Using Deep Learning for Forecasting Tuberculosis Morbidity Rate

Jasvinder Singh
Department of Computer Science
CDGI
Indore, India
jasvinder2604@gmail.com

Tushar Dhyani
Department of Information Technology
MITS University
Gwalior, India
dhyanitushar@gmail.com

Akshat Jain
Department of Information Technology
MITS University
Gwalior, India
jainakshat889@gmail.com

Parag Agrawal
Department of Computer Science
SVITS
Indore, India
agrawal.parag1997@gmail.com

***Abstract*—Tuberculosis is part of a global epidemic which is communicable and spreads through close contact with the carrier of Mycobacterium Tuberculosis. There are currently multiple statistical approaches to forecast the morbidity rate of Tuberculosis. Two of them are ARIMA and ARIMA-ARCH model as were discussed by Zheng et al. (2015). In this paper we will use Deep Learning algorithm - Long Short Term Memory (LSTM) for forecasting the morbidity rate and achieve an accuracy which is greater than the previously discussed statistical approaches.**

Keywords—Tuberculosis, forecasting, Deep Learning, LSTM

I. INTRODUCTION

Tuberculosis is a highly communicable disease caused by Mycobacterium Tuberculosis (MTB). The potential of Deep Learning is being harnessed in the fields of health Informatics where it is being deployed for solving problems through its efficient computational techniques. In this paper, we will use Recurrent Neural Networks (RNN) in conjunction with Long Short-Term Memory (LSTM) for predicting the morbidity rate for Tuberculosis. This algorithm is an improvement over the statistical techniques like ARIMA and ARIMA-ARCH as it will provide us an accuracy much greater than what was achieved with the latter approaches. We will use the same data as was used by Zheng et al. (2015). The data is available publicly and is derived from the Xinjiang Province, China which consists of the morbidity rate for ten years, from January 2004 to June 2014. The previous paper used ARIMA $(1, 1, 2)$ $(1, 1, 1)_{12}$ model and the combined ARIMA $(1, 1, 2)$ $(1, 1, 1)_{12}$-ARCH model for forecasting the Tuberculosis Morbidity Rate. We use a specific architecture of LSTM known as stacked LSTM which contains multiple hidden layers of LSTM and each layer contains multiple hidden cells. We will take an overview of the LSTM model and the stacked LSTM architecture which we will implement with RNNs for learning the time-series pattern of our dataset. We will also implement several statistical testing methods on our data and delineate visualizations for the same.

We also propose to host this as an analytics cloud service that can be accessed by the government authorities and people of importance for keeping in check the morbidity rate for further control and prevention.

II. MODEL AND ALGORITHMS USED

A. Recurrent Neural Network

Sequential problems consist of directed cycles dependent on time steps. The data to be processed is carried with these time steps having an arbitrary length. To compute such order of data, it is required by the artificial neural network to store a separate memory cell for storing data from previous time steps and also propagates the notion of context to the next time step. The data which we use is a series of time steps that contain a single independent variable denoting time and the output variable representing the morbidity rate. To compute this information and forecast predictions, we use Recurrent Neural Networks.

Recurrent Neural Networks are defined as a type of Neural Network that use sequential connection of nodes to form a

directed graph. This directed graph is useful for passing the information from previous time step to next time step. In a way, RNN maintains the flow of information by addressing another memory cell to hold data from previous time step. Due to constant propagation of data in the form of time-steps, RNNs are used in sequential learning problems like speech recognition and natural language processing. In our paper, we will use Recurrent Neural Networks for implementing the backbone of the morbidity forecasting.

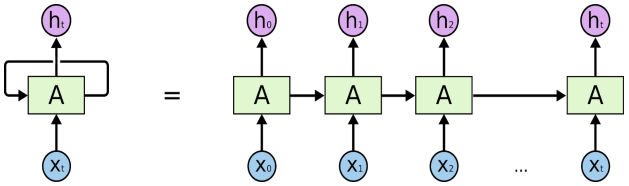

Fig 2. Representation of the unfolding of RNN timesteps

Recurrent Neural Networks can be delineated through following mathematical steps -

$$h_t = f(h_{t-1}, x_t)$$

h_t represents the current state of the hidden layer that encounters the input x_t passing through the previous hidden layer h_{t-1}.

Considering a minimal structure of recurrent neural network, we use the activation function *tanh* having weight at the recurrent neuron as W_{hh} and the weight for the input neuron as W_{xh} and obtain the following equation:

$$h_t = tanh(W_{hh} \times h_{t-1} + W_{xh} \times x_t)$$

The above equation represents the Recurrent Neural Network taking into account the previous time-step h_{t-1} multiplied with weight W_{hh}. The input x_t is coupled with the weight of the input neurons. Finally, the sum of the two units coupled together are added and passed through the *tanh* function.

Upon reaching the final state, the final output can be calculated after computing the sequences of time-steps.

$$y = W_{hy} + h_t$$

Our final loss calculation is known as a cross-entropy loss which is defined as follows:

$$E(y, \hat{y}) = -\sum_t y_t log\, \hat{y}_t$$

where y_t is the correct word at time step *t* and $\hat{y}_t$ is the prediction.

While the recurrent neural networks are apt at storing the previous context in their designated memory cells, they suffer from the problem of vanishing gradient. This problem causes the RNN to lose the context of sentence meaning over long sequences.

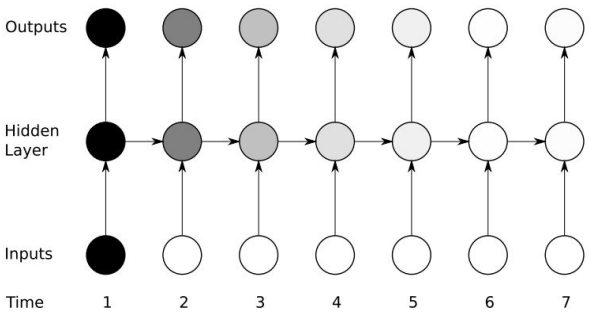

Fig 3. Visualization of Vanishing Gradient in RNN

This problem of vanishing gradient can be alleviated with the help of Long Short-Term Memory which are more specifically used in scenarios having the input sequences having a considerable length that can result in loss of gradient through several timesteps.

B. Long Short-Term Memory

The LSTMs were specifically designed to counter the problem of vanishing gradient. They work through mechanism of cell gates to selectively remember or forget values. There are three parts of the LSTM -

- The previous cell state
- The previous hidden state
- The input at the current time step

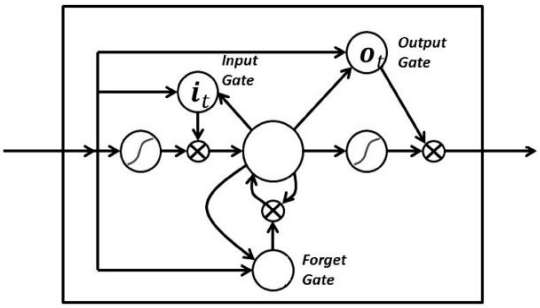

Fig 3. Long Short-Term Memory

The gates comprise of forget gate, input gate, output gate and has a hidden state for storing memory. The forget gate removes the information that is not required. The h_{t-1} and x_t output a number between 0 and 1 in the cell state C_{t-1}.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

If the output is 1, then the cell holds to the information, else it forgets it. The next step is determined by the input gate which has a sigmoid layer for updating the values.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

We then use *tanh* layer and use it to transfer values to the next cell state - $\hat{C}_t$

$$\hat{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c]$$

The next step involves combining $\hat{C}_t$ and i_t and updating the new cell state C_t, transitioning from C_{t-1}.

Multiplying the old state with f_t and combining $i_t * \hat{C}_t$ through addition, we obtain:

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t$$

Finally, we decide the output by passing the state through *tanh* and we obtain an output which is the product of the state and output of the sigmoid gate.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

and the final cell state is obtained as-

$$h_t = o_t * tanh(C_t)$$

The variation of LSTM that we use in our architecture is Stacked LSTM. This type of architecture consists of multiple LSTMs stacked over each other like blocks. We will further discuss about this architecture during its implementation in the Experiment section.

III. EXPERIMENT

A. Data

The data is collected from Xinjiang Province of China from January 2004 to June 2014 through the website of the Bureau of Health of the Xinjiang Uyghur Autonomous region of China. The TB cases were confirmed with pathological and clinical experiments. We will use this data to perform a time-series prediction for assimilating the stacked LSTM architecture.

The initial experiments were performed on ARIMA and ARCH ARIMA model but our special focus would be the deep learning model for assessing the morbidity rate.

B. Analysis of Data and Visualization

The first step towards creation of the model is data wrangling and eliciting the data to be experimented with. Furthermore, we also determine the residual errors, which are integral to the analysis of our model. Another important aspect is to check whether the residual error data is centered at 0 and if the distribution is Gaussian or Exponential. The final model to be implemented would have undergone various data cleaning and preprocessing for finding the much required optimized model generating valid predictions.

We proceed through the first step of data representation and understanding a sample taken from the dataset for discerning the independent and dependent variables. For this, we output the first five values of the dataset, that is, the first five months of year 2004 and their corresponding morbidity rate.

date	the tuberculosis morbitity
2004-01-01	16.07
2004-02-01	14.24
2004-03-01	18.03
2004-04-01	15.55
2004-05-01	13.64

Fig 4. First five samples of the TB morbidity dataset

The next step is visualizing all the data-points from year 2004 to 2014. This plot will represent the morbidity rate against the time of the year to generate an inferential visualization. The data for each year is split into twelve months.

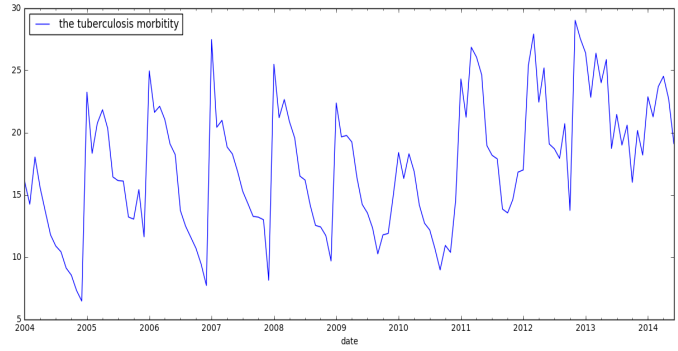

Fig 5. Graph showing the morbidity rate for 14 years

The next step is the visualization of the autocorrelation plot. The autocorrelation plot is useful for determining that the time-series data is not random and the data-points have some degree of correlation. This test plays a significant roles in developing time-series forecasting models for validating the type of data.

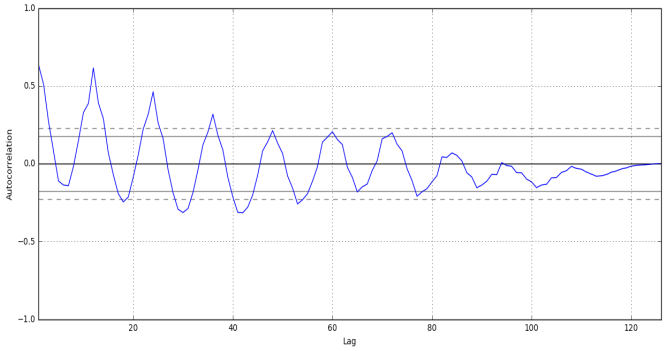

Fig 6. Graph showing the visualization of the autoregressive plot.

From the above visualization of autoregressive plot, we infer that one or more correlations are above 0. This means that we are safe to proceed with the development of the forecasting model where certain degree of accuracy is assured.

Another important aspect of a time-series model is the process of inferring residual errors from an autocorrelated dataset. The residual errors denote the difference between the observed values and their arithmetic values.

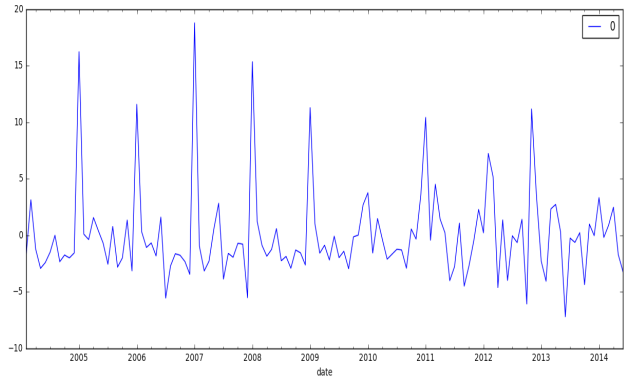

Fig 7. Residual Error Graph

We obtain the following description of the residual errors.

TABLE I.

Serial No.	Residual Error Description	
	Statistical Assessment	***Value***
1	Count	125.00
2	Mean	-0.0159
3	Standard Deviation	4.0964
4	Minimum	-7.2184
5	25%	-2.1967
6	50%	-0.8027
7	75%	0.9741
8	Max	18.7722

a.

The next step is the generation of the residual plot distribution for inference of the type of error distribution.

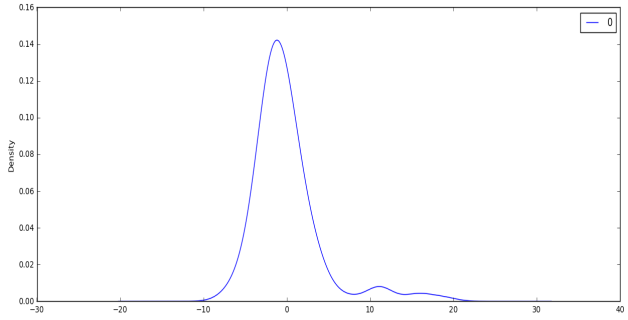

Fig 7. Residual Plot distribution

We then proceed with data uniformity using the MinMaxScaler function for transforming the data within the range of 0-1. The function is imported from the sklearn.preprocessing library and we fit our 'dataset' variable containing the time series data as follows:

```
MinMaxScaler(feature_range=(0, 1))
scaler.fit_transform(dataset)
```

We then split our dataset into training and validation/test sets. The training size is set to comprise of 70% of the dataset and remaining 30% is assigned for validation testing.

Finally, we implement our Stacked LSTM architecture. Using a batch size of 1 in the input, we proceed to add two layers of LSTMs, followed by a Dense Layer. We finally use Adam optimizer which works as an improvement over the traditional stochastic gradient descent optimizer that uses iterative updation of weights in the training set. We use Mean Squared Error (MSE) as our loss function which was also the metric for evaluating the score of the ARIMA and ARIMA-ARCH time series model as was discussed by Zheng et. al (2015). The following data-flow diagram delineates the flow of Stacked LSTM architecture for computing the predictions of our training and test sets.

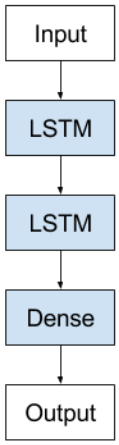

Fig 8. Architecture of the Stacked LSTM model

We then produce the graph for the original Tuberculosis Morbidity data on which we will generate our predictions on both the training set and validation set.

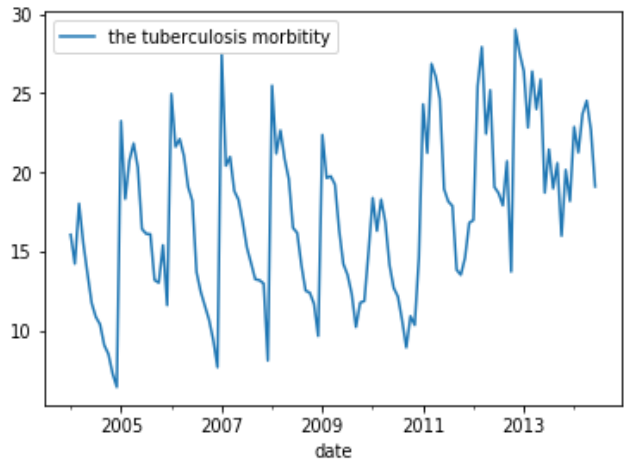

Fig 9. Graph of the original dataset over the course of 14 years

We then proceed to the the final predictions and outputting the RMSE for the model. One slight change to the stacked LSTM architecture we implemented was the addition of memory between the batches. This results in making the LSTM layer more "stateful". Stateful refers to creating new states over longer sequences and making predictions through maintenance of every state.

This requires an explicit iteration within each epoch and call model.fit() and model.reset_states() as follows:

for i in range(50):

model.fit(trainX, trainY, epochs = 1, batch_size = 1, verbose = 2, shuffle = False)

model.reset_states()

In order to obtain the predictions back to the original state, we implement the "scaler.inverse_transform" function and obtain the final predictions.

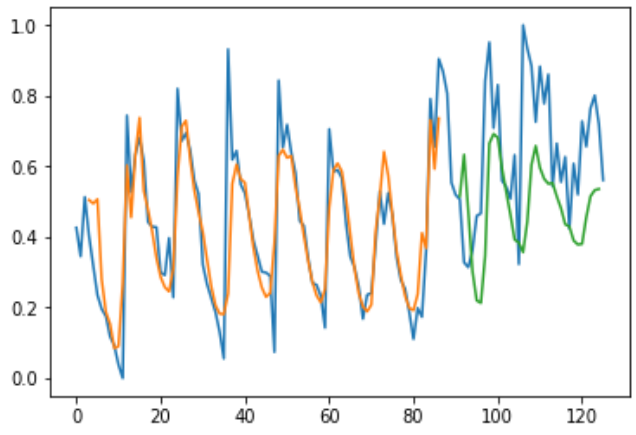

Fig 10. Graph showing the predicted output for training and test set

The blue line in the graph represents the original morbidity rate, the orange line represents the training set accuracy score and the green line denotes the predictions performed on the validation set.

We then calculate the accuracy score using the Root Mean Squared Error (RMSE) metric on both the training set as well as the validation set. The RMSE is defined as follows:

$$RMS(\hat{y}) = \left(\sqrt{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2} / n\right)$$

Where $\hat{y}$ is the predicted output and y is the original datapoint at position *i.*

We obtain the following RMSE for training and validation sets –

TABLE II.

Serial No.	RMSE Evaluation Table	
	Dataset Type	***RMSE***
1	Training Set	0.12
2	Validation Set	0.25

The results obtained by the ARIMA and ARIMA-ARCH model in the paper Zheng et. al. (2015), were 2.58 and 1.7 respectively. The validation test sample in the Stacked LSTM model achieved a validation score of 0.25 RMSE which implies that the error is significantly less in the above discussed deep learning architecture.

We performed 50 iterations per single epoch and about 10 epochs. The combined time from training the model to outputting the predictions took about 5 minutes. With more advanced GPUs it may take even less time. Furthermore, with increase in dataset over time, the accuracy score will also improve due to increment in dataset instances.

IV. FUTURE DEVELOPMENTS

The future developments of this project not only involves the morbidity rate prediction of Tuberculosis but also through implementation of several prevention strategies to combat the disease in case of an epidemic outbreak. Strategies involve monitoring of Tuberculosis diagnosed patients and their vicinities by members of health organizations.

The monitoring system is implemented on a web application powered by RESTful services that uses a database to allot special areas with TB diagnosed patients.

.

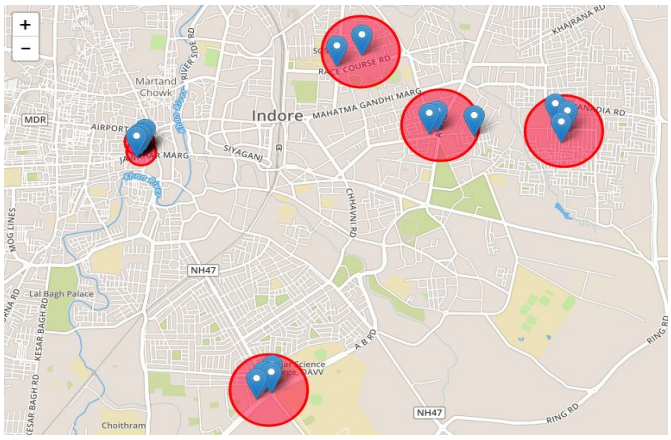

Fig 11. Monitoring System for mapping TB Diagnosed patients

The following UML diagram delineates the database design for user/patient interacting with the web-application.

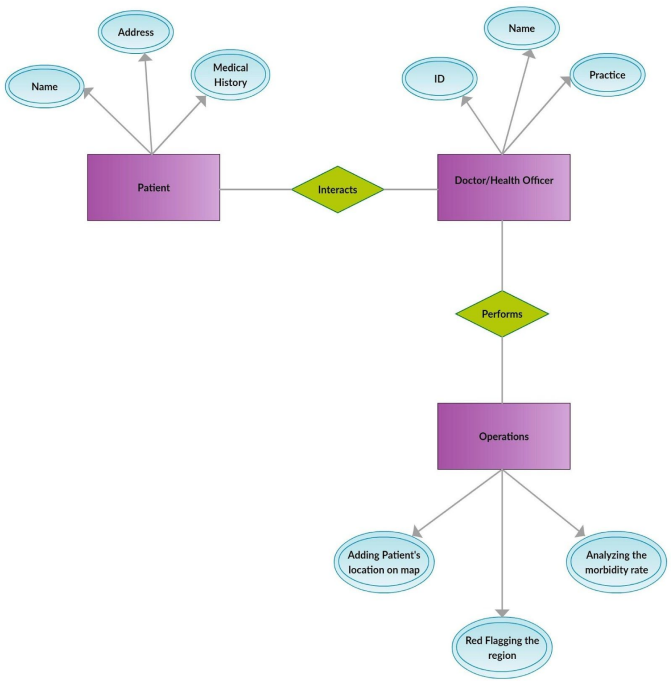

Fig 12. UML Diagram for the TB Prevention & Monitoring Database

All these strategies are aimed towards controlling Tuberculosis through knowledge of morbidity rate and monitoring the locations of TB diagnosed patients.

Acknowledgment

We would like to acknowledge Prof. Swapnil Soner of Department of Computer Science, CDGI for his immense help and supervision with this project. We would also express our gratitude towards Prof. K. Srikant, Head of the Department of Innovation, CDGI for making this project possible with his knowledge and assistance,

References.

[1] Chinese Center for Disease Control and Prevention (2010) (The current situation of tuberculosis). Available: http://www.chinacdc.cn/jkzt/crb/jhb/jhb_3868/201003/t20100322_24566.htm. Accessed July 6th, 2014

[2] Baidusection (2014) (Xinjiang Uygur Autonomous Region).Available: http://baike.baidu.com/subview/2824/14767092.htm?fr=aladdin#4_1. Accessed July 6th, 2014.

[3] Bureau of Health, Xinjiang Uyghur Autonomous Region (2014) (News release of the TB day in Xinjiang). Available: http://www.xjwst.gov.cn/zwgknry.jsp?urltype=news.NewsContentUrl&wbtreeid=913&wbnewsid=6941. Accessed July 6th, 2014.

[4] Wang YJ, Zhao TQ, Wang P, Li SQ, Huang Z, et al. (2006) Applying linear regression statistical method to predict the epidemic of hemorrhagic fever with renal syndrome. Chinese Journal of Vector Biology and Control 17: 333–334.

[5] Forecast Model Analysis for the Morbidity of Tuberculosis in Xinjiang, China

Yan-Ling Zheng, Li-Ping Zhang, Xue-Liang Zhang, Kai Wang, Yu-Jian Zheng, Published: March 11, 2015 https://doi.org/10.1371/journal.pone.0116832

[6] Long short-term memory, Sepp Hochreiter & Jurgen Schmidhuber. Neural Computation 9(8):1735{1780,1997. http://www.bioinf.jku.at/publications/older/2604.pdf

[7] LSTM: A Search Space Odyssey Klaus Greff, Rupesh K. Srivastava, Jan Koutn´ık, Bas R. Steunebrink, Jurgen Schmidhuber. https://arxiv.org/pdf/1503.04069.pdf

[8] A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction Abdelhadi Azzouni and Guy Pujolle LIP6 / UPMC; Paris, France {abdelhadi.azzouni,guy.pujolle}@lip6.fr. https://arxiv.org/pdf/1705.05690.pdf

[9] Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction Zhiyong Cui, Student Member, IEEE, Ruimin Ke, Student Member, IEEE, and Yinhai Wang. https://arxiv.org/ftp/arxiv/papers/1801/1801.02143.pdf

[10] L. Wei and Z. Zhen-gang, "Based on Time Sequence of ARIMA Model in the Application of Short-Term Electricity Load Forecasting," 2009 International Conference on Research Challenges in Computer Science, Shanghai, 2009, pp. 11-14.doi: 10.1109/ICRCCS.2009.12.URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5401272&isnumber=5401177

[11] Jun Zhang and K. F. Man, "Time series prediction using RNN in multi-dimension embedding phase space," SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218), San Diego, CA, USA, 1998, pp. 1868-1873 vol.2.doi:10.1109/ICSMC.1998.728168. URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=728168&isnumber=15679

[12] Nicole Fogel,Tuberculosis: A disease without boundaries, Tuberculosis,Volume 95, Issue 5,2015Pages 527-531,ISSN1472-9792, https://doi.org/10.1016/j.tube.2015.05.017.