Tools and Benchmarks for Automated Log Parsing

Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, Michael R. Lyu
Logs are used to **record runtime information** of a system.

```scala
def get[T: ClassTag](blockId: BlockId): Option[BlockResult] = {
  val local = getLocalValues(blockId)
  if (local.isDefined) {
    logInfo(s"Found block $blockId locally")
    return local
  }
  val remote = getRemoteValues[T](blockId)
  if (remote.isDefined) {
    logInfo(s"Found block $blockId remotely")
    return remote
  }
  None
}
```
Logs are used to **record runtime information** of a system.

```scala
logInfo(s"Found block $blockId remotely")
```
Logs are used to **record runtime information** of a system

```scala
logInfo(s"Found block $blockId remotely")
```

```
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_11 remotely
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_12 remotely
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_14 remotely
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_13 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_20 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_22 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_23 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_24 remotely
```
Logs are used to record runtime information of a system.

```scala
logInfo(s"Found block $blockId remotely")
```
Logs are used to **record runtime information** of a system

```scala
logInfo(s"Found block $blockId remotely")
```

```
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_11 remotely
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_12 remotely
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_14 remotely
17/06/09 20:11:10 INFO storage.BlockManager: Found block rdd_42_13 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_20 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_22 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_23 remotely
17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_24 remotely
```
Log analysis is widely employed to enhance system reliability
AI-Powered Log Analytics

Development Phase

Runtime Phase

Operation and Maintenance Phase

Where to log
What to log

Log Collection
Log Compression

Log Parsing
Log Analysis
LogAdvisor
• Learning to log: A framework for determining optimal logging points
  [ICSE’14, ICSE’15]

LogHub
• A collection of system log datasets for massive log analysis
  [FSE’19 under review]

Loglizer
• A log analysis toolkit for automated anomaly detection
  [ISSRE'16]

LoggingDescriptions
• A collection of Software Logging Statements
  [ASE’18]

LogParser
• A toolkit for automated log parsing
  [ICSE'19, TDSC'18, DSN'16]

Log3C
• Log-based Problem Identification
  [FSE’18]
Logs are unstructured

Raw Log 2008-11-11 03:41:48. Received block blk_90 of size 67108864 from /10.250.18.114

What’s this? I only understand **structured** data

Log Analysis AI
Log analysis models require structured input

2008-11-11 03:41:48. Received block blk_90 of size 67108864 from /10.250.18.114

<table>
<thead>
<tr>
<th>EventId</th>
<th>EventTemplate</th>
<th>ParameterList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event 1</td>
<td>Received block * of size * from *</td>
<td>[blk_90, 67108864, 67108864, /10.250.18.114]</td>
</tr>
</tbody>
</table>

Oh! This is Event 1 happened on block blk_90
The goal of log parsing is to distinguish between constant part and variable part from the log contents.
Log parsing

The goal of log parsing is to distinguish between constant part and variable part from the log contents.

**Raw Log**

2008-11-11 03:41:48. Received block blk_90 of size 67108864 from /10.250.18.114

**Structured Log**

<table>
<thead>
<tr>
<th>EventId</th>
<th>EventTemplate</th>
<th>ParameterList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event 1</td>
<td>Received block * of size * from *</td>
<td>[blk_90, 67108864, 67108864, /10.250.18.114]</td>
</tr>
</tbody>
</table>
The goal of log parsing is to distinguish between constant part and variable part from the log contents.
Log parsing is the first and a significant step towards the whole process of log analysis.
Many log parsers are produced

SLCT 2003
AEL 2008
LKE 2009
LFA 2010
LogSig 2011
IPLOM 2012
SHISO 2013

How they perform? Which to choose? How to use?

2015 LogCluster
2016 LenMa, LogMine, Spell
2017 Drain
2018 MoLFI
Many log parsers are produced

How they perform?
Which to choose?
How to use?
Contribution of our work

- Release an easy-to-use, open-source toolkit of 13 recent log parsing methods
- Evaluation on 16 real-world datasets in terms of accuracy, robustness, efficiency
- Success stories in an industrial application at Huawei
Contribution of our work

• Release an easy-to-use, open-source toolkit of 13 recent log parsing methods

• Evaluation on 16 real-world datasets in terms of accuracy, robustness, efficiency

• Success stories in an industrial application at Huawei
Contribution of our work

- Release an easy-to-use, open-source toolkit of 13 recent log parsing methods

- Evaluation on 16 real-world datasets in terms of accuracy, robustness, efficiency

- **Success stories** in an industrial application at Huawei
Toolkit: Logparser

The easy-to-use toolkit containing 13 log parsing methods is open source on https://github.com/logpai/logparser
## Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
<th>Time Span</th>
<th>Data Size</th>
<th>#Messages</th>
<th>#Templates (total)</th>
<th>#Templates (2k)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distributed system logs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDFS</td>
<td>Hadoop distributed file system log</td>
<td>38.7 hours</td>
<td>1.47 GB</td>
<td>11,175,629</td>
<td>48</td>
<td>14</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Hadoop mapreduce job log</td>
<td>N.A.</td>
<td>48.61 MB</td>
<td>394,308</td>
<td>298</td>
<td>114</td>
</tr>
<tr>
<td>Spark</td>
<td>Spark job log</td>
<td>3.46 days</td>
<td>35.35 MB</td>
<td>364,56</td>
<td>456</td>
<td>36</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>ZooKeeper service log</td>
<td>26.7 days</td>
<td>795 MB</td>
<td>74,380</td>
<td>395</td>
<td>50</td>
</tr>
<tr>
<td>OpenStack</td>
<td>OpenStack software log</td>
<td>N.A.</td>
<td>60.01 MB</td>
<td>207,820</td>
<td>207</td>
<td>43</td>
</tr>
<tr>
<td><strong>Supercomputer logs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGL</td>
<td>Blue Gene/L supercomputer log</td>
<td>214.7 days</td>
<td>708.76 MB</td>
<td>4,747,963</td>
<td>619</td>
<td>120</td>
</tr>
<tr>
<td>HPC</td>
<td>High performance cluster log</td>
<td>244 days</td>
<td>29.60 GB</td>
<td>211,212,192</td>
<td>4,040</td>
<td>149</td>
</tr>
<tr>
<td>Thunderbird</td>
<td>Thunderbird supercomputer log</td>
<td>244 days</td>
<td>29.60 GB</td>
<td>211,212,192</td>
<td>4,040</td>
<td>149</td>
</tr>
<tr>
<td><strong>Operating system logs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>Windows event log</td>
<td>226.7 days</td>
<td>26.09 GB</td>
<td>114,608,388</td>
<td>4,833</td>
<td>50</td>
</tr>
<tr>
<td>Linux</td>
<td>Linux system log</td>
<td>7.0 days</td>
<td>16.09 MB</td>
<td>107,283</td>
<td>2,14</td>
<td>341</td>
</tr>
<tr>
<td>Mac</td>
<td>Mac OS log</td>
<td>7.0 days</td>
<td>16.09 MB</td>
<td>107,283</td>
<td>2,14</td>
<td>341</td>
</tr>
<tr>
<td><strong>Mobile system logs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Android</td>
<td>Android framework log</td>
<td>N.A.</td>
<td>3.38 GB</td>
<td>30,348,042</td>
<td>76,923</td>
<td>166</td>
</tr>
<tr>
<td>HealthApp</td>
<td>Health app log</td>
<td>N.A.</td>
<td>3.38 GB</td>
<td>30,348,042</td>
<td>76,923</td>
<td>166</td>
</tr>
<tr>
<td><strong>Server application logs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache</td>
<td>Apache server error log</td>
<td>263.9 days</td>
<td>4.90 MB</td>
<td>56,481</td>
<td>64</td>
<td>8</td>
</tr>
<tr>
<td>OpenSSH</td>
<td>OpenSSH server log</td>
<td>263.9 days</td>
<td>4.90 MB</td>
<td>56,481</td>
<td>64</td>
<td>8</td>
</tr>
<tr>
<td><strong>Standalone software logs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proxifier</td>
<td>Proxifier software log</td>
<td>3.46 days</td>
<td>9.07 MB</td>
<td>363,435</td>
<td>363</td>
<td>58</td>
</tr>
</tbody>
</table>
## Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
<th>Time Span</th>
<th>Data Size</th>
<th>#Messages</th>
<th>#Templates (total)</th>
<th>#Templates (2k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDFS</td>
<td>Hadoop distributed file system log</td>
<td>38.7 hours</td>
<td>1.47 GB</td>
<td>11,175,629</td>
<td>48</td>
<td>14</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Hadoop mapreduce job log</td>
<td>N.A.</td>
<td>48.61 MB</td>
<td>394,308</td>
<td>298</td>
<td>114</td>
</tr>
<tr>
<td>Spark</td>
<td>Spark job log</td>
<td>N.A.</td>
<td>2.75 GB</td>
<td>33,236,604</td>
<td>456</td>
<td>36</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>ZooKeeper service log</td>
<td>26.7 days</td>
<td>9.95 MB</td>
<td>74,380</td>
<td>95</td>
<td>50</td>
</tr>
<tr>
<td>OpenStack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43</td>
</tr>
<tr>
<td>BGL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>120</td>
</tr>
<tr>
<td>HPC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>Thunderbird</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>149</td>
</tr>
<tr>
<td>Windows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Linux</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>118</td>
</tr>
<tr>
<td>Mac</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>341</td>
</tr>
<tr>
<td>Android</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>166</td>
</tr>
<tr>
<td>HealthApp</td>
<td>Health app log</td>
<td>10.5 days</td>
<td>22.44 MB</td>
<td>253,395</td>
<td>220</td>
<td>75</td>
</tr>
<tr>
<td>Apache</td>
<td>Apache server error log</td>
<td>263.9 days</td>
<td>4.90 MB</td>
<td>56,481</td>
<td>44</td>
<td>6</td>
</tr>
<tr>
<td>OpenSSH</td>
<td>OpenSSH server log</td>
<td>28.4 days</td>
<td>70.02 MB</td>
<td>655,146</td>
<td>62</td>
<td>27</td>
</tr>
<tr>
<td>Proxifier</td>
<td>Proxifier software log</td>
<td>N.A.</td>
<td>2.42 MB</td>
<td>21,329</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

**Time Span:** \(~5\) years  
**# of lines:** 440 million  
**Total Size:** 77 GB  
**2k labeled log message for each dataset**
Evaluation

RQ1: What is the **accuracy** of the state-of-the-art log parsing methods?

RQ2: What is the **robustness** of these methods on different types and volumes of logs?

RQ3: What is the **efficiency** of these methods?
RQ1: What is the accuracy of the state-of-the-art log parsing methods?

RQ2: What is the robustness of these methods on different types and volumes of logs?

RQ3: What is the efficiency of these methods?
RQ1: Accuracy  RQ2: Robustness  RQ3: Efficiency

Accuracy Distribution of Log Parsers on **Different Log Types**

Parsing accuracy across datasets
RQ1: Accuracy  RQ2: Robustness  RQ3: Efficiency

Drain* > IPLoM > AEL > Others

Accuracy Distribution of Log Parsers on Different Log Types

### Dataset Parsing Methods

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SLCT</th>
<th>AEL</th>
<th>IPLoM</th>
<th>LKE</th>
<th>LFA</th>
<th>LogSig</th>
<th>SHISO</th>
<th>LogCluster</th>
<th>LenMa</th>
<th>LogMine</th>
<th>Spell</th>
<th>Drain</th>
<th>MoLFi</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDFS</td>
<td>0.545</td>
<td>0.998</td>
<td>1*</td>
<td>1*</td>
<td>0.885</td>
<td>0.850</td>
<td>0.998</td>
<td>0.546</td>
<td>0.998</td>
<td>0.851</td>
<td>1*</td>
<td>0.998</td>
<td>0.998</td>
<td>1</td>
</tr>
<tr>
<td>Hadoop</td>
<td>0.423</td>
<td>0.538</td>
<td>0.954</td>
<td>0.670</td>
<td>0.900</td>
<td>0.633</td>
<td>0.867</td>
<td>0.563</td>
<td>0.885</td>
<td>0.870</td>
<td>0.778</td>
<td>0.948</td>
<td>0.957*</td>
<td>0.957</td>
</tr>
<tr>
<td>Spark</td>
<td>0.685</td>
<td>0.905</td>
<td>0.920</td>
<td>0.634</td>
<td>0.994*</td>
<td>0.544</td>
<td>0.906</td>
<td>0.799</td>
<td>0.884</td>
<td>0.576</td>
<td>0.905</td>
<td>0.920</td>
<td>0.413</td>
<td>0.994</td>
</tr>
<tr>
<td>Zookeeper</td>
<td>0.726</td>
<td>0.921</td>
<td>0.962</td>
<td>0.438</td>
<td>0.839</td>
<td>0.738</td>
<td>0.660</td>
<td>0.732</td>
<td>0.841</td>
<td>0.688</td>
<td>0.964</td>
<td>0.967*</td>
<td>0.839</td>
<td>0.967</td>
</tr>
<tr>
<td>OpenStack</td>
<td>0.867</td>
<td>0.758</td>
<td>0.871*</td>
<td>0.878</td>
<td>0.200</td>
<td>0.200</td>
<td>0.722</td>
<td>0.696</td>
<td>0.743</td>
<td>0.743</td>
<td>0.764</td>
<td>0.733</td>
<td>0.216</td>
<td>0.871</td>
</tr>
<tr>
<td>BGL</td>
<td>0.573</td>
<td>0.758</td>
<td>0.939</td>
<td>0.128</td>
<td>0.854</td>
<td>0.227</td>
<td>0.711</td>
<td>0.835</td>
<td>0.69</td>
<td>0.723</td>
<td>0.787</td>
<td>0.963*</td>
<td>0.960</td>
<td>0.963</td>
</tr>
<tr>
<td>HPC</td>
<td>0.839</td>
<td>0.903*</td>
<td>0.824</td>
<td>0.574</td>
<td>0.817</td>
<td>0.354</td>
<td>0.325</td>
<td>0.788</td>
<td>0.830</td>
<td>0.784</td>
<td>0.654</td>
<td>0.887</td>
<td>0.821</td>
<td>0.903</td>
</tr>
<tr>
<td>Thunderb.</td>
<td>0.882</td>
<td>0.941</td>
<td>0.663</td>
<td>0.813</td>
<td>0.649</td>
<td>0.694</td>
<td>0.576</td>
<td>0.599</td>
<td>0.943</td>
<td>0.919</td>
<td>0.844</td>
<td>0.955*</td>
<td>0.645</td>
<td>0.955</td>
</tr>
<tr>
<td>Windows</td>
<td>0.697</td>
<td>0.690</td>
<td>0.567</td>
<td>0.990</td>
<td>0.588</td>
<td>0.689</td>
<td>0.701</td>
<td>0.713</td>
<td>0.566</td>
<td>0.993</td>
<td>0.989</td>
<td>0.997*</td>
<td>0.406</td>
<td>0.997</td>
</tr>
<tr>
<td>Linux</td>
<td>0.297</td>
<td>0.673</td>
<td>0.672</td>
<td>0.519</td>
<td>0.279</td>
<td>0.169</td>
<td>0.701</td>
<td>0.629</td>
<td>0.701*</td>
<td>0.612</td>
<td>0.605</td>
<td>0.690</td>
<td>0.284</td>
<td>0.701</td>
</tr>
<tr>
<td>Mac</td>
<td>0.558</td>
<td>0.764</td>
<td>0.673</td>
<td>0.369</td>
<td>0.599</td>
<td>0.478</td>
<td>0.595</td>
<td>0.604</td>
<td>0.698</td>
<td>0.872*</td>
<td>0.757</td>
<td>0.787</td>
<td>0.635</td>
<td>0.872</td>
</tr>
<tr>
<td>Android</td>
<td>0.882</td>
<td>0.682</td>
<td>0.712</td>
<td>0.909</td>
<td>0.616</td>
<td>0.548</td>
<td>0.585</td>
<td>0.798</td>
<td>0.880</td>
<td>0.504</td>
<td>0.919*</td>
<td>0.911</td>
<td>0.788</td>
<td>0.919</td>
</tr>
<tr>
<td>HealthApp</td>
<td>0.331</td>
<td>0.568</td>
<td>0.822*</td>
<td>0.592</td>
<td>0.549</td>
<td>0.235</td>
<td>0.397</td>
<td>0.531</td>
<td>0.174</td>
<td>0.684</td>
<td>0.639</td>
<td>0.780</td>
<td>0.440</td>
<td>0.822</td>
</tr>
<tr>
<td>Apache</td>
<td>0.731</td>
<td>1*</td>
<td>1*</td>
<td>1*</td>
<td>0.582</td>
<td>1*</td>
<td>0.709</td>
<td>1*</td>
<td>1*</td>
<td>1*</td>
<td>1*</td>
<td>1*</td>
<td>1*</td>
<td>1</td>
</tr>
<tr>
<td>OpenSSH</td>
<td>0.521</td>
<td>0.538</td>
<td>0.802</td>
<td>0.426</td>
<td>0.501</td>
<td>0.373</td>
<td>0.619</td>
<td>0.426</td>
<td>0.925*</td>
<td>0.431</td>
<td>0.554</td>
<td>0.788</td>
<td>0.500</td>
<td>0.925</td>
</tr>
<tr>
<td>Proxifier</td>
<td>0.518</td>
<td>0.518</td>
<td>0.515</td>
<td>0.495</td>
<td>0.826</td>
<td>0.866</td>
<td>0.967*</td>
<td>0.517</td>
<td>0.951</td>
<td>0.508</td>
<td>0.517</td>
<td>0.527</td>
<td>0.527</td>
<td>0.967</td>
</tr>
<tr>
<td>Average</td>
<td>0.637</td>
<td>0.754</td>
<td>0.777</td>
<td>0.563</td>
<td>0.652</td>
<td>0.482</td>
<td>0.669</td>
<td>0.665</td>
<td>0.721</td>
<td>0.694</td>
<td>0.751</td>
<td>0.865*</td>
<td>0.605</td>
<td>N.A.</td>
</tr>
</tbody>
</table>

**Accuracy of Log Parsers on Different Datasets**

RQ1: Accuracy  
RQ2: Robustness  
RQ3: Efficiency
RQ1: What is the accuracy of the state-of-the-art log parsing methods?

RQ2: What is the robustness of these methods on different types and volumes of logs?

RQ3: What is the efficiency of these methods?
Drain achieves relatively stable accuracy.

Accuracy of Log Parsers on Varying Log Size

RQ1: Accuracy  RQ2: Robustness  RQ3: Efficiency
RQ1: What is the accuracy of the state-of-the-art log parsing methods?

RQ2: What is the robustness of these methods on different types and volumes of logs?

RQ3: What is the efficiency of these methods?
RQ1: Accuracy  RQ2: Robustness  RQ3: Efficiency

IPLoM ≈ AEL ≈ Drain > others

Efficiency of Log Parsers on Varying Log Size
Logs from System X

Automate parsing
- Accurate
- Robust
- Efficient

Log Parsing

Log Analysis
- Failure diagnosis
- Performance optimization
- User profiling
- Resource allocation

Industrial Deployment
Optimizing Drain

- **Preprocess**: Filter package name, number, IP, and file path
- **Deduplication**: Reduce data size
- **Partitioning**: Divide by level and component information
- **Parallelization**: Extend Drain with Spark Streaming

Attain over **90% accuracy** in System X
Our work are well accepted in the community

- Our github gains ~1000 stars
- Datasets downloaded ~2300 times by ~100 organizations

- **loghub**
  - A large collection of system log datasets for AI-based Impactful Problem Identification powered log analytics
  - Stars: 221, Watchers: 83

- **Log3C**
  - Log-based Impactful Problem Identification
  - Machine Learning [FSE'18]
  - Languages: Python
  - Stars: 83, Watchers: 11

- **logparser**
  - A toolkit for automated log parsing [ICSE'19, TDSC'18, DSN'16]
  - Stars: 218, Watchers: 119

- **loglizer**
  - A log analysis toolkit for automated anomaly detection [ISSRE'16]
  - Languages: Python
  - Stars: 401, Watchers: 134
Questions and Cooperations are welcome!

www.logpai.com
info@logpai.com