Neurocomputing 312 (2018) 177–186

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Period-aware content attention RNNs for time series forecasting with missing values

Yagmur Gizem Cinar[a,*], Hamid Mirisaee[a], Parantapa Goswami[b], Eric Gaussier[a], Ali Aït-Bachir[c]

[a] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble F-38000, France
[b] Viseo Technologies, Grenoble 38000, France
[c] Coservit, Grenoble 38100, France

ARTICLE INFO

Article history:
Received 4 December 2017
Revised 21 March 2018
Accepted 24 May 2018
Available online 31 May 2018

Communicated by Zechao Li

Keywords:
Recurrent neural networks
Attention model
Time series

ABSTRACT

Recurrent neural networks (RNNs) recently received considerable attention for sequence modeling and time series analysis. Many time series contain periods, e.g. seasonal changes in weather time series or electricity usage at day and night time. Here, we first analyze the behavior of RNNs with an attention mechanism with respect to periods in time series and illustrate that they fail to model periods. Then, we propose an extended attention model for sequence-to-sequence RNNs designed to capture periods in time series with or without missing values. This extended attention model can be deployed on top of any RNN, and is shown to yield state-of-the-art performance for time series forecasting on several univariate and multivariate time series.

1. Introduction

Forecasting future values of temporal variables is termed as *time series forecasting* and has applications in a variety of fields, as finance, economics, meteorology, or customer support center operations. A considerable number of stochastic [1] and machine learning based [2] approaches have been proposed for this problem. A particular class of approaches that has recently received much attention for modeling sequences is based on sequence-to-sequence Recurrent Neural Networks (RNNs) [3]. Sequence-to-sequence RNNs constitute a flexible class of methods particularly well adapted to time series forecasting when one aims at predicting a sequence of future values on the basis of a sequence of past values. They have furthermore led to state-of-the-art results in different applications (as machine translation or image captioning). We explore here how to adapt them to general time series.

Time series often display *pseudo-periods, i.e.* time intervals at which there is a strong correlation, *positive or negative*, between the values of the time series. Pseudo-periods can be due for example to environmental factors as seasonality or to the patterns underlying the activities measured (the workload on professional email servers for example has both weekly and daily periods). In a forecasting scenario, pseudo-periods correspond to the difference between the positions of the output being predicted and specific inputs. In the remainder of the paper, we will use the term *period* to refer to either a true period or a pseudo-period.

A first question one can ask is thus: *Can sequence-to-sequence RNNs model periods in time series?* In order to capture periods in RNNs, one needs a *memory* of the input sequence, *i.e.* a mechanism to reuse specific (representations of) input values to predict output values. As the input sequence is usually longer than the periods underlying the time series, longer-term memories that store information pertaining to past input sequences (as described in *e.g.* [4–6]) are not required. A particular model of interest here is the content attention model proposed in [7] and described in Section 2. This model allows one to reuse the content of the input sequence to predict the output values. However, this model was designed for text translation and one can wonder whether it can capture position-based periods in time series.

Furthermore, missing observations and gaps in the data are also common, due to *e.g.* lost records and/or mistakes during data entry or faulty sensors. In addition, for multivariate time series, underlying variables can have mixed sampling rates, *i.e.* different variables may have different sampling frequencies, resulting again in values missing at certain times when comparing the different variables. The standard strategy to tackle missing values is to apply numerical or deterministic approaches, such as interpolation and data imputation methods, to explicitly estimate or infer the missing values, and then apply classical methods for forecasting. In the

* Corresponding author.
E-mail address: yagmur.cinar@imag.fr (Y.G. Cinar).

https://doi.org/10.1016/j.neucom.2018.05.090
0925-2312/

context of RNNs, a standard technique, called padding, consists in repeating the hidden state of the last observed value over the missing values. All these methods however make strong assumptions about the functional form of the data, especially for long gaps, and can introduce sources of errors and biases. The second question we address here is thus: *Do sequence-to-sequence RNNs handle well missing values?*

The main contributions of this work address the above two questions and are based on:

- A complete and general framework for time series forecasting using a period-aware content attention mechanism;
- An experimental study showing the incapability of RNNs in detecting (pseudo-)periods in time series;
- Several new models and architectures to efficiently handle missing values and gaps in time series. These models are combined with the period-aware content attention mechanism mentioned above and yield state-of-the-art performances on several benchmark data sets.

The remainder of the paper is organized as follows. We first conduct in Section 2 simple experiments to illustrate how state-of-the-art RNNs behave on time series. Next, in the same section, different extensions of the attention mechanism are provided in order to handle periods and missing values for both univariate and multivariate time series. To evaluate the proposed approach, an extensive set of experiments has been conducted. This set of experiments is presented and discussed in Section 4. Related studies are then discussed in Section 5. Finally, Section 6 concludes the paper.

2. Handling periodicity and missing values in RNNs

We first focus on univariate time series. As mentioned before, time series forecasting consists in predicting future values from past, observed values. The time span of the past values, denoted by T, is termed as *history*, whereas the time span of the future values to be predicted, denoted by T', is termed as *forecast horizon* (in multi-step ahead prediction, which we consider here, $T' > 1$). The prediction problem can be formulated as a regression-like problem where the goal is to learn the relation $\mathbf{y} = r(\mathbf{x})$ where $\mathbf{y} = (y_{T+1}, \ldots, y_{T+i}, \ldots, y_{T+T'})$ is the output sequence and $\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_T)$ is the input sequence. Both input and output sequences are ordered and indexed by time instants. Here, we use bold characters for vectors and matrices.

2.1. Background

Sequence-to-sequence RNNs rely on three parts, one dedicated to encoding the input, and referred to as *encoder*, one dedicated to constructing a summary of the encoding, which we will refer to as *summarizer*, and one dedicated to generating the output, and referred to as *decoder*. The encoder represents each input x_j, $1 \le j \le T$ as a *hidden state* $\overrightarrow{\mathbf{h}}_j = f(x_j, \overrightarrow{\mathbf{h}}_{j-1})$, $\overrightarrow{\mathbf{h}}_j \in \mathbb{R}^n$, where the function f corresponds here to the non-linear transformation implemented in LSTM with peephole connections [8]. The equations of LSTM with peephole connections are given in Eq. (1). For bidirectional RNNs [9], the input is read both forward and backward, leading to two vectors $\overrightarrow{\mathbf{h}}_j = f(x_j, \overrightarrow{\mathbf{h}}_{j-1})$ and $\overleftarrow{\mathbf{h}}_j = f(x_j, \overleftarrow{\mathbf{h}}_{j+1})$. The final hidden state for any input x_j is constructed simply by concatenating the corresponding forward and backward hidden states, *i.e.* $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]^\top$, where now $\mathbf{h}_j \in \mathbb{R}^{2n}$:

$$
\begin{aligned}
\mathbf{i}_j &= \sigma(\mathbf{W}_i x_j + \mathbf{U}_i \mathbf{h}_{j-1} + \mathbf{C}'_i \mathbf{c}'_{j-1}) \\
\mathbf{f}_j &= \sigma(\mathbf{W}_f x_j + \mathbf{U}_f \mathbf{h}_{j-1} + \mathbf{C}'_f \mathbf{c}'_{j-1}) \\
\mathbf{c}'_j &= \mathbf{f}_j \mathbf{c}'_{j-1} + \mathbf{i}_j \tanh(\mathbf{W}_{c'} x_j + \mathbf{U}_{c'} \mathbf{h}_{j-1}) \\
\mathbf{o}_j &= \sigma(\mathbf{W}_o x_j + \mathbf{U}_o \mathbf{h}_{j-1} + \mathbf{C}'_o \mathbf{c}'_{j-1}) \\
\mathbf{h}_j &= \mathbf{o}_j \tanh(\mathbf{c}'_j)
\end{aligned}
\tag{1}
$$

The summarizer builds, from the sequence of input hidden states $\mathbf{h}_j$, $1 \le j \le T$, a context vector $\mathbf{c} = q(\{\mathbf{h}_1, \ldots, \mathbf{h}_j, \ldots, \mathbf{h}_T\})$. In its simplest form, the function q just selects the last hidden state [3]: $q(\{\mathbf{h}_1, \ldots, \mathbf{h}_j, \ldots, \mathbf{h}_T\}) = \mathbf{h}_T$. More recently, in [7], an attention mechanism is used to construct different context vectors $\mathbf{c}_i$ for different outputs y_i ($1 \le i \le T'$) as a weighted sum of the hidden states of the encoder representing the input history:

$$
\mathbf{c}_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j \tag{2}
$$

where α_{ij} are the *attention weights*. They are computed as:

$$
\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'=1}^{T} \exp(e_{ij'})} \tag{3}
$$

where

$$
e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a \mathbf{h}_j) \tag{4}
$$

with a being a feedforward neural network with weights $\mathbf{W}_a$, $\mathbf{U}_a$ and $\mathbf{v}_a$ trained in conjunction with the entire encoder-decoder framework. $\mathbf{s}_{i-1}$ is the hidden state obtained by the decoder (see below) at time $i-1$. One can note that a scores the importance of the input at time j (specifically its representation $\mathbf{h}_j$ by the encoder) for the output at time i, given the previous hidden state $\mathbf{s}_{i-1}$ of the decoder. This allows the model to concentrate, or *put attention*, on certain parts of the input history to predict each output.

The decoder parallels the encoder by associating each output y_i, $T+1 \le i \le T+T'$, to a hidden state vector $\mathbf{s}_i = g(y_{i-1}, \mathbf{s}_{i-1}, \mathbf{c}_i)$, where $\mathbf{s}_i \in \mathbb{R}^n$ and y_{i-1} denotes the output at time $i-1$. The function g corresponds again to an LSTM with peephole connections, with an explicit context added [3]. The equations of LSTM with peephole connections and the context are given in Eq. (5). Each output is then predicted in sequence through:

$$
y_i = \mathbf{W}_{\text{out}} \mathbf{s}_i + b_{\text{out}},
$$

where b_{out} is a scalar and $\mathbf{W}_{\text{out}} \in \mathbb{R}^n$:

$$
\begin{aligned}
\mathbf{i}_i &= \sigma(\mathbf{W}_i y_{i-1} + \mathbf{U}_i \mathbf{s}_{i-1} + \mathbf{C}'_i \mathbf{c}'_{i-1} + \mathbf{C}_i \mathbf{c}_i) \\
\mathbf{f}_i &= \sigma(\mathbf{W}_f y_{i-1} + \mathbf{U}_f \mathbf{s}_{i-1} + \mathbf{C}'_f \mathbf{c}'_{i-1} + \mathbf{C}_f \mathbf{c}_i) \\
\mathbf{c}'_i &= \mathbf{f}_i \mathbf{c}'_{i-1} + \mathbf{i}_i \tanh(\mathbf{W}_{c'} y_{i-1} + \mathbf{U}_{c'} \mathbf{s}_{i-1} + \mathbf{C}_{c'} \mathbf{c}_i) \\
\mathbf{o}_i &= \sigma(\mathbf{W}_o y_{i-1} + \mathbf{U}_o \mathbf{s}_{i-1} + \mathbf{C}'_o \mathbf{c}'_{i-1} + \mathbf{C}_o \mathbf{c}_i) \\
\mathbf{s}_i &= \mathbf{o}_i \tanh(\mathbf{c}'_i)
\end{aligned}
\tag{5}
$$

Below, we first illustrate that traditional RNNs with the standard attention mechanism fail to capture the periods and, as a result, cannot make use of this point in order to improve the prediction accuracy. We then detail several extensions of the attention mechanism that efficiently integrate periods and handle missing values.

2.2. Standard RNNs and periods

To illustrate how RNNs behave w.r.t. periods in time series, we retained two periodic time series, fully described in Section 4 and representing respectively (a) the electrical consumption load in Poland over the period of 10 years (1/1/2002-7/31/2012), and

Table 1
MSE values for RNN-A with and without position information on PSE and PW.

Dataset	RNN-A	RNN-A with position
PSE	0.034	0.048
PW	0.166	0.154

(b) the maximum daily temperature, again in Poland, over the period of 12 years (1/1/2002-7/31/2014). The first time series, referred to as PSE, has two main periods, a daily one and a weekly one, whereas the second time series, referred to as PW, has a yearly period. Fig. 1 (top) displays the autocorrelation [10] for each time series and shows these different periods.

To verify if RNNs with standard attention mechanism, denoted as **RNN-A** hereafter, are able to capture the periods, we plot, in Fig. 1 (bottom), the attention weights obtained with RNN-A. In addition, we conducted another experiment where we added the time stamp (or position) information j to the input to help the RNNs capture potential periods. When the positions are added, each input at time j contains two values: x_j and j, $1 \leq j \leq T$. The position is of course not predicted in the output. The results we obtained, evaluated in terms of mean squared error (MSE) and displayed in Fig. 1 (bottom), show that adding the time stamps in the input does not improve the RNNs. Were the original RNN-A able to capture periods, we should see that the corresponding weights in the attention mechanism are higher. However, as one can note, this is not the case: for RNN-A with time stamp information, a higher weight is put on the first instance, which does not correspond to any of the actual periods of the times series. With standard RNN-A, higher weights are put on the more recent history on PSE. This seems more reasonable, and leads to better results, even though the period at one day is missed. On PW, the weights with standard RNN-A are uniformly distributed on the different time stamps. This shows that the standard attention mechanism does not always detect (and make us of) the underlying periods of the data.

To further illustrate this point, we show, in Table 1, the MSE values for both PSE and PW datasets with and without position information. Although adding position can improve the results on PW, it fails to perform better than RNN-A on PSE. This suggests that the attention mechanism needs to be adapted to a different type of information in order to capture periods. We propose such extensions below.

2.3. Period-based content attention mechanism

We assume here that the periods of a time series lie in the set $\{1, \ldots, T\}$ where T is the history size of the time series[1]. One can then explicitly model all possible periods as a real vector, which we will refer to as $\boldsymbol{\pi}$, of dimension T ($\in \mathbb{R}^T$), the coordinates of which encode the importance of the relative position between the output and the input. From this, one can modify the weight of the original attention mechanism relating input j to output i as follows:

$$e_{ij} = \begin{cases} \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a(\boldsymbol{\pi}_{i-j}\, \mathbf{h}_j)) & \text{if } (i-j) \leq T \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$\boldsymbol{\pi}_{i-j}$, i.e. the $(i-j)$th coordinate of π, is used to increase or decrease e_{ij} and thus the weight α_{ij} relating input j to output i. Note that, as the history is limited to T, there is no need to consider dependencies between an input j and an output i that are beyond T time steps (hence the test: $i - j \leq T$).

[1] This assumption is easy to satisfy by increasing the size of the history if the periods are known or by resorting to a validation set to tune T.

The vector $\mathbf{e}_i$ can then be normalized using the softmax operator again, and a context can be built by taking the expectation of the hidden states over the normalized weights. For practical purposes, however, one can simplify the above formulation by extending the vectors with T' dimensions that are set to 0 by considering a scalar $\mathbb{1}_{(i-j\leq T)}$ that returns 1 on the first T coordinates and 0 on the last T' ones. The resulting position-based attention mechanism then amounts to:

$$e_{ij} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a(\boldsymbol{\pi}_{i-j}\, \mathbf{h}_j))\, \mathbb{1}_{(i-j\leq T)} \tag{7}$$

As one can note, $\boldsymbol{\pi}_{i-j}$ will either decrease or increase the hidden state vector $\mathbf{h}_j$ for output i. Since $\boldsymbol{\pi}$ is learned along with the other parameters of the RNNs, we expect that $\boldsymbol{\pi}_{i-j}$ will be high for those positions i and j that correspond to periods of the time series. Lastly, note that the original attention mechanism can be obtained by setting $\boldsymbol{\pi}$ to $\mathbf{1}$ (a vector consisting of 1 on each coordinate). We will refer to this model as RNN-π.

In the above formulation, the position information is used to modify the importance of each hidden state in the input side. It may be, however, that some elements in $\mathbf{h}_j$ are less important than others to predict output i. It is possible to capture this by considering a vector in $\mathbb{R}^{2n}$ that can now reweigh each coordinate of $\mathbf{h}_j$ independently. Here, using a matrix, $\boldsymbol{\Pi}$, one can have a finer control over the encoder hidden states. This leads to:

$$e_{ij} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a(\boldsymbol{\Pi}_{.(i-j)} \odot \mathbf{h}_j))\, \mathbb{1}_{(i-j\leq T)} \tag{8}$$

where $\odot$ denotes the Hadamard product (element wise multiplication) and $\boldsymbol{\Pi}$ is a matrix in $\mathbb{R}^{2n\times T}$; $\boldsymbol{\Pi}_{.(i-j)}$ represents the $(i-j)^{th}$ column of this matrix. We will refer to this model as RNN-Π. To calculate the attention weights $\boldsymbol{\alpha}_i$ and context $\mathbf{c}_i$, we rely on Eqs. (2) and (3).

2.4. Handling missing values

Provided that their proportion is not too important, otherwise the forecasting task is less relevant, missing values in time series can be easily identified by considering that the most common interval between consecutive points corresponds to the sampling rate of the time series. Points not present at the expected intervals are then considered as missing. It is thus possible to identify missing values in time series, and then use padding or interpolation techniques to represent them. Padded or interpolated inputs should however not be treated as standard inputs as they are less reliable than other inputs. Furthermore, be it with padding or interpolation techniques, when the size of a gap, *i.e.* consecutive missing values, is important, the further away the missing value is from the observed values (the last one in case of padding, the last and next ones in case of interpolation), the less confident one is in the padded hidden states or interpolated values. It is thus desirable to decrease the importance of missing values according to their position in gaps.

We propose to do so by another modification of the attention mechanism that again reweighs the attention weights. We consider two reweighing schemes: a first one that penalizes missing values further away from the last observed value through a decaying function, *a priori* adapted to padding, and a second one that penalizes missing values depending on whether they are at the beginning, middle or end of a gap, *a priori* adapted to interpolation methods which rely on values before and after the gap.

The reweighing schemes corresponds to the following equations:

$$\omega(j) = \begin{cases} \exp(-\mu_j(j - j_{\text{last}})) \\ 1 + \boldsymbol{M}_{.j}^\top \mathbf{Pos}(j; \theta_1^g, \theta_2^g) \end{cases} \tag{9}$$

In the first scheme, j_{last} denotes the last observed value before j and $\boldsymbol{\mu} \in \mathbb{R}^T$ is an additional vector of parameters that is learned

with other parameters of the RNN. The exponential reweighing scheme is referred to as RNN-μ. $\omega(j)$ equals to 1 when the value at position j is available. When there is a missing value at position j, down-weighting is applied via $j - j_{last}$: the further in the gap it is located, the more it is down-weighted. $\boldsymbol{\mu}$ enables to differentiate the amount of down-weight according to where j lies in the history $[1, \ldots, T]$.

In the second scheme, RNN-M, each gap is divided into three regions of equal length, corresponding to the beginning, middle, and end, and defined through two hyper-parameters θ_1^g and θ_2^g. $\mathbf{Pos}(j; \theta_1^g, \theta_2^g)$ is a three-dimensional vector that is null if j is not missing; otherwise, denoting the length of the gap by $|g|$, if $\frac{j-j_{\text{last}}}{|g|} < \theta_1^g$, then the first coordinate of $\mathbf{Pos}(j; \theta_1^g, \theta_2^g)$ is set to 1 and the others to 0, if $\theta_1^g < \frac{j-j_{\text{last}}}{|g|} < \theta_2^g$, then the second coordinate is set to 1 and the others to 0 and if $\frac{j-j_{\text{last}}}{|g|} > \theta_2^g$, then the third coordinate is set to 1 and the others to 0.

$\boldsymbol{M} \in \mathbb{R}^{3\times T}$ is a matrix which is learned with other parameters of the RNN, where each of the coordinates of a column aims at reweighing the impact of missing values on the prediction task according to their position in gaps. As mentioned before, θ_1^g and θ_2^g are hyper-parameters that are set such that the gap g is divided into three equal parts (corresponding to the beginning, middle and end of the gap). Here, the down-weighting corresponding to the missing input at position j is a vector of dimension 3 that can learn different coefficients for the beginning, middle and end of the gap and is specific to position j.

In addition, the aforementioned position-aware down-weighting procedures handle issues with non-uniformly distributed missing values by initializing $\boldsymbol{\mu}$ and $\boldsymbol{M}$ to 0 (*i.e.*, all values are set to 0). Doing so, a missing value at a given position that has never been observed in the training set will be multiplied by 0 as it should not affect the weights.

One can extend the models proposed for handling periods, given in Section 2.3, with the proposed missing reweighing schemes. For this, $\mathbf{h}_j$ (Eq. (4)) is replaced by $(\omega(j)(\boldsymbol{\pi}_{i-j}\mathbf{h}_j))$ or $(\omega(j)(\boldsymbol{\Pi}_{\cdot(i-j)} \odot \mathbf{h}_j))$. Eqs. (10) summarize the equations of e_{ij} for all models (the names of the models are given at the end of each equation).

$$e_{ij} = \begin{cases} \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_{i-1} + \mathbf{U}_a\,\pi_{i-j}\,(\omega(j)\mathbf{h}_j))\,\mathbb{1}_{(i-j\le T)} \\ \qquad \text{with } \omega(j) = \exp(-\mu_j(j - j_{\text{last}})) \qquad (\boldsymbol{\pi}\text{-}\boldsymbol{\mu}) \\ \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_{i-1} + \mathbf{U}_a\,\pi_{i-j}\,(\omega(j)\mathbf{h}_j))\,\mathbb{1}_{(i-j\le T)} \\ \qquad \text{with } \omega(j) = 1 + \boldsymbol{M}_j^\top\mathbf{Pos}(j; \theta_1^g, \theta_2^g) \qquad (\boldsymbol{\pi}\text{-}\boldsymbol{M}) \\ \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_{i-1} + \mathbf{U}_a\,(\boldsymbol{\Pi}_{\cdot(i-j)} \odot (\omega(j)\mathbf{h}_j)))\,\mathbb{1}_{(i-j\le T)} \\ \qquad \text{with } \omega(j) = \exp(-\mu_j(j - j_{\text{last}})) \qquad (\boldsymbol{\Pi}\text{-}\boldsymbol{\mu}) \\ \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_{i-1} + \mathbf{U}_a\,(\boldsymbol{\Pi}_{\cdot(i-j)} \odot (\omega(j)\mathbf{h}_j)))\,\mathbb{1}_{(i-j\le T)} \\ \qquad \text{with } \omega(j) = 1 + \boldsymbol{M}_j^\top\mathbf{Pos}(j; \theta_1^g, \theta_2^g) \qquad (\boldsymbol{\Pi}\text{-}\boldsymbol{M}) \end{cases} \tag{10}$$

Note that the calculations of α_{ij} (Eq. (3)) and $\mathbf{c}_i$ (Eq. (2)) remain the same.

3. Multivariate extensions

As each variable in a K multivariate time series can have its own periods, a direct extension of the above approaches to multivariate time series is to consider that each variable $k, 1 \le k \le K$, has its own encoder and attention mechanism. We thus construct, for each variable k, context vectors on the basis of the following equation:

$$\alpha_{ij}^{(k)} = \frac{\exp\left(e_{ij}^{(k)}\right)}{\sum_{j'=1}^{T}\exp\left(e_{ij'}^{(k)}\right)} \tag{11}$$

where $e_{ij}^{(k)}$ is given by Eqs. (7), (8) and (10).

The context vector for the ith output of the kth variable is then defined by $\mathbf{c}_i^{(k)} = \sum_{j=1}^{T}\alpha_{ij}^{(k)}\mathbf{h}_j^{(k)}$, where $\mathbf{h}_j^{(k)}$ is the input hidden state at time stamp j for the kth variable and $\alpha_{ij}^{(k)}$ is the weights given by the attention mechanism of the kth variable.

Lastly, to predict the output while taking into account potential dependencies between different variables, one can simply concatenate the context vectors from the different variables into a single context vector $\mathbf{c}_i$ that is used as input of the decoder, the rest of the decoder architecture being unchanged:

$$\mathbf{c}_i = [\mathbf{c}_i^{(1)^\top} \cdots \mathbf{c}_i^{(K)^\top}]^\top$$

As each $\mathbf{c}_i^{(k)}$ is of dimension $2n$ (that is the dimension of the input hidden states), $\mathbf{c}_i$ is of dimension $2Kn$. This strategy can readily be applied to the original attention mechanism as well as the ones based on $\boldsymbol{\pi}$ and $\boldsymbol{\Pi}$ with and without handling missing values.

It is nevertheless possible to rely on a single attention model for all variables while having separate representations for them in order to select, for each output, specific hidden states from different variables. To do so, one can simply concatenate the hidden states of each variable into a single hidden state ($\mathbf{h}_j = [\mathbf{h}_j^{(1)^\top} \cdots \mathbf{h}_j^{(K)^\top}]^\top$) and deploy the previous attention model on top of them. This leads to the multivariate model which we refer to as RNN-Π^m and that is based on the same ingredients and equations as RNN-Π, the only difference being that one uses now a matrix, $\boldsymbol{\Pi^m}$, in $\mathbb{R}^{2Kn\times T}$:

$$e_{ij} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_{i-1} + \mathbf{U}_a(\boldsymbol{\Pi}^{\boldsymbol{m}}{}_{\cdot(i-j)} \odot \mathbf{h}_j))\,\mathbb{1}_{(i-j\le T)} \tag{12}$$

Handling missing values can directly be done by replacing $\mathbf{h}_j$ with $(\omega(j)\mathbf{h}_j)$, with $\omega(j)$ given by Eq. (9).

In the following section, we illustrate the behavior of our proposals on several univariate and multivariate datasets.

4. Experiments

4.1. Datasets and experimental setup

To assess the models proposed, we retained six widely used and publicly available datasets:

1. PSE: Polish Electricity (http://www.pse.pl).
2. PW: Polish Weather (https://globalweather.tamu.edu).
3. NAB: Numenta Anomaly Benchmark (https://numenta.com).
4. AQ: Air Quality (https://archive.ics.uci.edu/ml).
5. AEP: Appliances Energy Prediction (https://archive.ics.uci.edu/ml).
6. OLD: Ozone Level Detection (https://archive.ics.uci.edu/ml).

that are described in Table 2. The values for the history size were set so as they encompass the known periods of the datasets. They can also be tuned by cross-validation if one does not want to identify the potential periods by checking the autocorrelation curves. In general, the forecast horizon should reflect the nature of the data and the application one has in mind, with of course a trade off between long forecast horizon and prediction quality. For this purpose, the forecast horizons of these sets along the sampling rates are chosen as illustrated in Table 2. All datasets were split by retaining the first 75% of each dataset for training-validation and the last 25% for testing. For RNN-based methods, the training-validation sets were further divided by retaining the first 75% for training (56.25% of the data) and the last 25% for validation (18.75% of the data). For the baseline methods, we used 5-fold cross-validation on the training-validation sets to tune the

Table 2
Datasets.

Name	Usage	#Instances	History size	Forecast horizon	Sampling rate
PSE	Univariate	46379	96	4	2 h
PW	Univariate	4595	548	7	1 days
NAB	Univariate	18050	72	6	5 min
AQ	Univ./Multiv.	9471	192	6	1 h
AEP	Univ./Multiv.	19735	216	6	10 min
OLD	Univ./Multiv.	2536	548	7	1 day

hyperparameters. Lastly, linear interpolation was used whenever there are missing values in the time series[2].

We compared the methods introduced before, namely RNN-(π/Π/Π^m), with the original attention model (RNN-A) and several baseline methods, namely ARIMA, an ensemble learning method, Random Forests (RF), and support vector regression. Among these baselines, we retained ARIMA and RF as these were the two best performing methods in our datasets. These methods, discussed in Section 5, have also been shown to provide state-of-the-art results on various forecasting problems (*e.g.* [11]). Note that for ARIMA, we relied on the seasonal variant [12]. To implement the RNN models, we used theano[3] and Lasagne[4] on a Linux system with 256GB of memory and 32-core Intel Xeon @2.60 GHz. All parameters are regularized and learned through stochastic backpropagation (the mini-batch size was set to 64) with an adaptive learning rate for each parameter [13], the objective function being the Mean Square Error (MSE) on the output. For tuning the hyperparameters, we used a grid search over the learning rate, the regularization type and its coefficient, and the number of units in the LSTM and attention models. The values finally obtained are 10^{-3} for the initial learning rate and 10^{-4} for the coefficient of the regularization, the type of regularization selected being L_2. The number of units vary among the set {128, 256} for LSTMs and {256, 512} for the attention models respectively. We report hereafter the results with the minimum MSE on the test set. For evaluation, we use MSE and the symmetric mean absolute percentage error (SMAPE). MSE corresponds to the objective function used to learn the model. SMAPE presents the advantage of being bounded and represents a scaled L_1 error.

4.2. Handling periods

We first study the behavior of our proposed methods on univariate time series, prior to consider multivariate time series.

4.2.1. Univariate time series

For univariate experiments using multivariate time series, we chose the following variables from the datasets: for PW, we selected the *max temperature* series from the Warsaw metropolitan area that covers only one weather recording station; for AQ we selected *C6H6(GT)*; for AEP we selected the *outside humidity (RH6)*; for NAB we selected the *Amazon Web Services CPU usage* and for OLD we selected *T3*. Table 3 displays the results obtained with the MSE (left value) and the SMAPE (right value) as evaluation measures. Once again, one should note that MSE was the metric being optimized. For each time series, the best performance among all methods is shown in bold and other methods are marked with an asterisk if they are significantly worse than the best method according to a paired t-test with 5% significance level. Lastly, the last column of the tables corresponding to the results, *Selected model*, indicates which method, among RNN-π and RNN-Π, was selected as the best method on the validation set using MSE.

As one can note, except for AEP where the baselines are better than RNN-based methods, for all other datasets, the best results are obtained with RNN-π and RNN-Π, these results being furthermore significantly better than the ones obtained with RNN-A and baseline methods, for both MSE and SMAPE. The MSE improvement varies from one dataset to another: between 8% (PW) and 26% (NAB) w.r.t RNN-A. Compared to ARIMA, one can achieve an improvement ranging from 18% (15%), in OLD, to 94% (75%), in PSE, w.r.t MSE (SMAPE). In addition, the selected RNN-(π/Π) method (column *Selected model*) is the best performing method on three out of six datasets (AQ, PW, PSE) and the best performing RNN-π method on AEP. It is furthermore equivalent to the best performing method on OLD, the only dataset on which the selection fails being NAB (a failure means here that the selection does not select the best RNN-π method). However, on this dataset, the selected method is still better than the original attention model and the baselines. Overall, these results show that RNN-π and RNN-Π significantly improve forecasting in the univariate time series we considered, and that one can automatically select the best RNN-π method.

Lastly, to illustrate the ability of RNN-π methods to capture periods, we display in Fig. 2 the average attention weights for PW (right) and PSE (left) obtained with RNN-A and RNN-Π (averaged over all test examples and forecast horizon points). As one can see from the autocorrelation plot, displayed in Fig. 1, PSE has two main weekly and daily periods. This two periods are clearly visible in the attention weights of RNN-Π that gives higher weights to the four points located at positions *minus 7 days* and *minus 1 day* (these four points correspond to the four points of the forecast horizon). In contrast, the attention weights of the original attention model (RNN-A) follow a general increasing behavior with more weights on the more recent time stamps. This model thus misses the periods. One should note that although we display attention weights of RNN-Π in this figure, those of RNN-π are similar

4.2.2. Multivariate time series

As mentioned in Table 2, we furthermore conducted multivariate experiments on AQ, AEP and OLD using the multivariate extensions described in Section 2. For AQ, we selected the four variables associated to real sensors, namely C6H6(GT), NO2(GT), CO(GT) and NOx(GT) and predicted the same one as the univariate case (C6H6(GT)). For AEP, we selected two temperature time series, namely T1 and T6, and two humidity time series, RH6 and RH8, and we predict RH6 as in the univariate case. For OLD, we trained the model using T0 to T3 and predicted T3, as we did on the univariate case. As RF outperformed ARIMA on five out of six univariate datasets and was equivalent on the sixth one, we retained only RF and RNN-A for comparison with RNN-π, RNN-Π, and RNN-Π^m.

Table 4 shows the results of our experiments on multivariate sets with MSE (SMAPE, not displayed here for readability

[2] We compared several methods for missing values, namely linear, non-linear spline and kernel based Fourier transform interpolation as well as padding for the RNN-based models. The best reconstruction was obtained with linear interpolation, hence its choice here.

[3] http://deeplearning.net/software/theano/.

[4] https://lasagne.readthedocs.io.

Table 3
Overall results for univariate case with MSE (left value) and SMAPE (right value).

Dataset	RNN-A	RNN-π	RNN-Π	ARIMA	RF	*Selected model*
AQ	0.282*(0.694)	0.257(0.661)	**0.25**(0.669)	0.546*(0.962)	0.299*(0.762)	Π
OLD	0.319*(0.595)	**0.271**(0.523)	0.275(0.586)	0.331*(0.619)	0.305*(0.606)	Π
AEP	0.025*(0.085)	0.029*(0.101)	0.027*(0.095)	**0.021**(0.066)	0.021(0.085)	Π
NAB	0.642*(0.442)	**0.475**(0.323)	0.54*(0.369)	1.677*(1.31)	0.779*(0.608)	Π
PW	0.166*(0.558)	**0.152**(0.547)	0.162*(0.565)	0.213*(0.61)	0.156(0.544)	π
PSE	0.034*(0.282)	**0.032**(0.264)	0.033*(0.256)	0.623*(1.006)	0.053*(0.318)	π

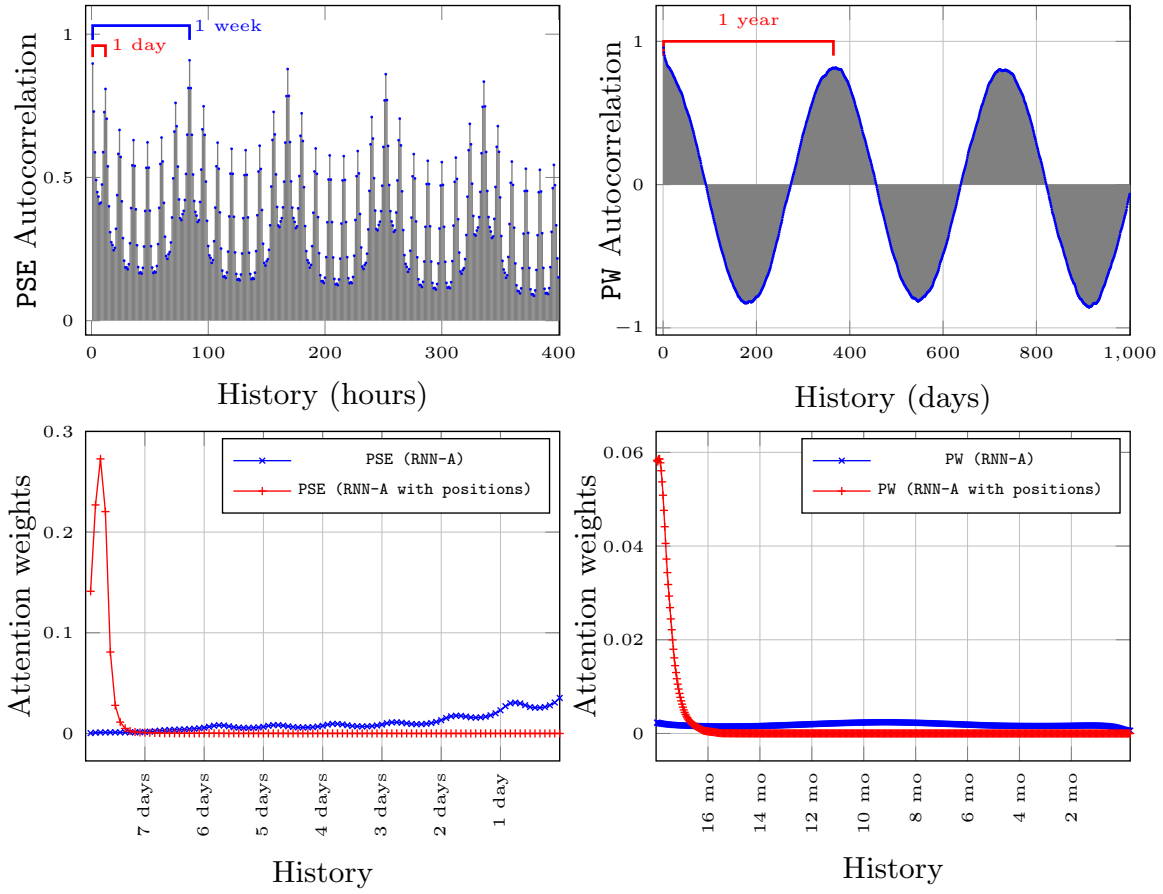

Fig. 1. Top: autocorrelation function illustrating the periods on PSE and PW. Bottom: weights of the attention mechanism for the RNNs without and with time stamp (or position) information on both PSE and PW (the weights are averaged over all test examples, see Section 4).

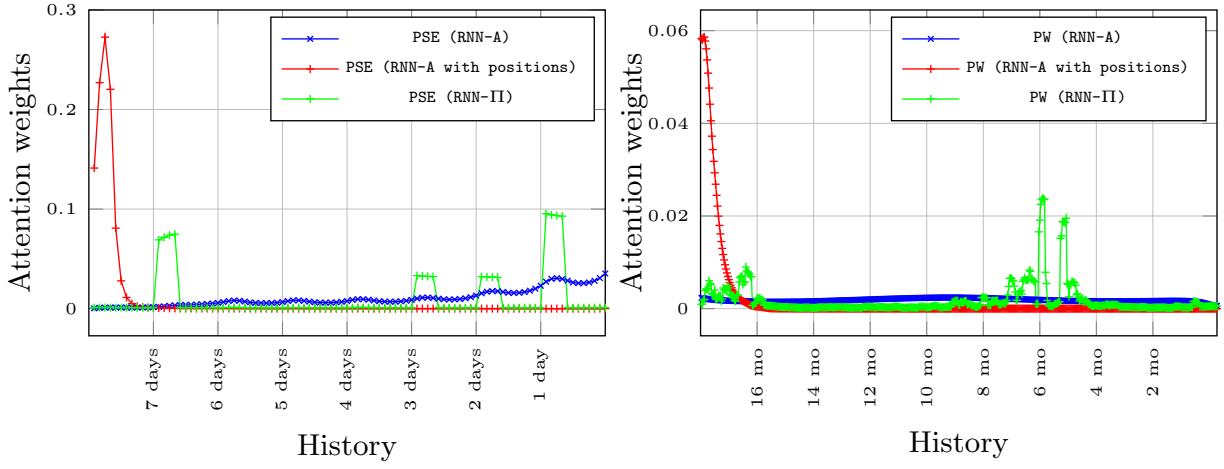

Fig. 2. Attention weights for PW (right) and PSE (left) for RNN-A, RNN-A with position and RNN-π. See Fig. 1 for the autocorrelation plots of these datasets.

Table 4
Overall results for multivariate case with MSE.

Dataset	RNN-A	RNN-π	RNN-Π	RNN-Π^m	RF	*Selected model*
AQ	0.352*	0.276*	**0.268**	0.3*	0.45*	Π^m
OLD	0.336*	0.328*	0.327*	**0.274**	0.315*	Π^m
AEP	0.029*	**0.024**	0.036*	0.026*	0.027*	Π^m

reasons, has a similar behavior). As before, for each time series, the best result is in bold and an asterisk indicates that the method is significantly worse than the best method (again according to a paired t-test with 5% significance level). Similarly to the univariate case, the best results, that are always significantly better than the other results, are obtained with the RNN-π methods: for AQ and OLD datasets, RNN-π can respectively bring 24% and 18% of significant improvement over RNN-A. Similarly, for AEP, the improvement is significant over RNN-A (17% with RNN-π). Compared to RF, one can obtain between 11% (AEP) and 40% (AQ) of improvement. As one can note, the selected method is always RNN-Π^m. The selection is this time not as good as for the univariate case as the best method (sometimes significantly better than the one selected) is missed. That said, RNN-Π^m remains better than the state-of-the-art baselines retained, RNN-A and RF.

4.3. Handling missing values

In order to assess whether the proposed methods are robust to missing values, we introduced different levels of missing values in the datasets with the following strategy: for each dataset, we added 5, 10, 15, 20, 30 and 40 percent of missing values. As the datasets should contain both random missing values and gaps, we introduced half gaps and half random missing values. For instance, to add 10% of missing values, we first introduce 5% of gaps and then 5% of random missing values that do not have any interference with the previously introduced gaps.

Furthermore, as the gaps could be of different sizes, we let the length of gaps vary from 5 to 100, picked at random. Note that for AQ, which already contains 18% of missing values, we first introduced new missing values until we reach half of the desired percentage, then we introduce the gaps. In our experiments, the percentage of introduced missing values are shown as a number along with the dataset name; for the original dataset we just mention the dataset name.

As the RNN-based models outperformed the others in our previous experiments, we focus on them here. Furthermore, in order to assess the impact of handling missing values, we report the results obtained with the standard RNN-A method, with the period based models, RNN-π/Π, and with the complete models integrating both periods and weighting schemes (RNN-$\pi/\Pi-\mu/M$). The results obtained are displayed in Table 5. The column *Selected-*π/Π indicates which model for handling periods performs best on the validation set, the column *Selected-*$\pi/\Pi-\mu/M$ which model, handling both periods and missing values, performs best on the validation set, and the column *Selected model* which model, among RNN-A, RNN-π/Π and RNN-$\pi/\Pi-\mu/M$ performs best on the validation set. The MSE/SMAPE scores are computed on the test set.

First, as one can note from the first two *selected* columns of Table 5, the different proposals introduced here are important and help to improve the results for time series forecasting. Indeed, from the methods for handling periods, RNN-π is selected 23 times (out of 33 cases) while RNN-Π is selected 10 times. Among the methods for handling both periods and missing values (column *Selected-*$\pi/\Pi-\mu/M$), RNN-$\pi-\mu$ is selected 11 times (33%), RNN-$\pi-M$ 8 times (24%), RNN-$\Pi-\mu$ 4 times (12%), and RNN-$\pi-M$ 9 times (27%). If RNN-$\pi-\mu$ is the preferred method for handling both periods and missing values for PSE and RNN-$\pi-M$ is the preferred one for OLD, there are no clear preferences for handling both periods and missing values for PW, AEP, and NAB.

We first analyze the forecasting performances by focusing on each dataset separately and later present overall performances. For PSE dataset, from 5% to 40% missing, RNN-$\pi/\Pi-\mu/M$ outperforms RNN-A significantly in all six cases and RNN-π/Π significantly in five cases. In addition, the *selected* models are concordant with the best performing method in all six settings. Similarly, for PW RNN-$\pi/\Pi-\mu/M$ and RNN-π/Π are the best performing models and are often significantly better than RNN-A. For PW, in four out of six cases, *i.e.* for 15%, 20%, 30%, and 40% of missing values, RNN-$\pi/\Pi-\mu/M$ outperforms RNN-π/Π and, except for PW5, it outperforms RNN-A. Furthermore, the selected model mostly agrees with the best performing method. For AQ, RNN-π/Π and RNN-$\pi/\Pi-\mu/M$ are the best performing ones in all three missing settings (original; up to 20 %, 30%, and 40%) and the best performing one always significantly outperforms RNN-A. RNN-$\pi/\Pi-\mu/M$ significantly outperforms RNN-π/Π for AQ40, and RNN-A for AQ, AQ30, and AQ40. Lastly, RNN-π/Π is the best predicting method for AQ and AQ30. RNN-$\pi/\Pi-\mu/M$ is overall the best method for AEP (in particular for AEP5, AEP15 and AEP20), followed by RNN-A and RNN-π/Π. A similar behavior is observed for OLD where RNN-$\pi/\Pi-\mu/M$ is the best performing method for OLD5, OLD15 and OLD30, followed by RNN-A and RNN-π/Π. For AEP and OLD, the selected model does not always coincide with the best performing method. We conjecture here that additional data is needed to really assess which model is the best one on the validation set[5]. Lastly, RNN-$\pi/\Pi-\mu/M$ is the best performing method on NAB with 10%, 15%, 30%, and 40% of missing values whereas RNN-π/Π performs the best on 5% of missing values. Moreover, the selected model mostly agrees with the best performing method (in five out of six cases).

For multivariate prediction with missing data we focused to RNN-Π^m and RNN-$\Pi^m-\mu/M$ methods, since Section 4.2.2 showed that RNN-Π^m was the main selected method in a multivariate setting. Table 6 illustrates the MSE values of multivariate prediction on the same multivariate datasets as the ones used in Section 4.2.2, with 5%, 10%, 15%, 20%, 30%, and 40% of missing values. For AQ, RNN-$\Pi^m-\mu/M$ is the best performing method in two cases out of three, followed by RNN-A. In all three cases, RNN-Π^m-M is chosen as the RNN-$\Pi^m-\mu/M$ method. For AEP, RNN-Π^m with 10%, 15%, and 40% and RNN-A with 5%, 20%, and 30% outperform the other methods. For AEP, RNN-$\Pi^m-\mu$ is chosen in all six cases as the best RNN-$\Pi^m-\mu/M$ method. For OLD, RNN-Π^m with 5%, 15%, and 30%, and RNN-$\Pi^m-\mu/M$ with 10%,20%, and 40% are the best performing methods, yielding results significantly better than the ones of RNN-A. Here, the best performing method and the selected model do not agree in most cases.

Overall, the results obtained are better than the ones obtained in the univariate case. Furthermore, in 73% of the cases, RNN-A is outperformed by RNN-Π^m or RNN-$\Pi^m-\mu/M$. As before, the complexity of the datasets may explain that it is difficult to select the best model on the validation set. In particular, in multivariate prediction, the missing portions of the dataset do not necessarily occur at the same time for the different variables, leading to potentially a high proportion of time stamps where at least one value is missing.

[5] This is especially true for OLD that has few instances and a relatively long history and forecast horizon.

Table 5
Univariate prediction on datasets with missing values and gaps, MSE(SMAPE).

Dataset	Selected-π/Π	Selected-$\pi/\Pi - \mu/M$	RNN-A	RNN-π/Π	RNN-$\pi/\Pi - \mu/M$	*Selected model*
PSE5	π	$\Pi - \mu$	0.064*(0.345)	0.059*(0.316)	**0.055**(0.31)	$\Pi - \mu$
PSE10	π	$\pi - \mu$	0.066*(0.353)	0.055*(0.318)	**0.052**(0.314)	$\pi - \mu$
PSE15	Π	$\pi - \mu$	0.09*(0.36)	0.083(0.357)	**0.081**(0.332)	$\pi - \mu$
PSE20	π	$\pi - \mu$	0.079*(0.374)	0.078*(0.369)	**0.074**(0.344)	$\pi - \mu$
PSE30	π	$\pi - \mu$	0.104*(0.419)	0.102*(0.386)	**0.098**(0.384)	$\pi - \mu$
PSE40	π	$\Pi - M$	0.113*(0.428)	0.119*(0.411)	**0.106**(0.393)	$\Pi - M$
PW5	π	$\pi - M$	0.161(0.556)	**0.157**(0.555)	0.164*(0.566)	π
PW10	π	$\pi - M$	0.16*(0.557)	**0.153**(0.54)	0.155(0.542)	π
PW15	π	$\pi - \mu$	0.167(0.569)	0.167(0.556)	**0.163**(0.55)	$\pi - \mu$
PW20	π	$\Pi - M$	0.209*(0.6)	0.182(0.551)	**0.177**(0.553)	$\Pi - M$
PW30	π	$\pi - M$	0.177*(0.571)	0.163(0.556)	**0.161**(0.549)	$\pi - M$
PW40	π	$\pi - \mu$	0.181*(0.568)	0.172(0.564)	**0.164**(0.534)	π
AQ	Π	$\pi - \mu$	0.299*(0.74)	**0.249**(0.668)	0.262*(0.665)	$\pi - \mu$
AQ30	π	$\Pi - \mu$	0.457*(0.879)	**0.35**(0.768)	0.357(0.797)	$\Pi - \mu$
AQ40	π	$\pi - \mu$	0.45*(0.881)	0.435*(0.819)	**0.403**(0.853)	$\pi - \mu$
AEP5	π	$\Pi - M$	0.028*(0.095)	0.027(0.093)	**0.027**(0.096)	$\Pi - M$
AEP10	π	$\pi - M$	0.038*(0.128)	**0.034**(0.111)	0.038*(0.114)	$\pi - M$
AEP15	Π	$\pi - M$	0.042*(0.14)	0.044*(0.118)	**0.04**(0.123)	A
AEP20	π	$\Pi - \mu$	0.036*(0.111)	0.035*(0.11)	**0.035**(0.105)	A
AEP30	π	$\pi - \mu$	**0.052**(0.119)	0.057*(0.134)	0.06*(0.147)	A
AEP40	π	$\pi - \mu$	**0.055**(0.124)	0.06*(0.138)	0.08*(0.16)	$\pi - \mu$
OLD5	π	$\Pi - M$	0.333*(0.534)	0.32(0.582)	**0.282**(0.625)	π
OLD10	Π	$\Pi - M$	0.457*(0.761)	**0.321**(0.621)	0.332(0.707)	Π
OLD15	Π	$\Pi - M$	0.305(0.565)	0.36*(0.611)	**0.288**(0.6)	Π
OLD20	Π	$\Pi - M$	**0.297**(0.521)	0.358*(0.657)	0.484*(0.826)	Π
OLD30	Π	$\Pi - M$	0.37(0.581)	0.461*(0.836)	**0.358**(0.71)	Π
OLD40	Π	$\Pi - M$	**0.348**(0.648)	0.38(0.614)	0.362(0.676)	$\Pi - M$
NAB5	π	$\Pi - \mu$	0.739*(0.523)	**0.702**(0.484)	0.973*(0.536)	$\Pi - \mu$
NAB10	π	$\pi - M$	0.758*(0.512)	0.767*(0.467)	**0.712**(0.465)	$\pi - M$
NAB15	π	$\pi - M$	0.95*(0.556)	1.014*(0.588)	**0.727**(0.47)	$\pi - M$
NAB20	Π	$\pi - \mu$	**1.152**(0.656)	1.324*(0.631)	1.364*(0.639)	A
NAB30	Π	$\pi - M$	1.309*(0.69)	1.1*(0.609)	**0.952**(0.551)	$\pi - M$
NAB40	π	$\pi - \mu$	1.287*(0.695)	1.44*(0.684)	**1.012**(0.547)	$\pi - \mu$

Table 6
Multivariate prediction on datasets with missing values and gaps, MSE.

Dataset	RNN-A	RNN-Π^m	RNN-$\Pi^m - \mu/M$	*Selected model*
AQ	0.352*	0.34*	**0.314**	Π^m
AQ30	**0.488**	0.646*	0.539*	$\Pi^m - \mu/M$
AQ40	0.825*	0.773*	**0.503**	$\Pi^m - \mu/M$
AEP5	**0.051**	0.089*	0.062*	$\Pi^m - \mu/M$
AEP10	0.09*	**0.067**	0.072*	$\Pi^m - \mu/M$
AEP15	0.084*	**0.065**	0.085*	A
AEP20	**0.062**	0.068*	0.081*	A
AEP30	**0.09**	0.1*	0.116*	A
AEP40	0.154*	**0.132**	0.166*	$\Pi^m - \mu/M$
OLD5	0.379*	0.366*	**0.332**	$\Pi^m - \mu/M$
OLD10	0.386	**0.343**	0.381*	$\Pi^m - \mu/M$
OLD15	0.442*	0.333	**0.327**	A
OLD20	0.437	**0.402**	0.661*	Π^m
OLD30	0.359*	0.419*	**0.308**	$\Pi^m - \mu/M$
OLD40	0.373*	**0.349**	0.398*	$\Pi^m - \mu/M$

5. Related work

The notion of stochasticity of time series modeling and prediction was introduced long back [14]. Since then, various stochastic models have been developed, notable among these are autoregressive (AR) [15] and moving averages (MA) [16] methods. These two models were combined in a more general and more effective framework, known as autoregressive moving average (ARMA), or autoregressive integrated moving average (ARIMA) when the differencing is included in the modeling [17]. Vector ARIMA or VARIMA [18] is the multivariate extension of the univariate ARIMA models where each time series instance is represented using a vector.

Neural networks are regarded as a promising tool for time series prediction [19,20] due to their data-driven and self-adaptive nature, their ability to approximate any continuous function and their inherent non-linearity. The idea of using neural networks for prediction dates back to 1964 where an adaptive linear network was used for weather forecasting [21]. But the research was quite limited due to the lack of a training algorithm for general multilayer networks at the time. Since the introduction of backpropagation algorithm [22], there had been much development in the use of neural networks for forecasting. It was shown in a simulated study that neural networks can be used for modeling and forecasting nonlinear time series [23]. Several studies [24,25] have found that neural networks are able to outperform the traditional stochastic models such as ARIMA or Box–Jenkins approaches [17].

With the advent of SVM, it has been used to formulate time series prediction as a regression estimation using Vapnik's insensitive loss function and Huber's loss function [26]. The authors showed that SVM based prediction outperforms traditional neural network based methods. Since then different versions of SVMs are applied for time series prediction and many different SVM forecasting algorithms have been derived [27,28]. More recently, random forests [29] are used for time series predictions due to their performance accuracy and ease of execution. Random forest regression is used for prediction in the field of finance [30] and bioinformatics [31], and are shown to outperform ARIMA [32].

Traditional neural networks allow only feedforward connections among the neurons of a layer and the neurons in the following layer. In contrast, *recurrent neural networks* (RNN) [33,34] allow both forward and feedback or recurrent connections between the neurons of different layers. Hence, RNNs are able to incorporate contextual information from past inputs which makes them an attractive choice for predicting general sequence-to-sequence data, including time series [3]. In this present study we use RNNs for modeling time series. Early work [35] has shown that RNNs (a) are a type of nonlinear autoregressive moving average (NARMA) model and (b) outperform feedforward networks and various types of

linear statistical models on time series. Subsequently, various RNN-based models were developed for different time series, as noisy foreign exchange rate prediction [36], chaotic time series prediction in communication engineering [37] or stock price prediction [38]. A detailed review can be found in [39] on the applications of RNNs along with other deep learning based approaches for different time series prediction tasks.

RNNs based on LSTMs [40], which we consider in our study, alleviate the *vanishing gradient* problem of the traditional RNNs proposed. They have furthermore been shown to outperform traditional RNNs on various temporal tasks [8,41]. More recently, they have been used for predicting the next frame in a video and for interpolating intermediate frames [42], for forecasting the future rainfall intensity in a region [43], or for modeling clinical data consisting of multivariate time series of observations [44].

Adding attention mechanism on the decoder side of an encoder-decoder RNN framework enabled the network to focus on the interesting parts of the encoded sequence [7]. Attention mechanism is used in many applications such as image description [45], image generation [46], phoneme recognition [47], heart failure prediction [48], as well as time series prediction [49] and classification [48]. Many studies also apply attention mechanism on external memory [50,51].

The previous work [49] uses attention mechanism to determine the importance of a factor among other factors that affect time series. However, we use extended attention mechanism to model period and emphasize the interesting parts of input sequence with missing values. Our approach is applicable for both univariate and multivariate time series prediction.

The attention mechanism has been specialized in [52], under the name pointer network, so as to select the best input to be reused as the output. This model would be perfect for noise-free, truly periodic times series. In practice, however, times series are noisy and if the output is highly correlated to the input corresponding to the period, it is not an exact copy of it.

A recent study [53] uses relative position information respect to object and subject for relation extraction task, more specifically slot filling. They add position information besides the original sequence encoding, whereas we propose a model adjust the sequence encoding according to period and missing values.

However, as far we know, no work has focused on analyzing the adequacy of RNNs (based or not on LSTMs) for time series, in particular w.r.t. their ability to model periods and handle missing values (for this latter case, some methods based on *e.g.* Gaussian processes have been proposed to handle irregularly sampled data and missing values [54], but none related to RNNs). To our knowledge, this study is the first one to address this problem.

6. Conclusion

We studied in this paper the use of sequence-to-sequence RNNs, in particular the state-of-the-art bidirectional LSTM encoder-decoder with a period-aware content attention model, for modeling and forecasting time series. If content attention models are crucial for this task, they were not designed for time series and currently are deficient as they do not capture periods. We thus proposed three extensions of the content attention model making use of the (relative) positions in the input and output sequences. These models are combined with novel models and architectures to efficiently handle missing values and gaps in time series. The experiments we conducted over several univariate and multivariate time series demonstrate the effectiveness of these extensions, on time series with either clear periods, as PSE, or less clear ones, as AEP. Indeed, these extensions perform significantly better than the original attention model as well as state-of-the-art baseline methods based on ARIMA and random forests.

In the future, we plan on studying formal criteria to select the best extension for both univariate and multivariate time series. This would allow one to avoid using a validation set that may be not large enough to properly select the best method.

Acknowledgments

This work was partially supported by the grants ANR-11-LABX-0025-01.3, FUI SSC (Smart Support Center) and the Grenoble Alpes Data Institute, supported by the French National Research Agency under the 305 Investissements d'avenir program (ANR-15-IDEX-02).

References

[1] J.G. De Gooijer, R.J. Hyndman, 25 years of time series forecasting, Int. J. Forecast. 22 (3) (2006) 443–473.
[2] G. Bontempi, S.B. Taieb, Y.-A. Le Borgne, Machine learning strategies for time series forecasting, in: Business Intelligence, Springer, 2013, pp. 62–77.
[3] A. Graves, Generating sequences with recurrentneural networks, *CoRR* 2013. abs/1308.0850.
[4] J. Weston, S. Chopra, A. Bordes, Memory networks, CoRR (2014). abs/1410.3916
[5] J. Weston, A. Bordes, S. Chopra, T. Mikolov, Towards AI-Completequestion answering: a set of prerequisite toy tasks, CoRR (2015). abs/1502.05698
[6] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwinska, S.G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A.P. Badia, K.M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, D. Hassabis, Hybrid computing using a neural network with dynamic external memory, Nature 538 (7626) (2016) 471–476.
[7] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, *CoRR* 2014. abs/1409.0473.
[8] F.A. Gers, N.N. Schraudolph, J. Schmidhuber, Learning precise timing with LSTM recurrent networks, J. Mach. Learn. Res. 3 (Aug) (2002) 115–143.
[9] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[10] C. Chatfield, The Analysis of Time Series: An Introduction, CRC press, 2016.
[11] M.J. Kane, N. Price, M. Scotch, P. Rabinowitz, Comparison of arima and random forest time series models for prediction of avian influenza H5N1 outbreaks, BMC Bioinf. 15 (1) (2014) 276.
[12] R. Hyndman, Y. Khandakar, Automatic time series forecasting: the forecast package for R, J. Stat. Softw. 27 (3) (2008) 1–22, doi:10.18637/jss.v027.i03. URL https://www.jstatsoft.org/v027/i03
[13] D. Kingma, J. Ba, Adam: a method for stochastic optimization, *CoRR* 2014. abs/1412.6980.
[14] G.U. Yule, On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers, Philosoph. Trans. R. Soc. Lond. Ser. A Contain. Pap. Math. Phys. Charact. 226 (1927) 267–298.
[15] G. Walker, On periodicity in series of related terms, Philosoph. Trans. R. Soc. Lond. Ser. A Contain. Pap. Math. Phys. Charact. 131 (818) (1931) 518–532.
[16] E. Slutzky, The summation of random causes as the source of cyclic processes, Econom. J. Econom. Soc. 5 (2) (1937) 105–146.
[17] G.E. Box, G.M. Jenkins, Some recent advances in forecasting and control, J. R. Stat. Soc. Ser. C Appl. Stat. 17 (2) (1968) 91–109.
[18] G.C. Tiao, G.E. Box, Modeling multiple time series with applications, J. Am. Stat. Assoc. 76 (376) (1981) 802–816.
[19] G.P. Zhang, An investigation of neural networks for linear time-series forecasting, Comput. Oper. Res. 28 (12) (2001) 1183–1202.
[20] S.F. Crone, M. Hibon, K. Nikolopoulos, Advances in forecasting with neural networks? Empirical evidence from the nn3 competition on time series prediction, Int. J. Forecast. 27 (3) (2011) 635–660.
[21] M.J.-C. Hu, Application of the adaline system to weather forecasting, Department of Electrical Engineering, Stanford University,g, 1964.
[22] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Cognit. model. 5 (3) (1988) 1.
[23] A. Laepes, R. Farben, Nonlinear Signal Processing using Neural Networks: Prediction and System Modelling, Technical Report, Los Alamos National Laboratory, Los Alamos, NM, 1987.
[24] P.J. Werbos, Generalization of backpropagation with application to a recurrent gas market model, Neural Netw. 1 (4) (1988) 339–356.
[25] R. Sharda, R. Patil, Neural networks as forecasting experts: an empirical test, in: Proceedings of the International Joint Conference on Neural Networks, vol. 2, IEEE, 1990, pp. 491–494.
[26] K.-R. Müller, A.J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, V. Vapnik, Predicting time series with support vector machines, in: Proceedings of the International Conference on Artificial Neural Networks, Springer, 1997, pp. 999–1004.
[27] T. Raicharoen, C. Lursinsap, P. Sanguanbhoki, Application of critical support vector machine to time series prediction. circuits and systems., in: Proceedings of the 2003 International Symposium on ISCAS, vol. 5, 2003, pp. 25–28.
[28] Y. Fan, P. Li, Z. Song, Dynamic least squares support vector machine, in: Proceedings of the Sixth World Congress on Intelligent Control and Automation (WCICA), vol. 1, IEEE, 2006, pp. 4886–4889.

[29] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[30] G.G. Creamer, Y. Freund, Predicting performance and quantifying corporate governance risk for latin American ADRS and banks, Financial Engineering and Applications, MIT, Cambridge (2004).
[31] A. Kusiak, A. Verma, X. Wei, A data-mining approach to predict influent quality, Environ. Monit. Assess. 185 (3) (2013) 2197–2210.
[32] M.J. Kane, N. Price, M. Scotch, P. Rabinowitz, Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks, BMC Bioinf. 15 (1) (2014) 276.
[33] M.I. Jordan, Serial Order: A Parallel Distributed Processing Approach, Technical Report 8604, Institute for Cognitive Science, University of California, 1986.
[34] J.L. Elman, Finding structure in time, Cognit. Sci. 14 (2) (1990) 179–211.
[35] J. Connor, L.E. Atlas, D.R. Martin, Recurrent networks and NARMA modeling, in: Proceedings of the NIPS, 1991, pp. 301–308.
[36] C.L. Giles, S. Lawrence, A.C. Tsoi, Noisy time series prediction using recurrent neural networks and grammatical inference, Mach. Learn. 44 (1–2) (2001) 161–183.
[37] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80.
[38] T.-J. Hsieh, H.-F. Hsiao, W.-C. Yeh, Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm, Appl. Soft Comput. 11 (2) (2011) 2510–2525.
[39] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognit. Lett. 42 (2014) 11–24.
[40] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[41] F.A. Gers, D. Eck, J. Schmidhuber, Applying LSTM to time series predictable through time-window approaches, in: Proceedings of the International Conference on Artificial Neural Networks, Springer, 2001, pp. 669–676.
[42] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, S. Chopra, Video (language) modeling: a baseline for generative models of natural videos, *CoRR* 2014. abs/1412.6604.
[43] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional LSTM network: a machine learning approach for precipitation nowcasting, in: Advances in Neural Information Processing Systems, 2015, pp. 802–810.
[44] Z.C. Lipton, D.C. Kale, C. Elkan, R. Wetzell, Learning to diagnose with LSTM recurrent neural networks, *CoRR* 2015. abs/1511.03677.
[45] K. Xu, J. Ba, R. Kiros, K. Cho, A.C. Courville, R. Salakhutdinov, R.S. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention., in: Proceedings of the ICML, vol. 14, 2015, pp. 77–81.
[46] G. Karol, I. Danihelka, A. Graves, D. Rezende, D. Wierstra, Draw: a recurrent neural network for image generation, in: Proceedings of the ICML, 2015.
[47] J.K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in: Advances in Neural Information Processing Systems, 2015, pp. 577–585.
[48] E. Choi, M.T. Bahadori, J. Sun, J. Kulas, A. Schuetz, W. Stewart, RETAIN: an interpretable predictive model for healthcare using reverse time attention mechanism, in: Proceedings of the NIPS, 2016, pp. 3504–3512.
[49] M. Riemer, A. Vempaty, F.P. Calmon, F.F. Heath III, R. Hull, E. Khabiri, Correcting forecasts with multifactor neural attention, in: Proceedings of the 33rd International Conference on Machine Learning, 2016, pp. 3010–3019.
[50] A. Graves, G. Wayne, I. Danihelka, Neural turing machines, *CoRR* 2014.abs/1410.5401.
[51] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S.G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al., Hybrid computing using a neural network with dynamic external memory, Nature 538 (7626) (2016) 471–476.
[52] O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, in: Proceedings of the NIPS, 2015, pp. 2692–2700.
[53] Y. Zhang, V. Zhong, D. Chen, G. Angeli, C.D. Manning, Position-aware attention and supervised data improve slot filling, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, Copenhagen, Denmark, 2017, pp. 35–45. URL http://aclanthology.info/papers/D17-1004/d17-1004
[54] M. Ghassemi, M.A. Pimentel, T. Naumann, T. Brennan, D.A. Clifton, P. Szolovits, M. Feng, A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data, in: Proceedings of the AAAI...Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 2015, NIH Public Access, 2015, p. 446.

Yagmur Gizem Cinar is a Ph.D. researcher at Laboratoire de Informatique Grenoble (Computer Science Lab), Univ. Grenoble Alpes, Grenoble, France. She received M.Sc. degrees in Artificial Intelligence and Electrical Engineering at KU Leuven in 2015 and 2014, respectively. Her research interests are information retrieval, machine learning and signal processing.

Hamid Mirisaee received his Ph.D. degree on Machine Learning from the University of Grenoble Alpes (UGA) in 2015. He then joined the University of Pierre and Marie Curie as a postdoc, followed by a second postdoc in UGA. His main domains of research includes matrix decomposition techniques, recommendation systems and natural language processing.

Parantapa Goswami received his Ph.D. from Université Grenoble Alpes on 2014 for his thesis on learning information retrieval models and parameters using machine learning techniques. He worked as a postdoctoral researcher at Laboratoire d'Informatique de Grenoble of Université Grenoble Alpes, focusing on temporal data prediction using deep learning. Presently, he is a researcher at Viseo Innovation, studying classical machine learning, deep learning and their applications to temporal data analysis, information retrieval and natural language processing.

Prof. Eric Gaussier is known for his work on the intersection of Artificial Intelligence (AI) and Data Science (DS), in particular for his contributions on models and algorithms to extract information, insights and knowledge from data in various forms. He has worked on three main subfields of AI and DS: machine learning, information retrieval and computational linguistics. He is also interested in modeling how (textual) information is shared in social (content) networks, and how such networks evolve over time. More recently, he has also been working on improving job scheduling techniques through machine learning, and in learning representations for different types of sequences, as texts and time series.

Ali Ait-Bachir received his Ph.D. in Computer Science at Univ. Grenoble Alpes in 2008. He is currently working as a JEE developer and architect at Coservit. He has more than 8 years of experience in software engineering, mainly in the development of applications focused on JEE Java technologies. He is working in Big Data and Data Science fields. He worked in a collaborative research project with LIG (Laboratoire Informatique de Grenoble) on time series and deep learning.