Constructing the Knowledge Base for Cognitive IT Service Management

Qing Wang[1], Wubai Zhou[1], Chunqiu Zeng[1], Tao Li[1],

Larisa Shwartz[2]

Genady Ya. Grabanrnik[3]

[1]School of Computing and Information Science, Florida International University Miami, FL, USA
[2]Cognitive Service Management, IBM T.J. Watson Research Center Yorktown Heights, NY, USA
[3]Dept. Math & Computer Science, St. John's University, Queens, NY, USA

Outline

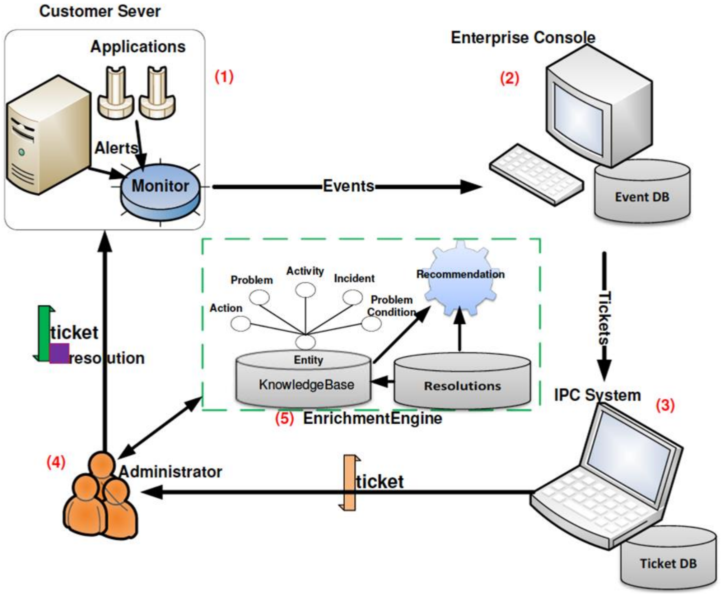
- ➔ Introduction
- ➔ System Overview
- ➔ Approaches to construct the knowledge base
  - ◆ Phrase Extraction Stage
  - ◆ Knowledge Construction Stage
  - ◆ Ticket Resolution Stage
- ➔ Experiment
- ➔ Conclusion and Future

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

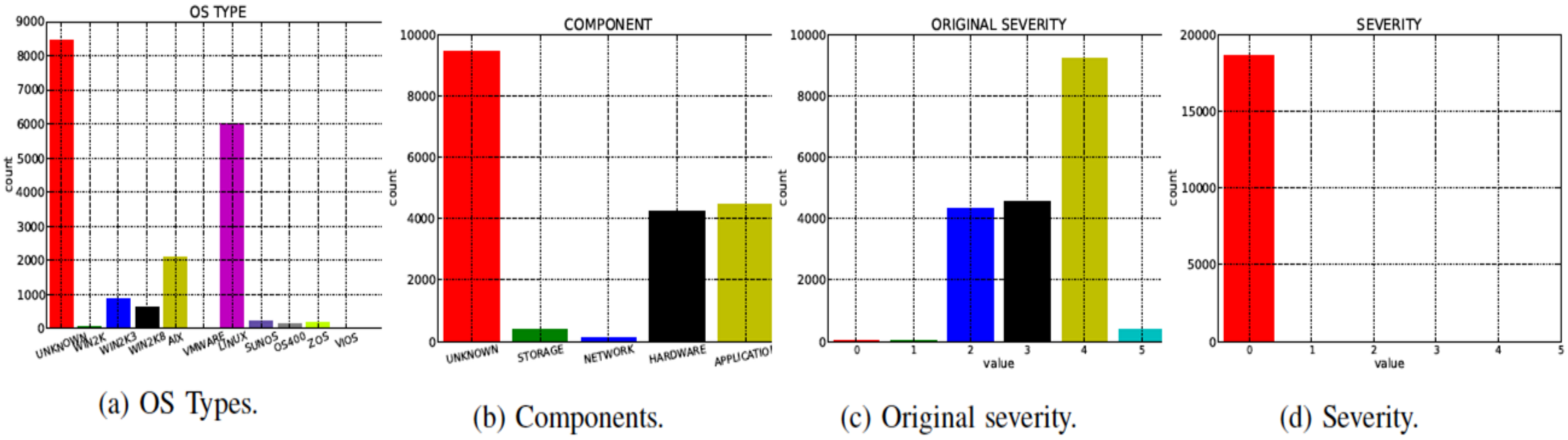
RESOLUTION

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:

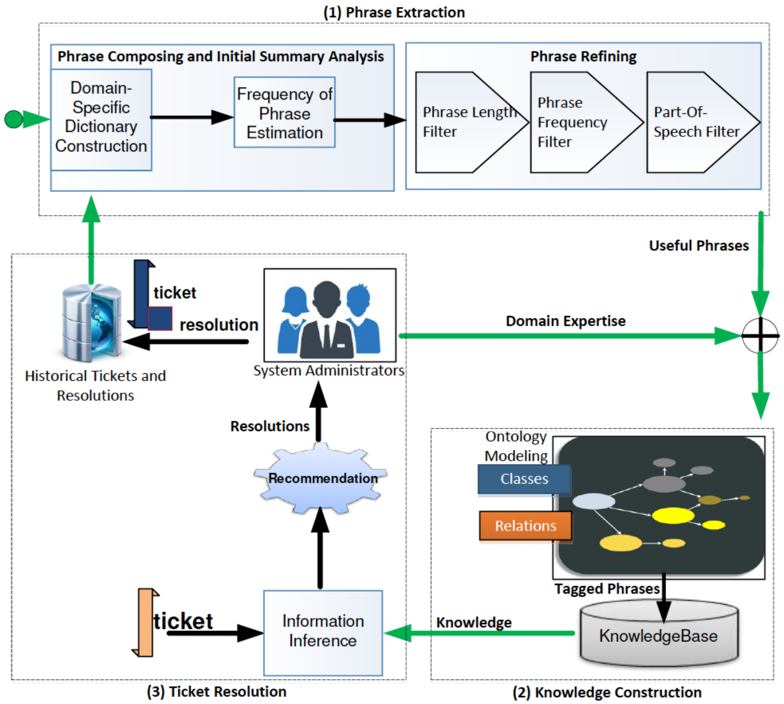
(1) Phrase Extraction Stage

(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

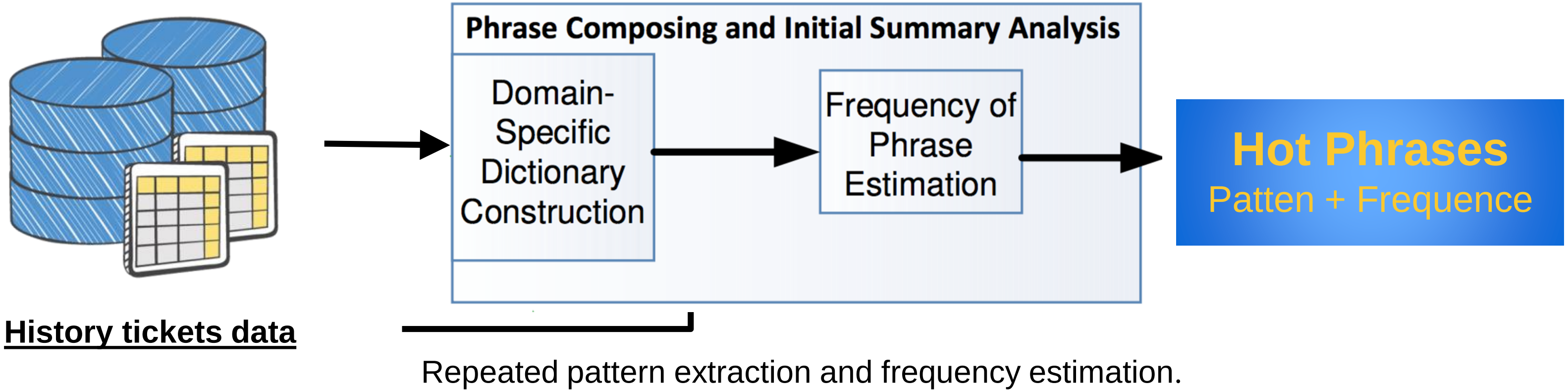
➔ In this stage, our framework finds important domain-specific words and phrases ('kernel').

◆ 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



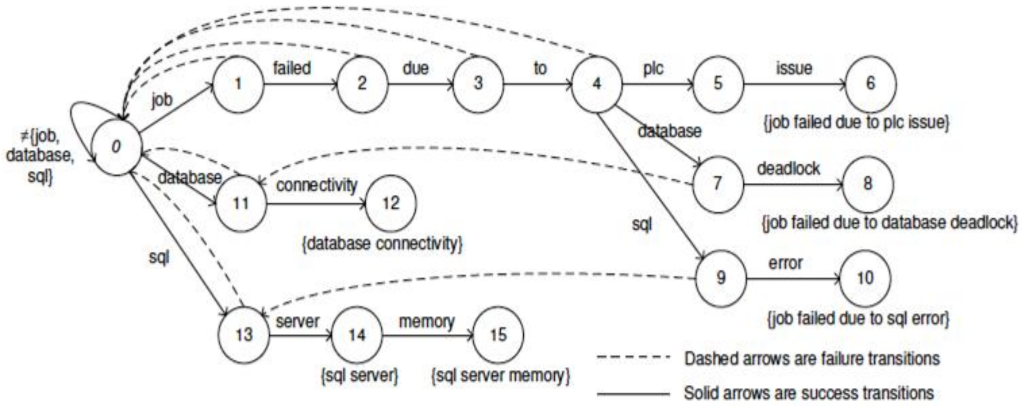
- ➔ Use StanfordNLPAnnotator for preprocessing ticket data.
- ➔ Build a domain dictionary by using Word-Level LZW compression algorithm.
- ➔ Calculate the frequency of the repeated phrases in tickets data by using Aho-Corasick algorithm.

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,"
"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

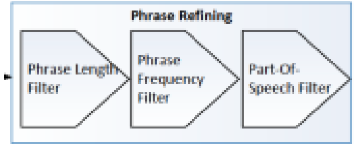


Table I: Definition of technical term's schemes.

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.

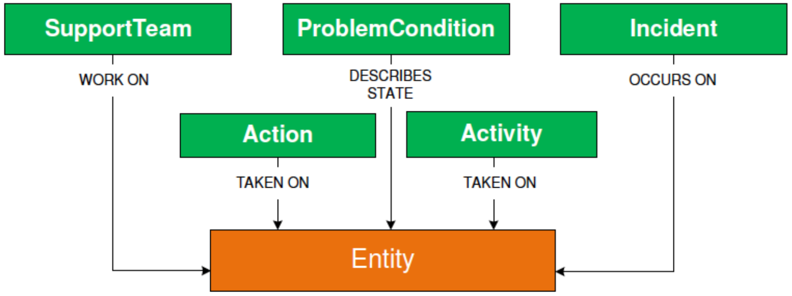


Figure 9: Ontology model depicting interactions among classes.

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

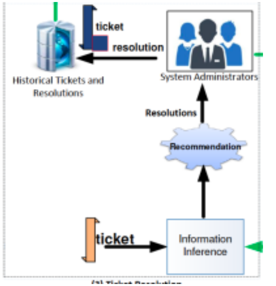
➔ Initial Domain Knowledge Base:

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

Class	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:

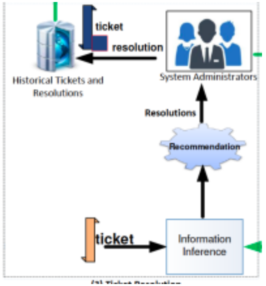
- Class Tagger Module processes incoming ticket tickets in three steps.
  (1) tokenize the input into sentences;
  (2) construct a Trie by using ontology domain dictionary;
  (3) find the longest matching phrases of each sentence using the Trie and knowledge base, then map them onto the corresponding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



- Problem, Activity and Action Extraction:

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)

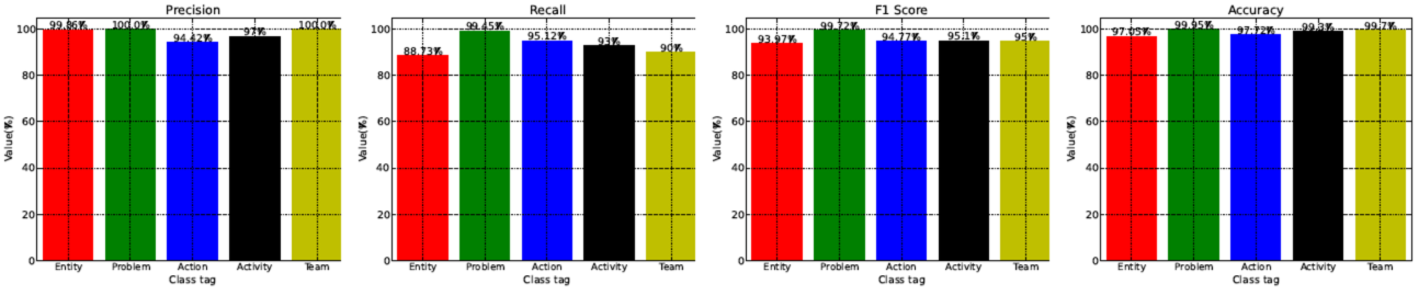
Experiment

➔ 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.

Conclusion and Future Work

➔ Contribution

- A novel domain-specific approach.
- Utilization of the ontology modeling techniques.
- Automation improvement of IT service management.
- A closed feedback loop system for continuously extending of the knowledge base.

➔ Future Work

- Investigate intelligent techniques to reduce human efforts on phrase tagging, such as training a conditional random field model.
- Leverage the ontology into Deep Learning model.
- Incorporate the obtained knowledge base into other tasks in the IT service management system.

Q & A