High Performance Computing Job Outcome Prediction By Mining System Logs

Alexandra DeLucia
Department of Math & Computer Science
Rollins College

Elisabeth Baseman
Ultrascale Systems Research Center
Los Alamos National Laboratory

Introduction

As the high performance computing (HPC) community anticipates the exascale computing era, automated computer maintenance and error prevention becomes increasingly important. In this work we apply machine learning techniques to automate compute job monitoring. We draw from system logs to extract features and use these feature sets to create models for predicting job state. We evaluate the feature set viability and the models on a variety of classification tasks.

Background

High Performance Computing (HPC)

A high performance computer is a powerful computer consisting of many compute cores called "**nodes**". HPC machines and facilities provide massive amounts of monitoring data regarding system health.

Trinity, a supercomputer at LANL, with almost 20,000 compute nodes and 78PB of capacity.

System Logs (Syslogs)

A **syslog** is a text file of recorded events from a computer. The syslog from each compute node is combined into a single file. Syslogs give insight to process activities and are crucial for failure analysis.

```
<Datetime> <Node> <Process Tag> <Message>

Mar 26 03:45:02 wf001 TEMP_SENSORS: coretemp +27.0°C
```

The syslog line format and an example syslog line corresponding to a core temperature check

Job Logs

A **job** is an allocation of resources to a user for a specified amount of time. Jobs are recorded by the job scheduler (e.g. Moab, Slurm) in a **job log file**. Each entry in the log file contains information related to the job, such as the user, the start and end time, the nodes that the job ran on, and the **job state**.

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

Job log entry format. The highlighted fields are used on our study.

The job state indicates normal or problematic outcomes, and is one of five options: COMPLETED, FAILED, NODE_FAIL, CANCELLED, and TIMEOUT.

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

Research Questions

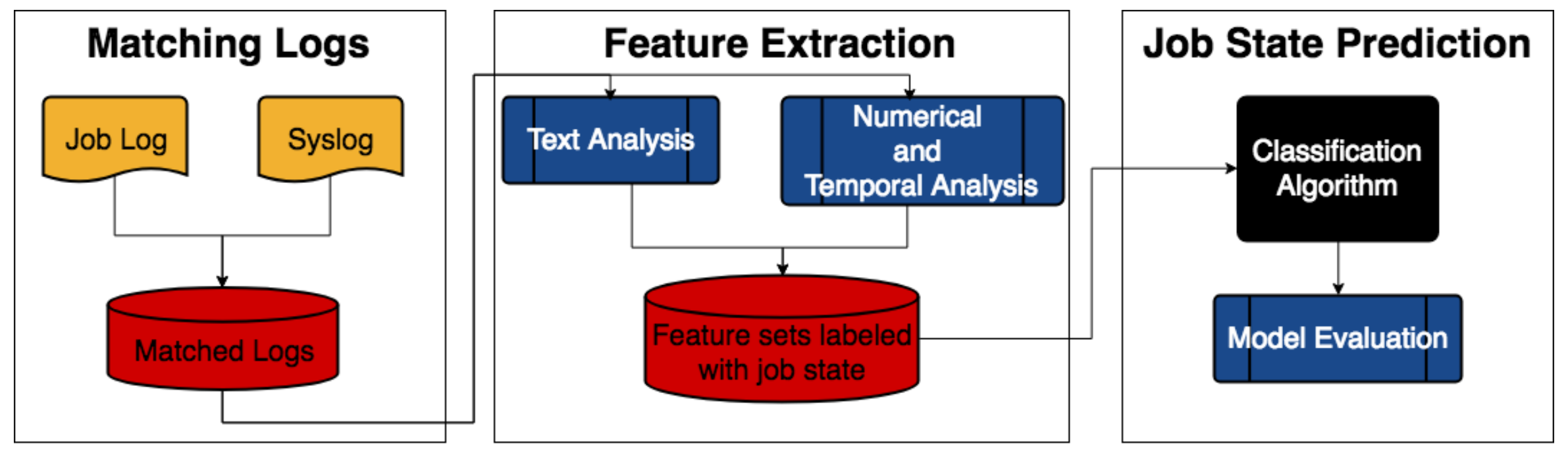
1. How accurately can syslogs predict job outcome?
2. Which features from syslogs are most informative?

Approach

Overview

Our study consists of three phases:

1) Matching syslogs to corresponding jobs,
2) extracting numerical and temporal features, and
3) extracting text-based features.



Matching Syslog and Job Log

Using node and time information from the job log, we match each job with corresponding syslog messages across all relevant compute nodes.

Job ID	Job State	Start Time	End Time	Nodes	Syslogs
1	CANCELLED	2017-03-13 T22:44:40	2017-03-13 T24:44:40	[wf040, wf050]	[msg1, msg2,…]
2	FAILED	2017-03-14 T12:25:40	2017-03-15 T19:30:50	[wf040]	[msg1]
3	TIMEOUT	2017-03-16 T04:20:30	2017-03-16 T05:00:16	[wf150, wf153]	[msg1, msg2, …]

After matching, we extract features from grouped syslogs. Syslog messages are an inhomogeneous combination of text, numerical, and temporal data. Therefore, we separate and analyze each aspect individually from a zero-resource perspective.

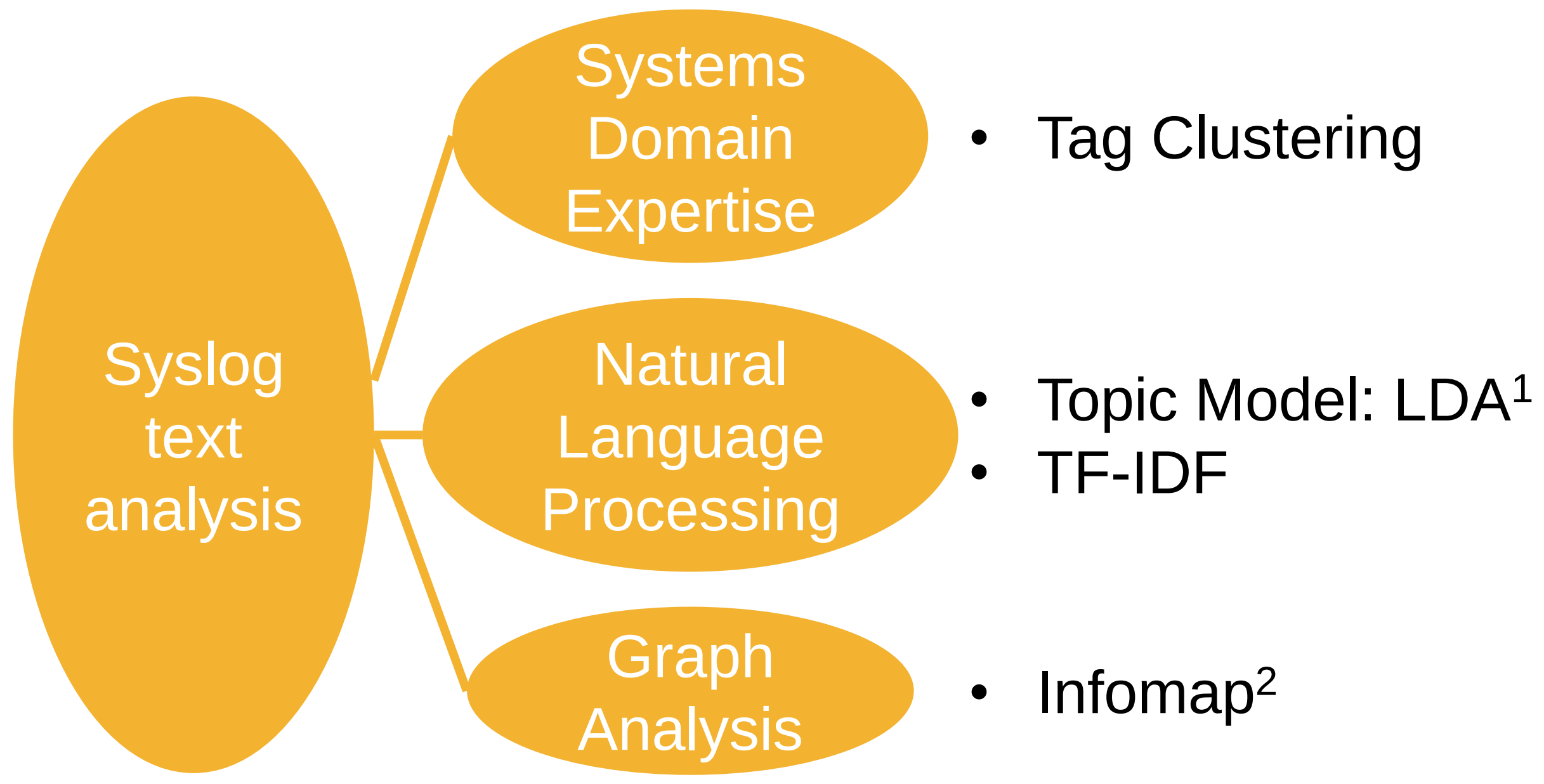
Feature Extraction | Numerical and Temporal

We use the following from each syslog group:

Numerical	Temporal
• Average of numbers • Standard deviation of numbers • Count of numbers	• Average time between messages • Standard deviation of time between messages • Total time between first and last message

Feature Extraction | Text

We use the following context-extraction techniques:



- Tag Clustering
- Topic Model: LDA[1]
- TF-IDF
- Infomap[2]

Experimental Setup

Job Outcome Prediction

After extracting features, we train and test a random forest model using a variety of feature combinations:

1) Numerical only
2) Temporal only
3) LDA distribution only
4) LDA distribution + numerical & temporal
5) Infomap distribution only
6) Infomap distribution + numerical & temporal
7) Tag distribution only
8) Tag distribution + numerical & temporal
9) TF-IDF only
10)TF-IDF + numerical & temporal

Evaluation

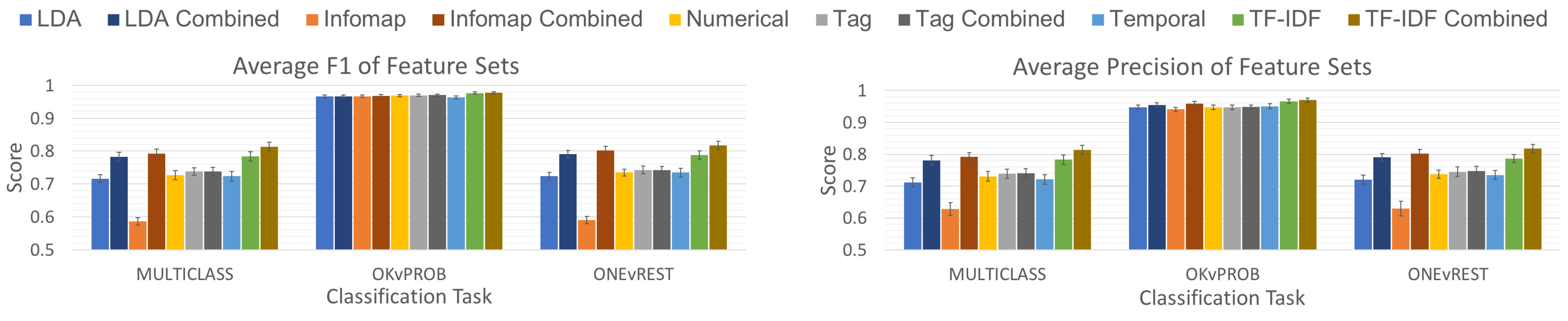
For each feature combination, we evaluate the model on three tasks: predicting each class, predicting ok/problem, and one vs. rest. We use precision, recall, and F1 score as our evaluation metrics.

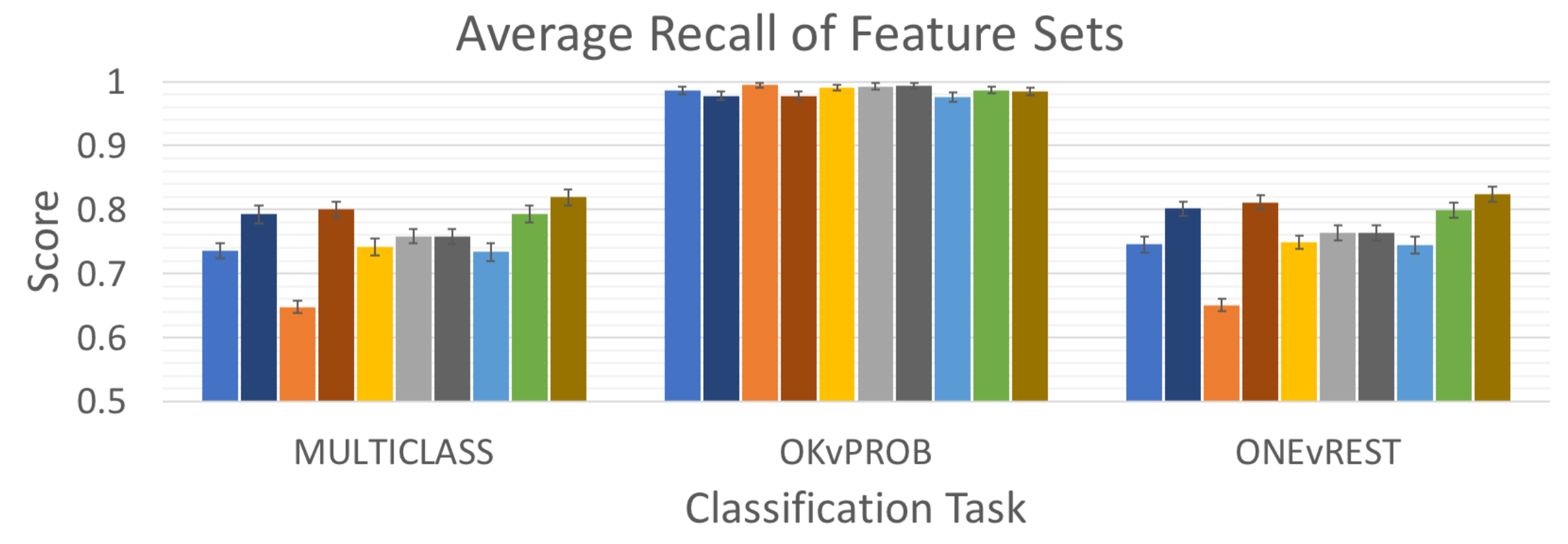
Summary

- The vast information produced by high performance computers are outpacing human analysts' ability to monitor these systems unassisted
- We focus on working towards the automation of using system logs (syslogs) to monitor compute jobs for potential problems
- Syslogs were matched with their corresponding jobs and analyzed for text, numerical, and temporal features, which were used to create models to predict the job state (outcome)
- Each feature set combination model was evaluated on multiple classification tasks
- The best performing feature set was TF-IDF cluster distribution with temporal and numerical analysis

Results

We have addressed our research questions of job state prediction accuracy and feature set viability. As seen below, all feature set models performed the best on the ok/problem classification task. Overall, the job state can be predicted with an F1 score of above 95% with the ok/problem classification task and above 70% with the multiclass and one vs. rest classification tasks (excluding the Infomap cluster feature set). The model feature set which consistently performed the best was the term frequency-inverse document frequency (TF-IDF) cluster distribution with the numerical and temporal analyses.





Top Model: TF-IDF with Temporal and Numerical Feature Set			
Model	F1	Precision	Recall
Multiclass	0.813 (0.013)	0.814 (0.013)	0.819 (0.013)
One v Rest	0.977 (0.004)	0.970 (0.006)	0.985 (0.006)
Ok vs Problem	0.817 (0.012)	0.818 (0.013)	0.824 (0.012)

Acknowledgements

We thank Sean Blanchard and Nathan DeBardeleben, both at USRC and LANL, for domain expert input. We also thank Professors Julie Carrington, Daniel Myers, and Mark Anderson at Rollins College for supporting this work as an honors thesis.

References

1. D. Blei, Ng, A., and Jordan, M. Latent Dirichlet Allocation. JMLR, 2003.
2. M. Rosvall, and Bergstrom, C. Maps of Information Flow Reveal Community Structure in Complex Networks. PNAS, 2008.