Spell: Streaming Parsing of System Event Logs

Min Du, Feifei Li
School of Computing,
University of Utah
Background

15/07/31 12:20:17 INFO SparkContext: Running Spark version 1.3.0
15/07/31 12:20:18 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
15/07/31 12:20:18 INFO SecurityManager: Changing view acts to: zhoulang
15/07/31 12:20:18 INFO SecurityManager: Changing modify acts to: zhoulang
15/07/31 12:20:18 INFO SecurityManager: SecurityManager: authentication disabled; ut acts disabled; users with view permissions: Set(zhoulang); users with modify permissions: Set(zhoulang)
15/07/31 12:20:18 INFO Slf4jLogger: Slf4jLogger started
15/07/31 12:20:18 INFO Remote: Starting remoting
15/07/31 12:20:18 INFO Remote: Remote started; listening on addresses [akka.tcp://sparkDriver@head:60626]
15/07/31 12:20:18 INFO Util: Successfully started service 'sparkDriver' on port 60626.
15/07/31 12:20:18 INFO SparkEnv: Registering MapOutputTracker
15/07/31 12:20:18 INFO SparkEnv: Registering BlockManagerMaster
15/07/31 12:20:18 INFO DiskBlockManager: Created local directory at /tmp/spark-379bb3c3-5275-499c-8b89-fa36eb931e/blockmgr-f7e60b37-c8c3-4f4f-bdec-2af10620c1e3
15/07/31 12:20:18 INFO MemoryStore: MemoryStore started with capacity 10.4 GB
15/07/31 12:20:19 INFO HttpFileServer: HTTP file server directory is /tmp/spark-c81a992b-d9d3-4751-8f2e-05ca644cb329/httpd-b9f5f0c8-0f7c-434c-aed4-20f27b9b3731
15/07/31 12:20:19 INFO HttpServer: Starting HTTP Server
15/07/31 12:20:19 INFO Server: jetty-8.y.z-SNAPSHOT
15/07/31 12:20:19 INFO AbstractConnector: Started SocketConnector@0.0.0.0:43664
15/07/31 12:20:19 INFO Util: Successfully started service 'HTTP file server' on port 43664.
15/07/31 12:20:19 INFO SparkEnv: Registering OutputCommitCoordinator
15/07/31 12:20:19 INFO Server: jetty-8.y.z-SNAPSHOT
15/07/31 12:20:19 INFO AbstractConnector: Started SelectChannelConnector@0.0.0.0:4040
15/07/31 12:20:19 INFO Util: Successfully started service 'SparkUI' on port 4040.
16/07/31 12:20:19 INFO AppClient$ClientActor: Connecting to master akka.tcp://sparkMaster@head:7077/user/Master...
15/07/31 12:20:19 INFO SparkDeploy:SchedulerBackend: Connected to Spark cluster with app ID
Background

System Event Log
**System Event Log**

*Exists practically on every computer system!*
Background

System Event Log

Exists practically on every computer system!

Automatic Analysis?
Background

System Event Log

Strucuted Data
Message/Event type
Log key

printf("Started service %s on port %d", x, y);

Anomaly Detection
Background

System Event Log

12:20:17 INFO SparkContext: Running Sp
12:20:10 WARN NativeCodeLoader: Unable to load classes where applicable
12:20:16 INFO SecurityManager: Changing context dir to
12:20:10 INFO SecurityManager: SecurityManager: Securi
permissions: set(zhoulang); users with
terminating remote
12:20:18 INFO REMoting: Start remoting start
12:20:18 INFO SparkEnv: Registering BlockManager: Create
12:20:18 INFO SparkEnv: Registering BlockManager: Create
12:20:18 INFO SparkEnv: Registering BlockManager: Create
12:20:18 INFO SparkEnv: Registering BlockManager: Create

L O G A N A L Y S I S

Strucuted Data
Message/Event type
Log key
……
printf("Started service %s on port %d", x, y);

Anomaly Detection

LOG ANALYSIS
Background

System Event Log

Strucuted Data
Message/Event type
Log key
......
printf(“Started service %s on port %d”, x, y);

Anomaly Detection

LOG ANALYSIS

- Message count vector:
  Xu’SOSP09, Lou’ATC10, Lin’ICSE16, etc.
Background

System Event Log

Strucuted Data
Message/Event type
Log key
......
printf(“\textit{Started service} \%s on port \%d”, x, y);

Anomaly Detection

LOG ANALYSIS

- Message count vector:
  Xu’SOSP09, Lou’ATC10, Lin’ICSE16, etc.

- Build workflow model:
  Lou’KDD10, Beschastnikh’ICSE14,
  Yu’ASPLOS16, etc.
Background

System Event Log

Structured Data
- Message/Event type
- Log key

......

printf("Started service %s on port %d", x, y);

LOG PARSING

Anomaly Detection
Background

System Event Log

Strucuted Data
Message/Event type
Log key

printf("Started service %s on port %d", x, y);

Anomaly Detection

LOG PARSING

- Use source code as template to parse logs:
  Xu’SOSP09
Background

System Event Log

Strucuted Data
Message/Event type
Log key
……
printf(“Started service %s on port %d”, x, y);

LOG PARSING

- Use source code as template to parse logs:
  - Xu’SOSP09
  - Problem: What if we don’t have source code?
Background

System Event Log

LOG PARSING

Strucuted Data
Message/Event type
Log key
......
printf("Started service \%s on port \%d", x, y);

Anomaly Detection

Use source code as template to parse logs:
Xu’SOSP09
Problem: What if we don’t have source code?

Directly parse from raw system logs:
Makanju’KDD09, Fu’ICDM09, Tang’ICDM10, Tang’CIKM11, etc.
Background

System Event Log

Strucuted Data

Message/Event type
Log key
......
printf(“Started service %s on port %d”, x, y);

LOG PARSING

Use source code as template to parse logs:
Xu’SOSP09
Problem: What if we don’t have source code?

DIRECTLY PARSING

Directly parse from raw system logs:
Makanju’KDD09, Fu’ICDM09, Tang’ICDM10, Tang’CIKM11, etc.
Problem: Offline batched processing, some very slow.
Our approach

**Spell**, a structured **Streaming Parser** for **Event Logs** using an **LCS** (longest common subsequence) based approach.
Our approach

**Spell**, a structured **Streaming Parser for Event Logs** using an **LCS** (longest common subsequence) based approach.

**Example:**

**Two log entries:**

- Temperature (41C) exceeds warning threshold
- Temperature (42C, 43C) exceeds warning threshold
Our approach

**Spell**, a structured **Streaming Parser for Event Logs** using an **LCS** (longest common subsequence) based approach.

Example:

Two log entries:

- Temperature (41C) exceeds warning threshold
- Temperature (42C, 43C) exceeds warning threshold

**LCS:**

- Temperature * exceeds warning threshold
Our approach

**Spell**, a structured Streaming Parser for Event Logs using an LCS (longest common subsequence) based approach.

Example:

Two log entries:
- Temperature (41\degree C) exceeds warning threshold
- Temperature (42\degree C, 43\degree C) exceeds warning threshold

LCS:
- Temperature * exceeds warning threshold

Naturally a message type!
- `printf("Temperature %s exceeds warning threshold")`
SPELL – Basic workflow

Add new log entry into LCSMap in a streaming fashion, update existing message type if $\text{length}(\text{LCS}) > 0.5 \times \text{length(new log entry)}$
SPELL – Basic workflow

new log entry: *Temperature (41°C) exceeds warning threshold*
SPELL – Basic workflow

new log entry:

```
LCSObject

LCSseq: Temperature (41C) exceeds warning threshold
linelds: {0}
paramPos: {empty}
```

LCSMap
SPELL – Basic workflow

new log entry: Temperature (43C) exceeds warning threshold

LCSObject

\[
\begin{align*}
\text{LCSseq}: & \quad \text{Temperature (41C) exceeds warning threshold} \\
\text{lineIds}: & \quad \{0\} \\
\text{paramPos}: & \quad \{\text{empty}\}
\end{align*}
\]
SPELL – Basic workflow

new log entry:

```
LCSObject{
LCSseq: Temperature * exceeds warning threshold
lineIds: {0, 1}
paramPos: {1}
}
```
new log entry: Command has completed successfully

LCSObject

\[
\begin{align*}
\text{LCSseq: } & \text{Temperature }* \text{ exceeds warning threshold} \\
\text{linelds: } & \{0, 1\} \\
\text{paramPos: } & \{1\}
\end{align*}
\]
SPPELL – Basic workflow

new log entry:

\[
\begin{align*}
\text{LCSObject} & \quad \text{LCSseq: } Temperature \ast exceeds \text{ warning threshold} \\
& \quad \text{linelds: } \{0, 1\} \\
& \quad \text{paramPos: } \{1\}
\end{align*}
\]

\[
\begin{align*}
\text{LCSObject} & \quad \text{LCSseq: } Command \ has \ completed \ successfully \\
& \quad \text{linelds: } \{2\} \\
& \quad \text{paramPos: } \{\text{empty}\}
\end{align*}
\]
SPELL – Basic workflow

new log entry: ......

LCSObject

\[
\text{LCSseq: Temperature } * \text{ exceeds warning threshold}
\]
\[
\text{linelds: } \{0, 1\}
\]
\[
\text{paramPos: } \{1\}
\]

LCSObject

\[
\text{LCSseq: Command has completed successfully}
\]
\[
\text{linelds: } \{2\}
\]
\[
\text{paramPos: } \{\text{empty}\}
\]

......

LCSMap
To compute LCS of two log entries, each one has $O(n)$ length:
To compute LCS of two log entries, each one has $O(n)$ length:

**Naïve way:** Dynamic Programing
SPELL – Improvement on efficiency

To compute LCS of two log entries, each one has $O(n)$ length:

Naïve way: Dynamic Programing

Time complexity:
- To compare a log entry with an existing message type: $O(n^2)$
- To compare a new log entry with $O(m)$ existing message types: $O(mn^2)$
SPELL – Improvement on efficiency

To compute LCS of two log entries, each one has $O(n)$ length:

**Naïve way:** Dynamic Programing

**Time complexity:**
- To compare a log entry with an existing message type: $O(n^2)$
- To compare a new log entry with $O(m)$ existing message types: $O(mn^2)$

*Can we do better?*
SPELL – Improvement on efficiency

Observation.

For a complex system,
number of log entries: millions
number of message types: hundreds
Observation.

For a complex system,
number of log entries: millions
number of message types: hundreds

For example:

Blue Gene/L log:
4,457,719 log entries, 394 message types

Hadoop log used in Xu’SOSP09:
11,197,705 log entries, only 29 message types
SPELL – Improvement on efficiency

Observation.

For a complex system,
number of log entries: millions
number of message types: hundreds

For example:

Blue Gene/L log:
4,457,719 log entries, 394 message types

Hadoop log used in Xu’SOSP09:
11,197,705 log entries, only 29 message types

For a majority of new log entries, their message types already exist in LCSMap!
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

Existing message types:
A B C
A C D
A D
E F
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

Existing message types:
A B C
A C D
A D
E F

Diagram:
- ROOT
  - A
    - B
    - C
  - E
    - D
    - F
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

New log entry: A B P C
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

New log entry: $A \ B \ P \ C$
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

New log entry: A B P C
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

New log entry: A B P C

Parameter: B C D E F
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

New log entry: A B P C

Parameter:
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

Time Complexity: \( O(n) \) for each log entry
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

Problem:
New log entry: $D A P B C$
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

Problem:
New log entry: D A P B C
Matches D A

Diagram:

- ROOT
  - A
    - B
    - C
  - D
  - E
  - F
  - A
SPELL – Improvement on efficiency

Improvement 1: Prefix Tree

Problem:
New log entry: D A P B C
Matches D A
Should be: A B C
# SPELL – Improvement on efficiency

## Improvement 2: Inverted Index

**Existing message types:**

- $A, B, C$
- $A, E, F$
- $D, A$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$(1, 1)$</td>
<td>$(2, 1)$</td>
<td>$(3, 2)$</td>
</tr>
<tr>
<td>B</td>
<td>$(1, 2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>$(1, 3)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>$(3, 1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>$(2, 2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>$(2, 3)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**SPELL – Improvement on efficiency**

**Improvement 2: Inverted Index**

**New log entry:**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>A</td>
<td>(1, 1) (2, 1) (3, 2)</td>
</tr>
<tr>
<td>$A$</td>
<td>B</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>$B$</td>
<td>C</td>
<td>(1, 3)</td>
</tr>
<tr>
<td>$P$</td>
<td>D</td>
<td>(3, 1)</td>
</tr>
<tr>
<td>$C$</td>
<td>E</td>
<td>(2, 2)</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>(2, 3)</td>
</tr>
</tbody>
</table>
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
<td>(3, 2)</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>(3, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>(2, 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(2, 3)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

D —— 2 —— A —— (1, 1) —— (2, 1) —— (3, 2) —— B —— (1, 2) —— C —— (1, 3) —— D —— (3, 1) —— P —— E —— (2, 2) —— F —— (2, 3)
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>(3, 1)</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>(2, 2)</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(2, 3)</td>
<td></td>
</tr>
</tbody>
</table>
Improvement 2: Inverted Index

New log entry:

No match, parameter position
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

No match, parameter position

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 1)</td>
<td>(1, 2)</td>
<td>(1, 3)</td>
<td>(3, 1)</td>
</tr>
<tr>
<td>2</td>
<td>(2, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E (2, 2)
F (2, 3)
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 1)</td>
<td>(1, 2)</td>
<td>(1, 3)</td>
<td>(3, 1)</td>
</tr>
<tr>
<td>2</td>
<td>(2, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>(3, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>(2, 2)</td>
<td></td>
</tr>
</tbody>
</table>

No match, parameter position
### SPELL – Improvement on efficiency

**Improvement 2: Inverted Index**

**Time complexity:** $O(cn)$ for each log entry, $c < m$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>(3, 1)</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>(2, 2)</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(2, 3)</td>
<td></td>
</tr>
</tbody>
</table>
**SPELL – Improvement on efficiency**

**Improvement 2: Inverted Index**

**Time complexity:** $O(cn)$
for each log entry, $c < m$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>(3, 1)</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>(2, 2)</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(2, 3)</td>
<td></td>
</tr>
</tbody>
</table>

For remaining log entries, compare it with each message type using simple DP.
We could be wrong.
More heuristics to adjust the result.

Example:
boot (command 2359) Error: Console-Busy Port already in use
wait (command 3964) Error: Console-Busy Port already in use

Initial message type:
* (command *) Error: Console-Busy Port already in use
SPELL – Improvement on effectiveness

We could be wrong.
More heuristics to adjust the result.

Example:
boot (command 2359) Error: Console-Busy Port already in use
wait (command 3964) Error: Console-Busy Port already in use

Initial message type:
* (command *) Error: Console-Busy Port already in use

Solution: Split heuristic
If a parameter position has very few unique tokens:
boot (command *) Error: Console-Busy Port already in use
wait (command *) Error: Console-Busy Port already in use
We could be wrong.
More heuristics to adjust the result.

Example:
Fan speeds ( 3552 3534 3375 4354 3515 3479 )
Fan speeds ( 3552 3534 3375 4299 3515 3479 )
Fan speeds ( 3552 3552 3391 4245 3515 3497 )
Fan speeds ( 3534 3534 3375 4245 3497 3479 )
Fan speeds ( 3534 3534 3375 4066 3497 3479 )

Initial message type:
Fan speeds ( * 3552 * 3515 * )
Fan speeds ( 3534 3534 3375 * 3497 3479 )
SPEL – Improvement on effectiveness

We could be wrong.
More heuristics to adjust the result.

Example:
Fan speeds ( 3552 3534 3375 4354 3515 3479 )
Fan speeds ( 3552 3534 3375 4299 3515 3479 )
Fan speeds ( 3552 3552 3391 4245 3515 3497 )
Fan speeds ( 3534 3534 3375 4245 3497 3479 )
Fan speeds ( 3534 3534 3375 4066 3497 3479 )

Initial message type:
Fan speeds ( * 3552 * 3515 * )
Fan speeds ( 3534 3534 3375 * 3497 3479 )

Solution: Merge heuristic
Merge similar message types together:
Fan speeds: ( * )
Evaluation

Methods to compare:

- IPLoM (Makanju’KDD09):
  Partition log file using 3-step heuristics (log entry length, etc.)
- CLP (Fu’ICDM09):
  Cluster similar logs together based on weighted edit distance

Log dataset:

<table>
<thead>
<tr>
<th>Log type</th>
<th>Count</th>
<th>Message type ground truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Alamos HPC log</td>
<td>433,490</td>
<td>Available online</td>
</tr>
<tr>
<td>BlueGene/L log</td>
<td>4,747,963</td>
<td>Available online</td>
</tr>
<tr>
<td>Openstack Cloud log</td>
<td>87,519</td>
<td>Manually parsed from source code</td>
</tr>
</tbody>
</table>
# Evaluation - Efficiency

## Number (Percentage) of log entries returned by each step

<table>
<thead>
<tr>
<th>Method</th>
<th>Los Alamos HPC log</th>
<th>BlueGene/L log</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix tree</td>
<td>397,412 (91.68%)</td>
<td>4,457,719 (93.89%)</td>
</tr>
<tr>
<td>inverted index</td>
<td>35,691 (8.23%)</td>
<td>288,254 (6.07%)</td>
</tr>
<tr>
<td>naive LCS</td>
<td>387 (0.09%)</td>
<td>1,990 (0.042%)</td>
</tr>
</tbody>
</table>

## Amortized cost of each message type lookup step in Spell

<table>
<thead>
<tr>
<th>Method</th>
<th>Los Alamos HPC log</th>
<th>BlueGene/L log</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix tree (ms)</td>
<td>0.006</td>
<td>0.011</td>
</tr>
<tr>
<td>inverted index (ms)</td>
<td>0.015</td>
<td>0.077</td>
</tr>
<tr>
<td>naive LCS (ms)</td>
<td>0.175</td>
<td>0.580</td>
</tr>
</tbody>
</table>
Evaluation - Efficiency

Openstack:

<table>
<thead>
<tr>
<th>Method</th>
<th>Los Alamos</th>
<th>Blue Gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLP (fixed threshold)</td>
<td>21053.22</td>
<td>21053.22</td>
</tr>
<tr>
<td>IPLoM</td>
<td>10.25</td>
<td>10.25</td>
</tr>
<tr>
<td>Spell</td>
<td>9.96</td>
<td>9.96</td>
</tr>
<tr>
<td>Spell (with split)</td>
<td>10.27</td>
<td>10.27</td>
</tr>
<tr>
<td>Spell (with split and merge)</td>
<td>10.30</td>
<td>10.30</td>
</tr>
</tbody>
</table>
Evaluation - Effectiveness

F-measure

Accuracy

- IPlOM
- Spell
- Spell (with split)
- Spell (with split and merge)
- CLP (fixed threshold)
Evaluation - Effectiveness

F-measure

Accuracy

log size (×10^5, Blue Gene)

log size (×10^5, Blue Gene)

IPLoM  Spell  Spell (with split)  Spell (with split and merge)
Thank you!
## Evaluation - Effectiveness

### Comparison of Spell with and without pre-filter

<table>
<thead>
<tr>
<th>Spell</th>
<th>With pre-filtering</th>
<th>Los Alamos HPC log</th>
<th>BlueGene/L log</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True</td>
<td>False</td>
<td>True message type found</td>
</tr>
<tr>
<td>basic</td>
<td>False</td>
<td>0.822786</td>
<td>0.822786</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td>0.822786</td>
<td>0.822786</td>
</tr>
<tr>
<td>with split</td>
<td>False</td>
<td>55</td>
<td>0.918985</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td>0.918985</td>
<td>0.918985</td>
</tr>
<tr>
<td>with split and merge</td>
<td>False</td>
<td>74</td>
<td>0.969210</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td>0.969210</td>
<td>0.969210</td>
</tr>
</tbody>
</table>
Evaluation

Methods to compare:

IPLoM (Makanju’KDD09):
  Partition log file using 3-step heuristics (log entry length, etc.)
CLP (Fu’ICDM09)
  Cluster similar logs together based on weighted edit distance

Log dataset:

<table>
<thead>
<tr>
<th>Log type</th>
<th>Source</th>
<th>Count</th>
<th>Message type ground truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Alamos HPC log</td>
<td>Available online</td>
<td>433,490</td>
<td>Available online</td>
</tr>
<tr>
<td>BlueGene/L log</td>
<td>Available online</td>
<td>4,747,963</td>
<td>Available online</td>
</tr>
<tr>
<td>Openstack Cloud log</td>
<td>Generated using CloudLab</td>
<td>87,519</td>
<td>Manually parsed from source code</td>
</tr>
</tbody>
</table>
Our approach

*Spell*, a structured *Streaming Parser* for *Event Logs* using an *LCS* (longest common subsequence) based approach.

**LCS of two sequences:**

The longest subsequence common to both sequences.
Our approach

*Spell*, a structured **Streaming Parser** for **Event Logs** using an **LCS** (longest common subsequence) based approach.

**LCS of two sequences:**

The longest subsequence common to both sequences.

E.g. LCS of:

1, 3, 5, 7, 9
1, 5, 7, 10

equals:

1, 5, 7
## Evaluation - Effectiveness

Effectiveness measures on Openstack Log

<table>
<thead>
<tr>
<th>methods</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLP (fixed threshold)</td>
<td>0.00015</td>
<td>0.44444</td>
<td>0.00030</td>
<td>0.36874</td>
</tr>
<tr>
<td>IPLoM</td>
<td>0.00011</td>
<td>0.16667</td>
<td>0.00021</td>
<td>0.06587</td>
</tr>
<tr>
<td>Spell</td>
<td>0.66667</td>
<td>0.77778</td>
<td>0.71795</td>
<td>0.99383</td>
</tr>
<tr>
<td>Spell (with split)</td>
<td>0.57692</td>
<td>0.83333</td>
<td>0.68182</td>
<td>0.99574</td>
</tr>
<tr>
<td>Spell (with split and merge)</td>
<td>0.57692</td>
<td>0.83333</td>
<td>0.68182</td>
<td>0.99574</td>
</tr>
</tbody>
</table>
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1) (2, 1) (3, 1)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>C</td>
<td>(1, 3) (2, 2)</td>
</tr>
<tr>
<td>D</td>
<td>(2, 3) (3, 2)</td>
</tr>
<tr>
<td>E</td>
<td>(4, 1)</td>
</tr>
<tr>
<td>F</td>
<td>(4, 2)</td>
</tr>
</tbody>
</table>
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>D</td>
<td>(2, 3)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>E</td>
<td>(4, 1)</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(4, 2)</td>
<td></td>
</tr>
</tbody>
</table>
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

No match, parameter position

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>D</td>
<td>(2, 3)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>E</td>
<td>(4, 1)</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(4, 2)</td>
<td></td>
</tr>
</tbody>
</table>
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

New log entry:

No match, parameter position

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>(1, 1)</th>
<th>(2, 1)</th>
<th>(3, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td>(2, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>(2, 3)</td>
<td>(3, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>(4, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(4, 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# SPELL – Improvement on efficiency

## Improvement 2: Inverted Index

New log entry:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

No match, parameter position

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>(1, 2)</td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>(2, 3)</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SPELL – Improvement on efficiency

Improvement 2: Inverted Index

Time complexity: $O(cn)$ for each log entry, $c < m$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(1, 3)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>D</td>
<td>(2, 3)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>E</td>
<td>(4, 1)</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>(4, 2)</td>
<td></td>
</tr>
</tbody>
</table>

For remaining log entries, compare it with each message type using simple DP.