Position-based Content Attention for Time Series Forecasting with Sequence-to-sequence RNNs

Yagmur Cinar*, Hamid Mirisaee*, Parantapa Goswami+, Eric Gaussier*, Ali Aït-Bachir†, and Vadim Strijov‡

* Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG
+ Viseo R&D, † Coservit, ‡ CCRAS
*firstname.lastname@imag.fr, ‡ Strijov@ccas.ru
+parantapa.goswami@viseo.com, †a.ait-bachir@coservit.com

Abstract. We propose here an extended attention model for sequence-to-sequence recurrent neural networks (RNNs) designed to capture (pseudo-)periods in time series. 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.

Keywords: Recurrent neural networks, attention model, time series

1 Introduction

Predicting 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. 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 times series. In a forecasting scenario, the pseudo-periods correspond to the difference between the positions of the output being predicted and specific inputs. Pseudo-periods may be due to seasonality or to the patterns underlying the activities measured.

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 modelling sequences is based on sequence-to-sequence Recurrent Neural Networks (RNNs) [3], hereafter referred to as Seq-RNNs. In order to capture pseudo-periods in Seq-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 pseudo-periods underlying the time series, longer-term memories that store information pertaining to past input sequences (as described in *e.g.* [4, 5, 6]) are not required. A particular model of interest here is the content attention model proposed in [7] and described in Section 3. 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 does not directly capture position-based pseudo-periods in time series. It has nevertheless been specialized in [8], 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 pseudo-period, it is not an exact copy of it. We propose in this paper extensions of the attention model that capture pseudo-periods and lead to state-of-the-art methods for time series forecasting.

The remainder of the paper is organised as follows: Section 2 discusses the related work. Section 3 presents the position-based content attention models for both univariate and multivariate time series. Experiments illustrating the behaviour of the proposed models are described in Section 4. Lastly, Section 5 concludes the paper.

2 Related Work

Various stochastic models have been developed for time series modeling and forecasting. Notable among these are autoregressive (AR) [9] and moving averages (MA) [10] models, that were combined in a more general and effective framework, known as autoregressive moving average (ARMA), or autoregressive integrated moving average (ARIMA) when the differencing is included in the model [11]. Vector ARIMA, or VARIMA [12], is the multivariate extension of the univariate ARIMA model. More recently, based on the development of statistical machine learning, time series prediction has been formulated as a regression problem typically solved with Support Vector Machines (SVM) [13] and, even more recently, with Random Forests (RF) [14, 15]. RF have been in particular used for prediction in the field of finance [14] and bioinformatics [15], and have been shown to outperform ARIMA in different cases [16].

In this study, RNNs are used for modeling time series as they incorporate contextual information from past inputs and are thus an attractive choice for predicting sequence data, including time series [3]. Early work [17] has shown that RNNs (a) are a type of nonlinear autoregressive moving average (NARMA) model and (b) outperform feed-forward 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 [18], chaotic time series prediction in communication engineering [19] or stock price prediction [20]. A detailed review can be found in [21] for different time series prediction tasks.

RNNs based on LSTMs [22], that we consider here, alleviate the *vanishing gradient* problem of the traditional RNNs. They have furthermore been shown to outperform traditional RNNs on various temporal tasks [23, 24]. Recently, they have been used for predicting the next frame in a video and for interpolating intermediate frames [25], for forecasting the future rainfall intensity in a region [26], or for modeling clinical data of multivariate time series [27]. The attention model in Seq-RNNs [3, 7] has been studied very recently for time series prediction [28] and classification [29]. In particular, the study in [28] uses the attention to determine the importance of a factor for prediction.

None of the previous studies, to the best of our knowledge, investigated the possibility to capture pseudo-periods in time series via the attention model. This is precisely the focus of the present study, that introduces generalizations of the content based attention model to capture pseudo-periods and improve forecasting in time series.

3 Theoretical framework

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. For clarity's sake, and without loss of generality, for the input sequence $\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_T)$, the output sequence $\mathbf{y}$ is rewritten as $\mathbf{y} = (y_1, \ldots, y_i, \ldots, y_{T'})$.

3.1 Background

Seq-RNNs with memories rely on three parts: one dedicated to encoding the input, and referred to as *encoder*, one dedicated to generating the output, and referred to as *decoder*, and one dedicated to the *memory model*, the role of which being to provide information from the input to generate each output element. The encoder represents each input x_j, $1 \leq j \leq T$ as a *hidden state*: $\overrightarrow{\mathbf{h}}_j = F(x_j, \overrightarrow{\mathbf{h}}_{j-1})$, with $\overrightarrow{\mathbf{h}}_j \in \mathbb{R}^n$ and where the function F is non-linear transformation that takes different forms depending on the RNN considered. We use here LSTMs with peephole connections as described in [24]. The function F is further refined, in bidirectional RNNs [30], by reading the input 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.* $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]^\mathsf{T}$, where now $\mathbf{h}_j \in \mathbb{R}^{2n}$.

The decoder parallels the encoder by associating each output $y_i, 1 \leq i \leq T'$ to a hidden state vector $\mathbf{s}_i$ that is directly used to predict the output:

$$y_i = \mathbf{W}_{\text{out}}\mathbf{s}_i + b_{\text{out}},\ s_i = G(y_{i-1}, \mathbf{s}_{i-1}, \mathbf{c}_i)$$

with $\mathbf{s}_i \in \mathbb{R}^n$. c_i is usually referred to as a *context* and corresponds to the output of the memory model. In this study, the function G corresponds to an LSTM with peephole connections integrating a context [3].

The memory model builds, from the sequence of input hidden states $\mathbf{h}_j, 1 \leq j \leq T$, the context vector $\mathbf{c} = q(\{\mathbf{h}_1, \ldots, \mathbf{h}_j, \ldots, \mathbf{h}_T\})$ that provides a summary of the input sequences to be used for predicting the output. In its most simple 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], a content attention model is used to construct different context vectors (also called attention vectors) $\mathbf{c}_i$ for different outputs y_i ($1 \leq i \leq T'$) as a weighted sum of the hidden states of the encoder representing the input history:

$$e_{ij} = \mathbf{v}_a^\mathsf{T} \tanh(\mathbf{W}_a\mathbf{s}_{i-1} + \mathbf{U}_a\mathbf{h}_j), \alpha_{ij} = \text{softmax}(e_{ij}), \mathbf{c}_i = \sum_{j=1}^{T} \alpha_{ij}\mathbf{h}_j \qquad (1)$$

where softmax normalizes the vector e_i of length T to be the attention mask over the input. The weights α_{ij}, referred to as the attention weights, correspond to the importance of the input at time j to predict the output at time i. They allow the model to concentrate, or *put attention*, on certain parts of the input history to predict each output. Lastly, $\mathbf{W}_a$, $\mathbf{U}_a$ and $\mathbf{v}_a$ are trained in conjunction with the entire encoder-decoder framework.

We present below two extensions for univariate time series to integrate pseudo-periods.

3.2 Position-based content attention mechanism

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

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

where $< ., . >$ denotes the scalar product and $\boldsymbol{\Delta}^{(i,j)} \in \mathbb{R}^T$ is a binary vector that is 1 on dimension $(i + T - j)$ and 0 elsewhere. $\boldsymbol{\Delta}^{(i,j)}$ thus selects the coordinate of $\boldsymbol{\pi}^{(1)}$ corresponding to the difference in positions between input j and output i. This coordinate is then used to increase or decrease the importance of the hidden state h_j in e_{ij}. 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 distant by more than T time steps (hence the test: $i + T - j \leq T$).

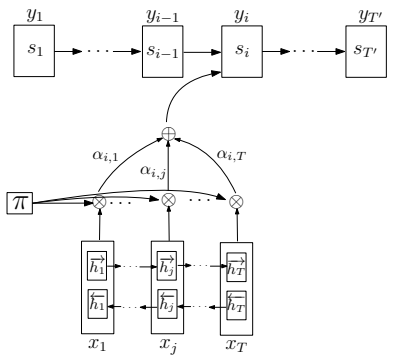

Fig. 1. The illustration of the proposed position-based content attention mechanism.

The vector e_i can then be normalized using the softmax operator again, and a context 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 $\boldsymbol{\pi}^{(1)}$ and $\boldsymbol{\Delta}^{(i,j)}$ with T' dimensions that are set to 0 in $\boldsymbol{\Delta}^{(i,j)}$, and by considering a vector $\boldsymbol{\Delta}$ of dimension $(T + T')$ that has 1 on its first T coordinates and 0 on the last T' ones. The resulting position-based attention mechanism then amounts to:

$$\text{RNN-}\pi^{(1)} : \begin{cases} e_{ij} = \mathbf{v}_a^{\mathsf{T}} \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + < \boldsymbol{\pi}^{(1)}, \boldsymbol{\Delta}^{(i,j)} > \mathbf{U}_a \mathbf{h}_j) \boldsymbol{\Delta}_{i+T-j} \\ \alpha_{ij} = \text{softmax}(e_{ij}),\ \mathbf{c}_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j \end{cases} \tag{2}$$

As one can note, $\boldsymbol{\pi}^{(1)}_{i+T-j}$ will either decrease or increase the hidden state vector $\mathbf{h}_j$ for output i. Since $\boldsymbol{\pi}^{(1)}$ is learned along with the other parameters of the Seq-RNN,

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

we expect that $\boldsymbol{\pi}^{(1)}_{i+T-j}$ will be high for those values of $i+T-j$ that correspond to pseudo-periods of the time series. We will refer to this model as RNN-$\pi^{(1)}$. Lastly, note that the original attention mechanism can be recovered by setting $\boldsymbol{\pi}^{(1)}$ to $\mathbf{1}$ (a vector consisting of 1 on each coordinate).

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 that, instead of having a scalar at each position relating the input to the output, one has a vector in $\mathbf{R}^{2n}$ that can now reweigh each coordinate of $\mathbf{h}_j$ independently. This leads to:

$$\text{RNN-}\pi^{(2)}: \begin{cases} e_{ij} = \mathbf{v}_a^\mathsf{T} \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a((\boldsymbol{\pi}^{(2)} \boldsymbol{\Delta}^{(i,j)}) \odot \mathbf{h}_j)) \boldsymbol{\Delta}_{i+T-j} \\ \alpha_{ij} = \text{softmax}(e_{ij}),\ \mathbf{c}_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j \end{cases} \tag{3}$$

where $\odot$ denotes the Hadamard product (element wise multiplication) and $\boldsymbol{\pi}^{(2)}$ is a matrix in $\mathbf{R}^{2n \times (T+T')}$. $\boldsymbol{\Delta}^{(i,j)}$ and $\boldsymbol{\Delta}$ are defined as before. We will refer to this model as RNN-$\pi^{(2)}$.

Figure 1 illustrates the overall network in which $\boldsymbol{\pi}$ is a vector for RNN-$\pi^{(1)}$ and a matrix for RNN-$\pi^{(2)}$.

3.3 Multivariate Extensions

As each variable in a K multivariate time series can have its own pseudo-periods, a direct extension of the above approaches to multivariate time series is to consider that each variable k, $1 \leq k \leq K$, of the time series has its own encoder and attention mechanism. The context vector for the i^{th} output of the k^{th} 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 k^{th} variable and $\alpha_{ij}^{(k)}$ are the weights given by the attention mechanism of the k^{th} variable. 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 to the decoder, the rest of the decoder architecture being unchanged:

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

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 $\boldsymbol{\pi}^{(1)}$ and $\boldsymbol{\pi}^{(2)}$.

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 the different variables. To do so, one can simply concatenate the hidden states of each variable into a single hidden state ($\mathbf{h}_j = [{\mathbf{h}_j^{(1)}}^T \cdots {\mathbf{h}_j^{(K)}}^T]^\mathsf{T}$) and deploy the previous attention model on top of them. This leads to the multivariate model which we refer to as RNN-$\pi^{(3)}$ and is based on the same ingredients and equations as RNN-$\pi^{(2)}$, the only difference being that RNN-$\pi^{(3)}$ is now a matrix in $\mathbf{R}^{2Kn \times (T+T')}$.

We now turn to the experimental validation of the proposed models.

4 Experiments

We retained six widely used and publicly available [31] datasets, described in Table 1, to assess the models we proposed. 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 1. 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 hyperparameters. Lastly, linear interpolation was used whenever there are missing values in the time series[2].

Table 1. Datasets.

Name	Usage	#Instances	History	Forecast horizon	Sampling rate
Polish Electricity (PSE)	Univariate	46379	96	4	2 hours
Polish Weather (PW)	Univariate	4595	548	7	1 days
Numenta Benchmark (NAB)	Univariate	18050	72	6	5 minutes
Air Quality (AQ)	Univ./Multiv.	9471	192	6	1 hour
Appliances Energy Pred. (AEP)	Univ./Multiv.	19735	216	6	10 minutes
Ozone Level Detection (OLD)	Univ./Multiv.	2536	548	7	1 day

We compared the methods introduced before, namely RNN-$\pi^{(1/2/3)}$, with the original attention model (RNN-A) and several baseline methods, namely ARIMA, an ensemble learning method (RF) and the standard support vector regression methods. Among these baselines, we retained ARIMA and RF as these were the two best performing methods in our datasets. These methods, discussed in Section 2, have also been shown to provide state-of-the-art results on various forecasting problems (*e.g.* [16]). For ARIMA, we relied on the seasonal variant [32]. 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 @2.60GHz. 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 [33], 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

[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 2. Overall results for univariate case with MSE (left value) and SMAPE (right value).

Dataset	RNN-A	RNN-$\pi^{(1)}$	RNN-$\pi^{(2)}$	ARIMA	RF	*Selected-*π
AQ	0.282*/0.694*	0.257/**0.661**	**0.25**/0.669	0.546*/0.962*	0.299*/0.762*	$\pi^{(2)}$
OLD	0.319*/0.595*	**0.271/0.523**	0.275/0.586*	0.331*/0.619*	0.305*/0.606*	$\pi^{(2)}$
AEP	0.025*/0.085*	0.029*/0.101*	0.027*/0.095*	**0.021/0.066**	0.021/0.085*	$\pi^{(2)}$
NAB	0.642*/0.442*	**0.475/0.323**	0.54*/0.369*	1.677*/1.31*	0.779*/0.608*	$\pi^{(2)}$
PW	0.166*/0.558	**0.152**/0.547	0.162*/0.565*	0.213*/0.61*	0.156/**0.544**	$\pi^{(1)}$
PSE	0.034*/0.282*	**0.032**/0.264*	0.033*/**0.256**	0.623*/1.006*	0.053*/0.318*	$\pi^{(1)}$

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.

Overall results on 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 2 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 table, *Selected-*π, indicates which method, among RNN-$\pi^{(1/2)}$, 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-$\pi^{(1)}$ and RNN-$\pi^{(2)}$, 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 method (column *Selected-*π) 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-$\pi^{(1/2)}$ significantly improves 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 to capture pseudo-periods, we display in Figure 2 (left) the autocorrelation plot for PSE, and in Figure 2 (right) the average attention weights for the same time series obtained with RNN-A and RNN-$\pi^{(2)}$ (averaged

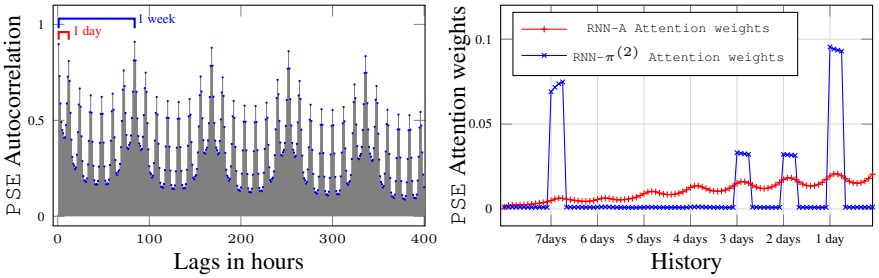

Fig. 2. Autocorrelation of PSE (left). RNN-A and RNN-$\pi^{(2)}$ attention weights (right).

over all test examples and forecast horizon points). As one can see from the autocorrelation plot, PSE has two main weekly and daily pseudo-periods. This two pseudo-periods are clearly visible in the attention weights of RNN-$\pi^{(2)}$ 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). The attention weights of RNN-$\pi^{(1)}$ (not shown here for space reasons) are very similar. In contrast, the attention weights of the original attention model (RNN-A) follow a general increasing behaviour with more weights on the more recent time stamps. This model thus misses the pseudo-periods.

Results on multivariate time series

As mentioned in Table 1, we furthermore conducted multivariate experiments on AQ, AEP and OLD using the multivariate extensions described in Section 3. 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-$\pi^{(1/2/3)}$.

Table 3 shows the results of our experiments on multivariate sets with MSE (SMAPE, not displayed here for readability reasons, has a similar behaviour). 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-$\pi^{(1)}$). Compared to RF, one can obtain between 11% (AEP) and 40% (AQ) of improvement. As one can note, the selected method is always RNN-$\pi^{(3)}$. 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-$\pi^{(3)}$ remains better than the state-of-the-art baselines retained, RNN-A and RF.

5 Conclusion

We studied in this paper the use of Seq-RNNs, in particular the state-of-the-art bidirectional LSTMs encoder-decoder with a content attention model, for modelling and

Table 3. Overall results for multivariate case with MSE.

Dataset	RNN-A	RNN-$\pi^{(1)}$	RNN-$\pi^{(2)}$	RNN-$\pi^{(3)}$	RF	*Selected-*π
AQ	0.352*	0.276*	**0.268**	0.3*	0.45*	$\pi^{(3)}$
OLD	0.336*	0.328*	0.327*	**0.274**	0.315*	$\pi^{(3)}$
AEP	0.029*	**0.024**	0.036*	0.026*	0.027*	$\pi^{(3)}$

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 pseudo-periods. We thus proposed three extensions of the content attention model making use of the (relative) positions in the input and output sequences (hence the term *position-based content attention*). The experiments we conducted over several univariate and multivariate time series demonstrate the effectiveness of these extensions, on time series with either clear pseudo-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. We conjecture that this is what happening on the multivariate time series we have retained.

References

1. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. International journal of forecasting 22(3), 443–473 (2006)
2. Bontempi, G., Taieb, S.B., Le Borgne, Y.A.: Machine learning strategies for time series forecasting. In: Business Intelligence, pp. 62–77. Springer (2013)
3. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
4. Weston, J., Chopra, S., Bordes, A.: Memory networks. CoRR abs/1410.3916 (2014)
5. Weston, J., Bordes, A., Chopra, S., Mikolov, T.: Towards AI-Complete question answering: A set of prerequisite toy tasks. CoRR abs/1502.05698 (2015)
6. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwinska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A.P., Hermann, K.M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., Hassabis, D.: Hybrid computing using a neural network with dynamic external memory. Nature 538(7626), 471–476 (2016)
7. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
8. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: NIPS. pp. 2692–2700 (2015)
9. Walker, G.: On periodicity in series of related terms. Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 131(818), 518–532 (1931)
10. Slutzky, E.: The summation of random causes as the source of cyclic processes. Econometrica: Journal of the Econometric Society pp. 105–146 (1937)
11. Box, G.E., Jenkins, G.M.: Some recent advances in forecasting and control. Journal of the Royal Statistical Society. Series C (Applied Statistics) 17(2), 91–109 (1968)
12. Tiao, G.C., Box, G.E.: Modeling multiple time series with applications. journal of the American Statistical Association 76(376), 802–816 (1981)

13. Sapankevych, N.I., Sankar, R.: Time series prediction using support vector machines: A survey (2009)
14. Creamer, G.G., Freund, Y.: Predicting performance and quantifying corporate governance risk for latin american adrs and banks (2004)
15. Kusiak, A., Verma, A., Wei, X.: A data-mining approach to predict influent quality. Environmental monitoring and assessment 185(3), 2197–2210 (2013)
16. Kane, M.J., Price, N., Scotch, M., Rabinowitz, P.: Comparison of arima and random forest time series models for prediction of avian influenza H5N1 outbreaks. BMC bioinformatics 15(1), 276 (2014)
17. Connor, J., Atlas, L.E., Martin, D.R.: Recurrent networks and NARMA modeling. In: NIPS. pp. 301–308 (1991)
18. Giles, C.L., Lawrence, S., Tsoi, A.C.: Noisy time series prediction using recurrent neural networks and grammatical inference. Machine learning 44(1-2), 161–183 (2001)
19. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. science 304(5667), 78–80 (2004)
20. Hsieh, T.J., Hsiao, H.F., Yeh, W.C.: Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. Applied soft computing 11(2), 2510–2525 (2011)
21. Längkvist, M., Karlsson, L., Loutfi, A.: A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters 42, 11–24 (2014)
22. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
23. Gers, F.A., Eck, D., Schmidhuber, J.: Applying LSTM to time series predictable through time-window approaches. In: International Conference on Artificial Neural Networks. pp. 669–676. Springer (2001)
24. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with LSTM recurrent networks. Journal of machine learning research 3(Aug), 115–143 (2002)
25. Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014)
26. Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c.: Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems. pp. 802–810 (2015)
27. Lipton, Z.C., Kale, D.C., Elkan, C., Wetzell, R.: Learning to diagnose with LSTM recurrent neural networks. arXiv preprint arXiv:1511.03677 (2015)
28. Riemer, M., Vempaty, A., Calmon, F.P., Heath III, F.F., Hull, R., Khabiri, E.: Correcting forecasts with multifactor neural attention. In: Proceedings of The 33rd International Conference on Machine Learning. pp. 3010–3019 (2016)
29. Choi, E., Bahadori, M.T., Sun, J., Kulas, J., Schuetz, A., Stewart, W.: RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In: NIPS. pp. 3504–3512 (2016)
30. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45(11), 2673–2681 (1997)
31. Datasets: PSE: `http://www.pse.pl`. PW: `https://globalweather.tamu.edu`. AQ/OLD/AEP: `http://archive.ics.uci.edu`. NAB: `https://numenta.com`
32. Hyndman, R., Khandakar, Y.: Automatic time series forecasting: The forecast package for r. Journal of Statistical Software, Articles 27(3), 1–22 (2008), `https://www.jstatsoft.org/v027/i03`
33. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)