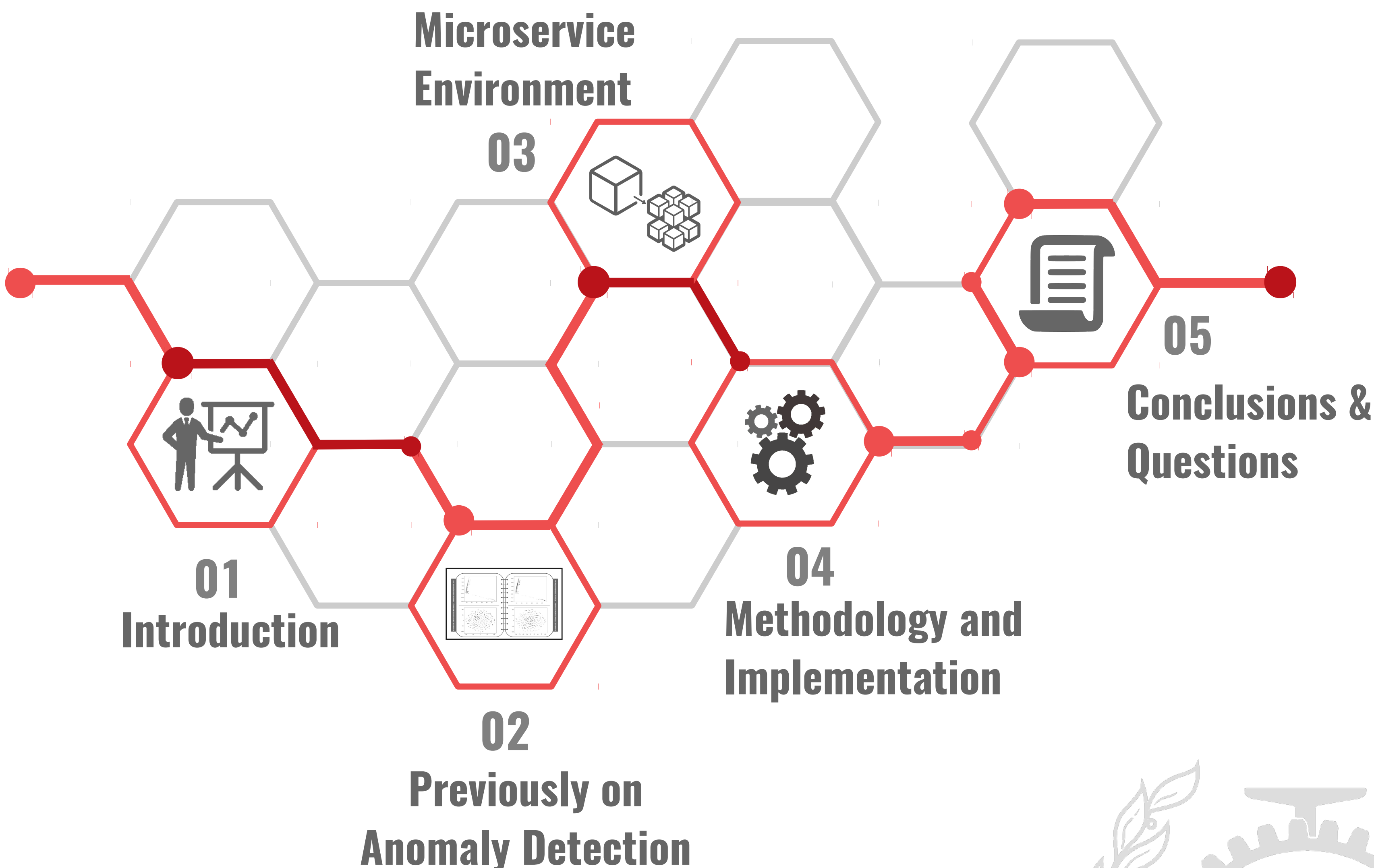
Anomaly detection in microservice systems using tracing data and Machine Learning

Iman Kohyarnejadfard
Prof. Daniel Aloise
Prof. Michel Dagenais

POLYTECHNIQUE MONTREAL

January 2021

Agenda



01
Introduction
02
Previously on Anomaly Detection
03
Microservice Environment
04
Methodology and Implementation
05
Conclusions & Questions

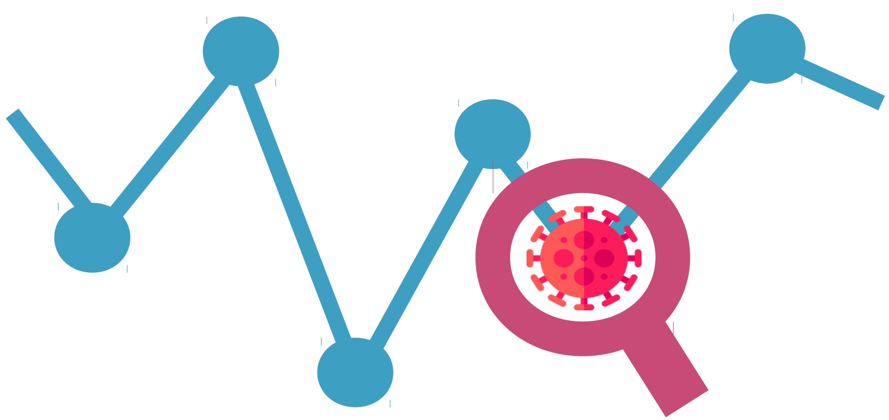
Anomaly Detection

Anomaly is something different which deviates from the common rule.

Anomalies are patterns in data that do not conform to a well defined notion of normal behavior.

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior.

Many anomaly detection techniques have been developed for various application domains.



Performance Anomaly

The most significant obstacles to the system to perform confidently and predictably

Sources

varying application load, application bugs, updates, and hardware failure

Workload

the application imposes continuous and higher than expected average workload intensity to the system.

Faults in system resources and Components

affect application performance at a high cost

Software bugs, operator errors, hardware faults, and security violations

system failures

Previously on Anomaly Detection

01 System performance anomaly detection

Supervised: A multi-class support vector machine approach was applied to detect the anomalous system call sequences.

Unsupervised: Using clustering techniques such as k-means or DBSCAN removes the need for providing a huge labeled dataset.

Semi-supervised: This method benefits from both supervised and unsupervised learning techniques to distinguish between normal and anomalous behavior.

Previously on Anomaly Detection

The concept of DevOps and agile approaches like microservice architectures and Continuous Integration becomes extremely popular since the need for flexible and scalable solutions increased.

Microservice-based applications

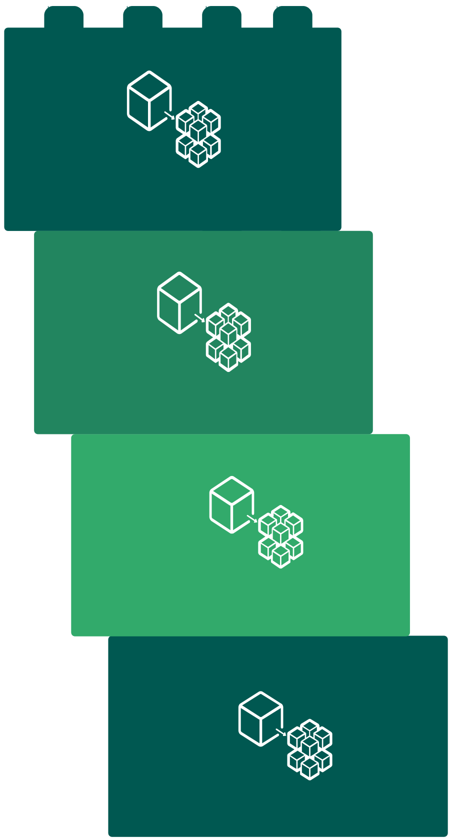
01 Microservices are small services that are interconnected with many other microservices to present complex services like web applications.

02 Microservices provide greater scalability and make distributing the application over multiple physical or virtual systems possible.

03 Microservices architecture tackles the problem of productivity and speed by decomposing applications into smaller services that are faster to develop and easier to manage; if one microservice fails, the others will continue to work.

04 Each microservice can be written using different technologies, and they enable continuous delivery.

Microservice-based applications

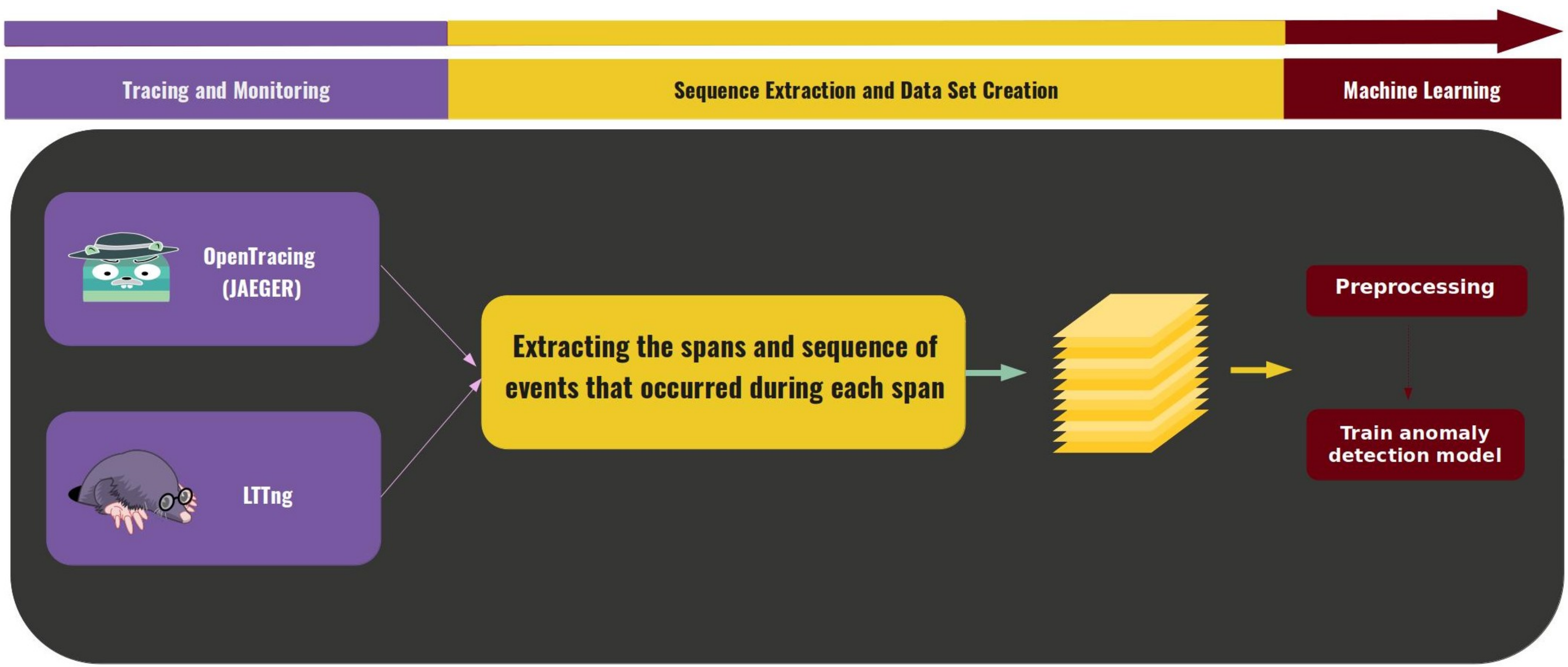


Despite all these benefits, by increasing the degree of automation and distribution, application performance monitoring becomes more challenging because microservices are possibly short-lived and may be replaced within seconds.

Hence new requirements in the way of anomaly detection have emerged as these changes could also be the cause of anomalies.

Methodology

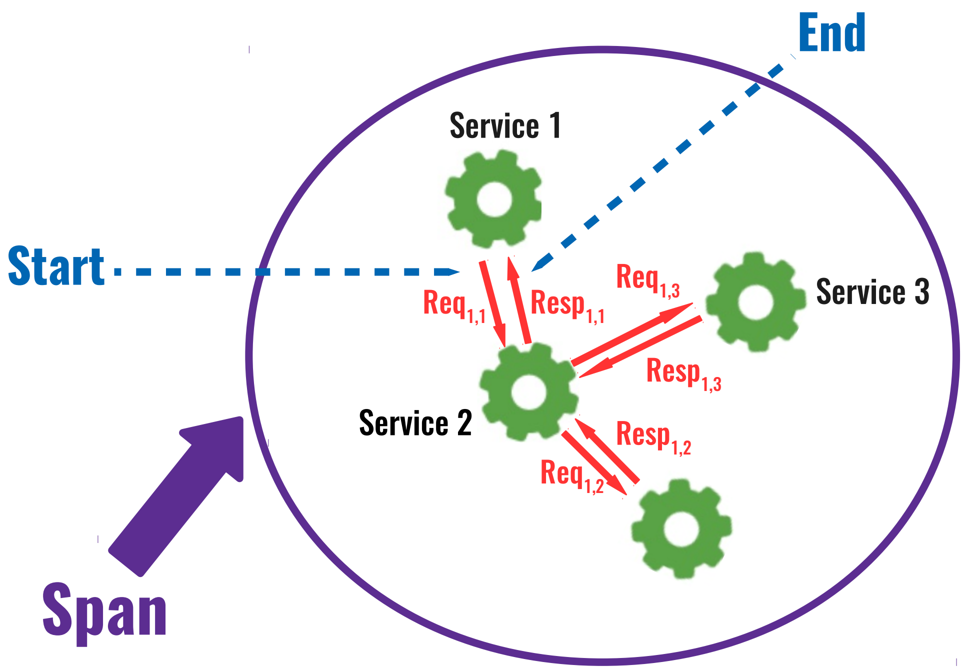
- The methodology is based on collecting sequences of events during spans and sending them to the Machine learning module.
- The model learns the possible sequence of events and predicts the next event.
- In the detection phase, we use this sequential information to make a prediction and compare the predicted output against the observed value.



Distributed tracing

A microservice-based application consists of tens, hundreds, or thousands of services running across many hosts, and it is no longer possible to rely on an individual trace.

Distributed tracing provides a view of a request's life as it travels across multiple hosts and services communicating over various protocols.

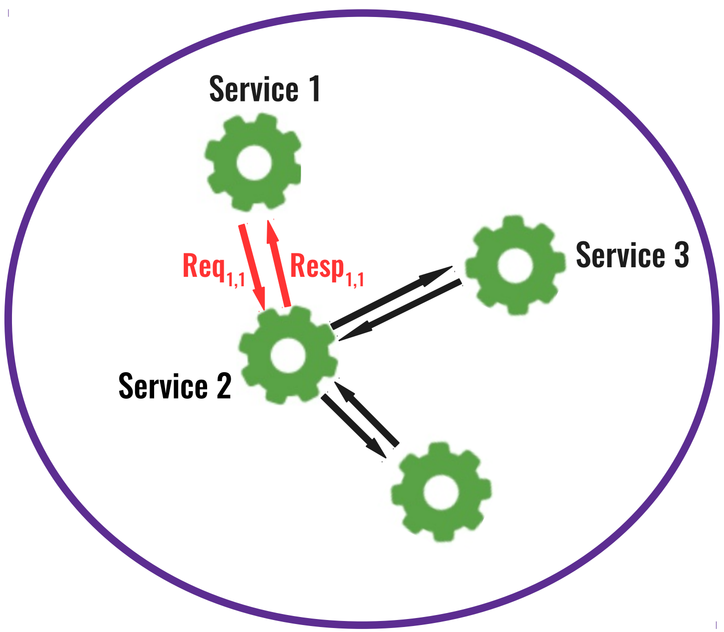


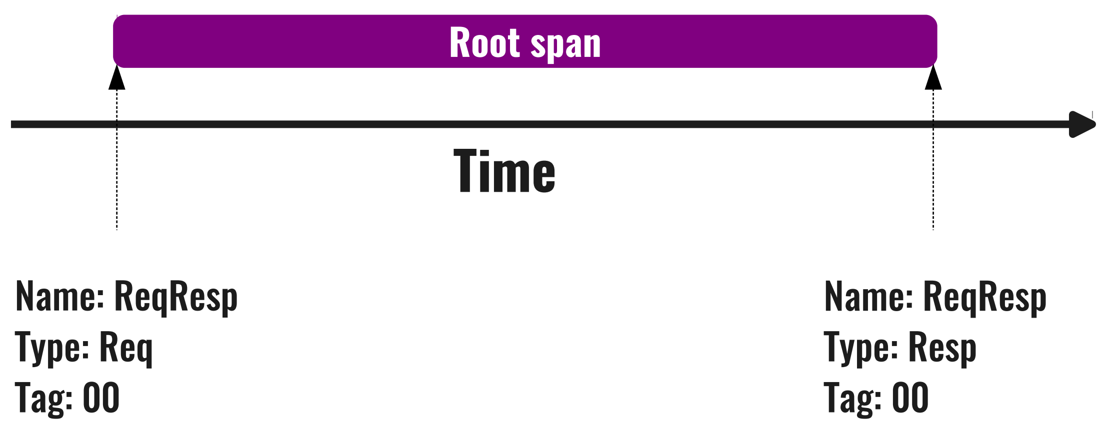
OpenTracing vs. LTTng: Different in the way we collect spans.

The "span" is the primary building block of a distributed trace, representing an individual unit of work done in a distributed system.

Extracting spans and sub spans

- In the traces we collected from the Ciena simulator, **ReqResp** events make spans.
- Spans are initiated with a **ReqResp** event of type **Req** and ended by a **Resp.**
- Requests and responses that happen during a span share a unique tag for example Tag = **00**.

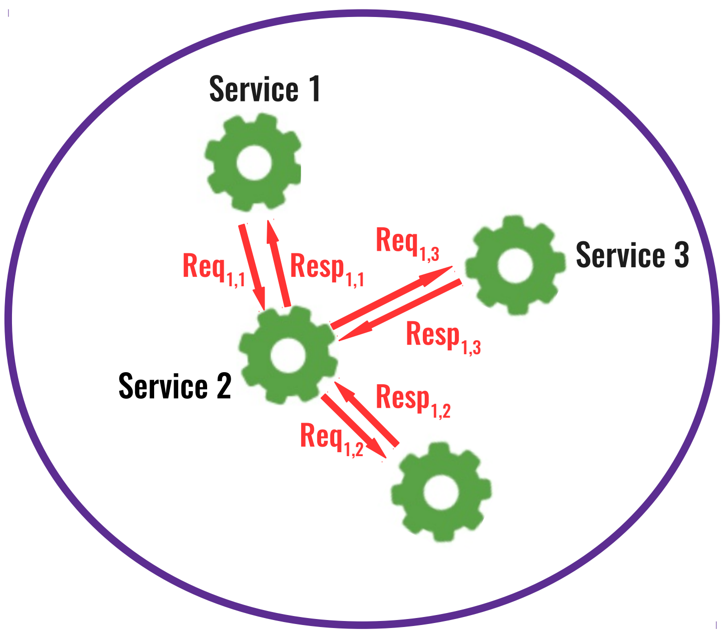


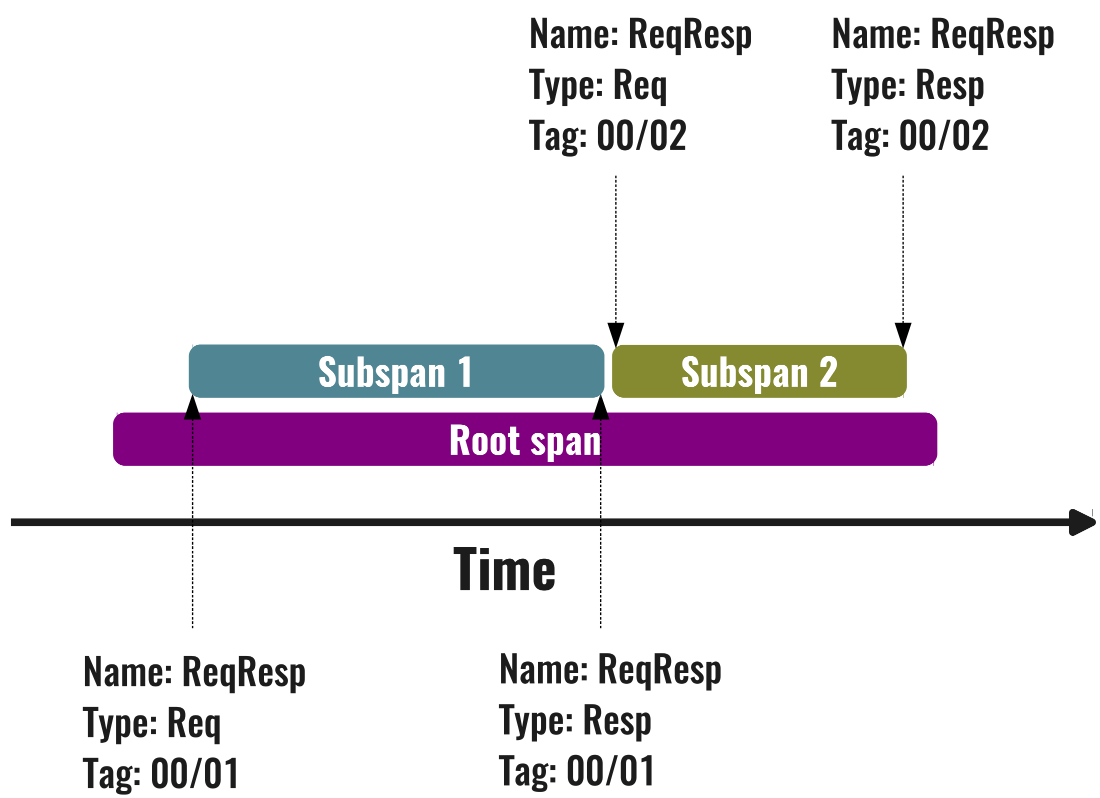


Extracting spans and sub spans

Many subspans may be generated during a span's lifetime.

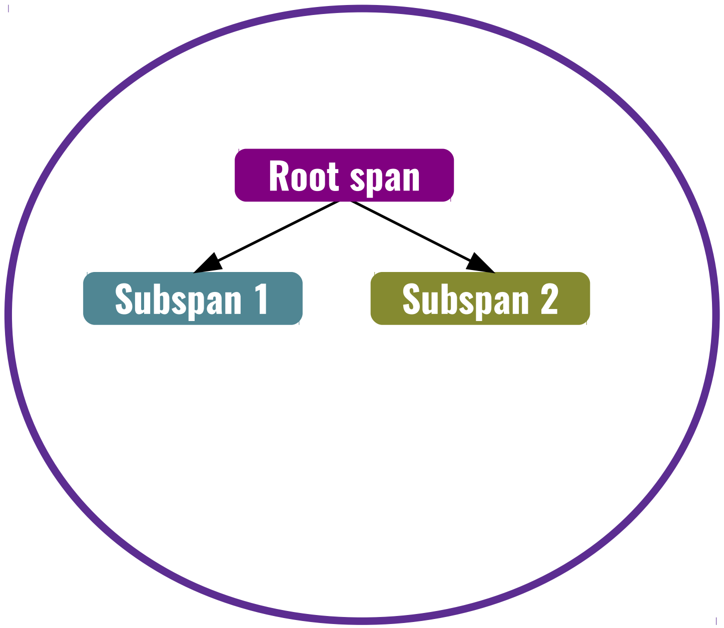
The tag of their parent is embedded in their tag.

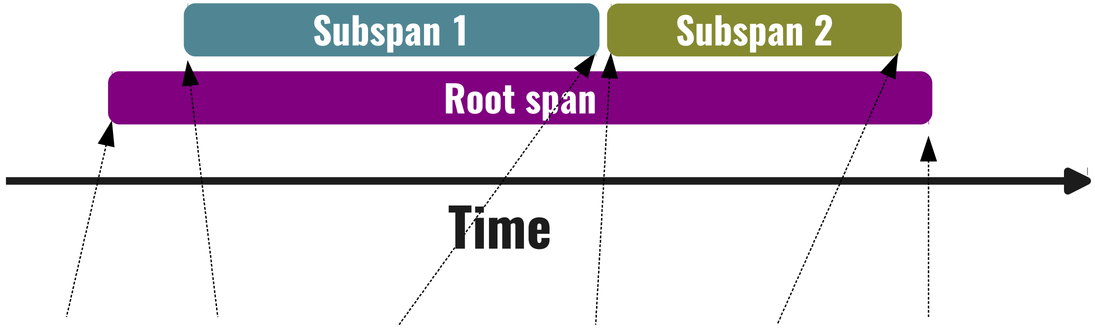




Extracting spans and sub spans

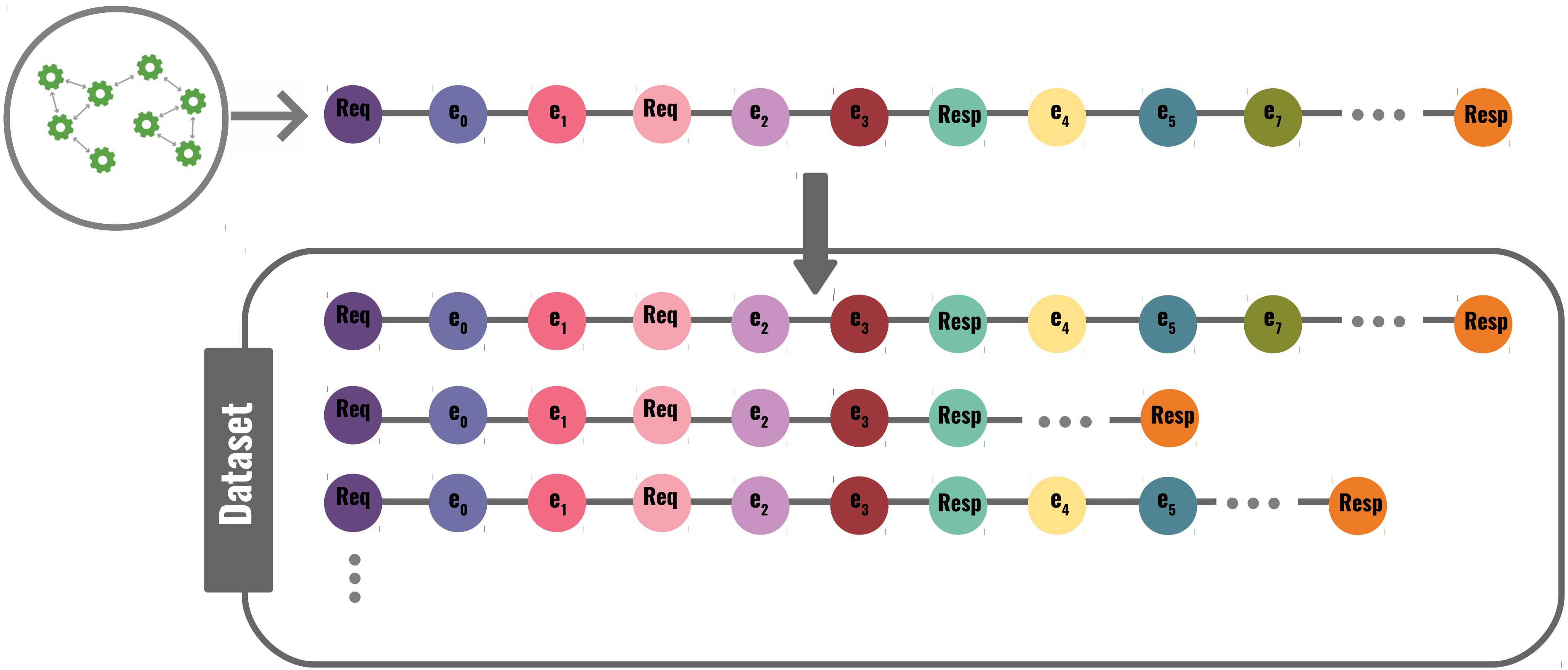
- Many subspans may be generated during a span's lifetime.
- The tag of their parent is embedded in their tag.





Dataset

Tens of userspace and kernel events happen during spans as well, and we put them in the right place in the sequence.



Machine Learning Module

Two main forms of anomalies are point anomalies and collective anomalies.

Point anomalies vs. Collective anomalies

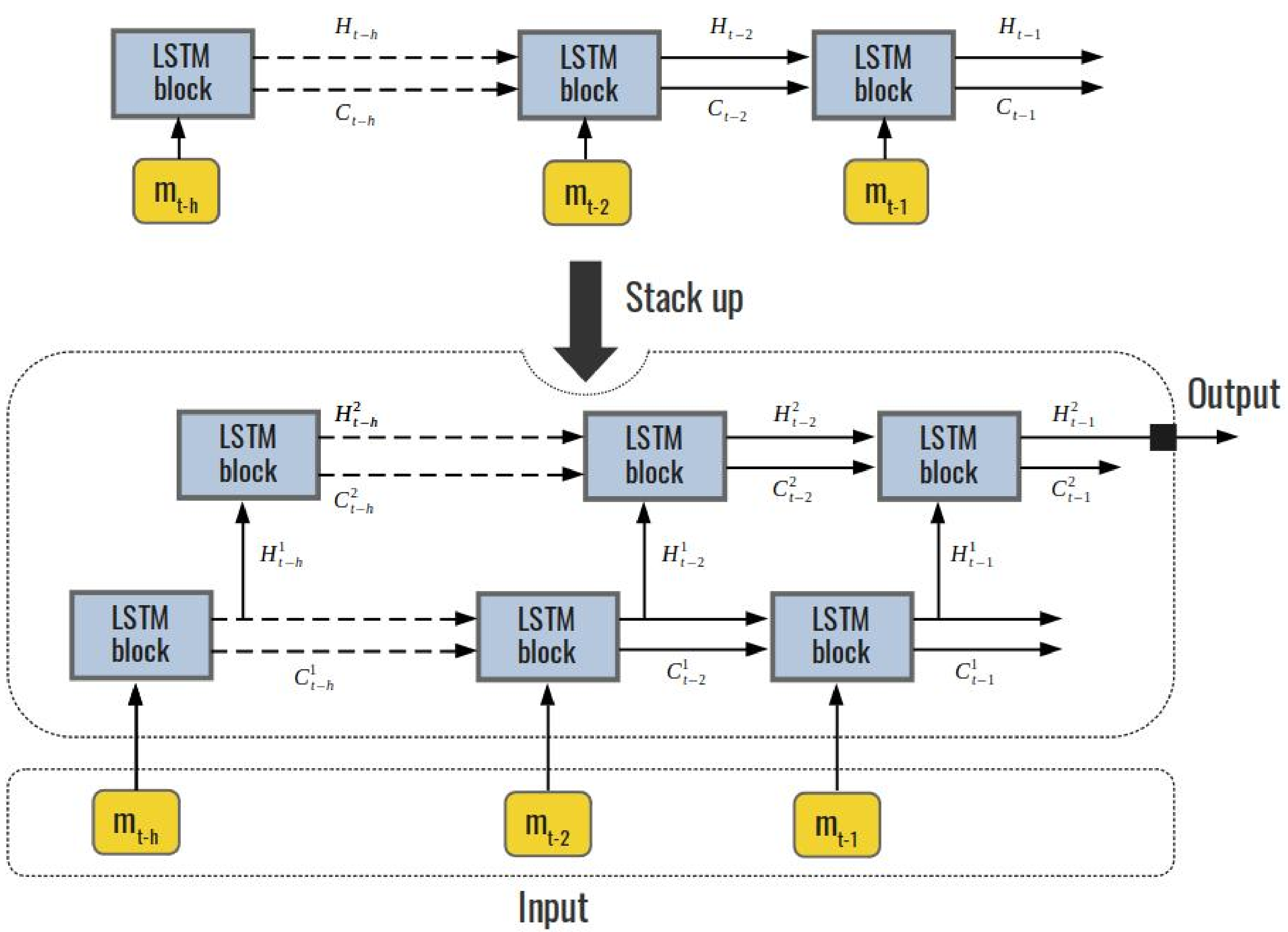
- **Point** anomalies are data points that are different from normal data.
- It's not always possible to detect the individual data points as anomalies by themselves.
- However, their occurrence together as a collection may cause **collective anomalies.**
- A limited number of events can be the result of an action. Therefore few of the possible events can appear as the next event in the sequence.

LSTM

- The fundamental intuition behind this work is natural language processing.
- Events as elements of a sequence follow specific patterns and grammar rules.
- We used an **LSTM** neural network to learn a model of event patterns from normal execution.
- In this way, we avoid creating labeled datasets for supervised learning and the difficulty of interpreting clustering.

L S T M

Learning Module



L S T M

Learning Module

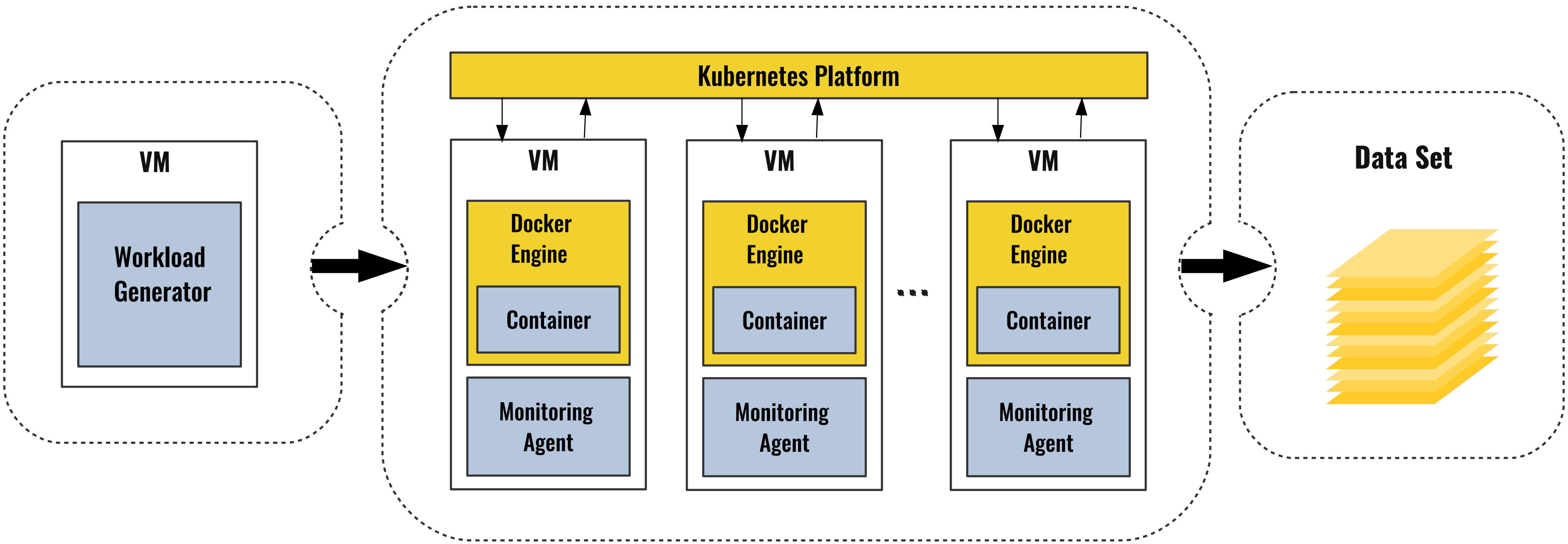
- The model learns a probability distribution $Pr(m_t = k_i \mid m_{t-h}, \ldots, m_{t-2}, m_{t-1})$ that maximizes the probability of the training sequence.
- The strategy is to sort the possible events based on their probabilities $Pr[m_t \mid w]$.
- An event is normal if it's among the top g candidates.
- Otherwise, that event is flagged as abnormal.

future work

An open-source microservices-based application is developed for evaluating our proposed anomaly detection method.

We deployed several instances of each service at the same time using Docker and Kubernetes.

This time we use Jaeger to collect the data.



Thank you for your attention!

Questions?

Iman.kohyarnejadfard@polymtl.ca
https://github.com/kohyar

References

[1] Z. Xu, X. Yu, Y. Feng, J. Hu, Z. Tari, and F. Han. A multi-module anomaly detection scheme based on system call prediction. In 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), pages 1376–1381, June 2013.

[2] A. Liu, C. Martin, T. Hetherington, and S. Matzner. A comparison of system call feature representations for insider threat detection. In Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, pages 340–347, June 2005.

[3] Michael Dymshits, Ben Myara, and David Tolpin. Process monitoring on sequences of system call count vectors. 2017 International Carnahan Conference on Security Technology (ICCST), pages 1–5, 2017.

[4] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. Deep learning for classification of malware system call sequences. In Byeong Ho Kang and Quan Bai, editors, AI 2016: Advances in Artificial Intelligence, pages 137–149, Cham, 2016. Springer Iternational Publishing.

[5] Mathieu Desnoyers and Michel Dagenais. The lttng tracer : A low impact performance and behavior monitor for gnu / linux. In OLS Ottawa Linux Symposium, 2006.

[6] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg, 2006.

[7] Ulrich H.-G. Kre. Advances in kernel methods. chapter Pairwise Classification and Support Vector Machines, pages 255–268. MIT Press, Cambridge, MA, USA, 1999.

[8] Du, Qingfeng, Tiandi Xie, and Yu He. "Anomaly Detection and Diagnosis for Container-Based Microservices with Performance Monitoring." International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, 2018.

[9] Cao, Wei, Zhiying Cao, and Xiuguo Zhang. "Research on Microservice Anomaly Detection Technology Based on Conditional Random Field." Journal of Physics: Conference Series. Vol. 1213. No. 4. IOP Publishing, 2019.

[10] Nikiforov, Roman. "Clustering-based Anomaly Detection for microservices." arXiv preprint arXiv:1810.02762 (2018).

[11] Pahl, Marc-Oliver, and François-Xavier Aubet. "All eyes on you: Distributed Multi-Dimensional IoT microservice anomaly detection." 2018 14th International Conference on Network and Service Management (CNSM). IEEE, 2018.

[12] Nandi, Animesh, et al. "Anomaly detection using program control flow graph mining from execution logs." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016.