Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu

Group V: Log Anomaly Detection System - Red Hat

Team

Group V had three students working on the Log Anomaly Detection project with Red Hat: Cassie, Gideon and Mark. Each member had an equivalent role in the team since our client demanded three methods of each implementation. Thus, each team member implemented a different machine learning module.

Background

Companies spend billions of dollars each year to conduct root-cause analysis of a system breakdown, invasion of hackers into their systems and also identify fraudulent activities. Support engineers and developers spend days, weeks and sometimes months going through log data generated by the software systems to conduct the root-cause analysis. Thus, the engineers and developers spend productive time going through log data to find any clue or abnormal log behavior that would point them to the problem. Apart from being expensive and time consuming, we can only imagine how boring it is to manually review thousands and millions of logs.

Key Goals

Transaction logs, logs generated by websites and software applications provide useful information for root-cause analysis of software or system crashes and market analytics. The goal of this project is to implement machine learning modules that would streamline the process of conducting **root-cause analysis** of problems when a software system breaks down. Thus, we hope to provide **real time log anomaly detection** tool that would minimize the number of logs reviewed by support engineers and developers.

Deliverables

Each team member came up with a classification model that successfully classified if a log was an anomaly or not. Cassie implemented Local Outlier Factor, Mark implemented K-Means and Gideon implemented Spectral Clustering and Cosine Similarity. Each team member also tried a new text encoding scheme that is compatible with the corresponding classification method implemented to replace word2vec.

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu

Environment and Tools

The data used in this project is log data, which are text strings. There are multiple datasets of log data that can be used for model training and validation. Data files are in json format and hadoop. We did the modeling with Python and implemented them in our Jupyter notebooks. Various packages in python were used in implementation, including numpy, pandas, gensim, sklearn. Github was used for file sharing and documentation.

Issues and Risks

- The dataset used for testing our models was too small and lacked diversity, and we did not have access to a supportive virtual environment to run bigger datasets.
    - Solution: We tried to make good use of the dataset we had at the moment while talking to our clients about our concerns and needs, and then they provided us with a lot of help based on our requests.

- We didn't have enough domain knowledge about the dataset at the beginning of this project, such as the definitions of application and log, what the structure of the dataset is like and what the high-level meaning of each log information is.

    - Solution: Every time we got a new dataset, we tried to do exploratory data analysis for the dataset using several data preprocessing techniques in python, and built up our knowledge of the definition and meaning of each term and data point gradually.

- We had to get familiar with open-source coding and its useful tools (git). Since none of us had experience with open-source coding before, especially git, we needed to learn its usages, such as how to pull and push on GitHub, etc.
    - Solution: Our clients helped us address this challenge by tutoring us through the process of open-source coding. We also improved our skills by practicing.
- Too many options for the text encoding schemes and learning algorithms, therefore it was difficult to determine which ones are effective and meaningful for our purposes.
    - Solution: We overcame this challenge by actively discussing with our clients, asking for their needs, and their knowledge and previous experience. Since they know this field of study better than us, they gave us a lot of crucial tips on what techniques we could try.
- How to quantify anomalousness?
    - Different methods detect anomalies in different ways, therefore, the way to quantify the anomalousness for each model is also different, and should be creative and stable. We decided the ways to quantify anomalousness

for our models by conducting experiments and then improving and justifying for our methodologies based on the experimental results

Accomplishments

- **Natural Language Processing Methods**

Since the logs are text data, they needed to be processed so they are analyzable. Natural language processing methods were used in this case. Word encoding is a language modeling technique used for mapping words to vectors of real numbers. It represents words or phrases in vector space with several dimensions. Word embeddings can be generated using various methods like neural networks and probabilistic models.

Three different text encoding were used to analyze the text part of the log, they are TF-IDF, Word2Vec, and Doc2Vec. Those two algorithms can convert a log line into a vector of numbers so we can analyze them.

1. *TF-IDF*

Tf-idf gives a score to each unique word in each document, which is positively proportional to the number of times that word appears in that document and offset by the number of documents that contain the word.

TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

This encoding scheme can help highlight words that are common in a document but not in the others, and this kind of words usually helps us identify the topic that document belongs to. Here, we can regard applicants as documents and logs in each application as words in each document, so we can use the Tf-idf encoder to transform the log sentences in each application into feature vectors that reflect the information of the topic of that application, which is the information of abnormality here.

2. **Word2Vec & Doc2Vec**

Word2Vec method consists of models for generating word embedding. These models are two layer neural networks having one input layer, one hidden layer and one output layer. Word2Vec learns feature representation for words.

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu

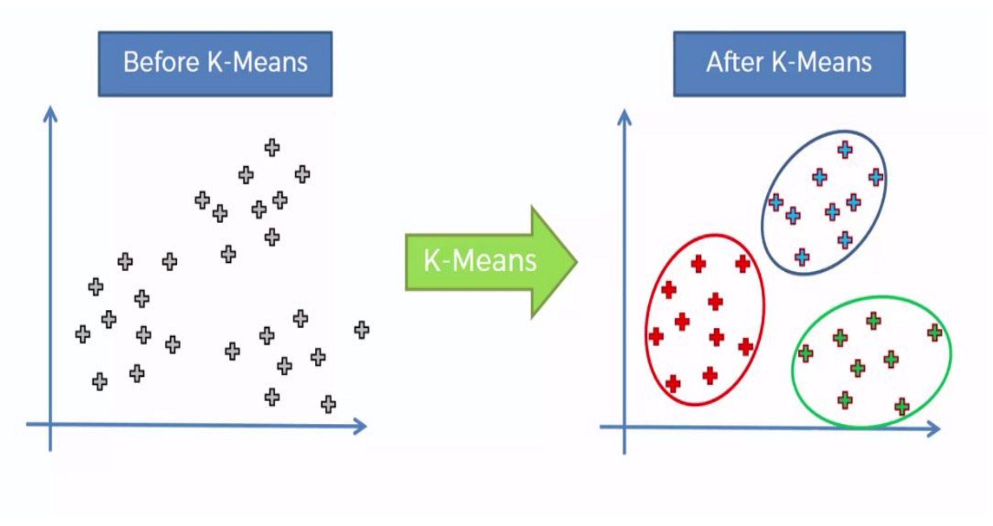
Word2Vec contains two neural network models, they are Continuous Bag of Words model and Skip Gram Model. Continuous Bag of Words model predicts the word given context words, it takes context words as input and output current words. Skip Gram Model predicts the context words given the current words. These methods group similar words together in vector space.
Doc2Vec is an extension of Word2Vec, where instead of learning feature representations for words, it learns it for documents.

- **NLP with Unsupervised Learning for Classification**

1. **Tf-idf with K-means**

The log anomaly detection Mark constructed uses Tf-idf (term frequency-inverse document frequency) as the encoder and K-means as the unsupervised machine learning classifier. How Tf-idf works is that it gives a score to each unique word in each document, which is positively proportional to the number of times that word appears in that document and offset by the number of documents that contain the word.

K-means is one of the popular unsupervised machine learning algorithms. It works by grouping data points in the feature space with certain similarities together as clusters, and each cluster is centered around a centroid. The number of clusters k is specified by users.



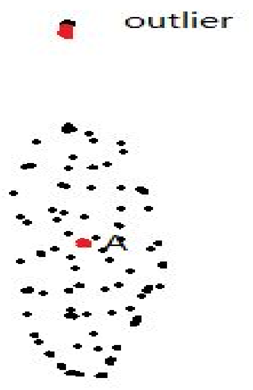
(How K-means work)

After transforming logs in applications into numerical representatives in the feature space, we can possibly differentiate the abnormal logs from normal logs by comparing their distances to their corresponding centroids. When we have only one centroid and we assume that the number of normal logs is far more than that of abnormal ones, then the centroid should be around the middle of normal logs in the feature space, with the abnormal logs located on the outer circle of the cluster. Therefore, we could label the outliers as abnormal and the ones that close to the centroid as normal. After several experiments, we decided to use the (mean distance + 0.1 * standard deviation) as the threshold for examining abnormality, which gave higher anomaly detection accuracy than using other thresholds we had tried.

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

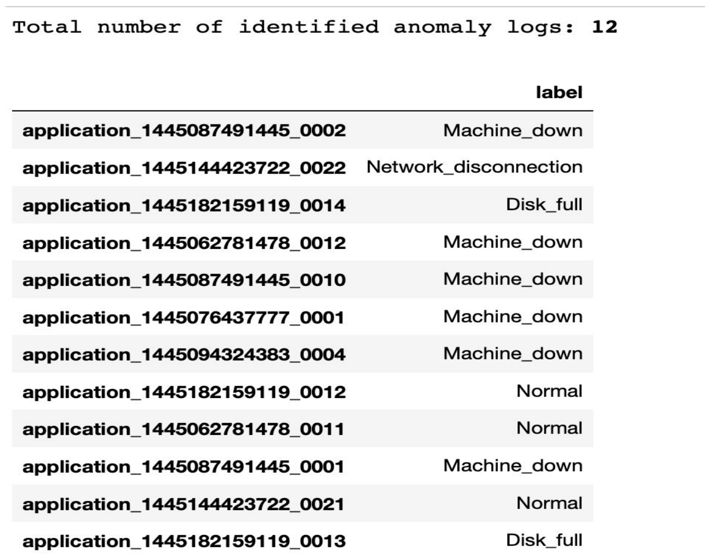
Wenren Zhou
Cornell University
wz366@cornell.edu



(Outlier of a cluster)

Moreover, after we got the list of outliers and labeled them as anomaly logs, how can we tell if an application is abnormal? The way we quantify the relationship between the anomaly logs and anomaly applications is to count the number of anomaly logs in each application, sort the applications in descending order, and select the first one-third part (limitedly proven by experiments to have better classification results) of the sorted application list to be labeled as abnormal.

One more variable we decided to add for this model is how the data points are defined when we classify them. We previously used log as the unit for clustering, however, we can also pass the dataset into the K-means clustering algorithm in the unit of application. In the unit of application, we cannot and do not need to count the anomaly logs in each application, as the data points to be classified are already applications instead of logs. After implemented in python, we validated the implementation of the Tf-idf and K-means model using the Hadoop dataset given by Red Hat, which contains 44 abnormal applications and 11 normal ones as prelabeled by the company. The classification results are shown below:



```
Total number of identified anomaly logs: 12
```

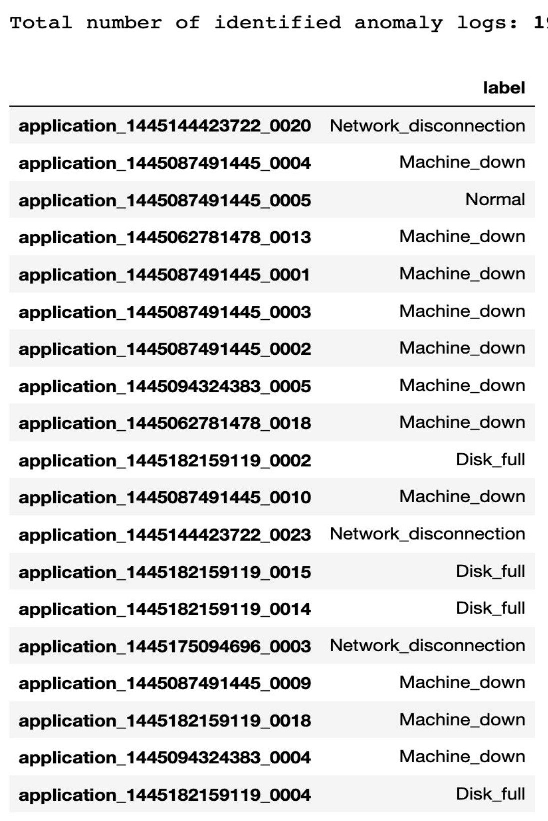
	label
application_1445087491445_0002	Machine_down
application_1445144423722_0022	Network_disconnection
application_1445182159119_0014	Disk_full
application_1445062781478_0012	Machine_down
application_1445087491445_0010	Machine_down
application_1445076437777_0001	Machine_down
application_1445094324383_0004	Machine_down
application_1445182159119_0012	Normal
application_1445062781478_0011	Normal
application_1445087491445_0001	Machine_down
application_1445144423722_0021	Normal
application_1445182159119_0013	Disk_full

(Word2Vec, unit = 'application' , sd = 0.5)

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

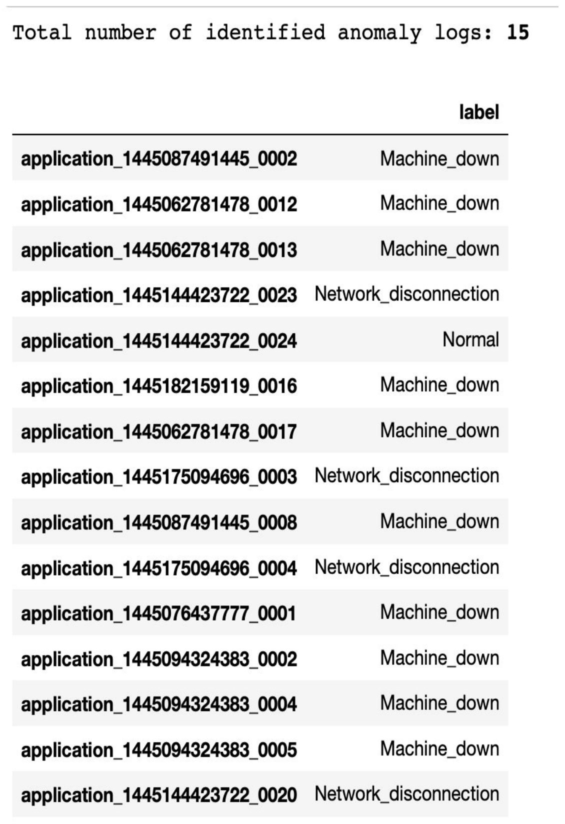
Wenren Zhou
Cornell University
wz366@cornell.edu



```
Total number of identified anomaly logs: 19
```

	label
application_1445144423722_0020	Network_disconnection
application_1445087491445_0004	Machine_down
application_1445087491445_0005	Normal
application_1445062781478_0013	Machine_down
application_1445087491445_0001	Machine_down
application_1445087491445_0003	Machine_down
application_1445087491445_0002	Machine_down
application_1445094324383_0005	Machine_down
application_1445062781478_0018	Machine_down
application_1445182159119_0002	Disk_full
application_1445087491445_0010	Machine_down
application_1445144423722_0023	Network_disconnection
application_1445182159119_0015	Disk_full
application_1445182159119_0014	Disk_full
application_1445175094696_0003	Network_disconnection
application_1445087491445_0009	Machine_down
application_1445182159119_0018	Machine_down
application_1445094324383_0004	Machine_down
application_1445182159119_0004	Disk_full

(Word2Vec, unit = 'log', sd = 3)

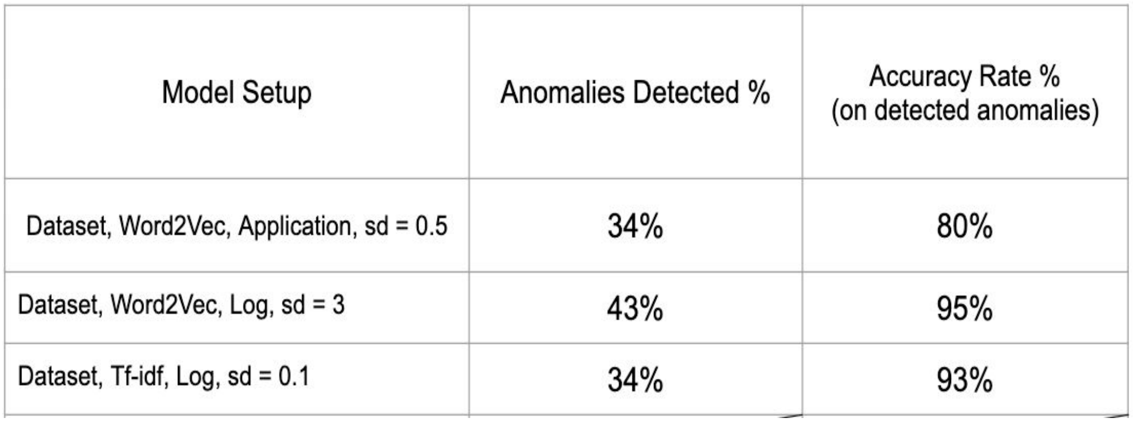


```
Total number of identified anomaly logs: 15
```

	label
application_1445087491445_0002	Machine_down
application_1445062781478_0012	Machine_down
application_1445062781478_0013	Machine_down
application_1445144423722_0023	Network_disconnection
application_1445144423722_0024	Normal
application_1445182159119_0016	Machine_down
application_1445062781478_0017	Machine_down
application_1445175094696_0003	Network_disconnection
application_1445087491445_0008	Machine_down
application_1445175094696_0004	Network_disconnection
application_1445076437777_0001	Machine_down
application_1445094324383_0002	Machine_down
application_1445094324383_0004	Machine_down
application_1445094324383_0005	Machine_down
application_1445144423722_0020	Network_disconnection

(Tf-idf, unit = 'log', sd = 0.1)

Based on the results, we calculated for the model with different specifications their anomaly detection rates and also accuracy rates on their corresponding detected anomalies. As we can see, the performance of passing the dataset into the classification algorithm in the unit of ‘application’ is worse than in the unit of ‘log’; the combination of Word2Vec encoder and K-means performs the best among all the setups, achieving an anomaly detection rate of 43% and an accuracy rate of 95%; the combination of Tf-idf encoder and K-means also performs pretty well, achieving an anomaly detection rate of 34% and an accuracy rate of 93%. Although the performance of Tf-idf is not the best here, we still believe that by tuning the parameters of the Tf-idf encoder and K-means algorithm, there is still space for this model to improve.



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%

(Result Comparison Table)

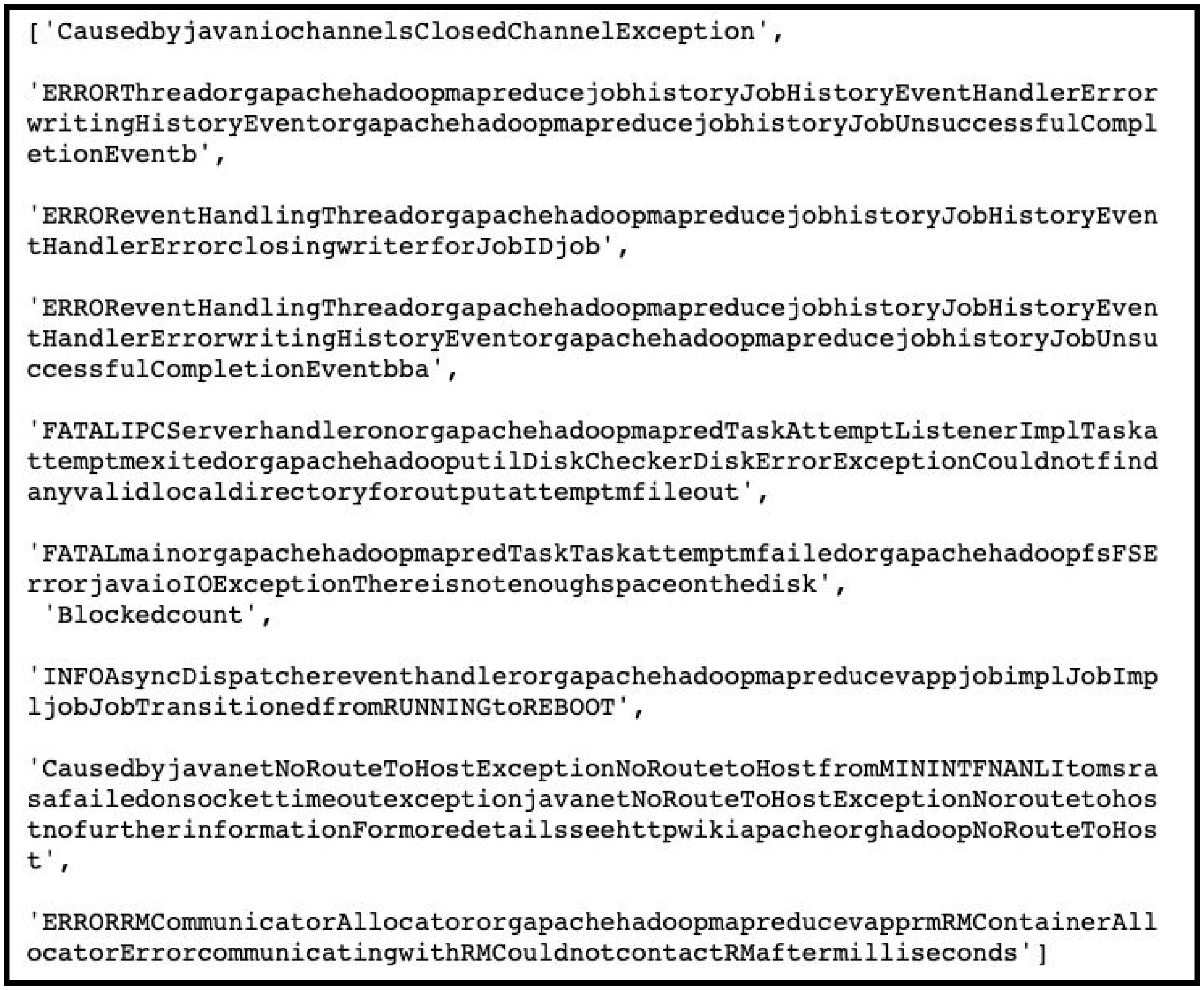
Furthermore, we planned to extract more information from the dataset using the model built so far by furtherly identifying the anomaly logs in addition to the anomaly applications we had detected. The way in which we achieved this goal is to firstly detect anomaly applications using the Tf-idf encoder with the K-means algorithm in the unit of 'log'. Furthermore, we grouped the anomaly applications together, and most of the anomaly logs should be in this new group, but how should we identify them? We came up with the idea to use the idf scores of logs in the anomaly application group to examine whether a log is an anomaly. Previously, we explained how the Tf-idf score for each word (log here in our context) is calculated and what it means. The idf score is half the process of calculating the Tf-idf score. Instead of multiplying the number of occurrences of a word in a document and the logarithm of the number of documents

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu

containing that word, the idf score only calculates the latter part. Therefore, it identifies the logs that are uncommon in most applications, which could be the indication of anomaly logs. The higher the idf score of a log is, the more likely it is an anomaly log. Based on the reasoning, we implemented the functions in python and generated the list of top ten anomaly logs shown below.



```
['CausedbyjavaniochannelsClosedChannelException',

'ERRORThreadorgapachehadoopmapreducejobhistoryJobHistoryEventHandlerError
writingHistoryEventorgapachehadoopmapreducejobhistoryJobUnsuccessfulCompl
etionEventb',

'ERROReventHandlingThreadorgapachehadoopmapreducejobhistoryJobHistoryEven
tHandlerErrorclosingwriterforJobIDjob',

'ERROReventHandlingThreadorgapachehadoopmapreducejobhistoryJobHistoryEven
tHandlerErrorwritingHistoryEventorgapachehadoopmapreducejobhistoryJobUnsu
ccessfulCompletionEventbba',

'FATALIPCServerhandleronorgapachehadoopmapredTaskAttemptListenerImplTaska
ttemptmexitedorgapachehadooputilDiskCheckerDiskErrorExceptionCouldnotfind
anyvalidlocaldirectoryforoutputattemptmfileout',

'FATALmainorgapachehadoopmapredTaskTaskattemptmfailedorgapachehadoopfsFSE
rrorjavaioIOExceptionThereisnotenoughspaceonthedisk',
 'Blockedcount',

'INFOAsyncDispatchereventhandlerorgapachehadoopmapreducevappjobimplJobImp
ljobJobTransitionedfromRUNNINGtoREBOOT',

'CausedbyjavanetNoRouteToHostExceptionNoRoutetoHostfromMININTFNANLItomsra
safailedonsockettimeoutexceptionjavanetNoRouteToHostExceptionNoroutetohos
tnofurtherinformationFormoredetailsseehttpwikiapacheorghadoopNoRouteToHos
t',

'ERRORRMCommunicatorAllocatororgapachehadoopmapreducevapprmRMContainerAll
ocatorErrorcommunicatingwithRMCouldnotcontactRMaftermilliseconds']
```

(Top 10 Anomaly Log List)

As we can see in the list, most logs start with or contain words or phrases such as ‘Error’, ‘Fatal’, ‘Exception’, ‘Could not find’, etc., which confirms that those logs could potentially be anomalies. Since we do not have enough domain knowledge about what each log means, we do not interpret their exact meanings here.

2. Doc2Vec with Local Outlier Factor

Doc2Vec computes a feature vector for every document. The vectors generated by doc2vec can be used for tasks like finding similarity between sentences/paragraphs/documents. For a paragraph Vector framework, every paragraph is mapped to a unique vector, represented by a column in matrix and every word is also mapped to a unique vector, represented by a column in another matrix. The paragraph vector and word vectors are averaged or concatenated to predict the next word in a context.

The input for a Doc2Vec model should be a list of TaggedDocument(sentence, tag 1). A good practice is using the indexes of sentences as the tags. Created tag as the index of each sentence in log. Use this tagged data to train the doc2vec model.The dimension of each sentence was set to 100 features, so the result of each sentence will be a vector of dimension of 100.This number was selected because each sentence is relatively small. According to the analysis, most of the features are not contributing to the result.

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu

Since the dimension of the feature is relatively large to interpret, we cannot visualize it. Principal Component Analysis is used to reduce the dimension of data. The number of PCA components is decided by using the ratio of variance explained by each of the selected components. The first component explained 97% of the variance, the second component explained 2% of the variance, and the third component explained less than 0.1% of the variance. Therefore, the first two components were selected. Same classification result obtained after PCA compared with before PCA.

After preprocessing data with Doc2Vec method, local outlier factor model was used to classify anomaly logs. Local outlier factor is a density-based method that relies on nearest neighbours search. The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors.

Preselected parameter k is the number of neighbors the LOF calculation is considering. The LOF is a calculation that looks at the neighbors of a certain point to find out its density and compare this to the density of other points. With k defined, the k-distance can be calculated, which is the distance of a point to its kth neighbor.

The k-distance is used to calculate the reachability distance. This distance measure is simply the maximum of the distance of two points and the k-distance of the second point.

$$reachability - dist(a, b) = max\{k - distance(b), dist(a, b)\}$$

The reach-dist is then used to calculate still another concept — the local reachability density (lrd). To get the lrd for a point a, we will first calculate the reachability distance of a to all its k nearest neighbors and take the average of that number. The lrd is then simply the inverse of that average.

$$lrd(a) = \frac{1}{sum(reach{-}dist(a,n))/k}$$

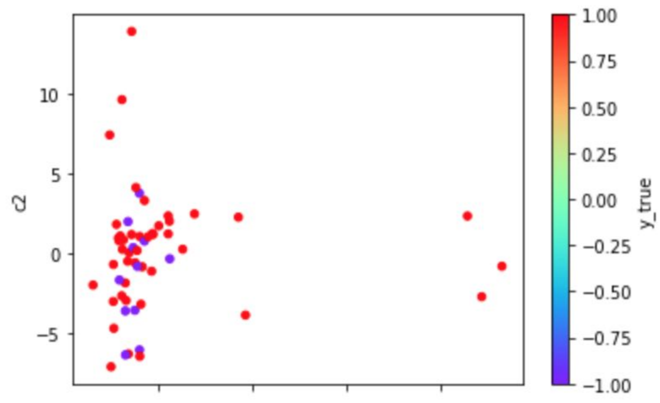
The lrd of each point will then be compared to the lrd of their k neighbors. More specifically, k ratios of the lrd of each point to its neighboring points will be calculated and averaged. The LOF is basically the average ratio of the lrds of the neighbors of a to the lrd of a. If the ratio is greater than 1, the density of point a is on average smaller than the density of its neighbors and, thus, from point a, we have to travel longer distances to get to the next point or cluster of points than from a’s neighbors to their next neighbors.

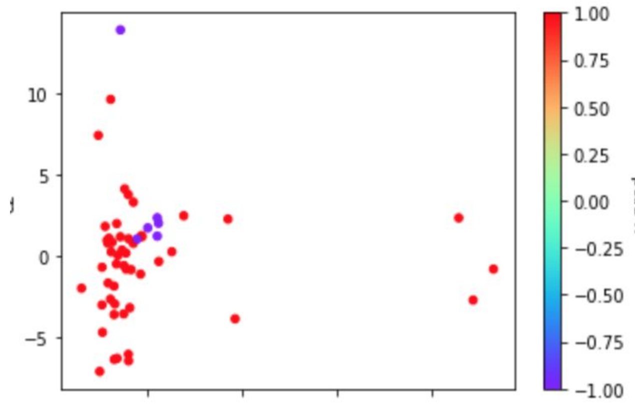
The LOF of a point tells the density of this point compared to the density of its neighbors. If the density of a point is much smaller than the densities of its neighbors (LOF $\gg$1), the point is far from dense areas and, hence, an outlier.

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu





As the analysis results, the above plots are the scatterplots of the final selected two PCA components for local outlier factor. The left picture indicates the true label of the data, the purple is normal data and red are anomalies. The right picture shows the predicted anomalies. The result shows that the algorithm detects 67% of the anomalies in this dataset, but on the other hand, the data set contains more anomalies than normal data, so the dataset itself has some limitations.

3. **Word2Vec with Spectral Clustering**

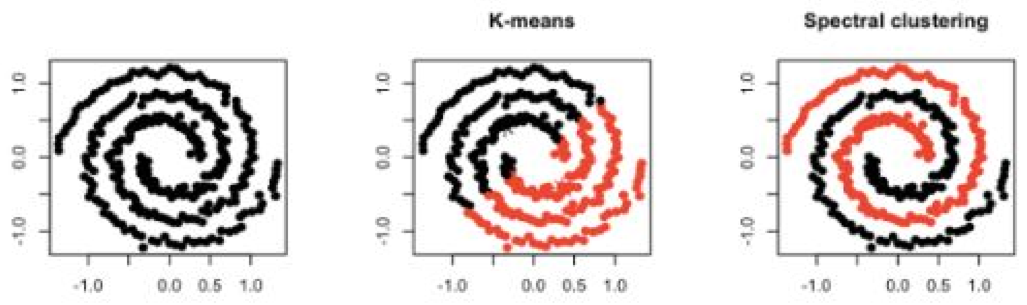
The word2vec module already in place by our client was kept and the vectors generated was classified using spectral clustering. Spectral clustering is a technique with roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them. The method is flexible and allows us to cluster non graph data as well. Spectral clustering uses information from the eigenvalues (spectrum) of special matrices built from the graph or the data set. Compared to the “traditional algorithms” such as k-means or single linkage, spectral clustering has many fundamental advantages. Results obtained by spectral clustering often outperform the traditional approaches, spectral clustering is very simple to implement and can be solved efficiently by standard linear algebra methods.

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

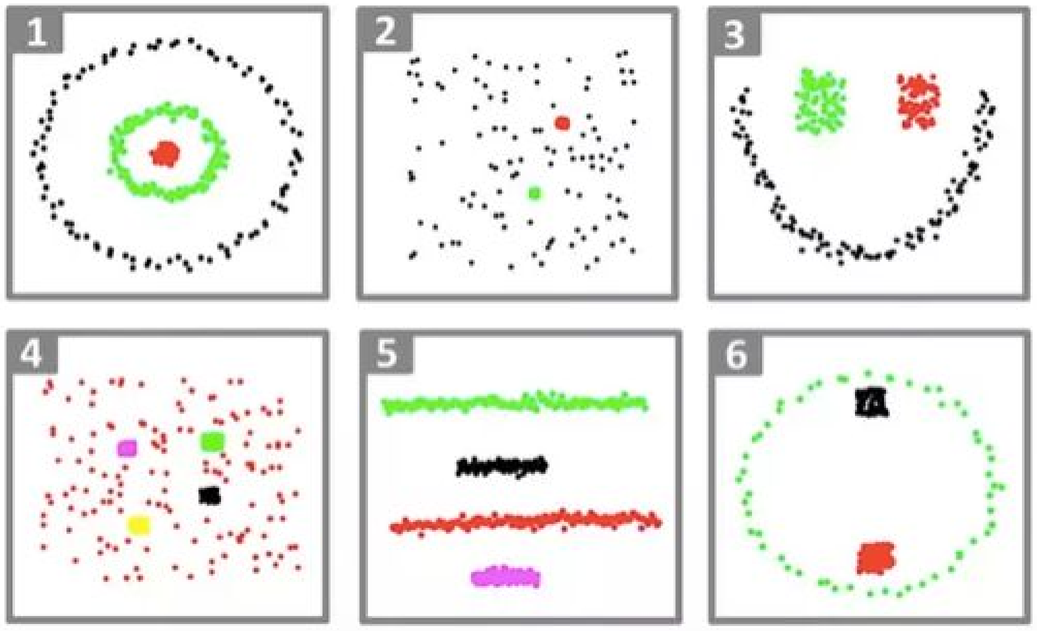
Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu

Spectral Clustering is a technique that follows this approach.



Datasets where spectral clustering is applied for clustering:



Lessons Learned

From this project, we learned how to learn with text data with natural language methods, and explored various text encoding methods. In addition, we learned how to work with unlabeled data by unsupervised learning methods. By working on this semester-long project, our team members learned how to do time management, how to break down a large project into smaller parts, and communicate with clients, appropriately presenting our works, and collaborate together as a group, which would be important in our future career.

Potential Future Work

The project mainly focuses on two parts, natural language processing of the log and unsupervised learning classification of them. Although several methods were implemented in this study, the result could still be improved by exploring other language processing methods and classification methods.

Gideon Sylvester Amoah
Cornell University
gsa45@cornell.edu

Cassie Xie
Cornell University
yx476@cornell.edu

Wenren Zhou
Cornell University
wz366@cornell.edu

References:

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000, May). LOF: identifying density-based local outliers. In ACM sigmod record (Vol. 29, №2, pp. 93–104). ACM.

https://i.stack.imgur.com/vg7G1.png

https://www.quora.com/How-does-TF-IDF-work

Artifact Index:

1. Final Project plan
2. Final 4-box
3. Risk Mitigation Matrix
4. Client Presentation
5. Python code: LAD_Word2Vec_Spectral_Clustering.Pdf
6. Python code: LAD_Doc2Vec_Local_Outlier_Factor.Pdf
7. Python code: LAD_TF-IDF_KMeans. Pdf
8. Dataset