Log Anomaly Detection

Red Hat

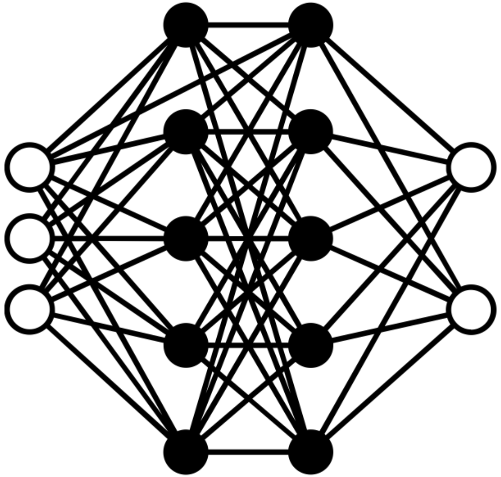
Gideon Sylvester Amoah
Wenren Zhou
Cassie Xie

Agenda

- Big Picture Summary
- Goal
- How it Works
- Tasks and Deliverables
- Key Challenges and Overcoming Them
- Solution Overview
- Demo
- Future Work / Conclusion

Big Picture Summary

Logs are imperative to the maintenance process of most software applications since they record detailed runtime information during system operation. This allows developers and support engineers to monitor and track abnormal behaviors. However, a single software application can generate thousands or millions of log data which a developer will spend countless hours going through when the application is down to conduct root cause analysis.



Created by Product Pencil
from Noun Project

Big Picture Summary

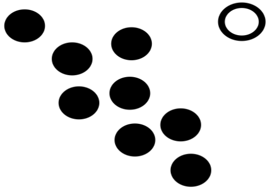
- Productive time spent going through logs

- Billions of Dollars lost

- Boring

Goal

Goal: Real Time Anomaly Detection



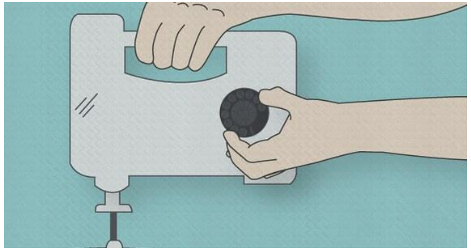
Created by sachin modgekar
from Noun Project

Outcome: Drastically minimize logs that needs to be reviewed

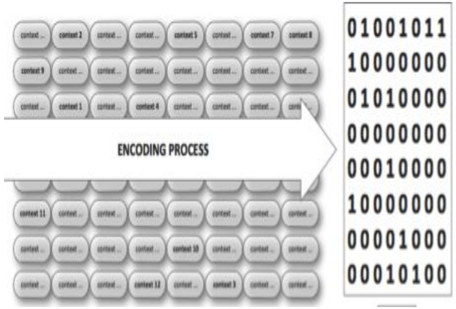
How it Works

How it works

Unlabelled Log Data



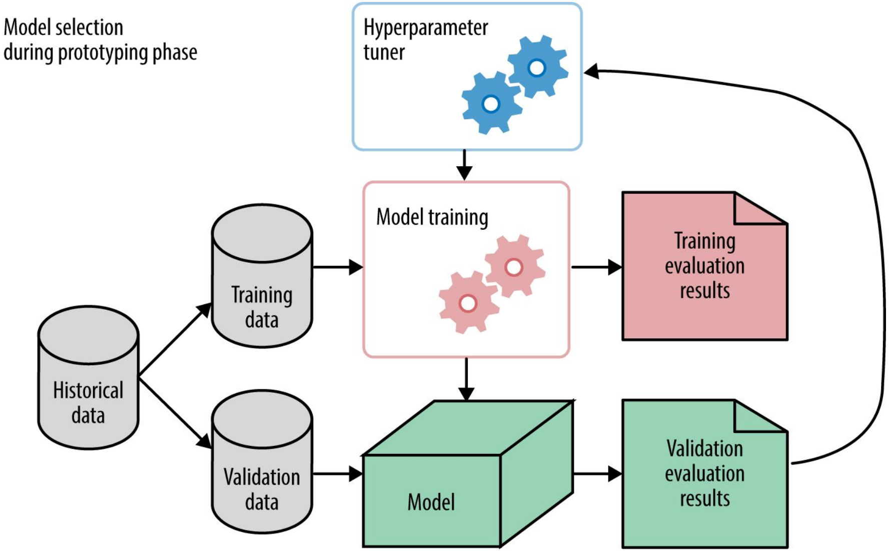
Text Encoding Scheme



Unsupervised Machine learning/ Inference

Tasks and Deliverables

- Each of us implement an alternative model
  - Encoding scheme
  - Learning algorithm
  - Qualification standard
- Validate three models implemented
  - Test on validation dataset
  - Accuracy
- Improve models
  - Tune parameters
  - Use alternative methods



Key Challenges & Overcoming Them

Challenges:

- Gain domain knowledge about the dataset
- Get familiar with open-source coding (git)
- Determine meaningful encoding scheme
- Determine effective learning algorithm
- Quantify anomalousness

How we overcome:

- EDA (exploratory data analysis)
- Tutorials, workshops and practice
- Did research about the pros and cons of each encoding scheme or learning algorithm
- Learn from the current implementation

Solution Overview: Framework and Raw Data

- Framework:
  - Python (scikit-learn and other machine learning library)
  - Jupyter Notebook
  - Github
- Example Raw Data:

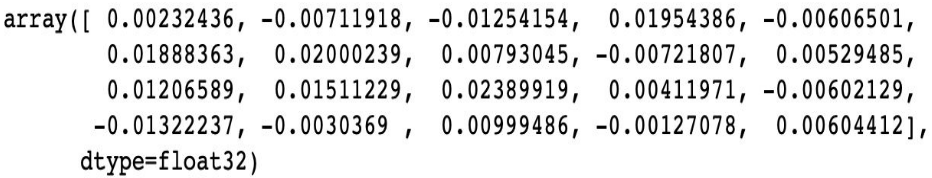
```
2015-10-17 15:37:57,036 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster:
Using mapred newApiCommitter.
```

```
2015-10-17 15:37:57,634 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster:
OutputCommitter set in config null
```

Solution Overview: Text Encoding

- ➢ Text Preprocess
- ➢ Natural language processing
  - ○ Word2Vec
  - ○ Doc2Vec
  - ○ TF-IDF
- ➢ Convert log line of text to vector of real numbers

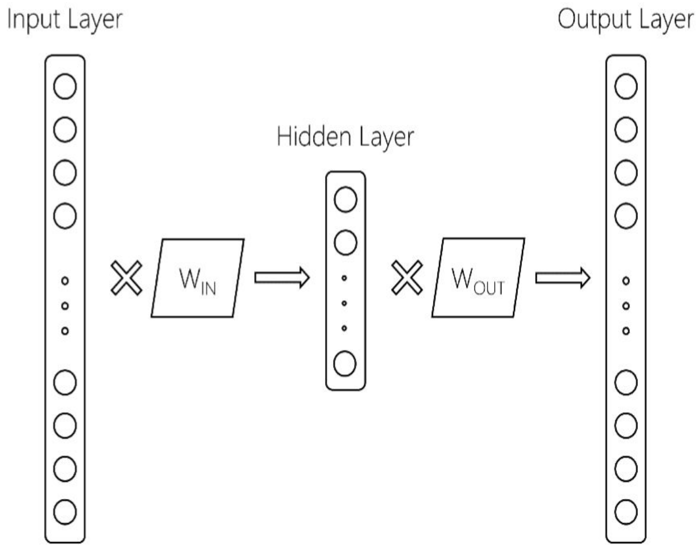
```
2015-10-17 15:37:56,547 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Created MRAppMaster for application appattempt_1445062781478_0011_000001
```



```
array([ 0.00232436, -0.00711918, -0.01254154,  0.01954386, -0.00606501,
        0.01888363,  0.02000239,  0.00793045, -0.00721807,  0.00529485,
        0.01206589,  0.01511229,  0.02389919,  0.00411971, -0.00602129,
       -0.01322237, -0.0030369 ,  0.00999486, -0.00127078,  0.00604412],
      dtype=float32)
```

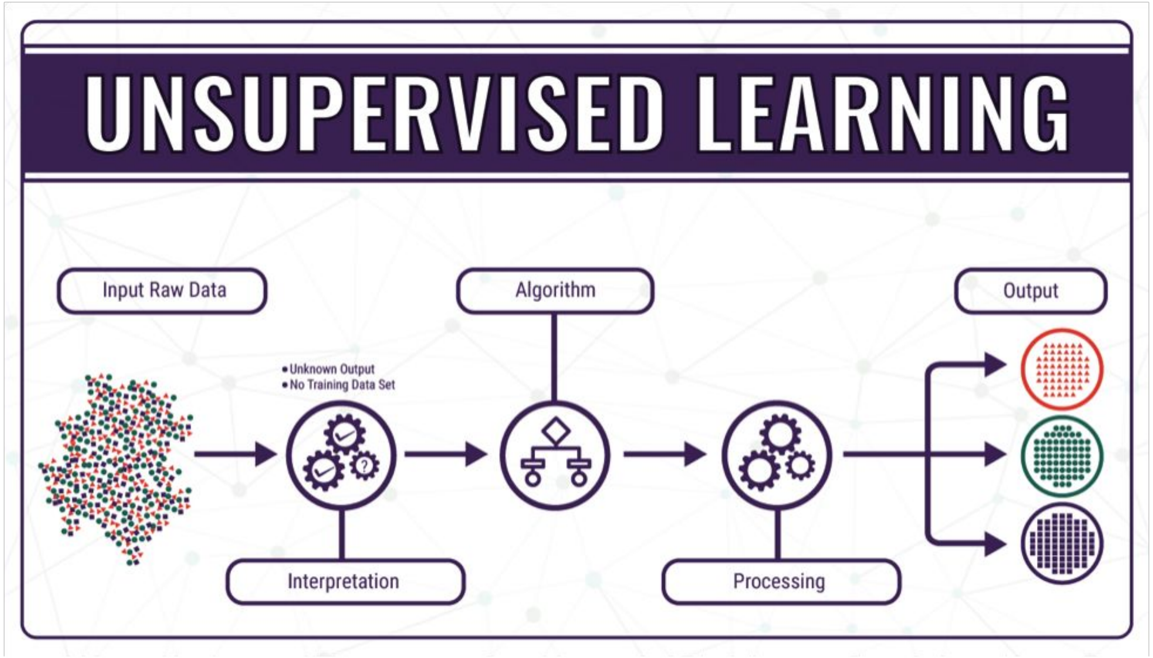
Solution Overview: Text Encoding

- Text Encoding Methods:
  - Word2Vec, Doc2Vec
    - Two layer neural networks
    - Detects similarities in words mathematically
  - TF-IDF (term frequency–inverse document frequency)
    - numerical statistic to reflect how important a word is to a document



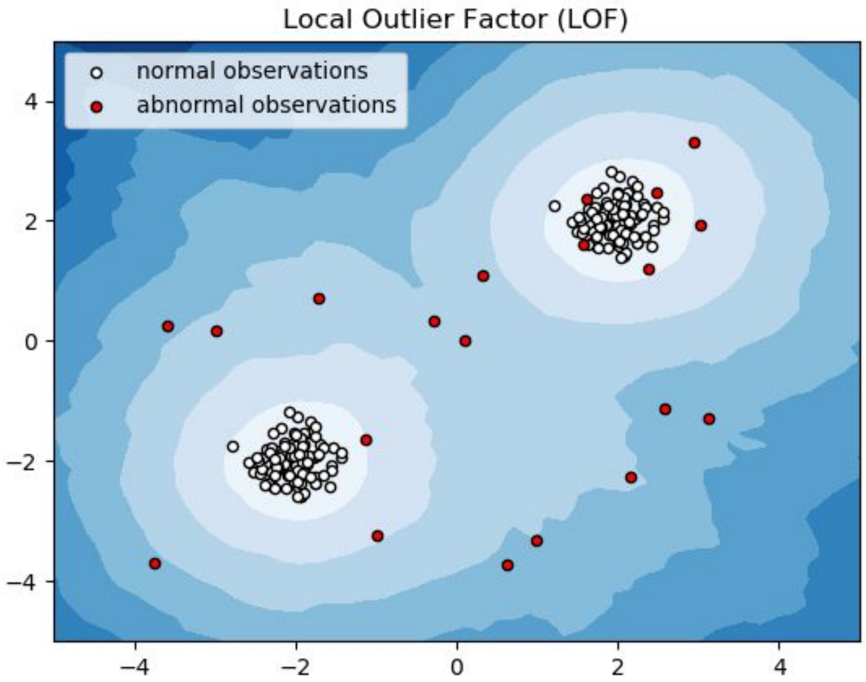
Solution Overview: Log Classification

- Identify anomaly:
  - Classify logs into two groups (normal or abnormal)
- Unsupervised learning:
  - Spectral Clustering
  - K-Means
  - Local outlier factors
- Model Evaluation:
  - Calculate accuracy
  - Test on new data



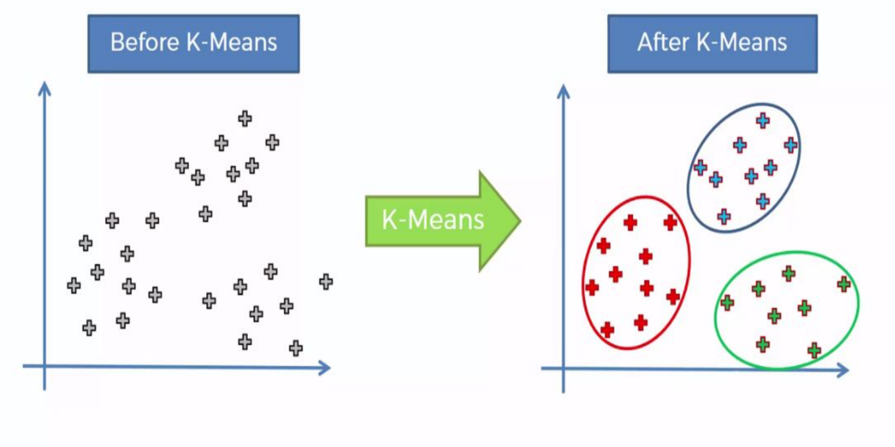
Solution Overview: Local Outlier Factor

- Local outlier factor:
  - Unsupervised outlier detection method
  - Computes local density of a given data point with respect to its neighbors
  - Classifies as outlier if has low density



Solution Overview: K-Means Clustering

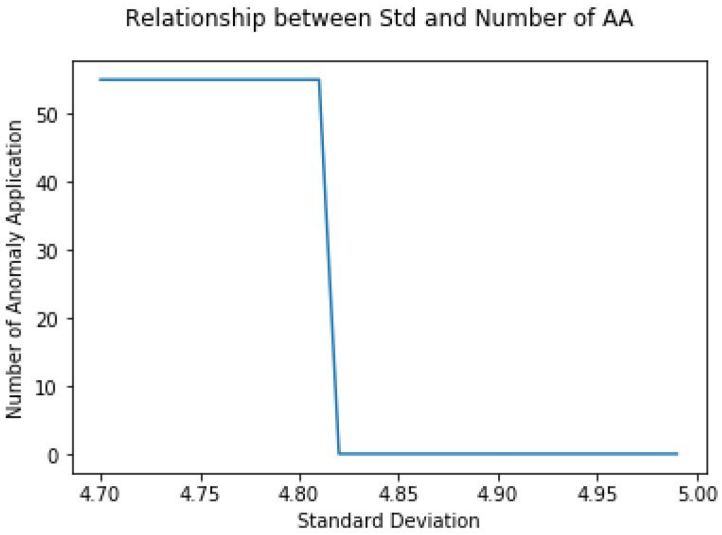
- K-Means Clustering:
  - groups similar clusters together
  - Find centroids, label points based on centroid they are closest to
- Spectral Clustering
  - identify communities of nodes based on the edges connecting them

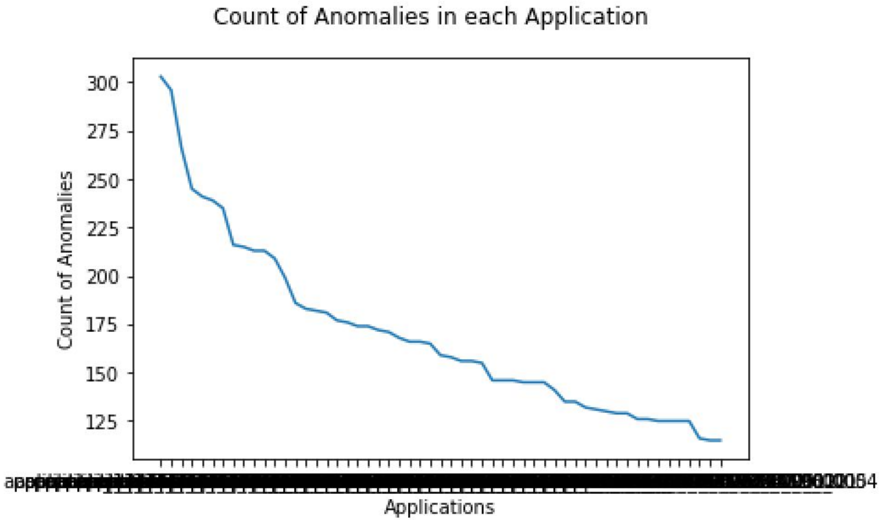


Result Comparison

Model Setup	Anomalies Detected %	Accuracy Rate % (on detected anomalies)
Dataset, Word2Vec, Application, sd = 0.5	34%	80%
Dataset, Word2Vec, Log, sd = 3	43%	95%
Dataset, Tf-idf, Log, sd = 0.1	34%	93%
Single log, Word2Vec, Log, sd = 3		
Single log, Tf-idf, Log, sd = 3		

Visualizations





Future Work and Concluding Summary

- Using natural language processing and unsupervised machine learning algorithms to build a log anomaly detector
  - Using Python, scikit-learn and other machine learning framework, Jupyter Notebook, Github
- Alternative methods could be further explored on encoding and classification

Questions

Thank You!

Special Thanks to Zak and Michael