Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks

Article *in* International Journal of Computer Applications · June 2016

DOI: 10.5120/ijca2016910497

2 authors:

Mohamed Akram Zaytar
Faculty of Science and Technology, Tangier
1 PUBLICATION **87** CITATIONS
SEE PROFILE

Chaker El Amrani
Abdelmalek Essaâdi University, Morocco
51 PUBLICATIONS **253** CITATIONS
SEE PROFILE

International Journal of Computer Applications (0975 - 8887)
Volume 143 - No.11, June 2016

Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks

Mohamed Akram Zaytar
Research Student
Department of Computer Engineering
Faculty of Science and Technology, Tangier
Route Ziaten
Tangier, 90000, Morocco

Chaker El Amrani
Associate Professor
Department of Computer Engineering
Faculty of Science and Technology, Tangier
Route Ziaten
PO. Box 416, Tangier, 90000, Morocco

ABSTRACT

The aim of this paper is to present a deep neural network architecture and use it in time series weather prediction. It uses multi stacked LSTMs to map sequences of weather values of the same length. The final goal is to produce two types of models per city (for 9 cities in Morocco) to forecast 24 and 72 hours worth of weather data (for Temperature, Humidity and Wind Speed). Approximately 15 years (2000-2015) of hourly meteorological data was used to train the model. The results show that LSTM based neural networks are competitive with the traditional methods and can be considered a better alternative to forecast general weather conditions.

General Terms

Machine Learning, Weather Forecasting, Pattern Recognition, Times Series

Keywords

Deep Learning, Sequence to Sequence Learning, Artificial Neural Networks, Recurrent Neural Networks, Long-Short Term Memory, Forecasting, Weather

1. INTRODUCTION

Weather Forecasting began with early civilizations and was based on observing recurring astronomical and meteorological events. Nowadays, weather forecasts are made by collecting data about the current state of the atmosphere and using scientific systems to predict how the atmosphere will evolve. The chaotic nature of the atmosphere, the massive computational power required to solve all of the equations that describe the atmosphere mean that forecasts become less accurate and more expensive as the range of the forecasts increase, this puts us in a position to think of new ways to forecast weather that can be more efficient and/or less expensive. In Machine Learning, Artificial Neural Networks (ANNs) are classes of models inspired by biological neural networks, which comprise interconnected adaptive processing nodes or units. What makes ANNs important is their adaptive nature, this feature makes ANNs a well suited tool to approximate highly nonlinear and multivariate functions.

Because the final Neural Network model predicts time series values, it uses LSTM layers in its architecture to counter time related problems like the "Vanishing Gradient Problem". the difference between LSTMs and other traditional Recurrent Neural Networks (RNNs) is its ability to process and predict time series sequences without forgetting unimportant information, LSTMs achieve state of the art results in sequence related problems like handwriting recognition [4, 3], speech recognition [6, 1], music composition [2] and grammar learning [8] (In natural language processing).

2. RECURRENT NEURAL NETWORKS AND LSTMS

The Recurrent Neural Network Architecture is a natural generalization of feedforward neural networks to sequences, RNNs are networks with loops in them, which results in information persistence.

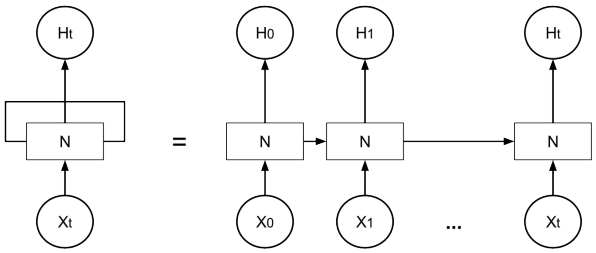

Fig. 1. The relation between simple RNNs and Feed-forward ANNs

Given a sequence of inputs $(x_1, x_2, ..., x_N)$, a standard RNN computes a sequence of outputs $(y_1, y_2, ..., y_3)$ by iterating over the following equation :

$$h_t = sigm(W^{hx}x_t + W^{hh}h_{t-1})$$

$$y_t = W^{yh}h_t$$

The RNN can map sequences to sequences whenever the alignment between the inputs and the outputs is known ahead of time. In the

context of predicting future weather values sequences of lengths 24 and 72 values were set.

2.1 The problem of Long-Term Dependencies

When thinking about weather data, One gets the impression that recent weather data is more important for forecasting the next 24 hours than data from the previous month. In this context, one might only need to look at recent information to perform the forecasting. However, there might be cases when older data can help the model recognize general trends and movements that recent data fail to show.
Unfortunately, as the gap grows between the present and the past data, general RNNs fail to learn to connect the inputs, and this is called the problem of Long-Term Dependencies.

2.2 Long-Short Term Memory Neural Networks

LSTMs are a type of Recurrent Neural Networks capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber [7]. LSTMs remember information for long periods of time thanks to their inner cells which can carry information unchanged at will. The network have the complete control over the cell state, it can add,edit or remove information in the cell using special structures called gates.

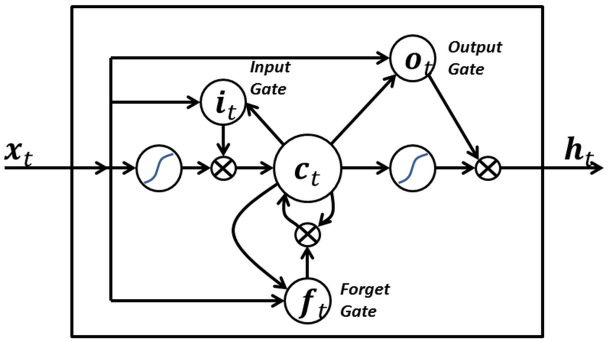

Fig. 2. A simple LSTM gate with only input, output, and forget gates.

To put it in simple terms, What makes the network have control over flowing information is largely due to its sigmoid layer which outputs numbers between zero and one ($S(t) = \frac{1}{1+e^{-t}}$) , this plays a role similar to a "Faucet" in controlling how much of each component should be let through. A value of zero means "let nothing through", while a value of one means "let everything through" and with this type of system a model can make a far distant value in the past significant in its prediction for the next hour as an example. And this is what makes LSTMs useful for the model in use.
Mathematically speaking, The goal of the LSTM is to estimate the conditional probability $p(y_1, ..., y_N|x_1, ..., x_N)$ where $(x_1, ..., x_N)$ is an input sequence and $(y_1, ..., y_N)$ is its corresponding output sequence with the same length. The LSTM computes this conditional probability by first obtaining the fixed-dimensional representation v of the input sequence $(x_1, ..., x_N)$ given by the last hidden state of the LSTM, and then computing the probability of $(y_1, ..., y_N)$ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of $(x_1, ..., x_N)$:

$$p(y_1, ..., y_N|x_1, ..., x_N) = \prod_{t=1}^{N} p(y_t|v, y_1, .., y_{t-1})$$

3. THE MODEL

A multi layer model consisting of two LSTM layers and a fully connected hidden layer with a 100 neuron was implemented to form the base architecture for the model, The general layers and their I/O shapes are in the following figure :

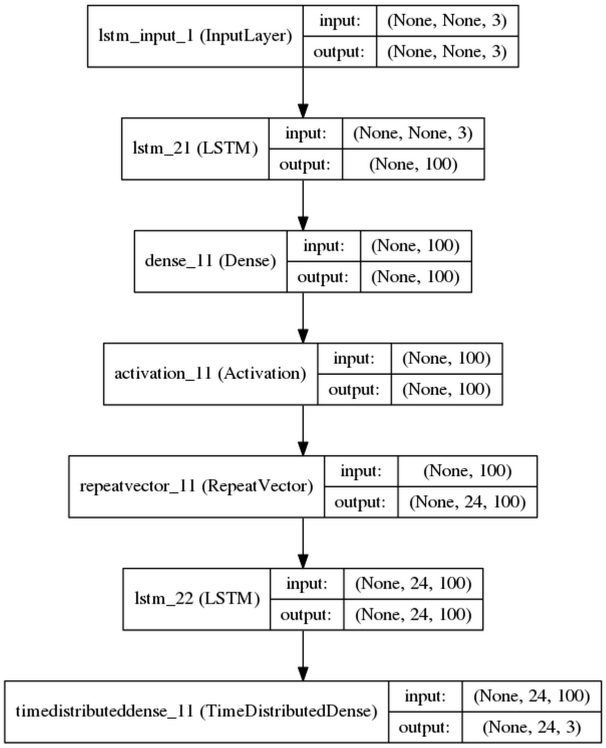

Fig. 3. The Model's Layers with I/O shapes.

—Dense : A fully connected layer of neurons, a value of 100 hidden neurons have been chosen to train based on multiple experiments.

—Activation : As an Activation function the Rectifier function have been chosen.

—Repeat Vector : this layer repeat the final output vector from the encoding layer as a constant input to each timestep of the decoder.

—Time Distributed Dense : Applies a same Dense (fully-connected) operation to every timestep of a 3D tensor.

As an optimizer for the network, it used "RMSprop", which is an unpublished, adaptive learning rate method proposed by Geoff Hinton in Lecture 6e of his Coursera Class [5]. It keeps a moving average of the squared gradient for each weight :

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

RMSprop as well divides the learning rate by an exponentially decaying average of squared gradients. Hinton suggests γ to be set to 0.9, while a recommended value for the learning rate η is 0.001. And as a loss estimator, the network used the mean squared error estimator which is one of the most widely adapted ones :

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i^p - Y_i^r)^2$$

Looking from the outside, the model's architecture is simple. it gets as input the previous 24 values and it gives us back the next 24 hours of predictions, the same model was constructed for 72 hours (3 days), the figure below is a graphical explanation (24 hours) :

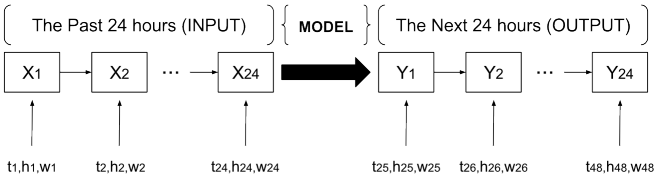

Fig. 4. Input/Output Data Flow.

ti, hi, wi signifies Temperature, Humidity and wind speed in time i, the result is a deep neural network based on LSTMs that can predict a fixed length sequence from the same length previous sequence.

4. EXPERIMENTS

Fifteen years worth of hourly data was trained, from nine cities in Morocco. With the following variables :

Table 1. METEOROLOGICAL VARIABLES

No.	Meteorological Variables every hour time frame	Unit
1	Temperature	Deg. C
2	Relative Humidity	%Rh
3	Wind Speed	km/h

4.1 Dataset Details

Weather data from nine cities were collected using Wunderground's API (Wunderground collect most of its data in morocco from weather stations in airports). The final data set consists of Temperature, Humidity and wind speed values for every hour, Here is a figure of Temperature values for fifteen years in the city of tangier :

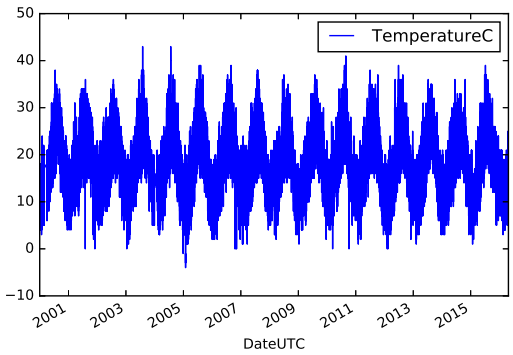

Fig. 5. 15 Years of Temperature values in Morocco, Tangier.

4.2 Data Preprocessing

Data was often not consistent, missing values or values out of range was common, however, there was no long continuous noise segments (for days as an example). The methods used for cleaning is to replace the missing or noisy values by forward filling them using previous points in time (ex. filling the missing temperature value with the last recorded temperature value).
The following figure demonstrates the procedure used for data preprocessing :

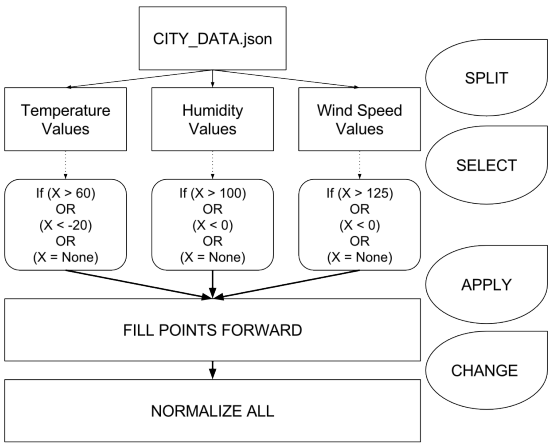

Fig. 6. The Data preprocessing graphical Model.

After cleaning the data, all of the values were normalized to points in $[-1, 1]$ to avoid training problems (example: local optima problem) and also weight decay and Bayesian estimation can be done more conveniently with standardized inputs [9]. this assertion was used :

$$X_i \leftarrow \frac{X_i - Mean(X)}{MAX(X) - MIN(X)}$$

4.3 Validation

At first, all of the values were reshaped into a serie of sequences to feed into the neural network, each input consists of 24 (or 72) triples $[T, H, WS]$ of values.

The chosen weather data was divided into three selected groups of sequences, the training group, corresponding to 80% of the sequences, the test group (untrained data) used to evaluate the models after the training phase (corresponding to 10% of sequences), and finally the local validation data for the local mini-batch training. The Mean Squared Error (MSE) was used as a measure of error made by the neural network model.

4.4 Training Details

It's known that LSTM models are fairly easy to train after preparing the data. A model consisting of two LSTM layers and a fully connected hidden layer in between with a 100 neuron was used. The resulting neural network had 132,403 parameters of which 122,000 are pure recurrent connections (41600 for the "encoder" LSTM and 80400 for the "decoder" LSTM). The training details are given below :

—All of the LSTM's parameters were initialized with the uniform distribution between -0.05 and 0.05.

—The Mini-batch gradient descent was used with a fixed learning rate of 0.001.

—For the gradient method, Batches of 512 sequences and 100 epochs were trained.

4.5 Parallelization

A Python implementation of deep neural networks with the configuration from the previous section was used to train the model. The Theano library was used to build and compile the model on a NVIDIA GRID K520 single GPU device, with 1536 CUDA cores, 800MHz Core Clocks and 4GB of memory size. Training took about 10 hours per model :

Table 2. AVERAGE TRAINING TIME PER MODEL

Model	Training Time
LSTM24	3.2h
LSTM72	6.7h

4.6 Experimental Results

The Mean Squared Error was used as a score function to evaluate the quality of the predictions. The scores were calculated using the untrained test data with the following results as an example :

Table 3. MEAN SQUARED ERROR ON TEST DATA FROM NINE CITIES IN MOROCCO

City	24 hours MSE	72 hours MSE
Tangier	0.00636	0.00825
Agadir	0.00775	0.01046
Al Hoceima	0.00827	0.00985
Fes-Sais	0.00744	0.00997
Marrakech	0.00839	0.01053
Nouasseur	0.00516	0.00698
Oujda	0.00536	0.00987
Rabat-Sale	0.00550	0.00675
Tetouan	0.00695	0.00863

Examples of graphical predictions on test data next.

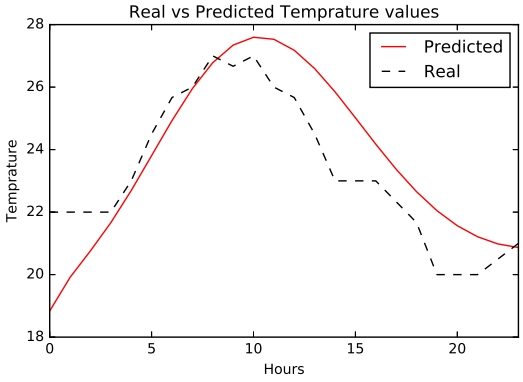

Fig. 7. Comparison of Temperature values for 24 hours.

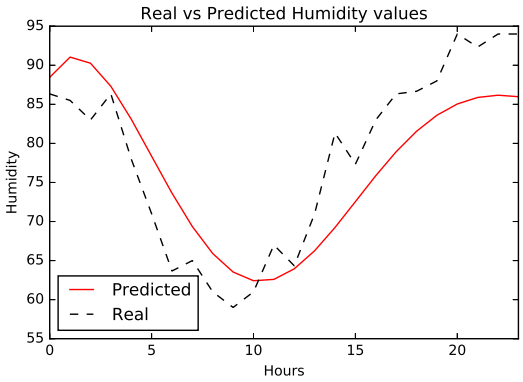

Fig. 8. Comparison of Humidity values for 24 hours.

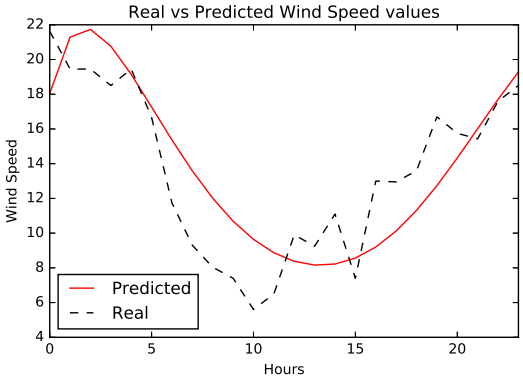

Fig. 9. Comparison of Wind Speed values for 24 hours.

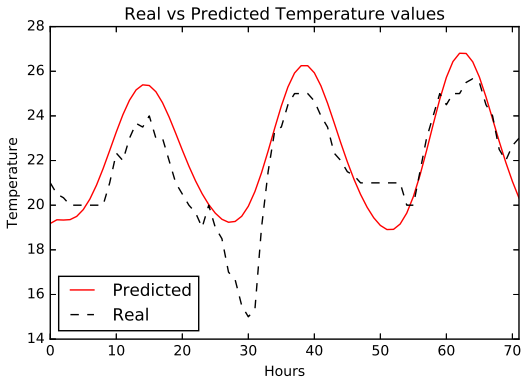

Fig. 10. Comparison of Temperature values for 72 hours.

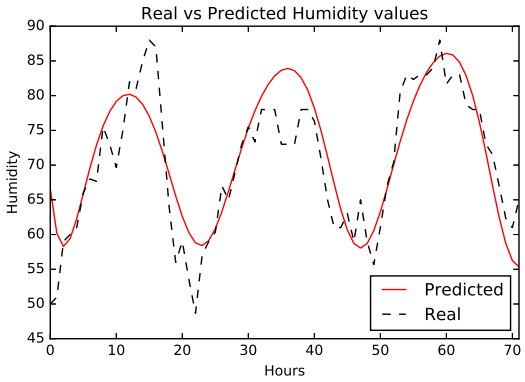

Fig. 11. Comparison of Humidity values for 72 hours.

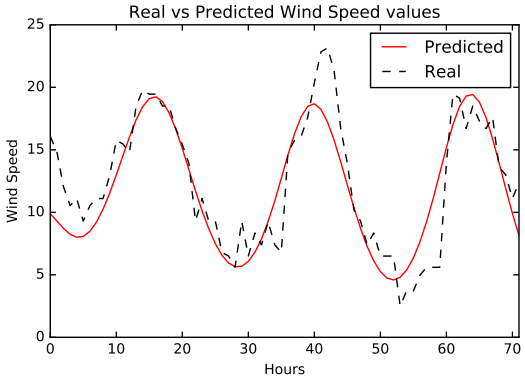

Fig. 12. Comparison of Wind Speed values for 72 hours.

5. CONCLUSION

The results of this paper shows that a deep LSTM network can forecast general weather variables with a good accuracy. The success of the model suggests that it could be used on other weather related problems, and while Theano provides excellent environment to compile and train models, it also gives the ability to carry any model into a production server and integrate them in pre-existing applications (as an example, one could perform real time predictions on top of an existing web application). Our vision is for this model to represent the cornerstone of an Artificial intelligence based system that can replace humans and traditional methods in weather forecasting in the future. combining numerical models and image recognition ones (in satellite images for example) might form the basis of a new weather forecasting system that can outperform and overcome the traditional expensive ones and become the new standard in weather prediction.

6. ACKNOWLEDGMENTS

We would like to express our gratitude to Weather Underground for providing us the required data to do this research, and also we want to thank the teams behind the python libraries, Theano and Keras for making it easy for us to implement our model and run it.

7. REFERENCES

[1] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

[2] Douglas Eck and Jürgen Schmidhuber. Learning the long-term structure of the blues. In *Artificial Neural NetworksICANN 2002*, pages 284–289. Springer, 2002.

[3] Alex Graves, Marcus Liwicki, Horst Bunke, Jürgen Schmidhuber, and Santiago Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 577–584, 2008.

[4] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[5] Geoffrey Hinton. Overview of mini-batch gradient descent. `http://goo.gl/A9lKPi`, 2014. [Online; accessed 09-June-2016].

[6] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

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

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[9] NC Warren S. Sarle, Cary et al. FAQ, Part 2 of 7: Learning. `http://goo.gl/gzJGse`, 2002. [Online; accessed 09-June-2016].