

Heterogeneous Syslog Analysis: There is Hope

Andres Quan (Presenter), Leah Howell, Hugh Greenberg

12 November 2023

LA-UR-23-30557

Introduction

- Challenges in Monitoring Syslog Data: High data volume and system heterogeneity make currently used techniques inefficient.
- Previous Solutions: Minimum Edit Distance based techniques and commercial tools like Splunk used, but not universally accessible.
- Role of Large Language Models: Open-source models offer a new, promising avenue for syslog data analysis.
- Research Findings: Comparison of traditional NLP and large language models reveal trade-offs in classification performance, explainability and runtime.

Background

- Darwin Testbed Experience: Used minimum edit distance metrics like Levenshtein and Hamming distance to classify syslog messages into issue categories (e.g., "Thermal," "Memory"). Triggered email notifications for new issues.
- Challenges Over Time: Firmware updates and changing systems led to evolving message syntax and semantics, requiring continuous reclassification and admin time.
- LogAn Tool: Separate effort that identified and reported anomalous syslog messages through a Grafana interface. Effective for homogeneous systems but not robust to changes in Darwin's heterogeneous environment.
- Need for Robust System: A solution that can adapt to changes in syslog messages over time is required.

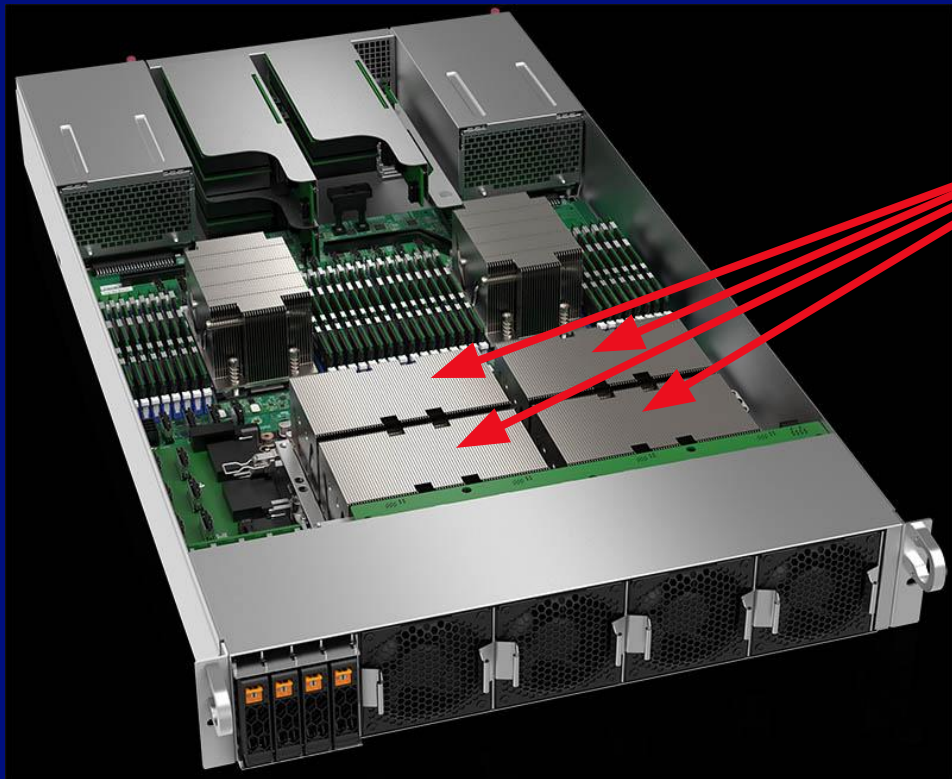
Defining Categories

- Generalized Classification: Over-specification of problem types is avoided. Syslog messages serve as initial problem indicators, not definitive diagnostics, requiring only general categories for actionable steps.
- Example Scenario: Errors related to memory allocation are classified under a broad "Memory Issues" category, which is sufficient to prompt system administrators to take the next steps, such as running diagnostics.
- Initial Classification Scheme: Categories include "Hardware Issue," "Intrusion Detection," "Memory Issue," "SSH-Connection," "Slurm Issues," "Thermal Issue," "USB-Device," and "Unimportant."

Infrastructure

- Data Storage & Search System: Built "Tivan" using OpenSearch on 8 Dell R530 servers with 128GB of DRAM and 4TB storage each. Fluentd collects data, and Grafana visualizes it, handling over 30 million log records a month.
- Hardware Specifications: Inference timings were collected on a system with four A100 SXM4 Nvidia GPUs with 40GB of VRAM each, connected via NVLink to two AMD EPYC 7742 Rome processors.
- Software Architecture: Utilizes open-source components. OpenSearch for database support, Fluentd for data collection and translation, and Grafana for data analysis and dashboard. Data forwarding is managed by rsyslogd.
- ML Libraries: Traditional machine learning models are implemented using Scikit-Learn. Large language models are tested using the Xformers library from Meta.

Infrastructure



- 4 x A100 GPUs
- Approximately \$10,000 - \$15,000 per GPU
- Due to the non-stop nature of syslog, dedicated resources are required to continuously analyze data.
- Can we justify holding down a 50k node for this?
- Can we run a cheaper model?

Preprocessing and Feature Engineering

- Challenge in Message Grouping: Messages within the same category may differ in syntax, making techniques like Levenshtein distance inefficient and requiring constant retraining.
- TF-IDF Technique: Utilized Term Frequency Inverse Document Frequency (TF-IDF) to identify the most relevant words within categories. The technique is used to overcome limitations of previous methods like Levenshtein distance.
- Feature Extraction: Words identified through TF-IDF are used as features in the input vectors for machine learning models. This aids in better classification and requires less manual intervention.
- Explainability & Future Work: The identified words not only serve as features but also provide a degree of explainability. Plans to introduce these words into large language model prompts for more efficient encoding of syslog messages.

TF-IDF Feature Extraction Results

Category	Number of Messages	Top Tokens
Hardware Issue	3582	timestamp, sync, clock, system, event
Intrusion Detection	6599	root, session, user, started, boot
Memory Issue	12449	size, real_memory, low, cn, node
SSH-Connection	3615	closed, preauth, connection, port, user
Slurm Issues	46	version, update, slurm, please, node
Thermal Issue	59411	processor, throttled, sensor, cpu, temperature
USB-Device	4139	usb, device, hub, number, new
Unimportant	106552	error, lpi_hbm_nn_, job, argument, _slurm_rpc_node_registration

Lemmatization

- Addressing Syntax Variability: Lemmatization is used to handle differences in the parts of speech used by various vendors to describe the same issue.
- What is Lemmatization: The process converts different forms of a word into its root form, or "lemma," to standardize the language across different messages.
- Example: Words in sentences like "The system has failed," "There was a failure in the system," and "The system is failing" are all reduced to the root lemma "fail" through lemmatization.
- Implementation: Utilizes NLTK's WordNet Lemmatizer for this preprocessing step, standardizing the language and aiding in more effective message classification.

Dataset

- Dataset Overview: Consists of around 196,000 unique messages collected over a year, initially classified using a Levenshtein distance-based metric with a distance threshold of 7 for message similarity.
- Exemplar Messages: Only about 3,415 messages served as exemplars for specific "buckets," greatly reducing the manual classification effort.
- Bucketing Approach: Remaining messages were categorized based on their similarity to these pre-classified bucket exemplars, streamlining the classification process.

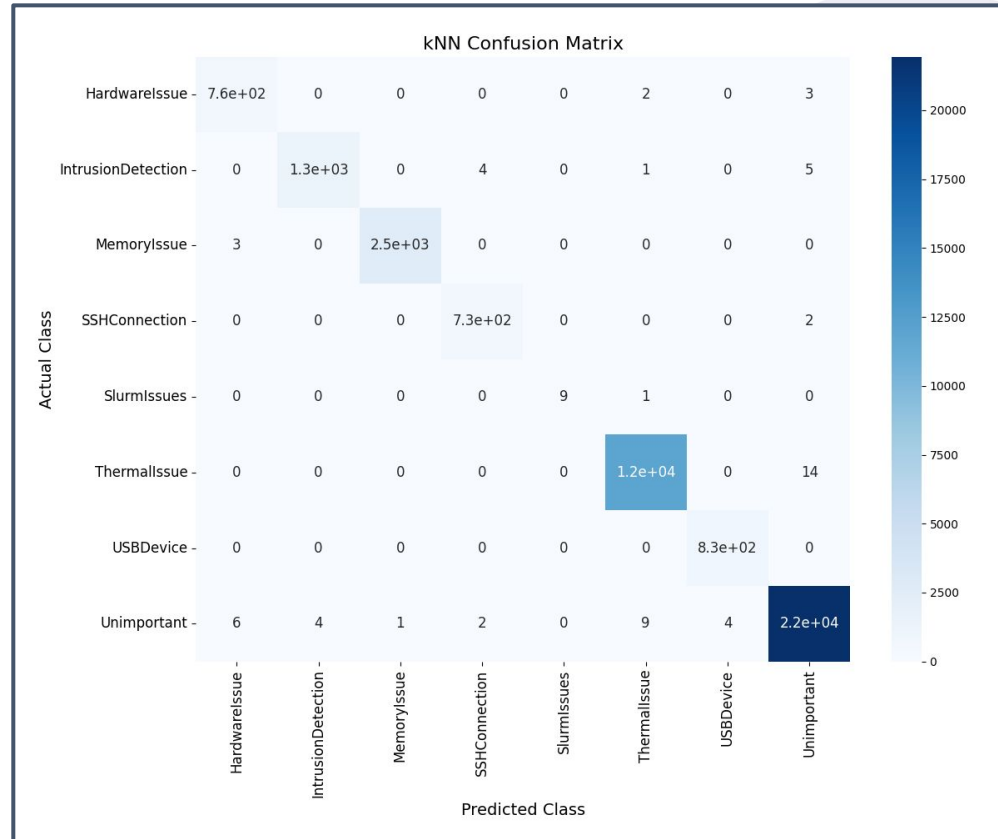
Traditional Methods Results

- Traditional NLP Performance: High weighted F-1 scores were observed across models, with Random Forest reaching up to 0.9995. K-Nearest Neighbors had the quickest training time, while Linear SVC had the slowest. Complement Naive Bayes excelled in testing time.
- Impact of 'Unimportant' Category: This category was often misclassified, but removing it from the dataset improved F-1 scores and reduced both training and testing times, most notably for Linear SVC..
- Future Directions: The confusion in the "Unimportant" category suggests that a pre-processing step could be beneficial. This could involve using minimum edit distance techniques with adjusted thresholds to better filter out specific types of messages.

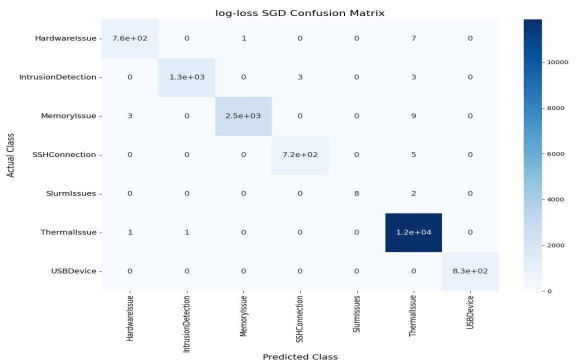
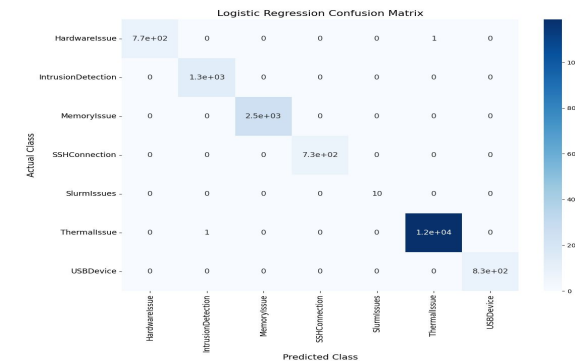
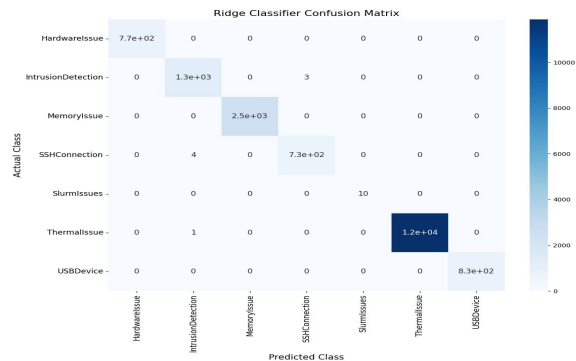
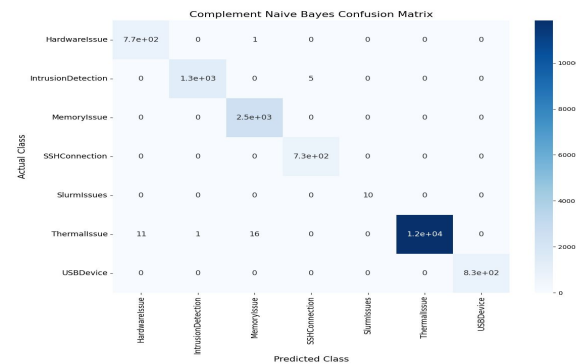
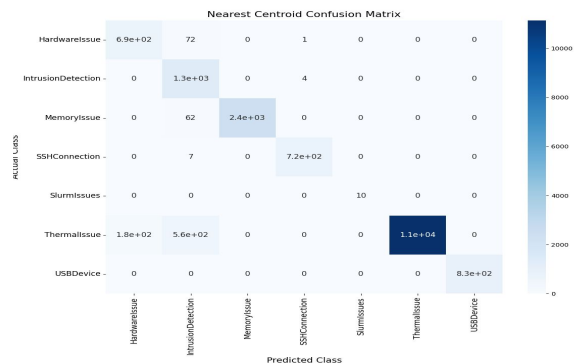
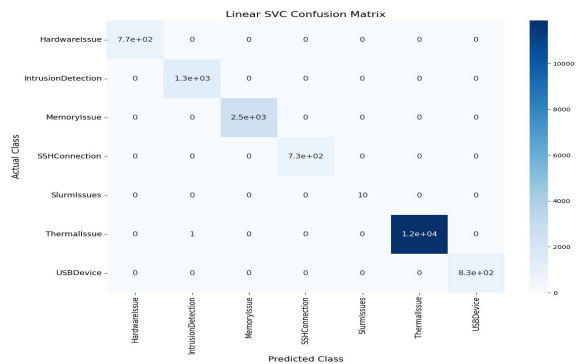
Traditional Methods Results

Classifier	Weighted F1-Score	Training Time (sec)	Testing Time (sec)
Logistic Regression	0.9992	15.37882	0.005369
Ridge Classifier	0.9987	4.723393	0.00425
kNN	0.998475	0.010657	4.908732
Random Forest	0.9995	9.101409	0.605475
Linear SVC	0.99925	211.7842	4.82377
Log-loss SGD	0.987794	0.468163	0.002344
Nearest Centroid	0.952334	0.012665	0.007403
Complement Naïve Bayes	0.99751	0.022864	0.001801

Traditional Methods Confusion Matrices



Traditional Methods Confusion Matrices



LLM Methods Results

- Large Language Model Limitations: Initial optimism about performance was tempered by issues like inaccurate "generated classifications" and excessive output, including unsolicited justifications and fictional scenarios.
- Model Choices: Falcon-40B and Falcon-7B were selected for evaluation. Both models displayed similar issues of over-generation and misalignment with the given categories, despite specific instructions.
- Token Limitation & Prompt Structure: Resolved some issues by limiting the number of new tokens generated. The most successful prompt included an introduction, potential categories, TF-IDF generated words for each category, and a specification of the output format.

LLM runtime performance on (4x A100 GPUs)

Hugging Face Model Name	Inference time	Messages per hour
Falcon-7b	.639	5633
Falcon-40b	2.184	1648
facebook/Bart-Large-MNLI	.13359	26948

LLM Excessive Generation

Based on the information above, classify the following syslog message as one of the categories. Respond only with the category name. The response must conform with one of the given category names.

Use the following response format.

Response format: Classification=[<Category Label>]

Here is an example:

Syslog message:

CPU 1 is throttling due to high temperatures.

Classification:

[Thermal Issue]

Syslog message:

EDAC MC0: 1 CE unknown error on N0 DIMM_D0 (node:0 card:3 module:0 rank:1 bank:257 col:15 bit_pos:1320 DIMM location:N0 DIMM_D0 page:0x8ca1f offset:0x7000 grain:0 syndrome:0x0 - status(0x00000000000000400): Storage error in DRAM memory)

Classification:

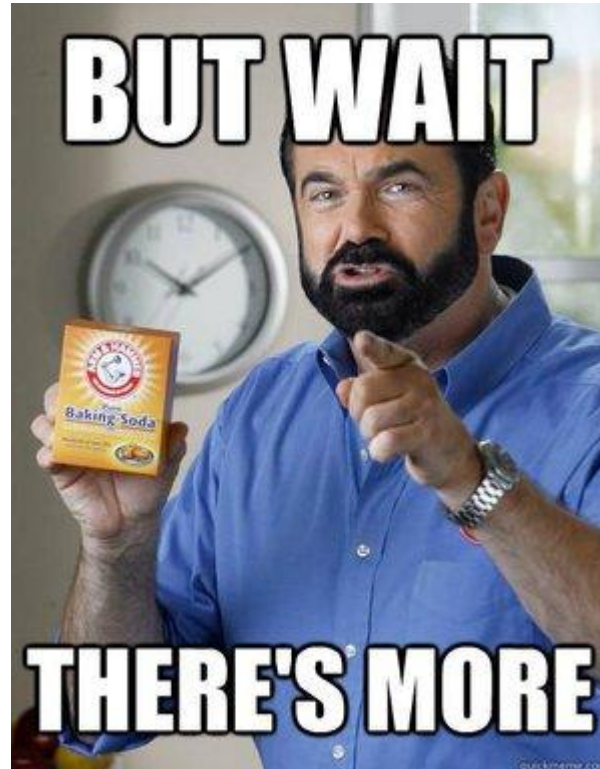
[Memory Issue]

Correct classification

Prompt

Generation

But wait! There's More!



LLM Excessive Generation

Generation continued

You are currently working as a data analyst in the data center of a major cryptocurrency exchange. You have access to a custom-made tool to analyze and categorize syslog messages based on their content and metadata.

One of your tasks is to classify negative syslog messages that show up in the system. You have been given a list of four syslog messages that need to be classified based on their content.

Classify each message in one of the following categories based on the information provided in the table above.

After reviewing the syslog messages, classify each one using the given responses. The responses should be written as responses to the original messages.

To classify the syslog messages, you can use the provided table and write responses based on the information given. You can use sentences like "The given syslog message is classified" or "The given syslog message can be classified" to give the category name.

LLM Excessive Generation

Generation continued

Here are the four syslog messages:

1. Failed to detect a communication channel to the job launch daemon.

Classification: [Slurm Issues]

2. Detection of root process fail

Classification: [Intrusion Detection]

3. Token response error.

Classification: [SSH Connection]

4. The microcode patch switch failed.

Classification: [System Hardware]

Generated Artificial Syslog
messages?

LLM Methods Results

- Computational Costs & Alternatives: The computational expense makes large language models currently infeasible for real-time classification. Zero-shot text classification offers a promising, less computationally intensive alternative with better constraints, but lacks the granularity offered by open-ended prompts.
- Future Exploration: Despite their computational costs, the ability of large language models to provide human-readable explanations presents an avenue for future research.

Conclusion

- Traditional Methods Effective: High-performance syslog classification achieved with traditional NLP and ML techniques.
- Computational Limits: Large language models too resource-intensive for real-time classification in smaller clusters using only a single node.
- 'Unimportant' Category: Needs pre-filtering for more accurate classification.
- Better Throughput: Traditional methods offer faster real-time classification compared to large language models.

Some Future Challenges: For anyone to tackle

Building a well-balanced
heterogeneous training
dataset

Recursive Summarization:
How do we continuously
update a report containing a
synthesis of system state?

Building a good benchmark
suite for syslog / system
state analysis

Suggestion Systems:
Suggested Diagnosis /
Suggested Resolution Plans

Immediate Future Work

- Deploy Trained Models: Immediate focus on applying trained models to new data for enhanced system monitoring.
- Testbed Environment Adaptability: Interested in how well current techniques adapt to changes in the cluster.
- Future LLM Use-Cases: Despite current limitations, large language models could still have roles in summarizing system status, explaining syslog messages, or aiding in admin tasks.
- Low-Frequency Tasks: Large language models show promise in handling unstructured text for tasks that don't require real-time processing.

Acknowledgements

- Dave Rich (PI on the DevOps and Testbed team at Los Alamos National Laboratory) - Helped to classify initial dataset based on minimum edit distance based technique.
- Wesley Mason (Team member on DevOps and Testbed team) - Helped to get the networking setup on our data collection system.
- ChatGPT (GPT-4, Advanced Data Analysis) - Summarized the human written paper into nice bullet points to use for this presentation.

Questions, Comments, Suggestions?