Research on Qualitative Data Cleaning for IoT Streaming Data

Chen Binger

1. RESEARCH TARGET

This Research focus on qualitative data cleaning method for IoT(Internet of Things) streaming data, especially the characteristics of time series and the multidimensional data in WSN(Wireless Sensor Networks)[1].

2. BACKGROUND

IoT is dramatically changing our lives. The abundant data produced contains valuable information. WSN is a typical IoT instance which detects environment or events through multi-sensor data collection and communication in real time. It's widely applied on smart homes, smart vehicles, smart health and so on. Data mining for IoT can promote more intelligent services. IoT data is usually streaming data with temporal characteristics which can provide significant research values. It is one of the most studied data types. A recent survey by KDnuggets found that 48% of analysts analyzed time series data in the past, second only to table data[2].

Due to the low cost of sensors, limited resources, and unstable IoT environment, there're many dirty data such as missing data, anomaly data[3][4]. It's very common in time series. For example, the abnormality of sensor readings in the GPS track[5]. Direct use of these incomplete, noisy, inconsistent data in real world can lead to erroneous decisions and unreliable analysis in applications such as pattern mining[6] or classification[7]. Improving data quality to support following work is necessary[8]. Data cleaning is a key area of big data management.

The most common errors in time series are spike errors and consecutive errors (including missing values)[9]. The data cleaning methods can be either quantitative or qualitative. Quantitative data cleaning often involve statistical methods to identify abnormal behaviors and errors[10][11], while qualitative techniques are based on a series of data quality rules like integrity constraints[12]. This process usually leverage data mining or expert knowledge. There are few studies on qualitative data cleaning for IoT streaming data. And improvements should be made in existing methods considering the performance and costs. This will be addressed in our research.

3. PROBLEM STATEMENT & RELATED WORKS

There have been plenty of research on time series data cleaning but with drawbacks. This research is committed to solving these problems.

Problem I Qualitative data cleaning for streaming data. Although IoT streaming data cleaning has become a popular research area, most of the previous research focused on the quantitative level, such as outlier detection[13][14], neglect the qualitative level. Existing methods are mostly used for spatial data and not suitable for time series[15]. As for the research on time series data cleaning, the general method can only be used for offline data rather than online streaming data. Even if some studies aimed at online cleaning, they still didn't make full use of the temporal characteristics of time series for rule mining[9].

Problem II Precise repair dirty data after detecting. Many methods regard dirty data as noise and discard them after detecting[16]. However, such incomplete time series can lead to the unreliability of following research. Repairing the abnormal data to approach true value can improve the results of subsequent data processing[7][17]. Current attempts to streaming data repair are neither precise nor general[9].

Problem III Expand the types of data errors that can be fixed, especially missing values. Many methods can only solve certain kinds of data errors, some only deal with spike errors[18]. [9] can cover both spike errors and consecutive errors, but it cannot solve the problem of missing values.

Problem IV Improve performance and reduce complexity for multivariate time series cleaning. Many other studies of IoT streaming data cleaning can only clean one-dimensional data[12][19][20]. However, in WSN there're multiple sensors working together to obtain effective information. Some research have studied multidimensional data, but they assumed only one sensor reading is faulty at a time[9]. Some attempts based on DNN(Deep Neural Networks) have been made before, but these attempts didn't solve the above problems at the same time[21][22].

4. PROPOSED SOLUTION

Solution to Problem I Most quantitative data cleaning methods are based on statistical algorithm. To extract the qualitative features of time series, we plan to leverage DNN which are expert in understanding the data on semantic level. Specifically, to capture the temporal characteristics of streaming data, we adopt LSTM(Long Short-Term Memory), which is dedicated to mining the long-term dependencies of sequences and very suitable for time series data cleaning. It is widely used in sequence data, such as natural language processing.

Solution to Problem II We leverage an encoder-decoder based on LSTM to repair the dirty data, not only to detect them. After encoding the input sequence, we use the hidden states of encoder and the attention context to decode the corrected sequence. This method can be seen in many generation tasks like machine translation.

Solution to Problem III We construct a mask vector with time interval to record the position and the duration of missing values. This will be input to the model together with the original data in dual-channel fashion to utilize the pattern of the incomplete time series.

Solution to Problem IV Since the multivariate data has large dimensions and is difficult to extract patterns when directly fed into the networks, we incorporate CNN(Convolutional Neural Network) to generate features automatically with little preprocessing. It uses several convolutional layers to filter the large dimensional data, and sometimes together with the pooling layer to dramatically reduce the number of parameters. These advantages make CNN a suitable way to learn complex multi-sensor dirty data.

The architecture of the proposed solution is shown in Fig.1(**X**-Sensor Data, **M**-Mask Vector, t-Timestamp, δ-Time Interval, **C**-Attention Context). Data in current stream is input to the model in chronological order, together with the missing value mask vector and the time interval. They will first be extracted patterns and reduced to a fixed scale. Then a LSTM encoder will be implement on the sequence until the $\langle eos \rangle$ is processed. The LSTM decoder is conduct to generate the detected and repaired data based on the hidden states of encoder and the attention context. We simplify the data sample in the figure for making example.

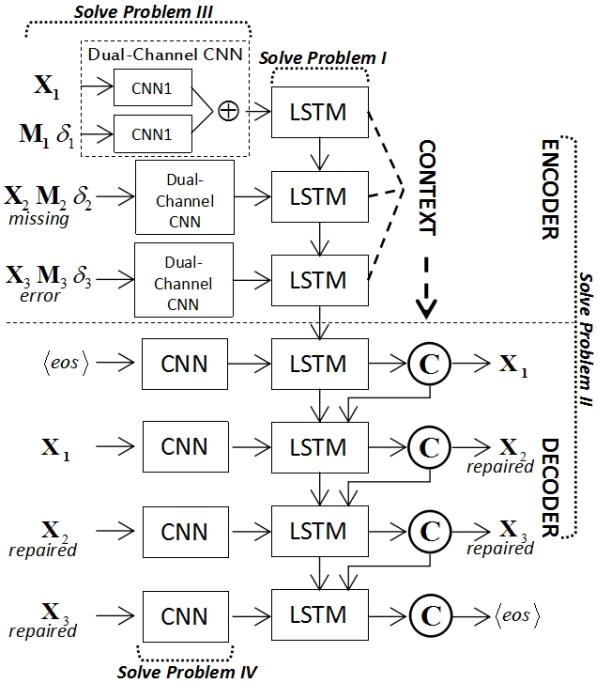

Figure 1. Solution Overview

4.1 Input Vector Construction

In order to extract more general patterns and do classification, the sensor data is vectorized. For example, if the value range of a sensor can be divided into [0, 1), [1, 2), [2, 3], when reads 1.5, it can be denoted as vector [0, 1, 0]. The missing value mask vector is constructed as follows: set 0 for sensor without value, 1 for others. And a time interval is added to record the duration of one reading. We design a dual-channel fashion: The vectorized sensor data will be input to the first channel and the mask vector with the time interval will be input to the second channel. As a result, the patterns of the original data and the missing value status will be extract in their specific models. We then concatenate the two channels' output for the next step processing.

4.2 CNN for Multivariate Data

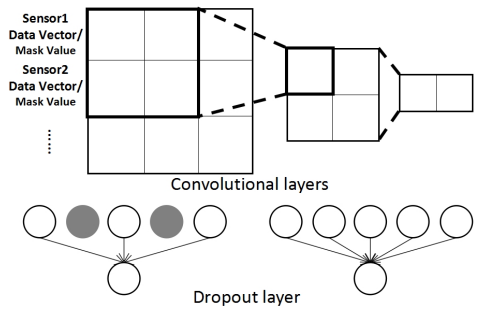

Figure 2. Schematic Diagram of CNN Model

Fig.2 is a schematic diagram of our CNN model. We train two CNN models for sensor data and missing value status separately to form a dual-channel CNN. The output of these two channels will be concatenated for LSTM encoding. In this research we use a regular two layers CNN to calculate the features of the original data sequence in a lower dimension space. However, the regular network is time-consuming and easy to over-fitting because of the huge numbers of parameters. We leverage dropout layer to counter this problem. It is a mechanism to shut down certain neurons with probability during the forward propagation process. None of the subnetworks are able to be repeated at each iteration. As a result, the model can accelerate the convergence rate and learn more generalized patterns. We choose the Bernoulli distribution to decide the neurons to be dropped.

4.3 LSTM Encoder for Time Series

LSTM is a variant of recurrent neural networks. It adds memory cell in the hidden layer, which can store the state for a period. This mechanism can memorize the long term dependencies of a sequence. Every time the LSTM unit reads a input, it will use the cell state and the previous hidden state to compute the current hidden state. There is also a forget gate to decide which memory should be abandoned. In this research we leverage the advantages of LSTM to encode our time series. The input of our LSTM encoder are the output of CNN described above and the hidden state of every timestamp. Then it outputs the hidden state $\mathbf{h}$ of current timestamp: $\mathbf{h}_i = \mathbf{LSTM}(\mathbf{h}_{i-1}, \mathbf{X}_i)$. The initial hidden state is set to zero. Moreover, the output of encoder will also be used as the context for every decoding step with attention mechanism.

4.4 Encoder-Decoder for Data Repair

We adopt a encoder-decoder to transform the original time series with dirty data to a repaired sequence. After we extract the features of the original data, we use it as a context for the decoder. The input of decoder are the decoded result and the hidden state from last timestamp, together with the attention context. After decoding, the repaired data is generated. The initial value is the hidden state from last step of encoder and $\langle eos \rangle$. The method to produce context vector $\mathbf{C}$ and the decoder hidden state $\mathbf{h}^d$ are as follows:

$$w_{j,i} = \frac{exp(\mathbf{h}^d_j \otimes \mathbf{h}_i)}{\sum_k exp(\mathbf{h}^d_j \otimes \mathbf{h}_k)} \tag{1}$$

$$\mathbf{C}_j = tanh(\sum_i w_{j,i}, \mathbf{h}_i; \mathbf{h}^d_j) \tag{2}$$

$$\mathbf{h}^d_j = \mathbf{LSTM}(\mathbf{h}^d_{j-1}, \mathbf{X}^d_j, \mathbf{C}_{j-1}) \tag{3}$$

5. OWN PREVIOUS WORK

April 2018 - present Data cleaning of smart TV user log(Association Rule Mining);
January 2018 - May 2018 End-to-end question answering network design for voice assistant based on deep learning(Attention-based CNN-LSTM Hybrid Model);
August 2017 - January 2018 Distributing anomaly detection and data reduction in IoT(Edge Computing);
August 2017 - October 2017 Anomaly detection in sensor data(Multi-Layer Perception);
March 2017 - June 2017 Human activity prediction based on time series data(Hidden Markov Model);
December 2016 - February 2017 Data feature extraction for human activity recognition(SVM);
September 2015 - June 2016 Time series data analysis of astronomical detecting sensor(Hidden Markov Model).

6. EXPECTED OUTCOME

This research is expected to build a well-performed data cleaning model based on DNN. The CNN feature extractor and The LSTM encoder-decoder are well convergence. The model is well fit the datasets. Our reliable system can detect and repair current data stream immediately. It is universal and compensates the drawbacks of previous studies. The repaired data are extremely close to true value and improve the performance of subsequent research. Our research also has practical significance. It can solve emerging IoT issues like cleaning the multi-sensor data from Internet of Vehicles which may promote the autonomous driving technology.

7. DATA

7.1 Dataset Description

There are plenty of opensouce datasets for multivariate time series. Considering we are trying to implement a deep learning method, we need a dataset with sufficient data amount to display the advantage of deep learning. In our scenario, we focus on the data cleaning in IoT environment. Therefore, the dataset we choose is better to be generated from sensor data. For these reasons, we plan to experiment on two datasets: Intel Lab Data(ILD)[23] and PhysioNet dataset[24].

ILD Data This dataset contains data from 54 sensors deployed in a research lab. Each sensor can collect timestamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. There are about 2.3 million readings collected from these sensors. We plan to choose data from one sensor for experiment since each sensor's reading is multivariate.

PhysioNet This dataset is from PhysioNet Challenge 2012. It is a clinical multivariate time series dataset which contains data from intensive care unit records. Each record collect values of 37 variables like blood pressure, heart-rate and temperature. Measurements were recorded at regular intervals ranging from hourly to daily, or at irregular intervals as required. The dataset contains about 8000 records from 48 hours. We plan to choose 3 to 5 variables for experiment.

7.2 Synthetic Errors

To the best of our knowledge, there are few annotated multivariate time series for data cleaning task. We plan to introduce synthetic errors to evaluate our mode more conveniently. We plan to shift the values based on Gaussian distribution with different parameters to create multiple samples. And for the missing value situation we address in this research, we plan to randomly choose a missing rate from a uniform distribution.

In addition, it is probably significant if we experiment on different kinds of errors to evaluate the robustness of our model. For example, the level shift and innovational errors[25].

7.3 Data Preprocessing

Since the ILD data is the original sensor data, whose quality is not guaranteed. That means there are noise and missing values in this dataset. We plan to downsample the original time-series to obtain non-overlapping sequences to avoid dirty data. The length of new sample need to be further discussed.

We plan to divide the data set into training set, developing set and testing set by the proportion of 6:2:2.

To train a well-performed deep learning network, the original data need to be transformed into vectors. We segment the value range of each variable into several intervals, and construct a 0-1 vector whose length is equal to the interval amounts. We implement data discretization on each value in the dataset, and the input of the network will be a 0-1 matrix.

8. IMPLEMENTATION

An overview of our implementation sketch is illustrated in Figure 3, which can be found at the end of this proposal.

8.1 Library

Since python is the widely used language in deep learning domain, and it has the largest communities and toolkits for scientific calculations. We choose python as our development language. Moreover its syntax is quite friendly compared to others.

PyTorch We plan to choose PyTorch as our library temporarily. It's originally designed for natural language processing by Facebook. Considering the similarity of natural language and streaming sequence, it works on our problem either. The biggest advantage of this library is it can generate computational graph dynamically and it can clearly print the intermediate values. We can make changes to our model pleasantly. And the autograd in PyTorch can save our time from writing back propagation. PyTorch is a very appropriate library for developing a prototype.

TensorFlow It is a critically acclaimed library developed by Google. It uses graph to describe the computing, which is easy to build a complex network. And it can run with multiple GPUs parallelly and make visualization with TensorBoard. But the static computational graph makes it hard to modify. Whether to employ this library needs to be further discussed.

8.2 API Sketch

Parameters needs to be set:

Input vector dimension
GPU use=True/False
CNN: batch size, number of filters, filter width
LSTM: sequence length, batch size, number of layers=4, hidden units
Dropout probability=0.5
Initial learning rate=1.0
Number of epochs

API Sketch

```
#divide the value range to several intervals
#(histogram analysis[16])
def discretization(data_sample)

def preprocess():
    downsampling(original_data)
    value2vec(data) #vectorize the data based on discretization
torch.save(data_cache), torch.load(data_cache)

#construct the network with PyTorch
class Net(torch.nn.Module):
    #dual-channel CNN
    torch.nn.Conv2d(in_channel=2,out_channel,kernel_size=
        (vec_dim,filter_width),bias=True)
    torch.nn.Dropout(p=0.5) #dropout layer
    def conv_layers(self, x) #define activation function in CNN
    ————————-
    #LSTM
    torch.nn.LSTM(input_size,hidden_size=vec_dim,num_layers
        =4,bias=True,dropout=0.5,batch_first=True)
    ————————-
    #fully-connected layer
    torch.nn.Linear(inputDimension,outputDimension)
    ————————-
    def forward()
    ————————-
    #encoder-decoder
    #we plan to follow the work of OpenNMT,
    #which is a high-performance LSTM encoder-decoder
    #for machine translation[26][27]
    encoder = onmt.encoders.RNNEncoder(hidden_size,
        num_layers=4,rnn_type="LSTM",bidirectional=False,
        embeddings=CNN_output)
    decoder = onmt.decoders.decoder.InputFeedRNNDecoder(
        hidden_size,num_layers=4,rnn_type="LSTM",
        bidirectional=False,embeddings=decoder_embeddings)
    model = onmt.models.model.NMTModel
        (encoder, decoder)
```

```
#train the model
def train(net,data,param):
    loss=torch.nn.CrossEntropyLoss() #loss function
    optimizer=torch.optim.SGD(net.parameters(),lr=
        learning_rate,momentum)
    loss.backward()
    #gradient clipping to avoid gradient explosion
    torch.nn.utils.clip_grad_norm()
    optimizer.step() #update the parameters
    torch.save(model)

#test the model
def test(net,data,param)

#calculate criteria
def RMSE(true_data,repaired_data)
```

8.3 Probable Problems and Solutions

Convergence Sometimes the model is hard to converge. This may be caused by the software bugs or the hyperparameters we set are not appropriate, such as the learning rate. We can fine tune the hyperparameter manually or automatically. Manual tuning is mostly based on the experience and knowledge. Automatically tuning is easier, we can leverage the grid search or random search to find the optimal hyperparameters. But it will raise the computing cost.

Overfitting Sometimes the model works well on training set, but doesn't work well on testing set. Its probably the overfitting problem. We will confirm this by validate the model every certain time with developing set. If the accuracy is far different from the training set, we consider the model is tend to be overfitting. To solve this problem, we will try to decrease the amount of layers or neurons to limit the fitting ability, such as leverage the dropout mechanism. We can also early stop the training. And the regularization is also a common method. It adds the weight to the cost function in the form of L1/L2-norm to limit the growth of the weight during training.

Software Defects Sometimes the model may not work well both on training set and testing set. It may be caused by unfitting or software defects. A method to figure this out is using the model to fit a small sample of data, if it still cant work, we will focus on the software defects.

Processing Time A biggest problem of deep learning method is the longer processing time. Although our model is for scientific research, the processing time may not as long as a commercial product, we still need to improve the performance as best as we can. For example, we can ditch some intermediate values to save the cache space, or we can save a set of intermediate parameters to avoid repeated computing. We can also employ some network compression algorithms. These need to be further considered in practice.

9. EVALUATION

9.1 Benchmarks

In the experiment, we plan to compare our method with several state-of-the-art methods. We choose AutoRegressive with External input (ARX)[28], Exponentially Weighted Moving Average (EWMA)[29], Stream Data Cleaning under Speed Constraints (SCREEN)[20] as our benchmarks. Each of them represent one kind of data cleaning approach. There is a new well-performed methods proposed by [18]. We haven't found the source code. We will try to reproduce the code to further evaluate our method if possible.

9.2 Evaluation Criteria

- We will plot the original time series and the repaired time series by our method and the benchmarks in one coordinate system, which can illustrates the general performance of the method.
- We choose Root Mean Square Error (RMSE) as our prime criteria. We will calculate the RMSE for each variable. In addition, we will plot the relation between RMSE and number of consecutive error. We will also plot the relation between RMSE and the amount of dirty samples.
- As a deep learning method, we also choose some typical criteria in deep learning field, such as the F1-score, which considers both the precision and the recall of a deep learning model.

9.3 Evaluation on Repairing Missing Values

To further evaluate our method on repairing the missing values, which is one of the main tasks in this research, we will plot the relation between RMSE and the frequency of missing data.

9.4 Evaluation on Multivariate Cleaning

Since multivariate time series is our experiment object, we will change the number of variables with dirty data at each timestamp, then analyse its impact on the final precision.

9.5 Online Evaluation

We will implement online evaluation for streaming data. We plan to plot RMSE-Sampling Time and RMSE-Sampling Size to evaluate online and offline performance respectively. We also plan to analyse the Throughput value of our online data cleaning system to test its practicability.

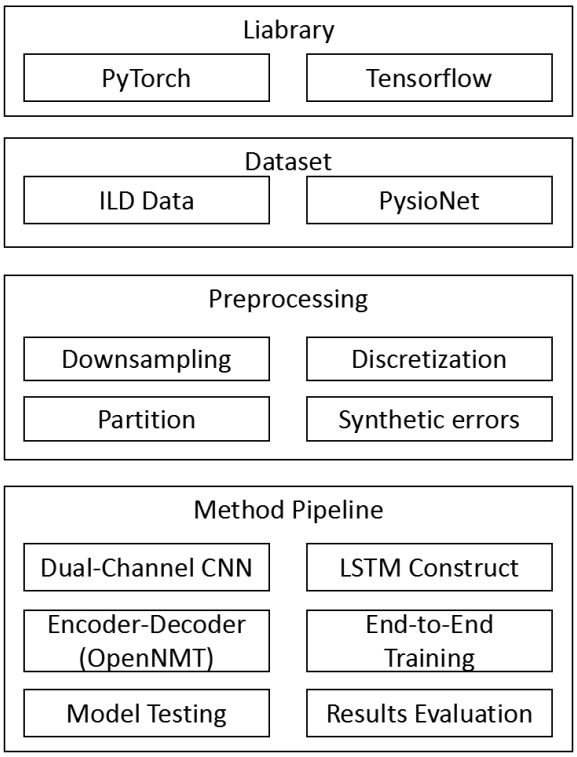

Figure 3. Implementation Overview

REFERENCES

[1] Tsai, chun-wei, et al. data mining for internet of things: A survey. *IEEE Communications Surveys and Tutorials*, 16(1):77–97, (2014).

[2] Piatetsky-shapiro g. url retrieved on april-2-2018. https://www.kdnuggets.com/polls/2014/data-typessources-analyzed.html, (2018).

[3] C batini, m scannapieco. data quality: concepts, methodologies and techniques. *Springer Publishing Company*, (2010).

[4] D ganesan, s ratnasamy, h wang, et al. coping with irregular spatio-temporal sampling in sensor networks. *ACM Sigcomm Comput Communication Rev*, 34(1):125C130, (2004).

[5] P. j. brockwell, r. a. davis.introduction to time series and forecasting. *Springer Science & Business Media*, (2006).

[6] Moerchen, fabian. algorithms for time series knowledge mining. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2006).

[7] Xing, zhengzheng, jian pei, and s. yu philip. early classification on time series. *Knowledge and information systems*, 31(1):105–127, (2012).

[8] Karkouch, aimad, et al. data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications*, 73:57–81, (2016).

[9] Nemati, hassan, et al. stream data cleaning for dynamic line rating application. *Energies*, 11(8), (2018).

[10] Hodge vj, austin j. a survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85C126, (2004).

[11] Aggarwal cc. outlier analysis. *Springer*, (2013).

[12] Chu x, ilyas if, papotti p. holistic data cleaning: Putting violations into context. In *In 2013 IEEE 29th International Conference on Data Engineering (ICDE)*, (2013).

[13] S. madden. database abstractions for managing sensor network data. *Proceedings of the IEEE*, 98(11):1879C1886, (2010).

[14] A. deligiannakis, y. kotidis, v. vassalos, v. stoumpos, and a. delis. another outlier bites the dust: Computing meaningful aggregates in sensor networks. In *IEEE International Conference on Data Engineering*, (2009).

[15] M. volkovs, f. chiang, j. szlichta, and r. j. miller. continuous data cleaning. In *IEEE International Conference on Data Engineering*, (2014).

[16] Han j, kamber m, pei j. data mining: Concepts and techniques. *Elsevier: New York*, (2011).

[17] S. song, c. li, and x. zhang. turn waste into wealth: On simultaneous clustering and cleaning over dirty data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM*, (2015).

[18] Zhang a, song s, wang j, yu ps. time series data cleaning: from anomaly detection to anomaly repairing. *Proceedings of the VLDB Endowment*, 10(10):1046–1057, (2017).

[19] Bohannon p, fan w, flaster m, rastogi r. a cost-based model and effective heuristic for repairing constraints by value modification. In *In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, (2005).

[20] Song s, zhang a, wang j, yu ps. screen: Stream data cleaning under speed constraints. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, (2015).

[21] Kieu, tung, bin yang, and christian s. jensen. outlier detection for multidimensional time series using deep neural networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, (2018).

[22] Krishnan, sanjay, et al. boostclean: Automated error detection and repair for machine learning. In *arXiv preprint arXiv:1711.01299*, (2017).

[23] http://db.csail.mit.edu/labdata/labdata.html, ild.

[24] https://physionet.org/challenge/2012/, physionet.

[25] R. s. tsay. outliers, level shifts, and variance changes in time series. *Journal of forecasting*, 7(1):1–20, (1988).

[26] Minh-thang luong, ilya sutskever, quoc le, oriol vinyals, and wojciech zaremba. addressing the rare word problem in neural machine translation. In *In Proc of ACL*, (2015).

[27] http://opennmt.net/opennmt-py/main.html, opennmt-py.

[28] Box, george ep, et al. time series analysis: forecasting and control. In *John Wiley & Sons*, (2015).

[29] Hellerstein, j.m., 2008. quantitative data cleaning for large databases. In *United Nations Economic Commission for Europe*, (2008).