Designing an Alert Correlation Engine using Mutual Information values

Abhijit Bhadra
MCS-DS
University of Illinois, Urbana-Champaign
abhadra2@illinois.edu

Sanjeev Kumar
MCS-DS
University of Illinois, Urbana-Champaign
sanjeev5@illinois.edu

Swati Nanda (captain)
MCS-DS
University of Illinois, Urbana-Champaign
swatin2@illinois.edu

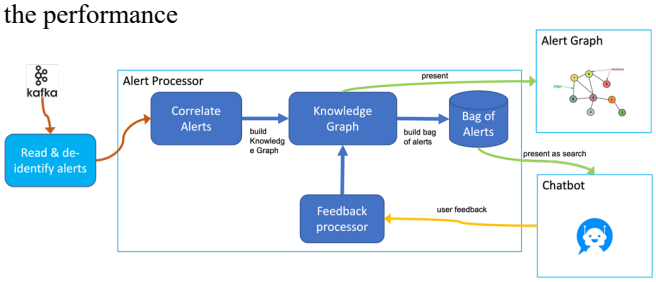
Abstract - *The modern scaled digital transformation has made it difficult for the humans to keep up with the ephemeral state of IT workloads and processes although most of them has made significant investment on monitoring the application, infrastructure, and network. In case of any outage, these monitoring systems generate different alerts, but they do not generate them at the same time and not in sequence. It takes time for IT operations to find out the relation between these alerts and they end up creating too many tickets for different teams. We intend to correlate these alerts by collecting, analysing, and building a knowledge graph between them so that it can predict and group the future similar events that may affect availability and performance. We also want to improve the performance of the model through relevant feedback channel.*

I. INTRODUCTION

Application monitoring is the process of collecting different performance metrics and log data to help developers track availability, bugs, resource use, and changes to performance in applications that affect the end-user experience. Network monitoring provides the information that network administrators need to determine, in real time, whether a network is running optimally. Infrastructure monitoring provides visibility on all the assets that are necessary to deliver and support IT services: data centres, servers, storage, and other equipment. IT operation performs their day-to-day task and mitigate error scenarios via multiple alerts generated from these monitoring systems. It empowers the users- SRE's, L1/L2, developers, who rely on these alerts to ensure health and well-being of the services and environment. Sometimes these alerts become overwhelming and create unwanted noise. Reading through each alert and analysing to get to the error pattern and identifying actionable alerts is a daunting task resulting in lengthy troubleshooting and remediation cycles and high mean time to resolution (MTTR).
In this project we are trying to analyse the alerts and associate them so that we have a model which by identifying the symptoms can predict probable problems upfront so that corrective actions can be taken to prevent system snags. By effectively identifying the correlated alerts, we could reduce the alert fatigue and work on most urgent problem first, with reduced MTTRs.

II. PROPOSAL

One approach for solving the problems of Alert Correlation, could be to find correlated alerts on the entities(services) based on their mutual information which is dependent on their occurrence in a near time window. We will be generating a score and subsequently a knowledge graph based on that mutual information. Higher similarity score means strong correlation between the events. We should be able to present it in a graphical form or use Language Model that can be searched or suggest probable related symptoms from past learning. If time permits, we will try to incorporate the feedback to the model to improve the performance



III. PROPOSED METHOD

Following technical steps will have to be performed in order to develop our solution :

- Prepare some fake alarm dataset based on different incidents. We might use Kafka to ingest the Alarms.
- Group the alerts by entity and type over a window of 5 minutes. Associate the alerts and build a weighted knowledge graph for the alerts as vertex and occurrence count in the window as the score of association/correlation.
- Create a bag of alerts using stemming, so that it can be used for symptom retrieval by the End user
- Over time use the feedback from user and update knowledge graph by scoring up the positive feedback correlations
- Present this correlation via some graphical representation or if time permits – we will create a chatbot to query for outage indicators

IV. TASKS IDENTIFIED

In order to achieve our end objective, these are the high-level tasks and estimation in hour identified

- Data cleanup / preparation – 6 hours
- Implement scoring function – 6 hours
- Develop REST API for frontend interaction – 4 hours
- Build Knowledge graph with limited datasets – 5 hours
- Create a bag of alerts – 7 hours
- Implement user interface – 10 hours

- Implement Feedback loop – 10 hours
- Integration and Testing – 8 hours
- Document Preparation – 3 hours
- Misc. items – 4 hours

Our code will be written mainly in Java and Python

V. EVALUATION METRICS

We will be using Cranfield evaluation methodology for the evaluation. Dummy alerts will be generated, and we will identify their correlation manually. A relevance matrix will be generated for each test alert, and this will be used to verify the accuracy and preciseness of our algorithm