Towards Auto-remediation in Services Delivery: Context-Based Classification of Noisy and Unstructured Tickets

Gargi B. Dasgupta, Tapan K. Nayak, Arjun R. Akula,
Shivali Agarwal, and Shripad J. Nadgowda

IBM Research, Bangalore, India
{gaargidasgupta,tapnayak,arakula,shivaaga,nadgowdas}@in.ibm.com

Abstract. Service interactions account for major source of revenue and employment in many modern economies, and yet the service operations management process remains extremely complex. Ticket is the fundamental management entity in this process and resolution of tickets remains largely human intensive. A large portion of these human executed resolution tasks are repetitive in nature and can be automated. Ticket description analytics can be used to automatically identify the true category of the problem. This when combined with automated remediation actions considerably reduces the human effort. We look at monitoring data in a big provider's domain and abstract out the repeatable tasks from the noisy and unstructured human-readable text in tickets. We present a novel approach for automatic problem determination from this noisy and unstructured text. The approach uses two distinct levels of analysis, (a) correlating different data sources to obtain a richer text followed by (b) context based classification of the correlated data. We report on accuracy and efficiency of our approach using real customer data.

1 Introduction

A *Service System (SS)* is an organization composed of (a) the resources that support, and (b) the processes that drive service interactions in order to meet customer expectations. Due to the labor intensive processes and their complex inter-dependencies, these environments are often at the risk of missing performance targets.

To mitigate this risk and conforming with the underlying philosophy of "what gets measured, gets done", every SS defines a set of measurement tools that provide insights into the performance of its operational processes. One such set of tools include the event management and ticketing tools. Event management is a key function for monitoring and coordinating across several infrastructure components. Ticketing systems record problems that are logged and resolved in the environment. When integrated with the event management setup, ticketing systems enable proactive and quick reaction to situations. Together they help in delivering continuous up-time of business services and applications.

Figure 1 shows an integrated event management and ticketing system, that traces the life-cycle of a problem ticket in the customer's domain. Lightweight agents or probes

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 478–485, 2014.

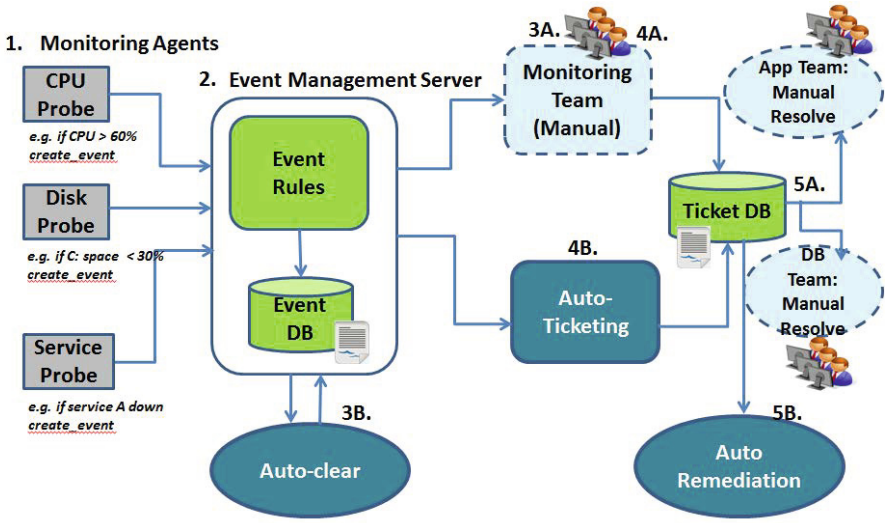

Fig. 1. Integrated event management and ticketing system

(shown on the left of Figure 1) are configured to monitor the health of the infrastructure including servers, storage and network. The collected data is fed into the event management server (i.e. component 2) whose main functions include: (a) continuous collection of real time data from probes on endpoints. (b) For each of the data streams, configure individual event rules to indicate when an event should be triggered. Some examples of an event rule are: *CPU usage is above a utilization threshold*, *available disk space is below a space threshold*, *service not running* etc. All the generated events are stored in event management database which can process up to a million events per day. All events could be routed to a manual monitoring team (3A). That team monitors the consoles for alerts, clears the ones that are informational, and raise tickets (4A) in the Ticket DB for the critical ones. Manual handling of large volume of events results in an human-intensive component. In contrast, automated handling (3B) of events, enables auto-clearing of certain classes of bulk alerts. Some event management systems also allow remediation actions to be automatically triggered on event occurrence. If the action is successful the event is auto-cleared and no ticket is created. If not, a ticket is raised automatically (4B). The path:1-2-3A-4A comprises the manual workflow for event-to-ticket handling whereas path:1-2-3B-4B comprises auto-ticket handling. At present majority of the systems continue to operate in manual workflow mode. The main reason for this is that the auto-ticketing causes large volume of tickets. In the absence of reliable auto-clear systems, all tickets have to be manually closed by the monitoring team, thereby adding more manual work in the system.

Thus in a service delivery environment (SDE), the auto-classification of the ticket symptom, and auto-remediation through a corrective action is critical. There has been many relevant works [6], [7], [4], [1], [3], [5] in the area of ticket analytics on structured and unstructured data. What makes SDE data particularly challenging is that it is extremely noisy, unstructured and often incomplete. In this paper we present a novel approach for automatic problem determination from the noisy and unstructured text.

The work in [1] comes very close to the text classification approach used in our work. It shows the limitation of SVM-like techniques in terms of scalability and proposes a notion of discriminative keyword approach. However, it falls short of using context based analysis to refine the results which is one of the key differentiating factors in our approach. Another work based on discriminative term approach is [2]. This work focuses on commonly used text classification data sets rather than service tickets. The work in [7] approaches the problem by mining resolution sequence data and does not access ticket description at all. The rest of the paper is organized as follows. Section 2 introduces our 2-step approach for correlating ticket with event data and classifying the correlated data using context based classification. In Section 3, we present our experimental results and we conclude with Section 4.

2 Analyzing Events and Tickets

2.1 Correlation Model

Because ticket resolutions are completely human generated, the text is often incomplete, noisy and highly contextual. Hence off-the shelf tools that are domain independent fail miserably while trying to understand context. However since every ticket originates from an event, and event data contains regularized text expressions, correlating the event data with ticket data can improve the classification.

If an event is auto-ticketed, then the automatic methods are reliable enough to ensure that the ticket identifier mapping exists in event and in the ticket data. In this case correlation is a simple join operation on ticket numbers. However when tickets are manually generated, this correlation is lost. Operators creating the tickets often do not update event data with the ticket numbers. As discussed in section 1 in reality majority of tickets continue to be generated manually. Hence in the absence of ticket numbers, for joining events and tickets we need smart correlation methods to decipher the event which created the ticket. We propose a correlation method based on multiple parameters and domain information available from the event and ticket information systems.

1. *Timestamp:* A ticket occurrence is *always* preceded by an event. Hence we use a timestamp threshold to narrow down the possible event choices for a ticket. Next, the following specific identifiers help further narrowing-down the correct original event.
2. *Server Name:* Matching server names in the event and ticket problem description indicate higher probability of correlation.
3. *Problem Description:* Similar descriptions in ticket and event fields have a higher probability of correlation. For example, if a ticket and event both describe database connectivity issues, then they are likely to be correlated. Text descriptions are matched w.r.t. syntax and context similarity using methods described in the following section.
4. *Group:* Additional information like application groups can help in event and ticket correlation. However this information may not be present in all data sources.

When the above heuristic is able to successfully identify the original event, the resulting text is enriched. The event text can be used for generic problem determination

while the ticket text gives context specific details. This combination is used for classification as described in the following section. In cases where the heuristic fails to identify the correct relevant event and the confidence is low, we proceed with ticket text only.

2.2 Classification Model

In this section we present our approach to classify the noisy and unstructured event and/or ticket text to pre-defined output categories. For the rest of this section, we use the term tickets to refer to both noisy event and ticket text descriptions. Firstly, we define a logical structure for the unstructured and noisy tickets. Next we categorize and classify the tickets based on the contextual information in logical structure.

Logical Structure for Unstructured and Noisy Tickets: Semantics is required to understand the contextual information in tickets. Identification of semantic information in the unstructured and noisy service delivery tickets is difficult. Furthermore, these tickets are syntactically ill-formed sentences. Hence we define a logical structure for these tickets as shown in Figure 2. The logical structure contains two sub-structures: category dependent and category independent. Category dependent structure stores the information corresponding to the specific output category. Category independent structure stores the information present in ticket which is independent of the output categories. Below we describe components of each of these two sub-structures.

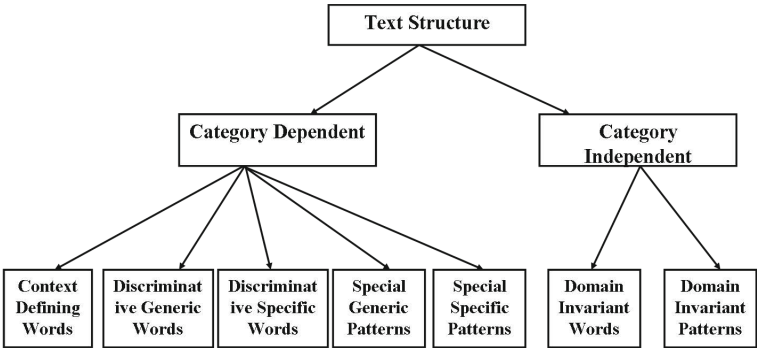

Fig. 2. Logical structure of Text

(a) *Discriminative specific words*: help in discriminating a output category from other categories.
(b) *Discriminative generic words*: help in discrimination of categories but less specific in comparison to discriminative specific words.
(c) *Context defining words*: constitute the contextual keywords. They by themselves do not help in discrimination but are useful to capture the contextual information of a category.
(d) *Special specific patterns*: regular expressions which help in discriminating a output category from other categories.
(e) *Special generic patterns*: regular expressions for discrimination but less specific in comparison to special specific patterns.

(f) *Domain invariant words*: help in identifying the contextual information. Context defining words help in identification of contextual information related to a particular category, whereas domain invariant words help in identification of contextual information in general.
(g) *Domain invariant patterns*: regular expressions which help in identifying the contextual information.

Table 1. Sample words and patterns for discriminating Disk C Full category

Category Dependent					**Category Independent**	
Context defining words	Discriminative generic words	Discriminative specific words	Special generic patterns	Special specific patterns	Domain invariant words	Domain invariant patterns
physical, percent, threshold, free	storage, space, volume, full	disk, drive	<none>	*c: drive*	too, not, need	*db2*, *.com*

For example, consider the *Disk C Full* output category. For discriminating this category from other output categories, words such as *disk*, *drive* and patterns such as *c: drive* will help most when compared to words such as *percent*, *threshold*, etc (Table 1). Words such as *not*, *too*, *db2* help in identification of contextual information.

For each of the pre-defined output categories, we instantiate a logical structure and populate the fields of the structure with domain specific keywords and patterns (which we call as domain dictionary). By comparing with the fields of logical structure of a category, we assign the words/tokens in the ticket to the appropriate fields of logical structures of the ticket. For a ticket, we define and populate N logical structures corresponding to N output categories. Category independent structure remains same in all these N logical structures. The following notation is used throughout this paper:
Let t_i denotes the i^{th} ticket, L_i denotes the set of logical structures of the ticket t_i, l_{ik} denotes the k^{th} logical element from the set L_i and f_{jik} denotes the j^{th} field in the logical structure l_{ik}. We define the pair of any two logical structures (l_{ij}, l_{ik}) of a ticket t_i as **contextually disjoint** if, for all the fields in the structure, either of the corresponding fields is empty. i.e. $\forall m,$ either $f_{mik} =$ empty *or* $f_{mij} =$ empty. Based on the contextual disjointness, we categorize the tickets into the following two categories:
(a) *Simple Tickets*: if all the highly ranked logical structures of a ticket are contextually disjoint.
(b) *Complex Tickets*: if any two highly ranked logical structures of a ticket are not contextually disjoint.

Classification of simple tickets: We use a linear weight based approach to score the logical structures of ticket. The output category corresponding to highest scored logical structure is assigned to the ticket. Weights are assigned to various fields of logical structure based on their discriminative capability. For example, keywords belonging to discriminative specific words gets higher weight compared to discriminative general words.

Classification of complex tickets: As the logical structures of complex tickets are not contextually disjoint, linear weight based approaches may fail to discriminate between the logical structures. Hence we need deeper level of context based analysis to

classify complex tickets. We use supervised learning approach to learn the contextual information from complex tickets. The keywords belonging to various fields of logical structure of output categories are used as features. Feature weights are assigned based on the discriminative capability of keywords.

To learn the global contextual information about all the output categories together, a large amount of training data is required. To circumvent this, we build a separate model for each category. A model for category i will have the knowledge about whether a ticket belong to category i or not (local contextual information). We used the Support Vector Machine (SVM) method with a Radial Basis Function (RBF) Kernel to build the classification engine. Complex tickets pass through all the individual models of output categories. Since each individual model knows about tickets belonging to it, globally all the tickets will be correctly classified.

Using rule/weight based approach for classifying simple tickets increases recall but can lower the precision. To maintain higher precision, one can further validate the output of rule based approach using context based analysis to filter out any misclassifications.

3 Evaluation

This section outlines the experimental analysis of the proposed approach. The methodologies have been implemented as part of a Ticket Analysis Tool from IBM called BlueFin and deployed to analyze events and tickets for several key customer accounts. We evaluate the performance of **BlueFin** in comparison with another popular ticket analysis tool **SmartDispatch** [1], a SVM-based text-classification tool, based on large datasets from some well-known real customer accounts.

For unbiased evaluation, we randomly select tickets from 7 different accounts and first manually label them into categories. For the analysis here, we consider 17 different categories of tickets for classification as shown in Table 2. Finally we choose 5000 tickets labeled with one of these 17 categories as the ground-truth data. To measure the accuracy, we computed the Precision, Recall and F1-score for each of these category. Note that in a multi-class or multinomial classification, precision of i^{th} category is the fraction of tickets classified correctly in i (true positives) out of all tickets classified as i (sum of true and false positives). Recall of category i is the fraction of tickets classified correctly as i (true positives) out of all tickets labeled as i in the ground-truth data (sum of true positives and false negatives). F1-score is the harmonic mean of precision and recall. Alternatively, these can be computed from the confusion matrix, by summing over appropriate rows/columns. Note that F1-score is a well-known measure of classification accuracy. The accuracy measures are computed for both BlueFin and SmartDispatch and the results are shown in Table 2. In addition, we also compute the overall accuracy measures for all the categories and present in Figure 3. Observe that the precision measure for each individual categories and the overall precision are extremely good for BlueFin. Moreover, it also maintains high recall values and thus the F1-score and results significantly better performance in comparison to the existing approach in SmartDispatch for all categories.

The major improvement in precision is attributed to the context based analysis in BlueFin while the higher recall is due to the enriched text set using event-ticket correlation model. To understand this in detail we look at the confusion matrix of BluFin

Table 2. Comparison of Precision, Recall and F1-Score for a labeled dataset

	BlueFin			SmartDispatch		
Category	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Database Space	0.65	0.70	0.67	0.14	0.59	0.22
Non-actionable	0.98	0.98	0.98	0.98	0.67	0.79
Job Failed	0.99	0.21	0.35	1.00	0.04	0.07
Server Down	0.97	0.92	0.94	0.83	0.26	0.39
Agent Offline	0.98	0.98	0.98	0.05	1.00	0.10
CPU Utilization High	0.97	1.00	0.98	0.84	0.96	0.89
Paging/Swap space	0.88	0.98	0.92	0.45	0.98	0.61
Zombie Processes	1.00	0.60	0.75	1.00	0.40	0.57
Network Down	0.98	0.45	0.61	0.31	0.73	0.43
Password Expired	1.00	0.83	0.90	0.99	0.79	0.87
Linux Space Full	0.94	0.98	0.95	0.16	0.07	0.10
Backup Failure	0.85	1.00	0.91	0.59	0.99	0.73
Database Inactive	0.94	0.44	0.59	NA	0.00	NA
Process Missing	0.99	0.38	0.55	NA	0.00	NA
Disk C Full	0.82	1.00	0.90	NA	0.00	NA
Win Non C Drive Full	0.89	0.84	0.86	NA	0.00	NA
Service Alert	1.00	0.59	0.74	NA	0.00	NA

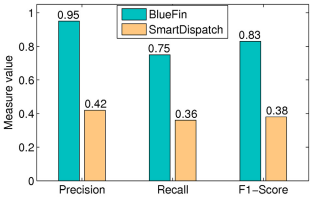

Fig. 3. Comparison of classification accuracy

(Figure 4) and SmartDispatch (Figure 5). Deeper color shades in cells represent higher volumes of tickets. The diagonal elements represent the correct classifications and the non-diagonals are the mis-classified tickets. SmartDispatch has a higher number of overall mis-classifications. For example, consider the tickets that are originally "Windows non C drive full ", but are mis-classified as "backup failed". There are 83 such mis-classifications in case SmartDispatch while BlueFin has only 11. SmartDispatch on the other hand, mis-classifies the ticket due to absence of contextual information. The performance of BlueFin far exceeds smartDispatch in all categories. The reason smartDispatch underperforms, it only uses discriminative keywords and completely ignores contextual keywords, special patterns, which provides important discriminations in case of noisy data.

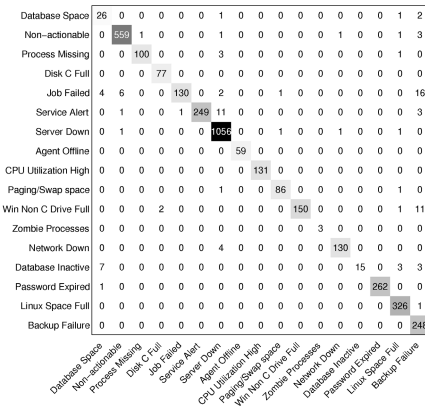

Fig. 4. Confusion matrix for Blufin

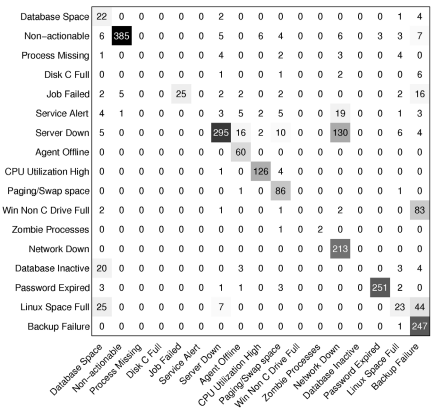

Fig. 5. Confusion matrix for SmartDispatch

4 Conclusion

In this paper, we proposed a novel approach for automatic problem determination from noisy and unstructured service delivery tickets. Central to our theme is the use of two distinct levels of analysis, namely, correlation of event and ticket data followed by context based classification of the correlated data to achieve higher precision and improved recall. Furthermore, we evaluated our approach on real customer data and the results confirm the superiority of the proposed approach. In the future, we plan to improve the precision of our approach by using bi-grams, tri-grams etc. as features and the recall by increasing the size of domain dictionaries.

References

1. Agarwal, S., Sindhgatta, R., Sengupta, B.: Smartdispatch: Enabling efficient ticket dispatch in an it service environment. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012, pp. 1393–1401. ACM, New York (2012), http://doi.acm.org/10.1145/2339530.2339744
2. Junejo, K., Karim, A.: A robust discriminative term weighting based linear discriminant method for text classification. In: Proceedings of the Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 323–332. IEEE (2008)
3. Kadar, C., Wiesmann, D., Iria, J., Husemann, D., Lucic, M.: Automatic classification of change requests for improved it service quality. In: Proceedings of the 2011 Annual SRII Global Conference, SRII 2011, pp. 430–439. IEEE Computer Society, Washington, DC (2011), http://dx.doi.org/10.1109/SRII.2011.95
4. Parvin, H., Bose, A., Van Oyen, M.P.: Priority-based routing with strict deadlines and server flexibility under uncertainty. In: Winter Simulation Conference, WSC 2009, pp. 3181–3188 (2009), http://dl.acm.org/citation.cfm?id=1995456.1995888
5. Potharaju, R., Jain, N., Nita-Rotaru, C.: Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In: Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2013), pp. 127–141. USENIX, Lombard (2013), https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/potharaju
6. Shao, Q., Chen, Y., Tao, S., Yan, E.A.X., Anerousis, N.: Easyticket: a ticket routing recommendation engine for enterprise problem resolution. Proc. VLDB Endow. 1, 1436–1439 (2008), http://dx.doi.org/10.1145/1454159.1454193
7. Shao, Q., Chen, Y., Tao, S., Yan, X., Anerousis, N.: Efficient ticket routing by resolution sequence mining. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, pp. 605–613. ACM, New York (2008), http://doi.acm.org/10.1145/1401890.1401964