南京邮电大学 Nanjing University of Posts and Telecommunications

FIU Florida International University

IBM Research

St. John's University

Event Mining: Algorithms and Applications

Tao Li, FIU/NJUPT

Larisa Shwartz, IBM Research

Genady Ya. Grabanrnik, St. John’s University

Contents

1 Introduction and Overview

2 Event Generation and System Monitoring

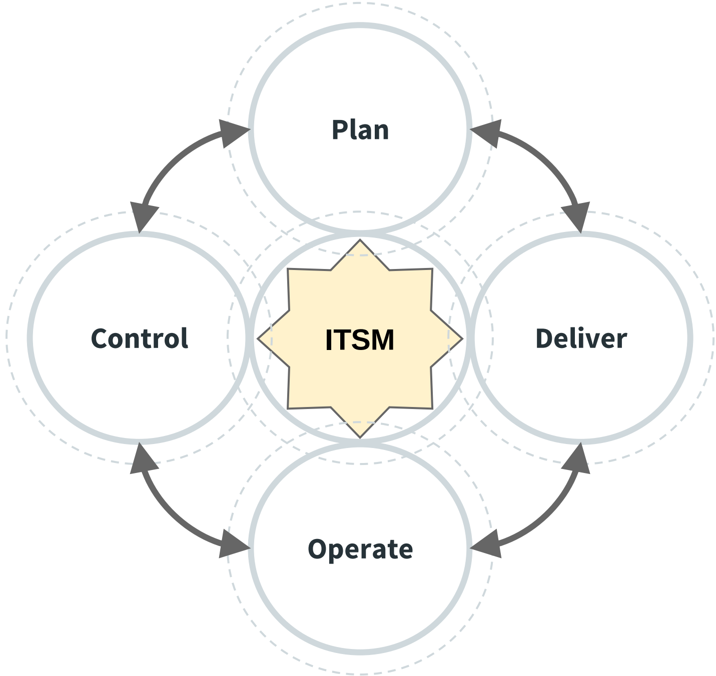
3 Pattern Discovery and Summarization

4 Mining with Events and Tickets

5 Conclusions

Service Management

- **IT Service Management** (**ITSM**) refers to the entirety of **activities** that are performed to **plan**, **deliver**, **operate** and **control** IT services offered to customers.

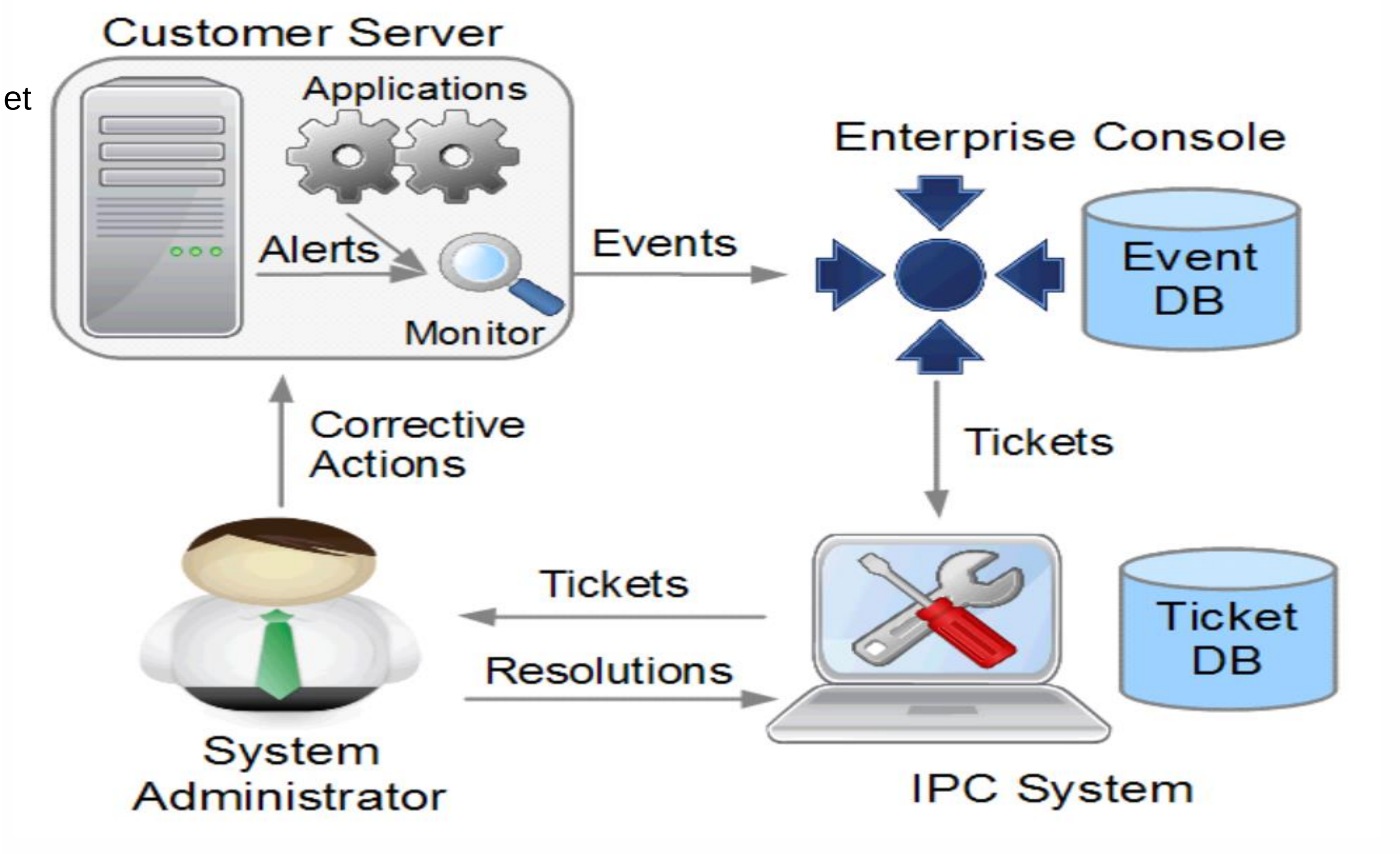


- **ITSM** grows in **popularity** over the last 30 years. Many **ITSM** products are booming from different companies.

IT Service Background

The typical workflow of IT service mainly includes four components (Tang et al., CNSM 2012; Tang et al., KDD 2013):

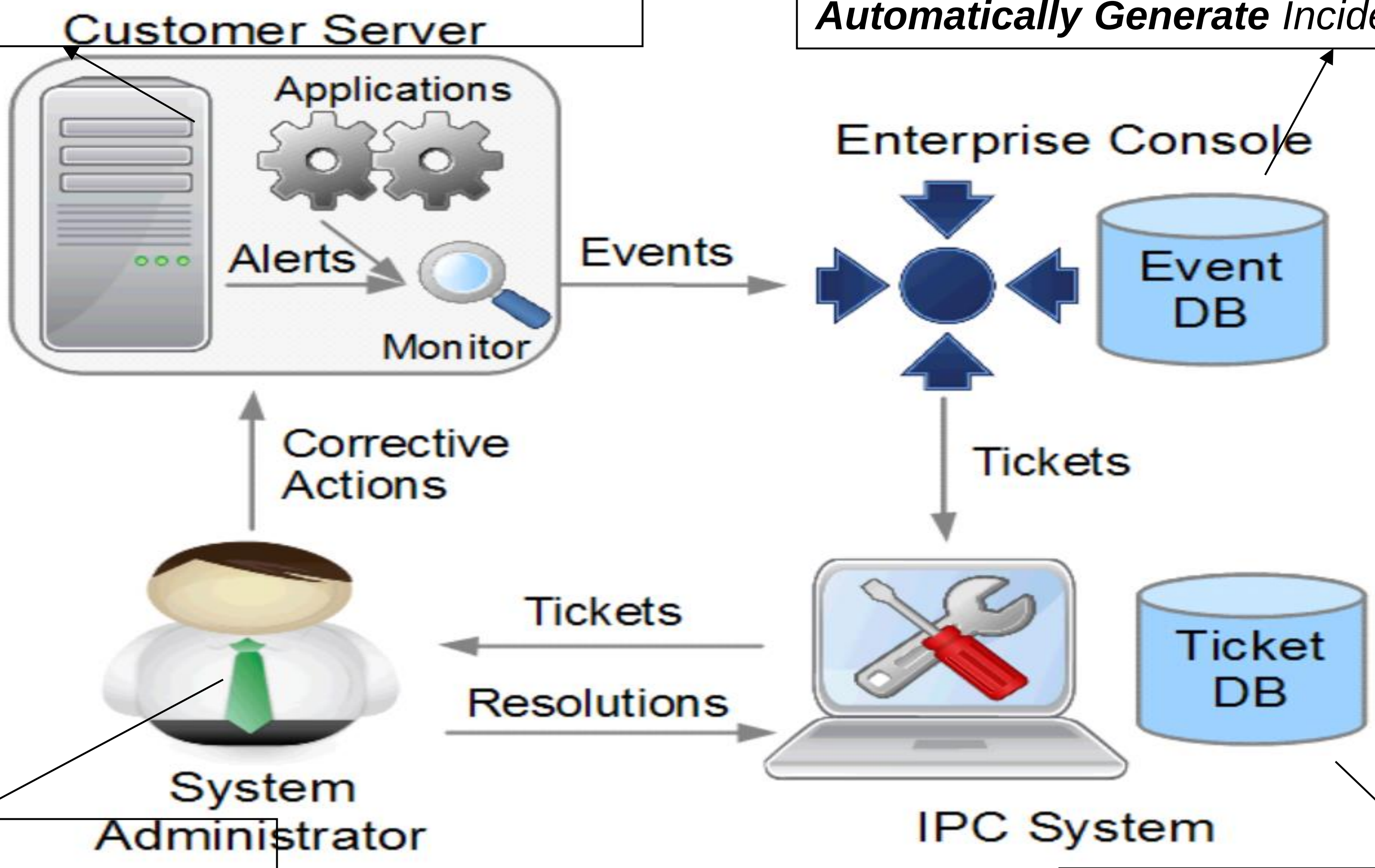
- Customer Servers
- Event DB
- Ticketing System
- System Administrators



How IT Service Works?

Checking (periodically):
If disk_name == “C:” and disk_free < 5%
If CPU_util > 80% and duration > 20 minutes
….

***Store** and **Explore** Events, **OLAP**,*
***Automatically Generate** Incident Tickets*

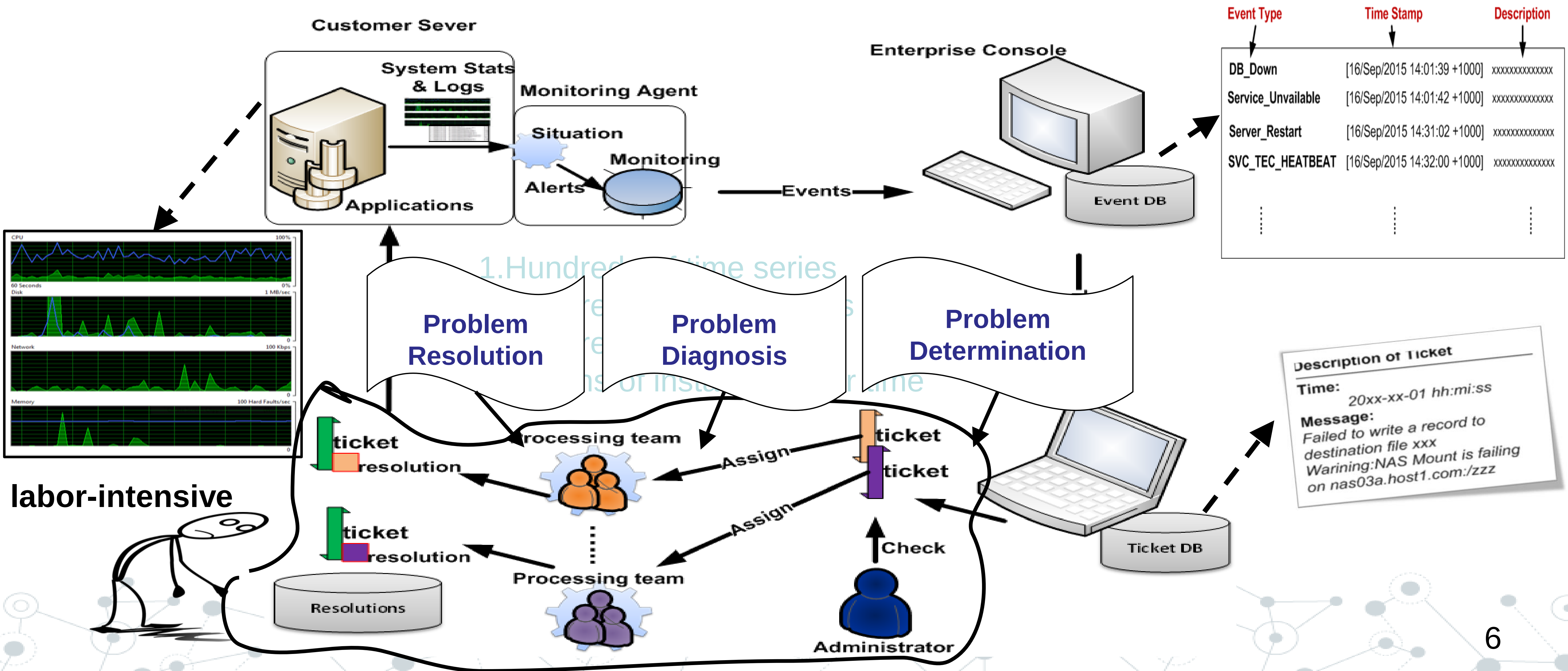


***Fix** Problems*
in Incident Tickets

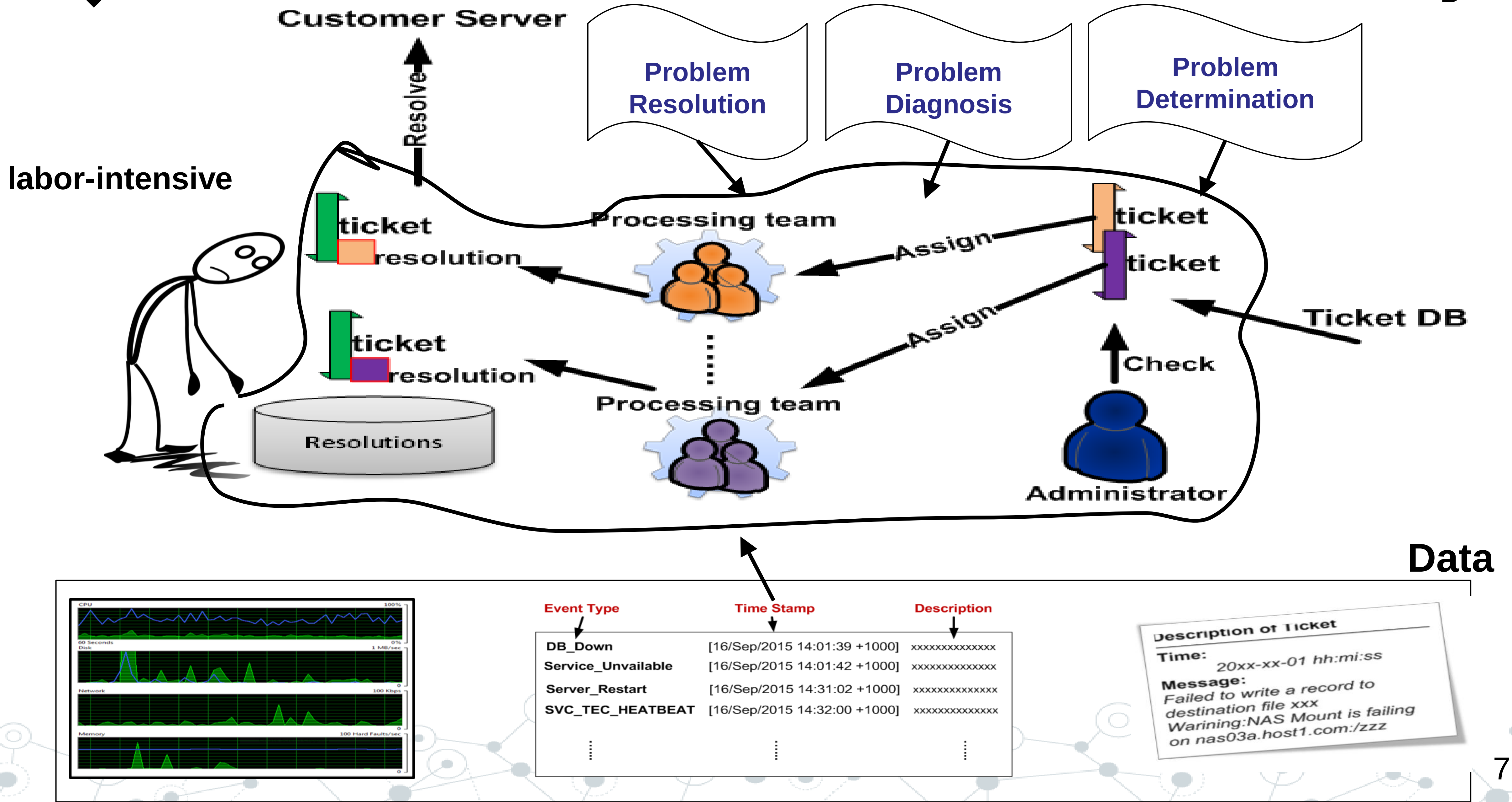
***Track** Incident Tickets*

The Workflow:Data Perspective

A typical workflow of IT Service Management involves an appropriate mix of **people**, **process**, **information** and **technology**.



Optimization with tickets, events and system stats data



Maximal automation

of

routine IT maintenance procedures

is one of ultimate goals of IT service management optimization

Different Phases of IT Service Management

1.0 — Simple Processing

2.0 — Distributed Processing

3.0 — Service Suites

4.0 — Intelligent Service Platform

Data Processing Perspective

Phase 1.0

- data size is relatively small: MB/GB

- Using testing tools（ping，traceroute，SNMP，tcpdump）or monitoring tools (e.g., Zabbix)

- problem identification、problem localization、 problem resolution，

◎Keyword search: *error*, *fatal*

Manual analysis

Unstructured event log

Sep 16 2014 23:33:33-04:30 PMTSOPLNE4001 %%01IFNET/4/LINK_STATE(l)[82791]:The line protocol None on the interface Tunnel0/0/111 has entered the DOWN state.
Oct 6 2014 09:10:33-04:00 LPZ_ALP_CX600-X8_A %%01LDP/4/SSNHOLDTMREXP(l)[458502]:Sessions were deleted because the session hold timer expired and the notification of the expiry was sent to the peer 172.24.11.3.

Semi-structured event log

{"timestamp":1364751315,"machineIP":"131.94.128.170","cpu":{"type":"cpu","userTime":0.08333333333333333,"sysTime":0.08374587458745875,"combinedTime":0.08374587458745875,"idleTime":0.9995874587458746,"cores":[{"userTime":0.0,"sysTime":0.0,"combinedTime":0.0,"idleTime":1.0},{"userTime":0.0,"sysTime":0.009900990099009901,"combinedTime":0.009900990099009901,"idleTime":0.9900990099009901} ...

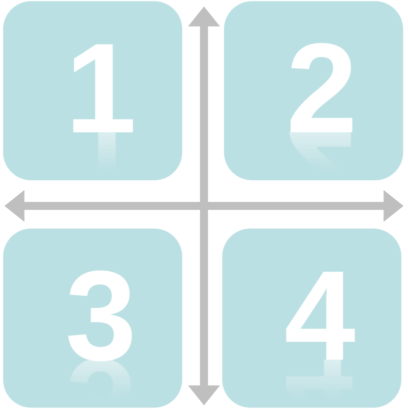
Structured event log

2014-11-07 21:22:20	6	Sohar-PE-CORE-NE80E-1-2014-11-17.12-21-43.log	50	last_message_repeated	Sohar-PE-CORE-NE80E-1
2014-11-07 21:22:38	6	Sohar-PE-CORE-NE80E-1-2014-11-17.12-21-43.log	50	last_message_repeated	Sohar-PE-CORE-NE80E-1
2014-11-07 21:23:08	6	Sohar-PE-CORE-NE80E-1-2014-11-17.12-21-43.log	50	last_message_repeated	Sohar-PE-CORE-NE80E-1
2014-11-07 21:25:01	6	Sohar-PE-CORE-NE80E-1-2014-11-17.12-21-43.log	50	last_message_repeated	Sohar-PE-CORE-NE80E-1

◎**Cmds**: head, tail, grep, cut, etc.
◎**Scripts**: awk, Perl

Phase 2.0

- Massive data size: TB/PB
- Distributed processing techniques / platforms
- Four steps of data processing:



Data Collection
Apache Chukwa，Facebook Scribe，Cloudera Flume，Fluentd

Data Storage
HDFS，NoSQL
Relational Database

Data Analysis
Hadoop MapReduce、Spark、Storm、Spark Streaming

Data Visualization
BIRT，Zeppelin

Phase 3.0

- Big data processing suites

splunk> sumologic 日志易 rizhiyi.com

Elasticsearch Logstash Kibana

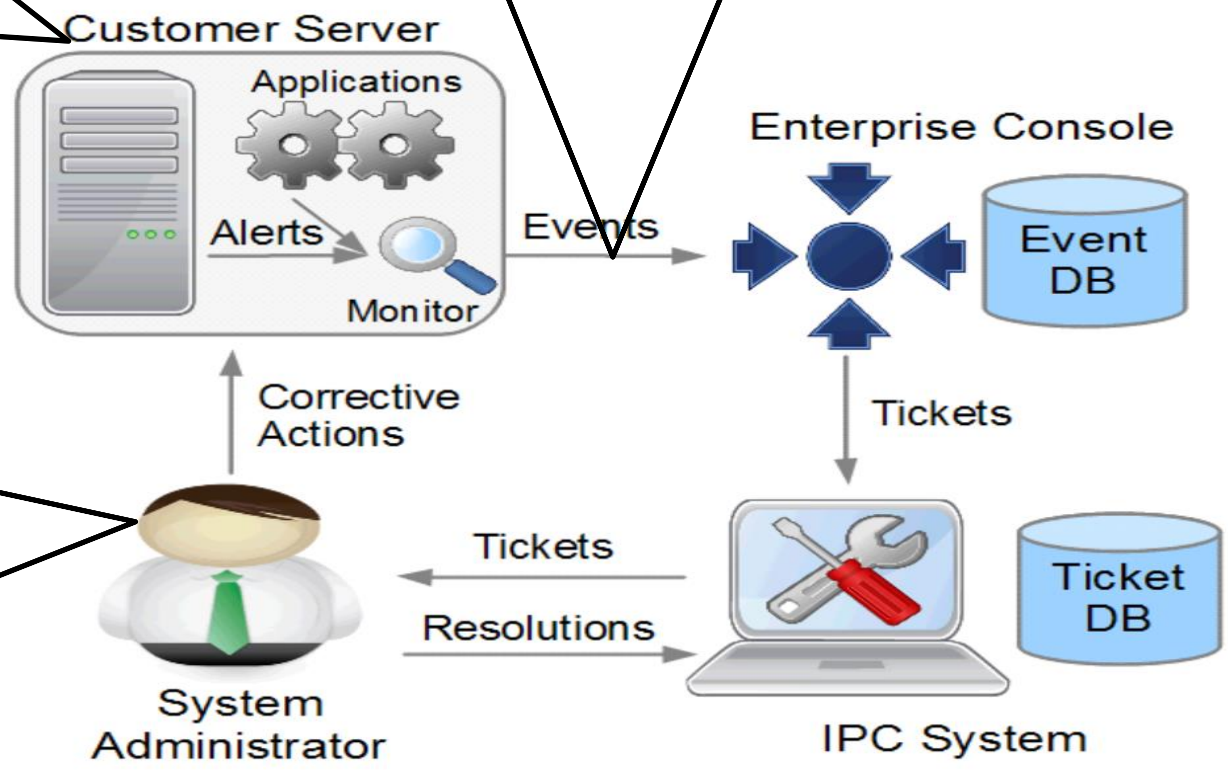
Phase 4.0

- Add more intelligent techniques (AI, Machine Learning and Data Mining Techniques) on top of the existing suites

Overview of Research Problems： Workflow

Convert Raw Textual Logs into System Events

Monitoring Configuration Optimization
- Reduce False positive (false alerts)
- Reduce False negative (missed alerts)

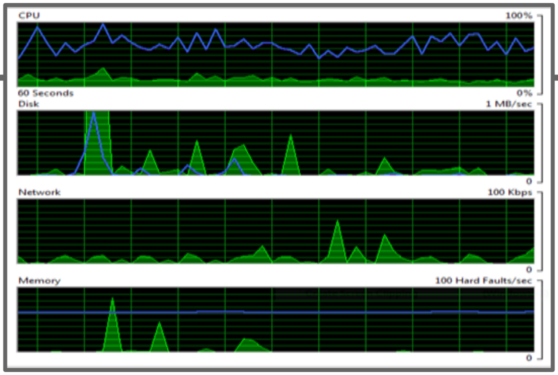


System Incidents Diagnosis
- Temporal Pattern Discovery
- Event Correlation
- Automatic Resolution Recommendation
- Problem Diagnosis
- Problem determination and resolution
- Root Cause Analysis
- Failure Prediction and Avoidance

Overview of Research Problems

Description of Ticket
Time: 20xx-xx-01 hh:mi:ss
Message:
Failed to write a record to destination file xxx
Warining:NAS Mount is failing on nas03a.host1.com:/zzz

Event Type	Time Stamp	Description
DB_Down	[16/Sep/2015 14:01:39 +1000]	xxxxxxxxxxxxxx
Service_Unvailable	[16/Sep/2015 14:01:42 +1000]	xxxxxxxxxxxxxx
Server_Restart	[16/Sep/2015 14:31:02 +1000]	xxxxxxxxxxxxxx
SVC_TEC_HEATBEAT	[16/Sep/2015 14:32:00 +1000]	xxxxxxxxxxxxxx
⋮	⋮	⋮



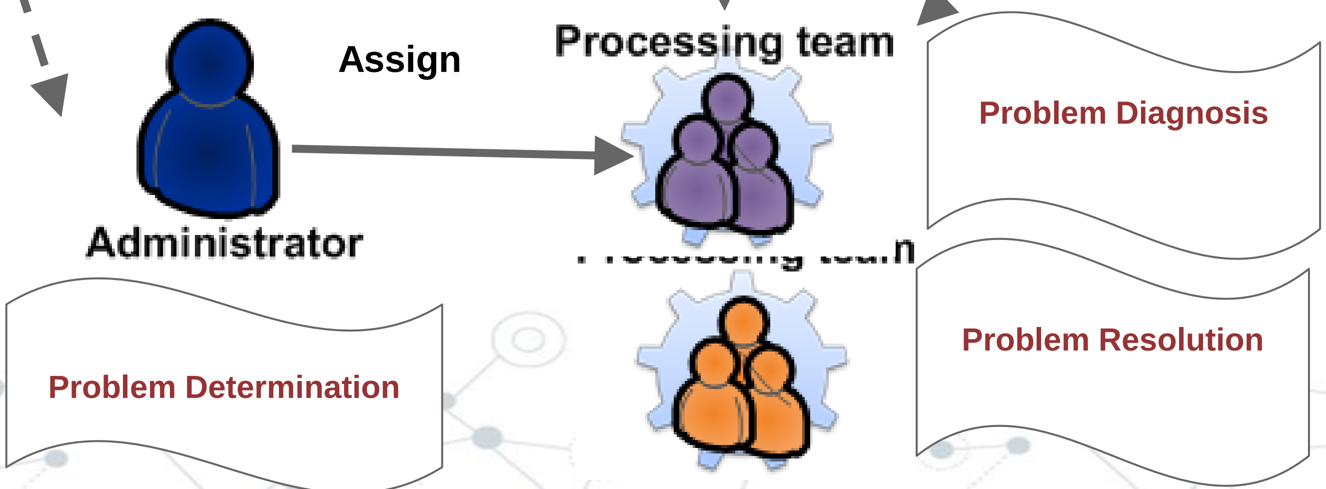
(1) Ticket classification with problem category
1. Hierarchy multi-label classification
2. Domain knowledge integration

(2) Temporal pattern mining from events
1. From logs to events
2. Fluctuating time lag modeling
3. Efficient pattern mining method

(3) Temporal dependency discovery from system stats
1. Granger causality inference
2. Online inference for time varying temporal dependency

(4) Monitoring Configuration Optimization

1. Reduce False positive (false alerts)
2. Reduce False negative (missed alerts)



(5) Ticket recommendation

Contents

1 Introduction and Overview

2 Event Generation and System Monitoring

3 Pattern Discovery and Summarization

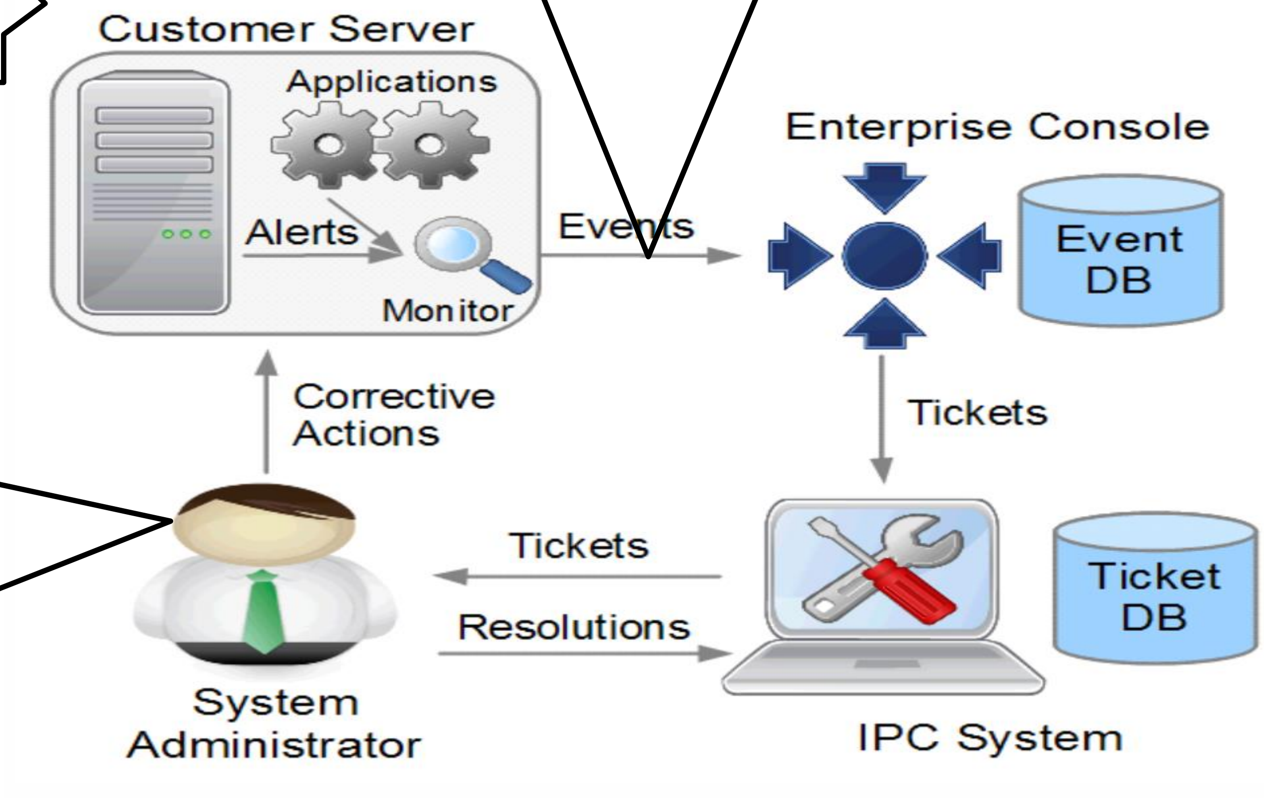
4 Mining with Events and Tickets

5 Conclusions

Outline

Convert Raw Textual Logs into System Events

Monitoring Configuration Optimization
- Reduce False positive (false alerts)
- Reduce False negative (missed alerts)

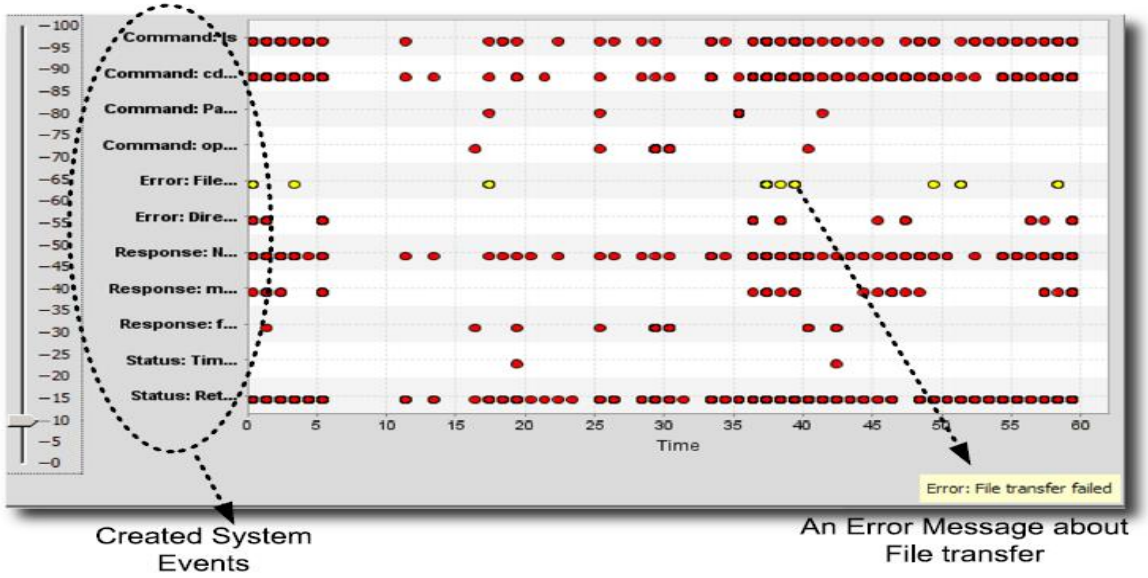


System Incidents Diagnosis
- Temporal Pattern Discovery
- Event Correlation
- Automatic Resolution Recommendation
- Problem Diagnosis
- Problem determination and resolution
- Root Cause Analysis
- Failure Prediction and Avoidance

Why Convert Textual Logs to System Events?

Table I: An Example of FileZilla's log.

No.	Message
s_1	2010-05-02 00:21:39 Command: put "E:/Tomcat/apps/index.html" "/disk/...
s_2	2010-05-02 00:21:40 Status: File transfer successful, transferred 823 bytes...
s_3	2010-05-02 00:21:41 Command: cd "/disk/storage006/users/lt...
s_4	2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt...
s_5	2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt...
s_6	2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record1.html" "/disk/...
s_7	2010-05-02 00:21:42 Status: Listing directory /disk/storage006/users/lt...
s_8	2010-05-02 00:21:42 Status: File transfer successful, transferred 1,232 bytes...
s_9	2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record2.html" "/disk/...
s_{10}	2010-05-02 00:21:42 Response: New directory is: "/disk/storage006/users/lt...
s_{11}	2010-05-02 00:21:42 Command: mkdir "libraries"
s_{12}	2010-05-02 00:21:42 Error: Directory /disk/storage006/users/lt...
s_{13}	2010-05-02 00:21:44 Status: Retrieving directory listing...
s_{14}	2010-05-02 00:21:44 Command: ls
s_{15}	2010-05-02 00:21:45 Command: cd "/disk/storage006/users/lt...
...	... ...



System events are easier to analyze other textual logs.

Converting log messages to events provides the capability of canonically describing the semantics of log data and improves the ability of correlating across the logs from multiple components.

Event Generation from Textual or Semi-structure Logs: Possible Solutions

- Log Parser (W. Xu et al., 2008)
  - Requires the understanding of all log messages.
  - Document or Source code are not available.
  - Implementation is time consuming.

- Information Extraction (Supervised):
  - Conditional Random Field.

- Clustering Based Methods (Unsupervised):
  - Bag-of-Word model
    - cosine similarity, Jaccard Index…

  - Log message matching (M. Aharon et al., 2009; A. Makanju et al, 2009)
    - Number of matched words in strings.
    - Edit distance of messages.

Clustering-based Methods

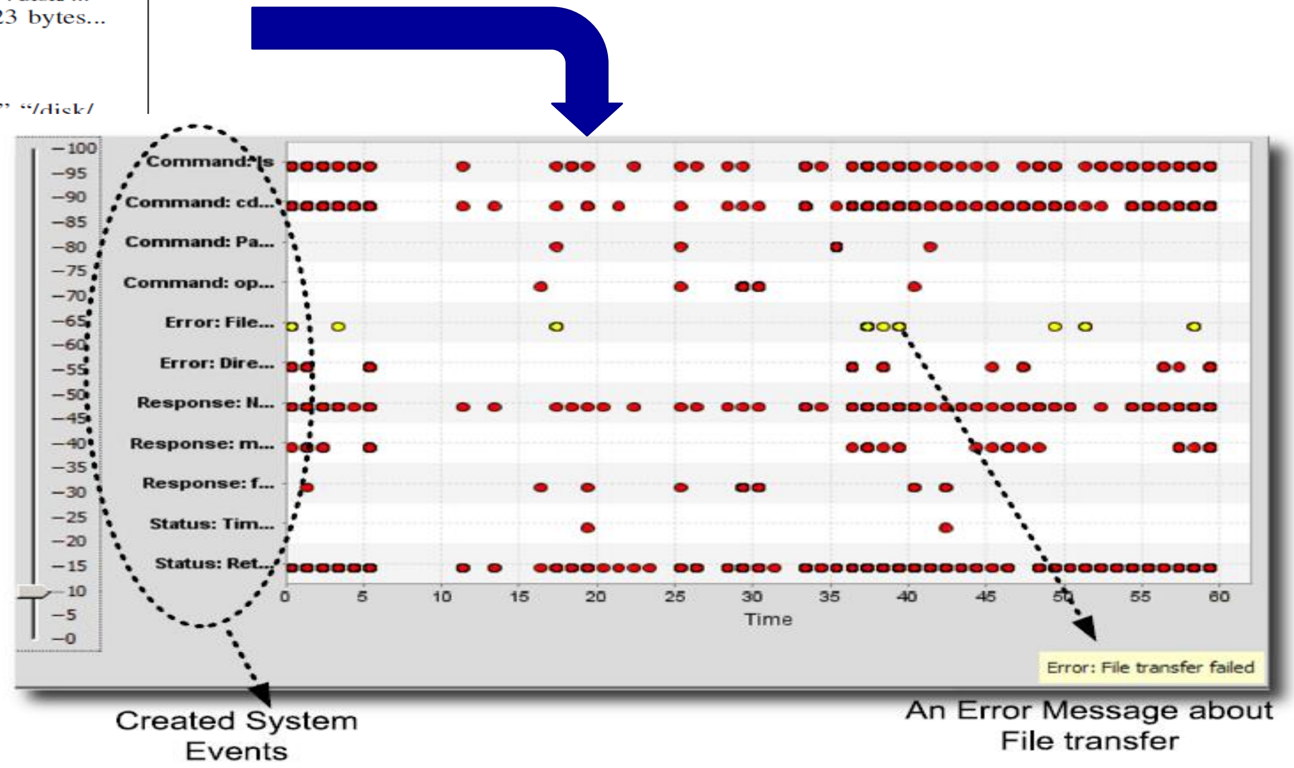
(Liang Tang and Tao Li, IEEE ICDM 2010)
(Liang Tang and Tao Li, ACM CIKM 2011)

- Goal: categorize textual or semi-structured system logs into system events.

Table I: An Example of FileZilla's log.

No.	Message
s_1	2010-05-02 00:21:39 Command: put "E:/Tomcat/apps/index.html" "/disk/...
s_2	2010-05-02 00:21:40 Status: File transfer successful, transferred 823 bytes...
s_3	2010-05-02 00:21:41 Command: cd "/disk/storage006/users/lt...
s_4	2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt...
s_5	2010-05-02 00:21:42 Command: cd "/disk/storage006/users/lt...
s_6	2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record1.html" "/disk/
s_7	2010-05-02 00:21:42 Status: Listing directory /disk/storage006/use
s_8	2010-05-02 00:21:42 Status: File transfer successful, transferred 1
s_9	2010-05-02 00:21:42 Command: put "E:/Tomcat/apps/record2.htm
s_{10}	2010-05-02 00:21:42 Response: New directory is: "/disk/storage0(
s_{11}	2010-05-02 00:21:42 Command: mkdir "libraries"
s_{12}	2010-05-02 00:21:42 Error: Directory /disk/storage006/users/lt...
s_{13}	2010-05-02 00:21:44 Status: Retrieving directory listing...
s_{14}	2010-05-02 00:21:44 Command: ls
s_{15}	2010-05-02 00:21:45 Command: cd "/disk/storage006/users/lt...
...	... ...

Solution1: utilizing a context-free grammar parser to help text clustering



Solution2: extracting the message signature to do message clustering

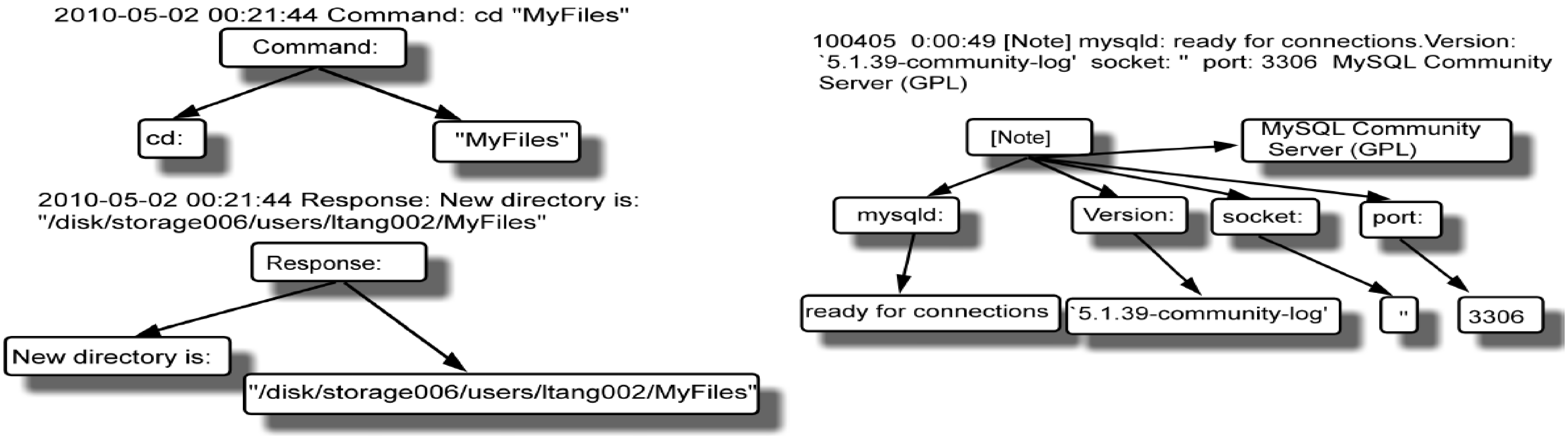
Tree-Structure based Clustering

Basic Idea

1) Convert the log messages into tree-structured data, where each node is a segment of message.

2) Do clustering based on tree-structured data.

Step 1: Convert into semi-structural log messages (log tree).



It is only a context-free grammar parser.
It separates log message by *comma*, *TAB*, etc.
It does **NOT** identify the meaning of terms (words).
It can be automatically created by JLex and JCup (or JAVACC) tools.

Step 2: Do clustering with Tree-based similarity function

Each log message is a tree. Similarity of two log messages is computed as the similarity of two trees.

Root node of s_1

Root node of s_2

$$F_C(s_1, s_2) = \frac{F'_C(r_1, r_2, \lambda) + F'_C(r_2, r_1, \lambda)}{2},$$

Message Segment at node v_1

Message Segment at node v_2

where

$$F'_C(v_1, v_2, w) = w \cdot d(L(v_1), L(v_2)) + \sum_{(v,u) \in M^*_C(v_1, v_2)} F'_C(v, u, w \cdot \lambda),$$

Decrease weight for lower layer
$\lambda < 1$

Best matching between subtree v_1's nodes with subtree v_2's nodes

Message Signature Based Clustering

Message signature is the signature of the template.

One type of log messages is generated by one template with different parameters.

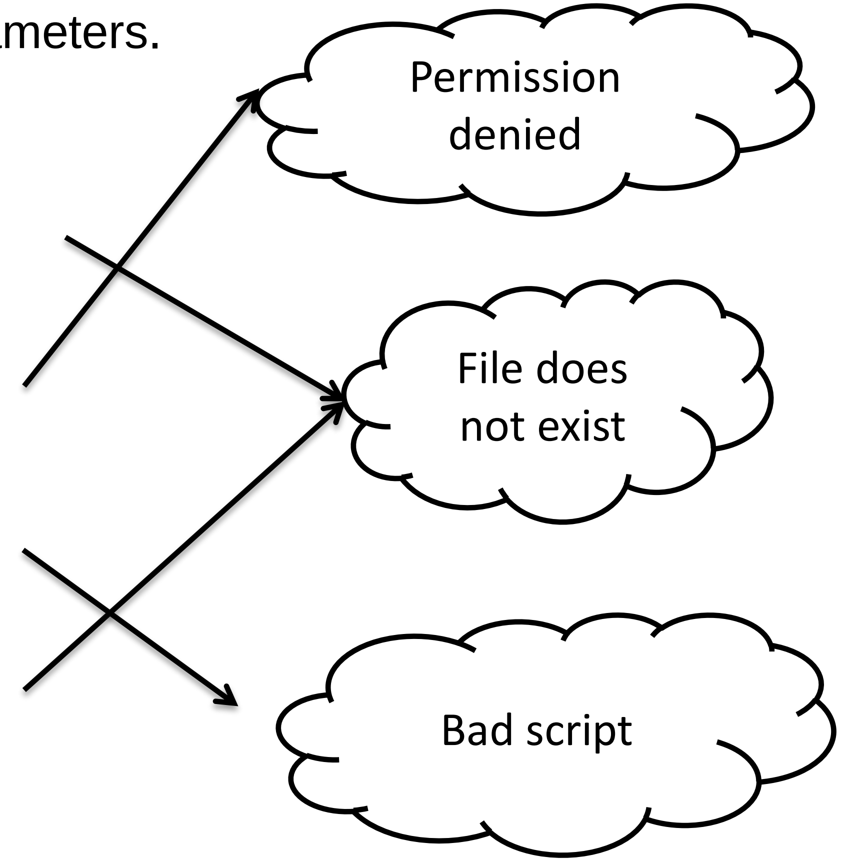
Message signature

[Thu Apr 01 00:07:31 2010] ***[error] [client*** *131.94.104.150]* ***File does not exist****: /opt/website/sites/users.cs.fiu.edu/data/favicon.ico*

[Thu Apr 01 03:47:47 2010] ***[crit] [client*** *61.135.249.68]* ***(13)Permission denied****: /home/public_html/ke/.htaccess pcfg_openfile:* ***unable to check*** *htaccess file,* ***ensure it is readable***

[Thu Apr 01 01:41:18 2010] ***[error] [client*** *66.249.65.17]* ***Premature end of script headers****: preferences.pl*

[Thu Apr 01 01:44:43 2010] ***[error] [client*** *207.46.13.87]* ***File does not exist****: /home/bear-011/users/giri/public_html/teach/6936/F03*



Each log message consists of a sequence of terms.

- Some of the terms are variables or parameters for a system event,
  - ✓ such as the host name, the user name, IP address, and so on.
- Other terms are plain text words describing semantic information of the event.

Message Signature based Clustering

- Problem: Find *k* most representative message signatures.
- Question: How to quantify the “representativeness” ?
- Definition:
  - Given a message X and a message signature S, the match score is the number of matched terms **minus** the number of unmatched terms.
  - *match*(X,S) = |LCS(X,S)| - (|S| - |LCS(X,S)|) =2|LCS(X,S)|- |S|, LCS=Longest Common Subsequence.
- Example:
  - X=“abcdef”, S=“axcey”, *match*(X,S)=|ace| - |xy| = 1

X	a	b	c	d	e	f
S	a	x	c		e	y

Problem Definition

Given a set of log messages ***D*** and an integer k, find k message signature ***S*** = $\{S_1,\ldots,S_k\}$ and a k-partition $C_1,\ldots,C_k$ of ***D*** to maximize:

$$J(\mathcal{S},\mathcal{D}) = \sum_{i=1}^{k} \sum_{X_j \in C_i} match(X_j, S_i).$$

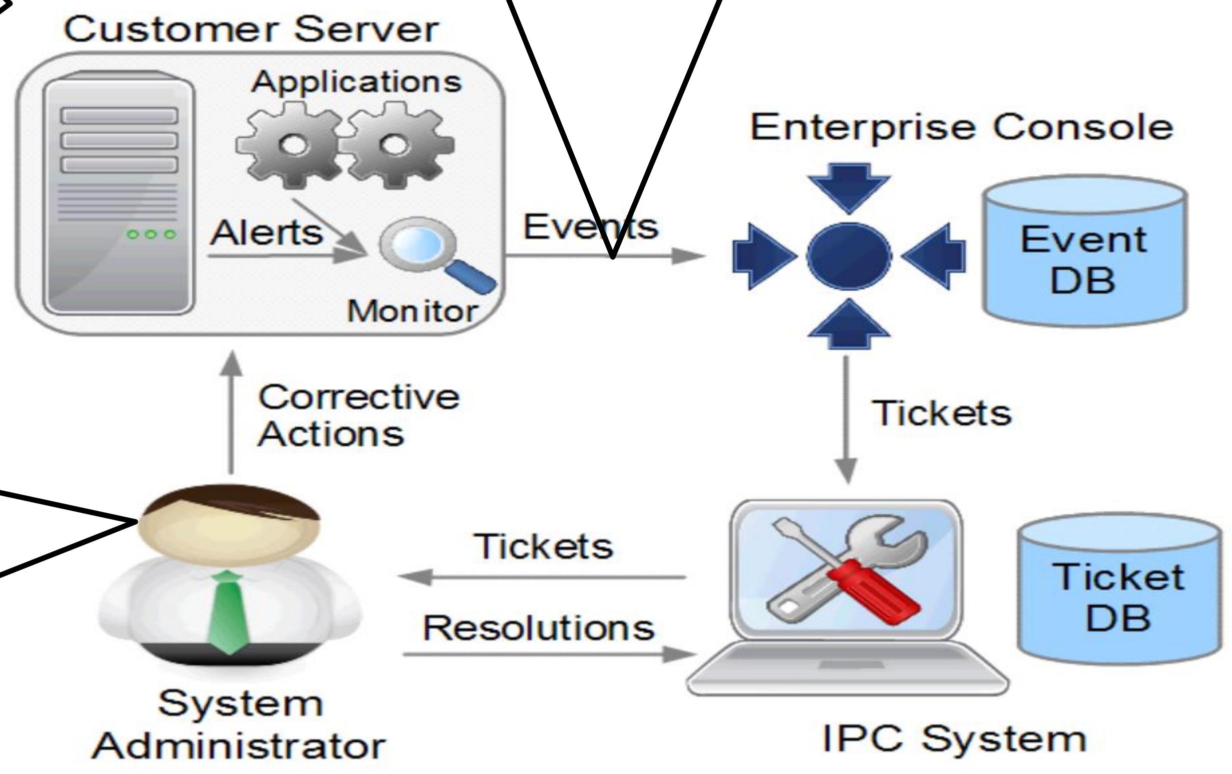
Problem Analysis:

- Similar to k-means problem, but NOT really.
- Finding the Optimal Solution is NP-Hard, even if k=1.
  - ***Multiple Longest Common Subsequence Problem*** can be reduced to our problem.

Outline

Monitoring Configuration Optimization
- Reduce False positive (false alerts)
- Reduce False negative (missed alerts)

Convert Raw Textual Logs into System Events



System Incidents Diagnosis
- Temporal Pattern Discovery
- Event Correlation
- Automatic Resolution Recommendation
- Problem Diagnosis
- Problem determination and resolution
- Root Cause Analysis
- Failure Prediction and Avoidance

What is False Positive (False Alarm)?

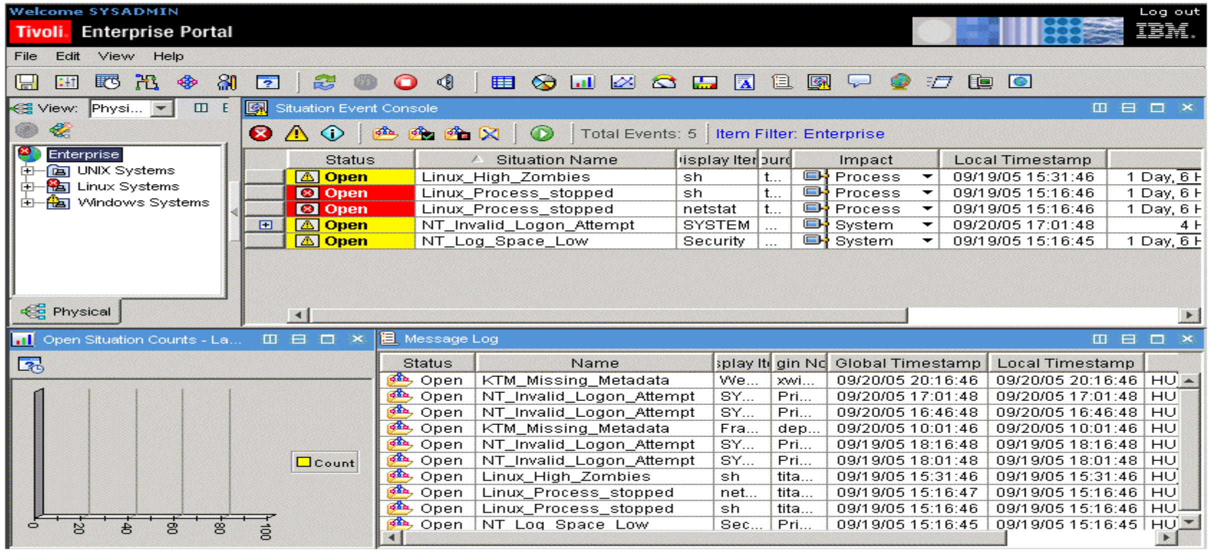
- *If PROCESS_CPU_UTILIZATION > 50% and duration > 10 minutes, then generates a CPU alert*
  - “rtvscan.exe” scans the system periodically, it is CPU intensive but it is normal, so it triggers a lot of false positives (false alerts).

- *If PAGING_UTILIZATION_15min > 400, then generate a paging alert* (default situation in IBM Tivoli monitoring)
  - Some customer servers have multiple CPU and huge memories. For those multi-CPU servers, it is normal for page swapping over thousands of times in 15 minutes.

Why We Have False Positives?

- Too Conservative Configurations
  - Missing a real alert would incur system crash, data loss.
- Changes of Monitored Servers
  - New servers and more powerful device are installed.
- Transient Alerts:
  - Temporal CPU, Paging, Disk Spike.
  - Restart of servers, processes, services, routers…

IBM Tivoli Monitoring



Complicated configurations for IBM Tivoli monitoring

Problem Statement & Challenge

- Problem Statement
  – Eliminate false positives by refining the Monitoring configurations

- Challenge
  – Retain all real alerts. No real alert is allowed to miss.

Related Work

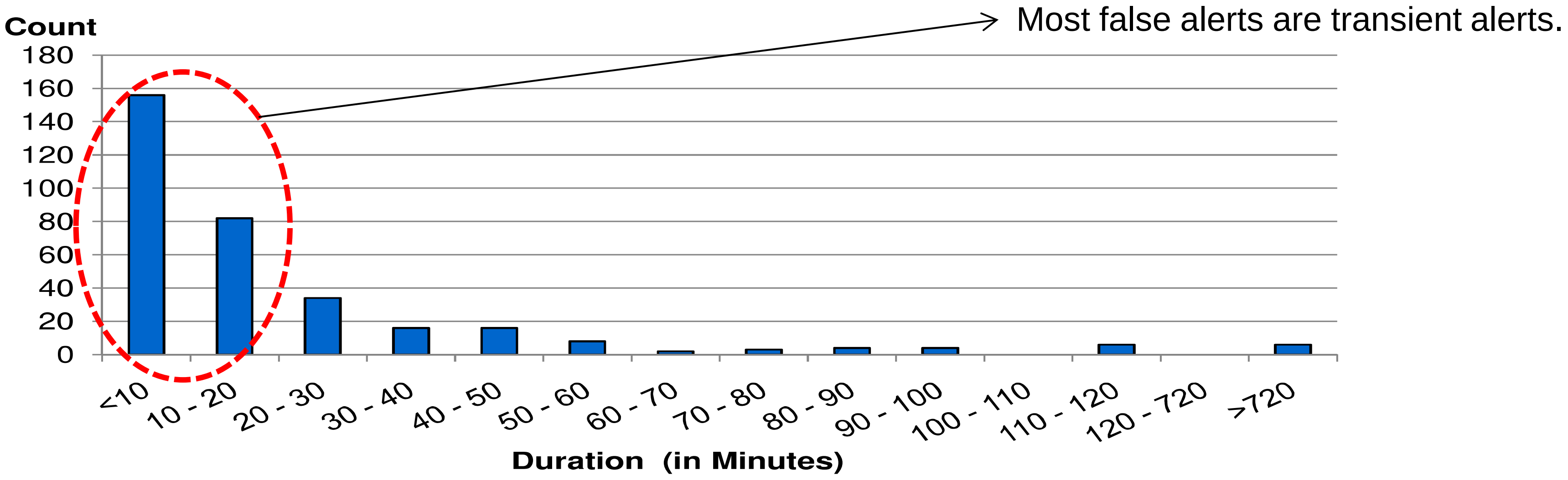
- Monitoring Products
  - IBM Tivoli, HP OpenView, Splunk

- System Alert Detection
  - Heuristic Methods (codebook…).
  - Supervised Learning Methods
    - Outlier Detection (S. Agrawal et al., 2007, K. Xu et al., 2005)
    - Adaptive threshold (S.R. Kashyap et al., 2008)
  - Supervised Learning Methods (classification).

However, they do not guarantee NO real alert is missed.

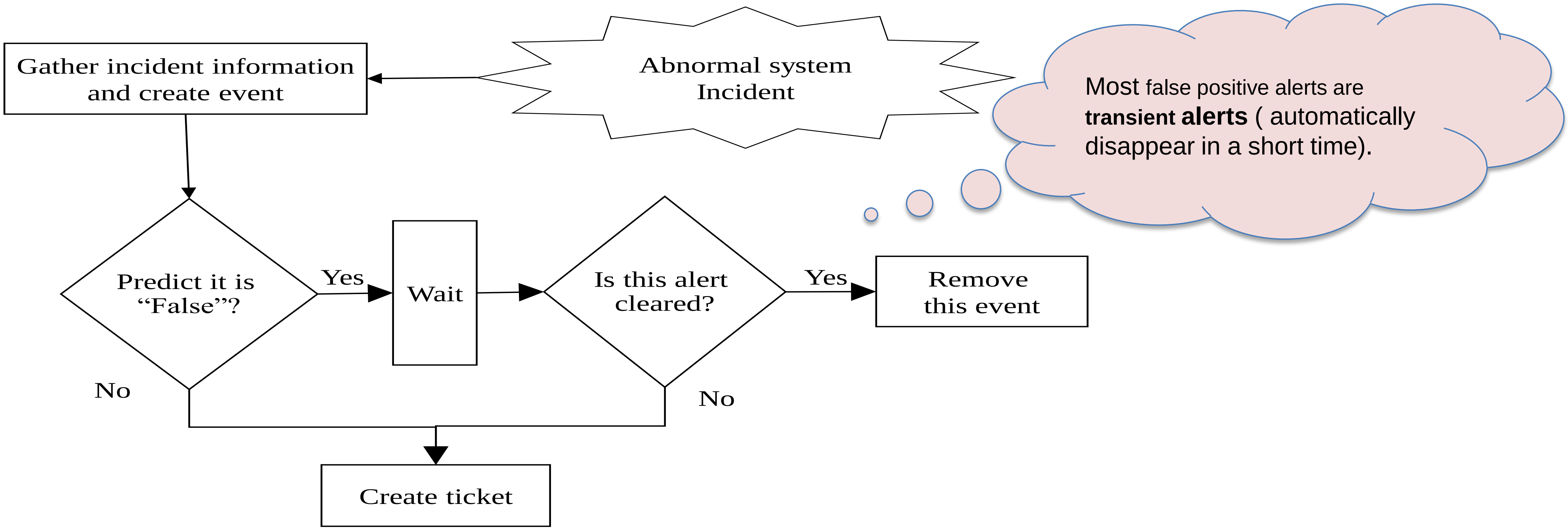
Motivation of Eliminating False Positives

- Most false positives are transient alerts and automatically disappear in a short time.



Some transient alerts may be indications of future real alerts and may be useful. But if those real alerts rise later on, the monitoring system will detect them even if the transient alerts were ignored.

Workflow



Waiting time is the maximum duration of covered false positives

$$wait_p = \max_{e \in \mathcal{F}_p} e.duration, \qquad \mathcal{F}_p = \{e | e \in \mathcal{F}, isCovered(p, e) =' true'\},$$

Implementation and Deployment

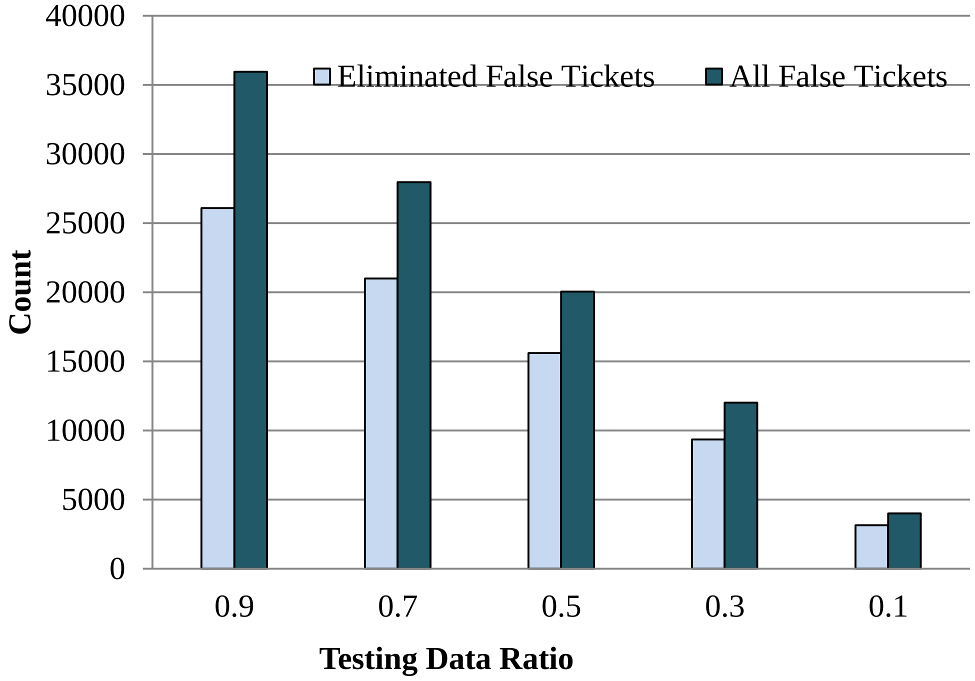
- The rules generated by a classifier can be directly translated into monitoring situations:
  - If PROC_CPU_TIME > 50% and PROC_NAME = 'Rtvscan', then it is false.

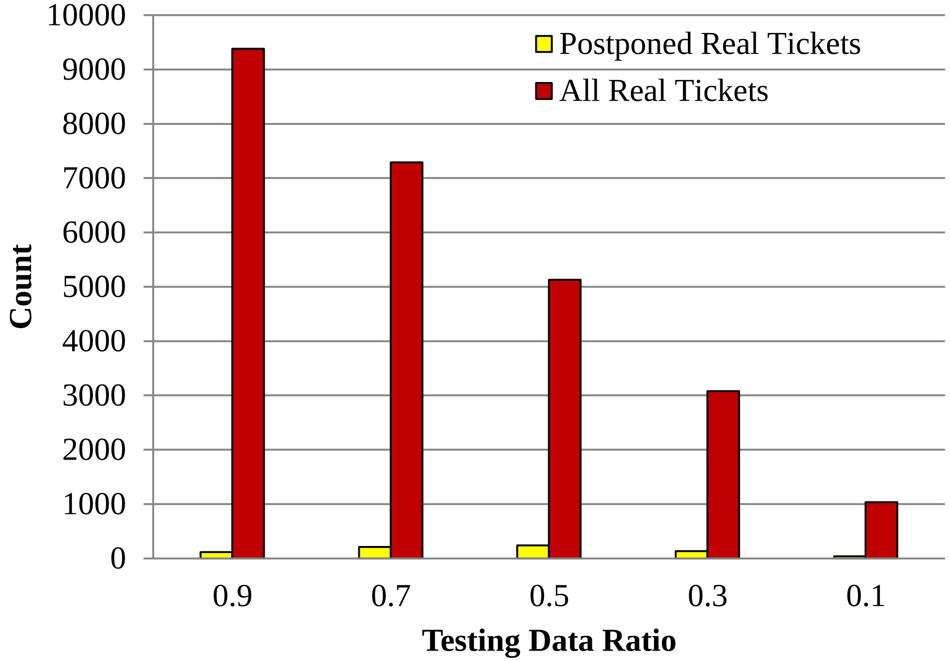
Predictor

- *Waiting time* is the polling interval of a monitoring situation.

Waiting Time

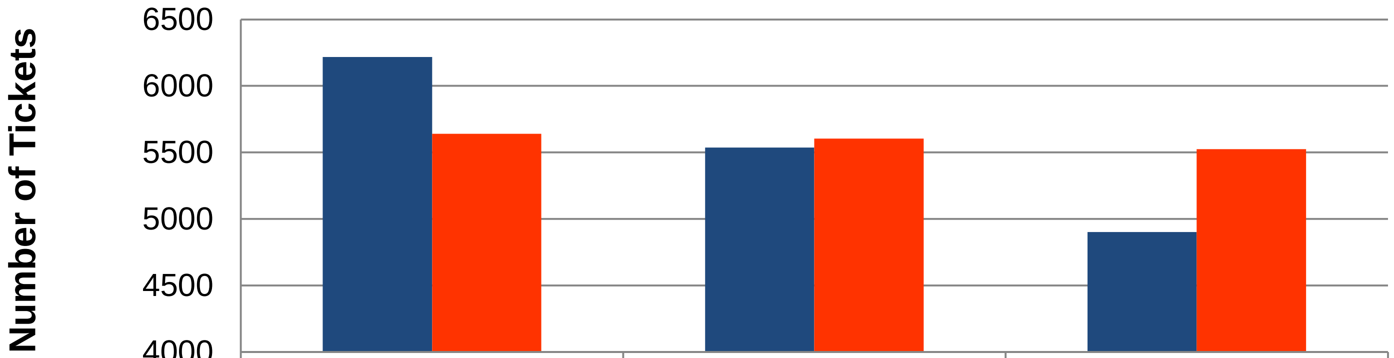
Offline Evaluation on Historical Data





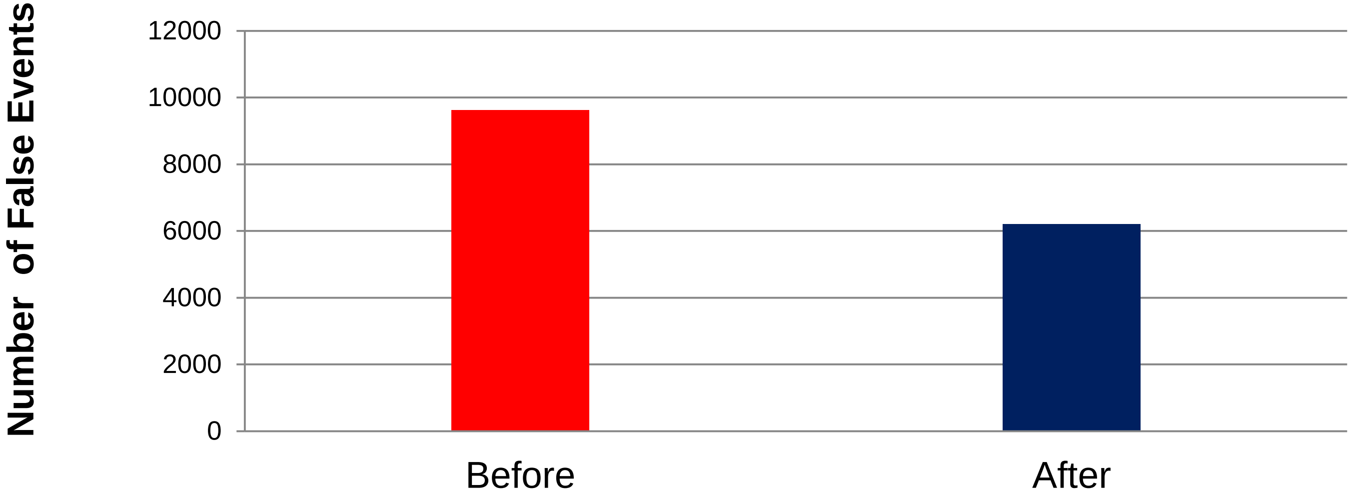
Ratio of the testing data size and training data size

Online Evaluation



	October 2011	November 2011	Januray 2012
False Tickets	6217	5536	4901
Real Tickets	5639	5605	5524

A large financial company.



An internal account in IBM.

What is False Negative (Missed Alert) ?

- False negatives are the missed alerts by the monitoring system.

- False negatives are usually captured by human (customers, helpdesk, system administrators).

- False negatives are not recorded in monitoring events, but only in manual tickets.

Why We Have False Negatives?

- New devices and software are installed, but are not added into the monitoring configurations

- Other changes for existing systems. Some thresholds may not be acceptable after changes.

About False Negative

- How to eliminate false negatives (missed alerts)?
  - False negative are quite few (less than 20-40 tickets for a situation).
  - No need an automatic approach to correct the misconfiguration.

- False negatives are missed alerts. Where can we find them?
  - **Manual Tickets (captured by human)**.
  - However, manual tickets contain other kinds of tickets, such as customer request.

Automatically identify related manual tickets and then refine the configuration

Problem Statement

- Eliminate false negatives by refining the monitoring configurations

- It consists of two parts:
  - Scan the historical manual tickets and provide a short list of potential false negatives to the monitoring team (automatically)
  - Change or add monitoring situations (manually)

Related Work

- **Reduce False Negative**
  - Focus on improving the accuracy of the monitoring
  - No prior work is based on discovery of false negatives (Because false negatives are missed alerts. There is no data record for tracking them).
- **Text classification**
  - Class label “1”: a missed alert; class label “0”: other issues, such as customer request. Features are the words in the ticket description.
  - Imbalanced classification: Cost-sensitive and over-sampling.

Two-Stage Text Classification

- A simple classification to rank all tickets based on their confidence of being false negative.
  - a simple word match algorithm based on given *domain words* (labeled features)

Situation Issue	Words
DB2 tablespace Utilization	DB2, tablespace
File System Space Utilization	space,file
Disk Space Capacity	space,drive
Service Not Available	service,down
Router/Switch Down	router

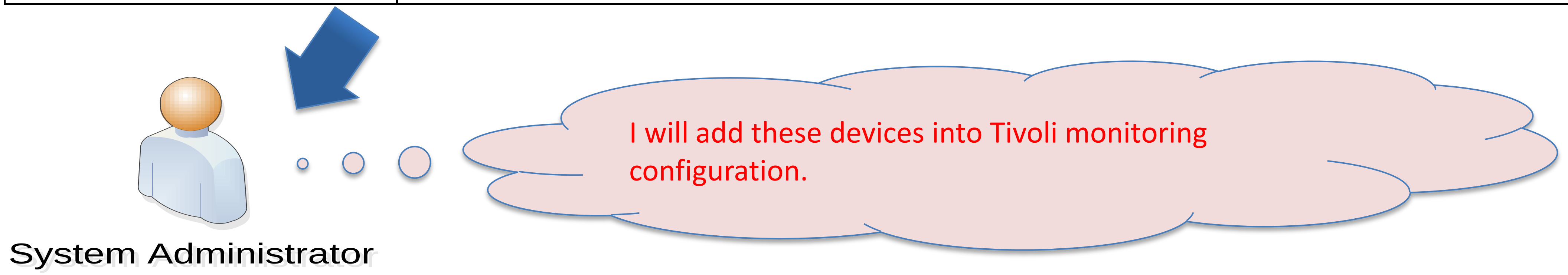
- Only select top ranked tickets for labeling and training and build the final text classifier.
  - Build a binary SVM classifier.

Avoid labeling all tickets and save the labeling cost.

A Case Study

Discovered False Negatives (Missed alerts)

Situation	**Ticket**
dsp_3ntc_std	*Please clear space from E drive xxxx-fa-ntfwwfdb Please clear space from E drive xxxx-fa-ntfwwfdb.it is having 2 MB free...*
fss_rlzc_std	*/opt file system is is almost full on xxx Hi Team@/opt file system is almost full. Please clear some space /home/dbasso>df -h /optFilesystem...*
svc_3ntc_std	*RFS101681 E2 Frontier all RecAdmin services are down Frontier RecAdmin services are not running on the batch server Kindly logon to the server : xxx.xxx.155.183/xxx ...*
…	…



Optimizing Monitoring Configurations based on Events and Tickets

(Liang Tang, Tao Li et. al, IEEE/IFIP NOMS 2012)

(Liang Tang, Tao Li et. al, CNSM 2013)

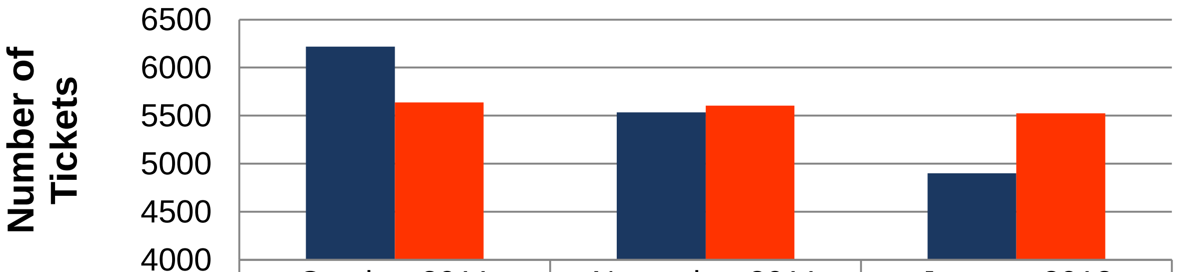
false positive

CPU_UTIL> 80%, *Duration* = 1 minute

Optimize

(1) *CPU_UTIL* > 80% and *PROCESS_NAME* = 'Rtvscan.exe' , *Duration* = 15 minutes

(2) *CPU_UTIL* > 80% and *PROCESS_NAME* <>'Rtvscan.exe' , *Duration* = 1 minute



	October 2011	November 2011	Januray 2012
False Tickets	6217	5536	4901
Real Tickets	5639	5605	5524

false negative

Situation	Ticket
dsp_3ntc_std	*Please clear space from E drive xxxx-fa-ntfwwfdb Please clear space from E drive xxxx-fa-ntfwwfdb.it is having 2 MB free...*
fss_rlzc_std	*/opt file system is is almost full on xxx Hi Team@/opt file system is almost full. Please clear some space /home/dbasso>df -h /optFilesystem...*
svc_3ntc_std	*RFS101681 E2 Frontier all RecAdmin services are down Frontier RecAdmin services are not running on the batch server Kindly logon to the server : xxx.xxx.155.183/xxx ...*
…	…

Contents

1 Introduction and Overview

2 Event Generation and System Monitoring

3 Pattern Discovery and Summarization

4 Mining with Events and Tickets

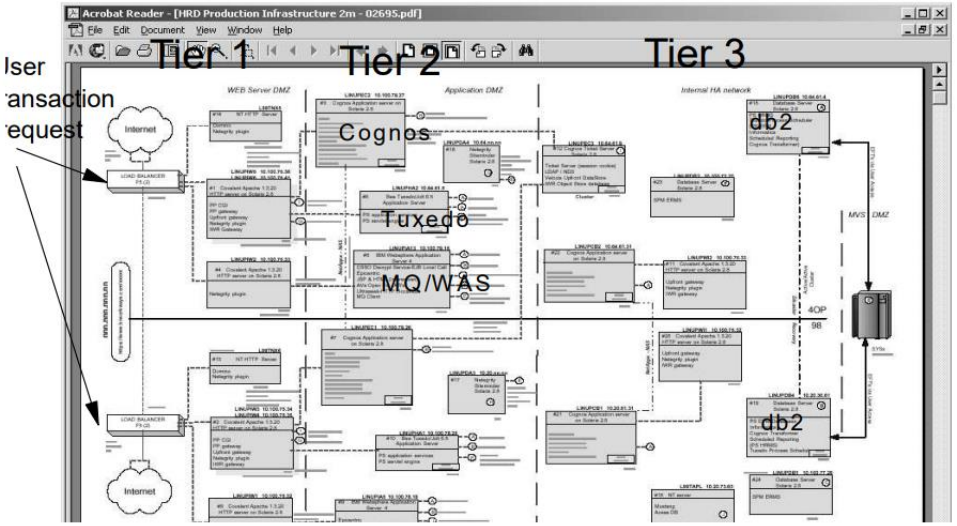
5 Conclusions

Outline

- History on Event Mining
- Overview of Temporal Patterns
- Mining Time Lags
  - Non-parametric Methods
  - Parametric Methods
- Event Summarization
- Temporal Dependency

History of issue and some apps

- Issue: complex cross platform, multiple applications working together, how to insure everything is working proper?
- Approach:
  - Consider system footprint of each component by querying system
  - Consider subcomponents and query subcomponents on working conditions
  - Evaluate subcomponent logs on status of components
  - Consider customer complains (tickets)
- The first two have unified monitoring solution
- Logs, tickets should be “preprocessed” for analysis
- Start with ‘visualization’ of logs, tickets
- Use unsupervised learning to deal with large volumes of informati



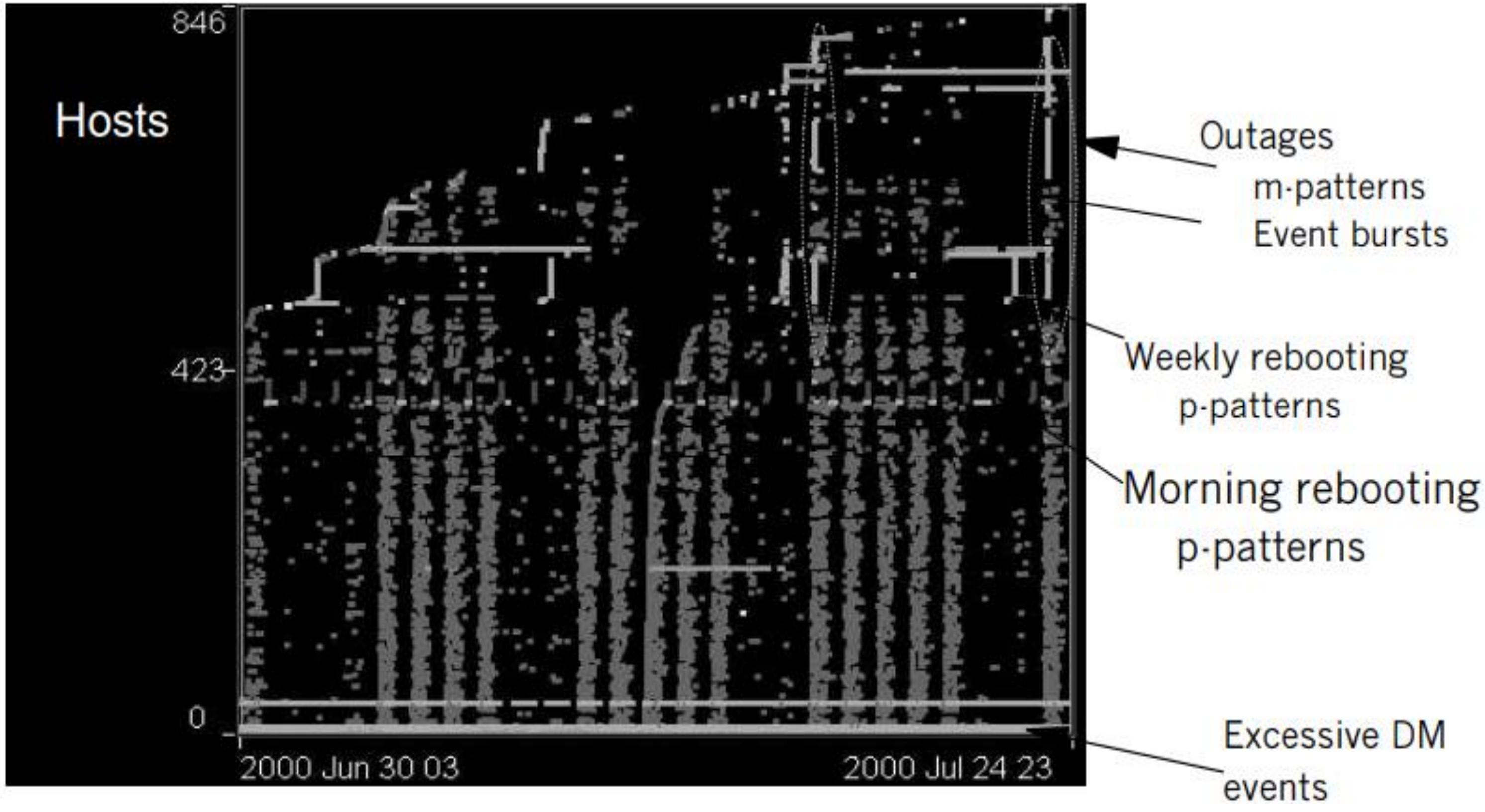
- Information generated by systems are events - have timestamp o occurrence
  - Rarely interdependent events has clear transaction like start and end
  - Mainly has vague ‘time of start’ and ‘time of end’

Events IDA (Interactive Data Analysis)

- EventBrowser, a few versions,
  - Started by S. Ma (Google), J. Hellerstein (?,UW), Visual C++
  - Features: In memory events storage, interactive query building, visualization, 3 views
  - Next version: Re-implemented on top of visualization framework, over 20 different views
    - Diamond by D. Rabenhorst (?), added rich visualization, better column base in memory storage, extended querying (including color based) capability,
    - D. Taylor (UWaterloo) added visual querying graphics->SQL->Diamond API->modified view;
- Good for initial intuition development
- Issues typical for IDA,
  - low throughput,
  - inconsistent by different people usage

Event Mining

- A number of events patterns was suggested, showing need to proceed
- To overcome IDA limitation used combined method: Build tool providing both IDA and Data Mining capabilities
- Event Miner was built (GG+S.Ma) on top of Diamond Framework, integrated data mining (frequent datasets, others) with visualization, round trip patterns search etc.
- Helped identify 20++ patterns types



Pattern Discovery I

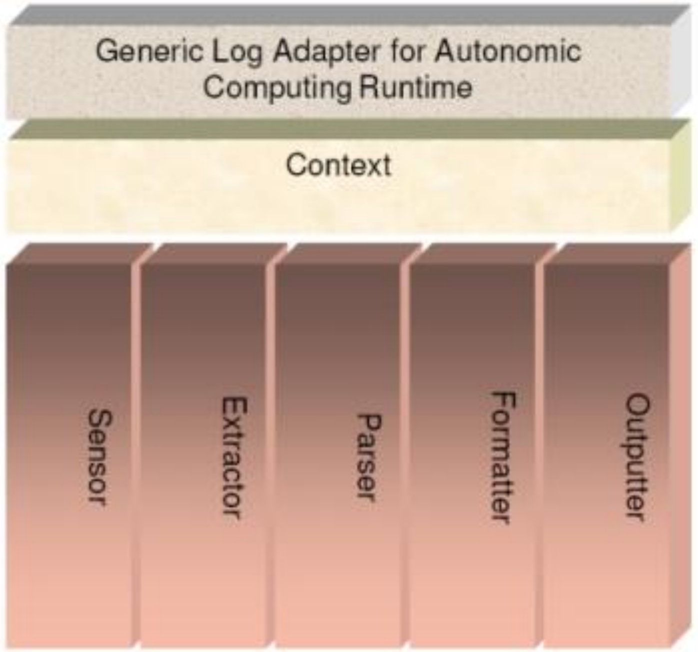
Sequential Pattern ([19],[181],[200],[111],[207],[81],[237],[27],[99],[174],[173] [162],[59])	Event sequences	Frequent event subsequences, e.g., $< \{A\}, \{B, C\} >$.	All the subsequences with occurrence frequency not less than a given threshold are discovered. Two categories of algorithms are presented, i.e., Apriori-based and pattern-growth-based algorithms.
Fully Dependent Pattern([141])	An event database	All the items in a pattern are correlated with each other, e.g., {A,B,C} is a fully dependent pattern iff any of its subsets is a fully dependent pattern.	Hypothesis test is applied for identifying the correlation of items in a pattern.
Partially Periodic Dependent Pattern ([152])	An event sequence	Periodic pattern with period p and tolerance δ, e.g., $A \rightarrow_{[p-\delta, p+\delta]} A$.	Periodic patterns are discovered from a given event sequence, where the periodic patterns happen on some segments of the sequence, rather than on the whole sequence. The partially periodic dependent pattern is identified by chi-squared hypothesis test.
Mutually Dependent Pattern([151])	An event sequence	Events in a mutually dependent pattern $\{A, B\}$ depend on each other, i.e., $A \rightarrow B$ and $B \rightarrow A$.	Mutually dependent patterns are identified if the conditional probabilities in both directions are greater than a predefined minimum dependence threshold.

Pattern Discovery II

T-Pattern([133, 134])	An event sequence	Patterns like $A \rightarrow_{[\tau-\delta,\tau+\delta]} B$ are discovered, where τ is the time interval and δ is the tolerance.	T-Pattern is defined on two events, indicating that an event implies the other one within a time interval.
Frequent Episode ([157, 158, 16, 15, 170])	An event sequence	Given window size p, an episode containing event pattern is frequent if its frequency is not less than a predefined threshold.	Three types of frequent episodes include the serial episode, the parallel episode, and the composite episode.
Event Burst([121, 201, 235, 164])	An event sequence	Event burst is defined over a period $[t_1, t_2]$ if the occurrence frequency of a given event is high.	The event burst detection can be used for monitoring the occurrence of a significant event automatically.
Rare Event([222])	An event sequence	Given a rare event T, a prediction rule is produced like $\{A, B\} \rightarrow E$.	An anomaly is typically a rare event. The prediction rule can be used to predict the anomaly according to historical events.
Correlation between Time Series and Event ([150])	An event sequence and a time series	Given an event E and a time series S, patterns like $S \rightarrow E$ or $E \rightarrow S$ are produced.	Such patterns are useful in practice, for example, the correlation between CPU usage and running a computing job.

Data Feed from Logs

- Capability of Event Miner allowed to process large amount of data
- By hand processing of logs was not sufficient anymore
- To provide appropriate feed and partially automate process Generic Log Adapter (GG, SM, AS (IBM) BS(Microsoft), with contr. CP(Google) was built
- Used inverse of control on top of piping architecture to provide extensible framework
- Eclipse based GUI for interactive log pattern development
- Was able to semi-automate log processing of many applications/components
- Contributed to Eclipse foundation TFTP framework
- Back to Event Mining

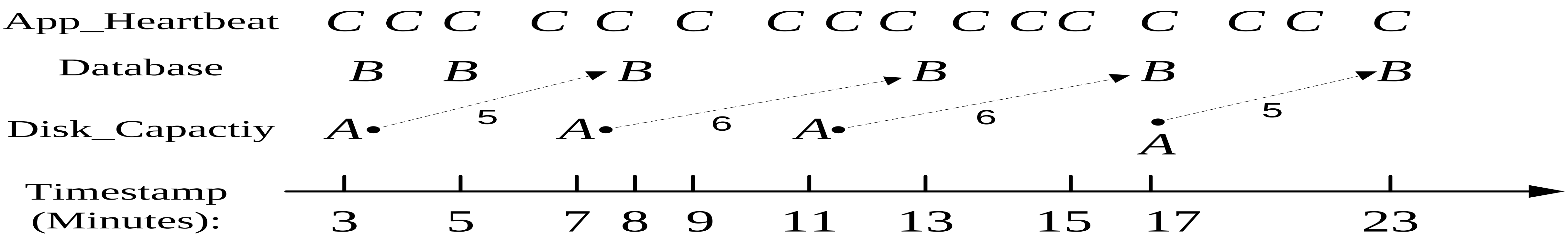


Outline

- History on Event Mining
- Overview of Temporal Patterns
- Mining Time Lags
  - Non-parametric Methods
  - Parametric Methods
- Event Summarization
- Temporal Dependency

Mining Event Relationships

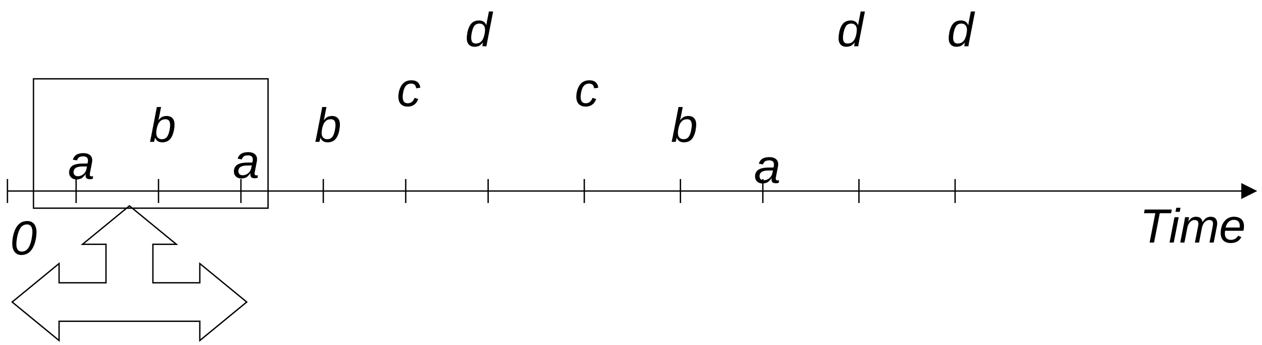
- Temporal Patterns (of System Events)
  - Wish: A sequence of symptom events providing a signature for identifying the root cause
  - Less ambition: ‘repeating’ (sub)sequences of events
    - ***Host Restart***: “host is down” followed by “host is up” in about 10 seconds
    - ***Failure Propagation***: “a link is cut”→ “connection loss” → “lost connection” → “application terminated unexpectedly”
- Examples of Temporal Dependency



- Disk_Capacity $\longrightarrow_{[5min,6min]}$ Database, [5min, 6min] is the lag interval.
- Reflects hidden process, here may be database inserts/updates, expect normality here

Issues in Temporal Data Mining

- Temporal correlation of events
  - **Previous work**
    - *Concept of transactions*
    - *Fix Time Sliding Window schemes*
  - **Problems**
    - *Size of windows*
    - *Can not mine temporal relationships longer than the window size,*
    - *time window varies with pattern*
    - *In our experiments, time distances range from one second to one day*
  - **Approach: distance methods**



- Characteristics of interesting patterns
  - **Previous work**: *Frequent patterns -- (normal operations)*
  - **Problems**
    - *Infrequent, but significant patterns -- (service disruptions)*
    - *Noisy environments*
    - *Time dimension*
  - **Approach: statistical dependency of inter-arrival times**

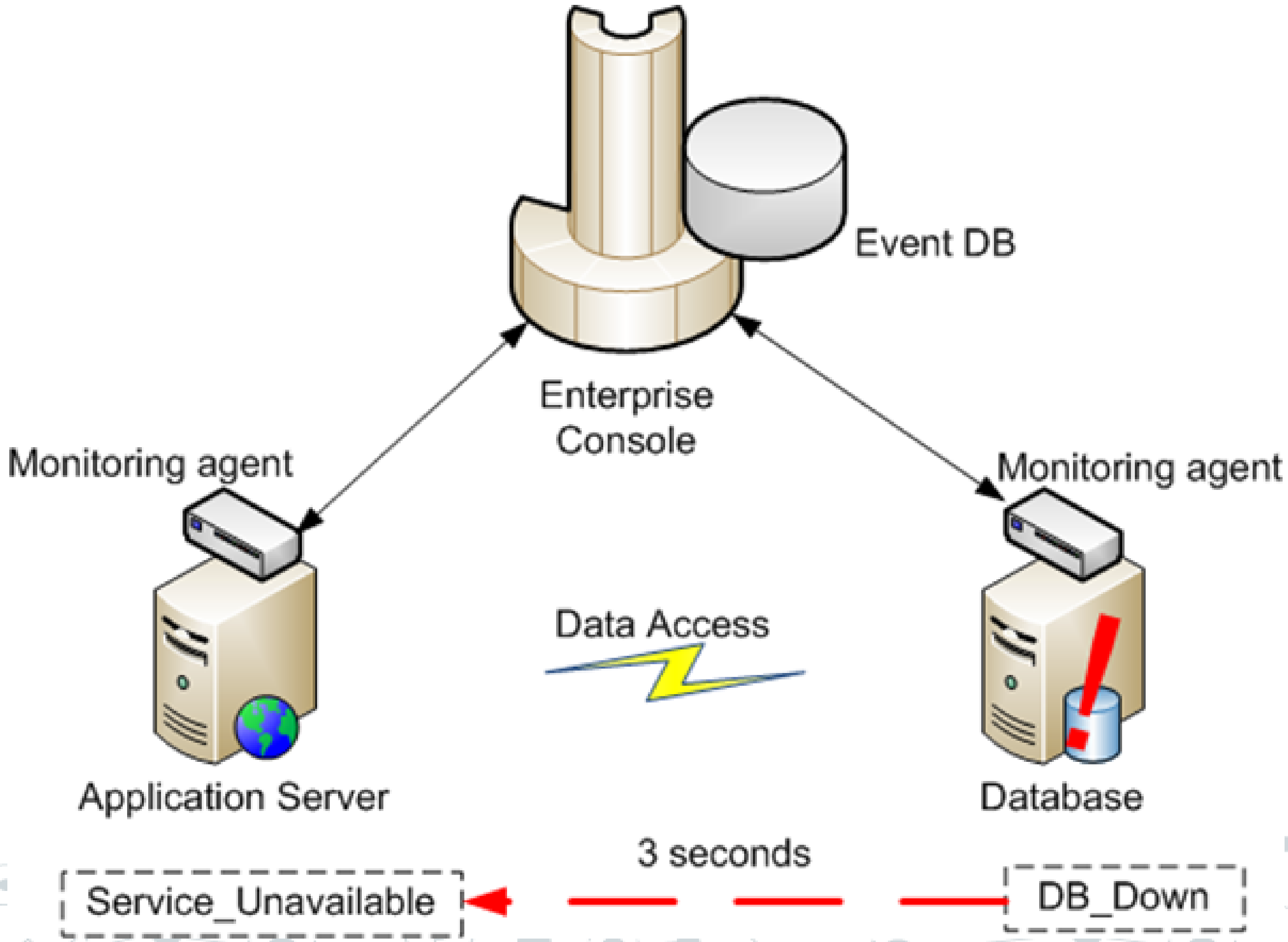
[Li et al., KDD 2004; Peng et al, KDD 2007; Zeng et al., 2015]

Outline

- History on Event Mining
- Overview of Temporal Patterns
- Mining Time Lags
  - Non-parametric Methods
  - Parametric Methods
- Event Summarization
- Temporal Dependency

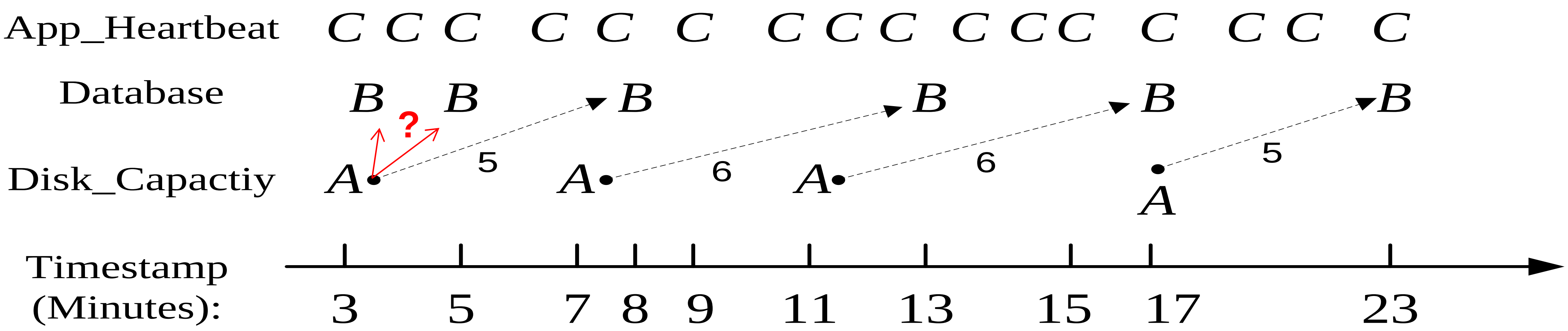
Time Lag

- The dependency between events helps for problem diagnosis
- Time lag plays an important role in predicting the incoming events and the evolving trends of systems’ behavior.



Preliminary Work

- Predefine the lag interval (H. Mannila et al., 1997)
- No interleaved dependency (T. Li et al., 2005, K. Bouandas et al., 2007)



Disk_Capacity $\longrightarrow_{[5min,6min]}$ Database, [5min, 6min] is the lag interval.

Example of disambiguation issue: Which B event depends on the first A event?

Relation with Other Temporal Patterns

Those temporal patterns can be seen as the temporal dependency with particular constraints on the time lag.

Mutually Dependent	$\{A,B\}$	$A \rightarrow_{[0,t]} B, B \rightarrow_{[0,t]} A$
Partial Periodic	A with periodic p and time tolerance δ	$A \rightarrow_{[p-\delta, p+\delta]} A$
Frequent Episode	A->B->C	$A \rightarrow_{[0,t]} B, B \rightarrow_{[0,t]} C$
Loose Temporal	B follows A before t	$A \rightarrow_{[0,t]} B$
Stringent Temporal	B follows A about t	$A \rightarrow_{[t-\delta, t+\delta]} B$

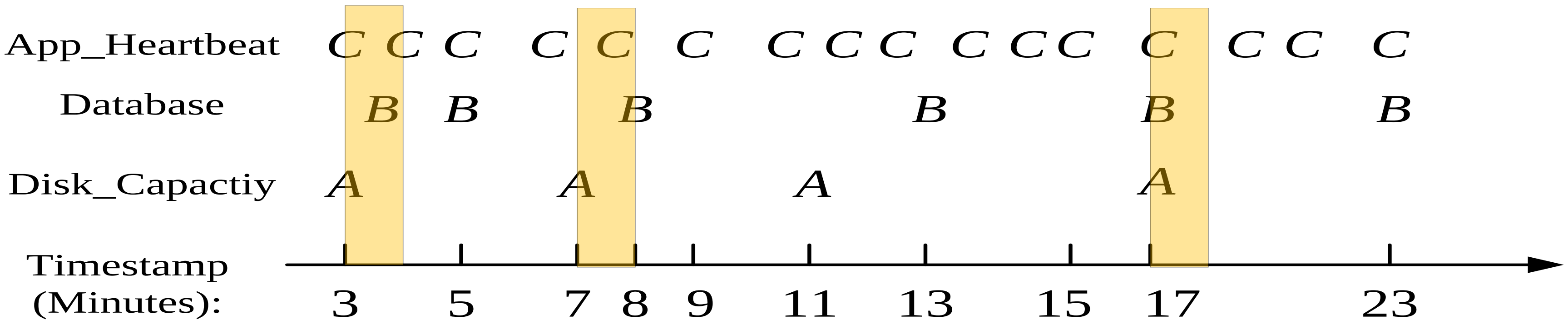
Challenges for Finding Time Lag

- Given a temporal dependency, $A \longrightarrow_{[t1,t2]} B$, what kind of lag interval [t1,t2] we want to find?
  - If the lag interval is too large, every *A* and every *B* would be "dependent".
  - If the lag interval is too small, real dependent *A* and *B* might not be captured.

- Time complexity is too high.
  - $A \longrightarrow_{[t1,t2]} B$, $t1$ and $t2$ can be any distance of any two time stamps. There are $O(n^4)$ possible lag intervals.

What is a Qualified Lag Interval

- If [t1,t2] is qualified, we should observe many occurrences for $A \longrightarrow_{[t1,t2]} B$.



Lag Interval	Number of Occurrences
[0,1]	3
[5,6]	4
[0,6]	4
[0,+∞]	4

Length of the lag interval is larger, the number of occurrences also becomes larger.

What is a Qualified Lag Interval

- Intuition (Statistical Testing):
  - If *A* and *B* are randomly and independently distributed, how many occurrences observed in a time interval [t1,t2]?

Expected value

- What is the minimum number of occurrences (threshold)?
  - Consider the number of occurrences in a lag interval to be a variable, n_r. Then, use the *chi-square* test to judge whether it is caused by randomness or not?

$$\chi_r^{\ 2} = \frac{(n_r - n_A P_r)^2}{n_A P_r (1 - P_r)}$$

$$P_r = |r| \frac{n_B}{T}$$

The number of *A*s

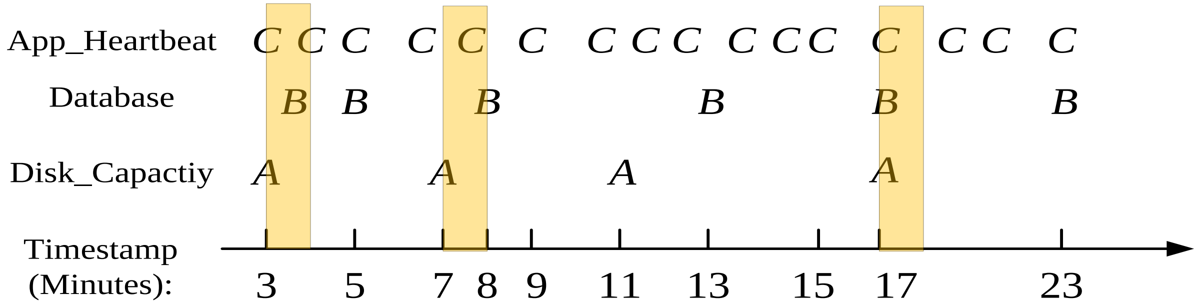
Total time length of the event sequence

Naive Algorithm for Finding Qualified Lag Intervals

- **(Brute-Force) Algorithm**: For $A \longrightarrow_{[t1,t2]} B$, for every possible $t1$ and $t2$, scan the event sequence and count the number of occurrences.
- Time Complexity
  - The number of distinct time stamps is $O(n)$.
  - The number of possible $t1$ and $t2$ is $O(n^2)$ (building distribution of t1,t2, linear space).
  - The number of possible $[t1,t2]$ is $O(n^4)$.
  - Each scan is $O(n)$. The total cost is $O(n^4)$.
- Cannot handle large event sequences.

Discovering Lag Interval for Dependent Events

(Liang Tang, Tao Li et. al, ACM SIGKDD 2012)



Lag Interval	Number of Occurrences
[0,1]	3
[5,6]	4
[0,6]	4
[0,+∞]	4

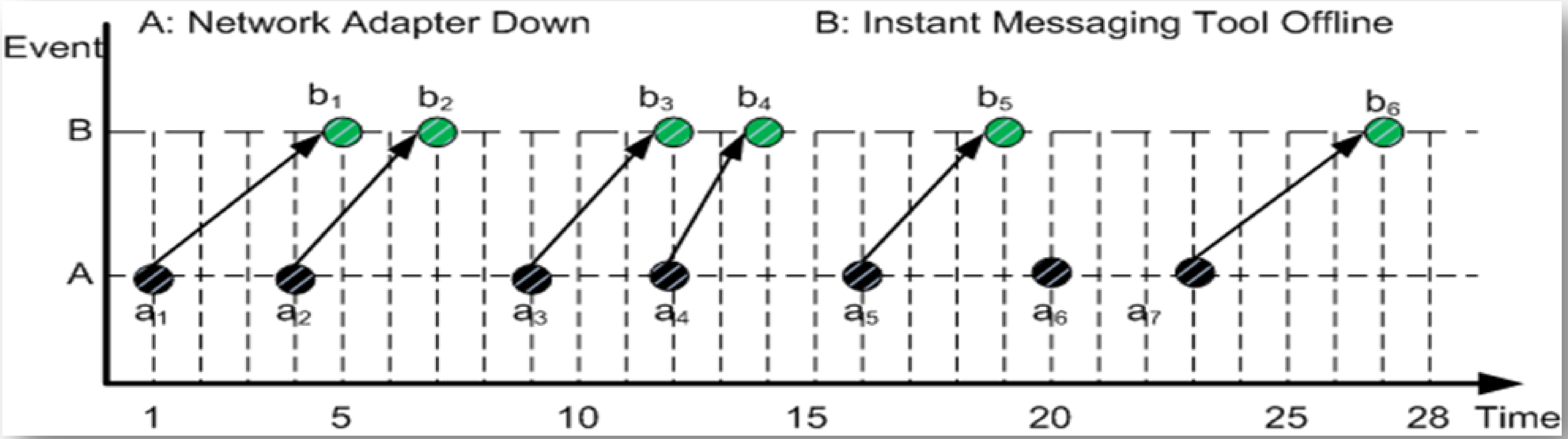
Length of the lag interval is larger, the number of occurrences also becomes larger.

Two experimental data sets from IBM customer monitoring events

Dataset	**Discovered Dependencies**
Account1	*MSG_Plat_APP* $\longrightarrow_{[3600,3600]}$ *MSG_Plat_APP*
	Linux_Process $\longrightarrow_{[0,96]}$ *Process*
	SMP_CPU $\longrightarrow_{[0,27]}$ *Linux_Process*
Account2	*TEC_Error* $\longrightarrow_{[0,1]}$ *Ticket_Retry*
	TEC_Retry $\longrightarrow_{[0,1]}$ *Ticket_Error*
	AIX_HW_ERROR $\longrightarrow_{[8,9]}$ *AIX_HW_ERROR*

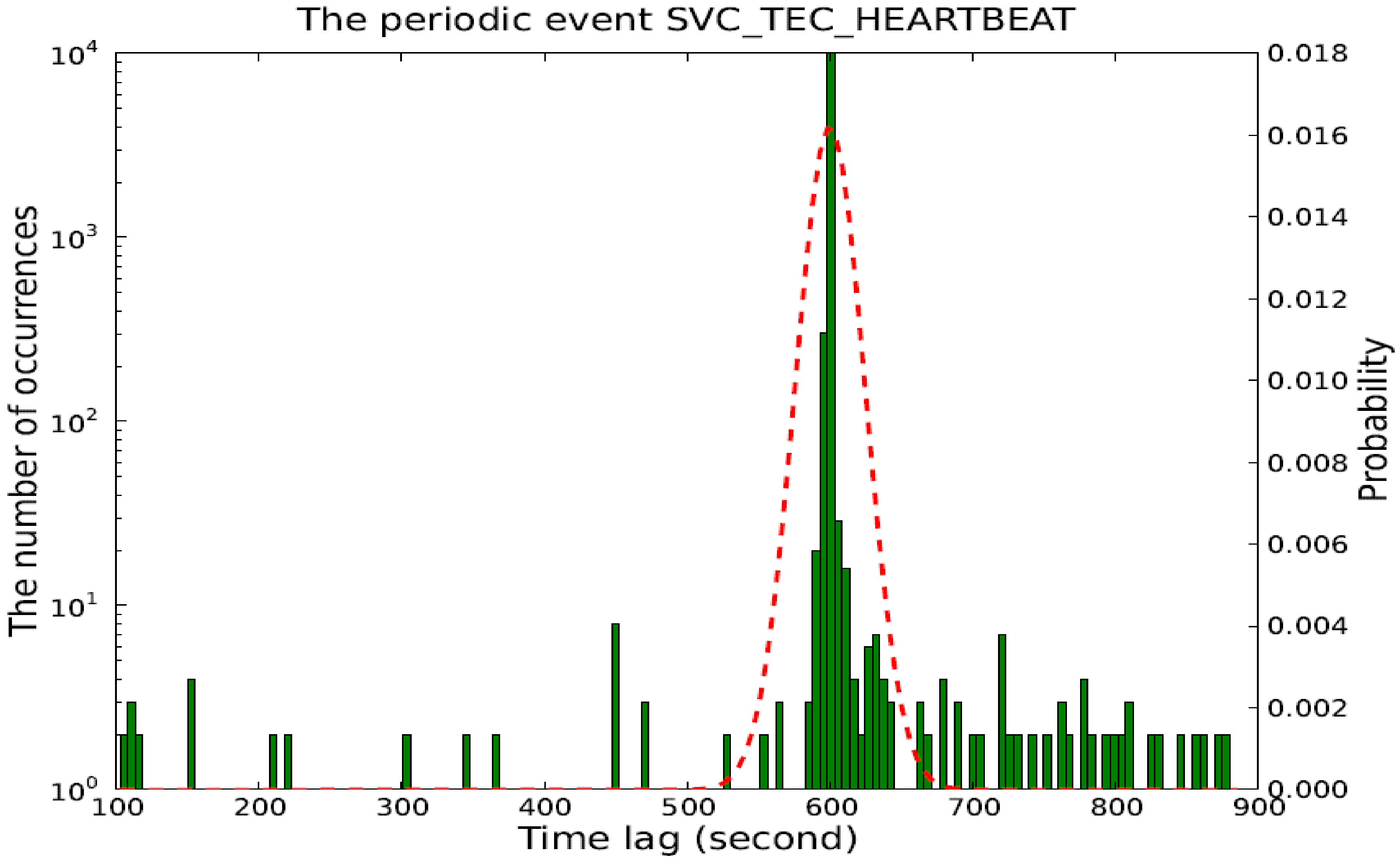
- The interleaved temporal dependency makes difficult to correct mapping between two events

- Noise leads to fluctuating time lag.



- A parametric model to formulate the randomness of time lags between events. This model is capable of
  - ✕ providing insight into the correlation of events.
  - ✕ describing the distribution of time lags.

Observation: Time Lag distribution for periodic event



- Given two events A and B, let μ be the true lag if A implies B. Then it can be denoted as:

$$\boxed{A} \longrightarrow_{\mu} \boxed{B}$$

- Let ε be the noise. Then the time lag observed is modeled as a random variable L which comprises μ and ε.

$$L = \mu + \varepsilon$$

Time Lag Modeling

- We assume ε follows Gaussian distribution with variance σ^2.

$$\varepsilon \sim \mathrm{N}(0, \sigma^2)$$

- As the result, the time lag is modeled as L, which comprises μ and ε, follows the Gaussian distribution.

$$\boxed{\mathrm{A}} \xrightarrow[L]{} \boxed{\mathrm{B}} \qquad L \sim \mathrm{N}(\mu, \sigma^2)$$

Time Lag Mining

■ Given two events A and B, our problem is reduced to learn the distribution of L. We need to determine:

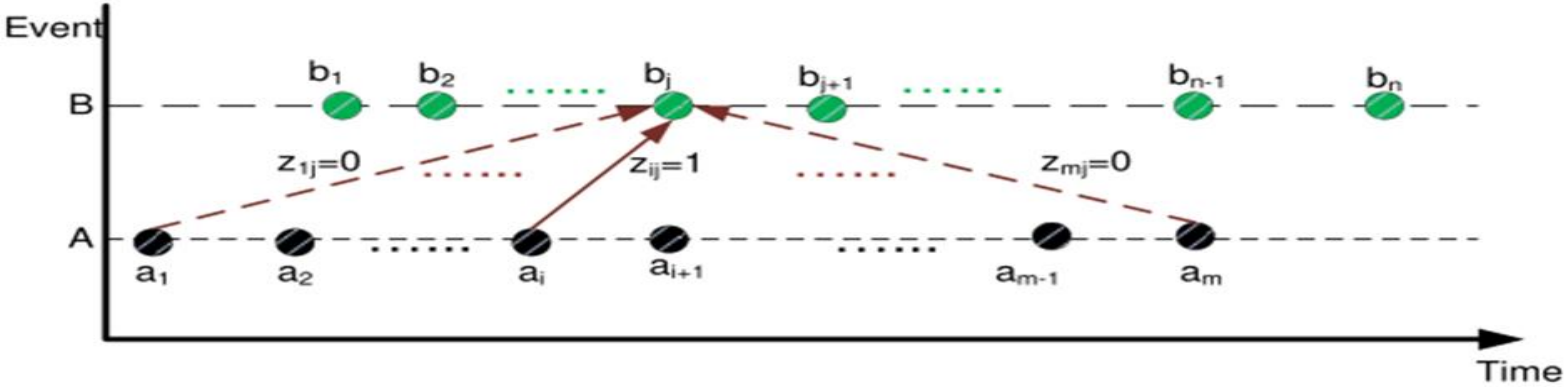
1.Parameter μ.

2.Parameter σ^2.

$$\boxed{A} \xrightarrow[L]{} \boxed{B} \qquad L \sim N(\mu, \sigma^2)$$

Problem Formulation

- Another problem is disambiguation, that there is no idea about which instance of event A implies a specific instance of event B.



$$z_{ij} = \begin{cases} 1, & \text{the } i^{th} \text{ event } A \text{ implies the } j^{th} \text{ event } B; \\ 0, & \text{otherwise.} \end{cases}$$

$\pi_{ij} \equiv P(Z_{ij} = 1)$, satisfying $\sum_{i=1}^{m} \pi_{ij} = 1$

Solution

Let S_A and S_B are the sequences of event A and event B respectively, a mixture model is proposed to formulate the log likelihood of the event data.

$$\ln P(S_B \mid S_A, \mu, \sigma^2) = \sum_{j=1}^{n} \ln \sum_{i=1}^{m} \pi_{ij} \times N(b_j - a_i \mid \mu, \sigma^2)$$

The parameters can be learnt by maximizing the log likelihood.

$$(\hat{u}, \hat{\sigma}^2) = \arg \max_{\mu, \sigma^2} \ln P(S_B \mid S_A, \mu, \sigma^2)$$

The problem can be solved with EM algorithm.

Algorithm(LagEM)

➢ LagEM is an EM-based algorithm, which mainly involves two parts: Expectation and Maximization.

1. Initialization

2. Loop until converge

1) Expectation:

$$r_{ij} = \frac{\pi'_{ij} \times N(b_j - a_i|\mu', \sigma'^2)}{\sum_i^m \pi'_{ij} \times N(b_j - a_i|\mu', \sigma'^2)}.$$

2) Maximization:

$$\mu = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{m} r_{ij}(b_j - a_i),$$

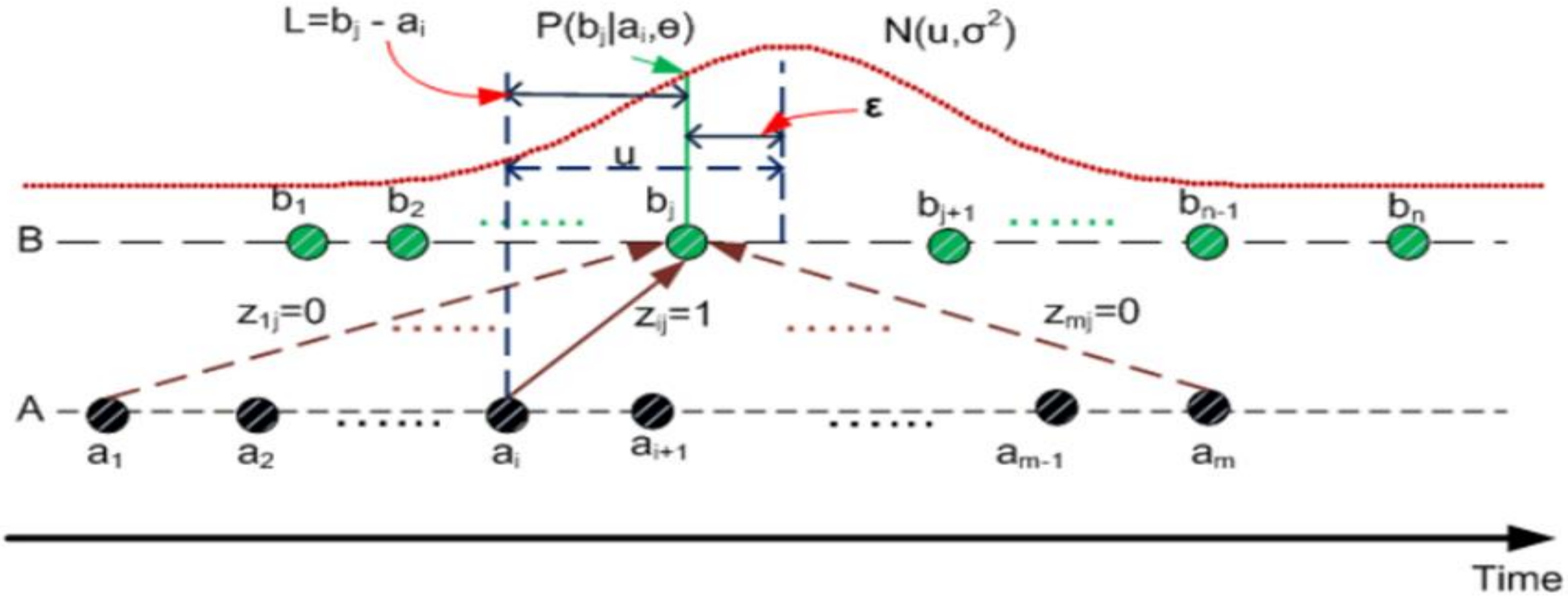
$$\sigma^2 = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{m} r_{ij}(b_j - a_i - \mu)^2.$$

$$\pi_{ij} = r_{ij}.$$

The time complexity is **O(r*n*m)**, where **r** is the number of iterations, **m** and **n** are the numbers of event **A** and event **B**, respectively.

Solution improvement

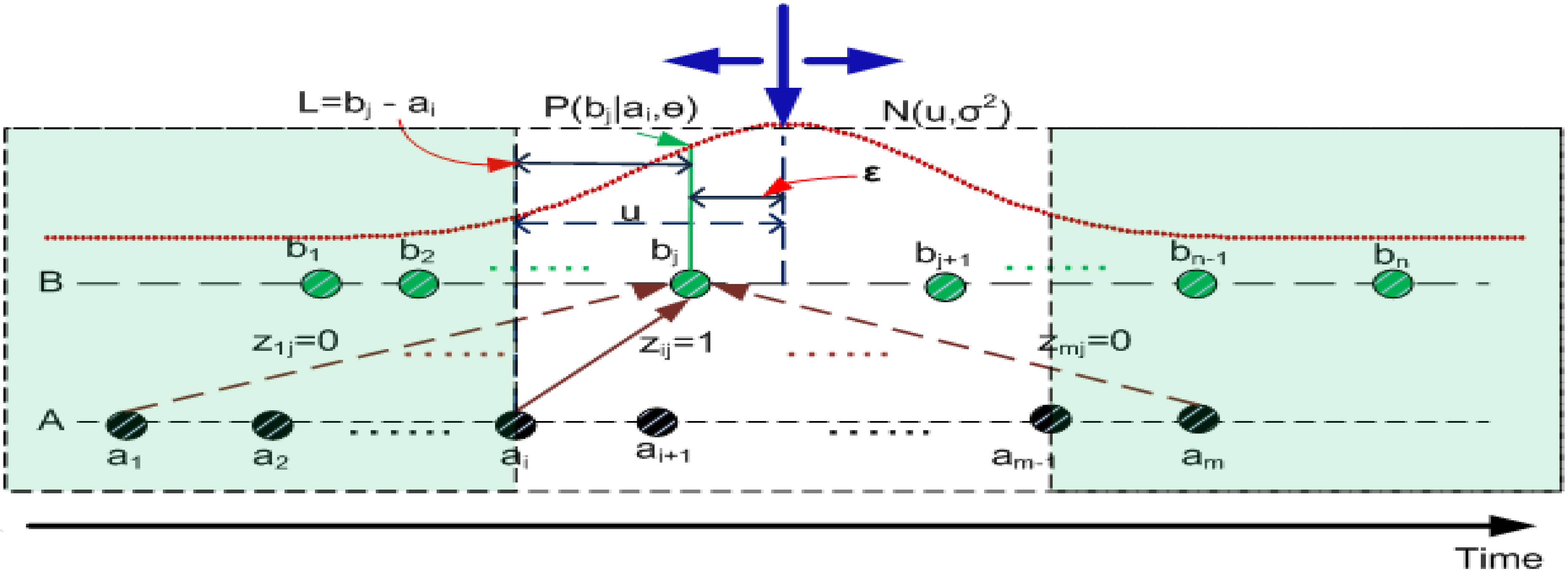
➢Observation: the most possible b_j ,which is implied by a_i, should be around the time $t=a_i + u$. The further b_j deviates from t, the less probable it is.



Approximation algorithm

- We only consider the most probable K pairs and neglect all others.

- Find the most possible time point and then search the possible time points by looking left and right until the proportion of the considered points exceeds 1-ε, where ε is very small, like 0.05.



Experimental Result

➢ Setup

- Synthetic data: noise and true lags are incorporated into data.
- Provided with ground truth, the experiment conducted on synthetic data allows us to demonstrate the effectiveness.
- Provided with different numbers of synthetic events, it allows to illustrate the efficiency of our algorithm.

➢ Real data: data is collected from several IT outsourcing centers by IBM Tivoli monitoring system.

- It shows that temporal dependencies with time lags can be discovered by running our proposed algorithm.
- Detailed analysis demonstrates the effectiveness and usefulness of our method in practice.

Experiment on synthetic data

➢KL(Kullback-Leibler) divergence is used to measure the difference between the distribution of time lag given by the ground truth and the discovered result.

- The KL divergence caused by appLagEM is almost as small as the one produced by LagEM

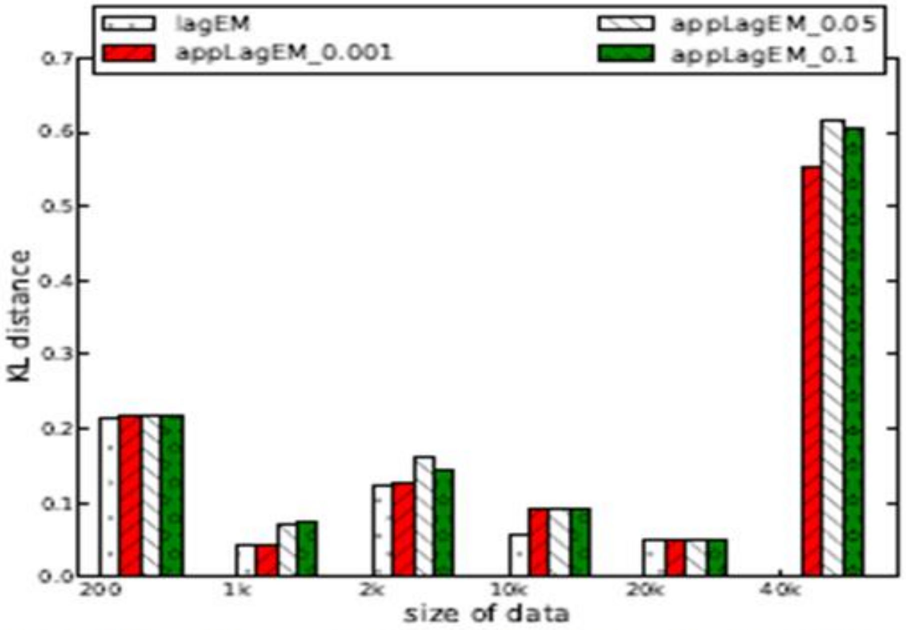


Fig. 4: The KL distance between the ground truth and the one learnt by each algorithm.

Experiment on synthetic data

➢The comparison of time cost over the svnthetic data is shown.

- appLagEM is much more efficient than lagEM
- The larger the ϵ is, the less time appLagEM takes to find the optimal distribution of the time lags
- Algorithm appLagEM with ϵ =0.001 is about two orders of magnitude faster than lagEM

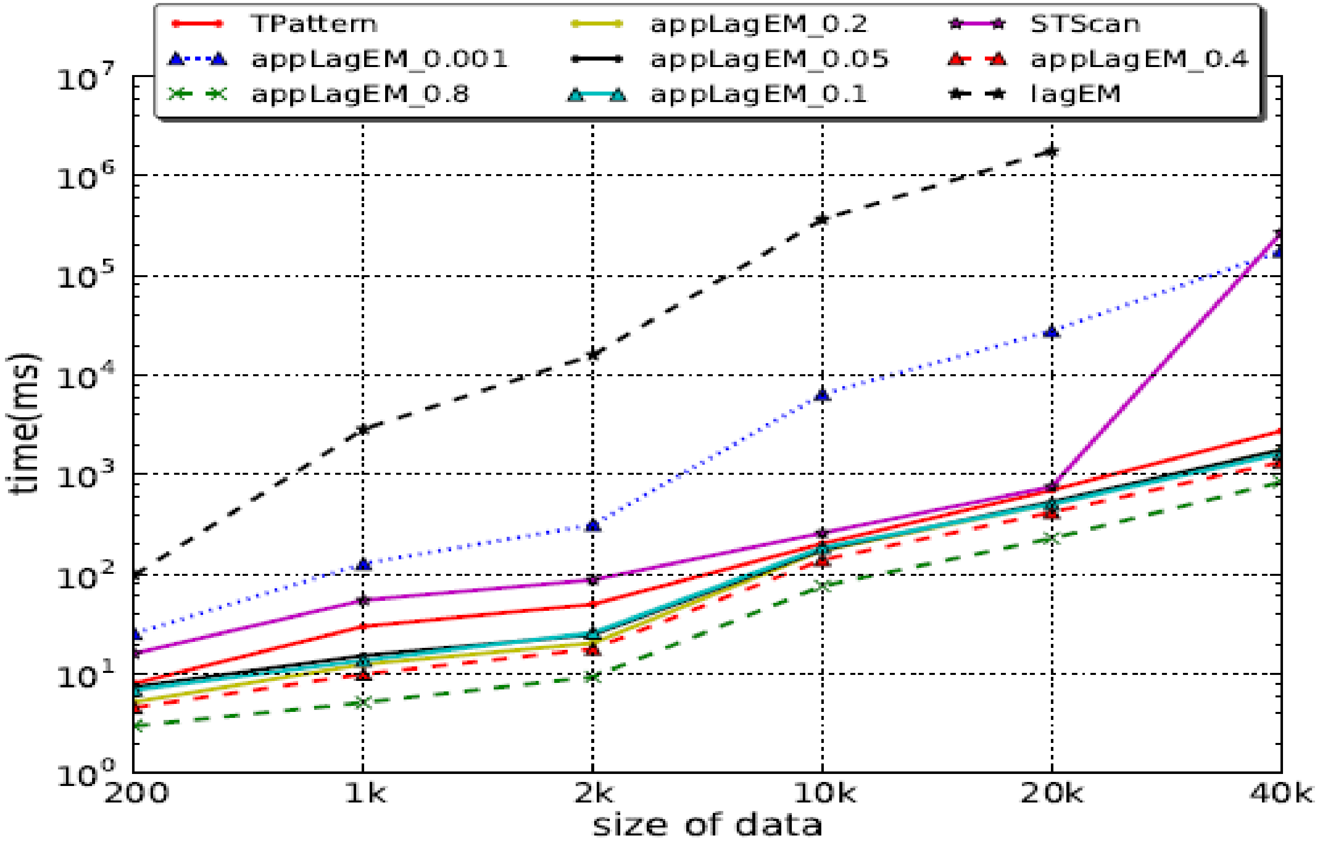


Fig. 9: Time cost comparison. ξ of $appLagEM$ is set with 0.001, 0.05, 0.1, 0.2, 0.4, 0.8. The existing algorithm for mining TPattern and the algorithm STScan for mining time intervals are from [30] and [4] respectively. The size of data set ranges from 200 to $40k$.

Experiment on real data

- The experiment is conducted over two real data sets from the IT outsourcing centers by IBM Tivoli monitoring system.

TABLE 5: Real event data set.

Name	# of events	# of types	Time span
dataset1	$100k$	104	32 days
dataset2	$1000k$	136	54 days

- Since there are large number of events in both two data sets, lagEM is infeasible. The algorithm appLagEM with ϵ =0.001 is used to mine the time lag of temporal dependency.

➢ We Apply the metric signal-to-noise ratio to filter the dependencies discovered by appLagEM.

➢ The Larger the SNR is, the less relative impact of noise to the expected time lags.

$$SNR = \frac{\mu}{\sigma}$$

Experiment on real data

- A snippet of interesting temporal patterns are highlighted in the below table.
  - It shows that our algorithm can find patterns with time lags of different scales.
  - The distributions of time lags present the confidence of the temporal dependencies.
  - The periodic patterns can be discovered by the proposed algorithm.

	Dependency	μ	σ^2	Signal-to-noise ratio
dataset1	$TEC\_Error \rightarrow_L Ticket\_Retry$	0.34059	0.107178	1.04
	$AIX\_HW\_ERROR \rightarrow_L AIX\_HW\_ERROR$	10.92	0.98	11.03
	$AIX\_HW\_ERROR \rightarrow_L NV390MSG\_MVS$	33.89	1.95	24.27
	$AIX\_HW\_ERROR \rightarrow_L Nvserverd\_Event$	64.75	2.99	37.45
	$AIX\_HW\_ERROR \rightarrow_L generic\_postemsg$	137.17	18.81	31.63
	$generic\_postemsg \rightarrow_L TSM\_SERVER\_EVENT$	205.301	39.36	32.72
	$generic\_postemsg \rightarrow_L Sentry2\_0\_diskusedpct$	134.51	71.61	15.90
	$MQ\_CONN\_NOT\_AUTHORIZED \rightarrow_L TSM\_SERVER\_EVENT$	1167.06	142.54	97.75
dataset2	$MSG\_Plat\_APP \rightarrow_L Linux\_Process$	18.53	2053.46	0.408
	$SVC\_TEC\_HEARTBEAT \rightarrow_L SVC\_TEC\_HEARTBEAT$	587.6	7238.5	6.90

Interesting temporal patterns are discovered by our algorithm.

- TEC_Error $\dashrightarrow_L$ Ticket_Retry, where $L \sim N(0.34, 0.107178)$. It indicates that the two events appear almost at the same time. In fact, TEC_Error is caused whenever the monitoring system fails to generate an incident ticket to the ticket system. And Ticket_Retry is raised when the monitoring system tries to generate the ticket again.
- AIX_HW_Error $\dashrightarrow_L$ AIX_HW_Error, where $L \sim N(10.92, 0.98)$. It shows a periodic pattern with 10 seconds. In real environment, the event AIX_HW_Error is raised when monitoring system polls an AIX server which is down, The failure to respond to the monitoring system leads to an event AIX_HW_Error almost every 10 seconds.

Outline

- History on Event Mining
- Overview of Temporal Patterns
- Mining Time Lags
  - Non-parametric Methods
  - Parametric Methods
- Event Summarization
- Temporal Dependency

Event Summarization - Introduction

What is Event Summarization?

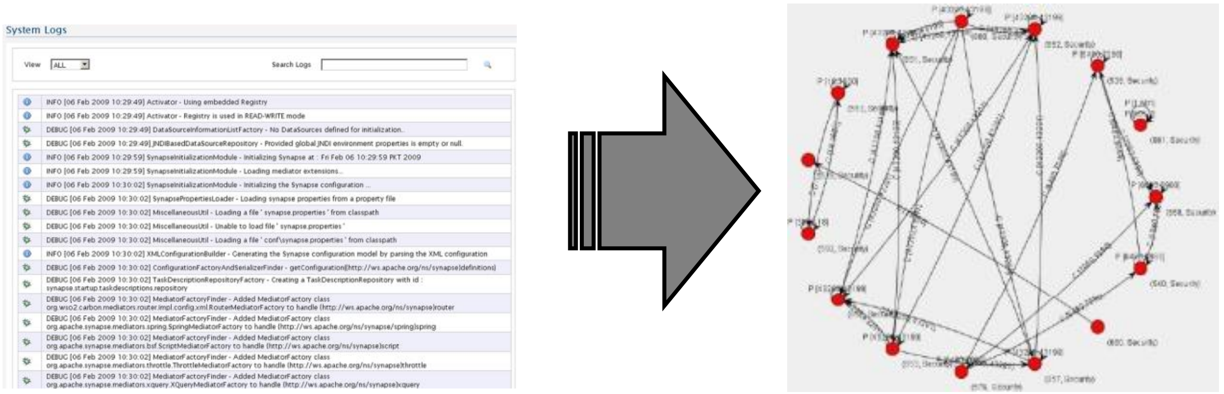
The techniques that provide a concise interpretation of the seemingly chaotic data, so that domain experts can take actions upon the summarized models.

Why summarize?

Traditional data mining algorithms output too many patterns.

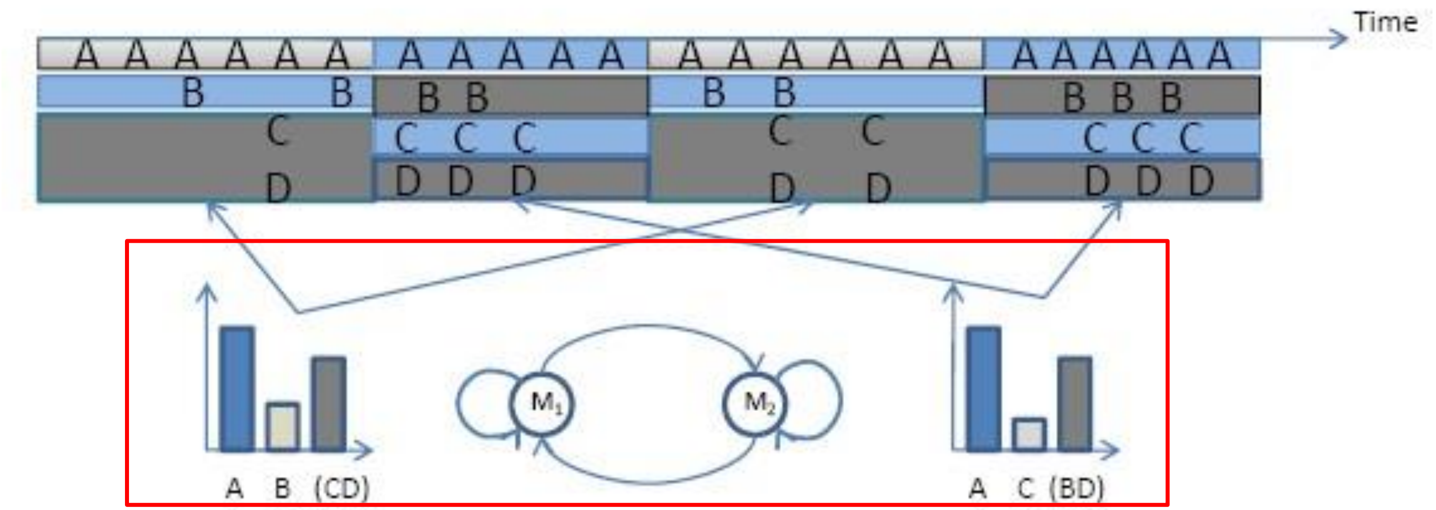
Properties of event summarization

- Brevity and accuracy
- Global data description
- Local pattern identification
- Minimize number of parameters



Existing Summarization Solutions

- Summarize events with frequency change segments

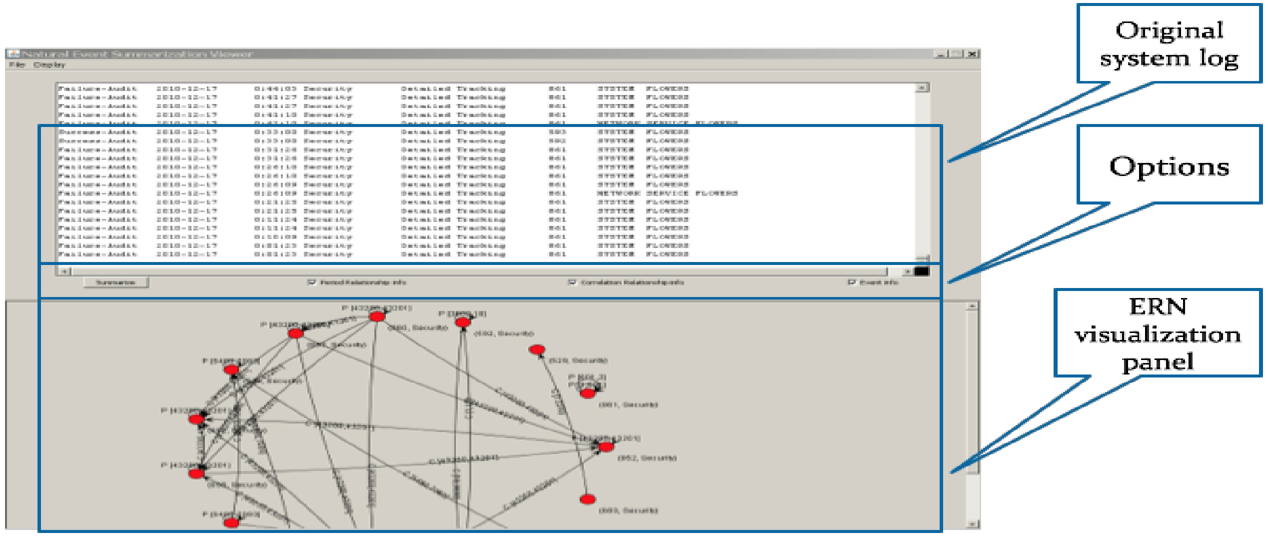


- Ignore temporal information within segments
- All segments have the same boundaries
- The generated summary is not easy to understand

Event Summarization (Jiang et al., CIKM 2011)

Solution

- Summarize with temporal patterns./dynamics
- Modeling temporal patterns with interval histograms.
- Leverage MDL to balance accuracy and brevity.
- Represent summarization result with ERN.



workflow

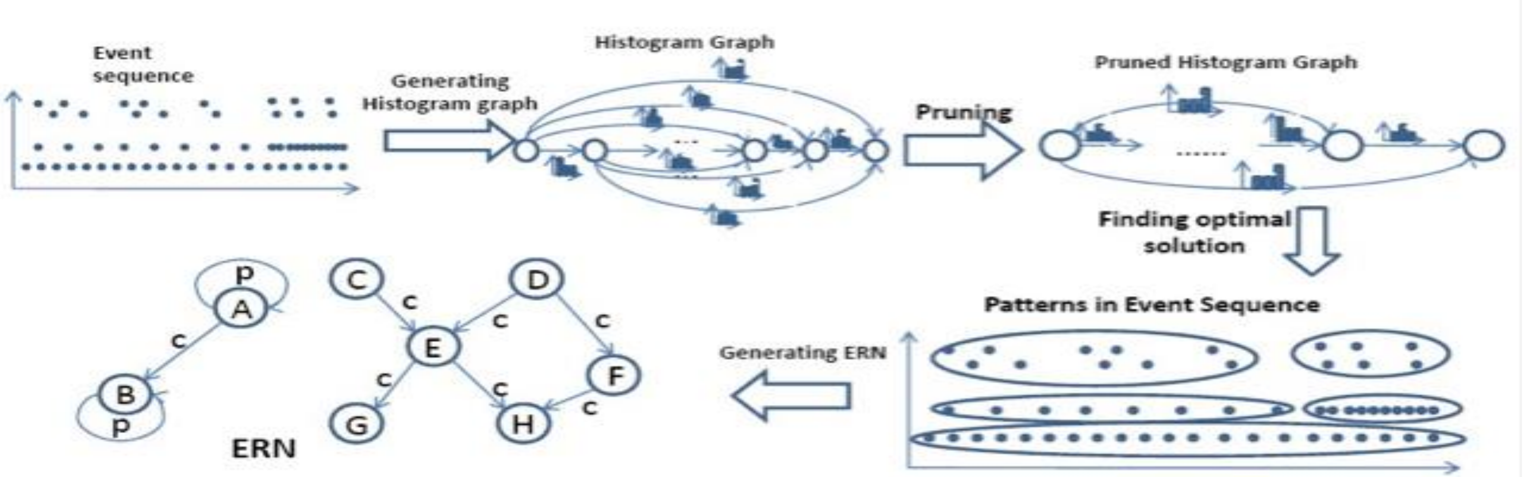


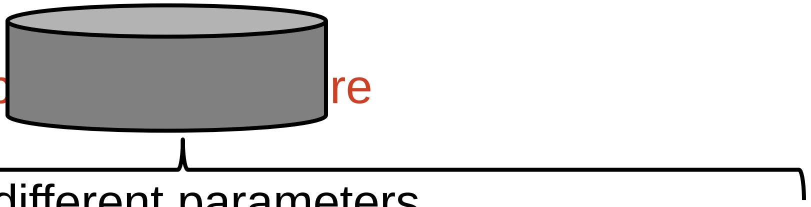
Table VI. A brief summary of the event summarization methods

Paper	Category	Description
Peng, Perng & Li, 2007 [Peng et al. 2007]	Temporal Dynamics	Using a correlation graph ERN to summarize the correlation between events.
Kiernan & Terzi, 2008 [Kiernan and Terzi 2008]	Frequency Change	Using segmentation to summarize changes over time and using the event frequency group to summarize events within each time period.
Aharon et al., 2009 [Aharon et al. 2009]	Other	Clustering the events and using the clusters as the summary.
Kiernan & Terzi, 2009 [Kiernan and Terzi 2009]	Frequency Change	Similar to [Kiernan and Terzi 2008], but allowing mismatch among segments.
Wang et al., 2010 [Wang et al. 2010]	Frequency Change	Extension of [Kiernan and Terzi 2008]. Using the Markov model to represent the transition between segments.
Schneider et al., 2010 [Schneider et al. 2010]	Temporal Dynamics	Using a graph to represent the relations of *AlwaysFollowedBy*, *AlwaysPrecededBy*, and *NeverFollowedBy* among events.
Jiang, Perng & Li, 2011 [Jiang et al. 2011]	Temporal Dynamics	A richer form of [Peng et al. 2007]. Summarizing the events from the perspective of periodic patterns and correlation patterns.
Tatti & Vreeken, 2012 [Tatti and Vreeken 2012]	Temporal Dynamics	Summarizing the events using a set of serial episodes under the guidance of MDL.

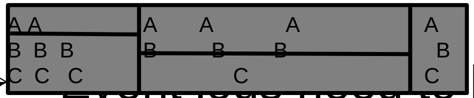
Multi-Resolution Event Summarization (Jiang et al., SDM 2014)

- Existing works focus on algorithmic solutions
- Event Summarization is a Rep... re
  - ➢ Different models have different parameters
  - ➢ Different users have diff...
  - ➢ Different applications ne... in differen...
  - ➢ Event logs need to be updated

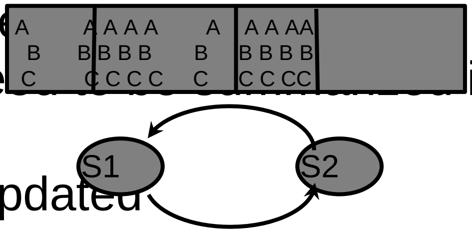
event log



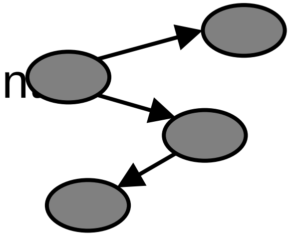
summary view



kiernan et al. 2008

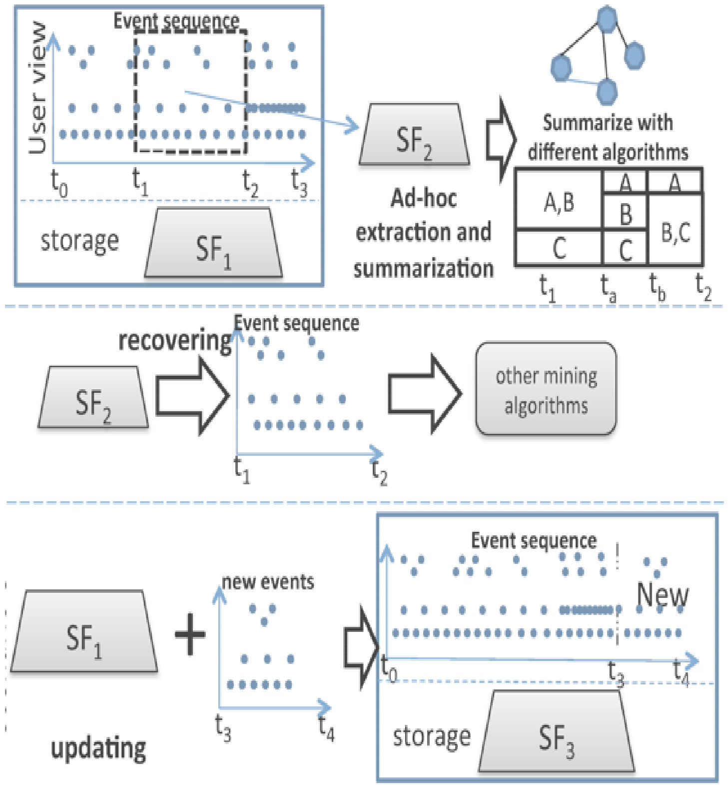


Wang et al. 2010



Jiang et al. 2011

- **A Multi-resolution Summarizaion Framework**
  - ➢ **10 Basic operators:** Vectorize, Unvectorize, Encode, Decode, Prune, Concatenate, Project, Select, Zoom, Describe.
  - ➢ **5 Tasks:** Summarization, Storing, Recovering, Merging, and Updating

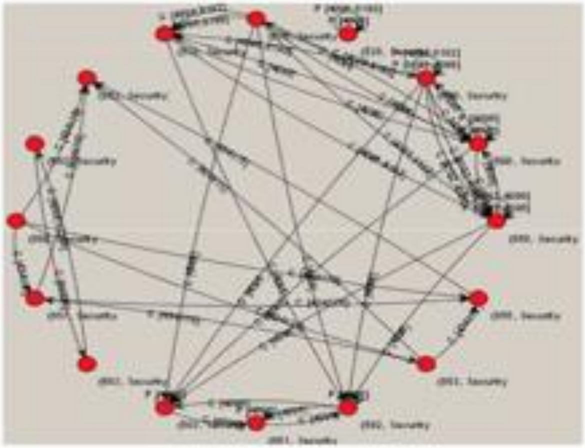


Illustrative Example

Task 1

Summarize the security-win log in day granularity

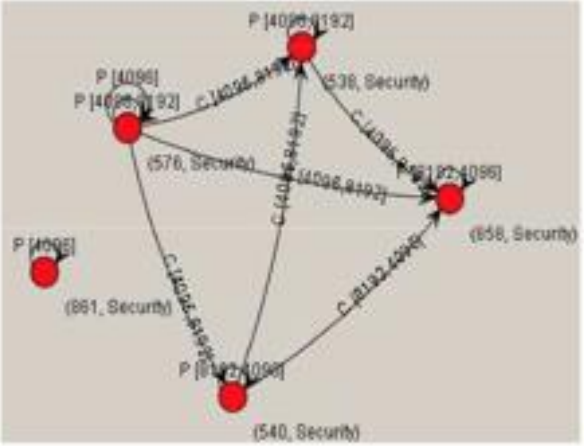
(1) **store** *security-win* **as** *SSF* **with resolution** *13*
(2) **describe** *SSF* **zoom to resolution** *16*



Task 2

Drill down to view summary with event type (538, 540, 576, 858 and 861) between 11/01/2011 and 11/29/2011 in hour granularity

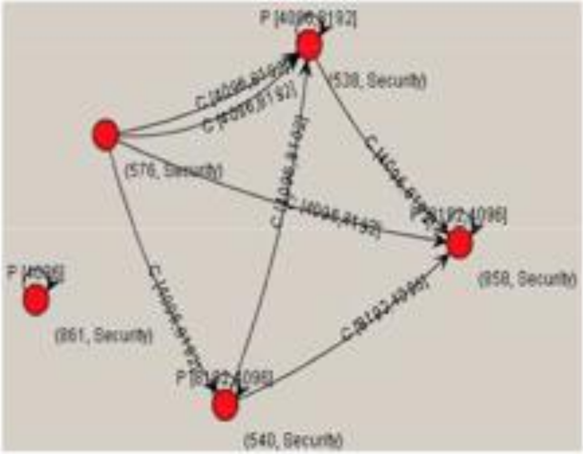
(3) **describe (select** *538,540, 576, 858, 861* **from** *SSF* **between** *01/11/2011* **and** *29/11/2011* **zoom to resolution** *6*)



Task 3

Update the summary with new log recorded from 11/30/2011 to 01/04/2012 and then summary them altogether in minute granularity

(4) **update** *SSF* **with** *new-security-log*
(5) Execute (3) again by changing resolution to 13

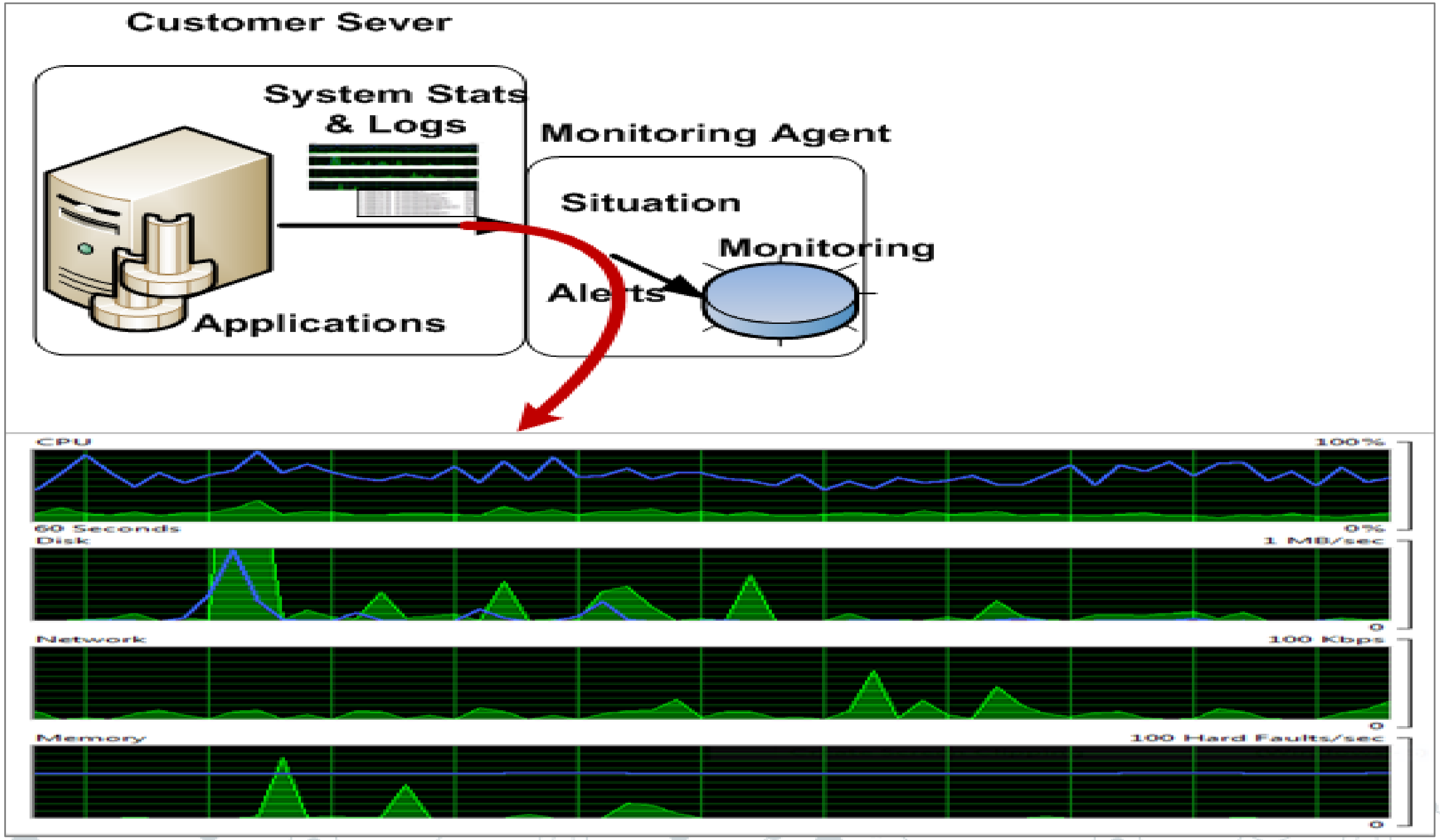


Outline

- History on Event Mining
- Overview of Temporal Patterns
- Mining Time Lags
  - Non-parametric Methods
  - Parametric Methods
- Event Summarization
- Temporal Dependency

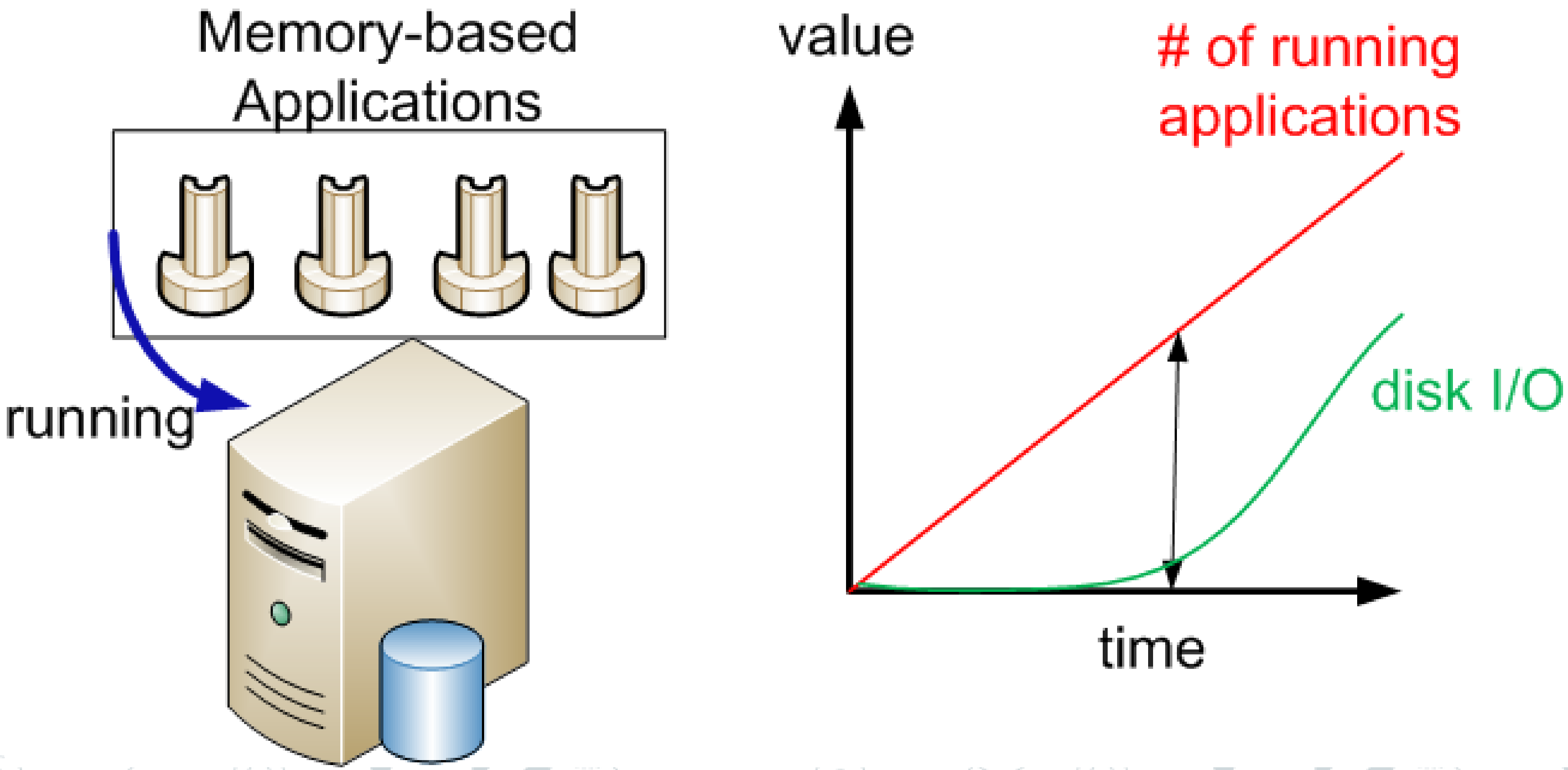
Problem and Challenge

➢System statistics is collected instantly as time series data.



Problem and Challenge

- ➢Temporal dependency is non-stationary.
- ➢Online inference for time varying temporal dependency is challenging.



Existing frameworks for temporal dependency discovery

➢Bayesian network modeling

- Take each time series as a random variable. Conditional probability is used to model the correlation among time series

➢Granger causality inference

- If X can (**Granger Causality**) infer Y, then the past of X should significantly help predict the future of Y, comparing with using the past of Y only.

Lasso Granger

Lasso Granger Method: learning regression model with L1 regulation.

Let $\mathbf{x}_t = vec([\mathbf{y}_{\cdot,t-1}, \mathbf{y}_{\cdot,t-2}, \ldots, \mathbf{y}_{\cdot,t-L}])$ egression for variable y_j is given as follows,

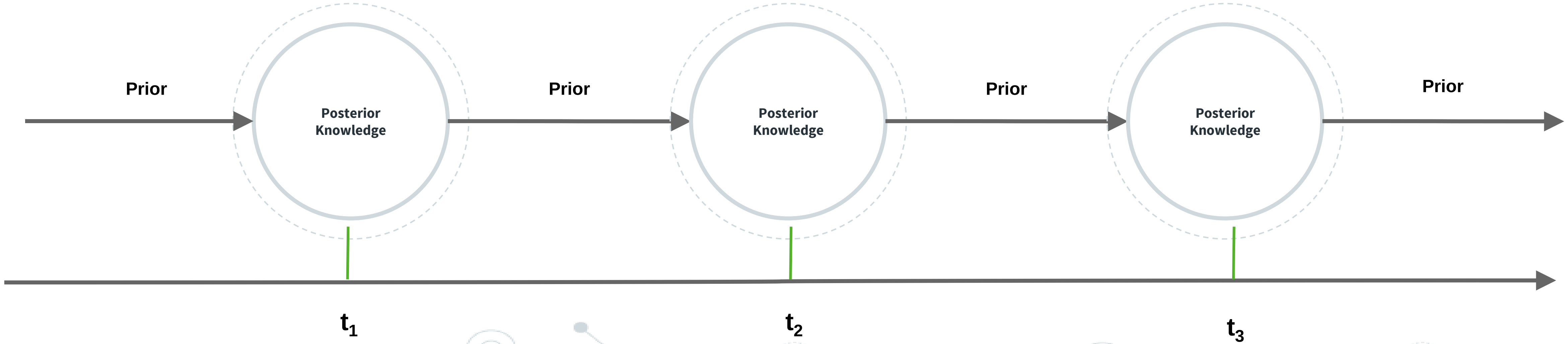
$$\min_{\mathbf{w}_j} \sum_{t=L+1}^{T} (\mathbf{y}_{j,t} - \mathbf{w}_j^{\intercal} \mathbf{x}_t)^2 + \lambda \parallel \mathbf{w}_j \parallel_1,$$

If the coefficient corresponding to $\mathbf{y}_{i,t-k}$ is non-zero, it shows a Granger Causality between $\mathbf{y}_i$ and $\mathbf{y}_j$.

Online inference of Lasso Granger

Regression with Lasso can be modeled with Bayesian Learning, refer to Bayesian Lasso.

Online inference for Lasso Granger can be implemented by Bayesian Lasso.



Bayesian Lasso for Granger Causality

Regression with Lasso can be modeled with Bayesian Learning, refer to Bayesian Lasso. Online inference for Lasso Granger can be implemented by Bayesian Lasso.

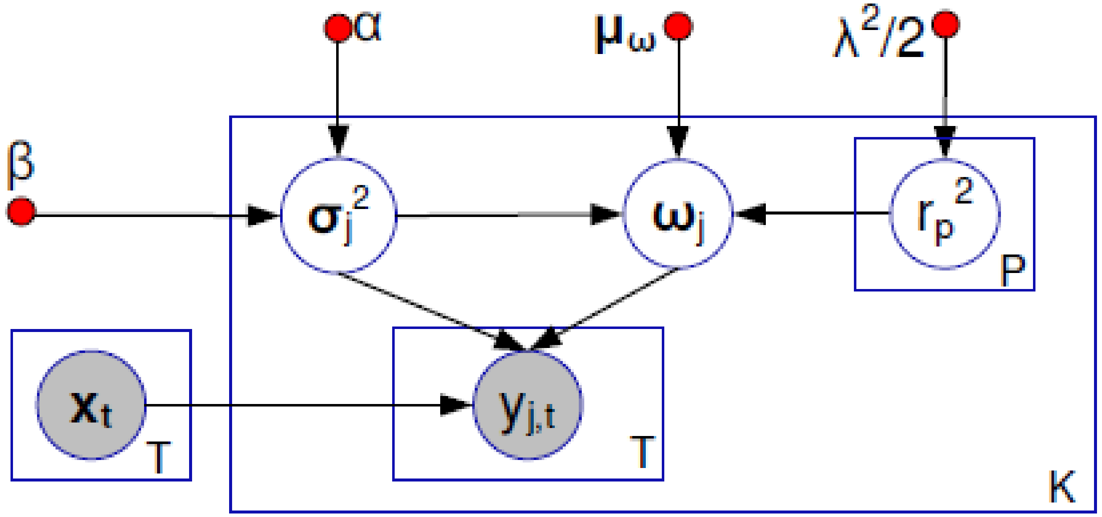
$$\mathbf{y}_{j,t} | \mathbf{w}_j, \sigma_j^2 \sim \mathcal{N}(\mathbf{w}_j^\intercal \mathbf{x}_t, \sigma_j^2).$$

$$\pi(\mathbf{w}_j | \sigma_j^2) = \prod_{p=1}^{P} \frac{\lambda}{2\sqrt{\sigma^2}} e^{-\lambda |\mathbf{w}_{j,p}| / \sqrt{\sigma^2}},$$

$$\mathbf{y}_{j,t} | \mathbf{w}_j, \sigma_j^2 \sim \mathcal{N}(\mathbf{w}_j^\intercal \mathbf{x}_t, \sigma_j^2).$$

$$\mathbf{w}_j | \sigma_j^2, \gamma_1^2, \ldots, \gamma_P^2 \sim \mathcal{N}(\mu_{\mathbf{w}}, \sigma_j^2 \mathbf{R}_{\mathbf{w_j}}),$$
$$\sigma_j^2 \sim \mathcal{IG}(\alpha, \beta),$$
$$\gamma_p^2 \sim Exp(\lambda^2/2), \quad 1 \le p \le P,$$

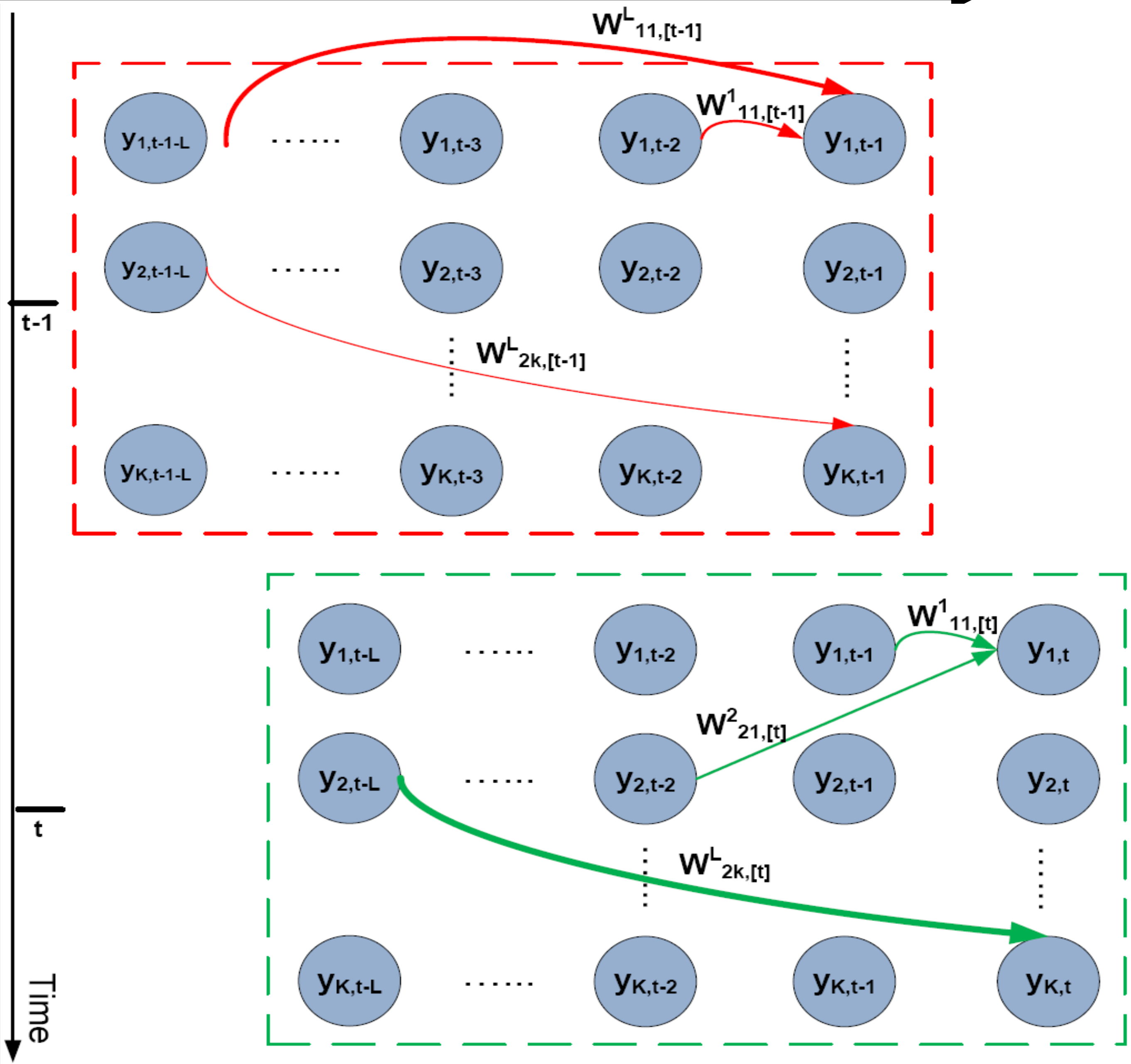
$$\mathbf{R}_{\mathbf{w_j}} = \mathrm{diag}(\gamma_1^2, \ldots, \gamma_P^2)$$



(a) Bayesian Lasso model.

Non-Stationary Granger Causality

- L is the maximum time lag for VAR model. Temporal dependency structure changes over time.
  - New dependency appears.
  - Old dependency disappears.
  - The strength of dependency changes
  - Sparsity
- The temporal structure is highly sparse.
  - Since patterns relatively short



Proposed Time-Varying Bayesian Lasso

$$\mathbf{w}_{j,t} = \mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t}, \qquad \mathbf{y}_{j,t} | \mathbf{c}_{\mathbf{w}_j}, \theta_j, \eta_{j,t}, \sigma_j^2 \sim \mathcal{N}((\mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t})^\intercal \mathbf{x}_t, \sigma_j^2).$$

Constant Component:

$$\mathbf{c}_{\mathbf{w}_j} \sim \mathcal{N}(\mu_c, \sigma_j^2 \mathbf{R}_{\mathbf{c_j}})$$

Time Varying Component:

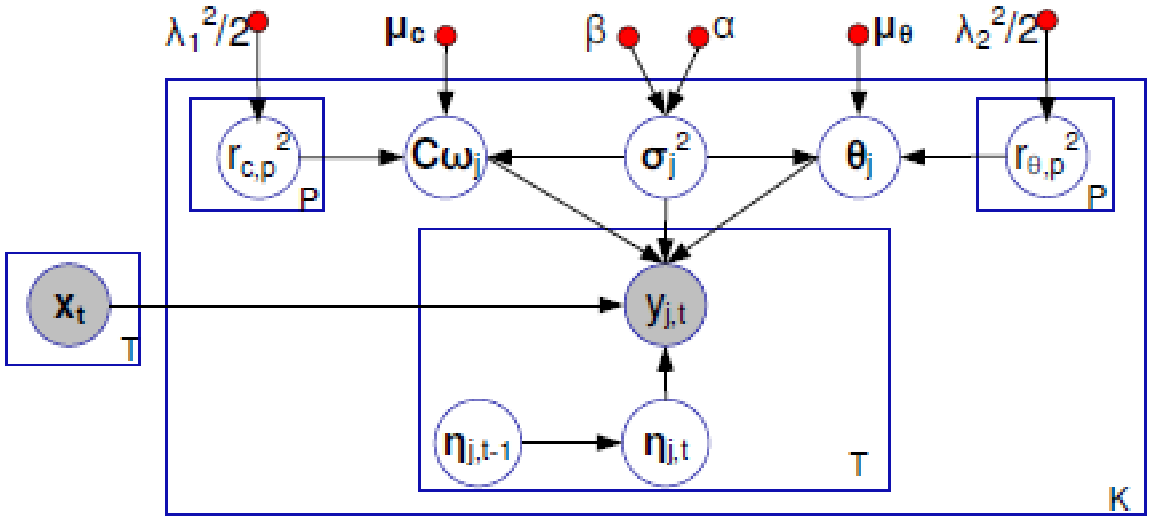
Varying scales

$$\theta_j \sim \mathcal{N}(\mu_\theta, \sigma_j^2 \mathbf{R}_{\theta_\mathbf{j}})$$

Varying direction: random walk

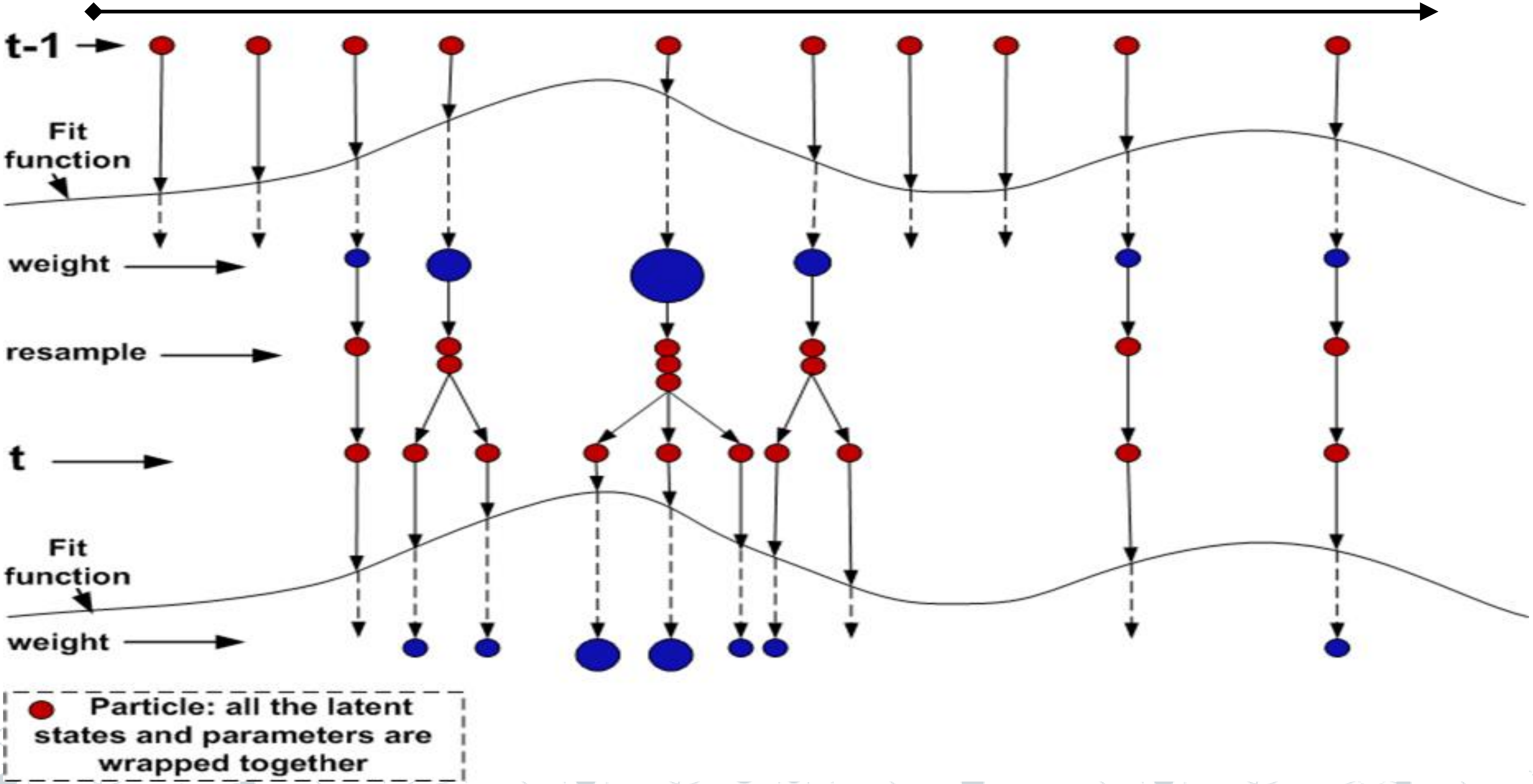
$$\eta_{j,t} \sim \mathcal{N}(\eta_{j,t-1}, \mathcal{I}_P)$$

$$\min_{\{\mathbf{w}_{j,t}\}} \sum_{t=L+1}^{T} (\mathbf{y}_{j,t} - (\mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t})^\intercal \mathbf{x}_t)^2 + \lambda_1 \parallel \mathbf{c}_{\mathbf{w}_j} \parallel_1 + \lambda_2 \parallel \theta_j \parallel_1,$$



(b) Time-varying Bayesian Lasso model.

Solution(particle learning)



Evaluation

Baseline Algorithms:
●*BLR*(**q**)**: Bayesian Linear Regression.**
●*TVLR*(**q**)**: Time Varying Bayesian Linear Regression.**
●*BLasso*(**λ**)**: Bayesian Lasso Regression.**
Our proposed algorithm:
TVLasso(**λ**)**: Time Varying Bayesian Lasso Regression.**

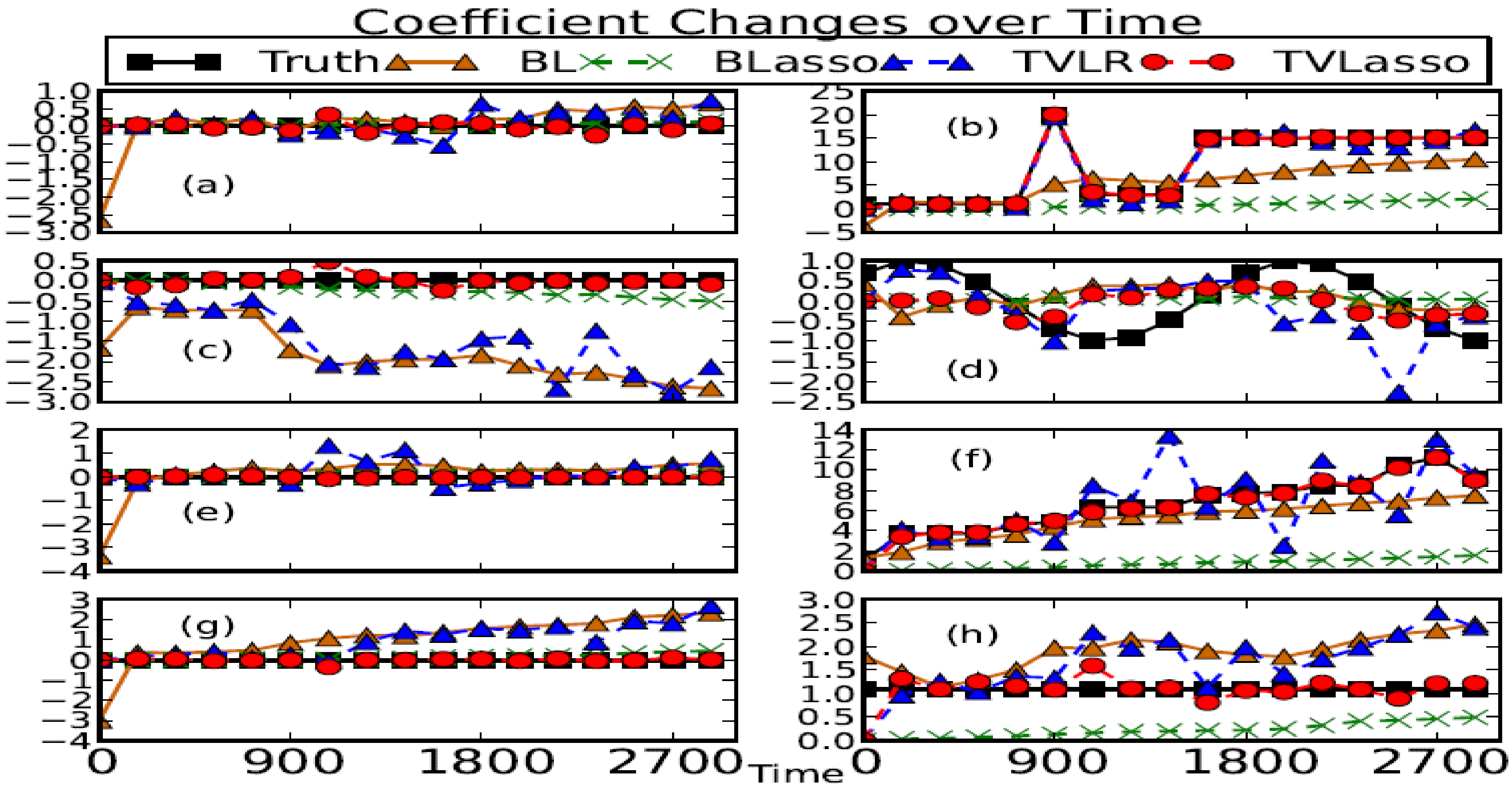
Evaluation Metrics:
➔**AUC Score:** The Area under the ROC.

➔**Prediction Error:**

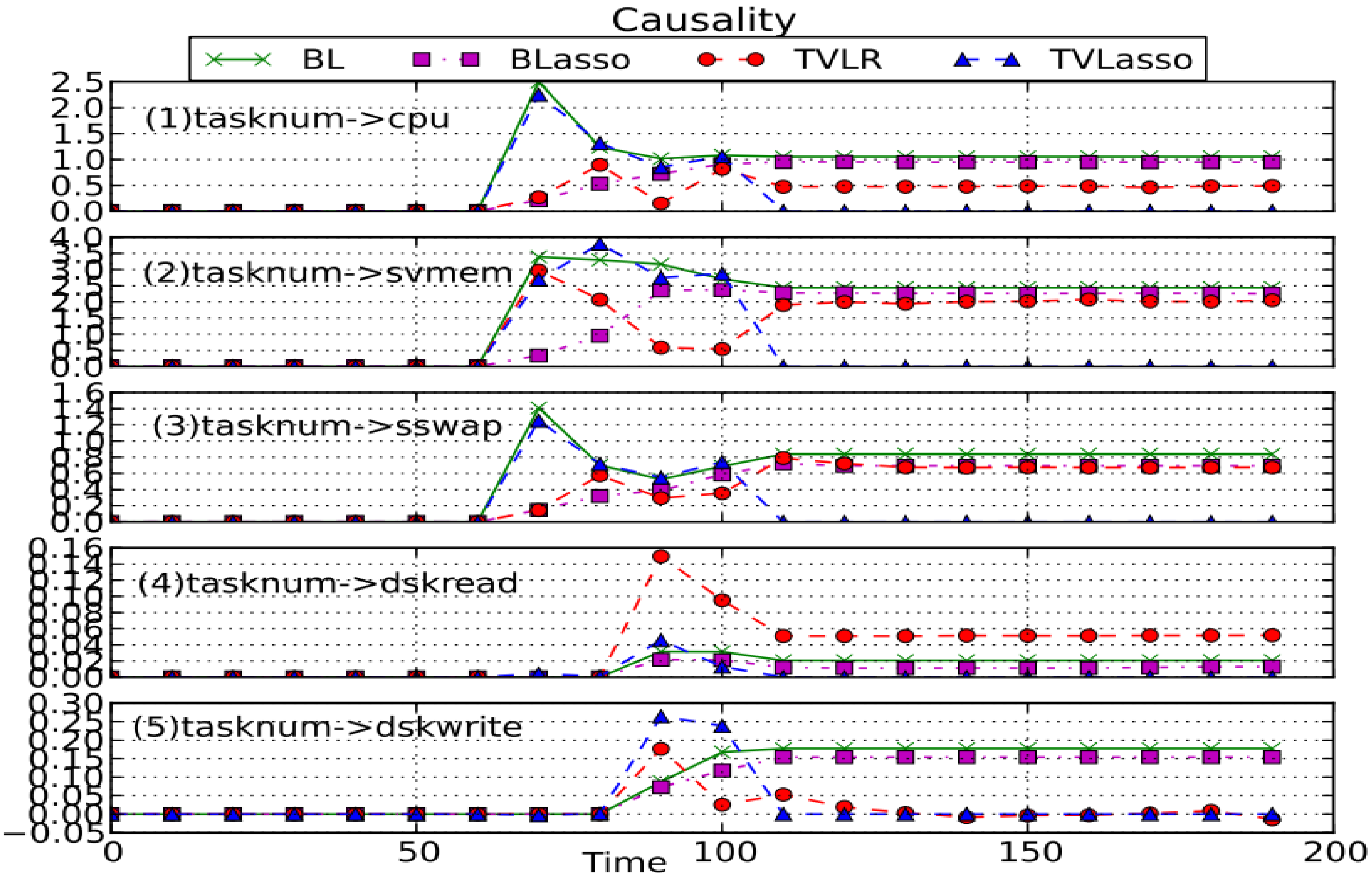
$$\Delta = ||\mathbf{W}_t - \widehat{\mathbf{W}}_t||_F$$

Evaluation over Synthetic Data

20 time series with different varying patterns. 8 coefficients are selected for illustration.



Evaluation over Real Data: **TVLasso can effectively identify the time varying dependency**



Contents

1 Introduction and Overview

2 Event Generation and System Monitoring

3 Pattern Discovery and Summarization

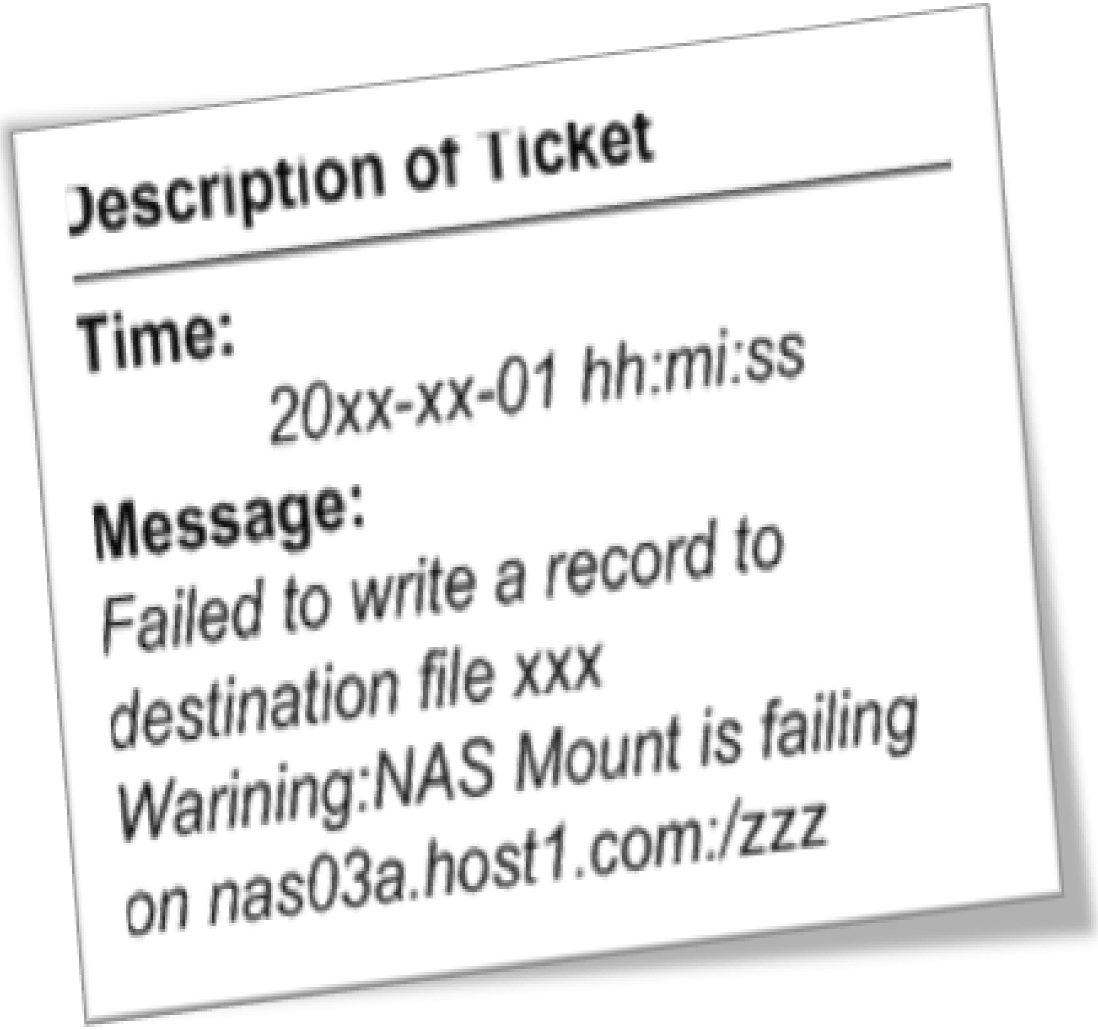
4 **Mining with Events and Tickets**

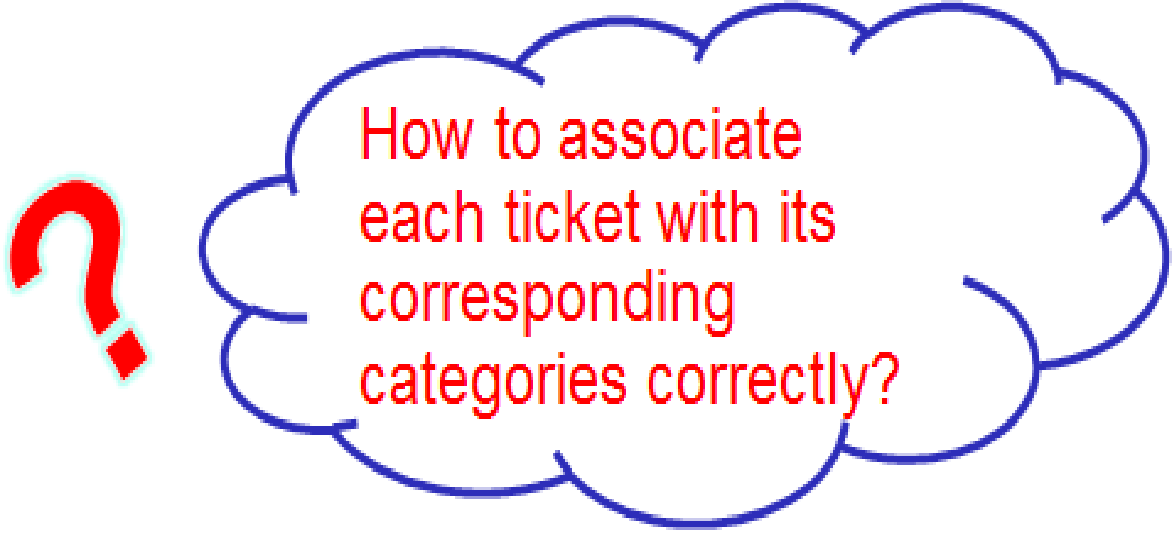
5 Conclusions

Outline

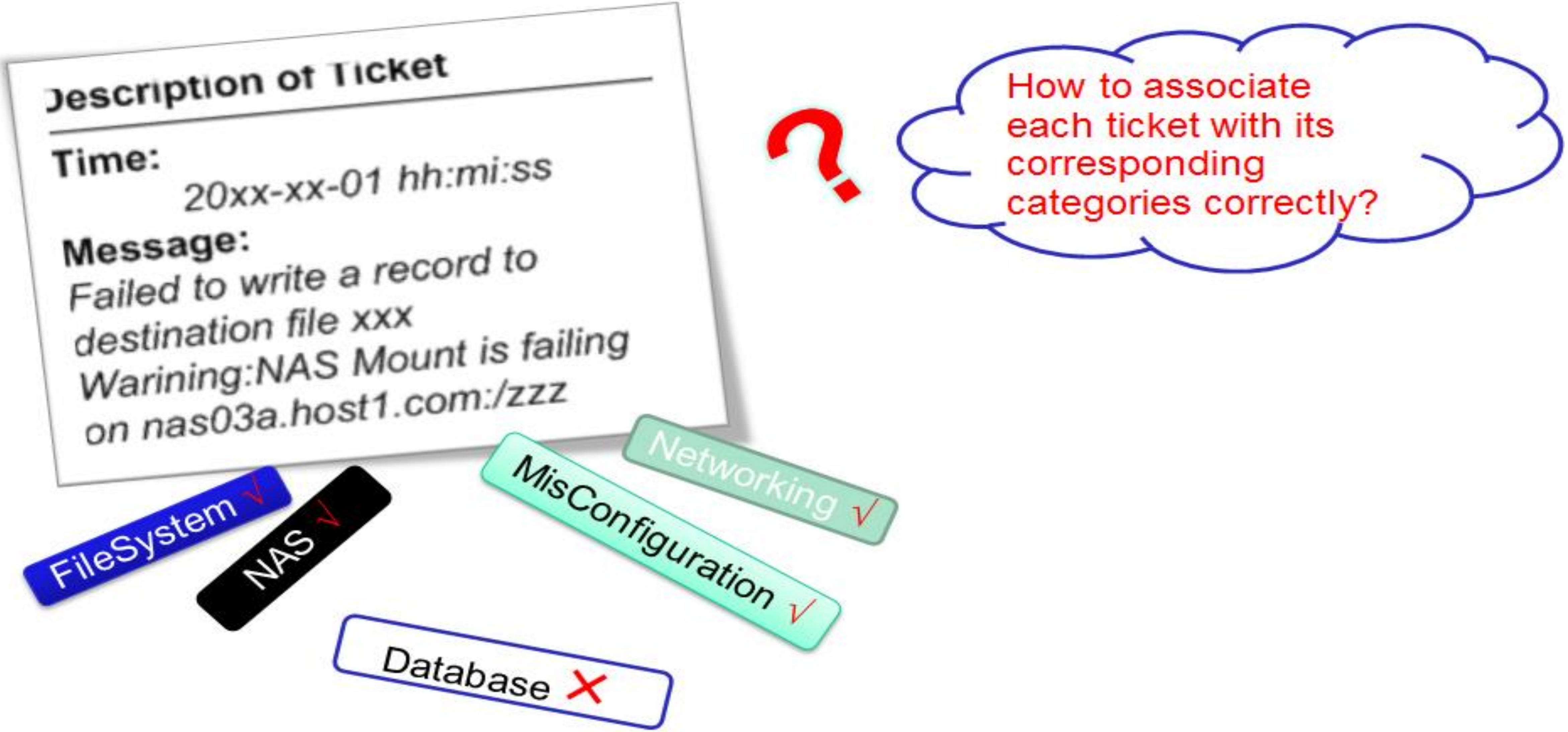
- Ticket Classification
- Ticket Resolution Recommendation
- Ticket Analysis (Knowledge Extraction)

IT Problem Category Determination by Tickets

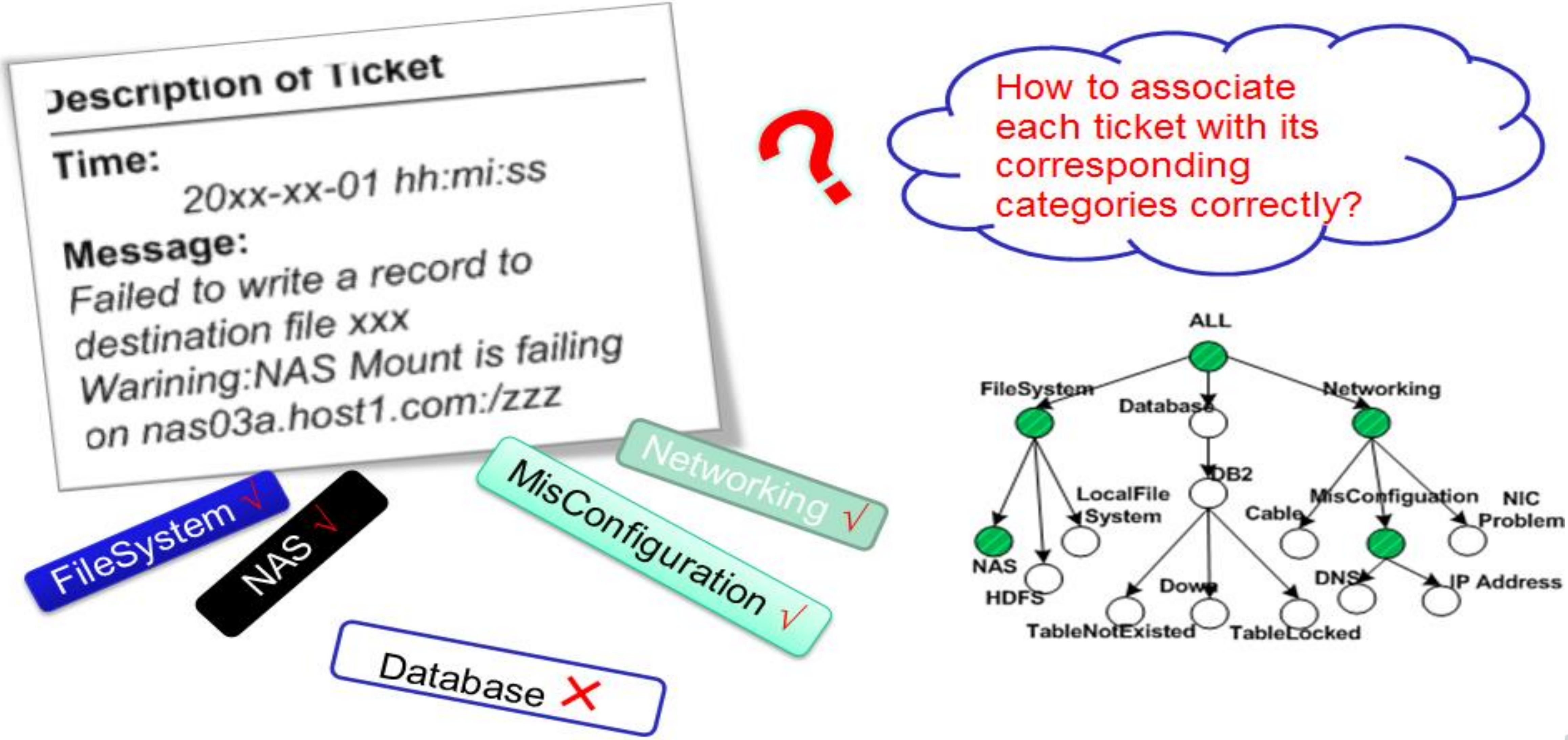




IT Problem Category Determination by Tickets



IT Problem Category Determination by Tickets



Related Work

➢Text classification (Without considering multi-label and label hierarchy)

- SVM,CART, KNN, Rule-based classification, logistic regression

➢Multi-label classification algorithm(Without considering label hierarchy)

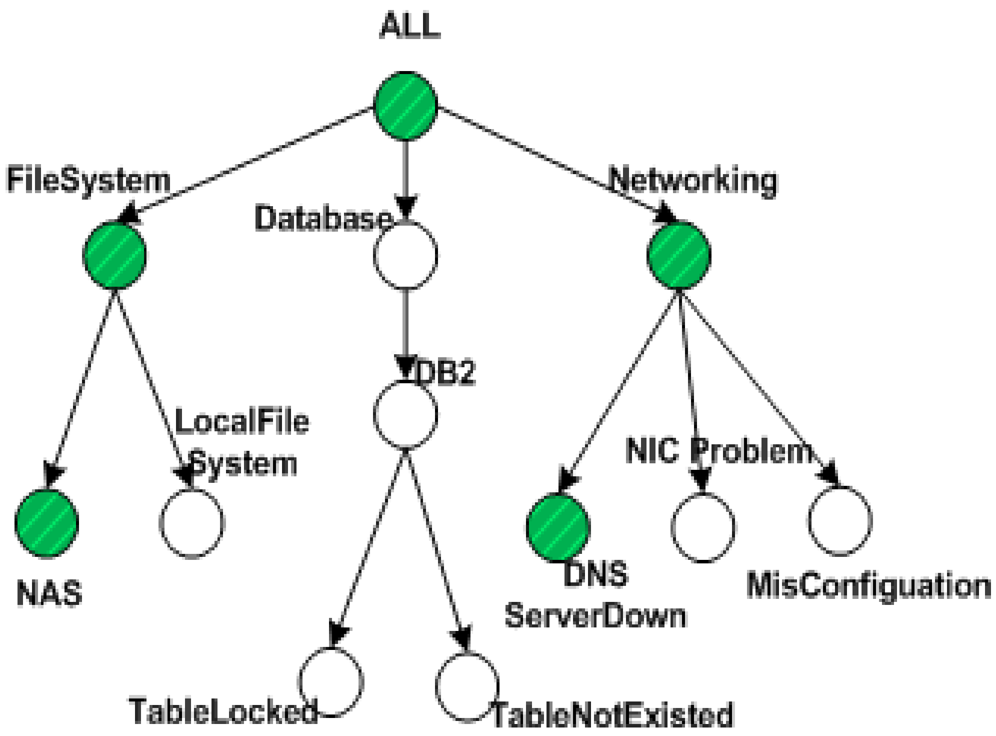
- Problem transformation based approach
- Algorithm adaption based approach

➢Hierarchical multi-label classification algorithm

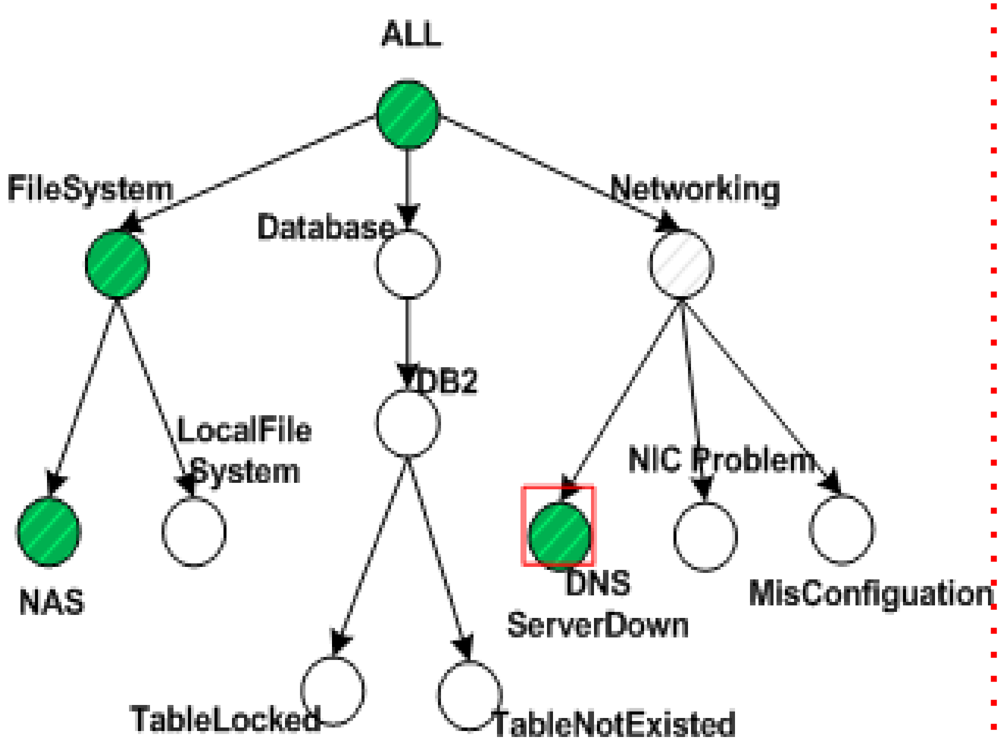
- Recursively split the training data(Overfitting)
- Hierarchical consistency is guaranteed by post-processing(Our method belongs to this category)

Hierarchical Consistency with Multiple Labels

➢ **Hierarchical Constraint:** Given a ticket, any node is **positive** (in green color) if it is the root node or **its parent is positive**.



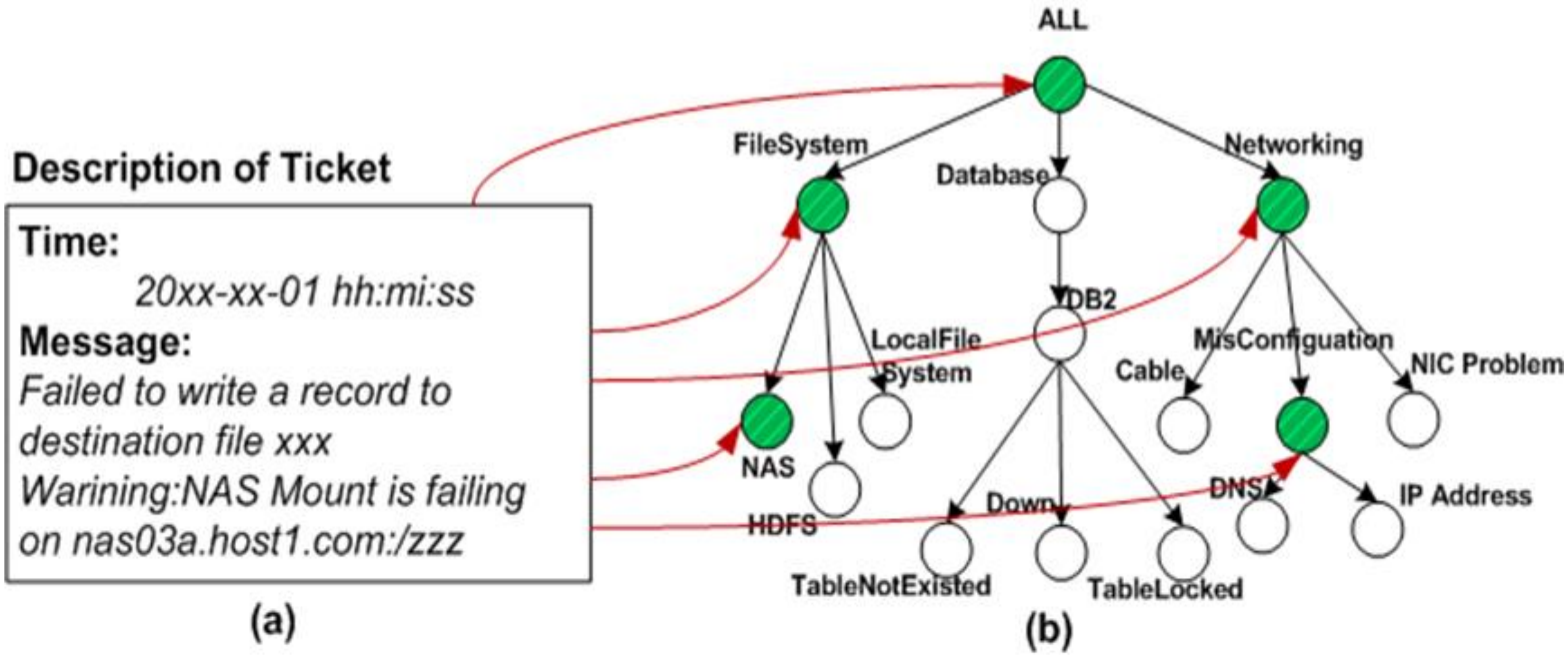
Satisfy hierarchical consistency



Violate hierarchical consistency

Hierarchical Consistency with Multiple Labels

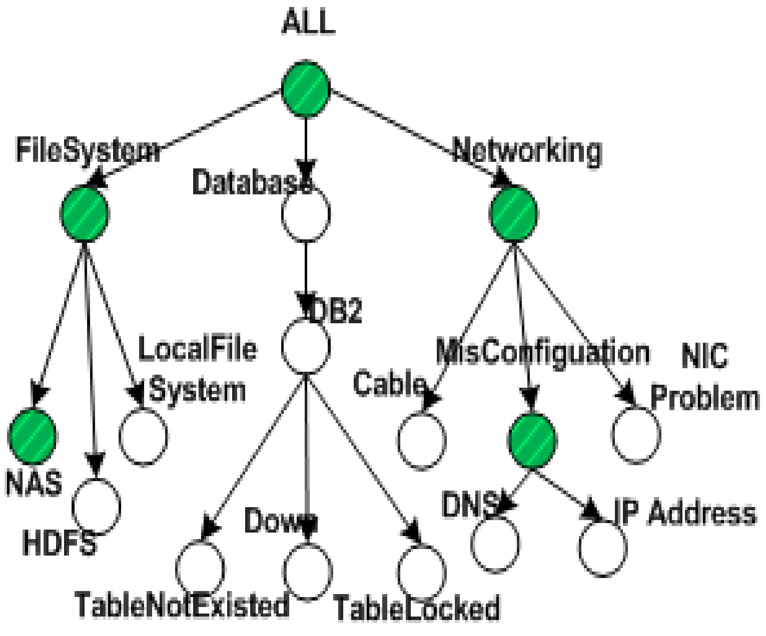
- The characteristics of hierarchical multi-label classification over the ticket data are listed as follows:
  - With multiple paths.
  - With a partial path.



Guarantee Hierarchical Consistency with Loss

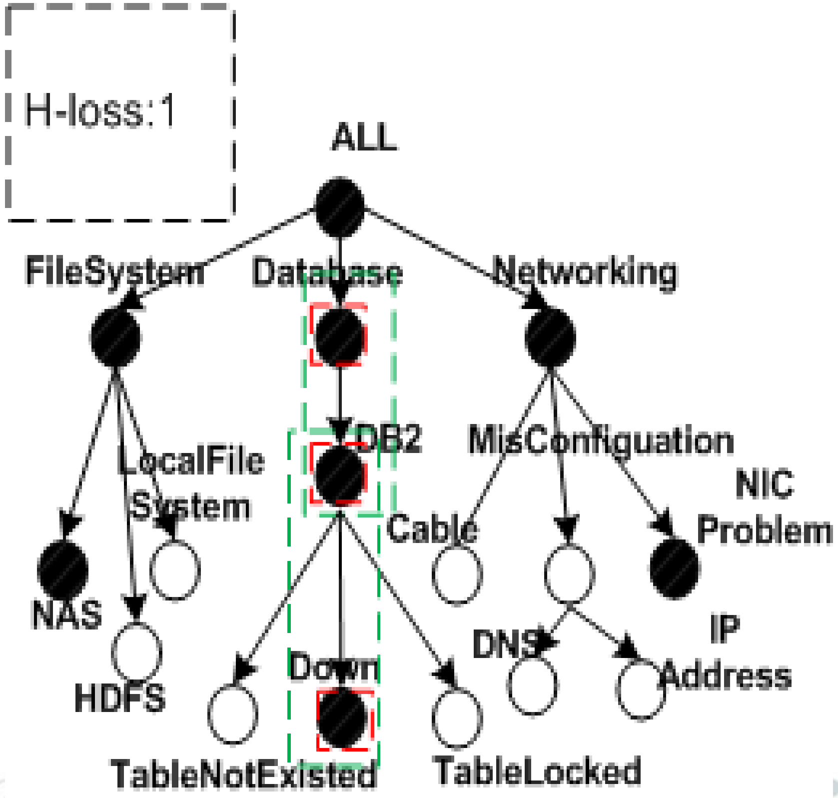
➢ **H-Loss.**

- Main idea: any mistake occurring in a subtree does not matter if the subtree is rooted with a mistake as well
- **1** loss because of the error at **Database node**, while **0** for both **DB2** and **Down** nodes



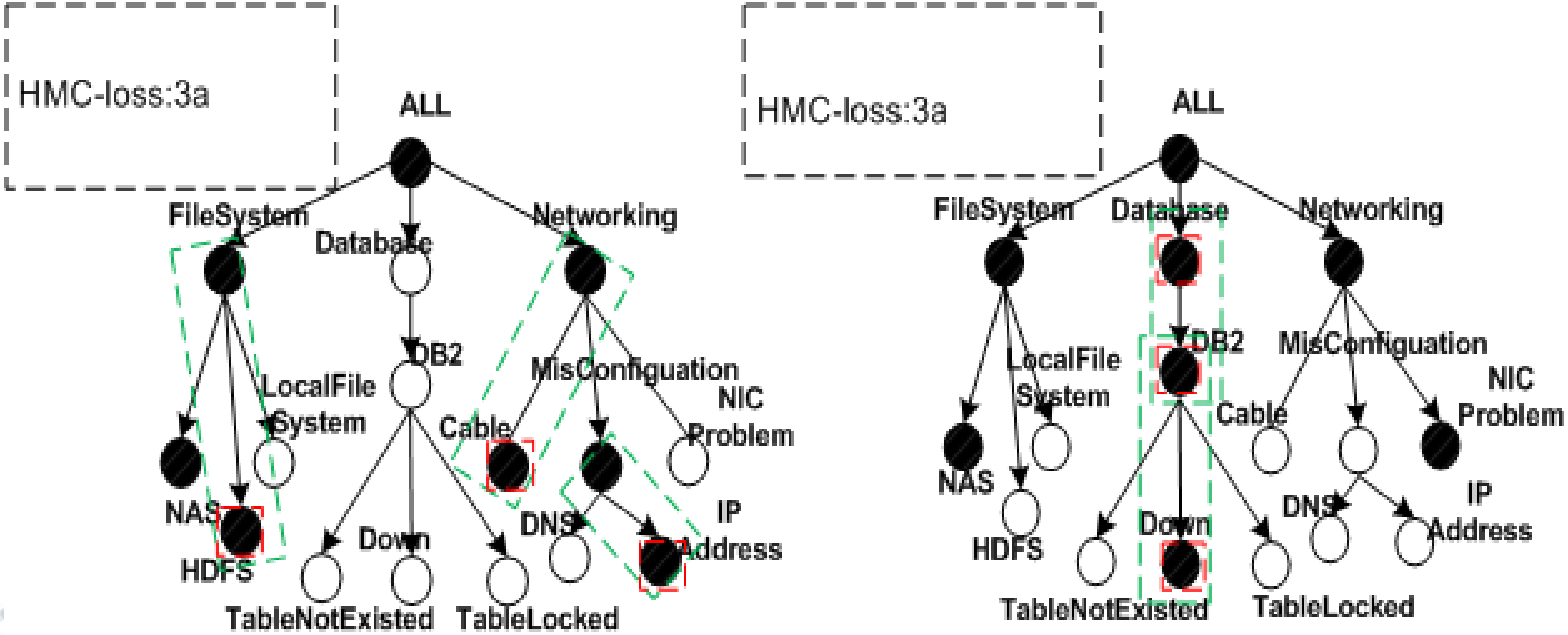
(b)ground truth

Assign to DB2 team and its sub-team Down will incur more cost due to wrong assignment.



Guarantee Hierarchical Consistency with Loss

- **HMC-Loss.**
  - Main idea: misclassification error at a node is weighted by its hierarchy information. It also weights FP and FN differently.

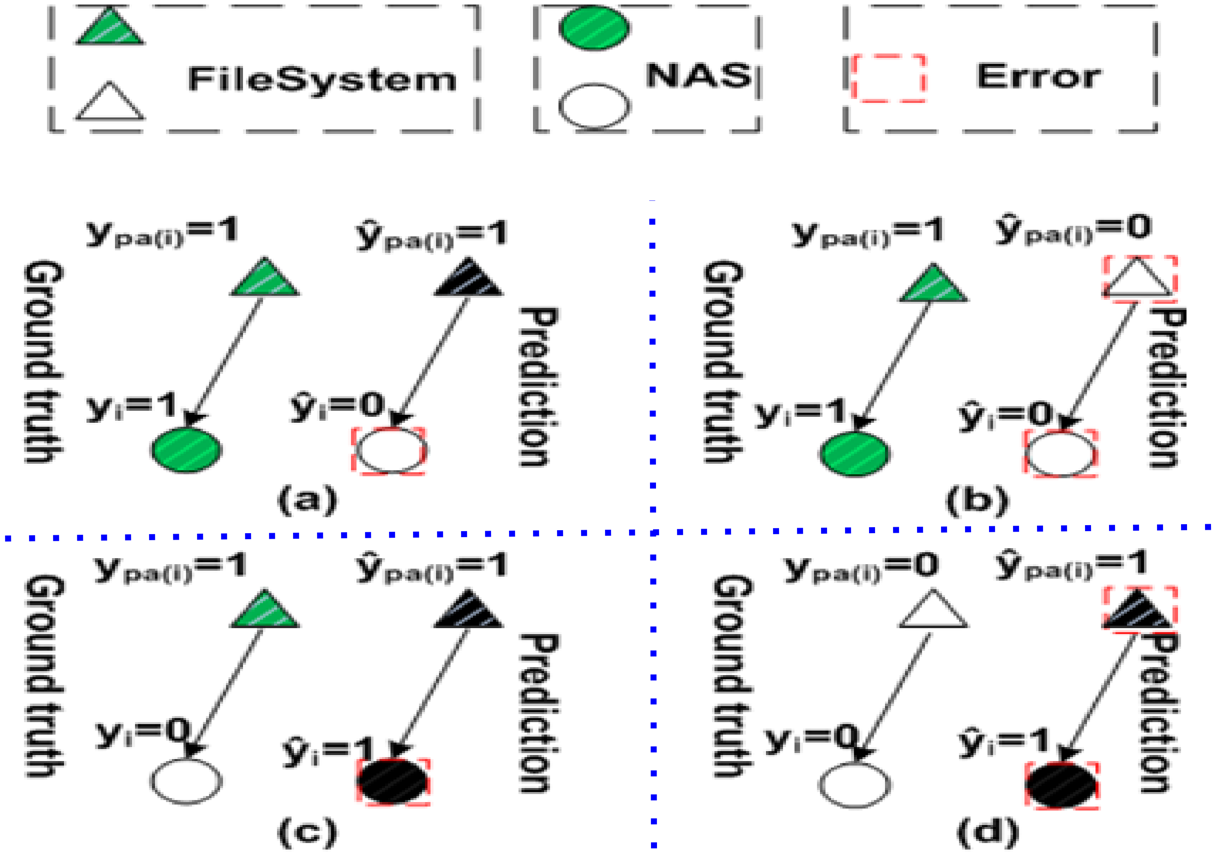


Propose CH-Loss

- **Contextual Misclassification Information.**
  - **Different weights for each case**

False Negative

False Positive



Communication with its direct parent

Communication with its grandparent

$$\ell(\hat{\mathbf{y}},\mathbf{y}) = w_1 \sum_{i>0}^{N-1} y_i y_{pa(i)} \tilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + w_2 \sum_{i>0}^{N-1} y_i y_{pa(i)} \tilde{\hat{y}}_i \tilde{\hat{y}}_{pa(i)} C_i + w_3 \sum_{i>0}^{N-1} \tilde{y}_i y_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i + w_4 \sum_{i>0}^{N-1} \tilde{y}_i \tilde{y}_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i.$$

Propose CH-Loss

➢ **CH-Loss generalize other Loss functions.**

Goal	*CH-Loss parameter settings*
Minimize Hamming Loss	$w_1 = w_2 = w_3 = w_4 = 1$, $C_i = 1$
Minimize HMC-Loss	$w_1 = w_2 = \alpha$, $w_3 = w_4 = \beta$, C_i = is defined by user
Minimize H-Loss	$w_1 = w_3 = 1$, $w_2 = w_4 = 0$, $C_i = 1$
Increase recall	w_1 and w_2 are larger than w_3 and w_4
Increase precision	w_1 and w_2 are smaller than w_3 and w_4
Minimize misclassification errors occur in both parent and children nodes	$w_1 < w_2$ and $w_3 < w_4$

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = w_1 \sum_{i>0}^{N-1} y_i y_{pa(i)} \tilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + w_2 \sum_{i>0}^{N-1} y_i y_{pa(i)} \tilde{\hat{y}}_i \tilde{\hat{y}}_{pa(i)} C_i + w_3 \sum_{i>0}^{N-1} \tilde{y}_i y_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i + w_4 \sum_{i>0}^{N-1} \tilde{y}_i \tilde{y}_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i.$$

Minimize Expected CH-Loss

P(y|x) is the probability of the label vector y given ticket x.

$$LE(\hat{\mathbf{y}}, \mathbf{x}) = \sum_{\mathbf{y} \in \{0,1\}^N} \ell(\hat{\mathbf{y}}, \mathbf{y}) P(\mathbf{y}|\mathbf{x})$$

CH-Loss

$$\hat{\mathbf{y}}^* = \arg\min_{\hat{\mathbf{y}} \in \{0,1\}^N} LE(\hat{\mathbf{y}}, \mathbf{x})$$

s.t. ŷ satisfying the hierarchical constraint

Issue: it's difficult to estimate the probability P(y|x), since y can be one of **$O(2^N)$** vectors.

Equivalent Derivation (Proposition IV.3 of our work)

$$\hat{\mathbf{y}}^* = \arg\max_{\hat{y}\in\{0,1\}^N} \sum_i y_i \sigma(i)$$

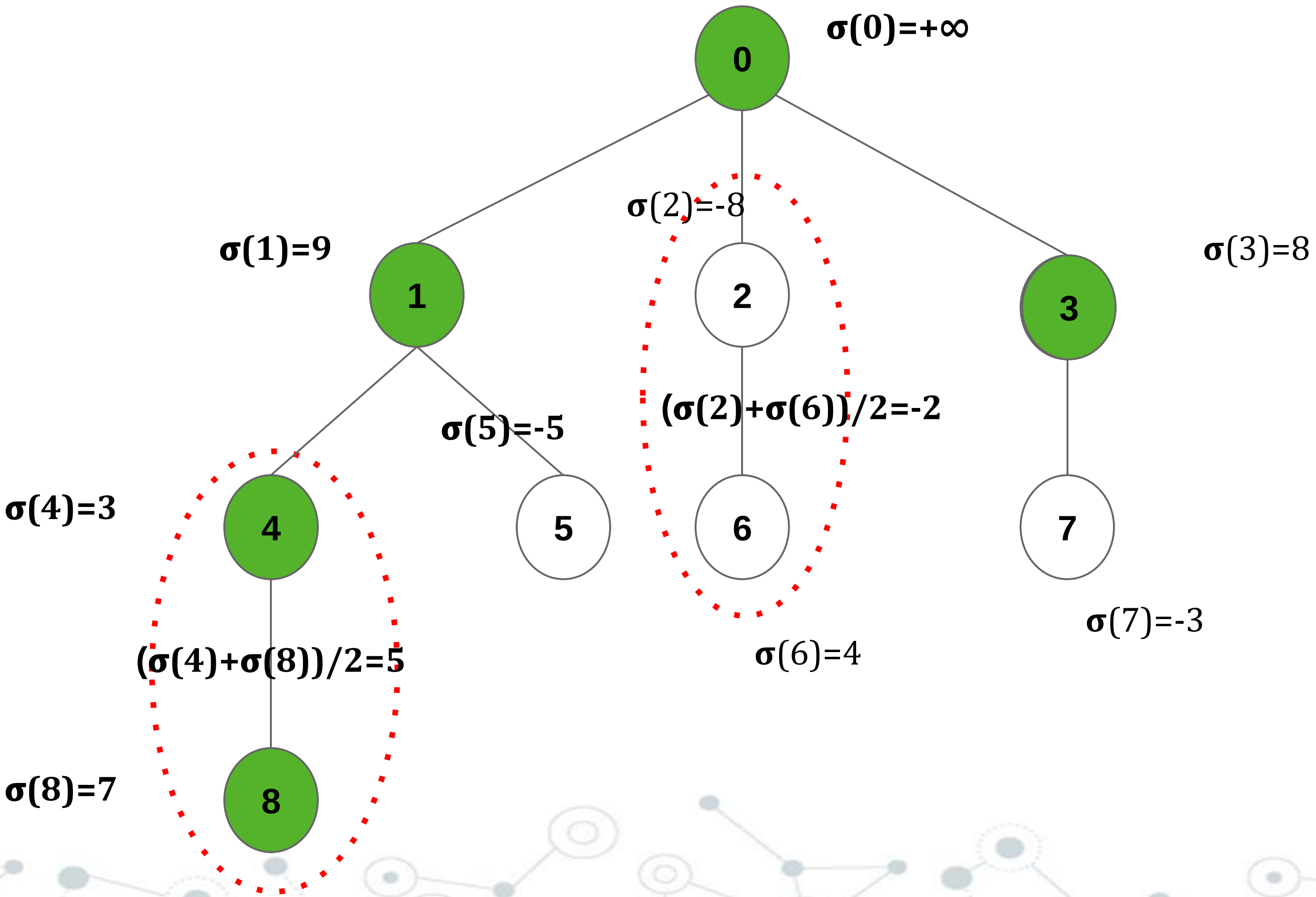
s.t. ŷ satisfying the hierarchical constraint

Where $\sigma(i)$ can be computed by $P(y_i|x)$ and $P(y_{parent(i)}|x)$. The above equation can be solved by proposing GLabel, a greedy algorithm.

$P(y_i|x)$ can be estimated with a binary classifier on node i.

GLabel Algorithm

Greedily choose the node i with largest σ(i) to label, considering the hierarchical constraint, Complexity : O(N lgN).



Node	Value
σ(0)	+∞
σ(1)	9
σ(2)	-8
σ(3)	8
σ(4)	3
σ(5)	-5
σ(6)	4
σ(7)	-3
σ(8)	7

Experiment

1. 23,000 tickets are collected from the real IT environment.

1. 20,000 tickets are randomly selected for training data

1. The remaining 3,000 tickets are used for testing.

GLabel vs. Flat Classifier

Metric	**SVM**	***GLabel***
CH-Loss	4.2601	***2.6889***
Parent-Child Error	0.3788	***0.1729***
Hierarchy Error	0.0102	***0.0***

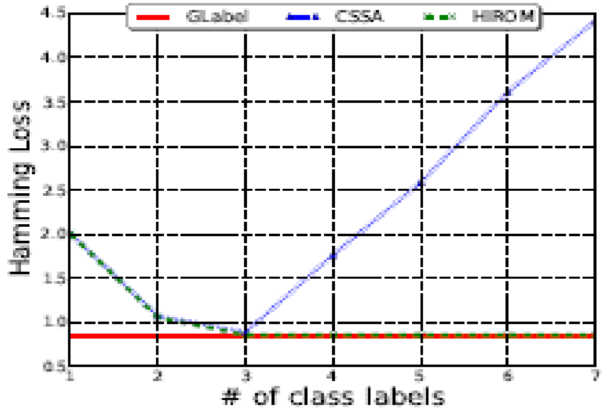
The state-of-the-art algorithm

1. **CSSA , which requires the number of labels for each ticket**

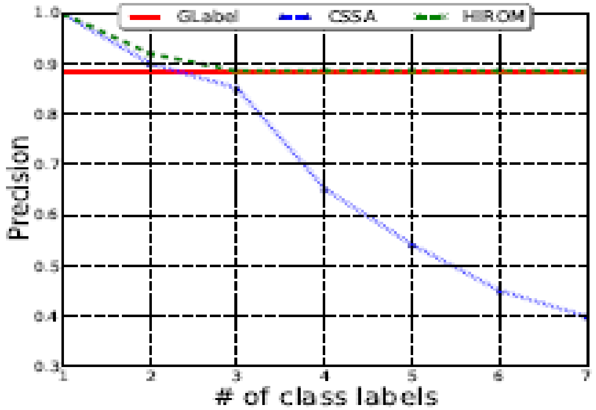
1. **HIROM, which requires the maximum number of labels for all the tickets**

The GLabel algorithm is capable of minimizing the loss automatically.

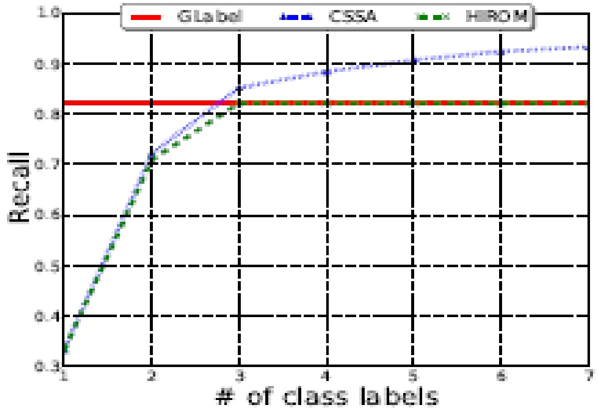
Optimizing varying loss: **GLabel can efficiently minimize the loss without any knowledge about the number of labels for tickets.**



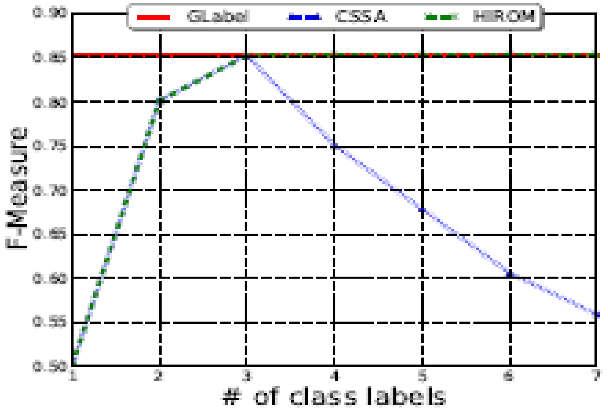
(a) The lowest Hamming loss: CSSA gets 0.8876 at # 3; HIROM gets 0.8534 at # 4; GLabel gets 0.8534.



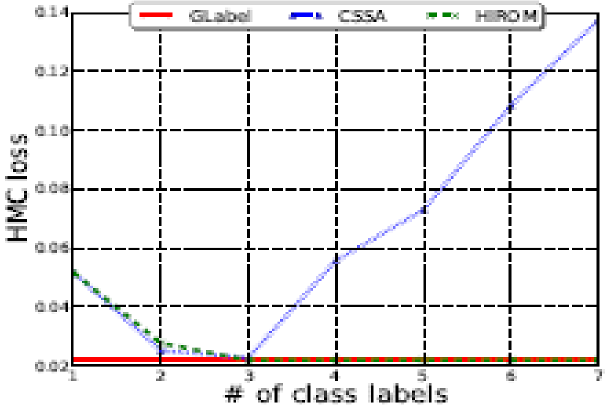
(b) Varying precision during minimizing the Hamming loss.



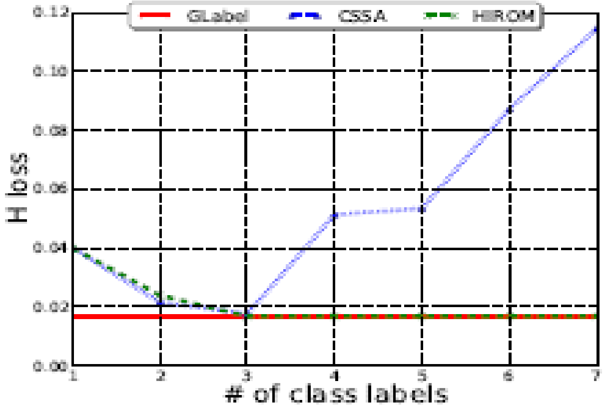
(c) Varying recall during minimizing the Hamming loss.



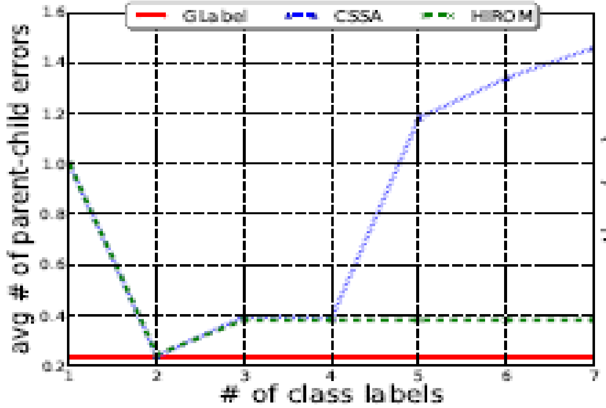
(d) Varying F-Measure score during minimizing the Hamming loss.



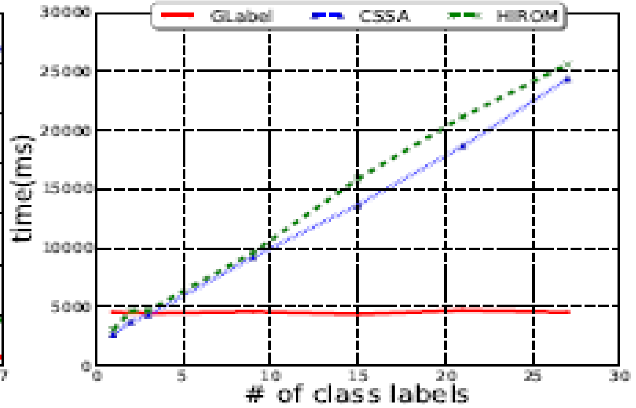
(e) The lowest HMC-Loss: CSSA gets 0.0227 at # 3; HIROM gets 0.0219 at # 4; GLabel gets 0.0219.



(f) The lowest H-Loss: CSSA gets 0.0176 at # 3; HIROM gets 0.0168 at # 3; GLabel gets 0.0167.

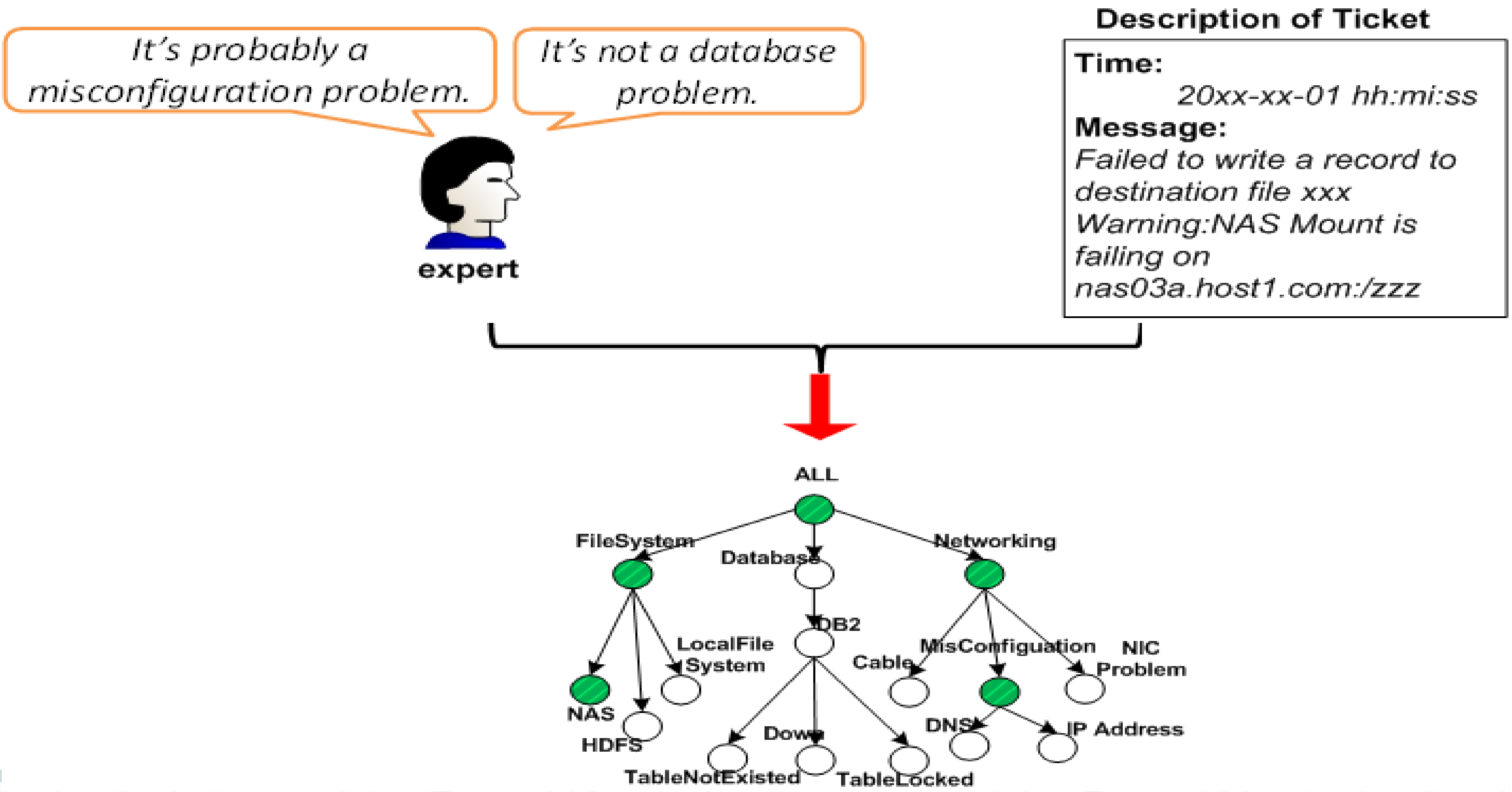


(g) The lowest AVG. parent-child error: CSSA gets 0.237 at # 2; HIROM gets 0.2440 at #2; Glabel gets 0.2304.

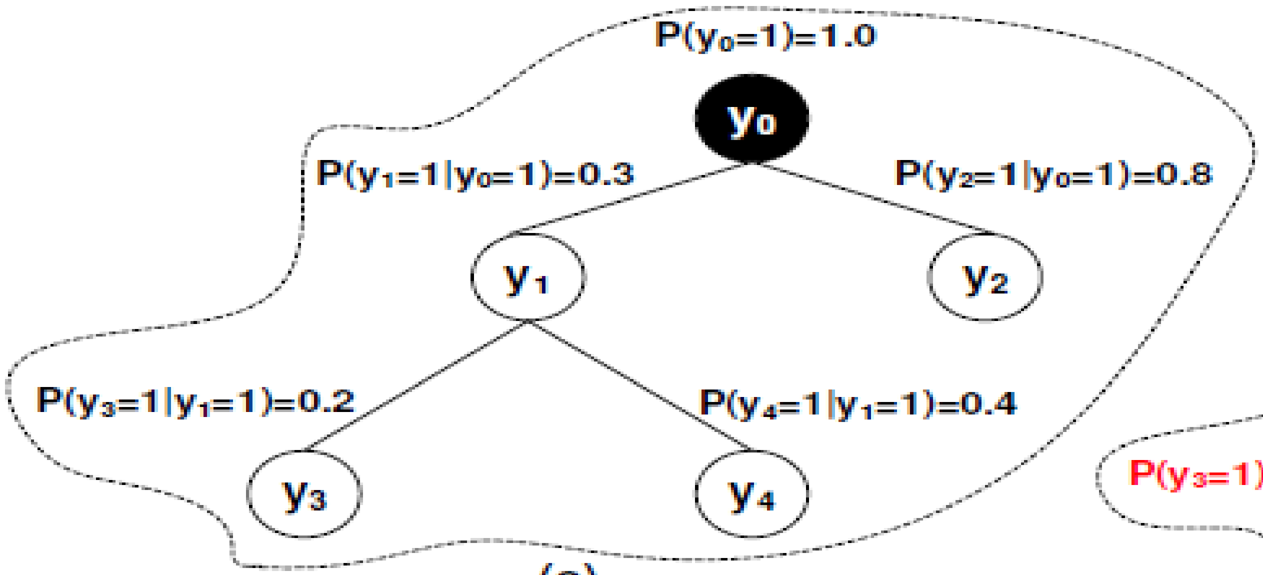


(h) Time complexity with respect to the number of classes related to each predicting ticket.

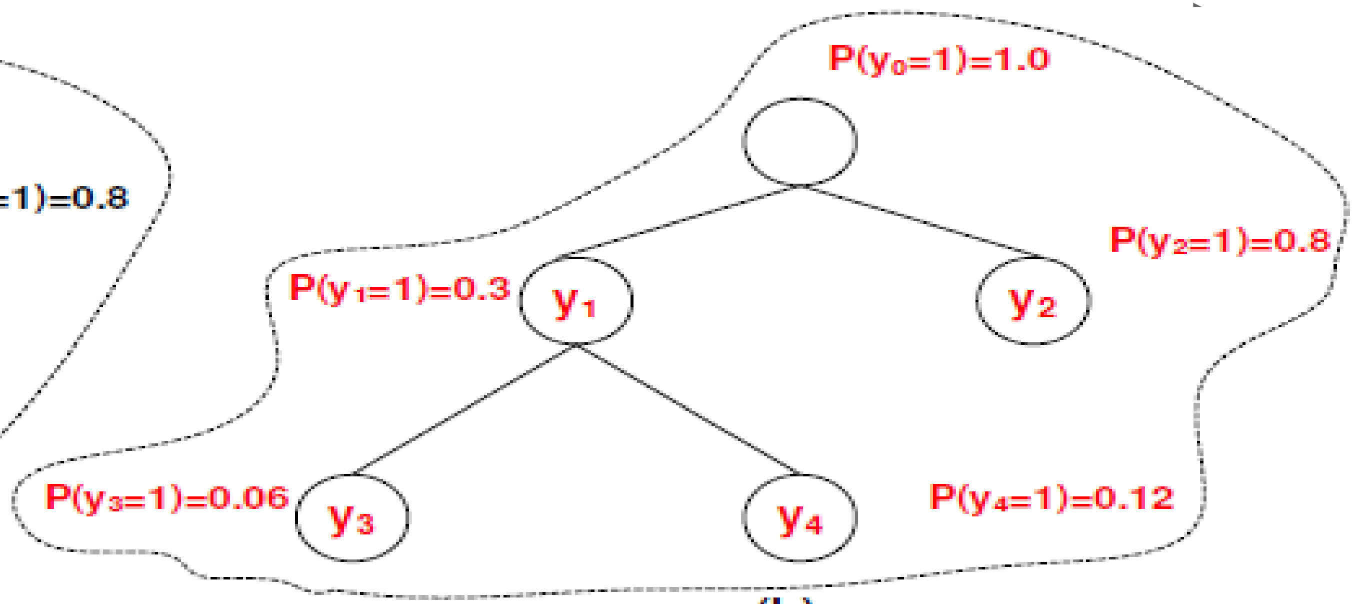
Domain Knowledge Integration



Domain Knowledge Integration (Kilo based on sum-product)



(a)



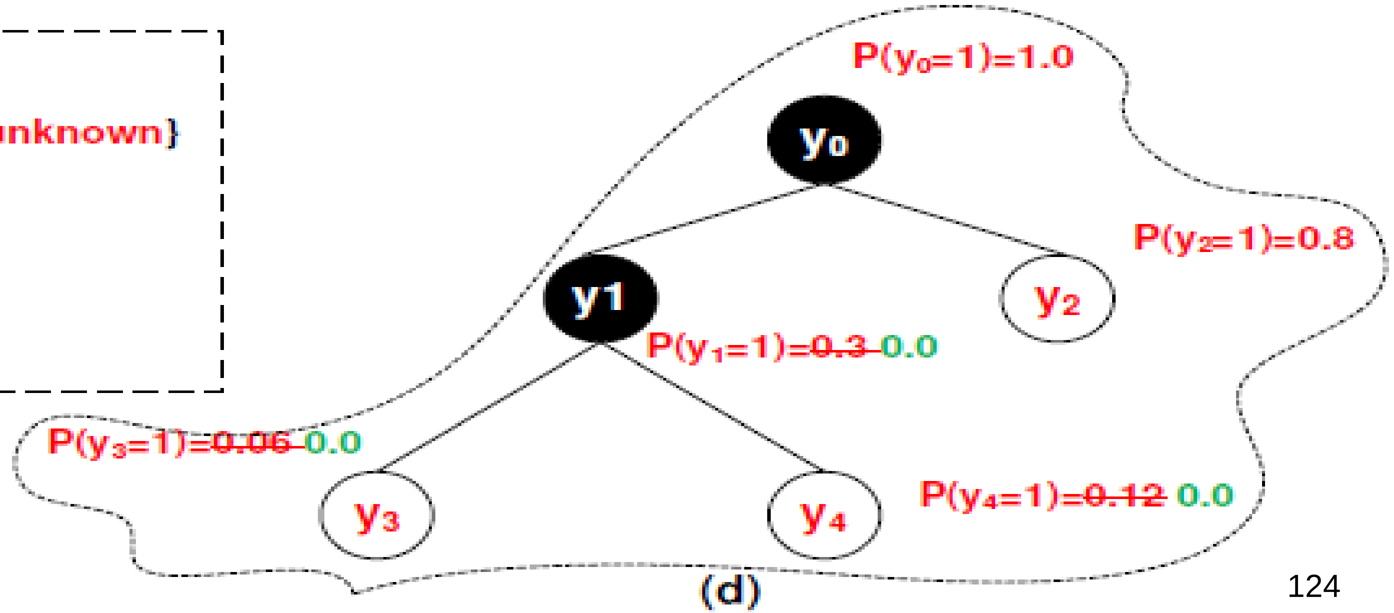
(b)

Prior Knowledge:
{y_0:positive, y_1:negative,y_2:unknown,y_3:unknown,y_4:unknown}

↕

$P(y_0=1)$ = 100% and $P(y_1=1)$ = 0%

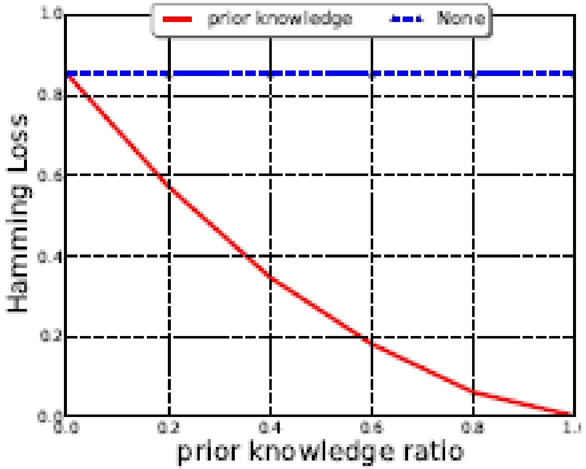
(c)



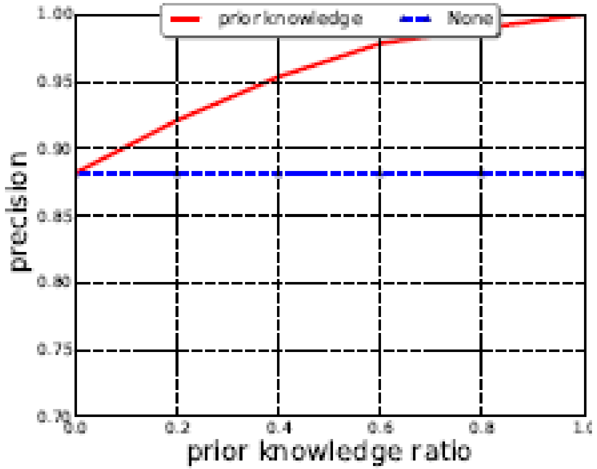
(d)

Prior knowledge on given node is propagated to its linked nodes.

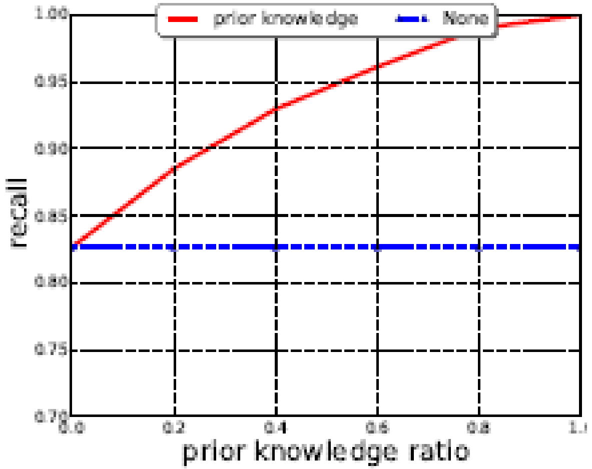
Domain Knowledge Integration Experiment over ticket data: **The more prior knowledge leads to more accurate result and smaller loss.**



(a) Varying Hamming loss with changing prior knowledge ratio.

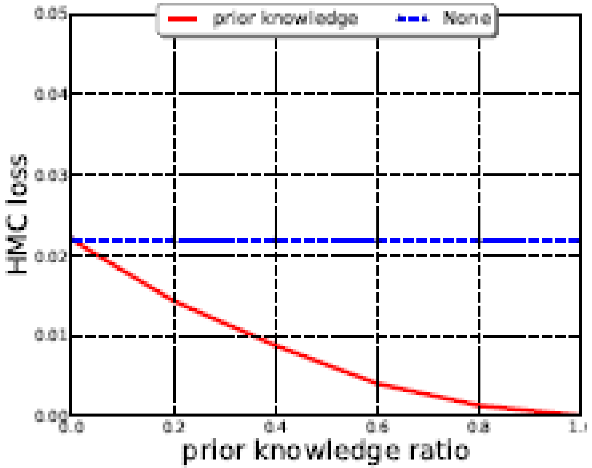


(b) Varying precision with changing prior knowledge ratio.

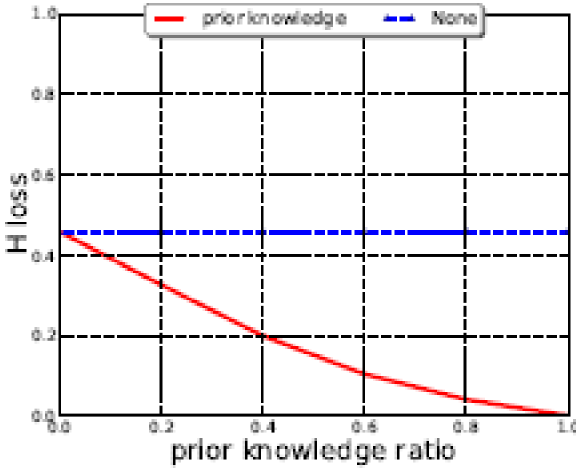


(c) Varying recall during with changing prior knowledge ratio.

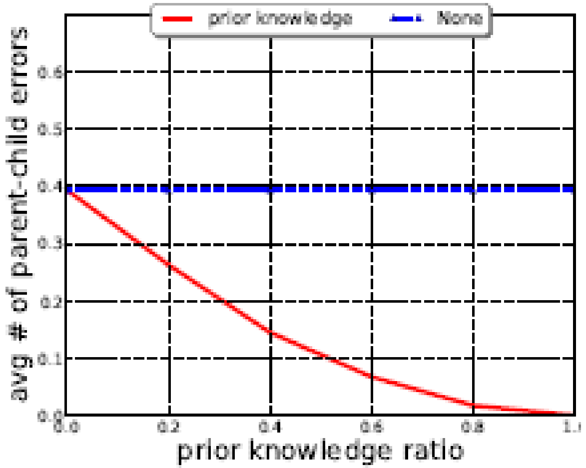
(d) Varying F-Measure score with changing prior knowledge ratio.



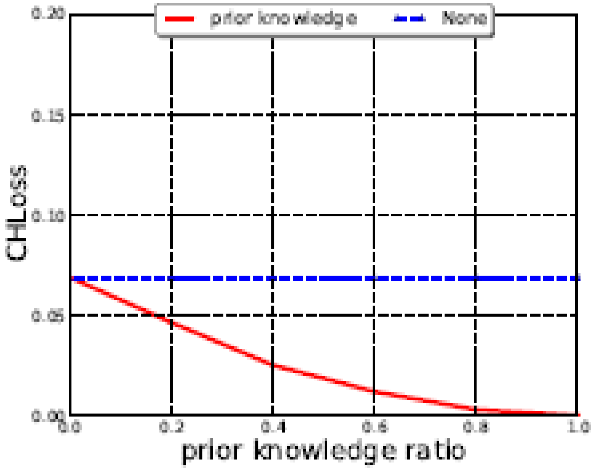
(e) Varying HMC-Loss with changing prior knowledge ratio.



(f) Varying H-Loss with changing prior knowledge ratio.



(g) Varying avg. Parent-Child Error with changing prior knowledge ratio.



(h) Varying CHLoss with changing prior knowledge ratio.

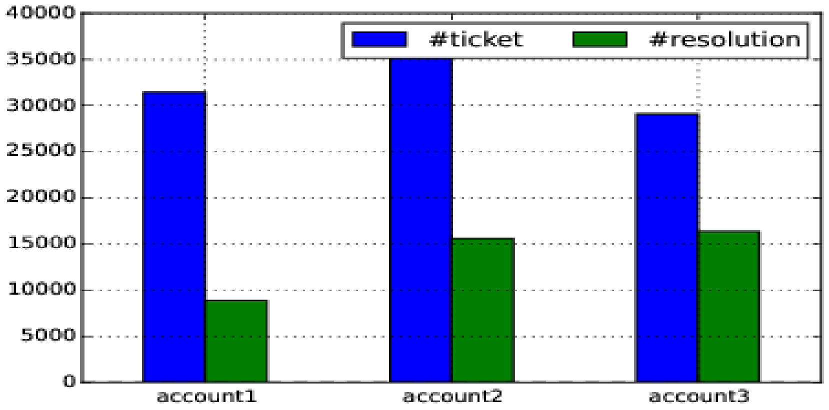
Outline

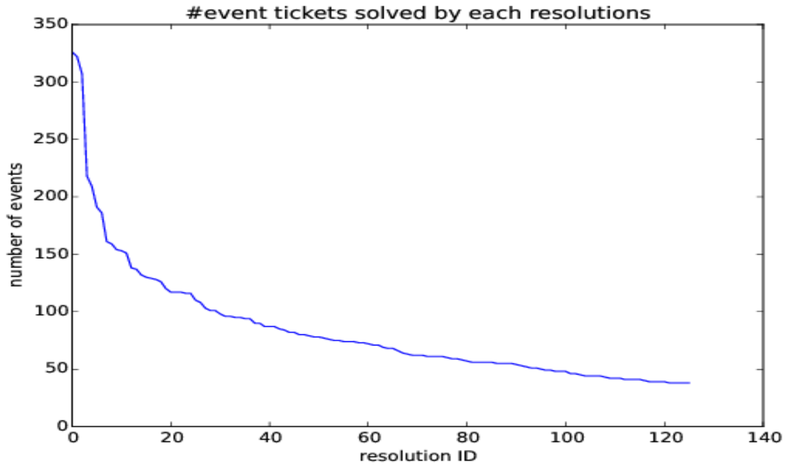
- Ticket Classification
- Ticket Resolution Recommendation
- Ticket Analysis (Knowledge Extraction)

Ticket Resolution Recommendation (Tang et al., CNSM 2014; Zhou et al., IM 2015)

◎Repeated Resolutions of Event Tickets

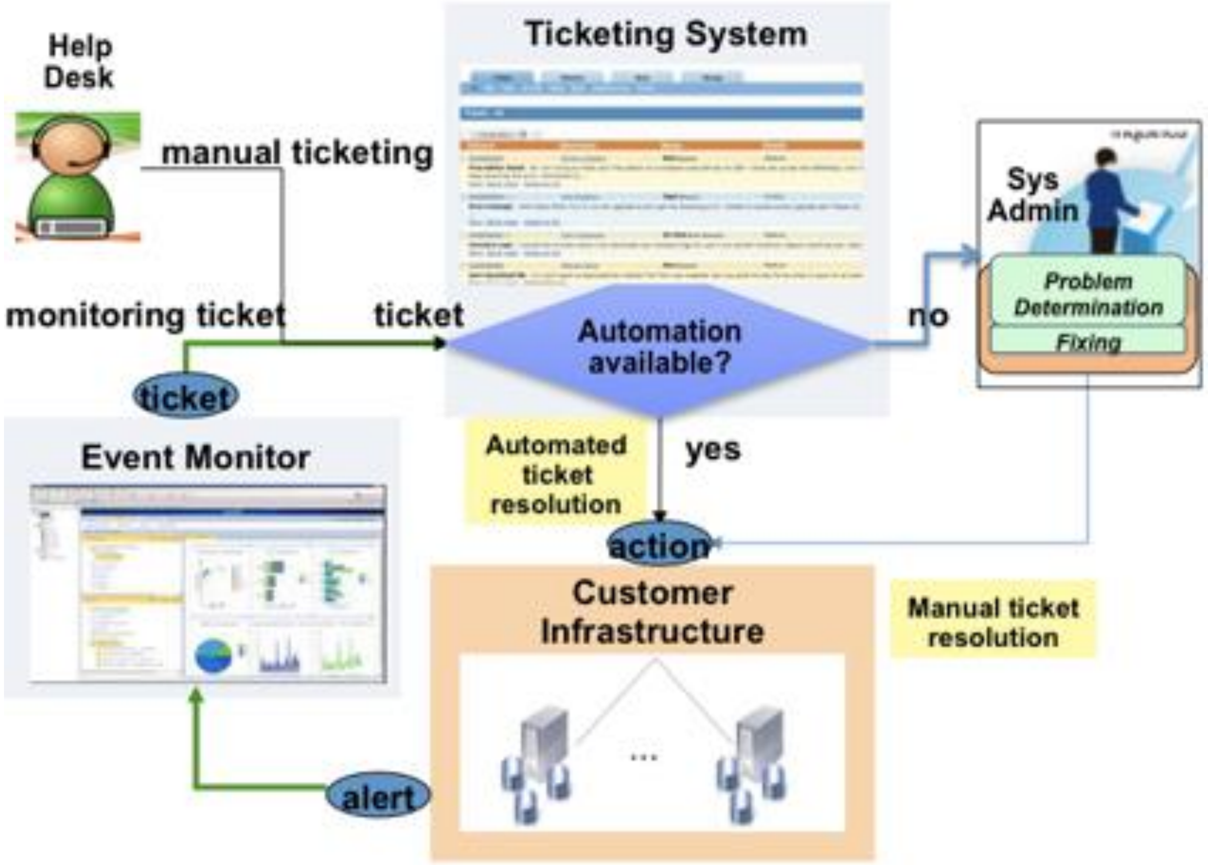
Data Set	# of tickets	Time Frame
account1	31,447	1 month
account2	37,482	4 month
account3	29,057	5 month





- Figure 1: Number of tickets and unique resolution for each account
- Figure 2: Number of tickets solved by the top most common resolutions
- **Conclusion**: Similar tickets resolved by similar resolutions

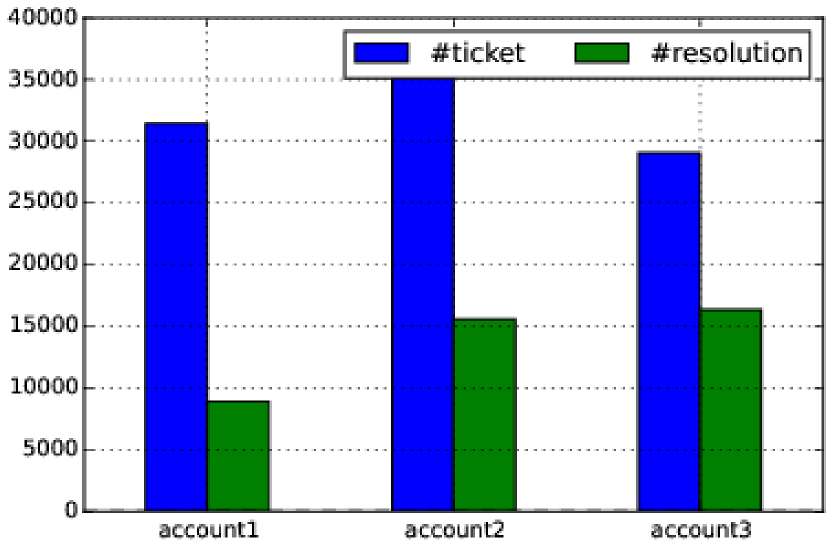
Motivations

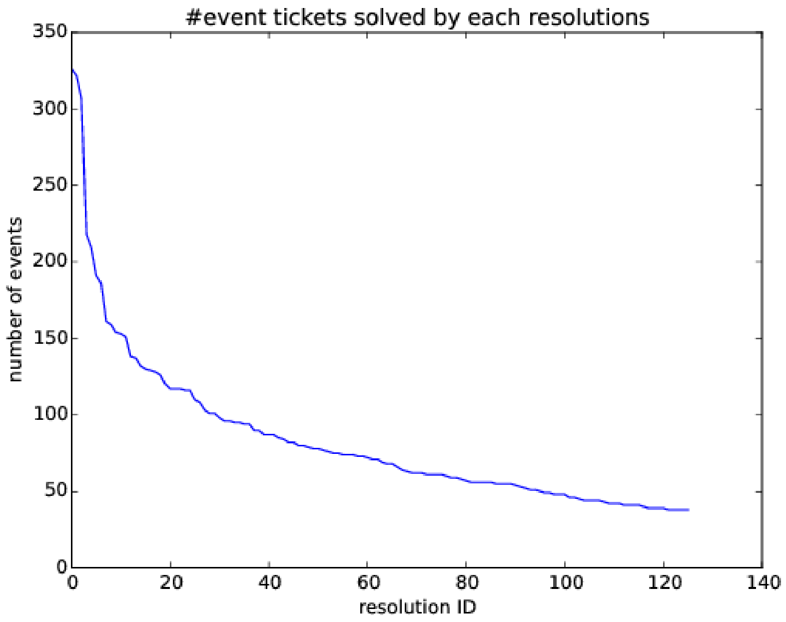


- Ticket resolving is very labor intensive
- Problems occurred periodically
- Repeated resolutions exist in historical tickets

Repeated Resolutions of Event Tickets

Data Set	# of tickets	Time Frame
account1	31,447	1 month
account2	37,482	4 month
account3	29,057	5 month





- Figure 1: Number of tickets and unique resolution for each account
- Figure 2: Number of tickets solved by the top most common resolutions
- Conclusion: **Similar tickets** resolved by similar resolutions

Related Work

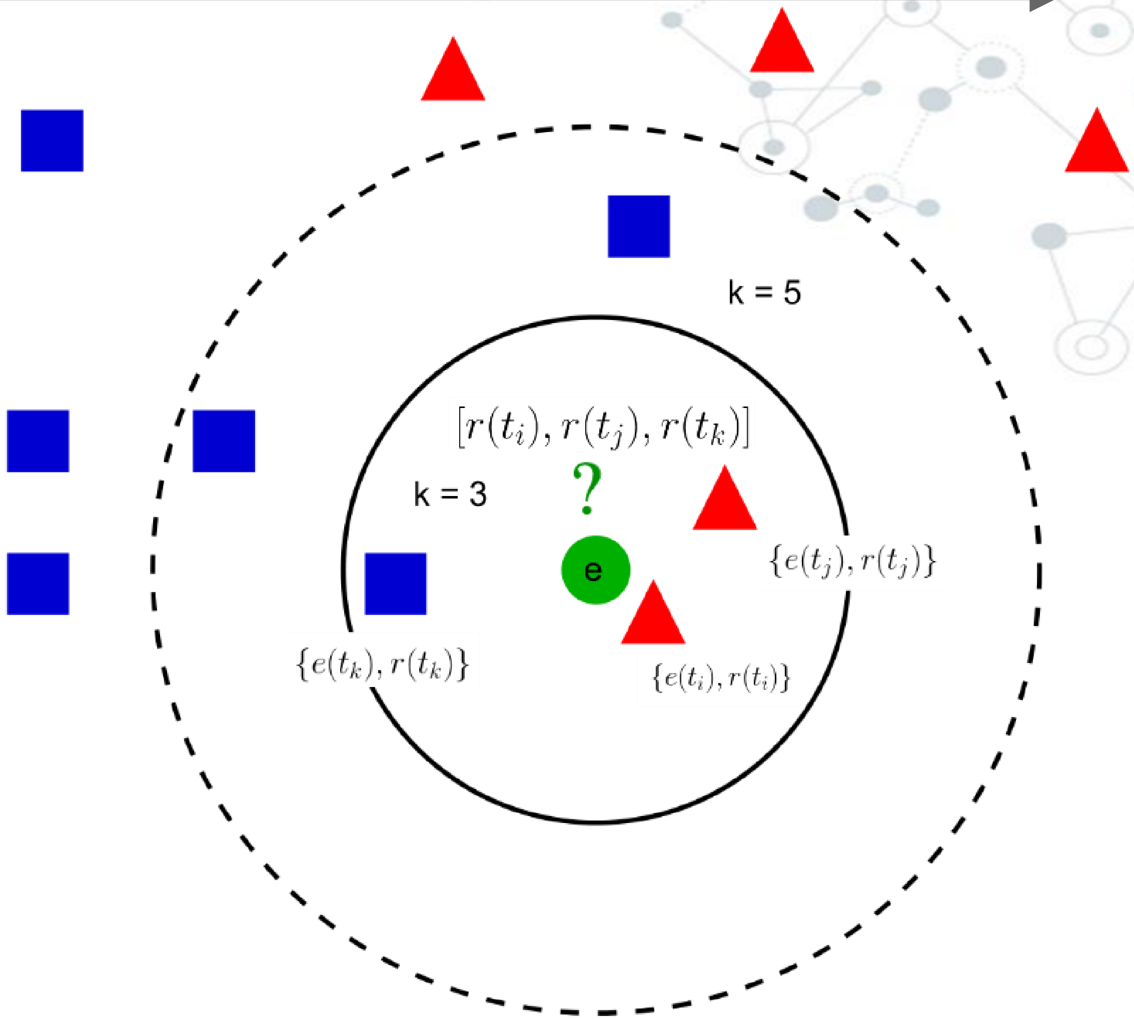
- User-based Recommendation Algorithms
- Item-based Recommendation Algorithms
- Constraint-based Recommender Systems
- Multiple Objective Optimization…

Existing Solution

- Every historical ticket t has two parts. e(t) is the symptom description, and r(t) is the resolution attached to it.

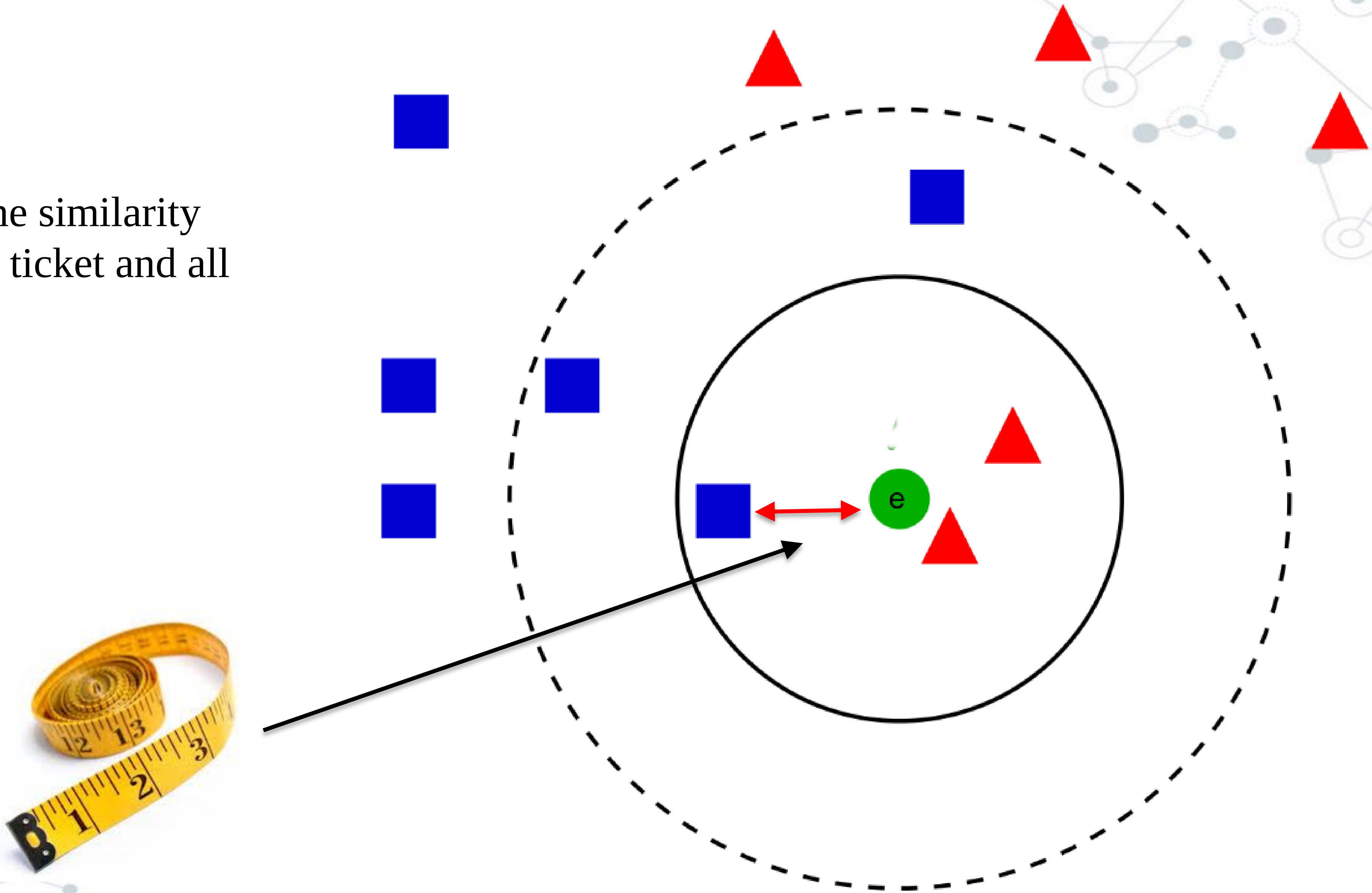
- Incoming ticket only has e(t) i.e., the symptom description

- User-based Top-K Recommendation (L. Tang et al., 2013)



Challenges

- How to measure the similarity between incoming ticket and all historical tickets ?



Challenges

- Given a ticket, how to encode it and represent it ?
  - is every attribute in tickets informative ?
  - How to featurize a ticket ?

- Given two tickets, how to measure their similarity ?
  - Tickets might be quite noisy
  - Ticket might be literally different but semantically similar

Type	Example
Categorical	OSTYPE, NODE, ALERTKEY,...
Numeric	SERVERITY, LASTUPDATE, ...
Textual	SUMMARY,...

T1: The logic disk has a low amount of space

T2: The percent of available space in the file system is 10 percent.

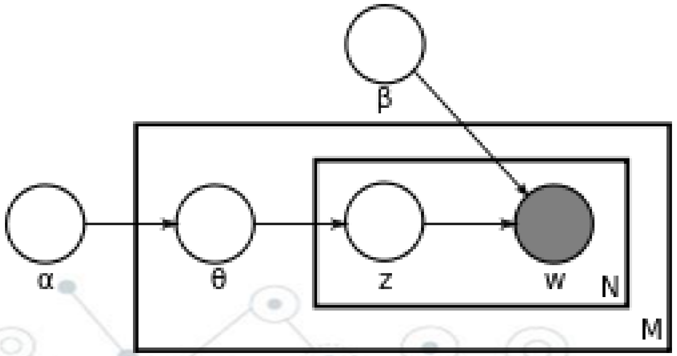
Prelminary work (Zhou , Tang, Zeng, Li, Shwartz, & Grabarnik, TNSM 2016)

Basic similarity measurement based on attribute level features

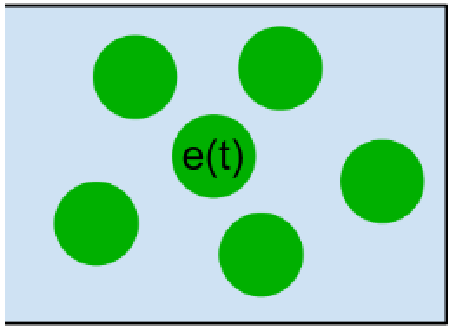
Type	Example
Categorical	OSTYPE, NODE, ALERTKEY,...
Numeric	SERVERITY, LASTUPDATE, ...
Textual	SUMMARY,...

$$sim_a(e_1, e_2) = \begin{cases} I[a(e_1) = a(e_2)], & \text{if } a \text{ is categorical,} \\ \frac{|a(e_1) - a(e_2)|}{max|a(e_i) - a(e_j)|}, & \text{if } a \text{ is numeric,} \\ Jaccard(a(e_1), a(e_2)), & \text{if } a \text{ is textual,} \end{cases}$$

Basic similarity measurement based on topic level features



training

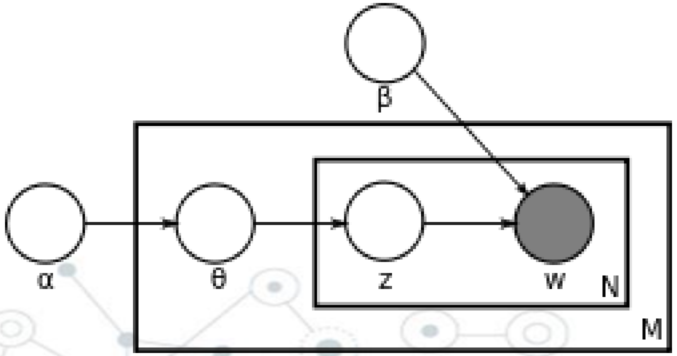


- Inference topic level feature vectors
- Apply cosine similarity

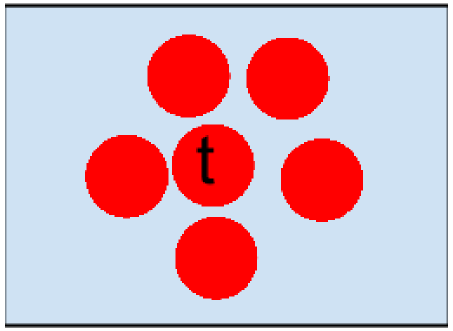
Incorporating Resolutions

ticketID	SUMMARY	RESOLUTION
1	The logical disk has a low amount of free space. Percent available: 2 Threshold: 5	After deleting old uninstall files, the logical disk has now over 10% of free disk space.
2	The percentage of used space in the logic disk is 90 percent. Threshold: 90 percent	After deleting old uninstall files, the logical disk has now over 15% of free disk space.
3	File system is low. The percentage of available space in the file system is 10 percent. Threshold: 90 percent	After delprof run, the server now has more than 4gb of free space
4	The logical disk has a low amount of free space. Percent available: 3 Threshold: 5	No trouble was found, situation no longer persists.

Implementation



training

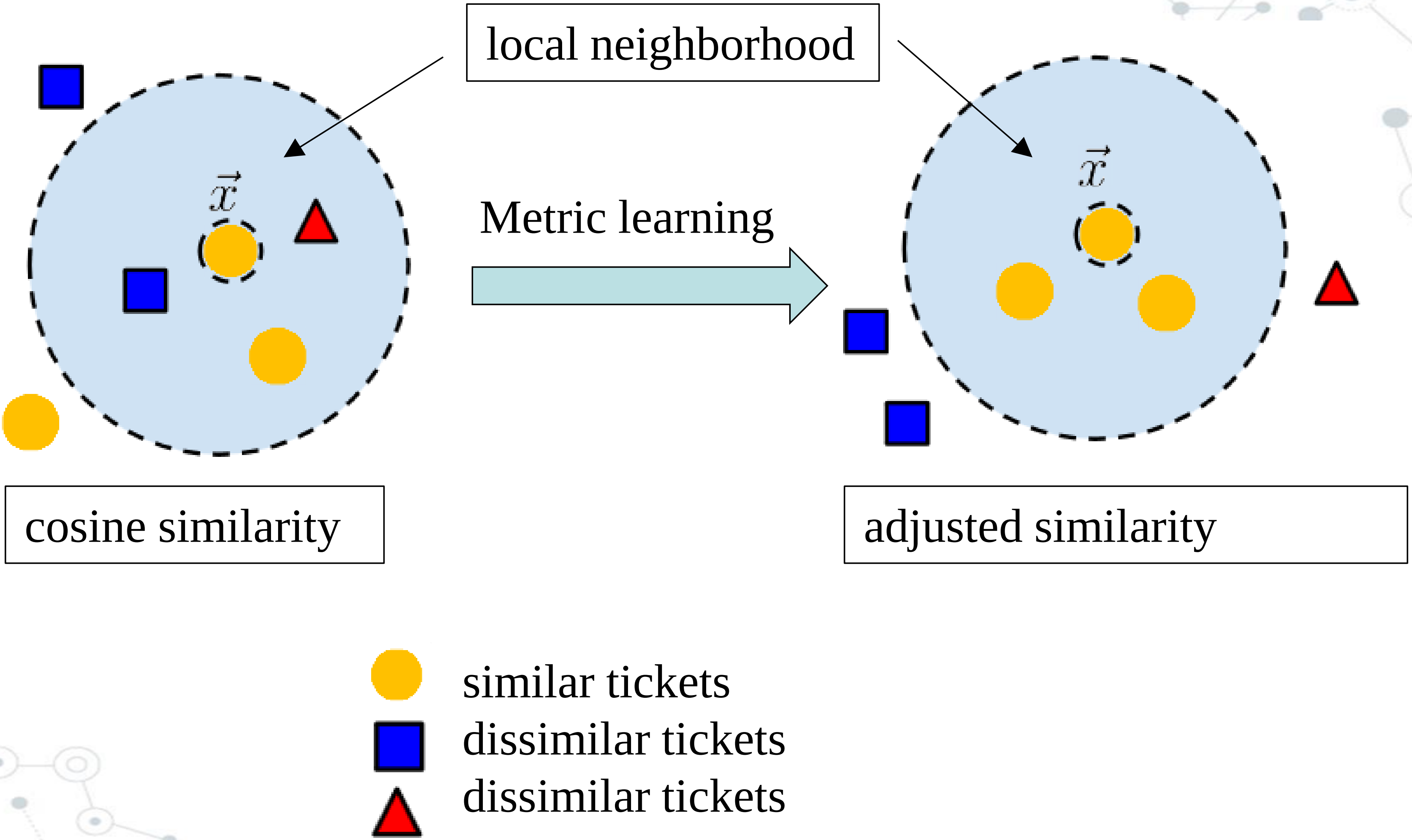


- Inference topic level feature vectors
- Apply cosine similarity

Feature differences

Topic ID	keywords
14	server wsfpp1 lppza0 lppzi0 nalac application
30	server hung condition responding application apps

Metric Learning



Implementation

$$
\begin{aligned}
f(A) &= min \sum_{i=1}^{n} \sum_{j=1}^{n} ||R_{i,j} - S_A(\vec{t_i}, \vec{t_j})||^2 \\
&= min||R - SAS^T||^2,
\end{aligned}
$$

Minimize this objective function, get projection matrix A for similarity calculation adjustment

$$
S_A(\vec{t_i}, \vec{t_j}) = \vec{t_i}^T * A * \vec{t_j}
$$

topic level feature vector for ticket j

$$
R_{i,j} = sim(r(t_i), r(t_j)) = \begin{cases} 1, & \text{if } r(t_i), r(t_j) \text{ are in same category,} \\ 0, & \text{otherwise.} \end{cases}
$$

Manually categorized label for ticket j

Outline

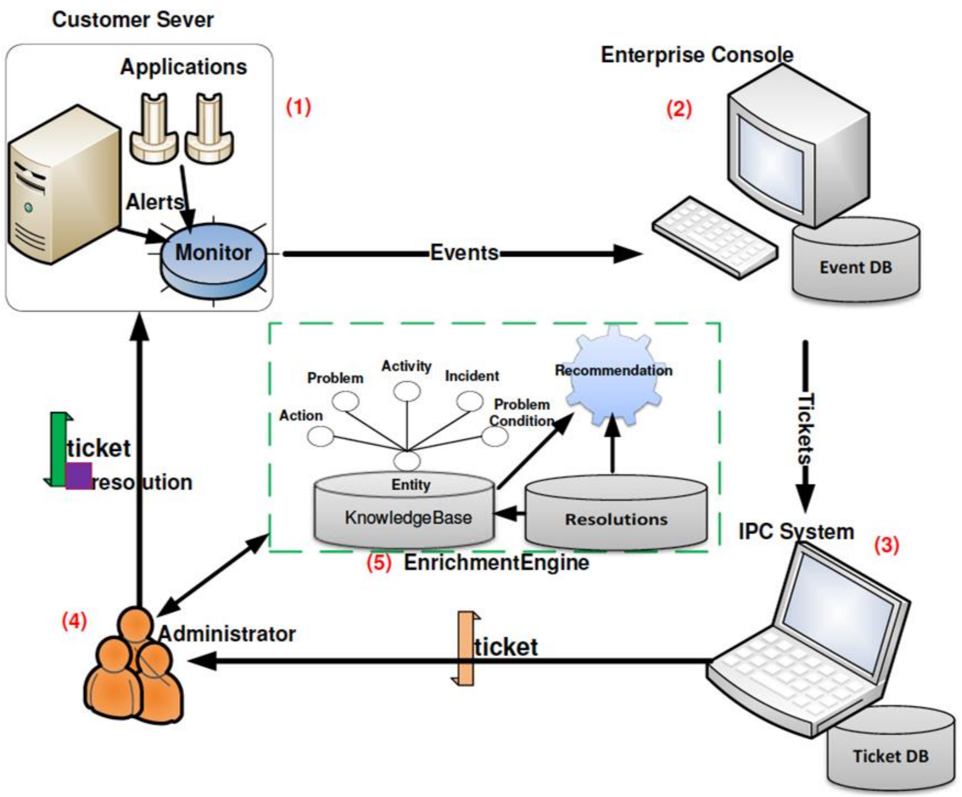
- Ticket Classification
- Ticket Resolution Recommendation
- Ticket Analysis (Knowledge Extraction)

Transitioning from practitioner-driven technology-assisted to technology-driven and practitioner-assisted delivery of services

- Enterprises and service providers are increasingly challenged with improving the quality of service delivery
- The increasing complexity of IT environments dictates the usage of intelligent automation driven by cognitive technologies, aiming at providing higher quality and more complex services.
- Software monitoring systems are designed to actively collect and signal anomalous behavior and, when necessary, automatically generate incident tickets.
- Solving these IT tickets is frequently a very labor-intensive process.
- Full automation of these service management processes are needed to target an ultimate goal of maintaining the highest possible quality of IT services. Which is hard!

Background

- Monitoring system: emits an event if anomalous behavior persists beyond a predefined duration.
- Event Management system: determines whether to create an incident ticket.
- IPC (Incident/Problem/Change) System: record keeping system that collects the ***tickets*** and stored them for tracking and auditing purposes.
- System Administrators (SAs): performs problem determination, diagnosis, and resolution.
- Enrichment Engine: uses various data mining techniques to create, maintain and apply insights generated from a *knowledge base* to assist in resolution of an incident ideally with an automation.
- This research focuses on Enrichment engine



The overview of IT service management workflow.

Motivation

Structured fields:

often inaccurate or incomplete especially information which is not generated by monitoring systems

Unstructured text:

written by system administrators in natural language. Potential knowledge includes:

1. What happened? **Problem**
2. What troubleshooting was done? **Activity**
3. What was the resolution? **Action**

STRUCTURED

TICKET IDENTIFIER:	WPPWA544:APPS:LogAdapter:NALAC:STARACTUAT_6600

NODE	FAILURECODE	ORIGINAL SEVERITY	OSTYPE	COMPONET	CUSTOMER
WPPWA544	UNKNOWN	4	WIN2K3	APPLICATION	XXXX

UNSTRUCTURED

TICKET SUMMARY:	STARACTUAT_6600 03/01/2014 04:30:28 STARACTUAT_6600 GLACTUA Market=CAAirMiles:Report_ID=MRF600:ReportPeriod From: 2014/02/01 to 2014/02/28:ErrorDesc=For CAAirMiles Actuate is out of balance with STAR BalanceMRF600 & MRF601 Counts. Reconcilation Difference = 2MRF600 & MRF601 Net Fee. Reconcilation Difference = 25MRF600 & MRF601 Gross Fee .Reconcilation Difference = 25

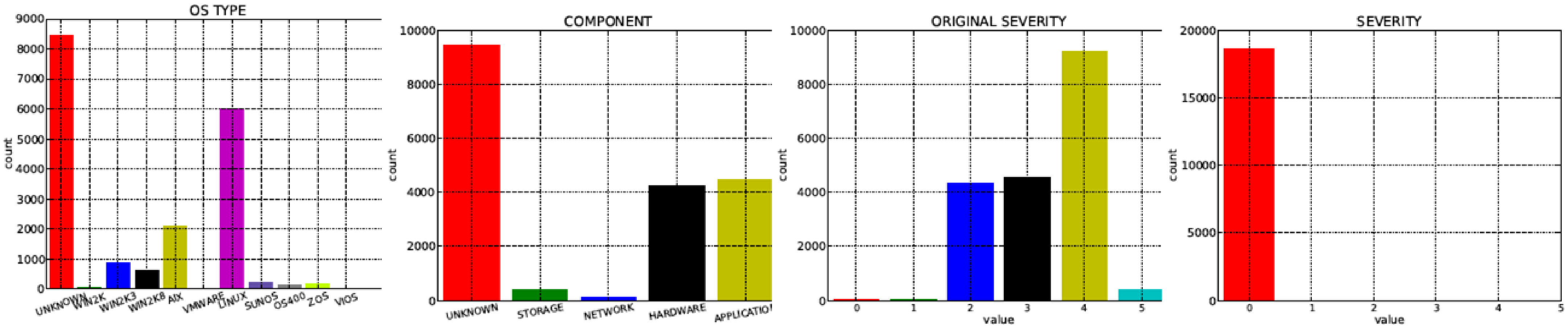
UNSTRUCTURED

RESOLUTION
ProblemSolutionText:***** Updated by GLACTUA ****** Problem Reported : Reconciliation difference Root cause : Reconciliation was run before all reports completed. This is as per the new SLAs. Solution provided : *Reconciliation was re-run after the next set of reports completed.* There was no user impact. Closure code : WRKS_AS_DSIGND RCADescription:***** Updated by GLACTUA ****** Problem Reported : Reconciliation difference Root cause : Reconciliation was run before all reports completed. This is as per the new SLAs. Solution provided : Reconciliation was re-run after the next set of reports completed.There was no user impact. Closure code : WRKS_AS_DSIGND

A ticket in IT service management and its corresponding resolution are given.

Challenge

- Challenge 1: Even in cases where the structured fields of a ticket are properly set, they either have small coverage or do not distinguish tickets well, and hence they contribute little information to the problem resolution
- Challenge 2: The ambiguity brought by the free-form text in both ticket summary and resolution poses difficulty in problem inference, although more descriptive information is provided



(a) OS Types. (b) Components. (c) Original severity. (d) Severity.

Ticket distribution with structure fields.

System Overview

- Our proposed integrated framework consists of three stages:

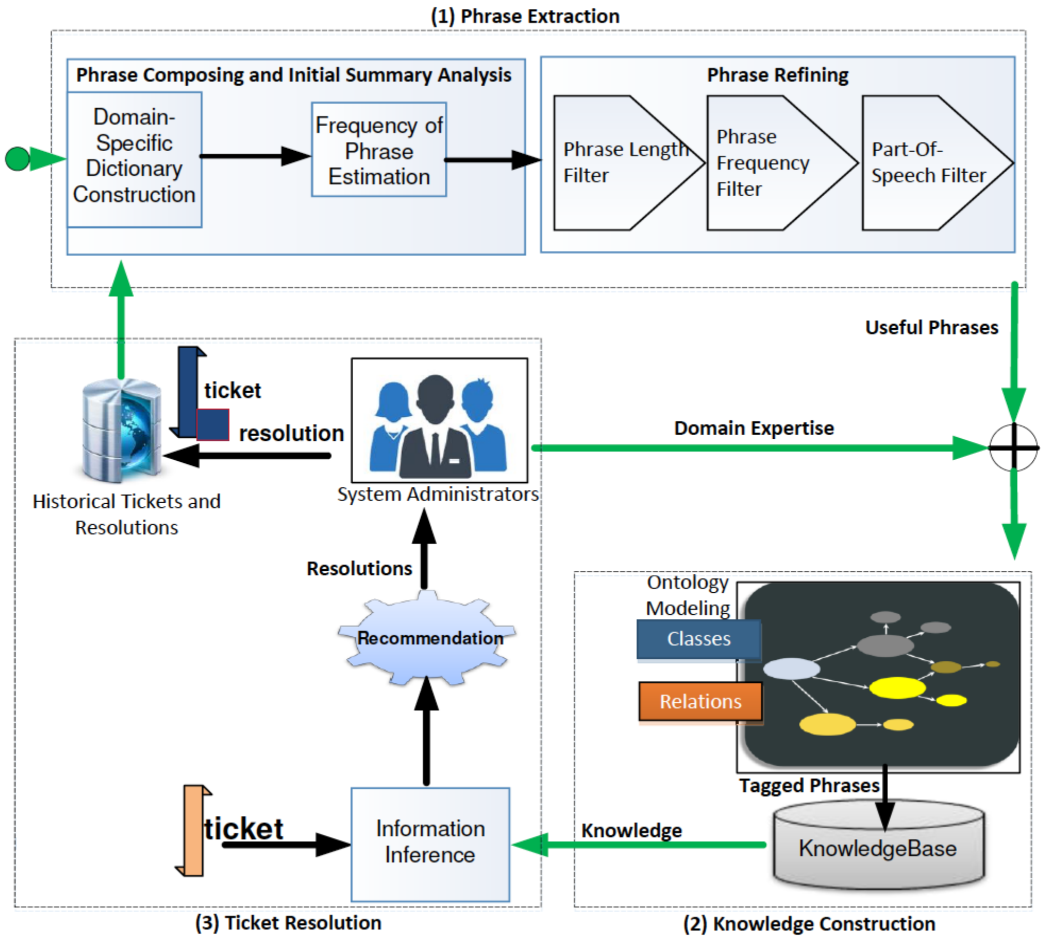
（1）Phrase Extraction Stage

(a) Phrase Composition and Initial Summary Analysis Component

(b) Phrase Refining Component

（2）Knowledge Construction Stage

（3）Ticket Resolution Stage

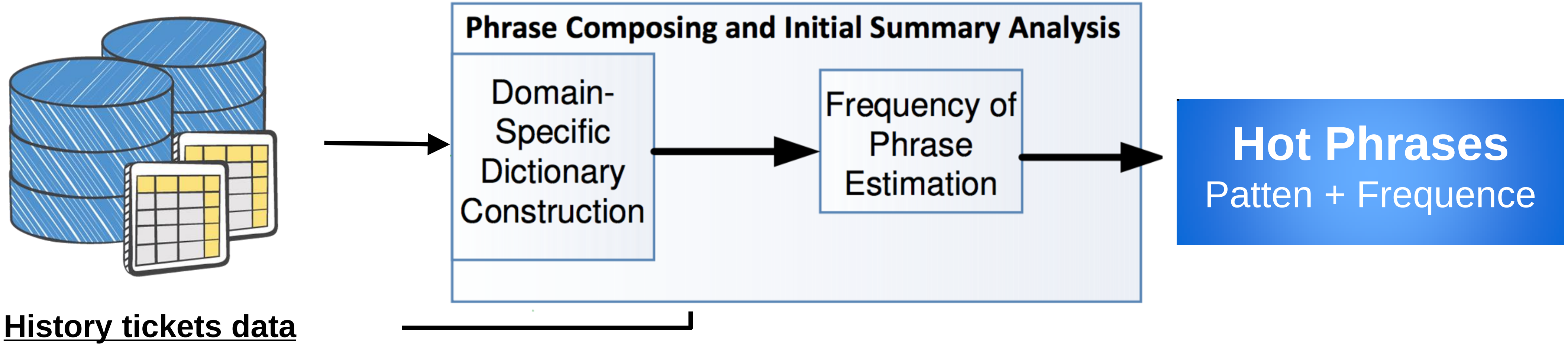


An overview of the integrated framework.

Phrase Extraction Stage

- In this stage, our framework finds important domain-specific words and phrases (‘kernel’).
  - Constructing domain-specific dictionary
    - Mining the repeated words and phrases from unstructured text field.
    - Refining these repeated phrases by diverse criteria filters (e.g., length, frequency, etc.).

Phrase Composition and Initial Summary Analysis



Repeated pattern extraction and frequency estimation.

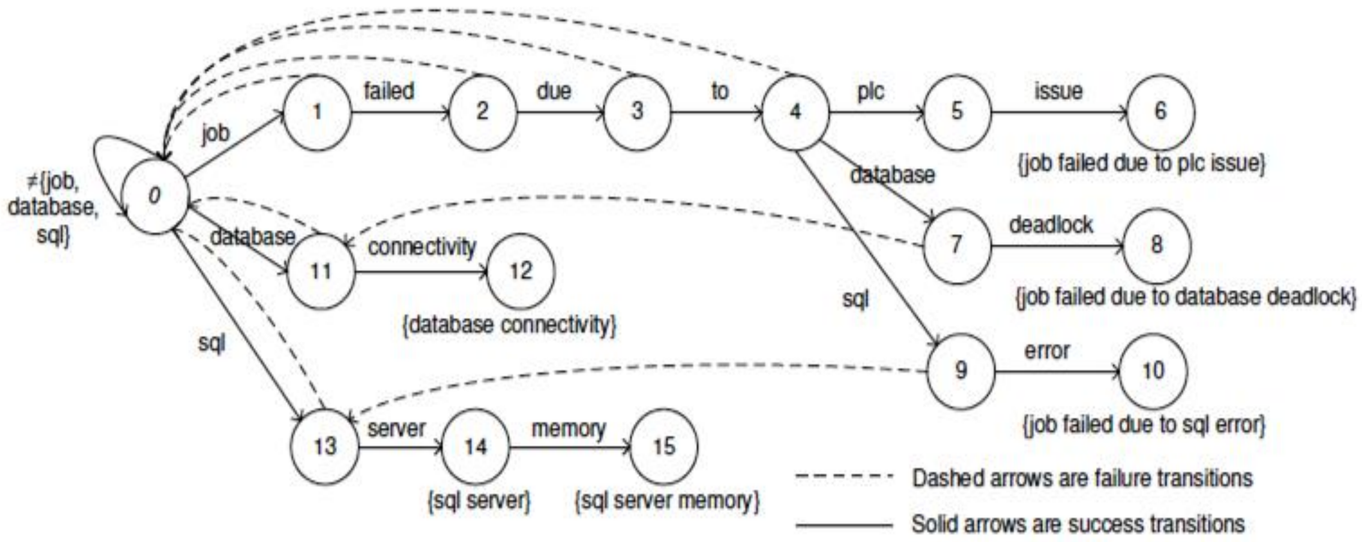
- Use StanfordNLPAnnotator for preprocessing ticket data.
- Build a domain dictionary by using Word-Level LZW compression algorithm.
- Calculate the frequency of the repeated phrases in tickets data by using Aho-Corasick algorithm.

Phrase Composition and Initial Summary Analysis

- Word-Level Lempel-Ziv-Welch (WLZW)
  - Seeks the trade-off between completeness and efficiency and attempts to find the longest n-gram with a repeated prefix
  - Time complexity: O(n)
- Aho-Corasick algorithm
  - Locate all occurrences of any of a finite number of keywords in a string of text.
  - Consists of constructing a finite state pattern matching machine from the keywords and then using the pattern matching machine processing the text string in a single pass.
  - Time complexity: O(n).

Phrase Composition and Initial Summary Analysis

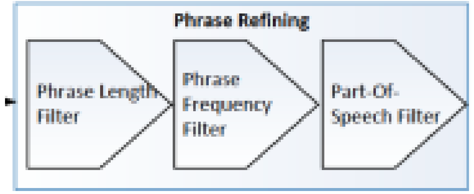
- Assume we have a dictionary D composing {
  “job failed due to plc issue,”
  “job failed due to database deadlock,”
  “job failed due to sql error,”
  “database connectivity,”
  “sql server v7.5,” “sql server v8,”
  “sql server memory”
  }.



An example of a finite state string pattern matching machine.

- AC algorithm first constructs finite State Automaton for dictionary using a Trie.
- And then estimates the frequency of the phrases in the dictionary for a single pass.

Phrases Refining



In this stage, we apply two filters to the extracted repeated phrases allowing the omission of <u>non-informative</u> phrases.

- Phrase Length & Frequency Filters (length > 20 & frequency >= 10)
- Part-Of-Speech Filter

Table I: Definition of technical term's schemes.

Justeson-Katz Patterns	Penn Treebank Entity Patterns	Examples in Tickets
A N	JJ NN[P|S|PS]*	global merchant
N N	NN[P|S|PS]* NN[P|S|PS]*	database deadlock
A A N	JJ JJ NN[P|S|PS]*	available physical memory
A N N	JJ NN[P|S|PS] NN[P|S|PS]	backup client connection
N A N	NN[P|S|PS] JJ NN[P|S|PS]	load balancing activity
N N N	NN[P|S|PS] NN[P|S|PS] NN[P|S|PS]	socket connectivity error
N P N	NN[P|S|PS] IN NN[P|S|PS]	failures at sfdc
A:Adjective, **N**: Noun, **P**: Preposition		
JJ: Adjective, **NN**: singular Noun, **NNS**: plural Noun, **NNP**: singular proper Noun, **NNPS**: plural proper Noun, **IN**: Preposition		

Table II: Definition of action term's schemes.

Penn Treebank Action Patterns	Examples in Tickets
VB[D|G|N]*	run/check, updated/corrected affecting/circumventing, given/taken
VB: base form Verb, **VBD**: past tense Verb, **VBG**: gerund Verb,**VBN**: past participle Verb,	

Table III: Result of Frequency/Length Filter and PoSTag Filter.

Applied Filter	Left Phrases
Frequency Filter >= 10	1117 items
Length Filter > 20	613 items
PoSTag Filter	323 items

Knowledge Construction Stage

- In this stage, we first develop an ontology model, and then tag all the phrases of the generated dictionary with the defined classes.
  - Build the ontology model
    - Define classes
    - Define relations
  - Knowledge Archive
    - Manually tag the important
- phrases in the dictionary with
- their most relevant defined classes.

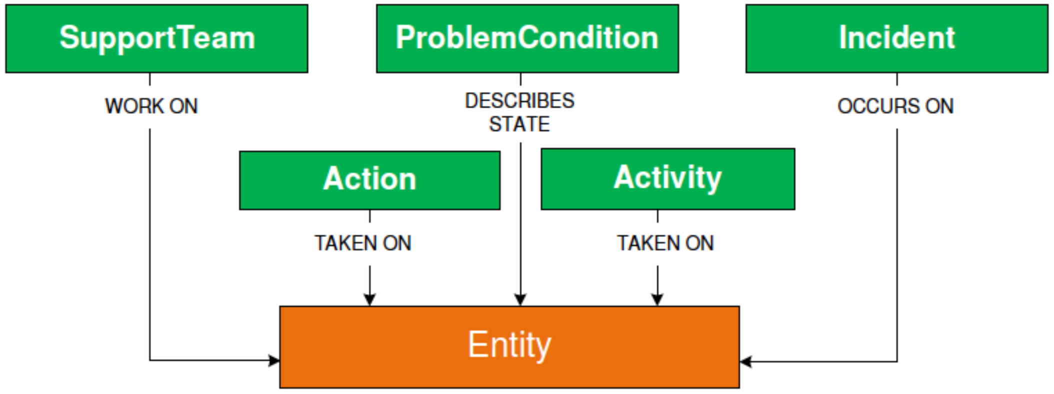


Figure 9: Ontology model depicting interactions among classes.

Class	Definition	Examples
Entity	Object that can be created/destroyed/replace	memory fault; database deadlock
Action	Requires creating/destroying an entity	restart; rerun; renew
Activity	Requires interacting with an entity	check; update; clean
Incident	State known to not have a problem	false alert; false positive
ProblemCondition	Describe the condition that causes a problem	offline; abended; failed
SupportTeam	Team that works on the problem	application team; databases team

Knowledge Construction Stage

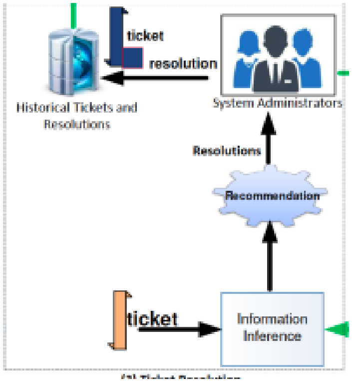
- Initial Domain Knowledge Base:

Entity	Activity	Action	ProblemCondition	Support Team
automated process	accept	reboot	abended	active direcory team
actual start	accepted	renew	bad data	app team
additional connection	achieved	rerun	deactived	application team
address information	acting	reran	disabled	aqpefds team
afr end	add	reset	dropped	bazaarvoice team
alert	added	restoring	expired	bmc team
alert imr	affecting	retransmit	fails	bsd team
alerts	affects	fixed	failed	Bureau team
alphanumeric values	altered	restart	false alert	business team
amex	aligned	restarted	false positive	bwinfra team
api calls	allocate	renewed	human error	cdm team
application	allocated	fixed	not working	CDM/GLEUDBD team
application code	applied	fixing	offline	cmit team
application impact	assign	recycle	stopped	control m team
atm messages	assigned	recycled	unavailable	convergys team
audit	blocks	recycling	under threshold	csp team
audit log	bring	reopen	wrong	cu team

	Number of Tagged Phrases
Entity	628 items
Activity	243 items
Action	24 items
Problem Condition	22 items
SupportTeam	76 items

Ticket Resolution Stage

The goal of this stage is to recommend operational phrases for an incoming ticket.



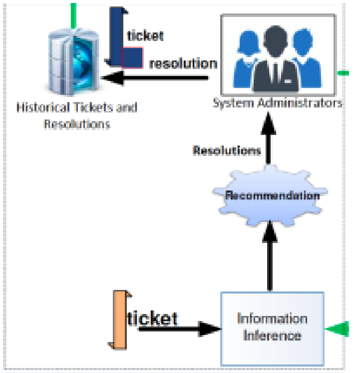
- Information Inference component:
- Class Tagger Module processes incoming ticket tickets in three steps.
  - (1) tokenize the input into sentences;
  - (2) construct a Trie by using ontology domain dictionary;
  - (3) find the longest matching phrases of each sentence using the Trie and knowledge base, then map them onto the corresponding ontology classes

  - Define Concept Patterns for Inference: concept patterns based on Problem, Activity and Action concepts:

1. Problem describes an entity in negative condition or state.
2. Activity denotes the diagnostic steps on an entity.
3. Action represents the fixing operation on an entity.

Concept	Pattern	Examples
Problem	Entity preceded/succeeded by ProblemCondition	(jvm) is (down)
Activity	Entity preceded/succeeded by Activity	(check) the (gft record count)
Action	Entity preceded/succeeded by Action	(restart) the (database)

Ticket Resolution Stage



- <u>Problem, Activity and Action Extraction:</u>
  - 1. Class Tagger module tokenizes the input into sentences and outputs a list of tagged phrases.
  - 2. We decide whether it is an informative snippet or not by checking if it exists in a Problem-Condition/Action list.
  - 3. The phrase is appended to the dictionary as a key, and all its related entities are added as the corresponding values via a neighborhood search. Each of the three key concepts has its own dictionary.

- Finally, we obtain the problem, activity, and action inferences.

(post loading)/(Entity) (failed)/(ProblemCondition) due to (plc issue)/(Entity). (updated)/(Activity) the (gft)/(Entity) after (proper validation)/(Entity) and (processed)/(Activity) the (job)/(Entity) and (completed)/(Action) successfully.

- Problem - {failed: plc issue, post loading}
- Activity - {update: gft, proper validation; process: job}
- Action - {complete: job}

Ticket Resolution Stage

- The goal of this stage is to recommend operational phrases for an incoming ticket.
  - Ontology-based Resolution Recommendation component
    - Previous study, the KNN-based algorithm will be used to recommend the historical tickets' resolution to the incoming ticket which have the top summary similarity scores.
    - Jaccard similarity performs poorly due to noisy text (many non-informative words): two tickets describes the same issue

Inside ProcessTransaction. DetermineOutcome failed. Database save failed: Tried an insert, then tried an update
CRPE3I1Server Database save failed on lppwa899 00:19:46 lppwa899 /logs/websphere/wsfpp1lppwa 899CRPE3I1Server/SystemOut.log [3/20/14 0:19:33:371 MST] 0000002b SystemOut 20140320 00:19:33, 371 [WebContainer:30] [STANDARD] [DI_US:01.22] (ng.AEXP_US_ISR_Work_Txn.Action) FATAL lp-pwa899—10.16.4.4—SOAP—AEXP_US_ISR_Roads3_Pkg —AEXPUSISRWork-Inquiry—ProcessInquiry

    - Ontology model can greatly facilitates our resolution recommendation task by better capturing the similarity between ticket summaries.

Experiment

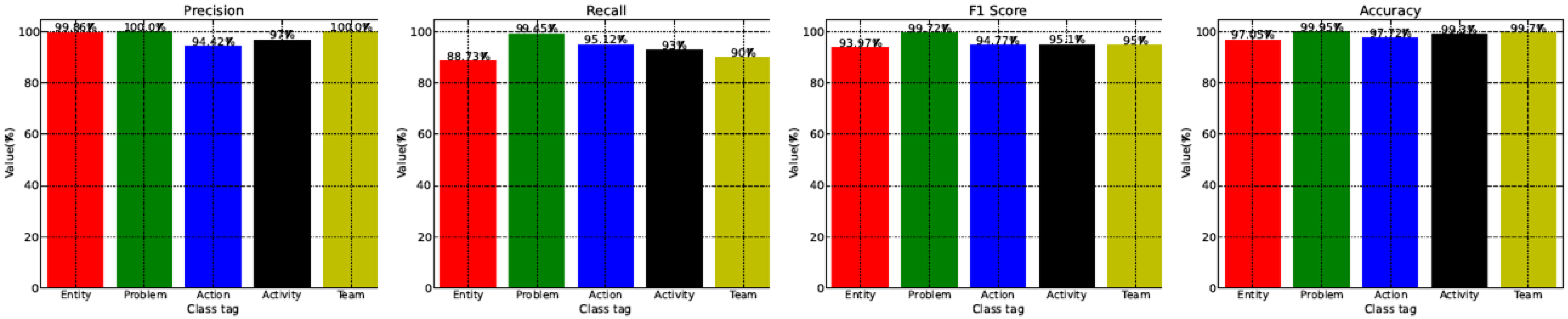
- Dataset
  - Experimental tickets are collected from real production servers of IBM Cloud Monitoring system covers three month time period containing $|D| = 22{,}423$ tickets.
  - Training data: 90% of total tickets
  - Testing data: 10% of total tickets

- Evaluation Metrics
  - Precision, Recall, F1 score and Accuracy.
  - Accuracy = (TP + TN)/(TP + TN + FP + FN)
  - Precision = TP/(TP + FP)   Recall = TP/(TP + FN)
  - F1 score = 2 Precision Recall / (Precision + Recall)

Experiment

- Ground Truth
  - Domain experts manually find and tag all phrases instances into six predefined classes in testing dataset.

- Evaluate our integrated system
  - Class Tagger is applied to testing tickets to produce tagged phrases with predefined classes. Comparing the tagged phrases with ground truth, we obtain the performance.



(a) Precision. (b) Recall. (c) F1-Score. (d) Accuracy.

Evaluation of our integrated system.

Experiment

- Evaluate Information Inference
    - Usability: we evaluate the average accuracy to be 95.5%, 92.3%, and 86.2% for Problem, Activity, and Action respectively.
    - Readability: we measure the time cost. Domain expert can be quicker to identity the Problem, Activity and Action which output from the Information Inference component from 50 randomly selected tickets.

Contents

1 Introduction and Overview

2 Event Generation and System Monitoring

3 Pattern Discovery and Summarization

4 Mining with Events and Tickets

5 Conclusions

Monitoring is the fundamental systems management operation

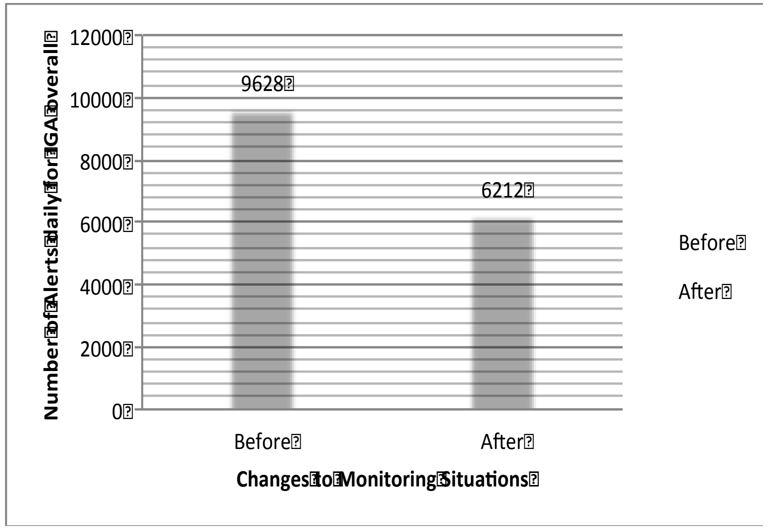
Needs to be optimized to meet quality expectations without expending excessive labor in managed environments

- Generically set monitoring situations and thresholds generate excessive alerts driving excessive consumption of labor
  - Impact can be exacerbated with auto-ticketing if not properly handled
- Adversely skews attention towards large volume of tickets instead of focusing on improvements – client value opportunity

–Initial study of sample accounts with automated ticketing shows that*:

- ***20-30%*** of Incident tickets are False Positives (tickets not requiring an immediate corrective action)
- ***10 - 15%*** of Labor relates to incident management

Account	% of monitoring tickets for the account	% of false positives tickets for the account
Accout_id_1	69.5%	21.0%
Accout_id_2	35.1%	6.7%
Accout_id_3	67.2%	27.7
Accout_id_4	90.0%	33.5%
Accout_id_5	40.3%	Many resolutions are blank:2.4%



** ROM analysis suggests that for a medium to large service provider the opportunity is in mil$*

Identification of complex or correlated event

⊟ per_dbinact_grzf_std & per_lstnr_grzf_std	6
⊟ spouxcrmd	2
spouxcrmd:2017-05-23 17:34:51	1
spouxcrmd:2017-05-18 11:11:03	1
⊟ spouxdbweb	1
spouxdbweb:2017-04-10 09:41:53	1
⊟ hor36	1
hor36:2017-06-28 10:30:16	1
⊟ spouxintbdp	1
spouxintbdp:2017-04-04 10:59:32	1

Suggested rule as combination of alerts

By Server

By 15 min Cluster

SUMMARY
[MONITORACAO_-_ORACLE_SEV1]Listener LISTENER_CJD is inactive.
[MONITORACAO_-_ORACLE_SEV1]Database Connection Name 'CJD' status is Inactive.

Detail of a sample 15 min Cluster

Benefit of Correlation Rules: significant QoS increase, reduction in ticket volume upto ***30%***

Ticketed Events	Events Associated With Earlier Ticket	Duplicates	Generic Correlation
11475	70	65	

Unique Tickets	Ticketed Events Already Correlated
11405	5

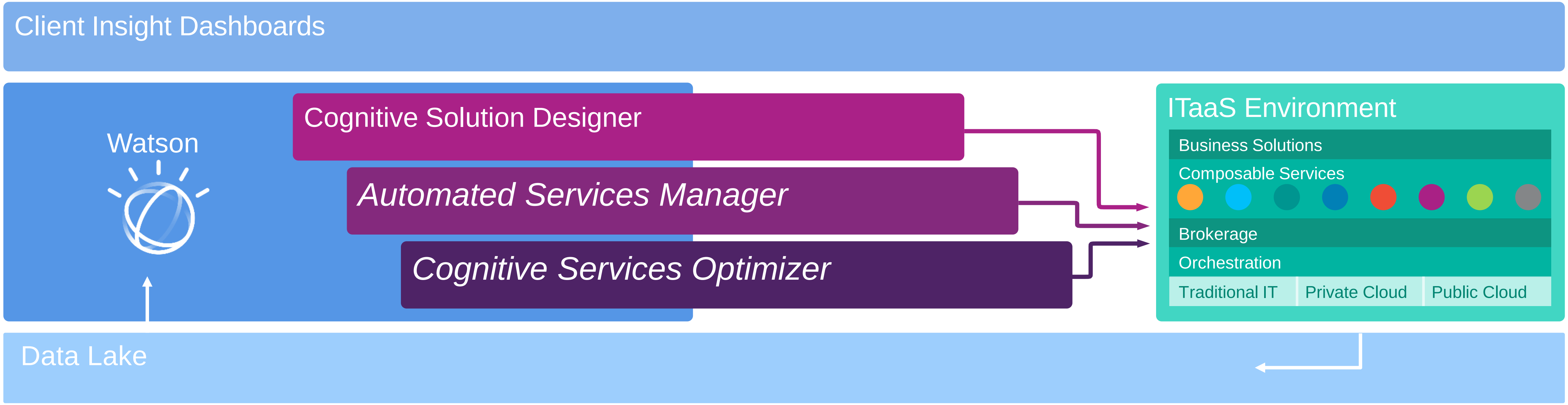
Generic Correlation Opportunity (Relative)	Generic Correlation Potential Currently Used
30%	0%

KPI "Correlation Opportunity" to track progress

% of tickets that can be reduced by implementing generic correlation rules for given account

Rule 1 Opportunity	Rule 2 Opportunity	Rule 3 Opportunity
935	2668	154

Cognitive IT Services Delivery Platform



•Next generation of easily consumed cognitive infrastructure services are basis for an open, standards-based, and integrated platform

•Continuously expanding the platform with additional cognitive and automation capabilities using

- operational data, historical and real-time,
- curated operational data, e.g. reports or insights derived from raw data and stored within the lake,
- automation content, patterns,
- knowledge, e.g. solutions, offering, reference architecture, cartridges, ontologies,
- user interaction, e.g. usage, feedback, customer satisfaction,
- meta data, e.g. data catalogues, taxonomies, classifiers, rules

What Log/Event Analysis Can Do I

- Proactively monitors system resources to detect potential problems and automatically respond to events.
  - By identifying issues early, it enables rapid fixes before users notice any difference in performance.

- Provides dynamic thresholding and performance analytics to improve incident avoidance.
  - This ‘early warning’ system allows you to start working on an incident before it impacts users, business applications or business services.

- Improves availability and mean-to-time recovery with quick incident visualization and historical look for fast incident research.
  - You can identify and take action on a performance or service interruption in minutes rather than hours.

What Log/Event Analysis Can Do II

- Collects data you can use to drive timely performance and capacity planning activities to avoid outages from resource over-utilization.
  - The software monitors, alerts and reports on future capacity bottlenecks.

- Facilitates system monitoring with a common, flexible and intuitive browser interface and customizable workspaces.
  - Can include an easy-to-use data warehouse and advanced reporting capabilities..

Looking Forward

- Real-time requirements
- Failure prediction
- Incorporation domain knowledge with mining results
- Integration of different types of information
- From systems to networks and devices
- Limited labeled data
- Interpretation and Transparency

Acknowledgements

- **Funding**
  - National Science Foundation
  - IBM Research
  - Simons Foundation

- **Ph.D. Students working on Log/Event/Ticket mining**
  - Dr. Wei Peng (Google)
  - Dr. Yexi Jiang (Facebook)
  - Dr. Liang Tang (LinkedIn)
  - Dr. Chao Shen (Amazon)
  - Dr. Chunqiu Zeng (Google)
  - Wubai Zhou (FIU)
  - Qing Wang (FIU)

Email: taoli@cs.fiu.edu
lshwart@us.ibm.com
grabarng@stjohns.edu

All the slides and references can be found at
http://www.cs.fiu.edu/~taoli/event-mining

Thank you!