Expectations to Machine Learning for Network Management

NML-RG meeting, IRTF/IETF 97 (Seoul)
November, 2016

Kohei Shiomoto (NTT)
Outline

- Network management issues (3min)
- Data-driven approach (3min)
- Two examples of our practice (9min)
  1. SYSLOG analytics
  2. Trouble Ticket analytics
     Explain only our goal and key idea
     Skip through several slides on math
- Discussion
Challenges in ICT Management

- Diversified applications & services/Modern Web traffic
  - Streaming, browsing, SNS, e-commerce, e-Health, ...

- High expectations for Availability & Quality
  - 99.9999% availability, high resolution, low noise, stalling free, quick response, ...

- Complex ICT system structure
  - Devices, software, protocols, ...

- Interaction of players
  - Customer, ISPs, CDN, ...

- Communications get encrypted. https://...
Disaggregation - why we pursue?

- Vertically integrated system
  - Network devices (router, switch, middle-box, etc.) have been vertically integrated system.
  - Those vertically integrated systems consist of many hardware and software components provided by different component vendors.
  - The end of life of some components could risk the life of entire system.
  - Adding new functionalities is under control of system vendor.
- Disaggregation could solve the issues.
SDN & NFV

Software-driven Control

Software-Defined Networking (SDN)
- Routing & Signaling, traffic engineering/steering,
- Commodity L2/L3 Switch-Router hardware

Network Function Virtualization (NFV)
- Middlebox (NAT, FW, IDS/IPS, VPN, CDN, ...)
- Commodity X86 machine hardware
Cost of disaggregation
-Increase of complexity

- Disaggregation brings about a number of benefits: cost reduction, elastic capacity, quick deployment of new functions.
- But those benefits could not be obtained without sacrifice.
- Network device, which is disaggregated into components, introduces further complexity caused by interactions among components.
- Each component is frequently and quickly replaced with newer one as new features are developed and released.
How to deal with complex system?

■ Powerful network management paradigm is required to deal with complexity.

■ We could not rely on traditional mechanism-driven approach, which pieces together accurate mechanisms of individual components.

■ We have to rely on a holistic data-driven approach to model an entire system by analyzing relationship between inputs and outputs.
  ■ *Conventional* Mechanism-driven approach
    ■ Given understanding precise mechanisms of components, build up a model of entire system.
  ■ *Towards* Data-driven approach
    ■ Given data, infer the relationship between inputs and outputs.
    ■ Machine learning is a key.
Varieties of data can be used.

- Traffic load
- Performance
- Syslog
- Trouble tickets
- SNS messages (e.g. Twitter)
- ...

Numerical, text, ...
Spatio-Temporal Factorization of Log Data for Understanding Network Events

Tatsuaki Kimura, Keisuke Ishibashi, Tatsuya Mori, Hiroshi Sawada, Tsuyoshi Toyono, Ken Nishimatsu, Akio Watanabe, Akihiro Shimoda, Kohei Shiomoto
Background

- Various network logs are gathered by NMSs and monitored
  - Switch, router, RADIUS sever, ...
  - syslog, server log, alarm, SNMP trap, ...
  - logs contain useful information for NW trouble shooting
Background

- Various network logs are gathered by NMSs and monitored
  - Switch, router, RADIUS sever, ...
  - syslog, server log, alarm, SNMP trap, ...
  - logs contain useful information for NW trouble shooting

- Diverse and massive amounts of logs
  - multiple vendors, multiple services, complex network events
  - over 1,000,000 of messages/day

⇒ Analyzing logs has become serious problem
Our research goal

Mining network (NW) event information from large and diverse NW log data

Network events = spatial and temporal patterns of log messages

messages associated with initialization of various process caused by router reboot event
multiple layer flaps caused by L-1 flap (L-2, OSPF re-convergence, BGP flap, ...)
virtual path dis-connection related to physical machine failure
Why mining NW events?

Automatically constructing domain knowledge of NW operators (without requiring skills and experience)

Many possible applications:

- Obtaining new alarm rules
- Quick understanding of root cause of problems
- May help in detection of “silent failure”
Key idea (observation)

Log data can be considered as **rank-3 tensor**
- types of log messages (templates), hostname, time

Observed log data = ‘superposition’ of NW events

<table>
<thead>
<tr>
<th></th>
<th>msg1</th>
<th>msg2</th>
<th>msg3</th>
<th>msg4</th>
<th>msg5</th>
<th>msg6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>◯</td>
<td>◯</td>
<td></td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>◯</td>
<td>◯</td>
<td>◯</td>
<td>△</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td></td>
<td>◯</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td></td>
</tr>
</tbody>
</table>

Extraction of NW events problem $\Rightarrow$ **Tensor Factorization** problem
Challenges & Summary of Contributions

[1] Unstructured and massive log messages

More than 1,000,000 lines/day

Formats of log messages depend on vendor or service

⇒ We present **Statistical Template Extraction (STE)**
  - automatically extract primary templates from large log data

[2] Log data are very complex

Underlying network events occur all around network

Network events span across several locations, network layers, and services

⇒ We present **Log Tensor Factorization (LTF)**
  - extract *spatial and temporal patterns* of log data
  - based on Nonnegative Tensor Factorization (NTF) approach
STE Motivation

Raw log messages cannot be used directly

log messages contain various *parameters* (IP address, host name, PID,...,etc.)

To correlate log messages, we need to know log *templates* = messages without parameters

```plaintext
%TRACKING-5-STATE: 1 interface Fa0/0 line-protocol Up->Down
%LINK-3-UPDOWN: Interface FastEthernet 0/9, changed state to down
%SYS-5-CONFIG I: Configured from console by vty2 (10.11.11.11)
```

```plaintext
%TRACKING-5-STATE: * interface * line-protocol Up->Down
%LINK-3-UPDOWN: Interface FastEthernet *, changed state to down
%SYS-5-CONFIG I: Configured from console by * (*)
```
STE Methodology

1. **Scoring frequency of words** among similar messages
   - *parameter words* appear infrequently compared to *template words* in each position

2. **Clustering score**, and determine *parameter words* for each message
   - Thresholds for score of *parameter words* differ depending on log messages' density-based clustering algorithm (**DBSCAN**)

### Raw Log Messages
- `<189>` security telnet connection 15720 with 10.7.11.11 broken
- `<189>` security telnet connection 18340 with 10.8.9.123 broken

### Log Template
- `<189>` security telnet connection * with * broken

---

**Scoring:**
- If *word* appears in *P*-th position in log that contains *L* words:
  \[
  Score(word, P, L) = \Pr(word \mid P, L)
  \]

**Clustering Scores (DBSCAN):**
- Distance between each cluster is > \(\delta\)
Spatial & temporal patterns we want to extract

Hierarchical correlations are observed in log data

Definition 1 [Template Group]
- group of templates that tend to co-occur: \( l = (i_1, i_2, \ldots) \)
  - represents \textit{event at individual host}
  - e.g. linkflap: linkdown + linkup
  - reboot: process initialization messages

Definition 2 [Network Event]
- set of tuples <host, template groups> that tend to co-occur
  \[ e = \{(h_1, l_1), (h_2, l_2), \ldots\} \quad (h_1, h_2 \in H, \quad l_1, l_2 \in L) \]
  - \textit{spatial extension} of template groups
  - e.g. link down event among neighboring hosts
LTF

Log Tensor $X$ ($I \times H \times J$)

$x_{ihj}$: # of occurrences of template $i$ at host $h$, time window $j$

- templates: $1,\ldots, i, \ldots, I$
- hosts: $1,\ldots, h, \ldots, H$
- time windows: $1,\ldots, j, \ldots, J$ (log data are partitioned)

- LTF factorizes $X$ into

1. Log template matrix $V = [v_I]$ ($I \times L$)
2. Network event tensor $Z = [z_{ik}]$ ($L \times K \times H$)
3. Weight matrix $W = [w_k]$ ($K \times J$)

* $K$: # of network events. $L$: # of template groups (given)
Intuitive interpretations of LTF

template group matrix $V$

$l$-th row of matrix $V$ represents $l$-th template group
⇒ corresponds to Template Group

network event tensor $Z$

$k$-th slice of $Z$ represents which template groups occur at which hosts
⇒ corresponds to Network Event
LTF Problem Formulation

Optimization problem with nonnegative constraints on each tensor

Objective function = KL-divergence between $X$ and $V, Z, W$

Mathematical formulation:

$$\min_{V, Z, W} D(X \| V, Z, W),$$

subject to:

- $V, Z, W \geq 0,$
- $\sum_i v_{il} = 1, \sum_{l, h} z_{lkh} = 1,$

$$D(X \| V, Z, W) = \sum_{i, h, j} x_{ihj} \log \frac{x_{ihj}}{\sum_k v_{il} z_{lkh} w_{kj}} - x_{ihj} + \sum_{k, l} v_{il} z_{lkh} w_{kj}$$

Algorithm (multiplicative update rules; type of EM algorithm)

Simple and iterative form:

- $v_{il} := \frac{\sum_{h, j, k} \tilde{v}_{ihj} \tilde{w}_{kij} \cdot x_{ihj}}{\sum_{h, j, k} \tilde{v}_{ihj} \tilde{z}_{lkh} w_{kj}} \tilde{v}_{il},$
- $z_{lkh} := \frac{\sum_{i, j} \tilde{v}_{ihj} \tilde{z}_{lkh} w_{kj}}{\sum_{i, j} \tilde{v}_{ihj} \tilde{z}_{lkh} w_{kj}} \tilde{z}_{lkh},$
- $h_{kj} := \frac{\sum_{i, h, l} \tilde{v}_{ihj} \tilde{z}_{lkh} w_{kj}}{\sum_{i, h, l} \tilde{v}_{ihj} \tilde{z}_{lkh} w_{kj}} \tilde{w}_{kj},$
STE Evaluation

Data Set
5 million lines of logs (captured in small network, 5 months)

Evaluation metrics = effectiveness

calculate # of extracted templates

5,000,000 logs ⇒ 2,000 ~ 3,000 templates (less than 1%)

False positive words ⇒ 0.07 ~ 0.08% of 1,000,000 words
LTF Evaluation Metrics

Data Set

- over 600,000 lines of 1-day log data
- dimensions are roughly $100 \times 150 \times 150$

Evaluation Metrics

*expressive power*: How well does LTF fit to real data?

⇒ use well known measure ‘average test log-likelihood’

- prediction power of randomly masked elements
- higher value means better fit to data
LTF Evaluation Results

average test log-likelihood with different K and L

we used normal NTF model as baseline

* L: # of template groups, K: # of network events

LTF fits better to real data than current NTF
### Neighboring link flap event

**NEIGHBORING LINK FLAP EVENT**

<table>
<thead>
<tr>
<th>Host Name</th>
<th>TG Weights</th>
<th>Weights</th>
<th>Templates</th>
</tr>
</thead>
</table>
| CoreRouterA    | 0.666      | 0.4     | `TIME : ifmgr [ * ] : %PKT_INFRA-LINK-3-UPDOWN : Interface * , changed state to Up`
|                |            | 0.4     | `TIME : ifmgr [ * ] : %PKT_INFRA-LINK-3-UPDOWN : Interface * , changed state to Down`
|                |            | 0.2     | `TIME : %SYS-3-LOGGER_DROPPED : System dropped * console debug messages.`   |
| EdgeRouterB    | 0.333      | 0.4     | `*: * : %LINK-3-UPDOWN : Interface * , changed state to up`                |
|                |            | 0.17    | `*: TIME : %LINK-3-UPDOWN : Interface * , changed state to administratively down` |
|                |            | 0.17    | `*: * : %LINEPROTO-5-UPDOWN : Line protocol on Interface * , changed state to up` |
|                |            | 0.17    | `*: * : %LINEPROTO-5-UPDOWN : Line protocol on Interface * , changed state to down` |
|                |            | 0.05    | `*: * : %LINK-3-UPDOWN : Interface * , changed state to down`              |

**Interface up/down**

- \( z_{ikh} \)
- \( v_{il} \)

**CoreRouter A** (neighbor)
### Tunneling path disconnection event

<table>
<thead>
<tr>
<th>Host Name</th>
<th>TG Weights</th>
<th>Template Weights</th>
<th>Templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreRouter</td>
<td>0.0253</td>
<td>0.4375</td>
<td>SNMP Trap: a status change for a module. Software image for the module is missing or invalid....</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0833</td>
<td>os: loader: * for * is *</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0833</td>
<td>id of requester is *</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0833</td>
<td>OsCrashDump: invalid crash record skipped, ...</td>
</tr>
<tr>
<td>EdgeRouterA</td>
<td>0.01656</td>
<td>0.92805</td>
<td>Tunneling Virtual Path * is disconnected, hardware unavailable.</td>
</tr>
<tr>
<td>EdgeRouterA</td>
<td>0.01594</td>
<td>0.98308</td>
<td>Tunneling Virtual Path * is disconnected, hardware unavailable.</td>
</tr>
</tbody>
</table>

#### Module reload event

[Tunneling path disconnection diagram](#)

- CoreRouter
- EdgeRouter A
- EdgeRouter A
- More than 50 connections
Summary of SYLOG Analytics

Presented STE

Extracting primary templates from noisy log messages
Compressing 5,000,000 lines to 3000 templates

Presented LTF

Modeling generation of logs as rank-3 tensor and factorizing into template groups and network events
Much better than current NTF
Can correctly extract hidden complex network events
Trouble Ticket Analytics

Workflow Extraction for Service Operation using Multiple Unstructured Trouble Tickets

Akio Watanabe, Keisuke Ishibashi, Tsuyoshi Toyono, Tatsuaki Kimura, Keishiro Watanabe, Yoichi Matsuo, Kohei Shiomoto

NTT Network Technology Laboratories
April 25, NOMS 2016
Background

• We would like to specify troubleshooting process
  – MTTR strongly depends on the time until deciding the resolution

Troubleshooting procedure for system management

Detection → Isolation → Resolution decision → Resolution

Research targets: Specify Troubleshooting

• Why process is not defined?
  – various process for thousands of failures
  – requiring tacit knowledge
  – including domain rule
Present understanding of process

- Operators search the resolution from *trouble tickets* – amount of valuable knowledge about failures
- Much information are written by *natural language*

<table>
<thead>
<tr>
<th>Ticket ID</th>
<th>Date/time of occurrence</th>
<th>Incident name</th>
</tr>
</thead>
<tbody>
<tr>
<td>012345</td>
<td>2015/03/20 10:00:00</td>
<td>Router went down</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Host name</th>
<th>Model</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>r001</td>
<td>A Inc. x-1001</td>
<td>Hardware failure</td>
</tr>
</tbody>
</table>

**Working history**

L1: 10:00 system detected the following error
L2: 10:00:00 Router xxx went down
L3: Connection OK
L4: We checked the router log.
L5: #login r001
L6: #show system
L7: Module error
L8: Error continuing, we decided on replacement.
L9: 11:15 Replacement began.
L10: 11:30 Replacement done.
L11: 11:30 Reboot message was detected.
L12: 12:00 We sent logs to A Inc.
L13: 02/02 10:00 received report.
L14: 
L15: Mail from: xxxx
......
L20: We closed the ticket.
Our goal

- **Automatically specifying** troubleshooting process from trouble tickets
- Generating *Workflow*, **graphical** flowchart of actions for each failure

- **clarifying decide rules**

- **summarizing multiple trouble tickets**
Two challenges

• Finding *action sequences* for each trouble ticket

<table>
<thead>
<tr>
<th>Ticket ID</th>
<th>Date/time of occurrence</th>
<th>Incident name</th>
</tr>
</thead>
<tbody>
<tr>
<td>012345</td>
<td>2019-09-20</td>
<td>Router went down</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Host name</th>
<th>Model</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>r001</td>
<td>A Inc. x-001</td>
<td>Hardware failure</td>
</tr>
</tbody>
</table>

- Working history:
  
  L1: 10:00 system detected the following error  
  L2: 10:00:00 Router xxx went down  
  L3: Connection OK  
  L4: We checked the router log.  
  L5: #login r001  
  L6: #show system  
  L7: Module error  
  L8: Error continuing, we decided on replacement.  
  L9: 11:15 Replacement began.  
  L10: 11:30 Replacement done.  
  L11: 11:30 Reboot message was detected.  
  L12: 12:00 We sent logs to A Inc.  
  L13: Mail from : A Inc.  
  L14: 02/02 10:00 received report.  
  L15:  
  L20: We closed the ticket.

- Seeing detected errors
- Checking Network Connectivity
- Checking system log
- Replacing module
- Send error logs to vendor
- Close Ticket

• Finding *isolating action*; that have multiple transitions to resolutions

- Seeing detected errors
- Checking Network Connectivity
- Checking system log
- Replacing module
- Send error logs to vendor
- Close Ticket
- Replacing Cable
- Close Ticket
Approach overview

- **3 steps** for extract workflow from multiple trouble tickets
  - 1. extract only action sentences from trouble tickets
  - 2. align the same messages in different tickets
  - 3. find operational change as a branch

1. **labeling**
   - we detected error
   - connectivity is OK
   - we replaced module

2. **alignment**
   - system detected error
   - verified connectivity
   - we rebooted server

3. **finding isolating action**
1. Action Sentence Labeling

• Extracting sentences about (operator/system’s) actions
  – Append sentences to labels indicating if is written about actions or not

• **Supervised learning** from labeled texts
  – Naive Bayes are used as classifier
  – Character-2gram is used as features

<table>
<thead>
<tr>
<th>Time</th>
<th>Action/Other</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00</td>
<td>Action</td>
<td>Management system detected the following error</td>
</tr>
<tr>
<td>10:00:00</td>
<td>Other</td>
<td>Router xxx went down</td>
</tr>
<tr>
<td>10:00:00</td>
<td>Action</td>
<td>Connection Okay</td>
</tr>
<tr>
<td>10:00:00</td>
<td>Other</td>
<td>We checked the router log.</td>
</tr>
<tr>
<td>10:00:00</td>
<td>Other</td>
<td>#login r001</td>
</tr>
<tr>
<td>10:00:00</td>
<td>Other</td>
<td>#show system</td>
</tr>
<tr>
<td>10:00:00</td>
<td>Other</td>
<td>Error continuing, we decided on replacement.</td>
</tr>
<tr>
<td>11:15</td>
<td>Action</td>
<td>Replacement began.</td>
</tr>
<tr>
<td>11:30</td>
<td>Action</td>
<td>Replacement done.</td>
</tr>
<tr>
<td>11:30</td>
<td>Action</td>
<td>Reboot message was detected.</td>
</tr>
<tr>
<td>12:00</td>
<td>Action</td>
<td>We sent system logs to A Inc.</td>
</tr>
<tr>
<td>2/2</td>
<td>Action</td>
<td>10:00 received report</td>
</tr>
<tr>
<td>2/2</td>
<td>Other</td>
<td>Mail from ...</td>
</tr>
</tbody>
</table>
2. Action Alignment

- Aligning sentences describing the same action

<table>
<thead>
<tr>
<th>Action sentences 1</th>
<th>Action sentences 2</th>
<th>Action sentences 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 System <strong>detected</strong> temporary error</td>
<td>Error is <strong>detected</strong>.</td>
<td>Error: Node down</td>
</tr>
<tr>
<td>2</td>
<td>We verified connectivity.</td>
<td></td>
</tr>
<tr>
<td>3 Ping is OK</td>
<td></td>
<td>Ping NG</td>
</tr>
<tr>
<td>4 Many <strong>errors</strong> were in log</td>
<td>There is no <strong>error</strong> in log</td>
<td></td>
</tr>
</tbody>
</table>

Consider aligned numbers as action sequence
Formulation as *maximal matching Problem*

- Corresponding similar sentences
  - maximize the sum of similarities of aligned sentences
    \[
    \hat{G} = \arg \max_G \sum_{j=1}^{J} \sum_{i,i' \in I \atop i \neq i'} \text{sim}(G_{ij}, G_{i'j})
    \]
- Solving by *multiple sequence alignment* (MSA) method
  - MUSCLE algorithm [18] is used for aligning multiple sequences of sentences
  - dice coefficient as similarity of sentence pair
    \[
    \text{dice}(s_{ij}, s_{i'j'}) = \frac{2|s_{ij} \cap s_{i'j'}|}{|s_{ij}| + |s_{i'j'}|}
    \]
3. Isolating Action Searching

- Finding **isolating action** i.e. *branch of transitions to two resolutions*
- Problem: many **imitate branches** caused by loss or noise in action sequence

[Diagram showing steps for isolating action]

- **Isolating action**
  - Find isolating action = branch of transitions
  - **Check connectivity**
    - Check if address can resolve
    - Check IP route
  - Imitate branch by noise
    - Replace module
    - Power off
    - Pull out cable
    - Insert new module
    - Boot module
  - Imitate branch by loss of action
    - Check IP route
    - Check session
    - Check IP route
Key observation

• the following actions of isolating action can be clustered by resolutions

seems to have two kinds of resolution

imitate branch
Method for finding isolating action

1. dividing the following actions using Spectral Clustering for each actions
2. choosing the action that has the best clustering score

\[
\text{Score of the action} = \text{coh}(\mathbf{G}, \mathcal{I}^{(+)}, \mathcal{I}^{(-)}, j) = \sum_{i, i' \in \mathcal{I}^{(±)}} (|\mathbf{G}_{i, j}^{(±)}| - \text{dist}(\mathbf{G}_{i, j}^{(±)}, \mathbf{G}_{i', j}^{(±)})),
\]

\[
\text{dev}(\mathbf{G}^{(+)}, \mathbf{G}^{(-)}) = \sum_{i+ \in \mathcal{I}^{(+)}} \sum_{i- \in \mathcal{I}^{(+)}} \text{dist}(\mathbf{G}_{i+, j}, \mathbf{G}_{i-, j}).
\]

\[
\text{coh}(\mathbf{G}^{(±)}) = \sum_{i, i' \in \mathcal{I}^{(±)}} (|\mathbf{G}_{i, j}^{(±)}| - \text{dist}(\mathbf{G}_{i, j}^{(±)}, \mathbf{G}_{i', j}^{(±)})),
\]
Experiments

- **Dataset:** practical trouble tickets for network system
  - Written by Japanese
  - Primitive linguistic preprocessing are executed
  - Separated into subsets by detected error

<table>
<thead>
<tr>
<th></th>
<th>the number of tickets</th>
<th>resolutions / of tickets</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>5</td>
<td>(A) turn on breaker (x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(B) wait &amp; see (x3)</td>
</tr>
<tr>
<td>(ii)</td>
<td>4</td>
<td>(A) replace module (x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(B) replace port interface (x1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(C) replace cable (x1)</td>
</tr>
<tr>
<td>(iii)</td>
<td>3</td>
<td>(A) power outage (x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(B) replace ONU (x1)</td>
</tr>
<tr>
<td>(iv)</td>
<td>29</td>
<td>(A) wait &amp; see (x12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(B) detail log analysis (x15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(C) replace module (x2)</td>
</tr>
</tbody>
</table>

- **Given parameter**
  - the threshold of similarity for alignment
  - the number of isolating actions
Quantitative evaluation

- We compared obtained result with ground truth
- Comparison of
  - alignment result & manually appended action ID
  - clustering result & manually checked true resolution for each ticket
  - words of extracted isolating actions & isolating action in true operation document

<table>
<thead>
<tr>
<th>set</th>
<th>tickets($)</th>
<th>Precision/Recall</th>
<th>isolating actions</th>
<th>extracted isolating actions</th>
<th>resolutions</th>
<th>clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>5</td>
<td>87.1%/79.6%</td>
<td>check ONU power off</td>
<td>ONU/power/off/trace</td>
<td>(A) turn on breaker, (B) wait and see</td>
<td>{A,A},{B,B,B}</td>
</tr>
<tr>
<td>(ii)</td>
<td>4</td>
<td>87.7%/84.4%</td>
<td>(a) check OSPF down log, (b) field check</td>
<td>(a) OSPF/Neighbor/down, (b) repaired/Alarm</td>
<td>(A) replace module, (B) replace port, (C) replace cable</td>
<td>{A,A},{B},{C}</td>
</tr>
<tr>
<td>(iii)</td>
<td>3</td>
<td>94.9%/78.9%</td>
<td>field check</td>
<td>field/check</td>
<td>(A) power outage, (B) replace ONU</td>
<td>{A,A},{B}</td>
</tr>
<tr>
<td>(iv)</td>
<td>29</td>
<td>83.5%/70.2%</td>
<td>temporal error by tool (b) reboot was reoccurred</td>
<td>confirm/Module/Fault/transition temporal/error</td>
<td>(A) wait and see, (B) detail log analysis, (C) replace module</td>
<td>{A × 12, B × 3}, {B × 12}, {C, C}</td>
</tr>
</tbody>
</table>

- miss of alignment is limited the case in
  - exchange of order
- miss of isolating action searching is caused by
  - the loss of isolating action
  - multiple (three or more) isolations
Case study result

- frequent actions are the same with the actions in manual document
- causes are described into the next actions of isolating actions

<table>
<thead>
<tr>
<th>ID</th>
<th>description</th>
<th>resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>node down alarm was detected</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>reboot alarm was detected</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>asked field operator to check</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>there is no trace of router power off.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>found OSPF neighbor down message in log</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>verified ping connection.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>reported ONU repair alarm</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>defective cable caused link down alarm when cable swapped for maintenance.</td>
<td>wait until maintenance is over</td>
</tr>
<tr>
<td>9</td>
<td>we found port failure.</td>
<td>wait and see</td>
</tr>
<tr>
<td>10</td>
<td>now under investigation, but it is likely line failure.</td>
<td>replacement</td>
</tr>
<tr>
<td>11</td>
<td>failure in relay point has been reported.</td>
<td>replacement</td>
</tr>
</tbody>
</table>
Summary of Trouble Ticket Analytics

• Proposed extracting method for workflow automatically from multiple trouble tickets
  – Action Sentence Labeling
  – Action Alignment
  – Isolating Action Searching

• Future works
  – relaxing the limitation of order of actions
  – finding multiple isolating actions
Expectations to Machine Learning

- Correlation and Causality Inference
- Anomaly Detection
- Root Cause Analysis
- Knowledge Discovery
Diagnosing & Trouble-shooting

- Prediction
- Detection
- Root Cause Analysis
- Recovery
Concluding remarks
- Data-driven approach

- Disaggregate vertically integrated system into components to achieve sustainable healthy growth.
- Hard to understand precise mechanisms of every component of entire system.
- Measure and collect big data on inputs and outputs of the system to infer the relationship between them.
- Mathematical tools, e.g., machine learning are available here.
- Key to success is inter-play between mathematics and network engineering.
Thank you for your attention