\hat{\mathbf{X}}^{(n)}$
    // forward pass to encoder network
    compute social context vector $\hat{\mathbf{a}}^{(n)}$ and temporal context vector $\mathbf{a}^{(n)}$
    //we omit the symbol of batch size $(n)$ in the following statements
    for $t$ in $1, 2, ..., T$ do
        sample $p \sim \mathcal{U}(1)$
        if $p > \gamma$ then
            $h_t^*, c_t^* = LSTM(x_{t-1}^*, h_{t-1}^*, c_{t-1}^*)$
        else
            $h_t^*, c_t^* = LSTM(x_{t-1}^* \cdot (1 - m_{t-1}) + x_{t-1} \cdot m_{t-1}, h_{t-1}^*, c_{t-1}^*)$
        end if
        $x_t^* = \phi(h_t^*, \hat{a}_t, a_t)$
    end for
    compute the loss function $\mathcal{L}$
    // backward pass
    compute gradients and apply updates
end while
```

4 EXPERIMENTAL RESULTS

4.1 Datasets Description

We employ two datasets from the State Grid, the major electric power company in China, to construct our experiments. Before introducing our datasets, we highlight several facts about electricity data collection. Electricity data, including values of real-time current, voltage, and power, is measured by watt-hour meters. Each meter periodically transmits different types of data to its corresponding data collector. Usually, a collector receives data from several meters that are geographically close. For instance, these meters are assembled in different rooms on the same floor of a building. Hence, we can consider that the meters connecting to the same collector record the electricity usage of people or families with social relationships. Therefore, given a user v (corresponds to a meter), we define v's neighbors, which have social relations with v, as other users that share the same collector with v. More details of our datasets are described below:

- **Electrical Consumption (EC) :** This dataset is provided by the State Grid. In total, it includes the daily electrical consumption recorded by 80,000 watt-hour meters. This data spans from January 1st 2018 to March 31st 2018. The length of each series is 90.
- **Real-Time Voltage (RV) :** This dataset, provided by the State Grid, consists of around 20,000 electricity load series, each of which describes voltage values in three phases, recorded every 45 minutes in each day, from June 1st 2018 to June 30th 2018. Each sequence has 32 time stamps.

To reduce computational complexity, we at most consider 8 neighbors of a user, which are selected randomly.

4.2 Baselines and Metrics

4.2.1 Comparative Baselines and Implementation Details. We compare our model with multiple baselines:

- *Mean, Median:* We take the mean and median values of the observations in each channel to replace the missing data.
- *Linear, Cubic:* Linear interpolation and cubic interpolation are two interpolation methods that are used to impute the data.
- *KNN:* This method uses k-nearest neighbor to find similar samples and imputed unobserved data by calculating the weighted average of similar observations.
- *MICE [11]:* This is a method that combines the multiple imputation technique [37] with the chained equation approach to impute the missing data.
- *MissForest [40]:* This is a non-parametric imputation method that uses random forests trained on the observed values to predict the missing values.
- *Soft-Impute [29]:* This approach iteratively updates the missing elements by efficiently solving the convex relaxation of the approximate matrix completion problem.
- *GRU-D [13]:* This is a GRU-based model that incorporates masking and time intervals inside the GRU architecture to capture the informative missing data. We implement it slightly different from the original model proposed in [13]. Rather than use the hidden features of the last cell to predict the label, we use the hidden features in each cell to predict the next observations.
- *LSTM-Impute [26]:* This method initializes the missing elements with mean values, and updates them by training an RNN model with a LSTM unit.
- *VAE [47]:* This is a MCMC-based missing data imputation technique with a trained Variational Autoencoder (VAE).
- *STI :* This is our proposed model. We initialize our model with different parameters for different datasets. For the RV dataset, we use a 1-layer 32-unit T-LSTM encoder and a 1-layer 32 unit LSTM decoder. The memory size is also 32. The model is optimized using an Adam optimizer with a learning rate of 1×10^{-3} in the encoder and 5×10^{-3} in the decoder, and the batch size is 256. For the EC dataset, we set all LSTM dimensions to 16 and the memory size to 16. Again, we train the model using an Adam optimizer with a learning rate of 2×10^{-4} in our encoder and 2×10^{-3} in our decoder, and the batch size is 512. We set the dropout rate as 0.2 (0.8 keep probability).

4.2.2 Evaluation Metrics: We use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to measure our model. Their formulas are:

$$\begin{aligned} MAE &= \frac{\sum |x - x^*|}{n} \\ RMSE &= sqrt(\frac{\sum (x - x^*)^2}{n}) \end{aligned}$$

where x is a prediction, x^* is the ground truth, and n is the number of cases.

4.3 Performance Comparison with Randomly Missing Elements

To evaluate the performance of our model on these two datasets, we first assume that elements are randomly missing. With a missing rate θ, we randomly drop values for each time series from a Bernoulli distribution $\mathbf{U}(\theta)$. We experiment with θ varying from 0.2 to 0.6. In the training phase, we select the observed sequences for training our model, and use the dropped values to test its performance.

Table 1 reports the performance of our proposed method compared to all the other competing methods mentioned in Section 4.2.1 on the RV and EC datasets. Our proposed model significantly outperforms all comparative methods by achieving the lowest RMSE and MAE on these datasets. Another finding is that performance drops as the missing rate increases, which might be caused by the less information these models can learn with. Specifically, the simple imputation methods (Median and Mean) do not perform well, because they only use the observed values to complete the fixed values in each channel. For interpolate-based methods (Linear and Cubic), they further consider the intra-stream information at the nearest timestamp, and thereby achieve better performance. However, they lack the insight of inter-stream information. KNN, SoftImpute, MissForest, and MICE are traditional imputation methods that consider both intra-stream and inter-stream useful information. However, they fail to capture the complex temporal dependence. Therefore, our proposed method significantly outperforms those methods.

Dataset	**Missing Rate**	0.2		0.3		0.4		0.5		0.6	
	Method	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
EC	Mean	3.3787	4.3235	3.3794	4.3263	3.3810	4.3295	3.3850	4.3375	3.3913	4.3498
	Median	3.2818	4.5337	3.2850	4.5394	3.2905	4.5478	3.3015	4.5654	3.3151	4.5838
	Linear	1.5783	2.5173	1.6246	2.5835	1.6674	2.6431	1.7249	2.7246	1.7972	2.8248
	Cubic	2.0246	3.1914	2.1461	3.4118	2.2667	3.6288	2.4358	4.0081	2.6691	4.7918
	KNN	2.2455	3.3251	2.4224	3.5077	2.5762	3.6617	2.7576	3.8407	2.9672	4.0431
	SoftImpute	2.4018	3.5193	2.6459	3.7814	2.8377	3.9767	2.9746	4.1007	3.0319	4.1303
	MissForest	4.0659	5.3842	4.0528	5.3695	4.0474	5.3664	4.0294	5.3412	4.0068	5.3174
	MICE	3.4634	4.5654	3.4590	4.5777	3.4578	4.5919	3.4538	4.6152	3.4550	4.6591
	VAE	1.5375	2.3085	1.5883	2.4382	1.6504	2.4979	1.6882	2.6148	1.7374	2.6515
	LSTM-Impute	3.0315	4.2238	3.1687	4.3324	3.2529	4.3206	3.4526	4.5627	3.7708	4.7990
	GRU-D	1.7024	2.5568	1.9385	2.7868	2.0511	2.9136	2.0780	2.9304	1.9568	2.8918
	STI	**1.4667**	**2.2172**	**1.4864**	**2.2574**	**1.5207**	**2.3745**	**1.5696**	**2.3924**	**1.6159**	**2.4505**
RV	Mean	4.0893	5.0340	4.0957	5.0435	4.1076	5.0581	4.1184	5.0835	4.1547	5.1397
	Median	4.0250	5.2811	4.0465	5.2929	4.0701	5.3301	4.0975	5.3541	4.1594	5.4246
	Linear	2.0697	3.4058	2.1316	3.4778	2.2179	3.5714	2.3255	3.7051	2.5487	3.9549
	Cubic	2.7329	4.4551	2.8801	4.7857	3.0976	5.3014	3.3495	5.8316	3.9971	7.7123
	KNN	3.1175	4.3509	3.3162	4.5230	3.5550	4.7334	3.8224	4.9665	4.1645	5.2793
	SoftImpute	4.0263	5.1599	5.4152	6.9389	6.4592	8.4186	6.4171	8.4777	5.3860	7.0291
	MissForest	4.1727	5.3729	4.1825	5.3942	4.2012	5.4243	4.2203	5.4701	4.2952	5.5940
	MICE	4.3518	5.7909	4.3806	5.8305	4.4099	5.8764	4.4302	5.9083	4.4641	5.9477
	VAE	2.3001	3.2631	2.7272	4.5136	3.3440	6.4581	3.6293	6.7901	4.4053	8.8703
	LSTM-Impute	3.0315	4.2238	3.1687	4.3324	3.2529	4.3206	3.4526	4.5627	3.7708	4.7991
	GRU-D	2.8582	4.1190	3.0640	4.3150	3.1822	4.3652	3.1583	4.4811	3.5772	4.7590
	STI	**2.0008**	**2.9426**	**2.0787**	**3.0858**	**2.1258**	**3.1306**	**2.2795**	**3.3187**	**2.4963**	**3.5972**

Table 1: Performance of different models in imputation using the EC and RV datasets.The best results are in boldface, and the second-best results are underlined.

For neural networks-based methods, our method outperforms VAE, LSTM-Impute, and GRU-D. LSTM-Impute performs poorly in our evaluation, and its performance on the EC dataset is even worse than some traditional imputation methods. This might be because replacing the missing elements with average values can influence the original temporal distribution, which harms performance. GRU-D designs a recurrent architecture that uses both time intervals and masking information to update the missing data. However, it overlooks the importance of global temporal contextual information. VAE is a good approach for replacing missing data, but it lacks the modeling of temporal patterns. The best performance of our method demonstrates that incorporating social and temporal contextual information through the attention encoder is helpful for imputation.

4.4 Ablation Analysis

This section aims to demonstrate the effectiveness of the different components of our proposed model. We first list some variants of our models:

- *STI :* This is our complete model. We have described the parameter settings for this model in Section 4.2.1.
- *STI - social:* This model drops the social context vector in the output layer.
- *STI (LSTM)*: This variant uses the original LSTM as the encoder instead of T-LSTM. The remaining settings for this variant are the same as for our proposed model.
- *STI - social - temporal*: This variant removes both temporal and social contexts in our proposed model, i.e., an RNN with a LSTM cell.

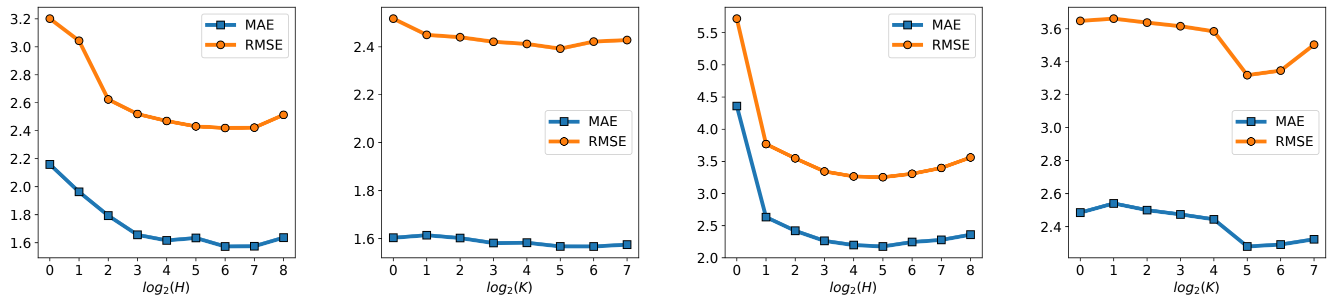
The result of different variants under various settings is reported in Table 2. We can observe the improvement due to adding the social attention network in most cases. This finding suggests that the factor of social relations is helpful, especially on the RV dataset. We also note that our model without social attention network still outperforms most cases when we drop our attention encoder, which demonstrates the importance to model temporal influence in a proper way. We also observed that when we replace the T-LSTM encoder as a LSTM encoder, performance drops. Therefore, using the T-LSTM encoder is helpful.

4.5 Parameter Analysis

In this section, we explore how the hyper-parameters influence the performance of our model. We fix the remaining hyper-parameters of our models when we vary one hyper-parameter. We present the results for the EC and RV datasets with the same missing rate of 0.5. Figure 5(a) and Figure 5(c) report the relationship between cell size H and final performance. We observed that our model, with

Dataset	Missing Rate	0.2		0.3		0.4		0.5		0.6	
	Method	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
EC	STI	1.4667	**2.2172**	**1.4864**	**2.2574**	**1.5207**	2.3745	1.5696	**2.3924**	**1.6159**	**2.4505**
	STI - social	**1.4628**	2.2337	1.4985	2.3364	1.5463	**2.3432**	**1.5672**	2.4208	1.6161	2.4593
	STI (LSTM)	1.4732	2.2894	1.4982	2.3165	1.5471	2.3747	1.5837	2.4231	1.6626	2.4938
	STI - temporal - social	1.5066	2.3134	1.5384	2.4002	1.5822	2.4175	1.5903	2.4510	1.6851	2.5350
RV	STI	**2.0008**	**2.9426**	**2.0787**	**3.0858**	**2.1258**	**3.1306**	**2.2795**	3.3187	**2.4963**	3.5972
	STI - social	2.0487	3.0071	2.1167	3.1362	2.1383	3.1392	2.2912	3.3465	2.5390	3.6977
	STI (LSTM)	2.0584	2.9972	2.0916	3.1781	2.1498	3.1482	2.2826	**3.2925**	2.5219	**3.5675**
	STI - temporal - social	2.0641	3.0035	2.1920	3.1756	2.2661	3.2115	2.3573	3.3520	2.6095	3.7819

Table 2: Ablation analysis for the EC and RV datasets. The best results are in boldface and the second-best results are underlined.



(a) Varing cell size H on EC dataset. (b) Varing memory size K on EC dataset. (c) Varing cell size H on RV dataset. (d) Varing memory size K on RV dataset.

Figure 5: Model parameter analysis. (a), (b) testing on EC dataset and (c), (d) testing on RV dataset. The Missing ratio is set as 0.5. (a), (c) presents the sensitivity of cell size H and (b), (d) shows the sensitivity of memory size K.

a cell size that is too small (less than 4 on both datasets) or too large (more than 32 on RV) can decrease imputation performance. Hence, choosing a proper cell size H is crucial. Memory size K is another crucial hyper-parameter. Figure 5(b) and Figure 5(d) present how K influences the model's performance. For memory-based attention, memory size K can be considered as a trade-off between computational time and representational power. A large K allows the model to compute complex source representations, while a small K of 1 limits the source representation to a single vector. Britz et al. [10] showed that a larger K can achieve better performance in neural translation task. In our experiment, we find that within a certain range, the performance increases as the K increases in size. However, we also note that performance drastically drops when K becomes relatively large. This might be related to the over-fitting phenomenon, whereby the capability of representation becomes large.

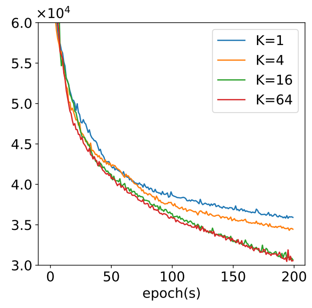
We also explore how varying Ks influences the learning procedure. Figure 6 shows that a larger K tends to have a faster convergence in terms of training loss on both two datasets.

4.6 Generalization Analysis

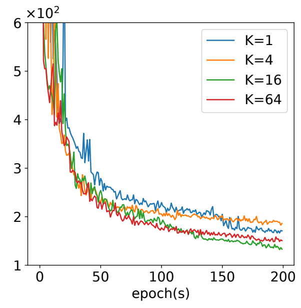
In the previous sections, we use the full dataset to train the model, and it achieves very good performance. However, in many real-world senarios, the method is impractical due to the huge scale of historical data and the new data that is being collected continuously. Based on these concerns, we conduct a new experimental setting to explore how our model performs when trained with a subset of collected data. In particular, we randomly draw $1/n$ ($n = 1, 2, 4, ..., 64$) from the EC dataset with a missing rate of 0.5 , and use this data to train the model. Then we use the trained model to fill the missing elements in the full datasets. Figure 7 presents the results. Not surprisingly, performance drops as our model is trained with less data in both metrics. However, we also observe that the degree of the drop is relatively slight. For instance, using only 1/32 (around 2.5K time series) of the datasets, our model can still outperform other baselines. This suggests that our model has a good capacity for generalization.

4.7 Comparison of Performance using Simulated Real-World Missing Data

In the previous section, we experiment with the assumption that elements are randomly missing and observed that our method outperforms other baselines with various missing rates. However,



(a) Training curve on the EC dataset.



(b) Training curve on the RV dataset.

Figure 6: Comparison concerning varing memory size K for the training loss curve on EC and RV datasets. It shows that a larger K tends to converge faster.

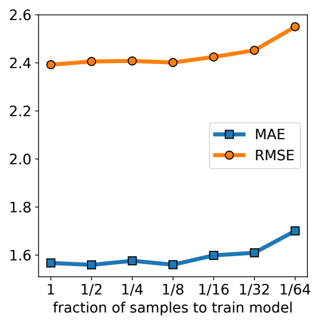


Figure 7: Comparison of performance when applying a subset of the whole EC dataset with a missing rate of 0.5 to train our model. The x-axis indicates the fraction of samples is used. The y-axis indicates the two evaluation metrics.

in the real-world, completely random missing is rare in time series , because data is usually missing in blocks. To evaluate the accuracy of our method in completing the elements missing in real-world situations, we establish a new setting in the EC dataset. Firstly, we extend the time span of the EC dataset from the first 90 days to the first 180 days in 2018. We then partition the extended dataset into two parts, named EC-A and EC-B , each of which has consecutive data for 90 days. We extract the masking matrix from EC-A and EC-B, and name them M_A and M_B, respectively. We drop the values of the elements in EC-A that correspond to its successor's (i.e., from the 91st day to the 180th day in 2018) masking matrix M_B. If those elements are present in EC-A, their values are used as a ground truth to measure the performance of our predictions. If a time series is all observed in EC-B, then we drop this series in our datasets. In total, the dataset used in this section consists of around 20K time series. In this section, we do not randomly drop some non-missing elements to generate ground truth, as real-world missing patterns are not random. Figure 8 shows an example of how to drop missing elements.

Table 3 reports the comparison between the levels of performance of the different models on this dataset. Our model achieves state-of-the-art performance on this dataset and its improvement on the performance of other baseline methods is significant. This

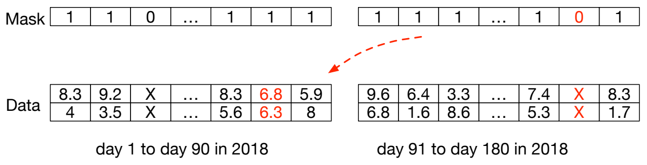

Mask
1 1 0 ... 1 1 1
1 1 1 ... 1 0 1
Data
8.3 9.2 X ... 8.3 6.8 5.9
4 3.5 X ... 5.6 6.3 8
9.6 6.4 3.3 ... 7.4 X 8.3
6.8 1.6 8.6 ... 5.3 X 1.7
day 1 to day 90 in 2018
day 91 to day 180 in 2018

Figure 8: An illustration of ground truth generation on the EC dataset to simulate real-world missing data.

finding demonstrates that our model can also achieve a good performance in the real-world situations in which the elements are not randomly missing. The performance of our model that does not consider social influence is lower than our complete model, but it still remains competitive concerning these benchmarks. Linear Interpolation is still a competitive method, and following are three neural imputation models. Another interesting finding is that while the performance of our model on this dataset is slightly better than that on the EC dataset, with a missing rate of 0.4, many baseline methods (e.g. MICE, Cubic) performs worse than they do under the same condition. This again demonstrates the adaptability and stability of our model under different missing conditions.

Method	MAE	RMSE	Method	MAE	RMSE
Mean	2.7626	4.1134	Median	2.8156	4.4493
Linear	1.7112	2.9973	Cubic	9.2609	67.5511
KNN	2.5144	3.9050	SoftImpute	2.5384	3.9342
MICE	2.8304	4.3208	MissForest	3.2628	4.9611
VAE	1.7067	3.0243	LSTM-Impute	2.4445	3.8235
GRU-D	1.9298	3.3543	STI - social	<u>1.6223</u>	<u>2.6731</u>
STI	**1.5837**	**2.6412**			

Table 3: Performance on the EC dataset with simulated missing data.The best results are in boldface, and the second-best results are underlined.

5 RELATED WORK

Time Series Modeling. For modeling time series data, plenty of parametric methods are proposed, including the auto-regressive models [4, 39], the kernel models [25], and the hidden Markov models [46, 49]. Recently, many time series model based on neural networks have been explored [21, 31]. More recent works explore to use deep learning methods to extract high-level representation of time series data [43, 44].

Traditional imputation methods for time series data. Over the years, quite a few imputation methods for time series data have been explored. The majority of them have concentrated on reconstructing missing elements by utilizing the temporal relationship within each data stream, such as the EM algorithm [18],the MissForest, [40] and the kernel method[35]. Multiple imputation [11] was also developed with these imputation methods to reduce uncertainty, by repeating the imputation procedure several times and giving the averaged estimates. However, running multiple models to fit the measured data during imputation introduces a great deal computational cost, which is a practical concern.

Imputation methods based on recurrent neural networks. Recently, recurrent neural networks (RNNs) have achieved the state-of-the-art performance in many tasks with sequential data, including machine translation and speech recognition. RNNs generate a sequence of hidden states h_t , as a function of the previous hidden states h_{t-1} and the input for position t. RNNs with LSTM or GRU units enjoy several benefits like capturing long-term sequential dependencies and supporting variable-length data input. The RNNs for missing data have also been studied in many works and have been applied in speech recognition [2] and mortality prediction [13]. Previous works in this area [8, 41] have tried to first replace the missing elements with fixed values (such as the mean), most recent observations or values learned by other imputation algorithms, and then use the RNNs to update the initial data. However, the input sequences with pre-imputed values might have a different data distribution from the original ones, which can influence final performance when applying the RNNs to fit the data with pre-imputed values.

More recent advances take time irregularity into account. In other words, these papers model observations and intervals at the same time [13, 15, 50]. By concatenating information of observations and time gaps, they use these concatenation as the input for their customized RNNs. Different approaches of concatenation have been tried in these works. For instance, Che et al. [13] concatenate the latest observations, mean values, and time intervals as inputs to fit the customized GRU network. However, these works lack the insight of the temporal context of the whole time series and do not consider incorporating temporal and social patterns .

Recurrent neural networks with irregular intervals. Most of the proposed studies on RNNs assume that the interval between two consecutive timestamps is a constant value. However, it may not hold true in many real-world scenarios [7, 52]. For addressing this issue, many studies consider modifying gate structures of the LSTM cell [7, 24, 34]. Some studies also explore to add new gates to better model time gaps [32, 52]. In particular, Neil et al. [32] introduce a new gate, named time gate, and equip it with LSTM to model time intervals between two successive user behavior. Besides, Baytas et al. [7] adjust the memory cell of LSTM by discounting the short-term memory, and Beutel et al. [9] incorporate time gaps in the RNN by performing an element-wise product of model's hidden states with the embedding of time gaps.

Sequence modeling with attention mechanism. Learning sequence alignments via attention is also relevant to our work. Recently, sequence-to-sequence models with attention mechanism have achieved remarkable successes in many tasks, such as machine translation [10, 28], speech recognition [6], and image caption generation [27].This success has motivated us to extend this mechanism to the time series imputation field. Attention mechanisms can be viewed as feature extractors that allow the modeling of dependencies without regard to their distance in the input or output sequences [5]. The most popular approaches first build an encoder-decoder architecture consisting of two RNNs and use an attention mechanism to align the target to source tokens [5]. Recently, research in this field has focused on reducing the computational cost. Luong et al. [28] introduce various attention mechanisms that are computationally simpler and perform just as well as the model in [28]. Kalchbrenner et al. [23] propose a linear time architecture based on stacked convolutional neural networks. Gehring et al. [20] also propose the use of convolutional encoders to speed up neural machine translation. de Brébisson and Vincent [17] explore a linear attention mechanism using covariance matrices for information retrieval. Britz et al. [10] propose an end-to-end memory network based on a recurrent attention mechanism instead of sequence-aligned recurrence, and this method has shown good performance on several language modeling tasks, and is used in our model. Vaswani et al. [42] try to increase the capability of parallelism of the attention mechanism by using pure attention networks rather than recurrent structures. Location-based attention [48] has also been studied in the field of image recognition.

Time series in social networks. Our work is also relevant to studies on the evolution of user behavior in social networks [3, 12, 16, 30]. Using data from social networks, these studies explore the temporal connection between individual behavior and network properties. For instance, Asur and Huberman [3] use the time series of users' tweets about movies to predict movies' revenue. McAuley and Leskovec [30] study the process of user expertise on review websites, and find that modeling users' experience helps to discover when users acquire product in rating systems. Many studies examine the spread of behaviors or health conditions based on temporal data collected in networks [12, 16]. In particular, Christakis and Fowler [16] examine the spread of obesity on a large social network of 12K people over 32 years, and find that social distance tends to be more important than geographic distance with people's networks.

6 CONCLUSION AND FUTURE WORK

To what extent can your friends disclose your information? In this paper, we proposed a novel social-aware time series imputation framework, that considers social influence and temporal influence to infer missing values in time series. To incorporate these two factors, we designed two attention mechanisms in our model. Experimental results on real-world datasets show that our model can consistently outperform 11 baseline methods.

There are at least two kinds of perspectives. From the social networks side, in this paper we only exploited a small part of contextual information in social networks, we would like to utilize other useful patterns such as network structures and user interactions in the future work. From the practical side, we would like to explore the use of our approach into more real-world scenes.

Acknowledgements. The work is supported by NSFC (61702447), the Fundamental Research Funds for the Central Universities, and a research funding from the State Grid of China.

REFERENCES

[1] Sahil Agarwal, Sanket Khade, Yogesh Dandawate, and Prasad Khandekar. 2015. Three dimensional image reconstruction using interpolation of distance and image registration. In *IC4*. IEEE, 1–5.

[2] Abdul Manan Ahmad, Saliza Ismail, and DF Samaon. 2004. Recurrent neural network with backpropagation through time for speech recognition. In *ISCIT*,

Vol. 1. IEEE, 98–102.

[3] Sitaram Asur and Bernardo A Huberman. 2010. Predicting the future with social media. In *WI/IAT*. IEEE Computer Society, 492–499.

[4] Anthony Bagnall and Gareth Janacek. 2014. A Run Length Transformation for Discriminating Between Auto Regressive Time Series. *Journal of Classification* 31, 2 (2014), 154–178.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[6] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *ICASSP*. IEEE, 4945–4949.

[7] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. 2017. Patient subtyping via time-aware LSTM networks. In *SIGKDD*. 65–74.

[8] Yoshua Bengio and Francois Gingras. 1996. Recurrent neural networks for missing or asynchronous data. In *Advances in neural information processing systems*. 395–401.

[9] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *WSDM*. ACM, 46–54.

[10] Denny Britz, Melody Guan, and Minh-Thang Luong. 2017. Efficient Attention using a Fixed-Size Memory Representation. In *EMNLP*. 392–400.

[11] S van Buuren and Karin Groothuis-Oudshoorn. 2010. mice: Multivariate imputation by chained equations in R. *Journal of statistical software* (2010), 1–68.

[12] Damon Centola. 2010. The spread of behavior in an online social network experiment. *science* 329, 5996 (2010), 1194–1197.

[13] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8, 1 (2018), 6085.

[14] Xi Chen, Jefrey Lijffijt, and Tijl De Bie. 2018. Quantifying and minimizing risk of conflict in social networks. In *SIGKDD*. ACM, 1197–1205.

[15] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*. 301–318.

[16] Nicholas A Christakis and James H Fowler. 2007. The spread of obesity in a large social network over 32 years. *New England Journal of Medicine* 357, 4 (2007), 370–379.

[17] Alexandre de Brébisson and Pascal Vincent. 2016. A Cheap Linear Attention Mechanism with Fast Lookups and Fixed-Size Representations. *arXiv preprint arXiv:1609.05866* (2016).

[18] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977), 1–38.

[19] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent Marked Temporal Point Processes:Embedding Event History to Vector. In *SIGKDD*. 1555–1564.

[20] Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A Convolutional Encoder Model for Neural Machine Translation. In *ACL*, Vol. 1. 123–135.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[22] Vijay Manikandan Janakiraman, Bryan Matthews, and Nikunj Oza. 2017. Finding Precursors to Anomalous Drop in Airspeed During a Flight's Takeoff. In *SIGKDD*. ACM, 1843–1852.

[23] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* (2016).

[24] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction.. In *IJCAI*. 2341–2347.

[25] T. Kurashima, T. Althoff, and J. Leskovec. 2018. Modeling Interdependent and Periodic Real-World Action Sequences. (2018).

[26] Zachary C Lipton, David Kale, and Randall Wetzel. 2016. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Machine Learning for Healthcare Conference*. 253–270.

[27] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, Vol. 6. 2.

[28] Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*. 1412–1421.

[29] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2010. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research* 11, Aug (2010), 2287–2322.

[30] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*. ACM, 897–908.

[31] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *ISCA*.

[32] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. 2016. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*. 3882–3890.

[33] Tim Op De Beéck, Wannes Meert, Kurt Schütte, Benedicte Vanwanseele, and Jesse Davis. 2018. Fatigue Prediction in Outdoor Runners Via Machine Learning and Sensor Fusion. In *SIGKDD*. ACM, 606–615.

[34] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. 2016. Deepcare: A deep dynamic memory model for predictive medicine. In *PAKDD*. Springer, 30–41.

[35] Kira Rehfeld, Norbert Marwan, Jobst Heitzig, and Jürgen Kurths. 2011. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics* 18, 3 (2011), 389–404.

[36] Christian H Reinsch. 1967. Smoothing by spline functions. *Numer. Math.* 10, 3 (1967), 177–183.

[37] Donald B Rubin. 2004. *Multiple imputation for nonresponse in surveys*. Vol. 81. John Wiley & Sons.

[38] Abraham Savitzky and Marcel JE Golay. 1964. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry* 36, 8 (1964), 1627–1639.

[39] Mohammad Shokoohi-Yekta, Yanping Chen, Bilson Campana, Bing Hu, Jesin Zakaria, and Eamonn Keogh. 2015. Discovery of Meaningful Rules in Time Series. In *SIGKDD*. 1085–1094.

[40] Daniel J Stekhoven and Peter Bühlmann. 2011. MissForest: non-parametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (2011), 112–118.

[41] Volker Tresp and Thomas Briegel. 1998. A solution for missing data in recurrent neural networks with an application to blood glucose prediction. In *Advances in Neural Information Processing Systems*. 971–977.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[43] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis. (2018), 2437–2446.

[44] Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2018. PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning. (2018).

[45] Norbert Wiener, Norbert Wiener, Cyberneticist Mathematician, Norbert Wiener, Norbert Wiener, and Cybernéticien Mathématicien. 1949. Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications. (1949).

[46] Tao Wu and David F Gleich. 2017. Retrospective Higher-Order Markov Processes for User Trails. In *SIGKDD*. 1185–1194.

[47] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *WWW*. ACM, 187–196.

[48] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. 2048–2057.

[49] Yun Yang and Jianmin Jiang. 2014. HMM-based hybrid meta-clustering ensemble for temporal data. *Knowledge-Based Systems* 56, C (2014), 299–310.

[50] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. 2017. Multi-directional Recurrent Neural Networks: A Novel Method for Estimating Missing Data. (2017).

[51] Chen Zhang, Yijun Wang, Can Chen, Changying Du, Hongzhi Yin, and Hao Wang. 2018. StockAssIstant: A Stock AI Assistant for Reliability Modeling of Stock Comments. In *SIGKDD*. ACM, 2710–2719.

[52] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*. 3602–3608.