Investigating the Efficacy of Unstructured Text Analysis for Node Failure Detection in Syslog

Virginia K. Felkner*
Information Sciences Institute
University of Southern California
Los Angeles, CA, USA
felkner@isi.edu

Elisabeth Moore
Information Sciences Group
Los Alamos National Laboratory†
Los Alamos, NM, USA
lissa@lanl.gov

***Abstract*—Each node of a supercomputer produces a detailed log of its operation, called a syslog. It is impossible for system administrators to review all syslog data produced by the thousands of compute nodes associated with a single HPC machine. However, analysis of these logs to detect and predict failures is crucial to maintaining the health of supercomputers. The majority of prior work using machine learning to study syslog has relied heavily on the semi-structured nature of system logs. This work investigates syslogs as unstructured, purely textual natural language data. We confirm that treating syslog output as unstructured natural language text does not perform well for node failure prediction, and that researchers must exploit the structure within syslog data to produce more useful results. In order to extract features from syslog text, we employ several popular word embeddings and then cluster both word and message level vectors. Finally, we prepare a dataset for supervised learning by aggregating the syslog into 15-minute time windows and extracting the distribution of clusters within that window. Our failure prediction models achieved a relatively low AUC of .59 using a gradient-boosted random forest. This performance barely out-performs random guessing, but does suggest the presence of signal that could be amplified in future work. We conclude that the incorporation of domain knowledge and structural information into predictive models, rather than a unilateral application of natural language processing techniques is crucial to build deployable tools.**

***Index Terms*—high-performance computing, machine learning, natural language processing, syslog**

I. Introduction

In a high performance computing (HPC) machine, each node produces its own syslog, which is a low-level log of all activities on each compute node. In large HPC systems, such as LANL's Trinity, the volume of syslog data produced is so large that it is impossible for human system administrators to review the syslog manually. However, the detailed analysis of this syslog is crucial for detecting, diagnosing, and mitigating compute node failures. There is clearly a need for automated methods for syslog analysis. In this study, we present preliminary results on the effectiveness of unsupervised natural language processing (NLP) and text mining techniques for compute node failure prediction. The main contributions of our work are:

- Evidence that unilateral application of natural language processing techniques without incorporation of domain knowledge does little better than random chance at predicting node failures.
- Preliminary indications that general syslog messages may not have the properties required to be treated as a subset of a natural language.

II. Related Work

Previous work on using machine learning and NLP to analyze syslog has relied heavily on assumptions about the structure of syslog messages. In particular, a syslog message consists of a timestamp followed by a hostname, process tag, and finally a message. Du et al. considered log messages as a finite set of possible strings of text, called keys, each of which has one or more numerical parameters [3]. This approach does not take advantage of the fact that many keys are similar in meaning and should therefore be grouped together during analysis. Baseman et al. use graph analysis techniques to study syslog messages, which captures collocation information but does not necessarily consider word semantics as fully as a natural language processing approach [1]. Finally, DeLucia et al. do apply natural language processing, specifically topic modeling, to syslog analysis [2]. However, their work leveraged both syslog and job log information. By contrast, our work presented here focuses on prediction of node failures using only data from syslog. We aim to exploit recent developments in natural language processing in order to fully analyze the semantic meaning of syslog text.

III. Dataset Generation

Syslog from Los Alamos National Laboratory's (LANL) HPC machines is generally considered sensitive and not publicly releasable because these machines typically run jobs related to national security. Data sensitivity presented a particular challenge for this project because the author was working remotely due to the COVID-19 pandemic and could not access syslog from actual LANL machines in a secure way.

*This work was performed during an internship at the New Mexico Consortium, affiliated with Los Alamos National Laboratory.

†This manuscript has been approved for unlimited release and has been assigned LA-UR-20-27121. This work has been authored by an employee of Triad National Security, LLC which operates Los Alamos National Laboratory under Contract No. 89233218CNA000001 with the U.S. Department of Energy/National Nuclear Security Administration.

To overcome the data access problem, we spent significant time generating a non-sensitive dataset. Our data was generated on a virtual compute cluster created using a LANL open-source tool called `hpc-collab` [12]. We built a 20-node virtual cluster (VC) to simulate a real HPC environment. Each of the 20 compute nodes had either 1 or 2 cores with 1GB of memory each. The VC also had several infrastructure nodes for file system management, job scheduling, accounting, and logging. The host machine was the New Mexico Consortium's `neutron`. Crucially, all syslog output of the VC is non-sensitive, because the VC did not run any sensitive codes.

We generated data by submitting approximately 10,000 compute-intensive jobs to the VC's Slurm queue, then injecting failures while the jobs were running (a feature available in the `hpc-collab` tool). All jobs ran CLAMR, an open-source fluid dynamics simulation [8], with varying numbers of time steps, ranging from 100 to 1,000,000. Note that the output of the CLAMR jobs themselves is not important for our study. We are not concerned with the results of the simulation; instead, we are using CLAMR as a computation-heavy benchmark job to put a significant load on the CPUs. The variety of time steps simulates users submitting jobs of different lengths, from a few minutes to several hours. To simulate real user submission patterns, jobs were submitted from several user accounts at random times, with each user assigned a predefined subset of potential jobs.

After allowing several thousand compute jobs to enter the scheduling queue, we began killing randomly chosen compute nodes at irregular intervals. This was accomplished by either a graceful or a forced shutdown of an individual virtual machine with a command issued outside the VC, on the host machine. This simulates a sudden hard stop of a user job due to a segmentation fault, hardware problem, or other catastrophic error. Additionally, our framework for issuing kill commands allows for the injection of other kinds of failures. For example, we tried to overwhelm the VC file system by forcing some jobs to write hundreds of MB per second, hoping to observe the effect of slow file system on other nodes. We were unable to collect data from this experiment because the VC ran out of space to store logs before the experiment finished. However, this issue has been fixed in a more recent version of `hpc-collab`, and observing failures caused by file system overload is one of many possible avenues for future work on this project. Our final dataset contains 750,636 lines of raw syslog.

IV. Data Processing Pipeline

Our data processing pipeline included text cleaning, feature extraction, and aggregation. The majority of work was in the feature extraction step, leveraging techniques from NLP to exploit language patterns in the textual data of syslog messages. The processing pipeline starts with raw syslog, which is collected from the VC and saved externally. Fig. 1 shows a summary of the processing pipeline.

The first step of the pipeline is text cleaning. This includes stripping off the timestamp, node name and process tag from

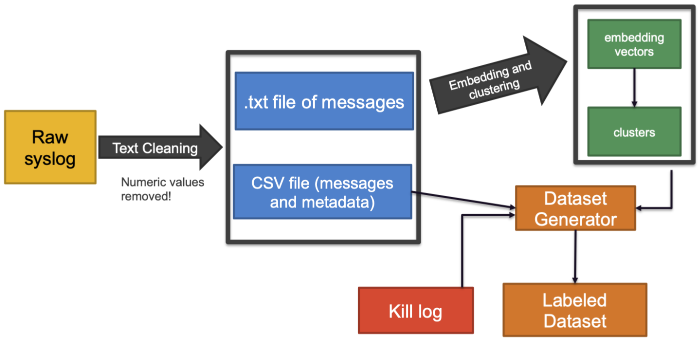

Fig. 1. Processing pipeline for dataset preparation. Raw syslog is cleaned and normalized, then features are extracted via NLP techniques and clustering. These extracted features are then matched with the node kill log to output a final, labeled dataset for supervised learning.

the line, leaving just the text of the log message. Next, numeric values in the message are removed and replaced with placeholder tokens for decimal numbers, hexadecimal numbers, and IP addresses. We chose to remove numeric values because we want to focus on the semantic meaning of numbers in the log message from a natural language perspective, rather than the numeric value itself. In many NLP pipelines, lemmatization is the next step of text preprocessing. Lemmatization is the process of replacing words with their dictionary forms. For example, "starting" and "started" would both be replaced with "start". However, we decided not to lemmatize our data, because in the syslog domain, the distinction between verb tenses such as "starting" and "started," or "stopping" and "stopped" is essential. "Starting" means the operating system is trying to start some process, and "started" means that the process was successfully started. This distinction is important in the context of differentiating between normal and abnormal operation of a compute node. The outputs of the text cleaning step are a plain text file of log messages and a CSV file of messages and metadata, including timestamps and node names.

After text cleaning, the next step of the pipeline is feature extraction. We use word embedding, a common NLP technique, to transform the textual data into vectors. There are a variety of algorithms for word and sentence embedding; these are discussed in detail in section V. Embedding produces a high-dimensional vector for each unique word in the input. Words that are close to one another in the learned vector space are generally similar in meaning and contexts. Due to the properties of these embedding spaces, we can obtain a summary vector for a full (multi-word) log message by taking the mean of the vectors for each individual word in the message. We then used unsupervised clustering to find groupings of word or sentence vectors. The cluster labels were then passed into the final step of the data processing pipeline. The final data processing step was to aggregate the data into time windows. One key difference between syslog text and textual data in other domains is that, while the order of sentences in a paragraph is meaningful, the exact order of log messages is not, because this ordering is affected by a variety of nondeterministic factors at runtime. However, the approximate timing of messages is useful for determining when failures may have occurred, so that system administrators can examine the relevant region of syslog. Therefore, we segment the log file for each node into time windows, then extract the frequency of messages in each time window as an additional feature. We experimented with both disjoint and sliding time windows and found that sliding windows perform marginally better. Finally, the syslog messages grouped by each time window are labelled with whether or not the associated compute node was killed during the given window.

V. Preliminary Experiments

After building a dataset of raw syslog messages grouped into time windows and labeled with the presence or absence of a node failure, we move on to feature extraction via word embeddings and clustering. We first ran a barrage of preliminary experiments to choose the best embedding method and the best clustering algorithm with hyperparameters. Results of these experiments are summarized in Table I below. We tested four different methods of text encoding. The first, TFIDF (term frequency-inverse document frequency), computes a vector for each message using simple frequency calculations and was included as a baseline against which to compare more sophisticated NLP techniques. Word2Vec is an n-gram-based word embedding technique that computes a high-dimensional vector for each unique word in the corpus [6], [7]. Global Vectors for Word Representation (GloVe) similarly computes a vector for each word, based on a word co-occurence matrix [10]. Embeddings from Language Models (ELMo) is a pre-trained deep language model [11]. Such models are extremely popular in the NLP community because they have shown remarkable success on a variety of downstream tasks, often with little or no fine-tuning. For purposes of these early experiments, we used the TensorFlow Hub pre-trained implementation of ELMo, with no additional fine-tuning. We wanted to determine whether a deep neural model would see the same dramatic performance improvements on syslog that it does on human language data, in order to determine whether it would be worth the time and computational resources to fine-tune the model.

We also investigated two clustering algorithms: k-means [5], which is centroid-based, and DBSCAN [4], which is density-based. Additionally, we clustered both individual word vectors and aggregated message vectors. The goal of embedding is to produce distinct and tightly grouped clusters, so that the distribution of clusters within a time window is a meaningful set of features for a ML model. Therefore, we used silhouette score as the evaluation metric for comparing various combinations of embedding and clustering algorithms. Silhouette score is a measure of cluster goodness that essentially measures how similar an element is to other elements in the same cluster and how different that element is from elements in other clusters (a higher silhouette score is considered a better clustering result).

Table I shows the five combinations of techniques that were promising enough to proceed into the next phase of experiments. We tuned the clustering parameters to find the best performance. For k-means clusters, the parameter to be tuned is k, the number of clusters. For DBSCAN clustering, we tuned ϵ, analogous to the minimum density required to define a cluster. For k-means clusters, we want to minimize

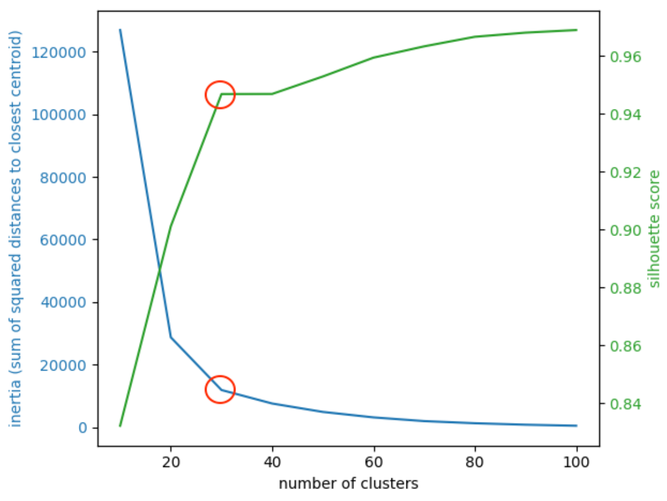

Fig. 2. Hyperparameter tuning for k-means clustering on sentences using word2vec embedding. Empirically, we pick the optimal k at 30.

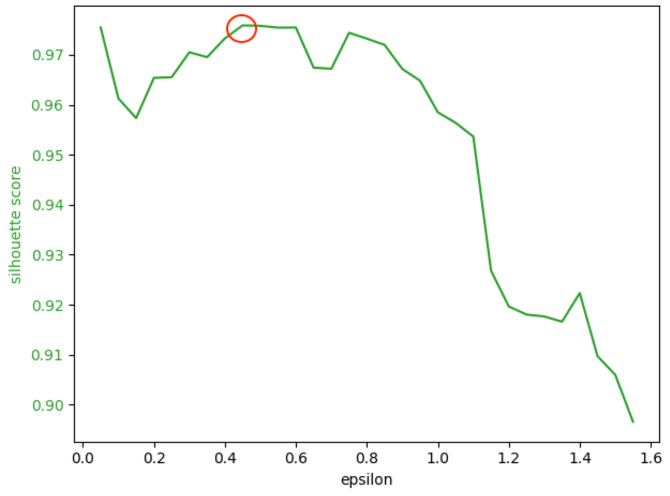

Fig. 3. Hyperparameter tuning for DBSCAN clustering on sentences using word2vec embedding. Optimal ϵ is 0.45.

TABLE I
Silhouette Score for Embeddings and Clustering Algorithms

Clustering Unit and Algorithm	Embedding Method			
	TFIDF	***W2V***	***GloVe***	***ELMo***
words, DBSCAN	0.69	0.87*	0.69	N/A
words, k-means	0.77	0.65	0.73	N/A
messages, DBSCAN	0.69	0.85*	0.75	0.90*
messages, k-means	0.77	0.84*	0.85*	0.71

* considered good enough to proceed with hyperparameter tuning.

TABLE II
ROC-AUC FOR MODELS PREDICTING FAILURES

Model	Max AUC
Random Forest	0.576
Gradient-boosted forest	0.598
Linear SVM	0.58

inertia and maximize silhouette score. Empirically, the optimal k is chosen by looking for the "elbow point" after which adding more clusters only marginally improves performance. This point is marked in Fig. 2 with a red circle. Similarly, Fig 3 shows an example of tuning ϵ. There is no notion of inertia for density-based clusters, but we used a similar empirical approach to find the point after which increasing epsilon no longer improves performance.

Having chosen our potential embedding/clustering pairings, we calculate the cluster distributions present in each time window of raw syslog, and use these cluster distribution vectors as feature inputs to supervised learning models.

VI. Supervised Learning Results

The final phase of the project was to test a variety of traditional supervised machine learning models on our prepared dataset. We used standard `scikit-learn` implementations with default parameters of random forests, gradient-boosted forests, and a linear SVM [9]. Because this was a pilot study to determine the usefulness of our NLP-based feature extraction technique, we did not spend a great deal of time on optimizing these models. The ROC-AUC (Area Under the Receiver Operating Characteristic Curve) scores achieved by each model are listed in Table II. For brevity, we report only the best score achieved by each model, across all combinations of embedding and clustering algorithms.

VII. Conclusions and Future Work

From the results in Table II, it is clear that the features extracted using word embedding and clustering are insufficient on their own for predicting node failures. However, ROC-AUC scores greater than 0.5 indicate that the model is doing better than random guessing, and that there is in fact some useful signal in the cluster distribution data. This means that the NLP methods described in this paper can be used to augment other failure prediction models. Additionally, we conclude that syslog cannot be treated simply as a subset of English, and instead needs to be treated as semi-structured data. Collaboration between machine learning researchers and HPC experts will be crucial in applying these insights to develop deployable and trustworthy ML tools.

There are many interesting directions for future work on this project. The first is improving the dataset itself by making the VC simulation more realistic, in order to better represent an actual HPC environment. In particular, more diverse modes of failure, such as file system and network failures, should be injected. Additionally, there are more directions to consider at the feature extraction stage, including fine-tuning ELMo, further optimizing embedding parameters, and experimenting with other neural language models, e.g. BERT and BERT derivatives. Finally, there are many directions for further exploring supervised learning on our final dataset and improving on current predictive performance.

References

[1] Elisabeth Baseman, Sean Blanchard, Zongze Li, and Song Fu. Relational synthesis of text and numeric data for anomaly detection on computing system logs. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 882–885. IEEE, 2016.

[2] Alexandra DeLucia and Elisabeth Baseman. Work in progress: Topic modeling for hpc job state prediction. In *Proceedings of the First Workshop on Machine Learning for Computing Systems*, pages 1–4, 2018.

[3] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298, 2017.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[5] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[8] D Nicholaeff, N Davis, D Trujillo, and RW Robey. Cell-based adaptive mesh refinement implemented with general purpose graphics processing units. *Los Alamos National Laboratory, Tech. Rep. LA-UR-11-07127*, 2012.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[11] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[12] Steven Senator. hpc-collab. https://github.com/hpc/hpc-collab, 2020.