PastProp-RNN: improved predictions of the future by correcting the past

André Baptista
up201505375@fe.up.pt
Faculdade de Engenharia,
Universidade do Porto

Yassine Baghoussi
baghoussi@fe.up.pt
INESC TEC, Faculdade de Engenharia,
Universidade do Porto

Carlos Soares
csoares@fe.up.pt
Fraunhofer AICOS and LIACC,
Faculdade de Engenharia,
Universidade do Porto

Miguel Arantes
miguel.arantes@inovretail.com
Inovretail

João Mendes-Moreira
jmoreira@fe.up.pt
INESC TEC, Faculdade de Engenharia,
Universidade do Porto

ABSTRACT

Forecasting accuracy is reliant on the quality of past available data. In turn, data quality is affected by the occurrence of anomalies (e.g., reduced sales due to stock shortage). We address this problem by *pastcasting*: predicting how data should have been in the past for it to explain the future better. We propose *PastProp-RNN*, an adaptation of the backpropagation algorithm used by LSTM that assigns part of the responsibility for errors to the training data and changes it accordingly. We test three variants of PastProp-RNN on artificial and benchmark data, as well as on a case study from the retail domain. Our results indicate that the proposed method not only is able to reconstruct data that was affected by anomalies but also to improve the forecasting accuracy when compared to a standard LSTM. Additionally, ARIMA seemed to benefit from learning with data corrected by our method.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

pastcasting, recurrent neural networks, time series forecasting

ACM Reference Format:
André Baptista, Yassine Baghoussi, Carlos Soares, Miguel Arantes, and João Mendes-Moreira. 2020. PastProp-RNN: improved predictions of the future by correcting the past. In *MileTS '20: 6th KDD Workshop on Mining and Learning from Time Series, August 24th, 2020, San Diego, California, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Demand forecasting has been recognized as a key process in retail for a long time. In earlier days, companies would solely rely on the knowledge and opinions of experts to predict future demand. Eventually, expert-based approaches were no longer satisfactory due to the increasing volumes of data involved. As a result, statistical and machine learning methods got involved [4]. More recently, deep learning techniques have been applied due to their ability to model complex patterns in long time series. A common concern to any forecasting method is the quality of the available data. In retail, past sales data are used to estimate future demand. However, sales may sometimes be unrepresentative of the demand. For example, when a product goes out of stock for a time period, the registered sales for that product will not take into account the number of lost sales opportunities. This is known as *censored observation* [9]. In other words, the observed value of sales is limited by the available stock (i.e. an upper bound), and the true demand is outside the measurable range. This type of anomaly effectively decreases the quality of the input data. Another example of anomalous data is the temporary closing of a store for some extraordinary reason. During that time period, there will be no recorded sales. Therefore, those zero values will probably be bad predictors of demand when the store reopens. Finally, an example of what would not be considered as an anomaly is the sales increase during the Christmas season. The reasoning behind this is that, despite constituting an outlier, the phenomenon is expected and likely helpful at explaining future time series values.

Motivated by these issues, we address the general problem of making changes to past data to make it a better predictor of the future. This is referred to as *pastcasting* [2, 7]. We propose *Pastprop*, which consists of modifying the backpropagation algorithm of an LSTM implementation with the goal of correcting anomalies in data and reducing their effect on forecasting accuracy. The idea is to make the learning process estimate the contribution of training data to the outputted errors. In other words, the responsibility for errors is shared not only among network weights but also with the data. Hence, instances of training data should also be updated in the direction that minimizes error.

2 RELATED WORK

The problematic of demand forecasting in the presence of censored sales data has been addressed in the literature. [10] tackles the problem through the use of ensemble methods. In the study, machine learning methods were combined while accounting and not accounting for censorship in data. [1] studied the issue while using

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MileTS '20, August 24th, 2020, San Diego, California, USA
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

sales data of the same products from multiple retail stores. Their method includes modeling the problem with a "latent variable" and using "matrix completion" related techniques [1]. Their results argue that the demand forecasting error would tend to zero as the number of considered stores and timestamps increased to infinity. [8] addresses the problem while considering that the retailer's stock levels might not be accurately known at all times. They concluded that ignoring the possibility of "inventory record inaccuracy" in the presence of censored sales data usually leads to a "systematic underestimation of demand" [8].

2.1 LSTM

Long-Short Time Memory (LSTM) is a type of Recurrent Neural Network (RNN). RNNs differ from feed-forward neural networks by the recurrent connections which allow them to learn from sequential data. They attempt to model and remember temporal dependencies in sequences. However, RNNs have been criticized due to the vanishing and exploding gradient problem [5]. In brief, RNNs are not suited to learn long dependencies in time series data. In fact, they are not effective at learning relationships more than 5 to 10 time steps apart in data [11]. To solve this issue, LSTM, a new architecture, was proposed by [6].

The LSTM architecture is composed of blocks, each of them containing an input gate, output gate and memory cell. A gate is set of processing units with *sigmoid* activation functions whose inputs are scaled by weights. There is also an input modulation gate which contains a *tanh* activation function instead. The memory cell holds hidden information resulting from previous inputs. The modulation gate learns to scale input features according to their importance and then squashes them between -1 and 1. Moreover, the purpose of the input gate is to regulate which of those features should used to updated the cell's state. The output gate filters which information from the cell should be passed to the next block. A forget gate was later proposed [3] to learn which data should be discarded from memory even before updating it with new information from the current input.

Similarly to RNN, LSTM learns by forward and backward passes. In the forward pass, the input is concatenated with an output from the previous block and a bias value. That information is processed within the LSTM block and then goes through a final layer of weights to obtain the predicted outputs. Each of them is compared to the real outputs and an error measure is obtained. The backward pass takes that error and applies a gradient descent algorithm to update the weights of each gate.

3 PASTPROP

We propose a novel idea of reworking the backpropagation algorithm so that error responsibility is extended to data. As such, similarly to the network weights, input data should be corrected in the direction that minimizes error, with a magnitude proportional to its contribution to that error. The goal is that the corrections improve the overall quality of the training data. In particular, we expect them to be effective at reconstructing anomalies. The premise is that anomalies can be viewed as sections of data with high responsibility for errors. Therefore, they should theoretically be subjected to more significant changes. The idea could potentially be applied to other neural network architectures, which also learn through backpropagation. However, our work is focused on LSTM only since they are well suited for time series forecasting - our task of interest.

3.1 PastProp Variants

The following subsections explain, in higher detail, the implementation of three different pastprop variants. All of them derive from a publicly available Python implementation of an LSTM [1]. Furthermore, it is assumed that the algorithms receive an univariate time series as training data. During the learning process, multi time step samples are used to predict multi time step labels. At the end of training, the outputs are both the network's weights and the corrected time series.

3.1.1 Calculating deltas. The key variables used by pastprop are the deltas. In a similar manner to how the input, forget and output gate's gradients are obtained, we can also calculate a gradient for the LSTM's hidden input at each block. This is the expression we are looking for:

$$\frac{\partial Error_t}{\partial hin_t} = \frac{\partial Error_t}{\partial (hin_t W_i)} W_i + \frac{\partial Error_t}{\partial (hin_t W_f)} W_f + \frac{\partial Error_t}{\partial (hin_t W_o)} W_o + \frac{\partial Error_t}{\partial (hin_t W_g)} W_g \tag{1}$$

Since the hidden input is composed of the input sample, previous hidden output and a bias value

$$hin_t = \{X_t, hout_{t-1}, b\} \tag{2}$$

We subset the gradient to just the part that respects to the input sample.

$$\frac{\partial Error_t}{\partial hin_t}[X_t]$$

The deltas to be added to the input samples are calculated from this gradient and the "data correction rate" constant, a Pastprop specific hyperparameter which serves as a learning rate for data instead of weights.

3.1.2 Regular Pastprop. In this variant, the corrections are applied to the whole time series - all at once - after the completion of each epoch. During an epoch, the "deltas" to be added to each data sample are calculated and stored until the end of the epoch. Right before starting the new epoch, the deltas and time series are added together. This means that every epoch deals with a different version of the data. It also means that the first epoch produces exactly the same weights that a normal LSTM would (in case the initial weights are the same). Figure 1 illustrates the described mechanism in a simplified way. We can see that most of the time steps have overlapping deltas associated with them. In those cases, an average of the corresponding deltas (not their sum) is used to build the final corrections array. This is done to make the amount of data corrections less dependent on the sample size.

[1] https://gist.github.com/karpathy/587454dc0146a6ae21fc

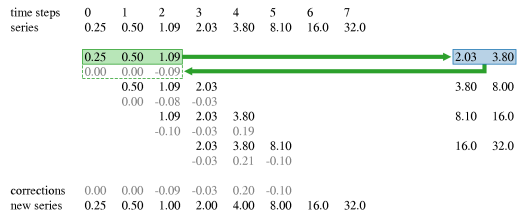

Figure 1: Simplified illustration of how regular pastprop works

3.1.3 Progressive PastProp. This variant differs from the "Regular" one in the sense that deltas are added to data over the course of epochs. As soon as they are obtained, deltas are first adjusted to compensate for overlapping and then added to samples.

Contrary to the first variant, each epoch deals with progressively different time series. For this reason, the weights differ from those produced by an LSTM right from the first epoch.

3.1.4 Selective Pastprop. This variant is built upon the "Regular" variant and introduces two new features. The first is an hyperparameter that dictates the amount of initial epochs to wait before applying data corrections. The second feature is a thresholding mechanism. At the end of an epoch, when the final corrections array is ready, the deltas are ranked by the average of theirs and their neighbours' absolute value. Higher values translate to higher ranks. The highest ranked deltas prevail while the other ones are changed to zeros (i.e. they will have no effect on data). A threshold parameter is used to establish the amount of deltas to be preserved. Furthermore, the number of previous and subsequent time steps that define a delta's neighborhood is also required as a parameter. In case that number is zero, each delta is ranked solely by its own absolute value. Keeping the time series unchanged for the first few epochs is an attempt at reducing the impact of the LSTM's random weight initialization on data corrections. Moreover, the inclusion of neighbor deltas in the ranking system incentivizes the preservation of deltas associated with anomalous zones. In turn, single point anomalies are not as targeted by the system.

4 EXPERIMENTAL SETUP

The LSTM related hyperparameters required for the pastprop variants are the: sample size, label size, the number of hidden units for each LSTM gate, learning rate, and the number of epochs.

The only pastprop specific hyperparameter, common between all variants, is the data correction rate. The Selective variant also needs the number of waiting epochs, the number of highest ranked deltas to keep on the corrections array, and the number of previous and subsequent time steps to consider when ranking deltas.

Anomalies were generated and characterized by the size, position in the time series, and magnitude. Since their size and position are self-explanatory, only the magnitude attribute is explained here. Three different levels of magnitude were considered. Level 0 substitutes the data by a sequence of zeros. The remaining two levels function differently. The anomaly zone is divided by equally sized chunks. It is randomly chosen whether each chunk should add or subtract from the original data. At a given time step, the value of the change is the maximum between 0.1 and a percentage of the time series' original value. The possible percentages are 25 and 50, which correspond to the magnitude levels 25 and 50. The minimum absolute change value of 0.1 was chosen in the context of data being normalized between 0 and 1. For a given time series and a complete set of parameters, a single experiment consisted of performing multiple operations. The first step was to generate random initial weights compatible with the problem at hand. For the sake of fairness in comparison, all tested methods worked with these same starting weights. The methods include a simple LSTM, all three pastprop variants, and also three additional LSTMs. The latter used the series corrected by the pastprop variants as training data. After execution, besides forecasting accuracies on test data, a few other results were kept. They included similarity measures between each corrected series and the original data, both on the anomaly zone and outside of it. With robustness in mind, the whole experimental procedure was repeated 5 times. The final results attributed to that combination of dataset and parameters were the average results of those runs. Additionally, we compared baseline algorithms including Autoregressive integrated moving average (ARIMA) and Exponential smoothing state space model (ETS) with Pastprop variants on anomalous data. ARIMA and ETS were applied to the corrected data produced by Pastprop. The results were then compared to those obtained earlier while using the original anomalous data.

4.1 Datasets

The research was performed on 9 different time series. The following subsections describe how the data was obtained and what hyperparameters were tested with them. A few parameters were kept constant throughout all experiments. For all presented datasets, the training data consisted of the first 70% of the time series, and the last 30% were used for testing. Furthermore, the learning rate and the number of hidden units were always 0.001 and 20 resp. Lastly, all series were normalized between 0 and 1.

4.1.1 Artificial data. The only artificially generated time series is a sinusoidal function with values ranging from 0 to 1. Its total length is of 500 time steps. An intensive study on hyperparameters combinations and their effect on the results has been conducted on this dataset. The values used for this study follows:

- epochs: 50, 200, 100 (respective waiting epochs were 10, 20, 50)
- data correction rate: 0.001, 0.01, 0.1
- anomaly size: 25, 50, 1000 time steps
- anomaly position: 5 different positions evenly spaced throughout the training data plus non-occurrence of anomaly
- anomaly magnitude: level 0, level 25, level 50

The sample and label sizes were of 50 time steps. Considering all possible combinations of these parameters and the definition of a single experimental setup, a total amount of 486 experiments were carried out on the artificial data.

4.1.2 Benchmark data. A total of 6 benchmark time series were used, originated from 2 publicly available datasets. The M5 competition dataset [2] was processed with the goal of extracting three time

[2] https://mofc.unic.ac.cy/m5-competition/

series, each representing the daily unit sales of a specific product category (Hobbies, Foods, and Household) registered in Walmart stores located in the USA. Another electricity consumption dataset from the UCI Machine Learning Repository [3] has been used. Three time series with a length of 1462 time steps has been used.

For all the benchmark data, the sample and label sizes were of 28 time steps. Regarding anomaly generation, a size of 56 time steps was used. Two magnitude levels were considered: level 0 and 50. The anomalies were inserted at the middle of training data. Lastly, the number of epochs were 50, 200 and 1000 while the data correction rates were 1e-3, 1e-2 and 1e-1.

4.1.3 Retail case study data. Data from an international fashion retail case study was also used. The main time series consisted of daily combined sales of two popular articles during a full-year period. Alongside this data, auxiliary time series were used, such as the daily average temperature at the store's location, precipitation amounts, stock levels, and basic calendar information.

The hyperparameters of the experiments done with these data were similar to the ones used for benchmark data. Instead, the sample and label sizes were 14 time steps, and the anomaly size was 28 time steps.

5 RESULTS DISCUSSION

5.1 Anomaly Reconstruction

The anomaly reconstruction ability was calculated by resorting to two variables: the MSE between the corrections and the original data; the MSE between the anomaly and the original data. The formula is obtained as follows:

$$\text{reconstruction ability} = 1 - \frac{\text{MSE}(\textit{corrections}, \textit{original})}{\text{MSE}(\textit{anomaly}, \textit{original})} \quad (3)$$

As such, a positive reconstruction ability indicates the corrections were able to push the anomalous zone closer to the original data. On the other hand, a negative value means the anomaly was accentuated even further.

The MSE between corrections and original data was also measured while excluding the anomalous part. This will be referred to as the "outside loss". Especially on the artificial data, where the series follows a well defined curve, there is no apparent reason to change data outside of the anomaly. Therefore we want this value to be as low as possible. When considering other types of data, this measure loses some of its meaning since corrections may be beneficial even outside of the anomaly. Finally, the measure of "outside loss" only exists when there are manually inserted anomalies in data.

Assuming the anomaly occurs in the middle of the training data, the global results on artificial data indicate a positive reconstruction ability of roughly 10% when using the Selective pastprop variant. Meanwhile, the Regular and Progressive variants showed a negative ability of 41% and 46%, respectively. However, the results differ when sorting the experiments by the product of epochs and data correction rate and then ignore the ones at the higher end of this spectrum. For instance, ignoring the pairs (1000; 1e-2), (200; 1e-1), and (1000; 1e-1), the reconstruction ability of both the Regular and Progressive variant increased to positive 32% while the Selective variant's ability only slightly dropped to 9%. The results assess that the Regular and Progressive variants are not tolerant to a high number of epochs and data correction rate. The average outside losses of the first two variants were of 0.007, while the Selective variant was just 0.001.

[3] https://archive.ics.uci.edu/

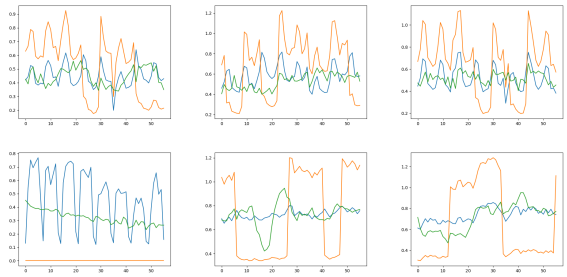

Figure 2: Anomaly reconstruction examples on benchmark data. Blue is the original time series, green is the reconstructed and orange is the anomaly.

Moreover, on the remaining benchmark and case study data, the global averages were of 4% reconstruction ability in the first two Pastprop variants and 1.5% in the Selective one. When ignoring the experiments with data correction rate equal to 1e-1, we obtain the following anomaly reconstruction abilities: 13.3% in Regular Pastprop; 12.96% in Progressive Pastprop; 0.86% in Selective Pastprop. The corresponding outside losses were 0.00662, 0.00683 and 0.00023. The previous experiments on artificial data suggested that the Regular and Progressive variants were undermined by the usage of high number of epochs combined with high data correction rates. However, in the benchmark and case study experiments, the data correction rate alone seemed to be the deciding factor, since using a value of 1e-1 lead to negative reconstruction abilities. The best results were obtained with (1000; 1e-2). Figure 2 shows some instances of anomaly reconstruction while using that combination of hyperparameters on benchmark data. It should be pointed out that the achievement of such reconstruction abilities was accompanied by very high outside losses, that is, too much change outside the anomaly zones focused in Figure 2.

5.2 Forecasting Accuracy

To investigate the impact of pastprop in the forecasting accuracy, all M5 and electricity time series were subjected to a 56 time step long anomaly in the middle of their training part (the first 70% of the series). The magnitudes applied were of level 0 and 50. Then, the baseline algorithms were given the full training data to perform forecasts at 28 time step maximum horizon. Immediately after, one time step was advanced into the future, and another forecast was made, now using the knowledge of extra real value. The process was repeated until the last forecasted value coincided with the end of available test data. Finally, multiple series were built with predicted values at specific horizons. For example, we considered 3 series where the values that constitute them were predicted at an horizon of 1, 7 and 28 days. This was the same forecasting mechanism adopted for LSTM and Pastprop. Forecasting accuracies were computed using mean squared error (MSE).

Table 1: Forecasting accuracies (MSE) at 1, 7 and 28 day horizons obtained by LSTM, Pastprop, ARIMA and ETS on univariate benchmark data (3 M5 series and 3 electricity series).

Horizon	Hyperparameters	LSTM	Regular	Progressive	Selective	ARIMA	ETS
1	(50; 1e-3)	0.02462	0.02775	0.02970	0.02790	0.01097	0.01351
	(50; 1e-2)	0.02729	0.03041	0.02458	0.02705		
	(50; 1e-1)	0.02615	0.02214	0.02303	0.03133		
	(200; 1e-3)	0.03994	0.06429	0.05774	0.05396		
	(200; 1e-2)	0.03665	0.03445	0.04312	0.05807		
	(200; 1e-1)	0.03930	0.04343	0.03777	0.03393		
	(1000; 1e-3)	2.06349	2.22513	6.98732	6.31422		
	(1000; 1e-2)	3.62515	1.99246	3.07272	7.89312		
7	(50; 1e-3)	0.02486	0.02396	0.02920	0.02722	0.01765	0.01679
	(50; 1e-2)	0.02836	0.02460	0.02526	0.02512		
	(50; 1e-1)	0.02186	0.01971	0.02050	0.02840		
	(200; 1e-3)	0.03810	0.07618	0.05311	0.04773		
	(200; 1e-2)	0.03349	0.03421	0.03418	0.05699		
	(200; 1e-1)	0.03457	0.04138	0.03591	0.03519		
	(1000; 1e-3)	2.20410	2.09030	7.33739	6.0169		
	(1000; 1e-2)	4.10815	1.62792	3.42391	7.81961		

Table 1 compares the forecasting abilities of Pastprop to LSTM, ARIMA and ETS at three different horizons: 1, 7 and 28 days. Following MSE results, higher number of epochs seems to always translate to worse results. Performances using 50 and 200 epochs were somewhat acceptable (even though worse than ARIMA and ETS), but using 1000 epochs produced extremely bad results. This might be related to our specific implementation of LSTM which does not use a *sigmoid* function at the output layer. Instead, the *hidden output* is simply multiplied by the output layer's weights to obtain the predicted values. Since forecasts are not filtered by sigmoid, they are not restricted to any scale. In our case, due to the high number of epochs, the weights increased to a point that resulted in forecasts no longer being in the interval of 0 to 1, hence the low accuracies. As a results, Horizon 7 displayed the best results. It produces, on average, 7% and 7.5% better accuracies compared to horizon 1 and horizon 28 respectively. For (50; 1e-2) and (50; 1e-1), Regular and Progressive were, on average, 9% and 13.3% better than LSTM resp., and Selective was 3.8% worse.

As to why ARIMA and ETS outperformed the other methods, there might be several reasons. More fine-tuning of the hyperparameters, such as the number of hidden units and learning rate, could be necessary. These were kept constant to prevent the total amount of experiments from exploding. Additionally, the base implementation of LSTM we used was the most "vanilla" possible so that implementing Pastprop would be more straightforward.

Table 2: Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by ARIMA when using data corrected by each Pastprop variant using (200; 1e-1) vs. ARIMA with the non corrected data.

Pastprop	mean	std	median	min	max
Regular	7.04%	8.74%	7.02%	-6.09%	21.70%
Progressive	8.04%	12.60%	2.89%	-4.08%	27.93%
Selective	13.25%	9.62%	9.52%	3.48%	29.54%

5.3 Learning with corrected data

Table 2 illustrates the improvements achieved on ARIMA after correcting univariate benchmark data (3 M5 series and 3 electricity series) using Pastprop variants. The data were not affected by anomalies and the horizon of the forecasts was 7 days. Results show that corrections produced by Pastprop variants have contributed to the improvement of ARIMA performance. It is worth noting that, in some datasets, corrections had a negative impact on the forecasts.

REFERENCES

[1] Muhammad J Amjad and Devavrat Shah. 2017. Censored demand estimation in retail. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–28.

[2] Brian Deal, Haozhi Pan, Stephanie Timm, and Varkki Pallathucheril. 2017. The role of multidirectional temporal analysis in scenario planning exercises and Planning Support Systems. *Computers, Environment and Urban Systems* 64 (2017), 91–102. https://doi.org/10.1016/j.compenvurbsys.2017.01.004

[3] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. (1999).

[4] Suleka Helmini, Nadheesh Jihan, Malith Jayasinghe, and Srinath Perera. 2019. Sales Forecasting Using Multivariate Long Short Term Memory Networks. *PeerJ Preprints* May (2019), 16. https://doi.org/10.7287/peerj.preprints.27712
[5] Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München* 91, 1 (1991).
[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[7] Hyun Hak Kim and Norman R. Swanson. 2017. Methods for Pastcasting, Nowcasting and Forecasting Using Factor-MIDAS: With an Application to Korean GDP. *SSRN Electronic Journal* (2017). https://doi.org/10.2139/ssrn.2998263
[8] Adam J Mersereau. 2015. Demand estimation from censored observations with inventory record inaccuracy. *Manufacturing & Service Operations Management* 17, 3 (2015), 335–349.
[9] Nagham Muslim Mohammad. 2014. Censored Time Series Analysis. (2014).
[10] Evgeniy Ozhegov and Daria Teterina. 2018. The Ensemble Method for Censored Demand Prediction. *SSRN Electronic Journal* 2019 (2018). https://doi.org/10.2139/ssrn.3272807 arXiv:1810.09166
[11] Ralf C. Staudemeyer and Eric Rothstein Morris. 2019. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. (sep 2019).

6 ACKNOWLEDGEMENTS

This work has been partially supported by the SONAE IM LAB@FEUP, under a research M.Sc. project funded by Inovretail.