Research Proposal: Anomaly Detection in Flexible Assembly Systems using LSTM Neural Networks with Process Traces

Tero Keski-Valkama

February 29, 2016

Keywords. Anomaly detection, Assembly systems, Artificial intelligence, Fault detection, Learning systems, Long short-term memory, Process traces, Machine intelligence, Machine learning, Pattern analysis, Predictive maintenance, Recurrent 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 flexible assembly systems. Traditionally malfunctions are detected through specific and predesigned fault codes and diagnostic metrics. However, especially in the context of predictive 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 assembly 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 few and static 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 practical limit on the complexity of such a rule set because the rule set needs to be maintained and updated. Rule-based anomaly detection can handle large data structures and extract rich domain specific information from the log messages such as calculated length of sessions.

There are also existing systems which are approximately equivalent to a naive Markov chain model, like [7]. These have the same issues as the trivial Markov chain model, namely that with multiple interlinked and parallel processes, the state transition space becomes exponential and a prohibitive amount of time would be needed to train such a system.

Sequences of events can be used to create an abstraction of causal chains in industrial processes under observation. Dmbedding intervals to the event sequence model can provide important additional knowledge about the patterns in the process.

Intervals 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.

Coding the intervals could be done by simply embedding the symbols into a feature space with preceding intervals to each of the symbol classes. Coding this as a convolutive list of delays will implement a direct coding of information about the dynamic state of the system. It is hypothesized that this will make it easier for machine learning methods to extract process models.

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 concept 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. The Markov chain model was roughly equivalent to a previously studied DFA model[7] which was ultimately deemed inadequate for situations where log messages of parallel processes are interleaved together into the same process trace. DFA and Markov chain models assume there is only one process instance reflected in the logs, and cannot learn to separate different processes. Additionally it might be impossible to manually multiplex log messages to separate state machines as there might be log messages that are shared for several separate processes.

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 flexible assembly system anomaly detection.

3 Research Proposal

I propose a novel method of anomaly detection using online unsupervised Long Short Term Memory (LSTM)[6] recurrent neural networks which can be used to predictive modelling[4] 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, that is using continual prediction. If the model fails to predict the following events, an anomaly is flagged. As a practical example of an anomaly using an elevator as a figurative example, sensors might indicate elevator doors opening twice in a row without being properly closed in between.

LSTM neural networks work well for tasks in which a limited amount of input data or some abstractions of it affect the output for a long time. Specifically it is hypothesized that LSTM neural networks are good in separating and learning to model separate parallel processes which generate interleaved log messages in an unsupervised fashion with limited or no information about the domain.

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 and transfer learning.

Existing anomaly detection and process deviation detection simulators and validation logs which might apply to this field of study have not been published, so it is necessary to start this research by creating a valid simulator for a Flexible Assembly 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, it is possible for an anomaly detection system to learn to expect an anomaly, and therefore not consider it anomalous anymore. This 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 a partially supervised 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.

There has been some research into coding delays or intervals into LSTM neural networks. Typically the delays are coded using a constant interval time steps[5]. However, for log-structured data using constant interval time steps is not a very good approach, but could in principle be implemented by adding a constant interval tick log message as an input so that the LSTM neural network can count the ticks between events. In any case, because we don't know the relevant time spans beforehand, the tick trains would potentially become prohibitively long, or on the other hand, might lose smaller grain detail. For log structured data the time spans should be coded into the input similarly to other real valued data.

Because the delays follow multimodal distributions and the neural network predicts the delay through coded output, the neural network output neurons that correspond to the coding of the delay should be able to code multimodal distributions. For example, a single real valued output neuron is not adequate, because it would only code one peak of a potentially multimodal distribution. Output neurons corresponding to histogram buckets of the probability density could be used. Histogram buckets of the probability density of course would have preset ranges, and this might potentially lose relevant information in the coding. Output could be coded as normalized with the output standard deviation, as long as the output standard deviation stabilizes relatively fast during learning.

It is also possible to code the delays as input values to lists of convolutive neurons.

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 net-

work. 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 predicted from multiple peaks by the neural network. 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.

It is hypothesized that coding the preceding intervals into each of the symbol classes provides a useful embedding for this information.

4 Proof of Concept

The first proof of concept using a simulated factory model and CURRENNT[8] 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 assembly 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 flexible assembly system 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. The next iterations of the LSTM neural network system are implemented in CuTorch.

It is suggested that the core of the research will be done in Tampere from Q1/2016 forwards. The research will be divided into several publications, roughly following:

- Simulating realistic flexible assembly system operation and malfunctions
- Validating the LSTM model learned from logs using visualization and generation
- Embedding delay modelling into the data model
- Investigation of transfer learning techniques for this application field
- Classification and retrieval of previously encountered anomalies possibly using Case Based Reasoning

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] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[5] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, 3:115–143, 2003.

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

[7] Falk Langer, Karsten Bertulies, and F Hoffmann. Self learning anomaly detection for embedded safety critical systems. *Schriftenreihe des Instituts für Angewandte Informatik, Automatisierungstechnik am Karlsruher Institut für Technologie. KIT Scientific Publishing*, pages 31–45, 2011.

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