Research Proposal: Anomaly Detection in Flexible Assembly Systems

Tero Keski-Valkama

July 14, 2015

Keywords. Anomaly Detection, Assembly, Long Short-Term Memory, Log-structured Input, Neural Networks

1 Problem Statement

Modern industrial plants are gaining advanced data acquisition capabilities rapidly because of Internet of Things technologies and open, standard APIs. Malfunctions are a significant cause for expensive downtime in automatic plants. Traditionally malfunctions are detected through specific and pre-designed fault codes and diagnostic metrics. However, especially in the context of preventive maintenance the fault codes are lacking as they typically signal only at the point where downtime actually happens. Preventive maintenance is done by periodical scheduling and at the discretion of human operators when something seems to start breaking apart. Often the human operators periodically check the process metrics and event logs to see any discrepancies and anomalies. There is a lot of potential in automatic anomaly detection, as the anomalies can be observed instantly as they happen and not only after the human operator checks the logs and metrics the next time. Automatic anomaly detection can also potentially detect anomalies that would otherwise not be immediately evident for human operators.

Automatic anomaly detection in its general form monitors the states and signals of the industrial processes and creates new warning signals based on perceived potential anomalies. The anomaly detection generally utilizes either a predictive model or a set of heuristics to define the normal mode of operation for the plant processes. To derive such a model, either machine learning approaches or existing human operator knowledge, or some combination of the two is used.

Modern flexible manufacturing and assembly sites are very heterogeneous and different end products require different workflows and sequences in the system. Operation that is normal for one site might be considered anomalous for another. However, there are some general characteristics that are common for all sites with approximately similar production processes, especially for distinct components of the system.

A commonly used approach for anomaly detection is forming a heuristic rule set for defining the normal operation modes of the system. These have an advantage of being relatively easy to set up and getting useful results immediately, especially in large systems with relatively simple constraints that are clearly defined. Forming a rule-based anomaly detection manually can become very cumbersome if the system is very complex. There is a limit on the complexity of such a rule set because the rule set needs to be maintained and updated. For this reason, manually defined heuristic rule sets for anomaly detection might be lax in their formulation for complex systems which limits their usefulness in such domains.

Sequences of events can be used to create an abstraction of causal chains in industrial processes under observation. Adding delays to the event sequence model can provide important additional knowledge about the patterns in the process. Delays in industrial processes typically follow approximately multimodal distributions with several distinct peaks. To use a figurative example, an elevator might take several different paths between the doors closing and opening in different floors, so the delays would form different peaks following the delays it takes to travel different distances between all the floors.

Some events are unrelated to an otherwise deterministic sequence being executed and might happen in between two events otherwise separated by a constant delay cutting the delay between the two related events into two arbitrary parts. Even if the system could extract and model the deterministic causally linked sequences from the process, the observed sequence might not include relevant information to estimate the correct peak of the multimodal delay distribution. This happens for example when the event for the figurative elevator button being pressed is not available for observation for the system.

2 Background

I got first hand experience in the problem field and potential solutions in #IndustryHack #HackTheFactory competition in Fastems in May 22th – May 24th, 2015[1]. My team won the competition with a consept utilizing a trivial and naive Markov chain model of the process with crude anomaly detection capability. There is a clear need of such solutions, and considerable improvement of the concept is possible through researching novel application of modern machine learning in the context of industrial automation system anomaly detection.

Events are observed through different log structured data sources and event source APIs. In the context of this research an anomaly means a sequence of observed events which deviates from the expected, normal and flawless system operation and possibly predicts a future fault.

Internet of Things and deep learning with Long Short Term Memory are relatively new fields and as far as I am aware no published solutions exist utilizing recurrent neural networks for industrial plant anomaly detection. Of course additional literature research is needed.

3 Research Proposal

I propose a novel method of anomaly detection using online unsupervised Long Short Term Memory (LSTM)[4] recurrent neural networks which can be used to predictive modelling the underlying automation process. Unsupervised neural networks are used in anomaly detection generally by making them model the events detected in the process and predicting the following states. If the model fails to predict the following events, an anomaly is flagged. As a practical example using an elevator as a figurative example, sensors might indicate elevator doors opening twice in a row without being properly closed in between.

Online model is required for the system to run in a live environment and generate useful alerts. Generalizing and specializing the learned models between similar but different assembly processes might warrant hierarchical modelling.

For training the models, it is possible to use logs that span multiple sites and significant time spans. Validation of the systems and solution alternatives is done by quantifying the faults that are correctly predicted by the anomaly detection system.

3.1 Anomaly Detection

Several novel specializations are suggested to the general LSTM model to make this kind of approach work in the practical business context. For example, utilizing the learned data from several sites so that actual anomalies do not become something the unsupervised neural network expects and therefore does not recognize as anomalies anymore is a fundamental problem with anomaly detection systems based on quantifying the "surprise" when comparing the prediction against the actual subsequent event. A hybrid model is suggested where the network classifies the situations into normal and specific classes of previously encountered anomalies in addition to predicting the next event. This requires the network to associate newly encountered situations into previously encountered surprising learning events in an unsupervised fashion. This might be implemented for example using a separate learning system classifying the internal states of the LSTM neural network into classes of previously encountered anomalies and normal operation. Newly encountered and detected anomalies are classified and tagged as relevant or irrelevant at a later point when the anomaly notifications are handled by a human operator opening up possibilities of associating new anomalies to past anomaly classes and situations, and to attached metadata such as comments by operators.

Anomalies can be detected at least through three distinct mechanisms. First, an event generates an anomaly event if the network fails to predict it. This means that if the predicted probability of an event is significantly smaller than the probability of the most probable subsequent event, an anomaly event is generated. Anomalies that are detected are classified into classes of encountered anomalies, which are then semantically tagged as true anomalies or anomalies which do not warrant an alert. It is also possible to attach metadata such as operator comments, or videos of anomalies in question. Second, an anomaly alert is generated if a situation which was previously classified as a true anomaly is encountered again. Third, if the system can predict that a proper fault code will happen in the subsequent time steps for example using a monte carlo algorithm, the probability of the fault happening in the near future can be estimated by the system.

3.2 Delays

Delays are an important part of detecting anomalies and a significant variation in the delay between events might certainly be relevant for anomaly detection. Again using the elevator as a figurative example, a change in the delay between the elevator moving between floors might indicate a future fault and should be indicated as an anomaly. This requires comparisons of novel combinations of statistical techniques with recurrent neural network base system or different kinds of modellings of delays in such a neural network.

Modelling the delay might require measuring the delay to the previous event of a certain type to prevent unrelated events from cutting the modelled delays into arbitrary parts, and even then we might get a multimodal distribution. Event types in this context correspond to the input neurons of the neural network. For some cases it might be possible to extract information from the event sequence so that the expected mode (peak) of the distribution can be selected from multiple peaks. For example, the elevator model might estimate the delay between doors closing and opening if it observes a certain floor button being pressed in between. A literature study is needed to search for good methods to use in modelling multimodal distributions in anomaly detection context. Possibly kernel density estimation with decimation could be used for this. This is also closely related to coding general numerical values in log events into neural network input values.

4 Proof of Concept

The first proof of concept using a simulated factory model and CURRENNT[5] recurrent LSTM neural network framework has been implemented and it shows that a toy example[3] is learned properly, measured by the error in the validation set approaching zero. The toy example was designed to be somewhat non-deterministic and to include specific features mixing separate plant sequences together by using shared intermediate states to make it impossible for a trivial Markov chain model to learn the process properly.

A fork of CURRENNT was made to make it work on a recent CUDA Toolkit[2].

5 Practical Next Steps

The simulated factory model will be augmented in the future to contain salient process dynamics and features from the real industrial plant logs to enable subsequent derivative research and faster iteration cycles. The real factory data contains critical business secrets such as fault frequencies and therefore cannot be published as such.

Additional validation requires visualization of the learned features and generative modelling not currently supported by CURRENNT. CURRENNT also does not support online, or stream execution model. It will be decided later whether extending CURRENNT or creating a completely new library for the neural network model is the best approach.

It is suggested that the core of the research will be done in Tampere from Q4/2015 forwards. The research will be divided into several publications, roughly following:

- Simulating realistic assembly plant operation and malfunctions
- Validating the LSTM model learned from logs using visualization and generation

- Comparison of LSTM model with other models, for example Markov Chain
- Incorporating delay modelling into the learned process model
- Classification and retrieval of previously encountered anomalies

Journals and Conferences to consider for research publication:

- Neural Information Processing Systems Conference: `https://nips.cc/`
- Annual Reviews in Control: `http://www.journals.elsevier.com/annual-reviews-in-control/`
- Computational Intelligence: `http://onlinelibrary.wiley.com/journal/10.1111/%28ISSN%291467-8640`
- Engineering Applications of Artificial Intelligence: `http://www.journals.elsevier.com/engineering-applications-of-artificial-intelligence/`
- IEEE Congress on Evolutionary Computation: `http://sites.ieee.org/cec2015/`
- IEEE International Conference on Information and Automation: `http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1002411`
- International Conference on Control, Automation, Robotics and Vision: `http://www.icarcv.org`
- International Conference on Emerging Technologies in Factory Automation: `http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000260`
- International Conference on Information, Communications and Signal Processing: `http://www.icics.org`
- International Journal of Engineering, Science and Technology: `http://www.ijest.info/`
- International Journal of Neural Systems: `http://www.worldscientific.com/worldscinet/ijns`
- Neural Computation: `http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6720226`
- Neurocomputing: `http://www.journals.elsevier.com/neurocomputing/`
- World Academy Of Science, Engineering And Technology: `http://www.waset.org`

Upcoming conferences:

- 2016 IEEE Multi-Conference on Systems and Control (MSC): `http://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=35601`
- 2016 IEEE 8th International Conference on Intelligent Systems (IS): `http://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=36804`

References

[1] Cybercom team won the #IndustryHack #HackTheFactory Fastems - Summary. https://www.linkedin.com/pulse/cybercom-team-won-industryhack-hackthefactory-fastems-keski-valkama.

[2] A fork of CURRENNT to make it run against the latest CUDA toolkit. https://github.com/keskival/currennt.

[3] Toy example of an assembly simulation: EventSimulator. https://github.com/keskival/EventSimulator.

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

[5] Felix Weninger. Introducing currennt: The munich open-source cuda recurrent neural network toolkit. *Journal of Machine Learning Research*, 16:547–551, 2015.