analysis

October 17, 2021

1 Log Parsing Benchmark Analysis

All log parsing algorithms were run 6 times each on a 12-core Intel Core i7-9750H CPU with 16GB of RAM running Pop_OS! 21.04.

[ ]:

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import re
```

1.1 Time Analysis

[ ]:

```
time_df = pd.DataFrame(columns=["algo", "dataset", "time"])
time_regex = r"(?<=Parsing done. \[Time taken: )\d:\d+:\d+.\d+"
time_regex_fallback = r"(?<=Parsing done. \[Time: )\d:\d+:\d+.\d+"

for filename in sorted(os.listdir("outputs")):
    [algo, iteration] = filename.split("_")
    with open("outputs/" + filename, "r") as f:
        times = re.findall(time_regex, f.read())
    if len(times) == 0:
        with open("outputs/" + filename, "r") as f:
            times = re.findall(time_regex_fallback, f.read())
    time_df = time_df.append(
        {
            "algo": algo,
            "dataset": "HDFS",
            "time": pd.Timedelta(times[0]).total_seconds(),
        },
        ignore_index=True,
    )
    time_df = time_df.append(
        {
            "algo": algo,
            "dataset": "BGL",
            "time": pd.Timedelta(times[1]).total_seconds(),
        },
        ignore_index=True,
    )

time_df
```

[ ]:

```
      algo dataset      time
0      AEL    HDFS  0.362798
1      AEL     BGL  0.270329
2      AEL    HDFS  0.285344
3      AEL     BGL  0.275296
4      AEL    HDFS  0.298853
..     …       …        …
151  Spell     BGL  0.655828
152  Spell    HDFS  0.398777
153  Spell     BGL  0.665933
154  Spell    HDFS  0.391038
155  Spell     BGL  0.660896

[156 rows x 3 columns]
```

[ ]:

```
avg_time_df = time_df.groupby(['algo', 'dataset']).mean(numeric_only=False).
 ↪reset_index()
avg_time_df
```

[ ]:

```
          algo dataset        time
0          AEL     BGL    0.265144
1          AEL    HDFS    0.303317
2        Drain     BGL    0.340670
3        Drain    HDFS    0.375423
4        IPLoM     BGL    0.302273
5        IPLoM    HDFS    0.305821
6          LFA     BGL    0.187106
7          LFA    HDFS    0.210758
8          LKE     BGL   57.855254
9          LKE    HDFS   54.928430
10       Lenma     BGL    2.359167
11       Lenma    HDFS    0.440831
12  LogCluster     BGL    0.203864
13  LogCluster    HDFS    0.196447
14     LogMine     BGL    3.815657
15     LogMine    HDFS    6.311744
16      LogSig     BGL  129.497067
17      LogSig    HDFS    2.398137
18       MoLFI     BGL   26.258780
19       MoLFI    HDFS    3.910462
20       SHISO     BGL    4.869288
21       SHISO    HDFS    1.304586
22        SLCT     BGL    1.299598
```

```
23        SLCT    HDFS    0.675247
24       Spell     BGL    0.663679
25       Spell    HDFS    0.395011
```

1.2 F1-measure and Accuracy Analysis

[ ]:

```
results_df = pd.DataFrame(columns=["algo", "dataset", "f1_measure", "accuracy"])

for filename in sorted(os.listdir("results")):
    algo = filename.split("_")[0]
    with open("results/" + filename, "r") as f:
        lines = f.readlines()
        lines = [line.strip() for line in lines]
        results_df = results_df.append(
            {
                "algo": algo,
                "dataset": "HDFS",
                "f1_measure": float(lines[1].split(",")[1]),
                "accuracy": float(lines[2].split(",")[1]),
            },
            ignore_index=True,
        )
        results_df = results_df.append(
            {
                "algo": algo,
                "dataset": "BGL",
                "f1_measure": float(lines[1].split(",")[2]),
                "accuracy": float(lines[2].split(",")[2]),
            },
            ignore_index=True,
        )

results_df
```

[ ]:

```
          algo dataset  f1_measure  accuracy
0          AEL    HDFS    0.999984    0.9975
1          AEL     BGL    0.999554    0.9570
2        Drain    HDFS    0.999984    0.9975
3        Drain     BGL    0.999599    0.9625
4        IPLoM    HDFS    1.000000    1.0000
5        IPLoM     BGL    0.999110    0.9390
6          LFA    HDFS    0.999545    0.8850
7          LFA     BGL    0.997902    0.8540
8          LKE    HDFS    1.000000    1.0000
9          LKE     BGL    0.399353    0.1275
10       Lenma    HDFS    0.999984    0.9975
11       Lenma     BGL    0.939369    0.6895
```

```
12  LogCluster    HDFS    0.951863    0.5460
13  LogCluster     BGL    0.996965    0.8350
14     LogMine    HDFS    0.998840    0.8505
15     LogMine     BGL    0.971268    0.7230
16      LogSig    HDFS    0.991767    0.8495
17      LogSig     BGL    0.934917    0.2265
18       MoLFI    HDFS    0.999984    0.9975
19       MoLFI     BGL    0.999778    0.9660
20       SHISO    HDFS    0.999984    0.9975
21       SHISO     BGL    0.994450    0.7110
22        SLCT    HDFS    0.965812    0.5450
23        SLCT     BGL    0.955247    0.5725
24       Spell    HDFS    1.000000    1.0000
25       Spell     BGL    0.956932    0.7865
```

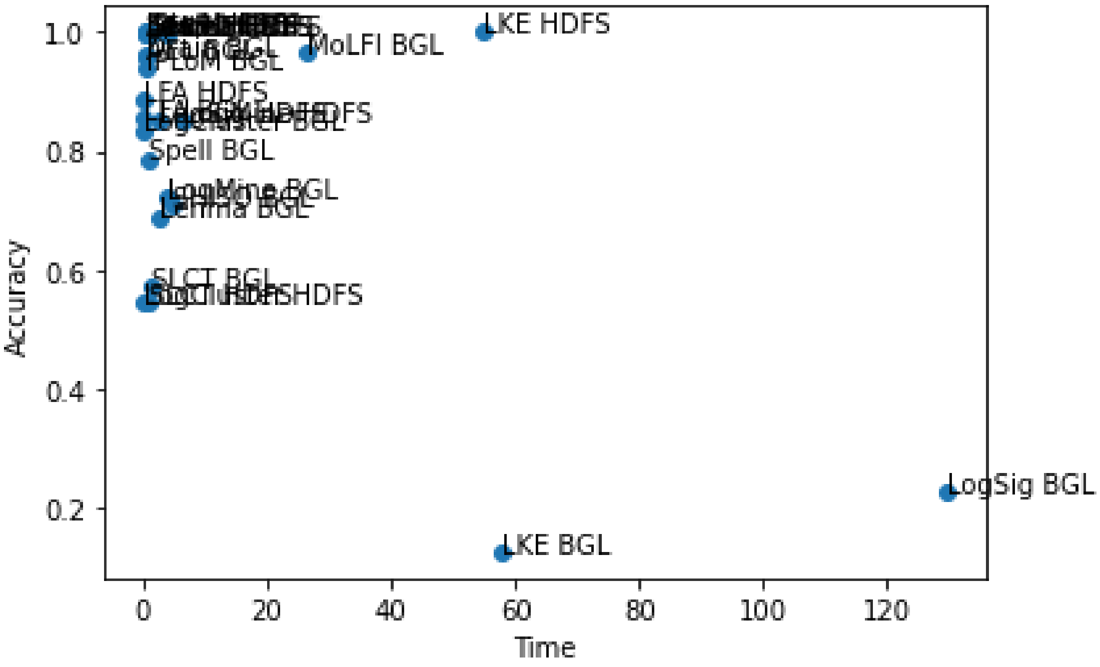
[ ]:

```
final_df = results_df.merge(avg_time_df, on=['algo', 'dataset'])
final_df
```

[ ]:

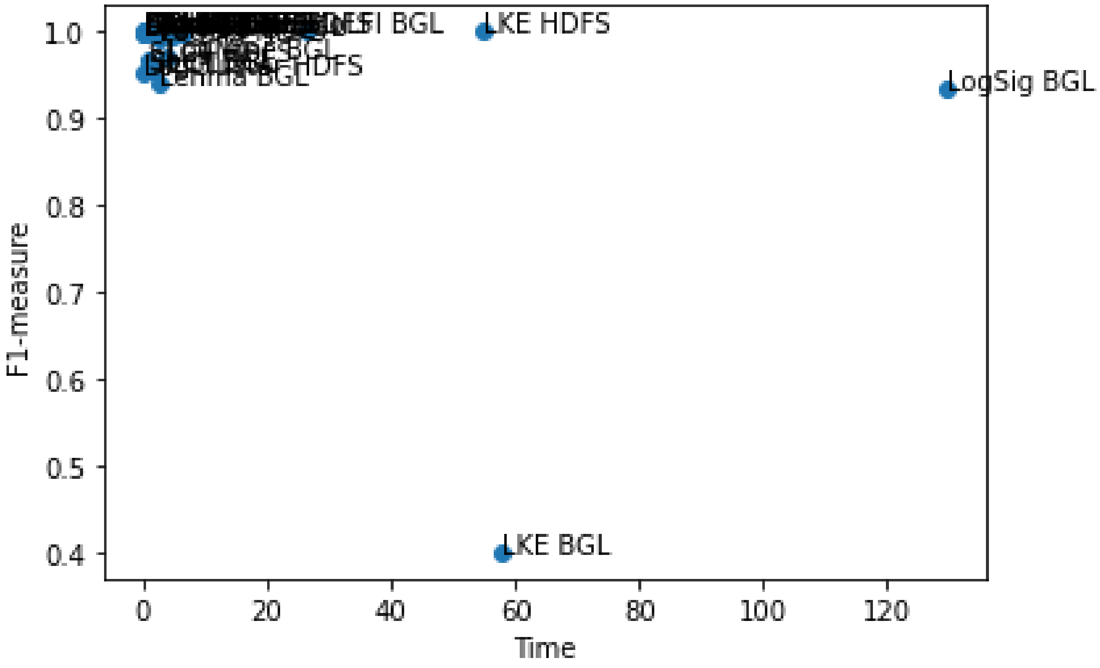
```
          algo dataset  f1_measure  accuracy        time
0          AEL    HDFS    0.999984    0.9975    0.303317
1          AEL     BGL    0.999554    0.9570    0.265144
2        Drain    HDFS    0.999984    0.9975    0.375423
3        Drain     BGL    0.999599    0.9625    0.340670
4        IPLoM    HDFS    1.000000    1.0000    0.305821
5        IPLoM     BGL    0.999110    0.9390    0.302273
6          LFA    HDFS    0.999545    0.8850    0.210758
7          LFA     BGL    0.997902    0.8540    0.187106
8          LKE    HDFS    1.000000    1.0000   54.928430
9          LKE     BGL    0.399353    0.1275   57.855254
10       Lenma    HDFS    0.999984    0.9975    0.440831
11       Lenma     BGL    0.939369    0.6895    2.359167
12  LogCluster    HDFS    0.951863    0.5460    0.196447
13  LogCluster     BGL    0.996965    0.8350    0.203864
14     LogMine    HDFS    0.998840    0.8505    6.311744
15     LogMine     BGL    0.971268    0.7230    3.815657
16      LogSig    HDFS    0.991767    0.8495    2.398137
17      LogSig     BGL    0.934917    0.2265  129.497067
18       MoLFI    HDFS    0.999984    0.9975    3.910462
19       MoLFI     BGL    0.999778    0.9660   26.258780
20       SHISO    HDFS    0.999984    0.9975    1.304586
21       SHISO     BGL    0.994450    0.7110    4.869288
22        SLCT    HDFS    0.965812    0.5450    0.675247
23        SLCT     BGL    0.955247    0.5725    1.299598
24       Spell    HDFS    1.000000    1.0000    0.395011
25       Spell     BGL    0.956932    0.7865    0.663679
```

```
[ ]: plt.scatter(final_df['time'], final_df['accuracy'])
     plt.xlabel('Time')
     plt.ylabel('Accuracy')
     for i, row in final_df.iterrows():
       plt.annotate(f"{row['algo']} {row['dataset']}", (row['time'],␣
      ↪row['accuracy']))
```



Most algorithms are fast, although they have varying accuracies. LKE and LogSig are clear outliers, due to a longer execution time.

```
[ ]: plt.scatter(final_df['time'], final_df['f1_measure'])
     plt.xlabel('Time')
     plt.ylabel('F1-measure')
     for i, row in final_df.iterrows():
       plt.annotate(f"{row['algo']} {row['dataset']}", (row['time'],␣
      ↪row['f1_measure']))
```

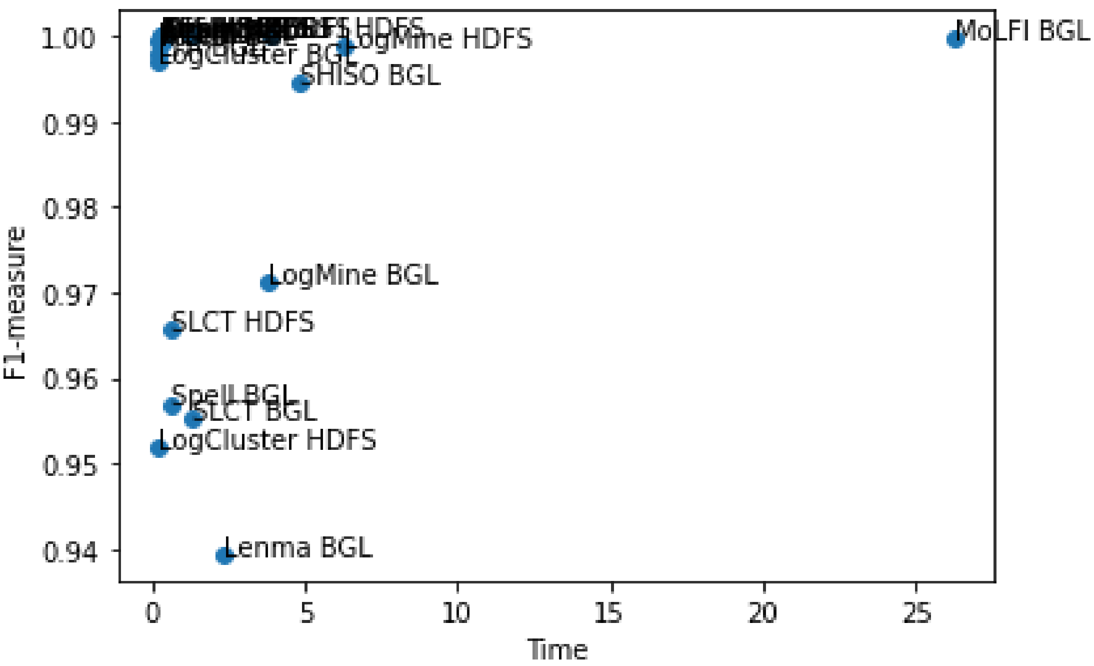


We can notice that LKE and LogSig are also outliers in terms of F1-measure over time. These algorithms are not efficient.

Removing them yields the following graph:

[ ]:

```
final_df = final_df[(final_df.algo != 'LKE') & (final_df.algo != 'LogSig')]
plt.scatter(final_df['time'], final_df['f1_measure'])
plt.xlabel('Time')
plt.ylabel('F1-measure')
for i, row in final_df.iterrows():
  plt.annotate(f"{row['algo']} {row['dataset']}", (row['time'],␣
 ↪row['f1_measure']))
```


MoLFI BGL
LogMine HDFS
LogCluster BGL
SHISO BGL
LogMine BGL
SLCT HDFS
Spell BGL
SLCT BGL
LogCluster HDFS
Lenma BGL
1.00
0.99
0.98
0.97
0.96
0.95
0.94
F1-measure
0
5
10
15
20
25
Time