# Work Distribution Report

Sushanth kulakrni

December 28, 2023

## Contents

## 1 Tasks

I have been allotted, the **Data Prepossessing** part of our project. Primarily I have been tasked with the following:

1. Find RAW log data, if unavailable find parsed log data.
2. Remove empty and duplicate log entries from RAW log data.
3. Parse RAW data into structured format, categorizing each entry into events.
4. Group the parsed, structured log data into set of events called a sequence.
5. Label the sequence of events as Normal or Anomalous and forward the same to Mahender.
6. Calculate evaluation metrics for a trained model.

## 2 Find Data

The following datasets have been considered for anomaly detection:

1. HDFS
2. BGL
3. Thunderbird
4. Spirit

RAW and parsed datasets for HDFS and BGL were available at:

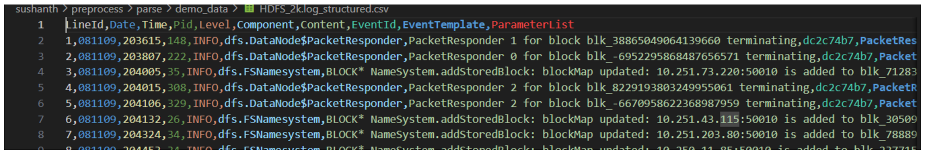
- `https://github.com/logpai/loglizer/tree/master/data/HDFS`
- `https://zenodo.org/records/8115559`
- `https://github.com/logpai/loghub/tree/master`

## 3 Parse Data

**Log Hub** only provides us 2k lines of raw data, which is not adequate, hence I've used 100k or more entries of log data available at **LogPai**.
Raw data is parsed using logparser python library and we use, **Drain** parser is primarily used.
The code segment for the same :

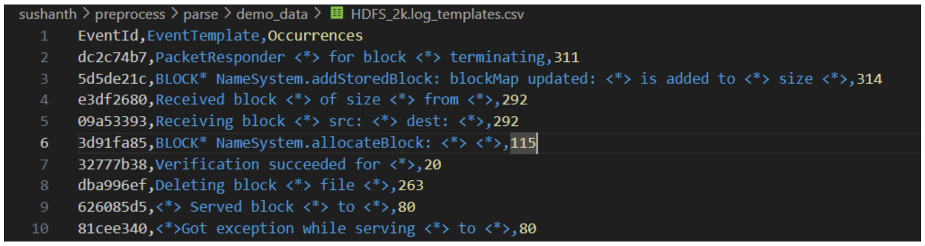
```python

import csv

input_log_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\sushanth\preprocess\parse\demo_data\HDFS_2k.log_structured.csv'

def extract_data(input_file, output_file):
    with open(input_file, 'r') as csvfile:
        reader = csv.DictReader(csvfile)

        data_list = [(row['EventId'], param) for row in reader for param in row['ParameterList'].strip("[]").replace("'", "").split(', ') if param.startswith('blk_')]

    with open(output_file, 'w', newline='') as csvfile:
        fieldnames = ['EventId', 'BlockString']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        writer.writeheader()
        for data in data_list:
            writer.writerow({'EventId': data[0], 'BlockString': data[1]})

extract_data(input_log_file, 'E:\Log_anomaly_detection\Log_anomaly_detection\sushanth\preprocess\group\ blk_eve_ids.csv')
```

The successful execution of this code results us with two files, a *Template* and a *Structured* csv file.
The structured file is of this format :



```
sushanth > preprocess > parse > demo_data > HDFS_2k.log_structured.csv
LineId,Date,Time,Pid,Level,Component,Content,EventId,EventTemplate,ParameterList
1,081109,203615,148,INFO,dfs.DataNode$PacketResponder,PacketResponder 1 for block blk_38865049064139660 terminating,dc2c74b7,PacketRes
2,081109,203807,222,INFO,dfs.DataNode$PacketResponder,PacketResponder 0 for block blk_-6952295868487656571 terminating,dc2c74b7,Packet
3,081109,204005,35,INFO,dfs.FSNamesystem,BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_71283
4,081109,204015,308,INFO,dfs.DataNode$PacketResponder,PacketResponder 2 for block blk_8229193803249955061 terminating,dc2c74b7,PacketR
5,081109,204106,329,INFO,dfs.DataNode$PacketResponder,PacketResponder 2 for block blk_-6670958622368987959 terminating,dc2c74b7,Packet
6,081109,204132,26,INFO,dfs.FSNamesystem,BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.43.115:50010 is added to blk_30509
7,081109,204324,34,INFO,dfs.FSNamesystem,BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.203.80:50010 is added to blk_78889
```

and the Template file is of this format :



```
sushanth > preprocess > parse > demo_data > HDFS_2k.log_templates.csv
EventId,EventTemplate,Occurrences
dc2c74b7,PacketResponder <*> for block <*> terminating,311
5d5de21c,BLOCK* NameSystem.addStoredBlock: blockMap updated: <*> is added to <*> size <*>,314
e3df2680,Received block <*> of size <*> from <*>,292
09a53393,Receiving block <*> src: <*> dest: <*>,292
3d91fa85,BLOCK* NameSystem.allocateBlock: <*> <*>,115
32777b38,Verification succeeded for <*>,20
dba996ef,Deleting block <*> file <*>,263
626085d5,<*> Served block <*> to <*>,80
81cee340,<*>Got exception while serving <*> to <*>,80
```

We can clearly see that we have multiple unique events, each such event represents the static part of a log entry.

## 4 Grouping parsed Data

The parsed data is now in a structured format, a key component here is *"Event Id"* ! Each eventId represent a particular static part of a log entry, we group these eventIds, let the set of evenets be called a sequence, $S_i$. We generate all such $S_i$ based on a grouping parameter.
Grouping parameters:

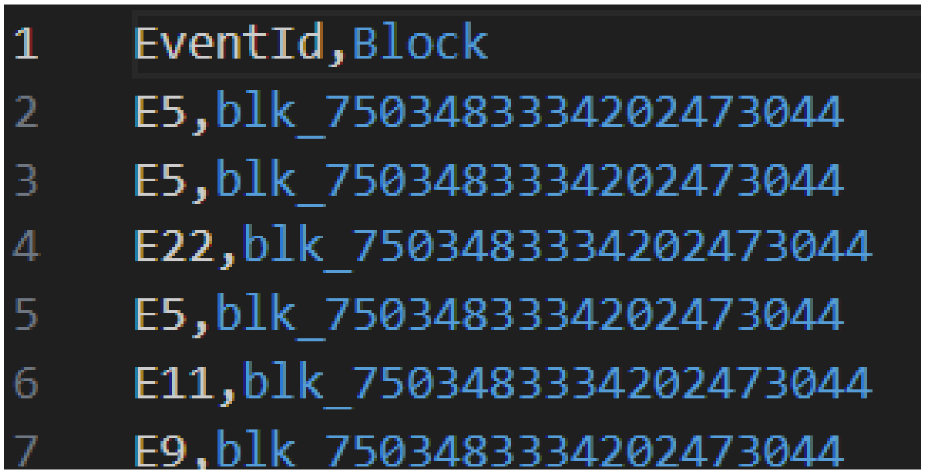
- HDFS : Session window protocol, **BlockId**
- BGL : Session window protocol, **Node Id**
- Thunderbird : Fixed Window, **Timestamp, window size: 900sec**
  *Prabas has helped me with grouping and labelling Thunderbird datset
- Spirit : Fixed Window protocol, **Timestamp, Window size: 113sec**

**Detailed Grouping for HDFS dataset :**

```
import csv
import re

def extract_event_and_block(content):
    block_pattern = r'blk_[\d]+'

    blocks = re.findall(block_pattern, content)

    return blocks

input_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\
    sushanth\preprocess\parse\proj_data\HDFS_100k.log_structured.
    csv'
output_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\
    sushanth\preprocess\group\proj\ blk_eve.csv'

with open(input_file, 'r') as infile, open(output_file, 'w',
    newline='') as outfile:
    reader = csv.DictReader(infile)
    fieldnames = ['EventId', 'Block']
    writer = csv.DictWriter(outfile, fieldnames=fieldnames)

    writer.writeheader()

    for row in reader:
        event_id = row['EventId']
        content = row['Content']
        blocks = extract_event_and_block(content)

        for block in blocks:
            writer.writerow({'EventId': event_id, 'Block': block})
```

In the above code, we group all the log entries for each and every blockId and then extract the Event Id from each such entry and print, The Block Id and it's corresponding Event Ids in a separate file, the fie is as follows:

Each of these are checked with Label.csv for hdfs dataset and the set of se-

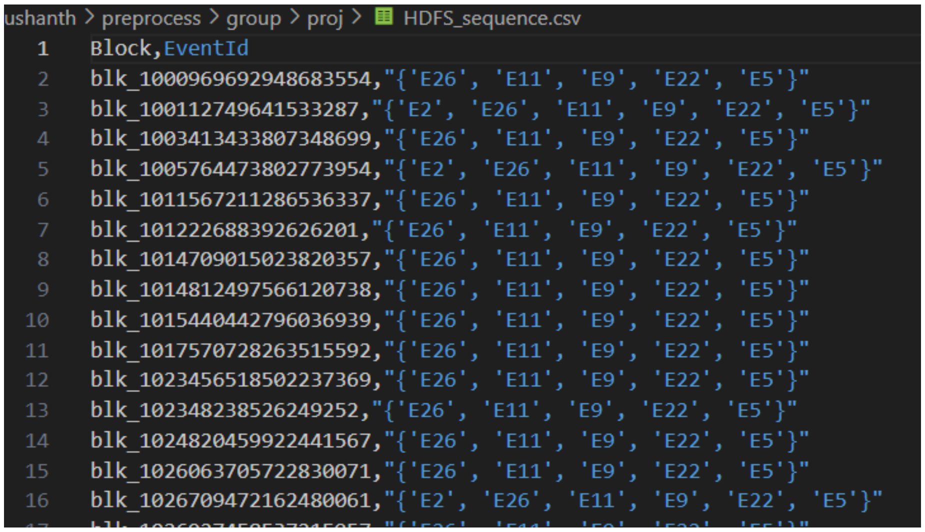


```
EventId,Block
E5,blk_7503483334202473044
E5,blk_7503483334202473044
E22,blk_7503483334202473044
E5,blk_7503483334202473044
E11,blk_7503483334202473044
E9,blk_7503483334202473044
```

quences are mapped either "Normal" or "Anomalous". //

```python
#grouping part 2 fro proj data HDFS
import pandas as pd

input_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\
    sushanth\preprocess\group\proj\ blk_eve.csv'  # Update with
    your actual file path
output_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\
    sushanth\preprocess\group\proj\sequence.csv'

df = pd.read_csv(input_file)

grouped_df = df.groupby('Block')['EventId'].agg(set).reset_index()

grouped_df.to_csv(output_file, index=False)

print("Grouping complete. Check the output file:", output_file)
```

The final, sequence and label mapping is of this format:



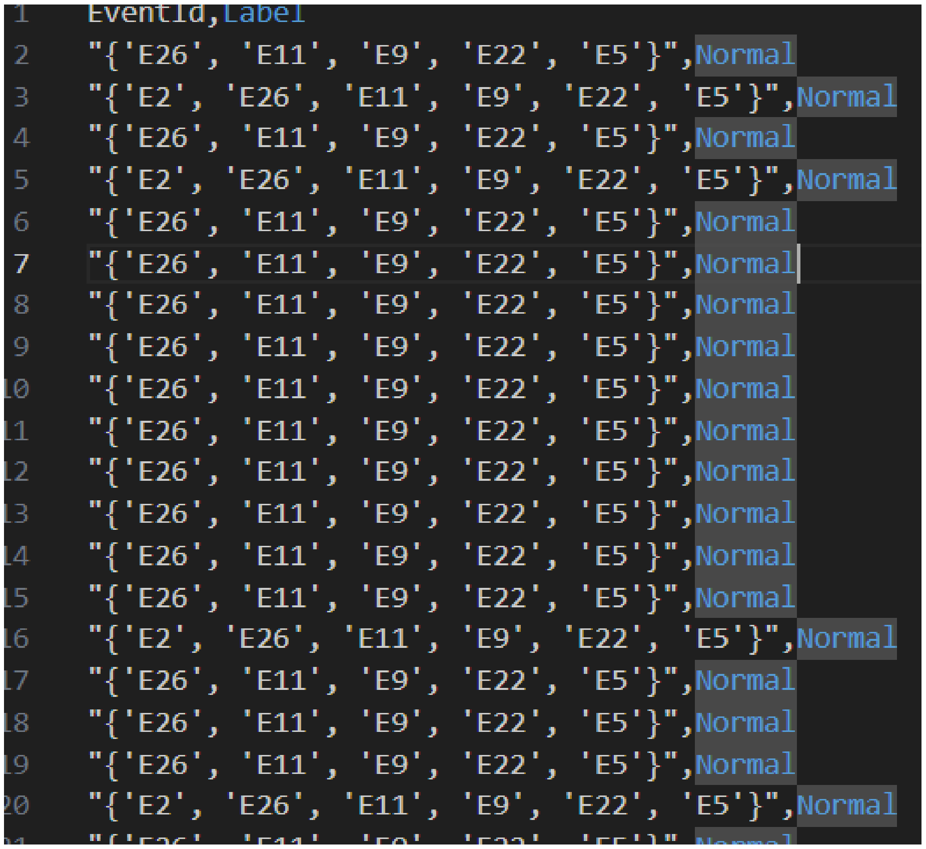
```
Block,EventId
blk_1000969692948683554,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1001127496415332287,"{'E2', 'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1003413433807348699,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1005764473802773954,"{'E2', 'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1011567211286536337,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_101222688392626201,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1014709015023820357,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1014812497566120738,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1015440442796036939,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1017570728263515592,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1023456518502237369,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_102348238526249252,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1024820459922441567,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1026063705722830071,"{'E26', 'E11', 'E9', 'E22', 'E5'}"
blk_1026709472162480061,"{'E2', 'E26', 'E11', 'E9', 'E22', 'E5'}"
```

## 5 Labelling Sequnence

The general idea: "*If all the events in a sequence are Normal, then the sequence is labelled 'Normal' else 'Anomalous'*"
Code for the same :

```python

import pandas as pd

extracted_info_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\sushanth\preprocess\group\proj\sequence.csv'  # Update with your actual file path
label_info_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\sushanth\preprocess\HDFS.anomaly_label.csv'  # Update with your actual file path
output_file = 'E:\Log_anomaly_detection\Log_anomaly_detection\sushanth\labelled_sequnce.csv'

extracted_df = pd.read_csv(extracted_info_file)
label_df = pd.read_csv(label_info_file)

merged_df = pd.merge(extracted_df, label_df, how='left', left_on='Block', right_on='BlockId')

merged_df = merged_df[['EventId', 'Label']]

merged_df.to_csv(output_file, index=False)

print("Merging complete. Check the output file:", output_file)
```

The labelled sequence is as follows:



```
EventId,Label
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E2', 'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E2', 'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E2', 'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
"{'E2', 'E26', 'E11', 'E9', 'E22', 'E5'}",Normal
```

## 6 Evaluation Metrics

The evaluation code is present in Mahender's repository, however he could not run the code due to some dependency issue, hence I've taken up Evaluation metrics.
Each model after being trained is evaluaed, and the metris obtained as :

- Confusion Matrix
- Accuracy
- precision

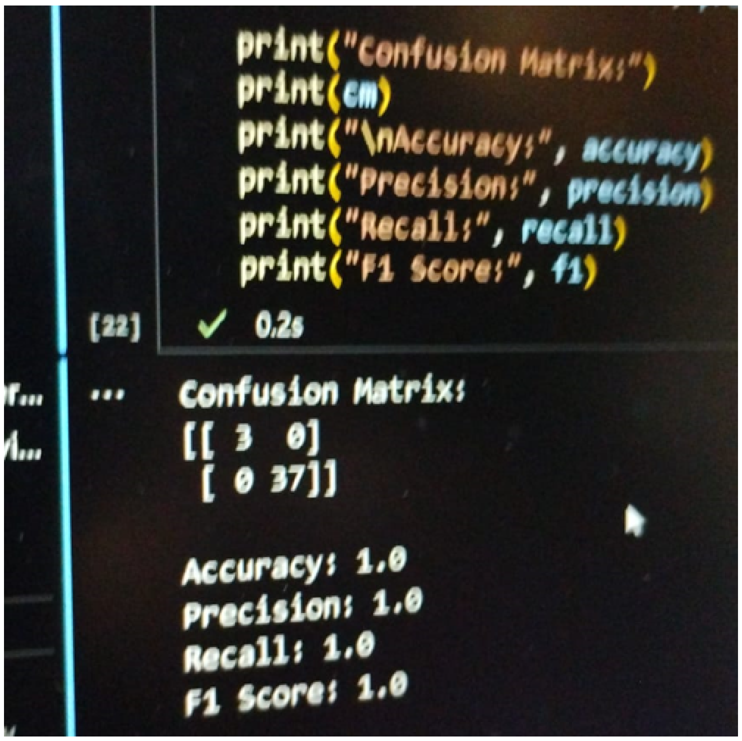
- Recall
- F1 Score

**NOTE:** Performing evaluation for 70,000 data entries required about 21gb of freely availale RAM, which was not available is any of our laptops(we had a maximum of 16gb RAM) hence, we performed evaluation metrics on about 1,000 entries.

The metrics for BGL trained model for about 1000 data entries of 70,000 entries is :
Confusion matrix:

$$\begin{bmatrix} 3 & 0 \\ 0 & 37 \end{bmatrix}$$

Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0



```
print("Confusion Matrix:")
print(cm)
print("\nAccuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

[22] ✓ 0.2s

```
Confusion Matrix:
[[ 3  0]
 [ 0 37]]

Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

## 7 Conclusion

A similar approach is taken for the remaining datasets as well to obtain labelled sequence files.

- HDFS : Session window protocol, **BlockId**
- BGL : Session window protocol, **Node Id**
- Thunderbird : Fixed Window, **Timestamp, window size: 900sec** *Prabas has helped me with grouping and labelling Thunderbird dataset
- Spirit : Fixed Window protocol, **Timestamp, Window size: 113sec**

## 8 Further Steps

This sequence file is generated for all the datasets and sent to Mahender for training BERT model at the client side in client-server federated environment which will be done by sathwik.