Boosted Ensemble Learning for Anomaly Detection in 5G RAN

Tobias Sundqvist, Monowar H. Bhuyan, Johan Forsman, Erik Elmroth

Boosted Ensemble Learning for Anomaly Detection in 5G RAN

Tobias Sundqvist[1][0000−0001−9013−6603], Monowar H. Bhuyan[1], Johan Forsman[2], and Erik Elmroth[1][0000−0002−2633−6798]

[1] Dept. of Computing Science, Umeå University, SE-901 87, Umeå, Sweden
{sundqtob, monowar, elmroth}@cs.umu.se
[2] TietoEvry, SE-907 36, Umeå, Sweden
johan.forsman@tietoevry.com

Abstract. The emerging 5G networks promises more throughput, faster, and more reliable services, but as the network complexity and dynamics increases, it becomes more difficult to troubleshoot the systems. Vendors are spending a lot of time and effort on early anomaly detection in their development cycle and majority of the time is spent on manually analyzing system logs. While main research in anomaly detection uses performance metrics, anomaly detection using functional behaviour is still lacking in depth analysis. In this paper we show how a boosted ensemble of Long Short Term Memory classifiers can detect anomalies in the 5G Radio Access Network system logs. Acquiring system logs from a live 5G network is difficult due to confidentiality issues, live network disturbance, and problems to repeat scenarios. Therefore, we perform our evaluation on logs from a 5G test bed that simulate realistic traffic in a city. Our ensemble learns the functional behaviour of an application by training on logs from normal execution time. It can then detect deviations from normal behaviour and also be retrained on false positive cases found during validation. Anomaly detection in RAN shows that our ensemble called BoostLog, outperforms a single LSTM classifier and further testing on HDFS logs confirms that BoostLog also can be used in other domains. Instead of using domain experts to manually analyse system logs, BoostLog can be used by less experienced trouble shooters to automatically detect anomalies faster and more reliable.

Keywords: Anomaly detection · AdaBoost · LSTM · Radio Access Network (RAN) · 5G · System logs · Functional area.

1 Introduction

The 5G Radio Access Network (RAN) is a complex large-scale system where parts are both virtualized and distributed. Detection and diagnostics of system issues (e.g., faults, outages, degradation) are of great importance due to end-user experience and brand reputation. A key to success for companies is to find anomalies fast and early in the development cycle [8], this reduces the cost and time to market when new features are developed. Finding a way to speed up

analytics and make it easier to analyze RAN traffic would therefore be very useful since it would both make the system more reliable and also earn money for the company. When it comes to anomaly detection in RAN, the main research has focused on using performance counters, e.g., network traffic and Key Performance Indicators [11], [16], [23], [2]. A foreseen area is the functional domain where system logs can be used to determine the behaviour of the applications. System logs are today mainly used to troubleshoot systems when testing or when live traffic fails, in those cases it is very time consuming to manually analyze logs since they are huge.

In this paper, we use machine learning to develop a system that will allow trouble shooters in the Telecom area to find anomalies faster and with less system knowledge. By tracking and separating call sessions, we turn system logs into sequential time series that is suitable for anomaly detection. A model that has shown to be very useful in predicting time series is the Long Short Term Memory (LSTM) model [14], [22], [17], [4]. One advantage of the LSTM model is that it can learn and remember long term dependencies which is useful when long sequences of events are analysed. By training on sequences of events in system logs, the LSTM model can also grasp the application behaviour without having any deep knowledge of the system. Recent research use the predictions made by the LSTM model to detect anomalies [18], [5], [6]. The common idea in these articles is that a LSTM model is used to detect single deviations from normal behaviour. The problem with these approaches are that they rely on single deviations and cannot detect multiple small deviations from the normal behaviour. Another aspect is that these anomaly detection methods are performed on numerical data, when it comes to analysing functional behaviour using system logs very little has been done using machine learning. In a recent survey of anomaly detection in system logs [1], the most matured method DeepLog [9], also the LSTM model and it shows the potential of the technique. In large scale, real time systems such as RAN, there are many concurrent threads that causes events to occur in different order from time to time. This makes it very difficult for a single LSTM model to predict the next state and as in the DeepLog case, one LSTM models are trained for each unique sequence. To overcome the real time challenges we propose an adaptive boosting ensemble [20], that uses LSTM models in a novel way to detect both single anomalies but also multiple deviations from the normal functional behaviour. The main contributions of this work are as follows:

- The design of an adaptive boosting ensemble, BoostLog, which outperforms single LSTM models and automatically detects anomalies in system logs.
- A novel, single LSTM classifier that can detect both multiple small deviations and also single larger ones in system logs.
- Evaluation of BoostLog and single LSTM classifier using real-time and benchmark data including the 5G test bed and HDFS log set.

This paper is organized as follows. Section 2 presents how machine learning can be used to find anomalies in system logs and why this is important in RAN. In Section 3 we propose our method and in section 4, we outline how

our experiments were carried out. Finally our conclusions and how to proceed in future work are presented in section 5.

2 Background

2.1 Recurrent Neural Network

Instead of using predefined log rules as in analysis tools such as Logsurfer [10], Swatch [13], or SEC [19], we want a model that can use a system log to learn the normal behavior without deep knowledge about the system. The state events in the log, are typically sequential and one model that has been proven to be useful for such time series is the Recurrent Neural Network (RNN) [27], [9]. RNN has the advantage of possessing an internal memory and can remember its input. This has been found very useful in areas such as speech, text, financial data, audio, video, weather and much more. An extension of the RNN is the Long Short Term Memory (LSTM) model [14], which expands the memory of the RNN to be able to remember the inputs over a long period of time. Just like RNN, the LSTM also uses a chain of repeating modules, in which each module takes an input, produces an output, and also forward some information to the next module. Instead of having just one neural network in each module as in the normal RNN case, the LSTM uses four neural networks that interacts with each other. The main idea of the LSTM is to allow each module to select its input using an input gate, delete the information that is not important using a forget gate, and finally let the information in each module control the output using an output gate, see Fig. 1.

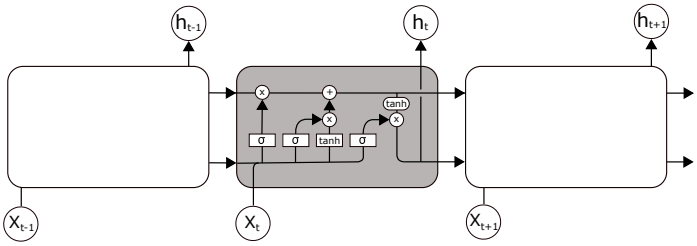


Fig. 1: LSTM architecture, X and h are input respectively output for a time step

2.2 Detecting anomalies

The first part in detecting an anomaly using a LSTM model is to make a prediction of the next state, given previous data. The prediction is then compared with the state that followed and if they differs it can be classified as an anomaly, see also Section 3.1. The normal procedure when using time series to make predictions of future events is to re-frame the events into a supervised learning problem. By using previous events in the sequence, the next time step can be predicted and compared with the next event. The number of time steps to look back are very important when the data is sequential [9]. If several sequence chains are similar, more time steps are needed in order to predict the correct outcome and if too many time steps are used it will be difficult for the model to

separate the sequence chains form each other. In Fig. 2, the importance of using many time steps, when predicting the events can be seen. In case A, only one time step is used and it is impossible to tell if 7, 8, or 9 is going to follow after event 6. In case B, two time steps are used and it is possible to tell the difference between 3-6-7 and 2-6-7 but it is still impossible to predict the 2-6-9 sequence. Only after looking at three time steps as in case C, we can tell the difference between all three sequences.

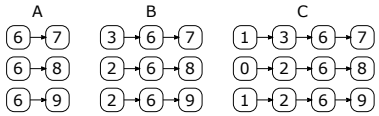


Fig. 2: Using previous time steps to predict the next, case A, B and C

2.3 RAN metrics

The 5G generation of wireless telecom networks are designed to increase the speed, throughput, and reliability. An important part of the 5G network is RAN. It consists mainly of base stations and antennas and its task is to make sure that the User Equipment (UE) can communicate with the core network, see Fig. 3.

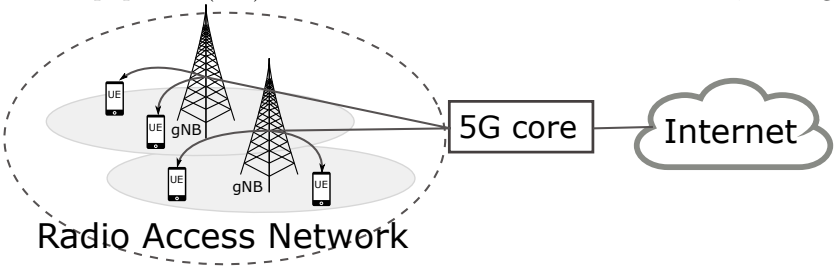


Fig. 3: 5G Radio Access Network (RAN)

In RAN there are three categories of metrics that are interesting to consider:

- Configurations, describes how the hardware and software are configured.
- Performance measurements, metrics that describes the performance of the system, often during a certain time period. Some performance measurements are used as quality indicators of the system and are often referred as Key Performance Indicators.
- System logs, contains time series of application specific events.

System logs contains most useful information when it comes to anomaly detection and root cause analytics. A faulty configuration can cause software to malfunction but logs are needed to find the configuration issues. Performance measurements can also hint that something is faulty during a certain time period but the logs are often needed to locate the fault. System logs are used to understand the behavior of the application, the vendor decides how often and what to log and it can contain start up sequences, important data, special events or alarms, etc. The best way to locate an anomaly in RAN is to use all three categories of data but we will limit our work in this article to look at how time series in system logs can be used to find the anomalies.

3 BoostLog: Proposed Method

3.1 LSTM classifier

RAN system logs contains a mix of events from thousands of session calls that occurs in parallel of each other. Such concurrency causes a random distributions of sequence events and makes it impossible to learn the behavior for any model. As in [26] a better approach is to separate work flows or in this case sessions, into separate sequences and let the model train on each sequence instead of the whole list of events. In Fig. 4 it can be seen how metrics are collected from each Base Station and stored in the metric collector. The system log are then filtered and separated into UE sessions before the ensemble of LSTMs uses them for training or prediction.

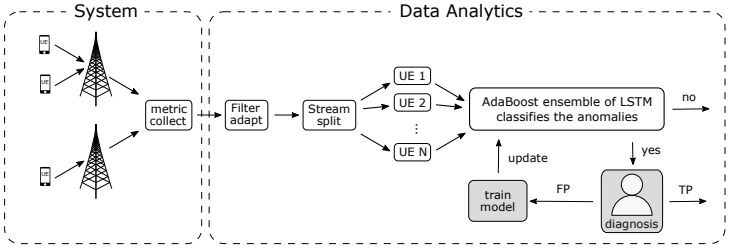


Fig. 4: BoostLog architecture

Even if it helps to separate the call sessions, all sessions will not look the same if sequence of events are analyzed. This is due to the nature of real time systems where several applications work in parallel to perform a task and depending on the load of applications, some events may occur in different order from time to time. This means that even if a LSTM model is trained on a large data set there will most likely be sequences that the model has not seen before. But even if the combined variations creates a sequence that has not been seen before, the single variations might be seen in other sequences that are used for training. This will also make the LSTM model more forgiving and sequences not seen before can be predicted normally if they only contains those small variations. Since sequences vary a lot we cannot use the most probable state from the LSTM model as the prediction, this would make the model to always choose the state that it has seen most times during training. Instead we let the model to output a vector with predictions for all states and look at the probability for the state it is trying to predict.

Fig. 5 shows the output probabilities predicted by a LSTM model after it has trained on several different sequences. The probability for a state to occur is written above the state (1.0 means 100% probability), states receiving 0 probability are not shown. To determine if a particular sequence contains any anomaly or not, the probability for each state is first compared with the highest probable state:

$$\frac{p(act)_i}{p(max)_i} = c_i \tag{1}$$

$p(act)_i$ is the probability for the state, $p(max)_i$ is the maximum probability among all states predicted, and c_i is the comparative value. Then all state comparatives is multiplied with each other resulting in a comparative value for the whole sequences:

$$c_1 \cdot c_2 \cdot ... \cdot c_n = C \tag{2}$$

Where C is the product of all comparatives for the whole sequence, we call this the sequence error. The top sequence in Fig. 5 has the highest probability and would produce a product of $1{\cdot}1{\cdot}1{\cdot}... = 1$. If a sequence shown in the bottom of the Fig. 5 occurs then it would get a product of $1{\cdot}1{\cdot}(0.1/0.7){\cdot}(0.1/0.7){\cdot}(0.1/0.7){\cdot}1{\cdot}1 = 0.0029$. The sequences can then be classified as anomaly if the sequence error is below a certain limit, this is defined as the error threshold.

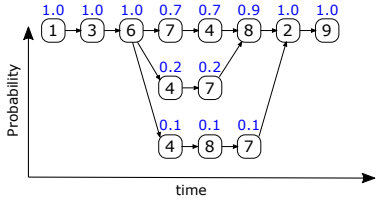


Fig. 5: Probabilities for a state to occur in different sequences detection.

The advantage of using an error threshold for the whole sequence is that it will both capture single events that are very unlikely and also sequences that has several minor deviations from the normal behaviour.

3.2 Filtering data

The RAN data used in this article contains several thousands of sequences for calls between users. Many of these sequences are similar but small differences in order of events causes a large number of unique sequences. If a LSTM model would train on the whole data it would learn to predict the most probable state sequences. Sequences that occur very seldom would be treated as an anomaly since they deviate from the normal behaviour. In order to address this issue the data is filtered so that the LSTM model only train on sequences it has not seen before.

3.3 AdaBoost ensemble of LSTM classifiers

As the single LSTM model is trained on more and more unique sequences, the probability for a certain event to occur will decrease since there are more events for the LSTM model to choose between. In these cases it would be more beneficial to use several models that trained on sequences that are similar to each other. Sequences can be compared in many ways using the length of the sequence, number of matching events, how close the events are to each other, the time difference between them, etc. Even if we divided the sequences based on these parameters, the LSTM model would perhaps not treat the sequences as similar anyway. The best would be to let the LSTM model determine if the sequences are equal or not and divide the sequences into groups and then retrain the models. This is similar of how the AdaBoost ensemble works [12], where multiple

weak learners are trained in sequence. The first learner trains and classifies the data used for training and then the next learner trains more on the data that were misclassified by the previous and so on until all learners have trained and classified the data. All the learners can then be used to classify new data by using a voting among the learners. The better a learner has been during training the more importance is given to the vote. AdaBoost in combination with LSTM models has also recently proven to be useful for predicting numerical time series [22],[24],[4]. The procedure we propose to train the AdaBoost LSTM ensemble for our purpose are the following:

1. Initialize an equal weight to all sequences used for training:
$$\omega_t(i) = \frac{1}{N} \tag{3}$$
2. The weights are used to select which sequences to train on, the higher number the more likely it is that the sequences are chosen for training.
3. A new LSTM model trains on the selected sequences.
4. The same model then predicts if there is an anomaly in any of the sequences.
5. If the accuracy is less then 0.55 then step 3-5 is repeated until the accuracy is larger than 0.55. If the accuracy does not reach 0.55 within 10 times the best model is chosen and we continue to step 6.
6. When all sequences is classified a value α is calculated based on the accuracy of the LSTM model, (t is the iteration of step 2-7):
$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t} \tag{4}$$
where ϵ_t is the sum of the weighted error for the prediction:
$$\epsilon_t = \sum_{i=1}^{m} \omega_t(i)[y_i \neq h(x_i)] \tag{5}$$
7. α is the used to increase the weights for sequences classified as anomaly and decrease weights if classified as normal:
$$\omega_{t+1}(i) = \omega_t e^{-\alpha^t h^t(x) y(x)} \tag{6}$$
$\omega(i)$ is then normalised so that the sum of all $\omega(i)$ is 1:
$$\omega_t(i) = \frac{\omega_t(i)}{\sum_{i=1}^{N} \omega_t(i)} \tag{7}$$
8. step 2-7 are repeated for all models selected to be part of the ensemble.
9. To detect anomalies, all models classifies a sequence and a voting procedure is used to determine the outcome. For each model, α is used to tell the importance of a vote, if the model had high accuracy during the training the vote will be more important.

During training of the ensemble it is also important that the training sequences does not contains any anomalies. If anomalies are present the LSTM models would also learn the anomaly behaviour and could not differ between normal and abnormal behaviour.

3.4 Anomaly feedback loop

During the continuous product development cycle, the software is constantly updated with new features and the behavior might change slightly for each software update. The update may cause the LSTM model to predict normal sequences as abnormal, this might also happen if the data contains a lot of sequences that has not been seen before (discussed in Section 3.1). The false positive (FP) sequences, can then easily be used to train the LSTM model and its weights will be adjusted during the training so that it recognizes the FP sequences. By analyzing all sequences that are classified as anomaly the FP cases can be separated and sent into the LSTM model for extra training, see Fig. 4.

4 Evaluation

4.1 RAN data

RAN system logs can contain data that are sensitive for users and vendors and are hard to obtain. Instead of using logs from commercial live systems we used a 5G test bed created by TietoEvry, which can simulate a one day scenario for a small city with 10 000 inhabitants. The advantage in using a test bed is that we can control the scenarios, repeat them, and also choose what to include in the logs without disturbing any live system. Two data sets containing 7458 respectively 7456 session calls were collected from the one day scenario in which all traffic were normal. By letting the test bed skip or add extra events in the system log, a third data set containing 26 individual anomalies were created. The anomalies were: missing event, extra event, impossible state transitions, events not seen before, multiple events missing, and multiple extra events. The session calls in the logs were separated into event sequences and the key values for each event were extracted and encoded to be suitable for the machine learning problem.

To address the imbalanced data we discussed in Section 3.2 the two normal data sets were filtered to create two lists with unique sequences. The third data set was filtered to only contain the anomaly sequences. The three data sets will later be referred to as training (173 seq), test (45 seq) and error data set (26 seq).

4.2 Benchmark data

To evaluate how BoostLog performs in other domains than RAN and also to compare our findings with recent research, a second data set was retrieved from a Hadoop Distributed File System (HDFS), in which 200 Amazon's EC2 nodes been running Hadoop-based map-reduce jobs. The block sequences in the data have been labeled by domain experts as anomaly or normal and contains 11 175 630 events in 575 061 sequence chains. This data is described more in detail in [28] and has been used for research in anomaly detection by [25] and [9]. In similar way as for the RAN data set only the unique sequences were extracted and split into train (12 406 seq), test (5317 seq) and anomaly data set (4375 seq). This data set is much larger than the RAN data set and in order to reduce

time to tune the model and make it similar to the RAN log we only used 10 % of the normal data while we kept all the anomalies.

4.3 Performance measures

The main idea of this research was to find new ways to automatically analyze the system logs in order to find anomalies faster without being expert on the domain. One important aspect is also to be able to trust the model, for trouble shooters in RAN it are very important to not classify normal data as an anomaly. If the model keeps classifying normal data as faulty then the same thing as happens in the legendary Aesop's fable *The Boy Who Cried Wolf* will occur, the users will stop relying on the model. To make the model useful and trustworthy it need to fulfill two things, it should be able to find many anomalies and seldom classify normal behavior as faulty.

4.4 The LSTM model

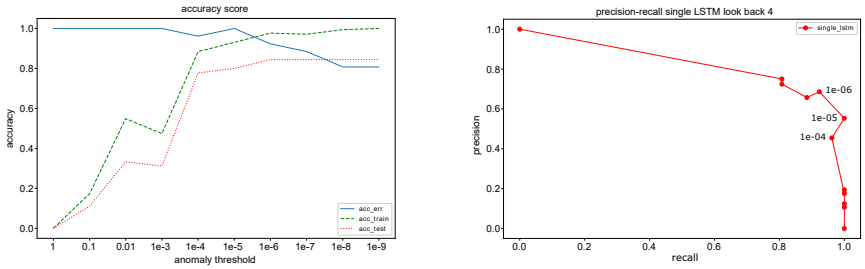
The single LSTM model used has 2 hidden layers with about 100 neurons, some models had fully connected layers and some layers used dropout. In the AdaBoost ensemble it is more beneficial to use many different classifiers and 15 LSTM models with a range from 2-3 hidden layers and 32-200 neurons were used. For all LSTM models, the keras [7] back-end with the implementation of the LSTM layer was used.

4.5 Parameter tuning

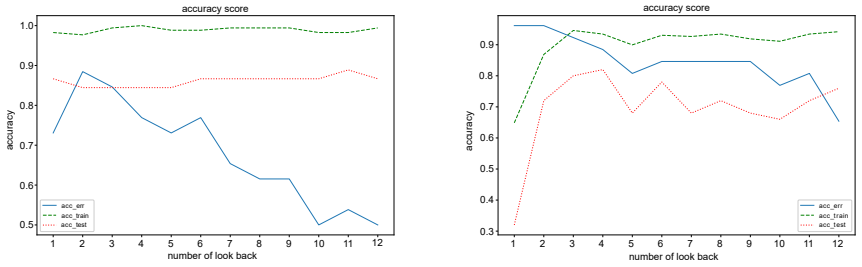
Single LSTM classifier. There are two important parameters to choose for the single LSTM classifier, firstly what error threshold that should be used and secondly how many states it should look back on in order to predict the next state.

To see the impact of the error threshold used for classifying the sequences we choose to set the look back to 4 and test error thresholds in range or 1 to $1e-9$. The model is first trained on the training data set and then the accuracy is measured on the training, test and error data sets, the result can be seen in Fig. 6a. When the error threshold is high all sequences will be classified as anomalies and if the error threshold is very low it will miss some anomalies but it will classify the normal sequences more correctly. It seems that the threshold can be lowered down to $1e-5$ before the true positive (TP) rate starts to drop and we still get a quite high accuracy for the normal sequences. This can also be seen in the precision-recall curve in Fig. 6a.

Next parameter to tune is the number of look backs, ranging the look backs from 1 to 12, we tested both the error threshold at $1e-5$ where the true positive and true negative (TN) rate was high and also at $1e-9$ were TN rate was highest. For $1e-9$ the accuracy for the normal sequences stayed stable at the same level as in Fig. 6a but the accuracy kept decreasing with the increasing amount of look backs, see Fig. 6b. The drop in accuracy is explained by the increasing certainty of the model when the number of states to look back on are increased, a higher threshold is needed to compensate for the more precise model but then the accuracy would also drop for the normal sequences. The



(a) Accuracy for error, train and test data. Precision-recall, normal sequences



(b) Left: Accuracy for error threshold $1e-9$, Right: Accuracy for error threshold $1e-5$

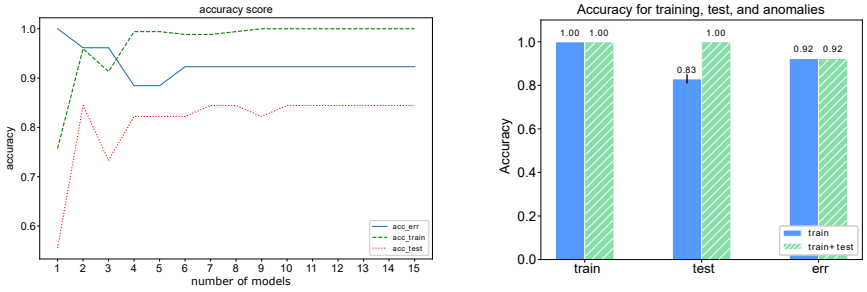
Fig. 6: Tuning of Single LSTM classifier

same tendency is seen for the $1e-5$ threshold and even if the accuracy for the training sequences increases the accuracy is not increased any more for the test sequences when the number of look backs are increased above 4, see Fig. 6b.

BoostLog. To start with, we used the same error threshold and look back parameter in BoostLog as for the single LSTM classifier. To see how the AdaBoost classifier improves with the amount of learners, the accuracy was calculated for the three data sets, see Fig. 7a. As we can see the accuracy are getting stable at 10 learners. When comparing with the single LSTM model we can also see that the accuracy is better for both test and error sequences. This is very promising since the accuracy improved for those sequences not seen yet. Normally it is better to use weak learners that achieve an accuracy just above 0.5 in an AdaBoost ensemble. As expected the ensemble improved even more when increasing the error threshold and decreasing number of look back to get learners that were not equally good as the best single LSTM model. Final result can be see in Table. 1.

4.6 Training using feedback loop

Both the single LSTM model and BoostLog classified some of the normal sequences as an anomaly. These anomalies are considered false positive (FP) and as we discussed in Section 4.3 and 3.4, we want to minimize the FP rate to make the model more trustworthy and useful. The models classifies these sequences as anomalies since they have not seen these variations before or at least very seldom. When retraining the single LSTM classifier with the normal sequences not seen before, the overall accuracy did not change, in mean $13\pm2\%$ of the normal sequences were still classified as anomaly. As we discussed earlier in



(a) BoostLog accuracy, (b):Anomalies detected before and after feedback

Fig. 7: BoostLog

Section 3.3 the single LSTM model can only handle a certain amount of different sequences.

When the same procedure was performed for the BoostLog the accuracy stayed the same for the anomaly sequences and increased from 84% to 100% for the sequences it had not seen before, see Fig. 7a.

Among the false negative cases, there were only two sequences that both the single LSTM and BoostLog classified as normal and both these cases missed one state in the sequence. These anomalies are difficult for the LSTM classifier to detect since the single missed state will not cause a very large drop in sequence error. If the LSTM predicts the probability for time step t+1 it will also know what state that is likely to occur in time step t+2. The probability for the state to occur in t+2 will be slightly higher than other states and if the state for time step t+1 is missing, the second highest probability is likely to be the state for t+2.

4.7 Evaluation

In this work we have limit our models to only look at one feature from the system logs. The main focus have been to investigate how the anomaly detection can be improved in system logs using an ensemble of LSTM models. To do this we have compared the single LSTM model with the new BoostLog ensemble. Precision, Recall, and F-score was calculated for each model before and after training on RAN testbed, see Table. 1, the models are called feedback model when training has been performed on normal sequences not seen before. The single LSTM model improved in detecting anomalies (the recall value), but instead it also classified more normal sequences as anomalies (the precision value). In total there was no improvement in the F-score. The BoostLog ensemble outperforms the single LSTM model and adapted very well to the new data while it kept the recall rate high. After training on test data it was very reliable and did not point out one single normal sequence as anomaly and still found 24 of the 26 anomalies.

The same models were then tuned for the HDFS data set and tested in the same way as for the RAN data, see Table. 2.

Table 1: Precision, recall, and F-score for RAN data set

Model	**Precision**	**Recall**	**F-score**
Single LSTM	0.64	0.88	0.74
AdaBoost LSTM	0.77	0.92	0.84
Feedback Single LSTM	0.59	1.0	0.74
Feedback AdaBoost LSTM	1.00	0.92	0.96

Table 2: Precision, recall, and F-score for HDFS data set

Model	**Precision**	**Recall**	**F-score**
Single LSTM	0.96	0.94	0.95
AdaBoost LSTM	0.94	0.98	0.96
Feedback Single LSTM	0.96	0.93	0.94
Feedback AdaBoost LSTM	0.97	0.98	0.97

It seems that it is much easier for the single LSTM model to learn the sequence behaviour of the HDFS block sequences compared to the RAN call chains, it can also more easily find the anomalies. A quick check with fast Dynamic Time Warping (fastDTW) [21], confirms that the unique sequences in RAN data set is much more similar to each other than for the HDFS data. There is also a larger difference between the normal and anomaly sequences for HDFS data compared with the RAN data. Similar sequences will make it more difficult to predict the outcome for the single LSTM model and if the anomaly sequences also are more similar to normal sequences it will get even more difficult to detect the anomalies. This might explain why the single LSTM model can detect the anomalies easier in the HDFS data. We can also see that, as in the RAN data set, there is no improvement for the single LSTM model when training extra on normal sequences not seen before. Since the single LSTM model already performs very well on the HDFS data set, the results cannot be improved much using an ensemble, we can only see a slight improvement using the BoostLog ensemble.

Our results cannot be directly compared with previous research that has used the HDFS data set [9],[25], since they use multi features and also look at the whole data set. But the results are promising, in [9], the best model, also got a F-score of 0.96 when it analysed the HDFS data set.

5 Conclusion and Future Work

We have created a novel method that turns the LSTM model into a classifier for anomaly detection in system logs, in this case, logs from a large scale system such as 5G RAN. Furthermore we have also shown how the AdaBoost algorithm can be adopted to use our LSTM classifier to create an even better anomaly detection ensemble, the BoostLog.

Evaluation of performance in two different datasets shows that it is more beneficial to use BoostLog in domains such as RAN, where the sequences are similar to each other and the anomalies are difficult to distinguish from the normal behaviour.

BoostLog also adapts well when a feedback loop was used to train extra on false positive sequences and maximized the precision for the RAN data set. We expect that deep ensemble based networks such as BoostLog, can be used to find

more anomalies early in RAN development cycle and will require less domain expertise in order to find anomaly behaviour in the system logs.

In this paper we investigated how one feature could be used to predict the anomalies and it will be interesting to see how our future work can be improved by selecting more features and also take into account the time that events occur. In our previous work we have examined how anomalies and security issues can be detected using performance metrics [15],[3]. It will be interesting to see if the anomaly detection can be further improved using a combination of ensembles that analyze both functional and performance behaviour. We must also test our method on system logs from more different areas to see if the ensemble also is suitable to use in those domains.

Acknowledgement

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation.

References

1. Ajith, D.: A survey on anomaly detection methods for system log data. International Journal of Science and Research (IJSR) **8**, 23 (07 2019)
2. Al Mamun, S.M.A., Valimaki, J.: Anomaly detection and classification in cellular networks using automatic labeling technique for applying supervised learning. Procedia Computer Science **140** (11 2018). https://doi.org/10.1016/j.procs.2018.10.328
3. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools. Springer Publishing Company, Incorporated, 1st edn. (2017)
4. Bian, G., Liu, J., Lin, W.: Internet traffic forecasting using boosting lstm method. DEStech Transactions on Computer Science and Engineering (01 2018). https://doi.org/10.12783/dtcse/csae2017/17517
5. Chalapathy, R., Chawla, S.: Deep learning for anomaly detection: A survey. CoRR **abs/1901.03407** (2019), http://arxiv.org/abs/1901.03407
6. Chniti, G., Bakir, H., Zaher, H.: E-commerce time series forecasting using lstm neural network and support vector regression. In: Proceedings of the International Conference on Big Data and Internet of Thing. p. 80–84. BDIOT2017, Association for Computing Machinery, New York, NY, USA (2017)
7. Chollet, F., et al.: Keras. https://keras.io (2015)
8. Damm, L.O.: Early and Cost-Effective Software Fault Detection : Measurement and Implementation in an Industrial Setting. Ph.D. thesis (2007)
9. Du, M., Li, F., Zheng, G., Srikumar, V.: Deeplog: Anomaly detection and diagnosis from system logs through deep learning. pp. 1285–1298 (10 2017). https://doi.org/10.1145/3133956.3134015
10. E. Prewett, J.: Analyzing cluster log files using logsurfer (06 2003)
11. Fernández Maimó, L., Perales Gómez, L., García Clemente, F.J., Gil Pérez, M., Martínez Pérez, G.: A self-adaptive deep learning-based system for anomaly detection in 5G networks. IEEE Access **6**, 7700–7712 (2018). https://doi.org/10.1109/ACCESS.2018.2803446

12. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55**(1), 119 – 139 (1997)
13. Hansen, S.E., Atkins, E.T.: Automated system monitoring and notification with swatch. In: Proceedings of the 7th USENIX Conference on System Administration. pp. 145–152. LISA '93, USENIX Association, Berkeley, CA, USA (1993)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
15. Ibidunmoye, O., Rezaie, A., Elmroth, E.: Adaptive anomaly detection in performance metric streams. IEEE Transactions on Network and Service Management **15**(1), 217–231 (March 2018). https://doi.org/10.1109/TNSM.2017.2750906
16. Iyer, A.P., Li, L.E., Stoica, I.: Automating diagnosis of cellular radio access network problems. In: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking. pp. 79–87. MobiCom '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3117811.3117813
17. Karevan, Z., Suykens, J.A.: Transductive lstm for time-series prediction: An application to weather forecasting. Neural Networks **125**, 1 – 9 (2020)
18. Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G.: Lstm-based encoder-decoder for multi-sensor anomaly detection. CoRR **abs/1607.00148** (2016), http://arxiv.org/abs/1607.00148
19. P. Rouillard, J.: Real-time log file analysis using the simple event correlator (sec). pp. 133–150 (01 2004)
20. Polikar, R.: Polikar, r.: Ensemble based systems in decision making. ieee circuit syst. mag. 6, 21-45. Circuits and Systems Magazine, IEEE **6**, 21 – 45 (10 2006). https://doi.org/10.1109/MCAS.2006.1688199
21. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**(5), 561–580 (Oct 2007)
22. Sun, S., Wei, Y., Wang, S.: Adaboost-lstm ensemble learning for financial time series forecasting. In: Shi, Y., Fu, H., Tian, Y., Krzhizhanovskaya, V.V., Lees, M.H., Dongarra, J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2018. pp. 590–597. Springer International Publishing, Cham (2018)
23. Szilagyi, P., Novaczki, S.: An automatic detection and diagnosis framework for mobile communication systems. IEEE Transactions on Network and Service Management **9**(2), 184–197 (June 2012). https://doi.org/10.1109/TNSM.2012.031912.110155
24. Xiao, C., Chen, N., Hu, C., Wang, K., Gong, J., Chen, Z.: Short and mid-term sea surface temperature prediction using time-series satellite data and lstm-adaboost combination approach. Remote Sensing of Environment **233**, 111358 (2019)
25. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. p. 117–132. SOSP '09, Association for Computing Machinery, New York, NY, USA (2009)
26. Yu, X., Joshi, P., Xu, J., Jin, G., Zhang, H., Jiang, G.: Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. In: ASPLOS (2016)
27. Zhang, K., Xu, J., Min, M.R., Jiang, G., Pelechrinis, K., Zhang, H.: Automated it system failure prediction: A deep learning approach. In: 2016 IEEE International Conference on Big Data (Big Data). pp. 1291–1300 (Dec 2016). https://doi.org/10.1109/BigData.2016.7840733
28. Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R.: Tools and benchmarks for automated log parsing. CoRR **abs/1811.03509** (2018)