Pattern detection in server log messages with support for multiple log management systems

Louise Harding

Master 1 Internship with the CC-IN2P3

Supervised by: Fabien Wernli

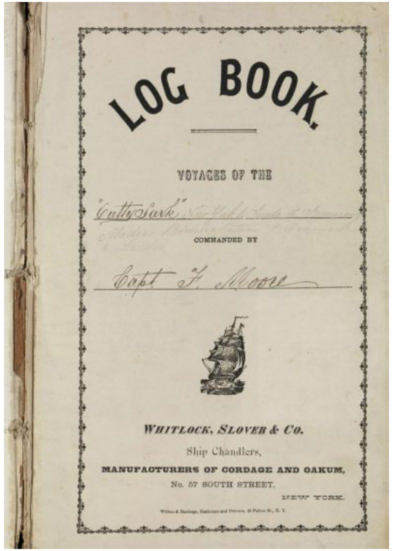
Plan

- Introduction
- Problem
- Goals
- Workflow
- Data Analysis
- SEQUENCE
    - How it works - Scanner, Analyser, Parser
    - Extensions for this project
- Results
- Limitations
- Next steps - Machine Learning/Anomaly detection
- Conclusion

Introduction

Why is logging important?

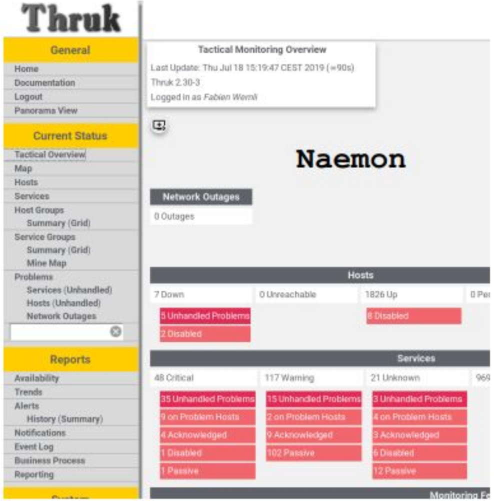
- Supplies **information** continuously on the **activities** of hardware, software and other equipment.
- Often **first**, if not only, **alert of a problem**.
- Critical for **issue diagnosis**, both **real time** and **post mortem.**
- **State: Current** and **historical, normal** and **abnormal**
- **Audits** and **security**



2 datacenters
2000 servers
network equipment
applications
operating systems...

100 million
logs per day

COLOSS
syslog-ng
patterndb



Naemon

Tactical Monitoring Overview

Last Update: Thu Jul 18 15:19:47 CEST 2019 (+90s)

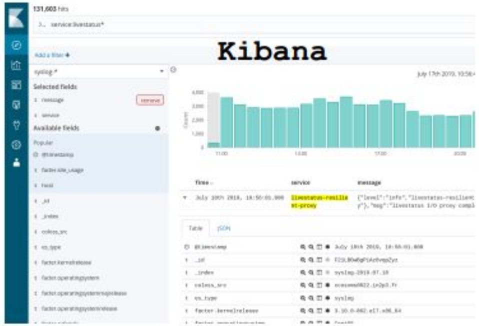
Thruk 2.30-3

Logged in as *Fabien Wernli*

0 Outages

7 Down	0 Unreachable	1826 Up	0 Per
5 Unhandled Problems 2 Disabled		8 Disabled	

48 Critical	117 Warning	21 Unknown	969
35 Unhandled Problems 9 on Problem Hosts 4 Acknowledged 1 Disabled 1 Passive	15 Unhandled Problems 2 on Problem Hosts 9 Acknowledged 102 Passive	3 Unhandled Problems 4 on Problem Hosts 3 Acknowledged 6 Disabled 12 Passive	



Kibana

131,603 hits

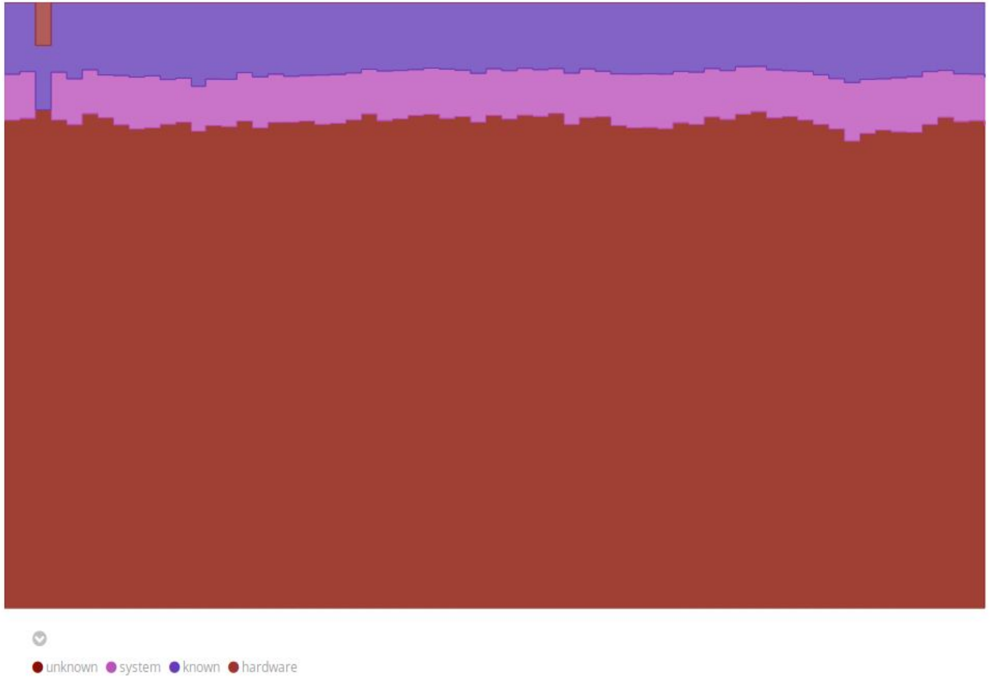
service:livestatus*

Time	service	message
July 18th 2019, 10:56:01.000	livestatus-resilient-proxy	{"level":"info","livestatus-resilient-proxy"},"msg":"livestatus I/O proxy compl

```
@timestamp             July 18th 2019, 10:56:01.000
_id                    F2ILB0wBgP1Az8vqpZyt
_index                 syslog-2019.07.18
coloss_src             ccosvms0022.in2p3.fr
es_type                syslog
facter.kernelrelease   3.10.0-862.el7.x86_64
```

Problem

- approximately **100 million** system logs **every day**.
- Uses a **pattern database,** for the **analysis**, whose **patterns** are **created by hand**.
- **Issues: Scalability and maintenance.**
- Events **constantly changing.**
- **Approx 75-80% unknown.**



Goals

- **Implement** a **pattern recognition algorithm** into the **message flow,** to **assist or automate** the **manual** pattern creation.

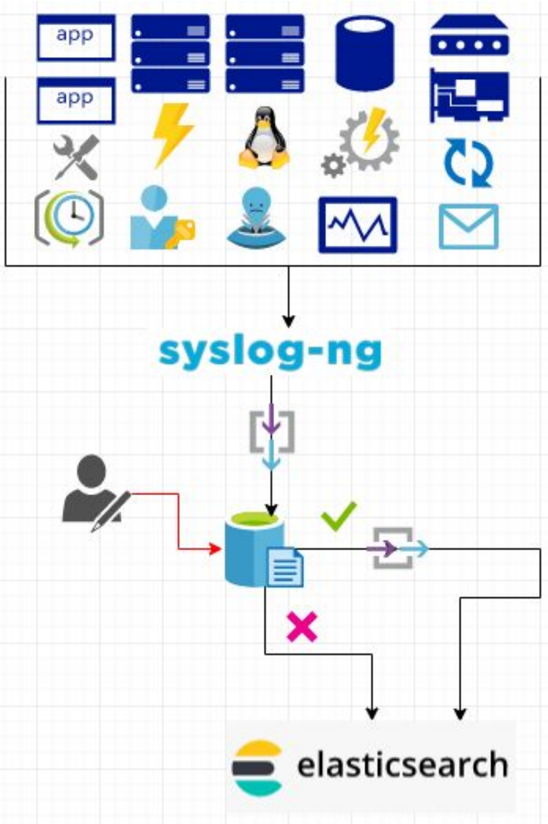
- **By adapting** the **SEQUENCE module** implemented in **Go language** for the **CC-IN2P3's data and workflow.**

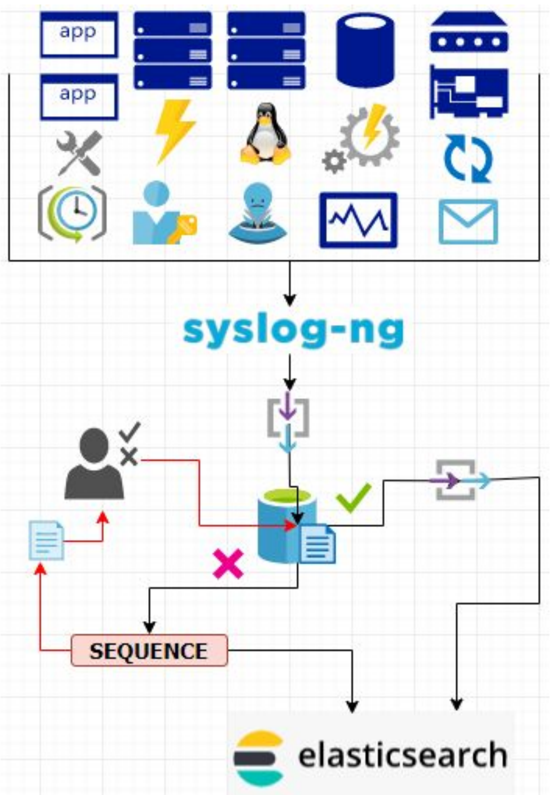
- Overall **goal** is to have **90%** or more of the messages **known** in production.

- Ideally return the modified software **back to the open source community** with support for **Syslog's patternDB** and **Logstash's Grok** pattern **parsers**.

Workflow and other considerations

- Volume
- Variety
- Constant change
- No preprocessing





Data analysis

Log messages take **many forms**:

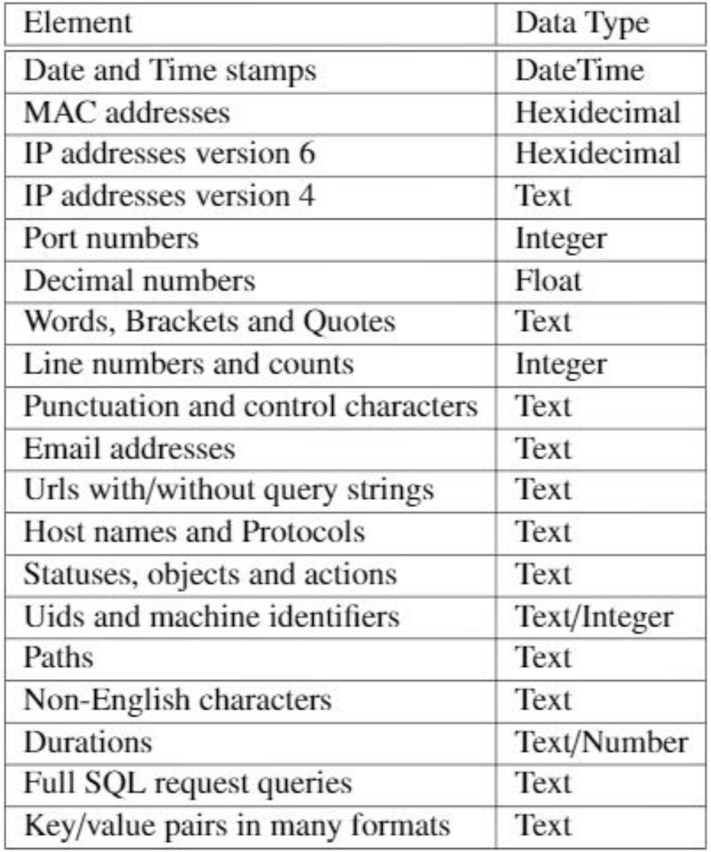
Length: 2-3 words to > ½ page of text.

No strict rules for construction, order or contents.

Not confined to one language.

Text and/or **JSON** format.

Elements that **do follow rules** that can be **extracted**.



Element	Data Type
Date and Time stamps	DateTime
MAC addresses	Hexidecimal
IP addresses version 6	Hexidecimal
IP addresses version 4	Text
Port numbers	Integer
Decimal numbers	Float
Words, Brackets and Quotes	Text
Line numbers and counts	Integer
Punctuation and control characters	Text
Email addresses	Text
Urls with/without query strings	Text
Host names and Protocols	Text
Statuses, objects and actions	Text
Uids and machine identifiers	Text/Integer
Paths	Text
Non-English characters	Text
Durations	Text/Number
Full SQL request queries	Text
Key/value pairs in many formats	Text

Examples

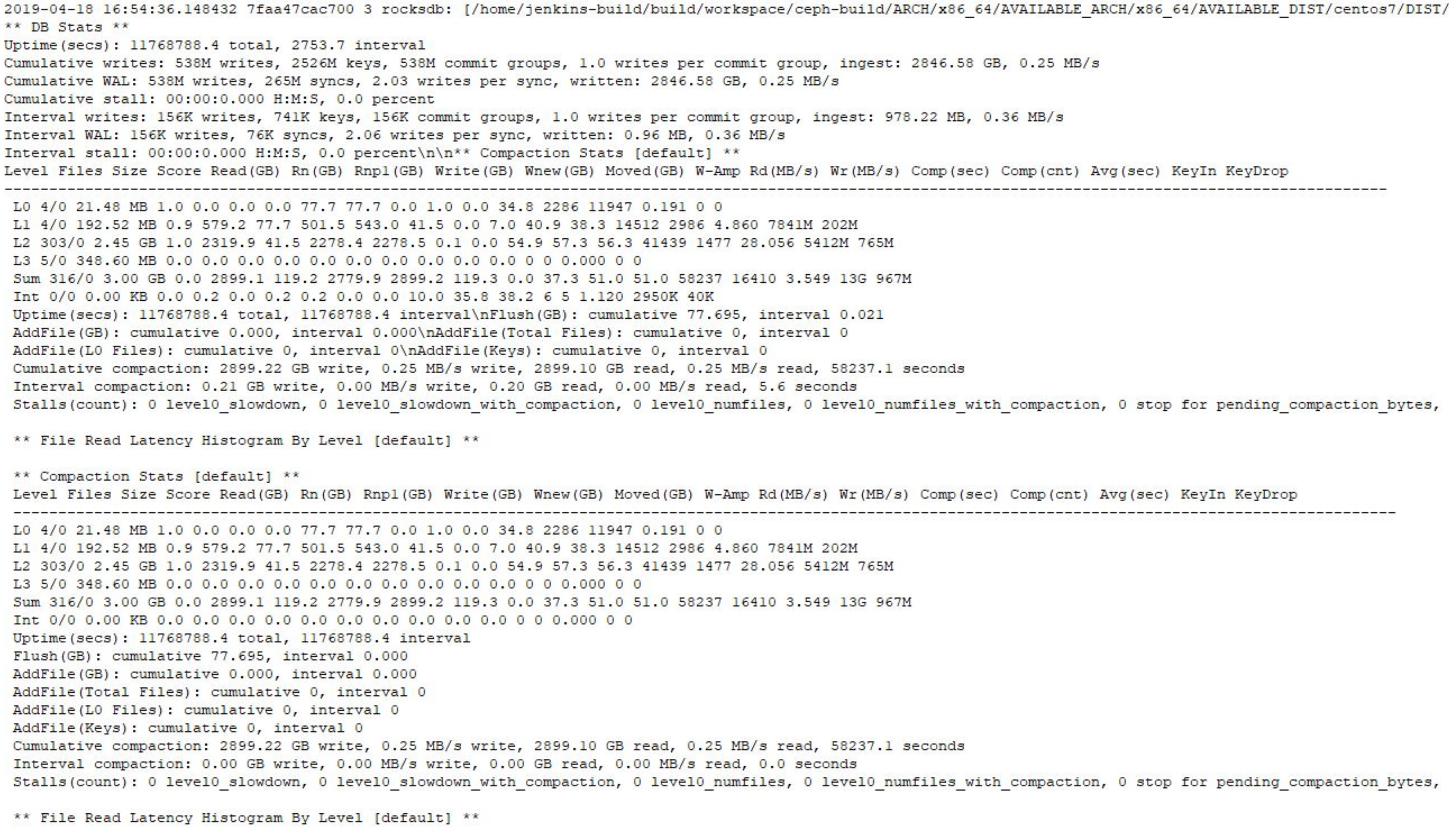
```
{"level":"debug","msg":"unregistering reader", "reader-id":"8c417dec-e854-469d-9744-c46f5bd14b2b",
"time":"2019-04-09T09:08:23+02:00"}
```

```
warning: maildrop/10E66A7: error writing 1A648332: queue file write error
```

```
lcas_userban.mod-plugin_confirm_authorization(): checking banned users in /etc/lcas/ban_users.db
```

```
Callout to "LCMAPS" returned local user (service file): "ops008
```

```
134.158.172.113:46408 [18/Apr/2019:16:43:43.255] frontend puppetserver/ccpuppet03 1/0/183322 7479 cD
41/41/41/12/0 0/0
```



```
2019-04-18 16:54:36.148432 7faa47cac700 3 rocksdb: [/home/jenkins-build/build/workspace/ceph-build/ARCH/x86_64/AVAILABLE_ARCH/x86_64/AVAILABLE_DIST/centos7/DIST/
** DB Stats **
Uptime(secs): 11768788.4 total, 2753.7 interval
Cumulative writes: 538M writes, 2526M keys, 538M commit groups, 1.0 writes per commit group, ingest: 2846.58 GB, 0.25 MB/s
Cumulative WAL: 538M writes, 265M syncs, 2.03 writes per sync, written: 2846.58 GB, 0.25 MB/s
Cumulative stall: 00:00:0.000 H:M:S, 0.0 percent
Interval writes: 156K writes, 741K keys, 156K commit groups, 1.0 writes per commit group, ingest: 978.22 MB, 0.36 MB/s
Interval WAL: 156K writes, 76K syncs, 2.06 writes per sync, written: 0.96 MB, 0.36 MB/s
Interval stall: 00:00:0.000 H:M:S, 0.0 percent\n\n** Compaction Stats [default] **
Level Files Size Score Read(GB) Rn(GB) Rnp1(GB) Write(GB) Wnew(GB) Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) Comp(cnt) Avg(sec) KeyIn KeyDrop
------------------------------------------------------------------------------------------------------------------------------------------------------------
 L0 4/0 21.48 MB 1.0 0.0 0.0 0.0 77.7 77.7 0.0 1.0 0.0 34.8 2286 11947 0.191 0 0
 L1 4/0 192.52 MB 0.9 579.2 77.7 501.5 543.0 41.5 0.0 7.0 40.9 38.3 14512 2986 4.860 7841M 202M
 L2 303/0 2.45 GB 1.0 2319.9 41.5 2278.4 2278.5 0.1 0.0 54.9 57.3 56.3 41439 1477 28.056 5412M 765M
 L3 5/0 348.60 MB 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0 0.000 0 0
 Sum 316/0 3.00 GB 0.0 2899.1 119.2 2779.9 2899.2 119.3 0.0 37.3 51.0 51.0 58237 16410 3.549 13G 967M
 Int 0/0 0.00 KB 0.0 0.2 0.0 0.2 0.2 0.0 0.0 10.0 35.8 38.2 6 5 1.120 2950K 40K
 Uptime(secs): 11768788.4 total, 11768788.4 interval\nFlush(GB): cumulative 77.695, interval 0.021
 AddFile(GB): cumulative 0.000, interval 0.000\nAddFile(Total Files): cumulative 0, interval 0
 AddFile(L0 Files): cumulative 0, interval 0\nAddFile(Keys): cumulative 0, interval 0
 Cumulative compaction: 2899.22 GB write, 0.25 MB/s write, 2899.10 GB read, 0.25 MB/s read, 58237.1 seconds
 Interval compaction: 0.21 GB write, 0.00 MB/s write, 0.20 GB read, 0.00 MB/s read, 5.6 seconds
 Stalls(count): 0 level0_slowdown, 0 level0_slowdown_with_compaction, 0 level0_numfiles, 0 level0_numfiles_with_compaction, 0 stop for pending_compaction_bytes,

 ** File Read Latency Histogram By Level [default] **

 ** Compaction Stats [default] **
 Level Files Size Score Read(GB) Rn(GB) Rnp1(GB) Write(GB) Wnew(GB) Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) Comp(cnt) Avg(sec) KeyIn KeyDrop
 ------------------------------------------------------------------------------------------------------------------------------------------------------------
 L0 4/0 21.48 MB 1.0 0.0 0.0 0.0 77.7 77.7 0.0 1.0 0.0 34.8 2286 11947 0.191 0 0
 L1 4/0 192.52 MB 0.9 579.2 77.7 501.5 543.0 41.5 0.0 7.0 40.9 38.3 14512 2986 4.860 7841M 202M
 L2 303/0 2.45 GB 1.0 2319.9 41.5 2278.4 2278.5 0.1 0.0 54.9 57.3 56.3 41439 1477 28.056 5412M 765M
 L3 5/0 348.60 MB 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0 0.000 0 0
 Sum 316/0 3.00 GB 0.0 2899.1 119.2 2779.9 2899.2 119.3 0.0 37.3 51.0 51.0 58237 16410 3.549 13G 967M
 Int 0/0 0.00 KB 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0 0.000 0 0
 Uptime(secs): 11768788.4 total, 11768788.4 interval
 Flush(GB): cumulative 77.695, interval 0.000
 AddFile(GB): cumulative 0.000, interval 0.000
 AddFile(Total Files): cumulative 0, interval 0
 AddFile(L0 Files): cumulative 0, interval 0
 AddFile(Keys): cumulative 0, interval 0
 Cumulative compaction: 2899.22 GB write, 0.25 MB/s write, 2899.10 GB read, 0.25 MB/s read, 58237.1 seconds
 Interval compaction: 0.00 GB write, 0.00 MB/s write, 0.00 GB read, 0.00 MB/s read, 0.0 seconds
 Stalls(count): 0 level0_slowdown, 0 level0_slowdown_with_compaction, 0 level0_numfiles, 0 level0_numfiles_with_compaction, 0 stop for pending_compaction_bytes,

 ** File Read Latency Histogram By Level [default] **
```

SEQUENCE

Open source module using Go Lang **written by Jian Zhen**

Tested on a **small range** of **common** log messages, but not the **variety** seen at CC

It consists of:

- **Scanner** - splits messages into pieces called tokens
- **Analyser** - compares the sets of tokens to find patterns
- **Parser** - tries to match new messages to already found patterns

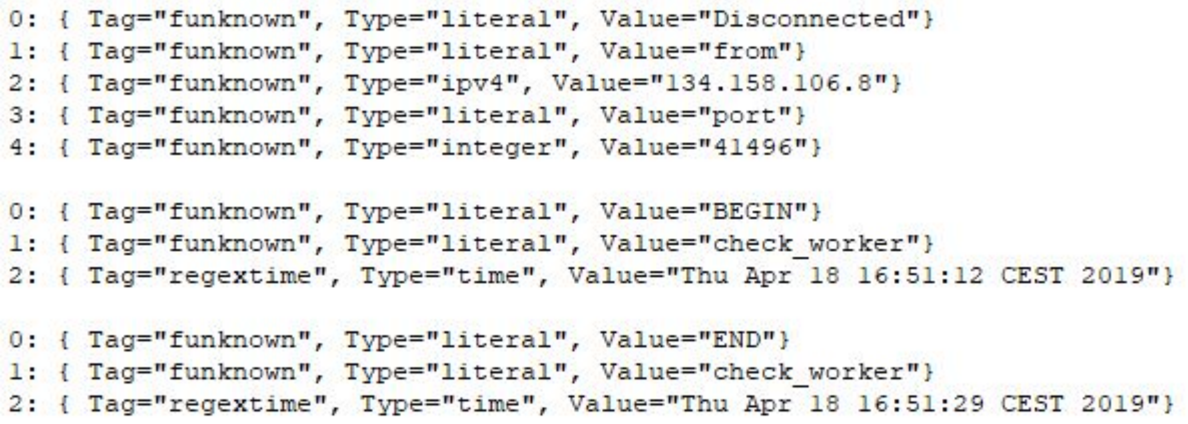
Scanner

- Breaks the **message into pieces: tokens**
- Uses **three separate processes**, one for **Hexidecimal** values, one for **Date/Time** formats, one for **everything else** to find the tokens.
- **Reads** each log message only once, **character by character** and passes each character **simultaneously to the three processes.**
- Each process **stops** when it **finds a valid value** or can't continue as it **hits something invalid.**
- **Fast**: > 200,000 msg/s

Scanner cont.

- **Token Types:**
  - Float
  - Integer
  - DateTime
  - IPv4, IPv6
  - Urls (http/https)
  - Literal
  - MAC address

```
Disconnected from 134.158.106.8 port 41496
Disconnected from 127.0.0.1 port 49570
BEGIN check_worker Thu Apr 18 16:51:12 CEST 2019
END check_worker Thu Apr 18 16:51:29 CEST 2019
```

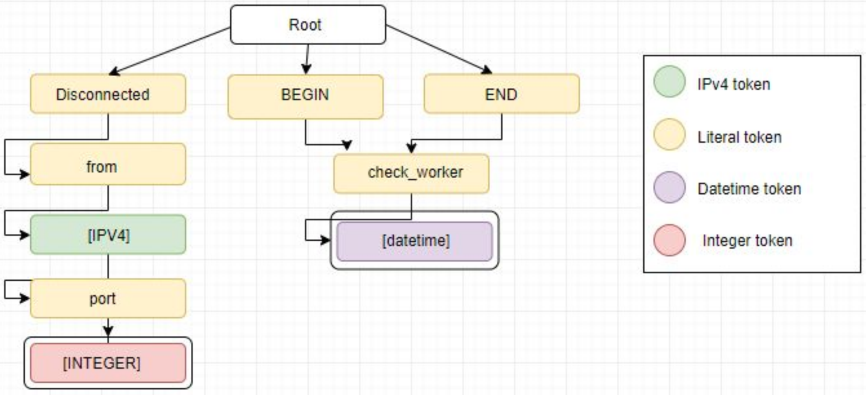


```
0: { Tag="funknown", Type="literal", Value="Disconnected"}
1: { Tag="funknown", Type="literal", Value="from"}
2: { Tag="funknown", Type="ipv4", Value="134.158.106.8"}
3: { Tag="funknown", Type="literal", Value="port"}
4: { Tag="funknown", Type="integer", Value="41496"}

0: { Tag="funknown", Type="literal", Value="BEGIN"}
1: { Tag="funknown", Type="literal", Value="check_worker"}
2: { Tag="regextime", Type="time", Value="Thu Apr 18 16:51:12 CEST 2019"}

0: { Tag="funknown", Type="literal", Value="END"}
1: { Tag="funknown", Type="literal", Value="check_worker"}
2: { Tag="regextime", Type="time", Value="Thu Apr 18 16:51:29 CEST 2019"}
```

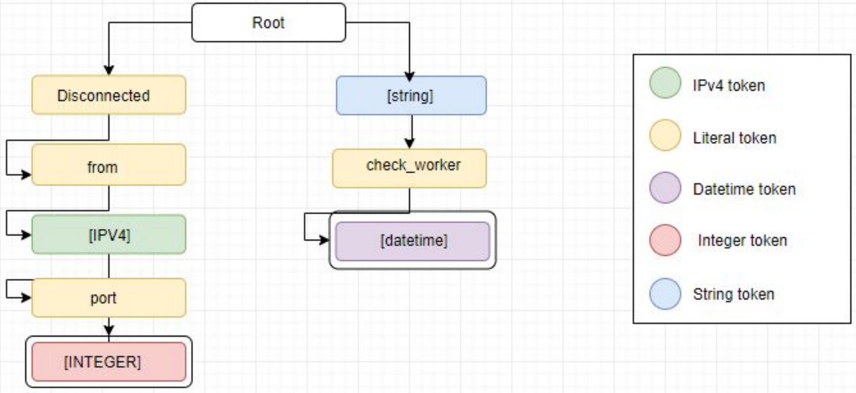
Analyser

1. Builds **a trie** from all the tokenised messages.

2. **Identifies and merges** the tokens of the **same type** at the **same level** with the **same parent and child** node.



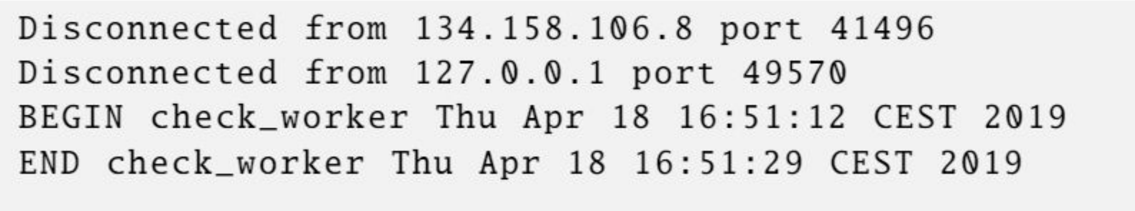
Example Analyser trie with three examples - before merge



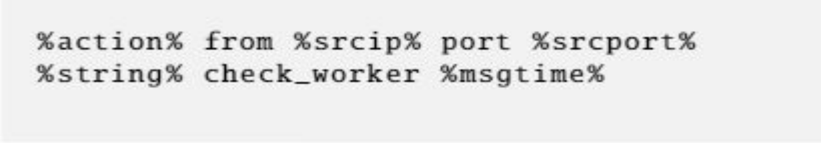
Example Analyser trie with three examples - after merge

Analyser cont.

1. Looks for **email addresses** and **hostnames** and tags them.
2. Uses **prekeys** like ‘from; and ‘to’ to tag source and destination variables.
3. Uses **keywords**, such as error, or file, to apply status or object tags for example.
4. Tries to find **IP and port combinations**.



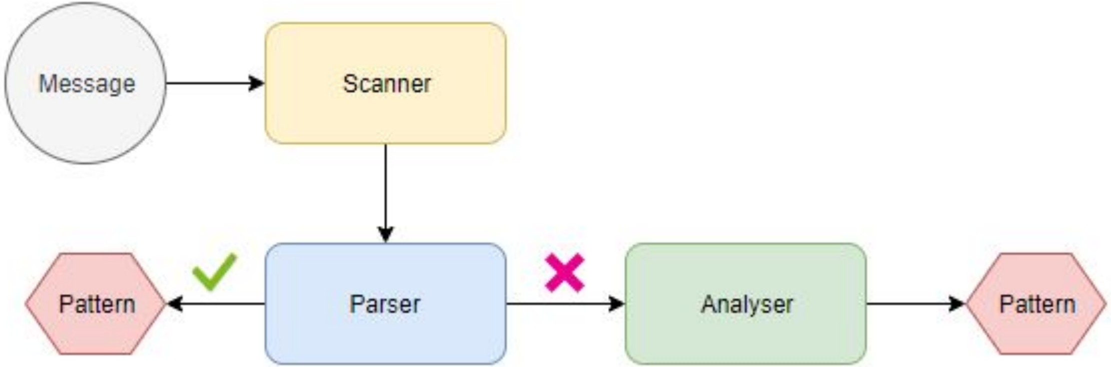
```
Disconnected from 134.158.106.8 port 41496
Disconnected from 127.0.0.1 port 49570
BEGIN check_worker Thu Apr 18 16:51:12 CEST 2019
END check_worker Thu Apr 18 16:51:29 CEST 2019
```



```
%action% from %srcip% port %srcport%
%string% check_worker %msgtime%
```

Parser

- Used for matching **new messages** to **existing known patterns.**
- If not matched **add to the analyser** for processing.



Extension of SEQUENCE

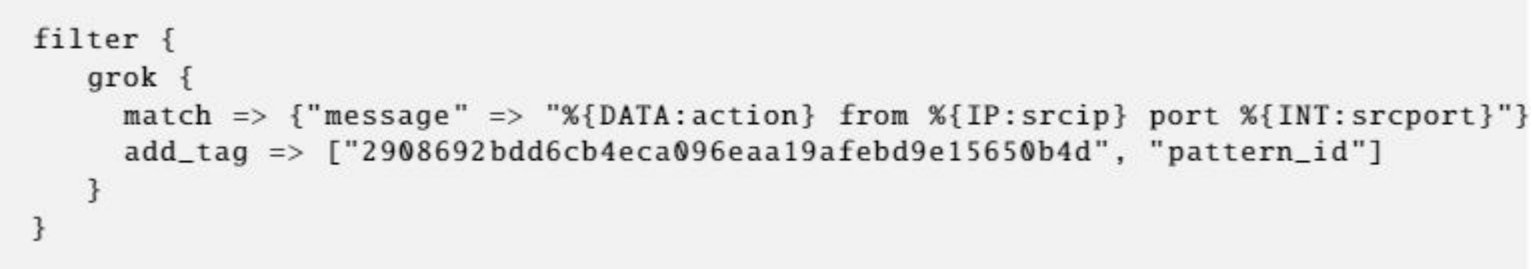
To output for custom parsers and to deal with the volume of log messages at CC-IN2P3, we needed to:

- **Added a database** so SEQUENCE could **run continuously.**
- Added functionality to handle **multi-line messages**
- **New approach** to handle the **volume of messages**.
- Create a **pattern ID that is reproducible** always for the same pattern.
- **Preserve examples with the patterns** for testing with patternDB.
- **Translate** SEQUENCE patterns to **use with patternDB/Grok parsers.**

SEQUENCE pattern and Logstash Grok example

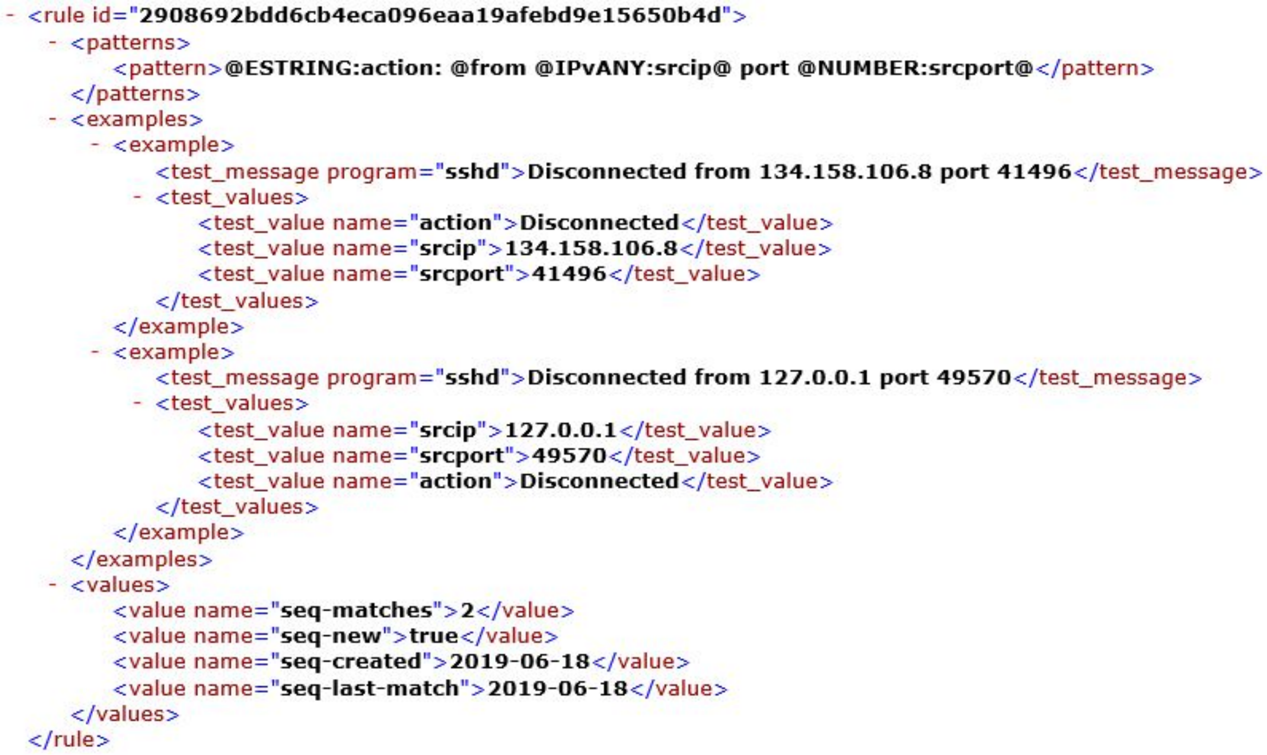
SEQUENCE: %action% from %srcip% port %srcport%

LOGSTASH:

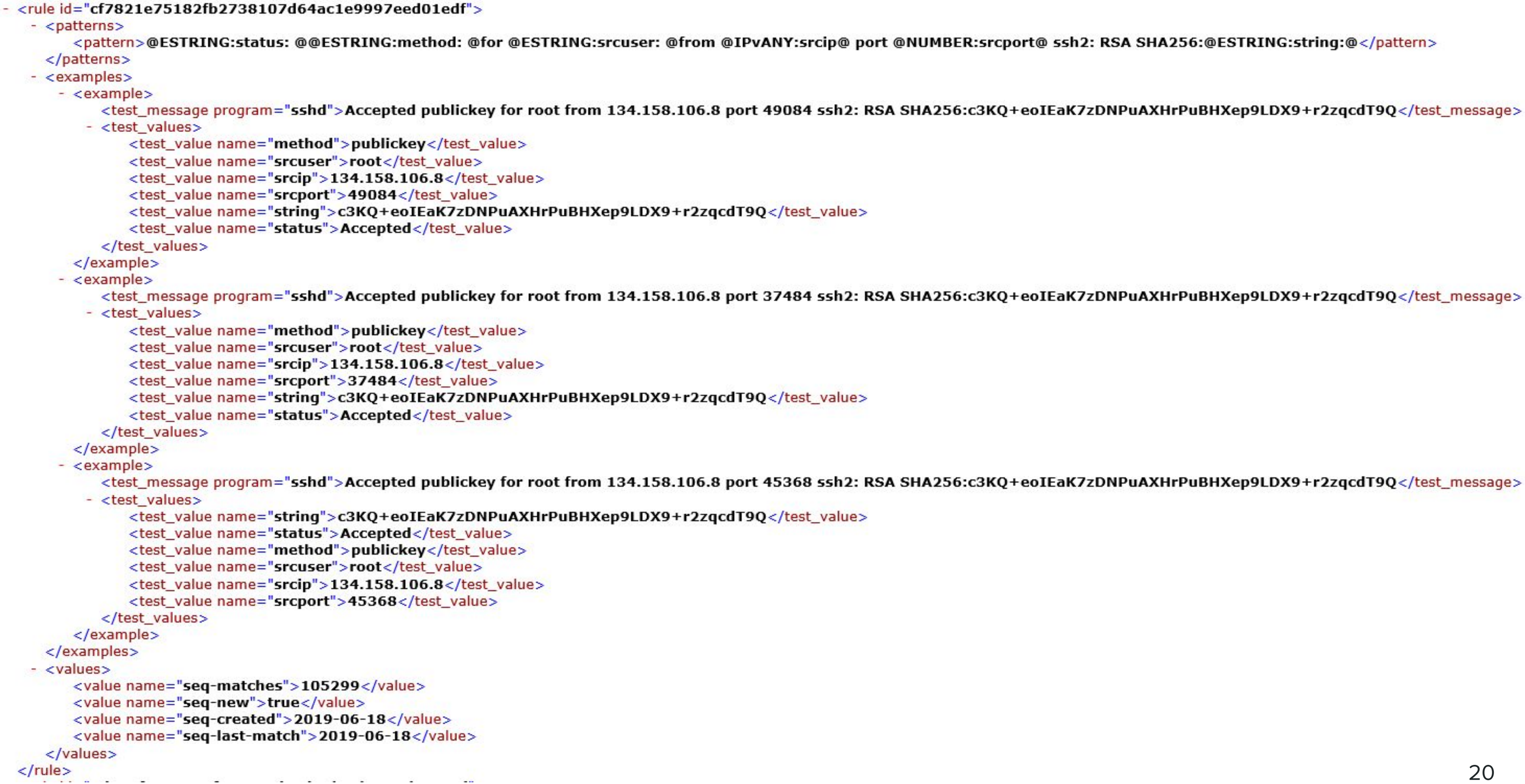


```
filter {
   grok {
     match => {"message" => "%{DATA:action} from %{IP:srcip} port %{INT:srcport}"}
     add_tag => ["2908692bdd6cb4eca096eaa19afebd9e15650b4d", "pattern_id"]
   }
}
```

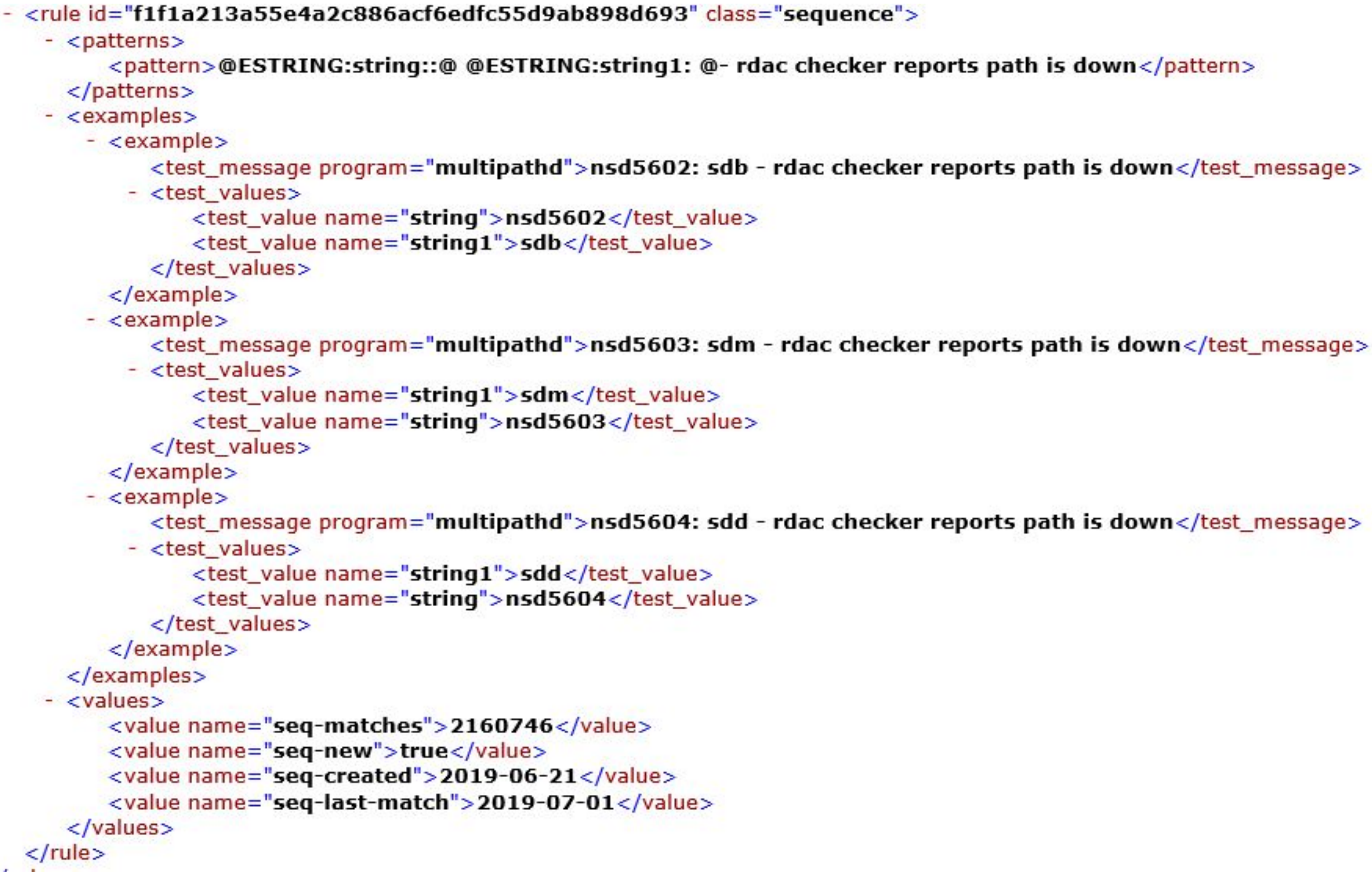
PatternDB output



```
- <rule id="2908692bdd6cb4eca096eaa19afebd9e15650b4d">
  - <patterns>
      <pattern>@ESTRING:action: @from @IPvANY:srcip@ port @NUMBER:srcport@</pattern>
    </patterns>
  - <examples>
    - <example>
        <test_message program="sshd">Disconnected from 134.158.106.8 port 41496</test_message>
      - <test_values>
          <test_value name="action">Disconnected</test_value>
          <test_value name="srcip">134.158.106.8</test_value>
          <test_value name="srcport">41496</test_value>
        </test_values>
      </example>
    - <example>
        <test_message program="sshd">Disconnected from 127.0.0.1 port 49570</test_message>
      - <test_values>
          <test_value name="srcip">127.0.0.1</test_value>
          <test_value name="srcport">49570</test_value>
          <test_value name="action">Disconnected</test_value>
        </test_values>
      </example>
    </examples>
  - <values>
      <value name="seq-matches">2</value>
      <value name="seq-new">true</value>
      <value name="seq-created">2019-06-18</value>
      <value name="seq-last-match">2019-06-18</value>
    </values>
  </rule>
```



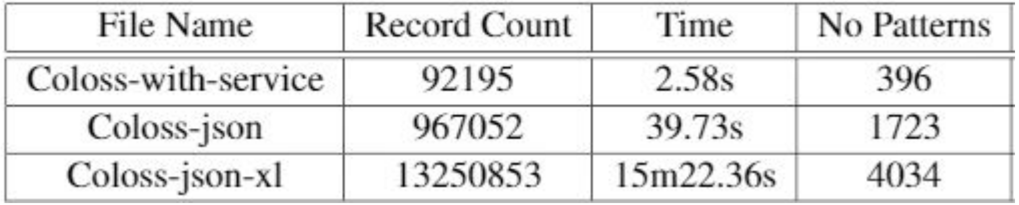
```
- <rule id="cf7821e75182fb2738107d64ac1e9997eed01edf">
  - <patterns>
      <pattern>@ESTRING:status: @@ESTRING:method: @for @ESTRING:srcuser: @from @IPvANY:srcip@ port @NUMBER:srcport@ ssh2: RSA SHA256:@ESTRING:string:@</pattern>
    </patterns>
  - <examples>
    - <example>
        <test_message program="sshd">Accepted publickey for root from 134.158.106.8 port 49084 ssh2: RSA SHA256:c3KQ+eoIEaK7zDNPuAXHrPuBHXep9LDX9+r2zqcdT9Q</test_message>
      - <test_values>
          <test_value name="method">publickey</test_value>
          <test_value name="srcuser">root</test_value>
          <test_value name="srcip">134.158.106.8</test_value>
          <test_value name="srcport">49084</test_value>
          <test_value name="string">c3KQ+eoIEaK7zDNPuAXHrPuBHXep9LDX9+r2zqcdT9Q</test_value>
          <test_value name="status">Accepted</test_value>
        </test_values>
      </example>
    - <example>
        <test_message program="sshd">Accepted publickey for root from 134.158.106.8 port 37484 ssh2: RSA SHA256:c3KQ+eoIEaK7zDNPuAXHrPuBHXep9LDX9+r2zqcdT9Q</test_message>
      - <test_values>
          <test_value name="method">publickey</test_value>
          <test_value name="srcuser">root</test_value>
          <test_value name="srcip">134.158.106.8</test_value>
          <test_value name="srcport">37484</test_value>
          <test_value name="string">c3KQ+eoIEaK7zDNPuAXHrPuBHXep9LDX9+r2zqcdT9Q</test_value>
          <test_value name="status">Accepted</test_value>
        </test_values>
      </example>
    - <example>
        <test_message program="sshd">Accepted publickey for root from 134.158.106.8 port 45368 ssh2: RSA SHA256:c3KQ+eoIEaK7zDNPuAXHrPuBHXep9LDX9+r2zqcdT9Q</test_message>
      - <test_values>
          <test_value name="string">c3KQ+eoIEaK7zDNPuAXHrPuBHXep9LDX9+r2zqcdT9Q</test_value>
          <test_value name="status">Accepted</test_value>
          <test_value name="method">publickey</test_value>
          <test_value name="srcuser">root</test_value>
          <test_value name="srcip">134.158.106.8</test_value>
          <test_value name="srcport">45368</test_value>
        </test_values>
      </example>
    </examples>
  - <values>
      <value name="seq-matches">105299</value>
      <value name="seq-new">true</value>
      <value name="seq-created">2019-06-18</value>
      <value name="seq-last-match">2019-06-18</value>
    </values>
  </rule>
```



```
- <rule id="f1f1a213a55e4a2c886acf6edfc55d9ab898d693" class="sequence">
  - <patterns>
      <pattern>@ESTRING:string::@ @ESTRING:string1: @- rdac checker reports path is down</pattern>
    </patterns>
  - <examples>
    - <example>
        <test_message program="multipathd">nsd5602: sdb - rdac checker reports path is down</test_message>
      - <test_values>
          <test_value name="string">nsd5602</test_value>
          <test_value name="string1">sdb</test_value>
        </test_values>
      </example>
    - <example>
        <test_message program="multipathd">nsd5603: sdm - rdac checker reports path is down</test_message>
      - <test_values>
          <test_value name="string1">sdm</test_value>
          <test_value name="string">nsd5603</test_value>
        </test_values>
      </example>
    - <example>
        <test_message program="multipathd">nsd5604: sdd - rdac checker reports path is down</test_message>
      - <test_values>
          <test_value name="string1">sdd</test_value>
          <test_value name="string">nsd5604</test_value>
        </test_values>
      </example>
    </examples>
  - <values>
      <value name="seq-matches">2160746</value>
      <value name="seq-new">true</value>
      <value name="seq-created">2019-06-21</value>
      <value name="seq-last-match">2019-07-01</value>
    </values>
  </rule>
```

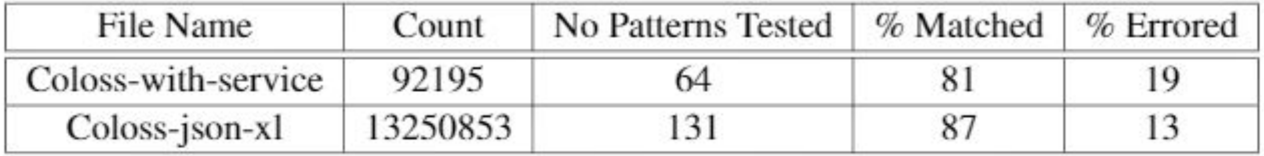
Results

SEQUENCE Testing



File Name	Record Count	Time	No Patterns
Coloss-with-service	92195	2.58s	396
Coloss-json	967052	39.73s	1723
Coloss-json-xl	13250853	15m22.36s	4034

Syslog-ng PatternDB parser testing



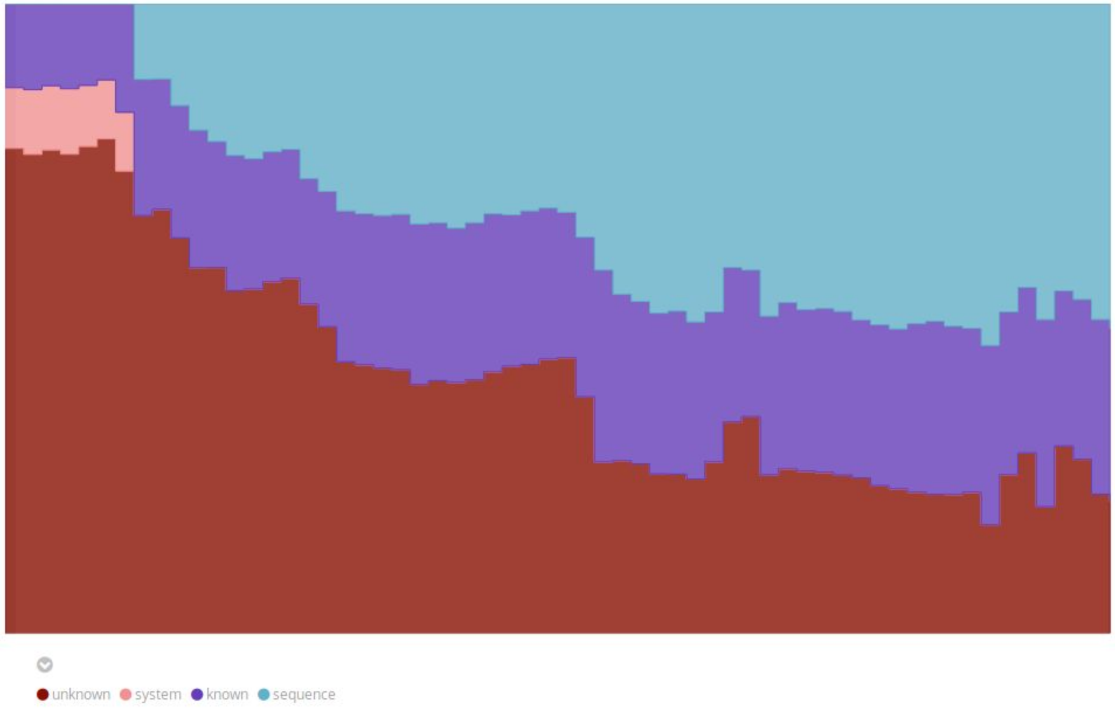
File Name	Count	No Patterns Tested	% Matched	% Errored
Coloss-with-service	92195	64	81	19
Coloss-json-xl	13250853	131	87	13

Results: Production

Known/unknown in production:

SEQUENCE	**51.6%**
Other known	27.4
Unknown	**21%**

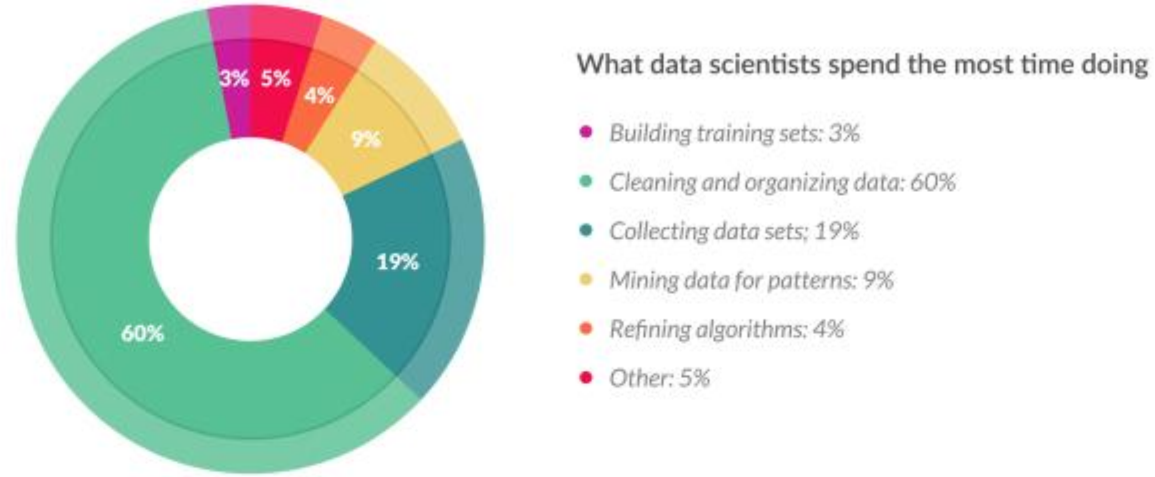
Runs every **15 mins**, takes **7 seconds** to process 100,000.



Limitations

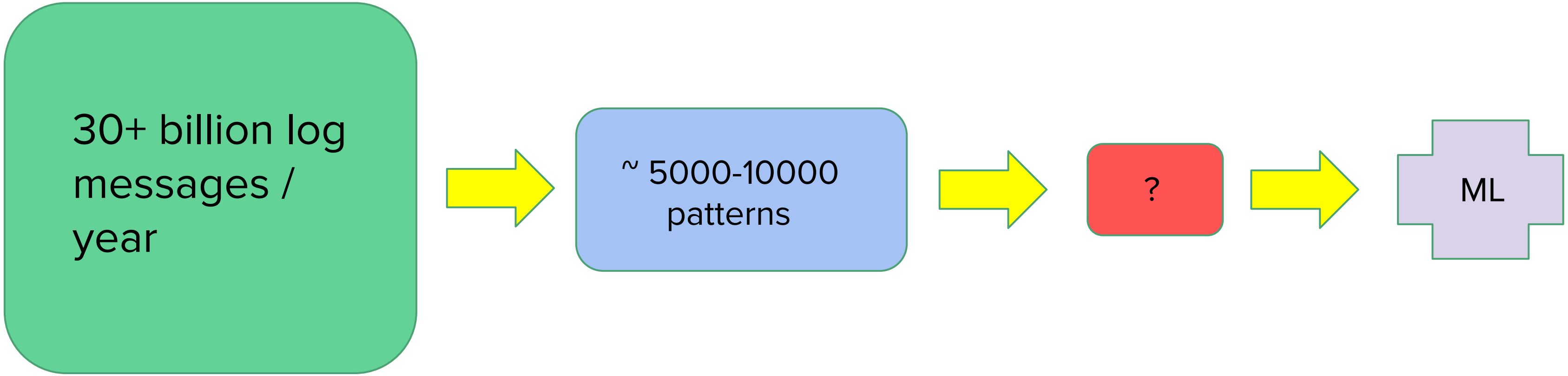
- Needs a **few examples** to find a good pattern.
- Some log messages have a pattern that **matches a token type incorrectly**.
- **Keywords** can cause **more than one pattern for similar** log messages.
- **Struggles with some key/value pairs** when the value is **not delimited.**
- Converting between **different parser types** will never be exact.

Machine Learning - Where does this fit?



What data scientists spend the most time doing

The big picture



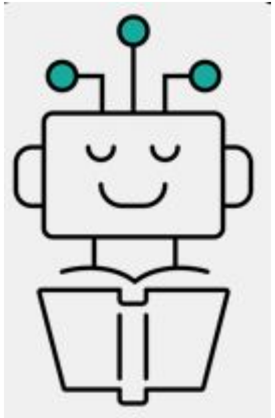
30+ billion log messages / year

~ 5000-10000 patterns

?

ML

Possible next steps



- Machine Learning - Anomaly Detection
    - Types of anomalies
        - Known events - frequency
        - New/unseen events
        - Change in sequence of events
        - Change in event parameters
    - Considerations
        - Frequency of change/maintenance
        - Definition of ‘normal’
        - Volume of messages
        - Privacy?
        - How do we communicate anomalies?
        - Feedback - dealing with false positives

Conclusion

- With close to **80% known** log messages, well on our way to the **goal of 90%**
- **Pattern discovery and creation** has made the **maintenance** of the patternDB more **manageable**.
- With the **extra meta-data** and **patternID**'s in **Elastic Search**, **easier to search** when diagnosing **issues** or looking for **information**,
- **First steps in preprocessing** the data for Machine Learning approaches like anomaly detection have been taken.
- **On track** for release back into **Open Source Community**.

Supervised vs unsupervised

Supervised learning is where you have **input variables (x)** and an **output variable (Y)** and you use an algorithm to **learn the mapping function** from the input to the output.

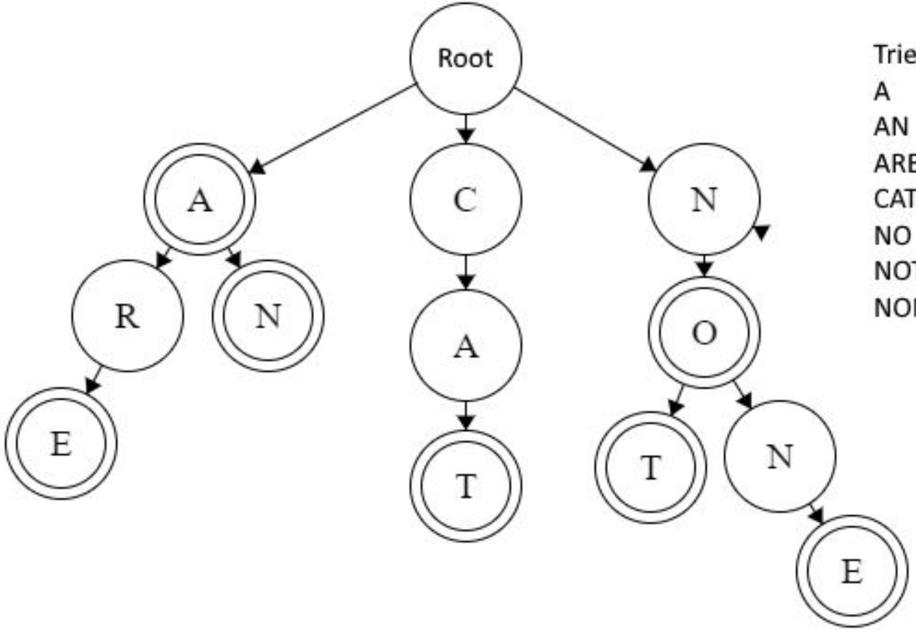
Unsupervised learning is where **you only have input data (X)** and no corresponding output variables.The goal for unsupervised learning is **to model the underlying structure or distribution** in the data in order to learn more about the data.

What approach to use? - Latest research

- Unsupervised, semi-supervised?
- Neural network using Long Short Term Memory (LSTM) for time series prediction to model frequency per pattern.
  - Compare predicted to actual - significant difference = anomaly
- LSTM for learning log message sequences or,
- Learned finite state machines, markov models.

Analyser: Trie Introduction

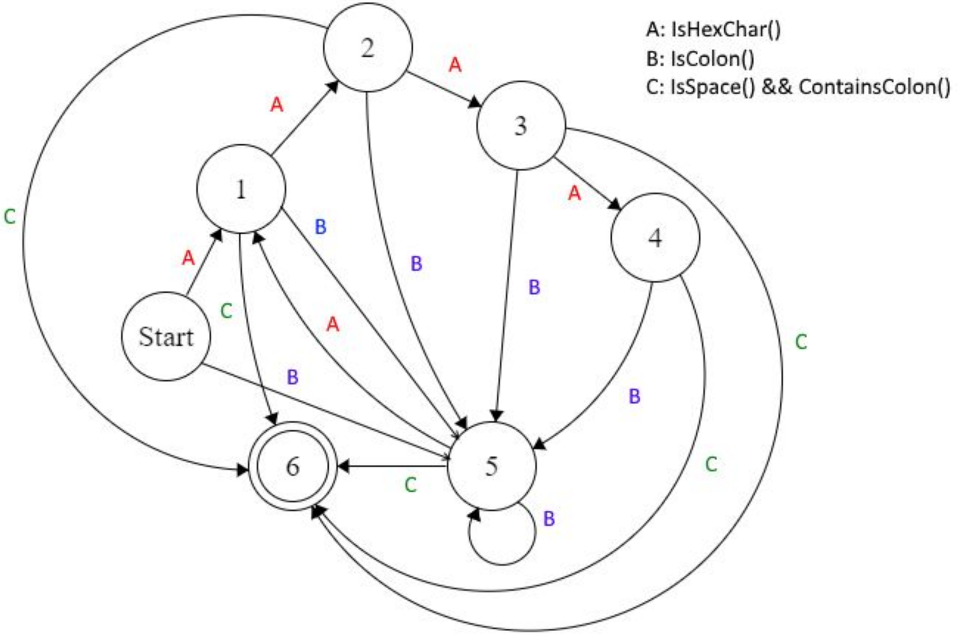
- data structure that **specialises** in working with **strings**
- allows for **very fast search** and retrieval of values.
- Most common use - **autocomplete**



Trie for words:
A
AN
ARE
CAT
NO
NOT
NONE

Finite State Machine - Hexidecimal

1. First character
2. Second character
3. Third character
4. Fourth character
5. Colon
6. Space



Finite State Machine for Hexidecimal tokens

Date Time

Supports 49 different formats

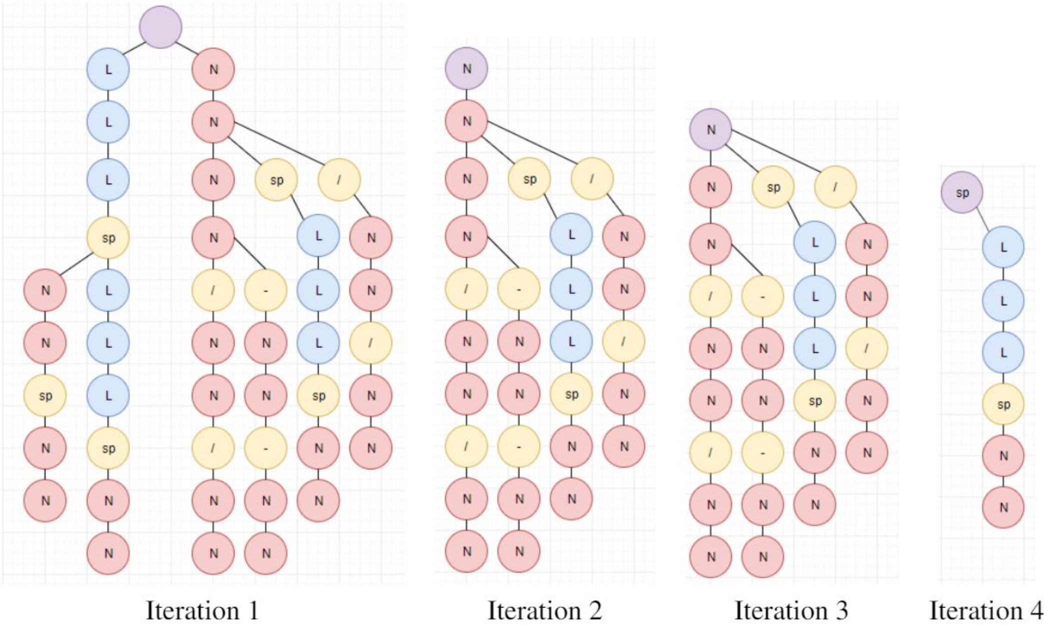


Figure 4: Trees passed to the time FSM with each iteration for example date 15 Jun 19.