Modeling High Performance Computing System Log Messages for Early Prediction of Job Outcome

Alexandra DeLucia[*]
Department of Computer Science & Mathematics
Rollins College
Winter Park, Florida
adelucia@rollins.edu

ABSTRACT

As the high performance computing community prepares for the exascale computing era, progress in unsupervised computer maintenance and error prevention becomes increasingly important. Computer maintenance must become semi-supervised and eventually, unsupervised, due to the large amount of information to monitor from the machines, such as system logs, job logs, and temperature reports. A human analyst cannot keep up with this large information flow. Since unassisted human monitoring is not efficient, we turn to other, semi-supervised machine learning methods. Our work uses machine learning techniques to investigate correlations between a computer's system logs and job logs, specifically whether system log behavior can predict the outcome of a job. Outcome refers to the successful completion of a job. We match job log entries with corresponding system log entries and evaluate the usefulness of various system log features for outcome prediction. With these features, we create a job prediction model and test its effectiveness on predicting the outcome of a job at various points in the job's run time, such as number of messages into the job and time passed since the job's start.

CCS CONCEPTS

• Computing methodologies → Supervised learning by classification; **Topic modeling**; **• Computer systems organization → Maintainability and maintenance**;

KEYWORDS

topic modeling, system log analysis, job prediction, machine learning, high performance computing

ACM Reference format:
Alexandra DeLucia. 2018. Modeling High Performance Computing System Log Messages for Early Prediction of Job Outcome. In *Proceedings of Alexandra DeLucia's Senior Honors Thesis, Rollins College, Florida USA, May 2018 (Rollins College '18),* 10 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[*]This work was performed for Honors in the Major. This work was partially performed during a summer internship sponsored by the Science Undergraduate Laboratory Internship program, at the Ultrascale Systems Research Center (USRC) at Los Alamos National Laboratory, supported by the U.S. Department of Energy contract DE-AC52-06NA25396. The publication has been assigned the LANL identifier LA-UR-18-23745.

1 INTRODUCTION

High performance computing (HPC), or supercomputing, is the practice of aggregating powerful computers and using them in parallel to solve complex computational problems. As these high performance computers grow larger and more powerful in scale, with some currently achieving petaflops per second, the challenge in maintaining these computers also grows. The challenge lies in not just keeping pace with the vast amount of data produced by the system, but also analyzing this data for useful information about the system's health. The data is in the form of logs which report on different aspects of the system, from the physical side, such as the processing core temperatures and the computer power usage, to the digital side, such as the processes running on the computers. In our research we hope to alleviate this burden of computer maintenance on system administrators by using machine learning to semi-automate the analysis of system logs to predict errors in programs, or jobs, that run on the computers. Predicting that a job will fail, or worse that a compute node will fail, will help administrators greatly with user support and system maintenance.

There are a multitude of system log analysis tools currently in practice, but these techniques will not be useful in the exascale computing era due to their very manual and, sometimes, system-specific hard-coding. In order for a tool to stand the test of time (and system architecture), it must be generalizable and not require architecture-specific expertise. Since the current tools did not meet these requirements, we decided to borrow system log analysis techniques from a field other than systems—the field of data science. Data science techniques have been successfully applied to other fields, and we can follow their lead by treating the log analysis problem as a big data mining problem. High performance compute logs, which are essentially text and numbers, are ripe for data science analysis. We use a combination of numerical, temporal, and text content analysis techniques to extract features, or create numerical representations, of system logs. These representations are then used as input to train a random forest model to predict job outcome.

The contribution of this work is the exploration of various methods in system log analysis and the creation of a model that can predict the outcome of a job, referred to as a **job state**. We also evaluate how well the model can predict the job outcome at various stages in the job's run time. The intention for this work is to be a stepping stone in the path for a real-time job state prediction tool for system administrators.

This work is organized as follows. We discuss the background information of related works and the system and job logs used in this experiment in Sections 2 and 3, respectively. Then we explore the logs in Section 4 and discuss the system log feature extraction techniques in Section 5. The details of the first phase of our experiment, feature set model testing, is in Section 6, and the second phase, model evaluation on early prediction, is in Section 7. The discussion of the project applications and conclusion follow in Sections 8 and 9, respectively.

2 RELATED WORK

This project is not the first venture into mining system logs (syslogs) to evaluate or monitor system performance, but as far as we know, it is the first to mine syslogs specifically for job monitoring. Most applications of syslog mining are anomaly detection [1], fault diagnosis [12], and computer log event correlating [9]. Although our analysis goals are different, we explored some of the syslog analysis techniques used in these works. The current landscape of syslog analysis has two main groups: a systems-oriented group and a data science-embracing group.

The most common technique of syslog analysis in systems is to create regular expression patterns for syslog lines, usually leaving placeholders for numbers and other variables, and then use these patterns to assign syslog lines into groups. This approach mimics the system administrator job failure debugging process of searching syslogs for keywords that indicate failure. The pattern definitions are usually hand-coded, but some programs exist to generate patterns automatically [10, 11]. We ventured away from this technique because even though some tools can generate the regular expressions, the patterns are still system-specific and not generalizable across machines. However, we mimic this technique in our project by grouping syslog lines based on their process tag, described in Section 5.1.4.

The systems group is aware of limitations in its analysis techniques in the face of system upgrades, which can change log formatting, rendering any defined regular expressions outdated [6]. This self-awareness is indicated by the trend in log analysis research toward applying data science techniques, such as relational analysis and natural language processing techniques to system logs [1, 2]. This data science approach views the logs as a rich inhomogeneous combination of text, numerical, and temporal data. We incorporate this approach in our syslog feature extraction methods and the use of a random forest classification model to predict the job outcome.

We have attempted to represent both approaches in our work in order to explore and evaluate which syslog features work best for our purpose.

3 AVAILABLE DATA

This experiment uses system logs (syslogs) and job logs from the Wolf cluster hosted at Los Alamos National Lab. Wolf has 616 compute nodes, each equipped with 64GB RAM and 16 processing cores. The syslog file contained 2,116,952 lines collected over a one week period and the job log contained 17,172 entries collected over a 17-day period. The two log files overlapped for a period of 5 days, which contained 1,775 jobs and 1,074,157 syslog lines, excluding cancelled jobs. In the following sections we provide background knowledge on the structure and details of syslogs and job logs.

3.1 System Log Description

A system log, or syslog, is a log file of all recorded events on a computer. In the case of a high performance computer, it is a file of events recorded from all nodes, or individual computers, as shown below in Figure 1. Each line in the log is one syslog message with a timestamp, node ID, process tag, and message. The timestamp and node ID together identify the time and node origin of each message. This information is used later to match a message with the compute job that created it. The process tag denotes the process that wrote the message. In the example below, the process tags are "sshd" and "temp_sensors".

```
    <timestamp> <node ID> <process tag> : <message>
Mar 26 03:45:02 wf105 sshd[151500]: pam_unix(sshd:session): session opened for user root by (uid=0)
Mar 26 03:45:02 wf087 TEMP_SENSORS: coretemp-isa-0000 Physical id 0: +28.0°C
Mar 26 03:45:02 wf064 sshd[52958]: pam_unix(sshd:session): session closed for user root
Mar 26 03:45:02 wf087 TEMP_SENSORS: coretemp-isa-0001 Physical id 1: +26.0°C
Mar 26 03:45:02 wf078 TEMP_SENSORS: coretemp-isa-0000 Physical id 0: +30.0°C
```

Figure 1: The format and example syslog lines from a high performance computer.

As seen above, syslogs contains an inhomogeneous combination of text, numerical, and temporal data. Conveniently, most of the text is human-readable, which helps human analysts search through the syslog with keywords to find specific content.

3.2 Job Log Description

A job is an allocation of resources to a user for a specified amount of time. A user submits a piece of code that is too computationally intensive to run on a single CPU, such as a molecular simulation, as a job. The job log keeps records of all jobs that run on the machine. The job log is maintained by the job scheduler, such as Moab or Slurm (Note: the job log used in this experiment was maintained by Moab). The log records details about the job including the job's ID, IDs of the nodes the job ran on, start and end time, and job state, or outcome. An example job log is shown in Figure 2.

```
 <Job ID> <User ID> <GroupID> <Job Name> <Job State>
<Partition> <Time Limit> <Start Time> <End Time> <Node
 List> <Node Count> <Processor Count> <Work Directory>
JobId=108809 UserId=user1 GroupId=group1 Name=program JobState=COMPLETED
Partition=standard TimeLimit=960 StartTime=2017-03-13T22:19:18 EndTime=2017-03-
14T03:24:32 NodeList=wf175 NodeCnt=1 ProcCnt=16 WorkDir=/user1/files

JobId=108810 UserId=user2 GroupId=group2 Name=simulation JobState=COMPLETED
Partition=standard TimeLimit=960 StartTime=2017-03-13T22:19:18 EndTime=2017-03-
14T03:34:58 NodeList=wf[498,261] NodeCnt=2 ProcCnt=32 WorkDir=/user2/files
```

Figure 2: Example job log entries from a high performance computer.

The job state either indicates no inherent problem with the computation (completed, cancelled, timeout), or that some problem occurred (fail, node fail). The job state details are in Table 1.

Table 1: Description and categorization of job states. The "cancelled" job state is not used in our experiment.

Job State	Description	Okay or Problem
CANCELLED*	User cancelled the job	Okay
COMPLETED	Job completed successfully	Okay
FAILED	Job did not complete for some reason (e.g. program bug)	Problem
NODE_FAIL	One or more of the job's compute nodes failed (e.g. filesystem error)	Problem
TIMEOUT	Job did not finished in the allocated time limit	Okay

3.2.1 A Note on Cancelled Jobs. While cancelled jobs are considered "okay", they give us no insight to possible problems on the computer. Jobs are cancelled for reasons outside of the available syslog knowledge. For example, a user can submit an unintended program to run. No line of syslog can reflect this error, because it is human error and not a computer error. For this reason **jobs that are cancelled are not used in this experiment**. Instead, we focused our attention on jobs that can be predicted based on syslog: completed, failed, node fail, and timeout. The rest of the experiment concerns only these four job states.

4 RAW DATA EXPLORATION

Before delving into syslog feature extraction, we explored the job log and syslog datasets. We looked for features that could be used later in the experiment and at general trends in jobs and syslogs.

Our job dataset had an uneven class, or state, distribution, as shown in Figure 3. As expected, the states that are okay, i.e. completed and timeout, greatly outnumbered the jobs that were problematic, i.e. failed and node fail. This low frequency of problematic jobs is good for users but less useful for training a model to predict job state. In addition to an uneven state frequency, there was a trend for jobs to be very short. The majority of jobs in our dataset were under 30 minutes, with most running under 5 minutes as seen in Figure 4. While most of the actual durations of the jobs were very short, we noticed that the time limit setting for a lot of jobs was left at the default value of 960 minutes. The short jobs ran for under 10% of their allocated time (Figure 5). This is most likely due to users overestimating their job's runtime, or more likely, accepting a default time limit set by the scheduler. A possible consequence for this discrepancy in planned versus actual job duration is lower efficiency in job scheduling. A simple solution to this problem could be to lower the default time limit for jobs. The consequence in the scope of this experiment is that we have no way of predicting how close a job is to finishing. We cannot use the time allocation to judge how far along a job is since a job could be 100% finished only 10% into its allocated time. This affects Phase 2 of our experiment, early prediction, discussed in Section 7.

We also examined the syslog messages that were associated with the jobs. As to be expected, the general trend was for the longer jobs to have more syslog messages than shorter jobs. A breakdown of number of syslog messages associated with each job in our dataset is in Figure 6. The data points are colored by

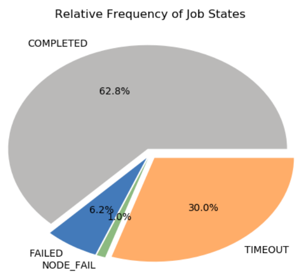


Figure 3: The uneven job state distribution in the dataset. Jobs that are successful occur more frequently than those that fail.

job state. The completed jobs do not have any noticeable trends, but the timeout jobs mainly ended at the same time which lines up with the default time allocation of 960 minutes. The failed jobs would end at different times, so it is not the case that failed jobs failed immediately. The node fail jobs have no noticeable trends. A possible reason for the trend for syslog message frequency to be directly related to job duration is due to scheduled processes. For example, there is a process, TEMP_SENSORS, that logs the processing core temperatures of the node every few minutes or so. Other processes commonly run on the nodes but are not scheduled.

In Figure 7 we look at all of the processes that ran during a job, grouped by job state. Each data point is the distribution of processes that ran during a job. A process is identified by the process tag in each syslog message, as shown in Figure 1. The distribution of processes for a job was obtained by dividing the number of times each process appeared by the number of syslog messages in the job's correlated syslog group. The two most common processes occurred with similar frequency across job states, the OpenSSH daemon (sshd) and the CPU temperature check (TEMP_SENSORS). This was expected since the nodes track all logins, which occur when a job begins, and the temperature check process is a regularly scheduled event. However, there was variation in the frequency of other processes, which indicated that some processes are more correlated with one job state over the others. In Section 5, we discuss using this process frequency distribution as a feature to describe the syslog content.

5 SYSLOG FEATURE EXTRACTION

Syslogs are a challenge for data mining due to the unique inhomogeneous structure of text, numerical, and temporal data. A strategy for handling this data type mix is to separate each data type and analyze it individually. This has been done before with syslog feature extraction for anomaly detection [1]. In this work, we follow their lead and separate the text and numerical and temporal content for individual analysis. Different from other syslog analysis experiments and current state-of-the-art high performance computing (HPC) log analysis tools, we take a zero-resource approach and do not provide any knowledge of the meaning of the syslog messages to the statistical models. Since our work is geared towards exascale

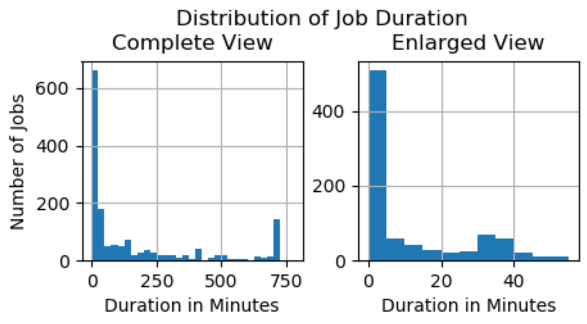


Figure 4: The frequency of all job durations in minutes. Most of the jobs were under 30 minutes.

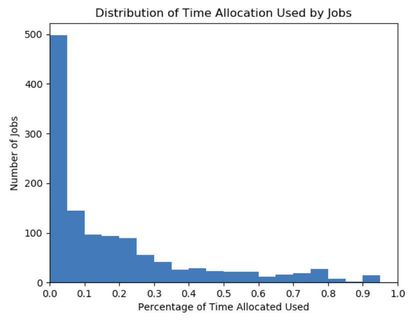


Figure 5: The percentage of allocated time used by the jobs. The default time allocated is 960 minutes.

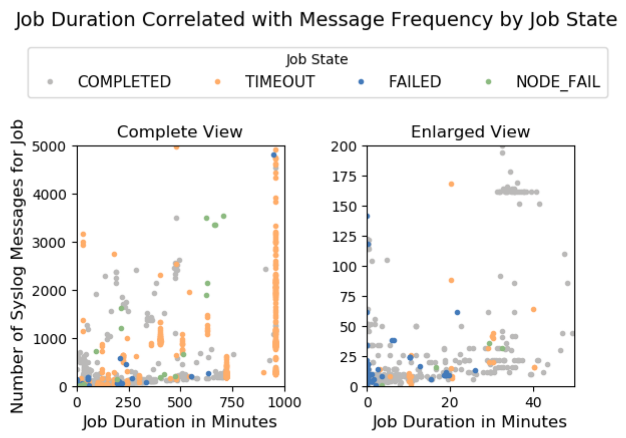


Figure 6: Correlation between the job's duration and the number of syslog messages it produced, colored by job state.

computing, we employ feature extraction techniques that do not require human labor (i.e. avoiding manually-labeled features such as crafted regular expressions). This adaptability ensures that the features can be used in the face of HPC system upgrades or changes.

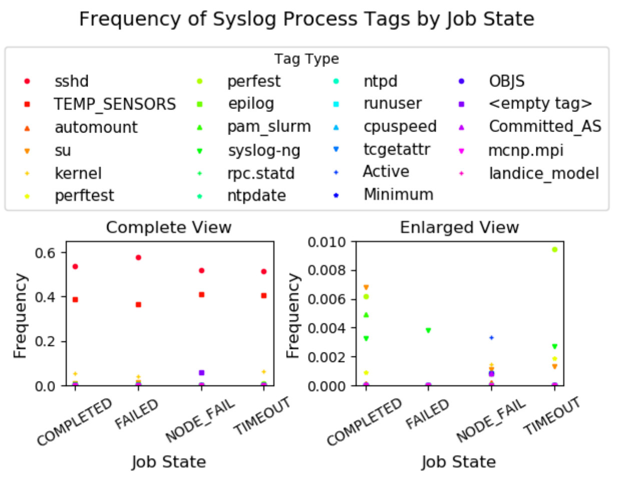


Figure 7: Frequency of processes that run during jobs, grouped by job state.

The following sections discuss our approaches to text, numerical, and temporal feature extraction.

5.1 Extracting Text Features with Topic Modeling and Text Clustering

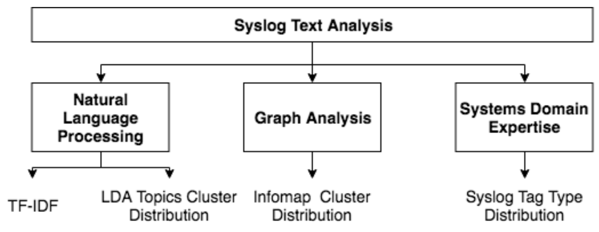

Syslog Text Analysis
Natural Language Processing
Graph Analysis
Systems Domain Expertise
TF-IDF
LDA Topics Cluster Distribution
Infomap Cluster Distribution
Syslog Tag Type Distribution

Figure 8: Text analysis techniques used in this experiment

After the text is separated from the numerical and temporal content, it is analyzed from a variety of approaches. As shown in Figure 8, we employ analysis techniques from the fields of natural language processing, graph analysis, and HPC systems expertise. The end goal of each analysis technique is to describe the text content from a group of syslogs as a numerical feature vector. This compression into a vector is to include the text content as input to the random forest classification model used later in the experiment.

A random forest classifier, discussed in [4], is a collection, or "forest", of decision trees. A decision tree is a hierarchy of decision points which sort the input into different branches based on the values from the inputted feature vector. Each branch ends in a leaf, which represents the tree's classification for the input. A leaf's label is from the set of possible outcomes, e.g. {completed, failed, node fail, timeout}. The random forest classifier uses a collection of decision trees and selects the most popular decision tree prediction as the final prediction.

5.1.1 *Term Frequency-Inverse Document Frequency (TF-IDF) Term Weighting.*

Term frequency-inverse document frequency (TF-IDF) term weighting is a popular “bag-of-words" natural language processing technique[7]. It is the more sophisticated version of counting all unique words, or “tokens", in a syslog group and normalizing their frequency. This is a naive technique because there are words that are common to all syslog groups, and so do not provide any information to the content of any specific group. TF-IDF fixes this problem by taking into account the token's frequency across all syslog groups. It is a weighted distribution across tokens that suppresses popular, i.e. uninformative, tokens, and boosts the unique tokens. We use the Scikit-learn TF-IDF implementation in this project[7].

5.1.2 *Latent Dirichlet Allocation (LDA) Topic Modeling.*

Latent Dirichlet allocation (LDA) topic modeling is another natural language processing technique. LDA is a statistical model centered on the idea that documents, i.e. syslog groups, have multiple “topics"[3]. Topics are latent ideas represented by multiple tokens across documents. For example, a topic in a European history textbook is royalty, and is identified by words such as princess, king, and queen. This topic appears in the textbook across chapters among other topics like war (Crusades, battle, army, armada) and religion (church, Islam, Judaism). In this work, we used topic modeling to generate topics, or groups of tokens from the syslog messages, and then found the distribution over each topic for a group of syslog messages. We use MALLET, a Java package for natural language processing, to find the syslog topics [5].

5.1.3 *Infomap.*

Infomap is a graph clustering algorithm. In order to use it, we created a graph of words, or tokens, from the syslog file. Each node is a token and the edges exist between tokens that appear in the same syslog line. The edges are weighted based on how often tokens appear together. This represents the syslog as a relational structure. This technique has been used in other syslog analyses [1]. Once the graph is created, Infomap finds clusters by using the probability of a random walker to transition within communities of tokens and between communities[8]. Each cluster is a set of tokens. The feature set is similar to the LDA feature set, except we find the distribution over clusters instead of topics.

5.1.4 *Syslog Process Tag Distribution.*

The syslog process tag, or tag, distribution is the frequency of each unique tag in a group of syslog messages. The tags were described in Section 5.1.4. This feature set is the normalized distribution of process tags over a group of syslog messages.

5.2 Extracting Numerical and Temporal Features

As aforementioned, we analyzed the numerical and temporal content separately from the text content. For the numerical content we calculated the average and standard deviation of all the numbers present and the number of numbers for each syslog group. Similarly, for the temporal content we calculated the average and standard deviation of the time difference between successive timestamps for each message in a syslog group.

5.3 Creating Feature Sets

We created feature sets by using each extracted feature alone and creating a new set by pairing each text feature with the temporal and numerical features. This resulted in a total of 11 feature sets, shown in Figure 9. These feature sets are used to build the job state prediction models in Phases 1 and 2.

Figure 9: List of feature sets used in experiment

(1) Temporal
(2) Numerical
(3) Temporal & Numerical
(4) Infomap distribution
(5) LDA distribution
(6) Tag type distribution
(7) TF-IDF distribution
(8) Infomap distribution with Temporal & Numerical
(9) LDA distribution with Temporal & Numerical
(10) Tag type distribution with Temporal & Numerical
(11) TF-IDF distribution with Temporal & Numerical

6 PHASE 1: CREATING JOB STATE PREDICTION MODELS

This first phase of the experiment answers two of our research questions:

(1) How accurately can we predict job state using syslog features?
(2) Which syslog features are most informative?

Our approach to these two questions is described in the following sections.

6.1 Experimental Overview

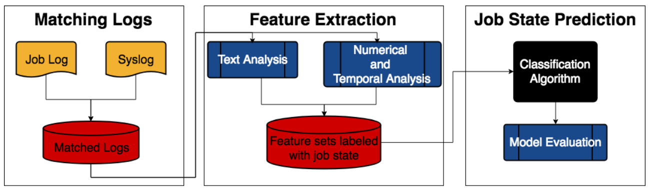


Figure 10: Overview of Phase 1 of the experiment

This phase of the experiment had three main steps, shown in Figure 10. The first step was matching the syslogs to their corresponding job logs. The job logs and syslogs were parsed and organized by time and node origin, and were matched based on this information. Since only one job can run on a node at any time, each syslog message is associated with only one job. An example resulting data set after this step is in Figure 2. Any jobs that did not have matching syslogs were not used in the later steps.

Table 2: An example row from the matched logs table. Syslogs were grouped and matched with their corresponding jobs based on their timestamp and node origin.

Job ID	Job State	Start Time	End Time	Node IDs	Syslogs
1	FAILED	2017-03-13 T22:44:40	2017-03-13 T24:44:40	[40, 60, 200]	[message 1, message 2...]

Once the syslogs were matched to their corresponding jobs they were grouped and pre-processed. The pre-processing was to separate the text, numerical, and temporal data for feature extraction. The text was additionally processed by removing symbols. The methods for extracting features and the list of combinations were discussed in Section 5. Once the features sets were created, we used them as inputs for a random forests model. We evaluated our model on three different prediction tasks: how well it could predict a job's state among all possible states, how well it could predict whether a job is "okay" or is a "problem" (see Table 1), and how well it could distinguish one job state versus the others. We refer to these tasks as "Multiclass", "Okay v Problem", and "One v Rest". The model was evaluated on the precision-recall metrics of F1 score, precision, and recall. Each metric is on a scale from 0 to 1. A high precision indicates that the model has a low false positive rate and a high recall indicates that the model has a low false negative rate. For this model to be useful for system administrators a high recall and precision is necessary so that the problematic jobs can be flagged and looked at, but also the model does not overwhelm the administrators with jobs that will actually be okay. The F1 score is the harmonic mean of the precision and recall scores, where a 1 is perfect and a score of 0.5 is equivalent to random guessing. The results are discussed in the next section.

6.2 Results

A random forests model was trained on each feature set listed in Figure 9, and evaluated based on the tasks "Multiclass", "Okay v Problem", and "One v Rest". For each task the model was assessed on the precision-recall metrics of F1, precision, and recall score. For "One v Rest", the score is a weighted average of the scores for each job state. The weighting is based on the state's frequency in the test set. The results are shown in Figure 11. The scores represent the average score of each feature set model after 200 trials of training and testing. An important note is the y-axis on the figure starts at 0.5, since this is the lowest score relevant for the precision-recall metrics.

The best and worst performing feature set models across the tasks were the term frequency-inverse document frequency (TF-IDF) with numerical and temporal features and the Infomap model, respectively. The TF-IDF with numerical & temporal features model followed the trend of the other models for the text analysis features to work best when paired with the numerical and temporal features. It is a common phenomena in data science for a model to perform better with more context, which the text analysis was able to provide. This phenomenon is also supported by examining the numerical, temporal, and numerical and temporal combination feature set models. The feature set model trained on numerical & temporal feature set outperformed the models trained on the numerical and temporal features alone. Also, all of the feature set models trained on all available features (a text analysis technique paired with temporal and numerical features) outperformed the temporal & numerical model.

All of the feature set models performed the best on the "Okay v Problem" task. This better performance is most likely because it was the easiest task, since it was a binary classification problem. While the "One v Rest" was also a binary classification problem, the score is a weighted average of the model's ability to discriminate between one state versus the others. The lower scores suggest that one or more of the states is pulling down the overall average. The low performing state is probably one with lower frequency, such as node fail or failed. The models' performance on the "One v Rest" task was very similar to their performance on the "Multiclass" task. This close performance is most likely from the same issue of not performing well on predicting a low frequency state. A glimpse into the TF-IDF with temporal & numerical feature set model's prediction decision process is in the next section.

6.3 Analysis of Feature Importance

A machine learning classification model such as random forests is usually regarded as a "black box," where the function that led from the input to the prediction is unknown. While there are concerns with interpreting the inner functions of a black box model, there are ways to try to understand the underlying function. One way is to look at the feature importance of the model. A feature importance score is how influential that specific feature is on the model's prediction. Now, in this context a "feature" is a specific input value, such as the frequency of the token "user" or the average of numbers in a syslog group. The feature importance of the top ten features for the best performing model, TF-IDF with temporal & numerical feature set model, is shown in Figure 12.

For the "Mutliclass" task, the model regarded all of the temporal features and most of the numerical features to be important, along with a few syslog keywords. The importance of these features aligns with the results since the temporal and numerical feature set models performed well even without the text analysis features. The "count_num" feature refers to the numbers in the syslog group while "avg_num" refers to the average number. Syslog messages are very diverse with a wide range of numbers, so it is interesting that the model regarded these numerical features as important. Similar results were obtained in [1] for these numerical features. All of the timing features of total time difference between the first and last syslog message (total_time_diff), the average time difference between messages (avg_time_diff), and the standard deviation of the time differences (std_time_diff) were also very important to the model. The timing features make intuitive sense because the health of a job can be indicated by its activity level, which is shown by how often a job produces syslog messages. The other features the model regarded as important, "pam_unixsulsession", "id", "physical", "root", and "pam_unixsshdsession", are all very common words in the syslogs. The model must have noticed that certain job states have higher frequencies of these tokens.

Different from with the "Multiclass" task, the model regarded more of the syslog keywords to be important for the "Okay v Problem" task. The only temporal or numerical feature in the top 10 list is the total time difference between the first and last syslog message. The rest of the features are popular keywords, except for "user_1" and "user_2", which are two anonymized users from the lab. The model possibly was detecting the presence of a username instead of focusing on two specific users. Or, maybe these two specific users simply ran a lot of bad jobs.

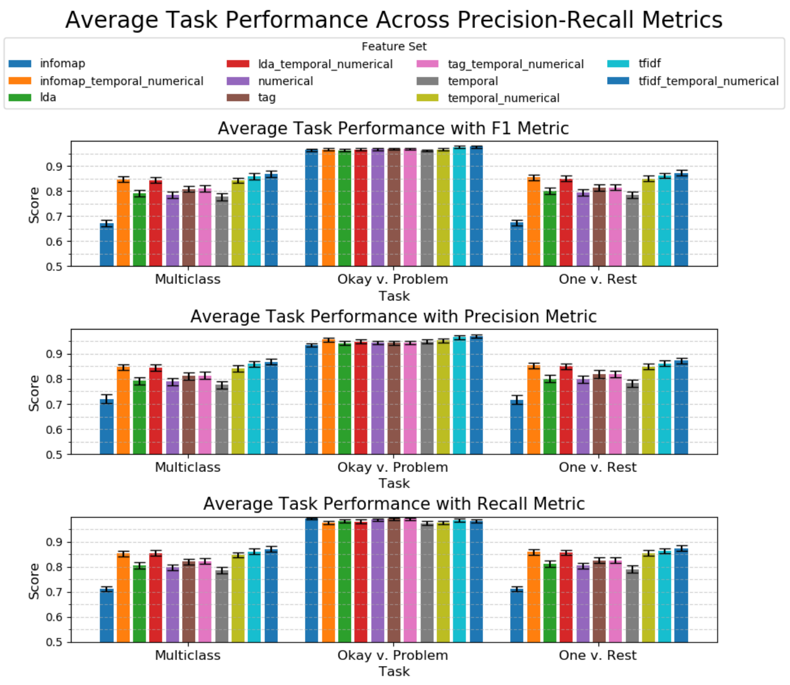


Figure 11: The F1, precision, and recall scores of the random forests model built on the feature sets. The model was evaluated on three different tasks. The error bars denote standard deviation.

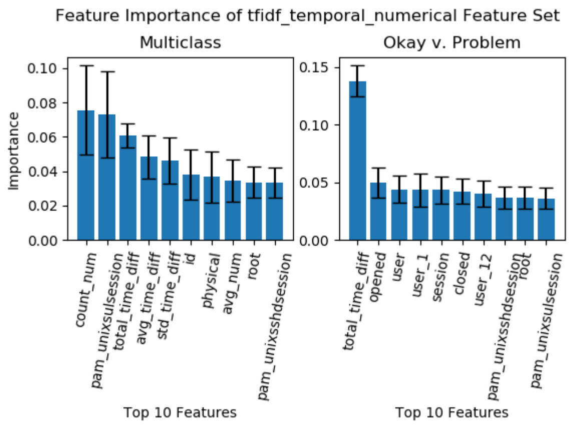


Figure 12: Feature importance of the TF-IDF with temporal & numerical analysis set. These importance scores demonstrate the most influential features the model uses to classify a job.

6.4 A Return to Cancelled Jobs

In Section 3.2.1 we discussed our reasoning for not including cancelled jobs in our experiment. Now, we return to this specific job state and ask: how would our model classify a cancelled job? Or in other words, what label, {completed, failed, node fail, timeout}, would our model predict the cancelled jobs to be? The answer, shown in Figure 13, is the model predicts that most cancelled jobs would have completed successfully. This agrees with our reasoning for removing the cancelled jobs from our experiment. The workload manager labels a job as "complete" when it does not terminate unexpectedly, however the job may not have been the user's definition of "complete", e.g. saving a result in the correct file name, running the correct program, etc. The outside factors needed to predict a job as cancelled are not provided in the available data.

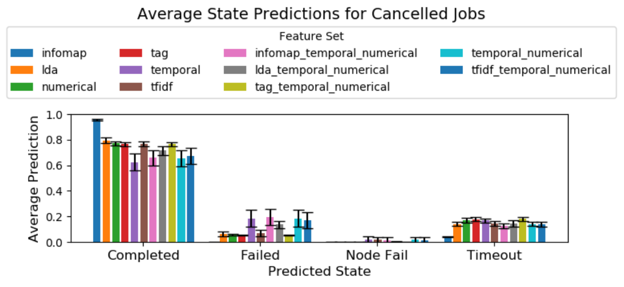


Figure 13: The models' predictions of CANCELLED jobs.

7 PHASE 2: EVALUATING MODELS ON EARLY PREDICTION OF JOB STATE

This second phase of our experiment answers our final research question:

(3) How early can we predict the job state?

We define "early" in two ways: by number of syslog messages and by length of time passed into the duration of the job. The steps for this phase is very similar to the first phase of the experiment, except now we limit the syslog messages that are matched with each job. The rest of the work flow is exactly the same as in Phase 1. Note that in this phase we trained and tested on the limited dataset, and did not use the complete dataset for the training.

7.1 Early Prediction with Limiting Number of Syslog Messages

In order to test how many syslog messages our model needed to predict the job state we limited the number of syslog messages available to train and test the model. Since our goal was early detection, we evaluated our models' performance using only the first 1 to 30 messages. Only incomplete jobs were considered. The results are shown in Figure 14, along with a baseline comparison to the best performing model from Phase 1, the TF-IDF temporal and numerical feature set model. As expected, the models did not perform as well with the limited syslog messages. However, the TF-IDF with temporal and numerical feature set model was still the best performing model across the tasks. The exception to this observation is the "Okay v Problem" task, on which all of the models still performed on par with the baseline. As expected, the models performed better as they were provided more messages.

7.2 Early Prediction with Limiting Time into Job

Similar to the methods used to test the models with limited number of syslog messages, we tested the early prediction ability of the model with regards to time. How many minutes into the job can we predict the job's state or category? We evaluated our models' performance using the syslog messages from the first 1 to 30 minutes of a job. The results, shown in Figure 15, confirm what was expected: the models performed better with more information, or more time into the job. This also relates to the fact that more time allows for more syslog messages to be produced,as shown in Figure 6. The results were similar to those from limiting the number of syslog messages, with the TF-IDF with temporal & numerical feature set model performing the best across tasks.

These early prediction results provide insight on possible settings to include in a job monitoring tool for system administrators, such as the ability to adjust how "early" the tool checks on the jobs.

8 APPLICATIONS OF OUR WORK

While the first phase of our experiment was purely theoretical since we used all of the available syslog, our second phase is tailored directly to applications, since we use the premise of not having the completed syslog file from the job. From our results in the early detection phase of the experiment, we are positive that our model can be used as a tool to help system administrators monitor the jobs. The tasks can be used for different applications. A model trained for the "One v Rest" to discriminate between a node fail and the other states can be used to monitor overall system health. Also, the "Okay v Problem" task is more tailored for overall job monitoring, since the exact state may not matter to a system administrator.

Our model could be expanded and set up as a real-time process on a high performance computer. This process can notify the administrators when a job is predicted to fail, so that the user can be notified and the problem rectified. Or, to completely automate the system, the process can work with the job scheduler to implement automatic checkpointing of jobs that are predicted to fail. With this implementation, a user is able to be notified of their failed job and has a partial snapshot from which to continue the job.

9 CONCLUSION

Our work uses log analysis techniques from the field of systems and data science to extract syslog features and create input for a random forest model to predict job state. The goal of this work is to lay the foundation for a real-time log analysis tool for job monitoring. From the extracted features we created feature sets with a combination of text, numerical, and temporal features from the syslogs. The feature set models had varying rates of success, with the term-frequency inverse document frequency term weighting (TF-IDF) with temporal & numerical feature set model performing the best across all tasks. In addition to determining the best feature set model we also tested our model's ability to predict job state with limited syslog messages, meant to represent how far into a job we can predict the outcome. As expected, the feature set models did not perform as well as with the entire syslog. Still, the results are promising and we plan to build upon this work in the future.

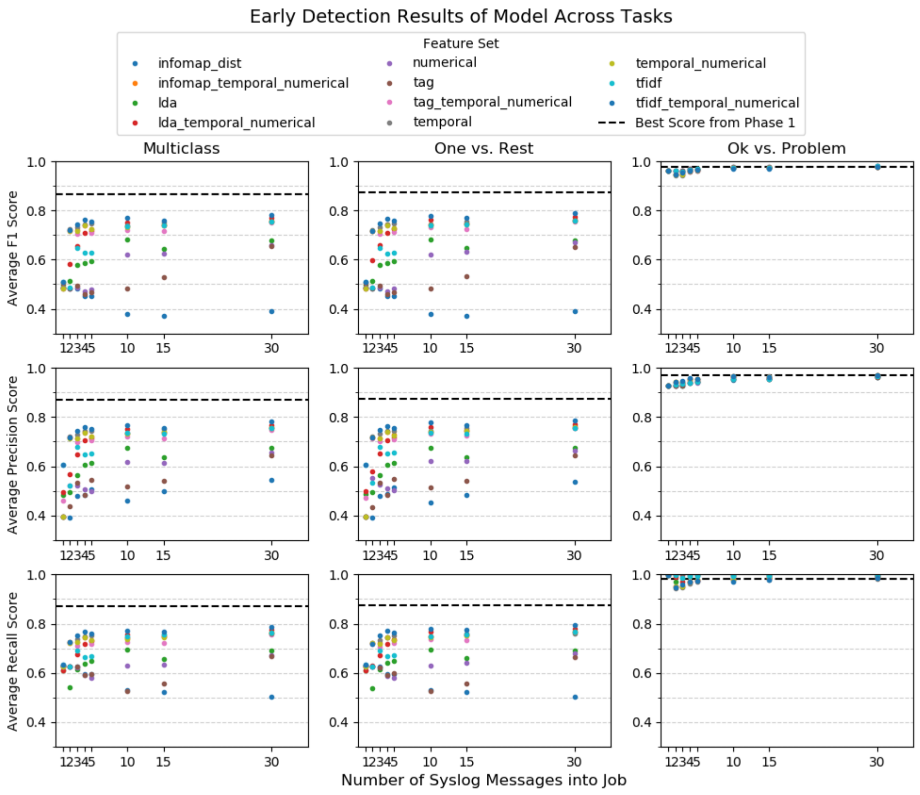
10 FUTURE WORK

The work presented in this paper is meant to be a stepping stone in the path for a real-time log analysis tool for job monitoring. In order to further the development of this tool, there is more work that must be done. In Phase 2 we trained and tested our model on the limited syslogs that met the timing or message number requirement, but next we would like to use our models that are trained on the full syslog set to predict the job outcome with the limited syslogs. Also we would like to find the point at which the models can match the baseline performance of using the full syslog set. This data can help provide system administrators to better set tool parameters for job monitoring. Further, since one of our goals is to create a general tool that is not system-specific, we would like to test our models on logs from another computer.

11 ACKNOWLEDGMENTS

This research is the culmination of a project that began during my summer at LANL as a student in the SULI program. I was lucky enough to be given the opportunity to continue my project as my senior thesis. I would like to thank my mentor, Lissa Baseman, for teaching me over the summer, and my thesis advisor, Dr. Julie Carrington, for letting me continue this project. I would also like to thank my thesis committee members, Dr. Daniel Myers and Dr. Mark Anderson. Lastly, I thank everyone mentioned for their time and support.

Figure 14: All of the models' performance on predicting the job state with a limited number of syslog messages from the job.



REFERENCES

[1] E. Baseman, S. Blanchard, Z. Li, and S. 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, Dec 2016.

[2] C. Bertero, M. Roy, C. Sauvanaud, and G. Tredan. Experience report: Log mining using natural language processing and application to anomaly detection. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 351–360, Oct 2017.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[4] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[5] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[6] A. Oliner and J. Stearley. What supercomputers say: A study of five system logs. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pages 575–584, June 2007.

[7] 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.

[8] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Science*, 105:1118–1123, Jan. 2008.

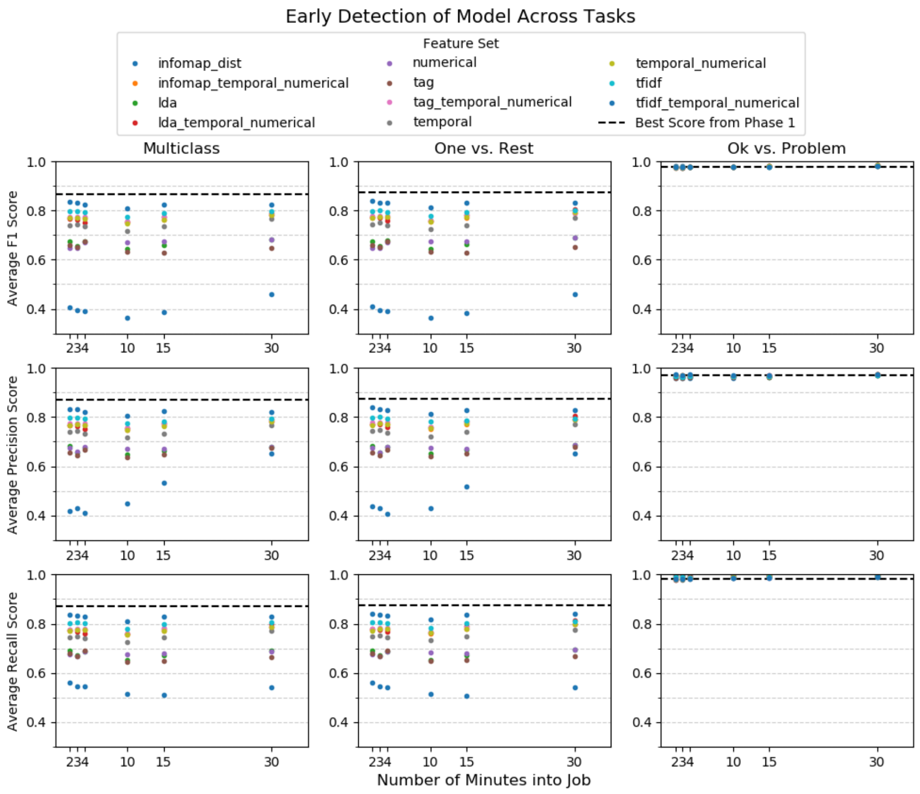
[9] A. Sirbu and O. Babaoglu. A holistic approach to log data analysis in high-performance computing systems: The case of ibm blue gene/q. In *Euro-Par 2015: parallel Processing Workshops, LNCS 9523*. Springer, Springer, 2015.

[10] J. Stearley. Towards informatic analysis of syslogs. In *2004 IEEE International Conference on Cluster Computing (IEEE Cat. No.04EX935)*, pages 309–318, Sept 2004.

[11] R. Vaarandi and M. Pihelgas. Logcluster - a data clustering and pattern mining algorithm for event logs. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 1–7, Nov 2015.

[12] D.-Q. Zou, H. Qin, and H. Jin. Uilog: Improving log-based fault diagnosis by log analysis. *Journal of Computer Science and Technology*, 31(5):1038–1052, Sep 2016.

Figure 15: All of the models' performance on predicting the job state using only the syslog messages a specific number of minutes into the job.


Early Detection of Model Across Tasks
Feature Set
infomap_dist
infomap_temporal_numerical
lda
lda_temporal_numerical
numerical
tag
tag_temporal_numerical
temporal
temporal_numerical
tfidf
tfidf_temporal_numerical
Best Score from Phase 1
Multiclass
One vs. Rest
Ok vs. Problem
Average F1 Score
Average Precision Score
Average Recall Score
Number of Minutes into Job