Work in Progress: Topic Modeling for HPC Job State Prediction

Alexandra DeLucia*
Department of Math & Computer Science
Rollins College
Winter Park, Florida
aadelucia@gmail.com

Elisabeth Baseman†
Ultrascale Systems Research Center
Los Alamos National Laboratory
Los Alamos, New Mexico
lissa@lanl.gov

ABSTRACT

As high performance computing approaches the exascale era, progress in automatic computer monitoring becomes increasingly important. Monitoring will no longer to able to rely only on human experts, due to the overwhelming amount of monitoring data, such as system logs, job logs, and temperature reports. Because a human analyst cannot keep up with terabytes of monitoring data per day, we turn to techniques from the statistical machine learning community to assist with analysis of monitoring data. Specifically, we use machine learning techniques predict compute job outcomes using features extracted from system log messages. Our preliminary results show that not only do statistical topics extracted from log messages provide a signal correlated with job outcome, but that the correlation is strong enough that two canonical classification algorithms can achieve very high predictive performance using only topic distributions and basic temporal information as features.

CCS CONCEPTS

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

KEYWORDS

topic modeling, log analysis, job prediction

ACM Reference Format:
Alexandra DeLucia and Elisabeth Baseman. 2018. Work in Progress: Topic Modeling for HPC Job State Prediction. In *MLCS'18: First Workshop on Machine Learning for Computing Systems, June 12, 2018, Tempe, AZ, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3217871.3217874

*This work was completed during a summer internship at Los Alamos National Laboratory, sponsored by the Science Undergraduate Laboratory Internships program.
†This work was performed 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-18-23441.

MLCS'18, June 12, 2018, Tempe, AZ, USA

ACM ISBN 978-1-4503-5865-1/18/06…$15.00
https://doi.org/10.1145/3217871.3217874

1 INTRODUCTION

In the exascale computing era, high performance computing machines and facilities will grow to approximately 25 times their current size. This will result in the amount of monitoring data produced growing at a similar rate, while human operators are already struggling to efficiently manage HPC facilities at their current sizes. Due to the massive amount of monitoring data already being produced (at a rate of terabytes per day on the largest machines). alternatives to purely human operation must be explored. For these reasons, we turn to statistical techniques from the machine learning community that have already proven effective in other domains.

In this work-in-progress report, we investigate the question of monitoring the textual system logs produced by every compute node in an HPC cluster. We aim to correlate these system logs with job logs from the scheduler in order to use system logs to predict job outcomes. The ultimate goal is to develop a tool that will raise alerts in real time for compute jobs which are likely to experience a failure or timeout. Although this work is ongoing, our work presented here makes the following contribution:

- Discovery that topic modeling and machine learning can predict job outcome based on system log messages

2 RELATED WORK

While some work exists on the use of statistical techniques for mining system log messages in order to evaluate/monitor system performance, there has been little work on mining system log messages specifically for predicting job outcomes. In general, previous work on statistical analysis of system log messages focuses on anomaly detection [2, 10–13], while our goal is job outcome prediction.

The current landscape of system log analysis has two main cohorts: a systems-oriented group, and a data science-oriented group. The most common technique for log message analysis in the systems community is to create regular expressions for messages, and then use these patterns to assign syslog lines into groups. The pattern definitions are usually hand-coded, but some programs exist to generate patterns automatically [8]. The data science approach views the logs as a rich inhomogeneous combination of text, numerical, and temporal data. Specific methods in this space include graph analysis and natural language processing [1, 3, 9].

While we draw from previous work in both the systems and data science communities, particularly in the area of natural language processing, our work distinguishes itself in that we wish to accomplish a clear predictive task rather than detect anomalies. For this reason, we are able to not only develop a predictive technique, but

also report on its usefulness (as opposed to the anomaly detection task where quantitative evaluation is difficult).

3 AVAILABLE DATA

For this initial study, we use monitoring data collected from Wolf, a high performance computing cluster located at Los Alamos National Laboratory. The Wolf system contains 592 compute nodes with a total of 9,856 Dual 8-core Intel Xeon (Sandy Bridge) processors with Infiniband [6]. Wolf is a 205 TF system that is available for use by scientists at Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Sandia National Laboratories.

3.1 System Logs (Syslogs)

System logs (syslogs) gathered from the Wolf system at Los Alamos are similar to syslogs from any other large machine. These logs contain near-natural language messages that describe a wide variety of activities occurring on the machine.

Textual log message are extremely important to human operators seeking to understand out the overall state of a cluster or a particular compute node, especially in the event of a failure. Although a single compute node may generate a variety of textual logs, the human operator's starting place for root cause analysis is usually the syslog, as it records all processes and system events, such as user logins and BIOS boot sequences. Each syslog message is saved in a text file or database by a monitoring utility and contains a timestamp, node name (unique to a multi-node system such as a supercomputer), a "tag" — i.e., the name of the daemon or other system process that generated the message), and the textual message itself. Figure 1 shows an example syslog message and its components.

```
<Datetime> <Node> <Process Tag> <Message>
Mar 26 03:45:02 wf001 TEMP_SENSORS: coretemp +27.0°C
```

Figure 1: Example syslog line and its high-level components. Red components are features in this preliminary work.

3.2 Job Logs

Job schedulers, such as Slurm, on HPC clusters generate a record for each job that runs. These records are stored in what is known as a "job log." Typically these compute jobs are computationally intensive, such as a large-scale climate simulation or training of a large machine learning model. Each job log record contains detailed information regarding the job itself and the resources it requested, including job start and end times, which particular nodes were allocated, and the state in which the job ultimately finished. The end state of the job in the logs from Wolf can be one of four labels: `COMPLETED`, `FAILED`, `NODE_FAIL`, `CANCELLED`, and `TIMEOUT`. Figure 2 shows an abbreviated job log record format.

```
JobID=# UserID=# GroupID=# Name=<program name>
JobState=[COMPLETED,FAILED, NODE_FAIL,CANCELLED,TIMEOUT]
Partition=<> TimeLimit=# StartTime=<time> EndTime=<time>
NodeList=[] NodeCnt=# ProcCnt=# WorkDir=../../
```

Figure 2: An example abbreviated job log record format from Wolf. Components in red are used in this preliminary work.

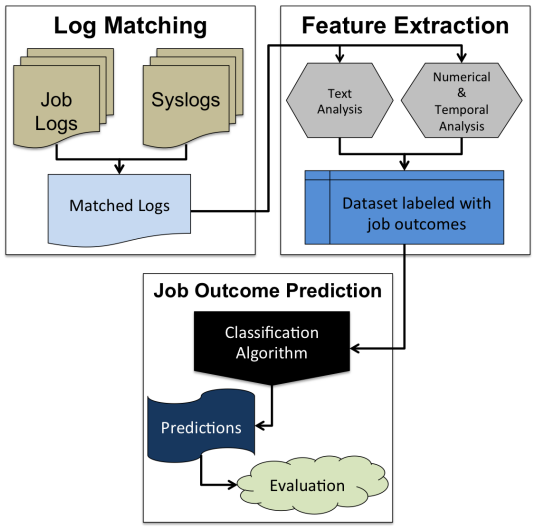

Figure 3: The general workflow for the project. First, syslogs and job logs are correlated based on time and node origin. Second, the grouped syslogs are summarized using numeric descriptions, and then the matched job logs with syslog descriptors are used to build a prediction model.

4 APPROACH

Our approach to job outcome prediction is a 3-stage process: (1) data wrangling to match job log records with the corresponding syslog messages, (2) feature extraction using latent Dirichlet allocation, and (3) prediction. Figure 3 shows an overview of the workflow.

4.1 Matching Job Log Records with Corresponding Syslog Messages

We create an intermediate dataset by parsing through the text files for syslog messages and job logs. For each job log record, we extract all syslog messages from nodes on which the job ran between the start and end time of the job. For example, if job #2 ran on compute nodes 130, 131, and 132 from 9:00AM to 11:00AM. All of the syslog messages from nodes 130, 131, and 132 that occurred between 9:00AM and 11:00AM would be "matched" to job #2. The resulting intermediate labeled dataset becomes the largely unstructured near-natural language data from which we extract features.

4.2 Feature Extraction

With job log records now matched to their corresponding syslog messages and labeled with the final job outcome, we extract two types of features: (1) topic distributions, and (2) temporal features.

4.2.1 Topic Distributions. Topic modeling is a common technique in the natural language processing field to summarize a corpus of documents (in this case, each group of syslog messages corresponding to a single job constitutes a "document"). In the natural language processing community, a "topic" is a vector of words that serve to statistically summarize a document. There are a variety of statistical techniques for automatically extracting these topics, and we make the common choice of using latent Dirichlet

Topic	Keywords
1	lustre mdt mdc connection obdclass init fault lock page task ptlrpc will lost using alloc progress mutex service operations handle
2	session user unix pam sshd root uid opened closed access map found key
3	iterations msgsize ranks pattern jobid wolf cluster sequential avgbw tonode fromnode send read usec avglat test write session aggregate ring
4	session user unix pam root sshd coretemp isa physical opened uid closed inet server connection log time while idle reopen
5	ost lustre session connection osc previous similar skipped pam unix root user sshd messages restored physical isa coretemp closed opened

Figure 4: Topic vectors learned using 5 topics.

Topic	Keywords
1	session user pam unix sshd uid closed opened access granted root
2	iterations msgsize ranks pattern jobid wolf cluster sequential avgbw tonode fromnode send read usec avglat test write session aggregate ring
3	write sys file error channel socket addr cpu memory not page open task obdclass time own read check generic area
4	session sshd root pam unix user isa coretemp physical closed opened uid
5	session lustre mdt user root mdc sshd connection pam unix init fault obdclass coretemp physical isa uid will lost operations
6	session unix user pam root sshd opened uid closed
7	session pam sshd user unix root physical coretemp isa lnet connection lustreerror import closed inet opened uid own server routers
8	ost lustre osc connection previous similar skipped messages restored progress will service operations using lost recovery complete message kiblnd iblnd

Figure 5: Topic vectors learned using 8 topics.

allocation (LDA) [4], as implemented in MALLET [7]. Because the number of topics in LDA is a user-defined parameter, we generate compare topic distributions using 5, 8, 10, 15, and 20 topics.

In order to confirm that the topics learned will be useful, we investigate the topic distributions across each job outcome type. If the topic distributions appear to differ based on job outcome, then we assume that there is useful predictive signal in the learned topics, and proceed with our study. Figures 6 and 7 shows the distribution of topics by job outcome with 5 and 8 topics, respectively. In both cases, we note that the distributions for `FAILED`, `NODE_FAIL`, and `TIMEOUT`, appear to differ significant from each other and from `CANCELLED` and `COMPLETED`. Distributions for `CANCELLED` and `COMPLETED` appear similar, but this is to be expected because a canceled job would generally appear to be progressing correctly until killed by a user. For specific topic vectors learned, see Figures 4 and 5, which show actual topics learned for 5 and 8 topics, respectively.

4.2.2 Temporal Features. In addition to extracting statistical topics from the syslog message "documents," we also extract basic temporal information using the message timestamps. Specifically, we include as features the total time duration of the job as well as the average time between syslog messages during the job.

5 EXPERIMENTAL SETUP

We examine the usefulness of our extracted features for predicting job outcome, using two discriminative statistical machine learning models: (1) a random forest, and (2) a support vector machine (SVM), both as implemented in the scikit-learn Python package [5]. All random forest experiments were run with default settings, while SVM experiments used a radial basis function kernel and a gamma of 1. The statistical topic modeling was completed through latent Dirichlet allocation as previously described, using 5, 8, 10, 15, and 20 topics, with an interval optimization of 10 and 3,000

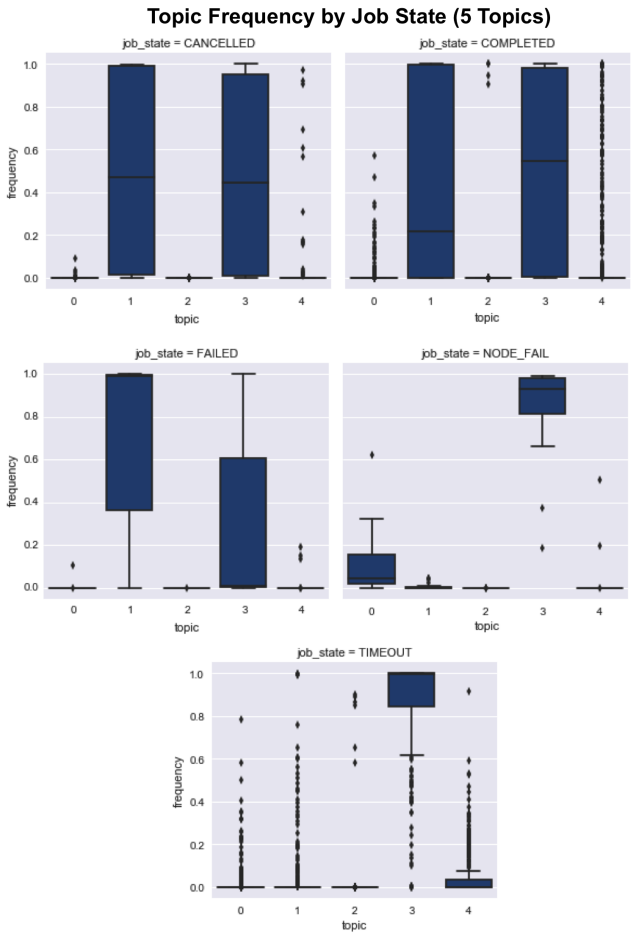

Figure 6: Topic frequency within syslog messages by job state for 5 topics. While `COMPLETED` and `CANCELLED` appear similar, the other outcomes do not.

iterations (which, by observation, was sufficient for the model to converge). For each machine learning model / topic number pairing, we evaluate the predictive performance of the model, while also giving the model access to the basic temporal features discussed earlier. We evaluate each of our model / topic number pairings using weighted multiclass Area Under the Receiver Operating Characteristic Curve (AUC). In general, an AUC of 1.0 indicates perfect predictive performance, and 0.5 indicates random guessing.

6 PRELIMINARY RESULTS

Figure 8 shows our preliminary results. We find that across all topic numbers, random forests outperformed SVMs. While there does seem to be some variation in performance based on number of topics, that variation appears to be slight. Using 8 topics may give slightly better performance, but these preliminary results are inconclusive. In general, these results are encouraging because they indicate that there is indeed predictive signal within syslog messages regarding job outcomes, and that this signal may be somewhat invariant to specific feature extraction choices.

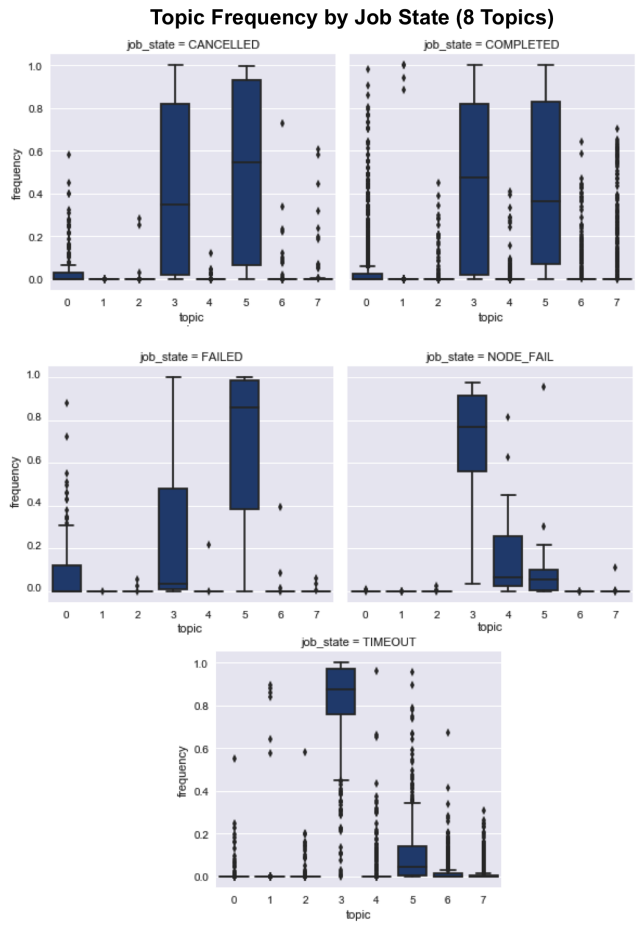

Figure 7: Topic frequency within syslog messages by job state for 8 topics. While `COMPLETED` and `CANCELLED` appear similar, the other outcomes do not.

7 CONCLUSION AND FUTURE WORK

We find that the use of relatively simple natural language processing techniques on near-natural language system log messages is indeed useful for prediction of job outcomes. In fact, using only statistically-extracted topics and basic temporal information, random forests are able to achieve an average AUC of approximately 0.87 for predicting job outcome. While significantly more work is required in this area, our preliminary results are very promising, and may point to the ability to eventually develop tools that give early warning alerts for jobs that may have an unfavorable outcome. Next steps include other strategies for feature extraction in an attempt to improve predictive performance, and then investigation into early detection.

ACKNOWLEDGEMENTS

The authors thank Nathan DeBardeleben and Sean Blanchard, both at the Ultrascale Systems Research Center and Los Alamos National Laboratory, for domain expert input and assistance obtaining data.

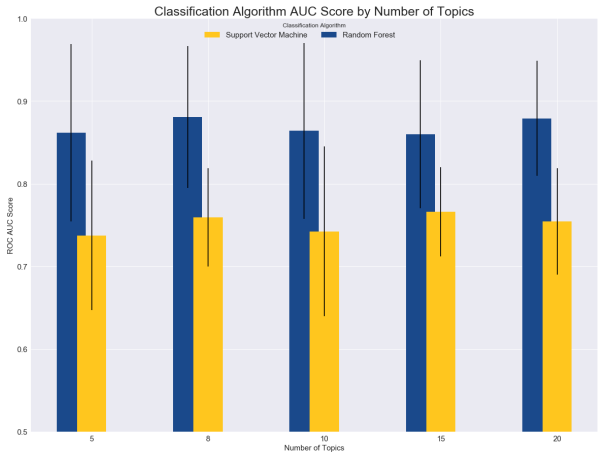

Figure 8: AUC with standard deviation for each model / topic number pairing. Random forests outperformed the SVM across all trials. Note that while some variations in AUC exist, the overall values appears relatively stable across number of topics.

REFERENCES

[1] Elisabeth Baseman, Sean Blanchard, Nathan DeBardeleben, Amanda Bonnie, and Adam Morrow. 2016. Interpretable Anomaly Detection for Monitoring of High Performance Computing Systems. *22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining - Outlier Definition, Detection, and Description On-Demand Workshop* (2016).

[2] E. Baseman, S. Blanchard, Z. Li, and S. Fu. 2016. Relational Synthesis of Text and Numeric Data for Anomaly Detection on Computing System Logs. *International Conference on Machine Learning Applications* (2016).

[3] C. Bertero, M. Roy, C. Sauvanaud, and G. Tredan. 2017. Experience Report: Log Mining Using Natural Language Processing and Application to Anomaly Detection. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*. 351–360. https://doi.org/10.1109/ISSRE.2017.43

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022. http://dl.acm.org/citation.cfm?id=944919.944937

[5] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[6] Los Alamos National Laboratory. 2018. Tri-Lab Computing Resources. (2018). http://www.lanl.gov/asc/tri-lab-resources.php

[7] Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. (2002). http://mallet.cs.umass.edu.

[8] A. Oliner and J. Stearley. 2007. What Supercomputers Say: A Study of Five System Logs. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. 575–584. https://doi.org/10.1109/DSN.2007.103

[9] R. Vaarandi and M. Pihelgas. 2015. LogCluster - A data clustering and pattern mining algorithm for event logs. In *2015 11th International Conference on Network and Service Management (CNSM)*. 1–7. https://doi.org/10.1109/CNSM.2015.7367331

[10] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan. 2009. Online system problem detection by mining patterns of console logs. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 588–597.

[11] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. 2009. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 117–132.

[12] Wei Xu, Ling Huang, Armando Fox, David A Patterson, and Michael I Jordan. 2008. Mining Console Logs for Large-Scale System Problem Detection. *SysML* 8 (2008), 4–4.

[13] Wei Xu, Ling Huang, and Michael I Jordan. 2010. Experience Mining Google's Production Console Logs.. In *SLAML*.