Recommendations for Achieving Service Levels within Large-scale Resolution Service Networks

Kayhan Moharreri Jayashree Ramanathan Rajiv Ramnath
Department of Computer Science and Engineering
The Ohio State University
{moharrer, jayram, ramnath}@cse.ohio-state.edu

ABSTRACT

A new recommendation framework that addresses the correct and quick resolution of incidents that occur within the complex systems of an enterprise is introduced here. It uses statistical learning to mediate problem solving by large-scale Resolution Service Networks (with nodes as technical expert groups) that collectively resolve the incidents logged as tickets. Within the enterprise a key challenge is to resolve the tickets arising from operational big data (1) to the customers' satisfaction, and (2) within a time constraint. That is, meet the service level (SL) goals. The challenge in meeting SL is the lack of a global understanding of the types of needed problem solving expertise. Consequently, this often leads to ticket misrouting to experts that are inappropriate for solving the next increment of the problem. The solution here proposes a general two-level classification framework to recommend a SL-efficient sequence of expert groups that jointly can resolve an incoming ticket. The experimental validation shows 34% accuracy improvement over existing locally applied generative models. Additionally, recommended sequences are above 96% likely to meet the enterprise SL goals, which reduces the SL violation rate by 29%. Recommendations are suppressed in the case of non-routine content which is automatically flagged for special attention by humans, since here the humans outperform statistical models.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*;
H.5.3 [**Information Interfaces**]: Group and Organization Interfaces—*Computer-supported cooperative work.*

General Terms

Experimentation, Human Factors, Measurement, Reliability

Keywords

Classification, Complex Enterprise, Human-in-the-loop, Knowledge Management, Resolution Service Network, Service Levels, Text Mining, Ticket Resolution Sequence

1. INTRODUCTION

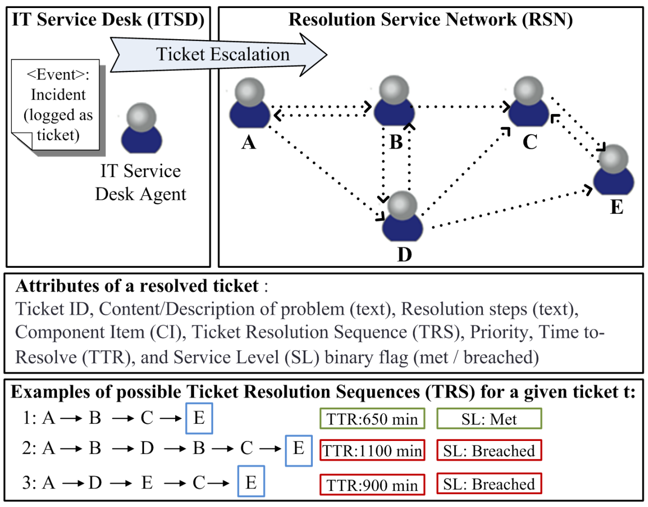
Enterprises today are beginning to realize the important role Big Data plays in achieving business goals through **operational Information Technology efficiency**. However, decision making on operational data is proving a difficult challenge. A class of operational challenges includes large-scale collaboration and human-based decision-making needed to resolve service interruptions. These interruptions are due to the dynamically changing conditions and components of the IT infrastructure that can impact the business and customer satisfaction. The collaboration and human-based decision-making introduced above is often referred to as *'triage'*. Other examples of triage include service centers, emergency rooms, and even disaster recovery.

This research focuses on a particular application — the Information Technology Service Desk (**ITSD**) within a large enterprise, which resolves data center incidents that cause service interruptions perceived by customers. The incidents are logged as *tickets*. Our goal is to develop a framework for predictive algorithms to guide decision-making that meets Service Levels (**SL**) in the real world. SL is a *time-and-satisfaction-based* metric that is defined for and contracted with different lines of business customers. The related application research challenges that are addressed here are summarized as follows:

1. From an enterprise operations perspective: Thousands of incidents generated weekly due to complex (diverse, layered, networked, evolving) hardware and software have to be resolved within real-world time constraints. It is essential to detect the cause of each one of these incidents and resolve the problem with minimum *resources and time* in order to meet the SL goals.

2. From a methodology perspective: During analysis we found that the incidents with irregular resolution paths were difficult to train on, also those incidents were causing greater SL overruns or breaches. Thus, as in current typical research, it is not enough to compare alternative learning models relative to each other and determine which one performs better, statistically. We had to come up with methods that improved SL's for incidents in the real world and **identify and address** cases of sparsity that are difficult to train on using conventional machine learning models.

To address the challenges, our research uniquely integrates **statistical and human intervention** recommendations to improve collective decision making that resolves incidents and achieves SL. Figure 1 illustrates the underlying model of a Resolution Service Network (**RSN**), which represents expert groups as nodes that problem-solve on a given in-


IT Service Desk (ITSD)
Resolution Service Network (RSN)
Ticket Escalation
<Event>: Incident (logged as ticket)
IT Service Desk Agent
A
B
C
D
E

Attributes of a resolved ticket :
Ticket ID, Content/Description of problem (text), Resolution steps (text), Component Item (CI), Ticket Resolution Sequence (TRS), Priority, Time to-Resolve (TTR), and Service Level (SL) binary flag (met / breached)

Examples of possible Ticket Resolution Sequences (TRS) for a given ticket t:

1: A → B → C → E	TTR:650 min	SL: Met
2: A → B → D → B → C → E	TTR:1100 min	SL: Breached
3: A → D → E → C → E	TTR:900 min	SL: Breached

Figure 1: Resolution Service Network (RSN) terminology.

put ticket. Directed edges represent existence of transition history between two experts. We define a Ticket Resolution Sequence (**TRS**) with respect to a specific ticket as a sequence of transitions between the expert groups representing progressive discovery that leads to resolution for the ticket. Also we define a 'distinct TRS' as a class of TRSs with identical sequences of transitions. The enterprise RSN studied has 900 expert groups, and has exhibited 7250 distinct TRSs to resolve incidents generated from over 7400 Configured Component items (CIs) in the IT infrastructure. Important ticket attributes are also shown in Figure 1. In addition, as illustrated in the same figure, some TRSs help the ticket meet its SL goals while others do not.

Our analysis of operational data revealed some facts also observed before in [7, 14]. There are two kinds of knowledge — ***'content'*** and ***'transition'*** — that are applied by the involved expert groups based on the specific ticket. That is, based on description of the problem (i.e. content), each group tries to resolve the ticket (fix the problem) with content knowledge; or use its local RSN network knowledge to transition the ticket to the most effective expert group that can resolve it. Consequently existing research [14, 7, 3] mainly uses statistical learning models such as Markov Model, Greedy Transfer Model, etc., by training on a set of pairs in the form of <content, next-transition> to infer next most likely transition. They reported experimental reduction in the Mean Steps To Resolve (**MSTR**) for a set of test tickets where 'steps' are noted as number of transitions.

In addition and in contrast with current research, our deeper data analysis also shows that a reduction in the number of transitions, which is steps in MSTR through locally applied methods, is neither necessary nor sufficient for improving SL. Often the involved expert groups of entire TRS are needed in a content-specific sequence for resolution. And, the application of each group is very dependent on the global context governing what is to be done next, based on what has been done by previous expert nodes. This means that the Markov property which makes statistical learning feasible does not generally hold in the RSN context. To illustrate with a simple case consider the following ticket content: "Application X is not able to connect to database D". The TRS for this is the following sequence: 'IP&Connectivity' $\rightarrow$ 'Database Administration' $\rightarrow$ 'IP&Connectivity'. The TRS here has global characteristics, e.g. IP&connectivity group needs to execute problem solving *twice*: the first time partially contributing to problem solving as a 'collective contributor'; and the second time as 'resolver' who also tests the solution. We define a resolver as a group which adds the last increment of contribution leading to resolution. In the real world RSNs, *not all groups act as problem resolver groups*.

Framework for recommendations on each ticket:
Routine ⟶ TRS guided transitions to expert problem solvers.
Non-routine ⟶ Human-in-the-loop both for transitions and problem solving

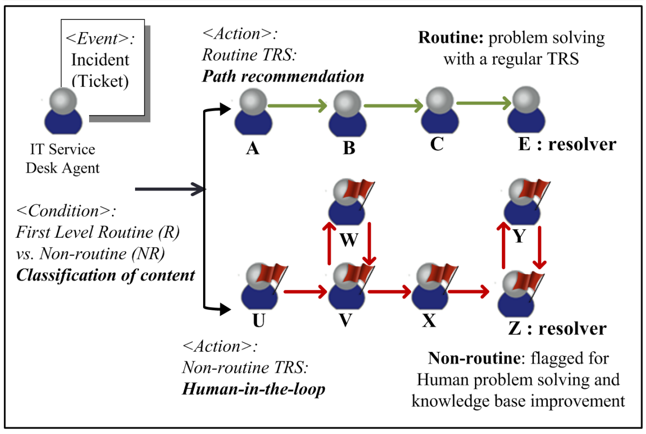

<Event>: Incident (Ticket)
IT Service Desk Agent
<Action>: Routine TRS: Path recommendation
Routine: problem solving with a regular TRS
A
B
C
E : resolver
<Condition>: First Level Routine (R) vs. Non-routine (NR) Classification of content
W
Y
U
V
X
Z : resolver
<Action>: Non-routine TRS: Human-in-the-loop
Non-routine: flagged for Human problem solving and knowledge base improvement

Figure 2: Overview of the proposed RSN recommendation framework for achieving Service Levels for Routine (R) and Non-routine (NR) tickets.

Thus simply reducing the steps to resolve as in most existing research methods does not necessarily ensure with certainty that necessary sequencing *dependencies* are preserved and the last group resolves the ticket. And, meeting the expected SL time is also not ensured by reducing the number of steps. Certain expert groups have better content problem solving knowledge and getting them involved early may indeed reduce the Time To Resolve (**TTR**) of the ticket. We also observed during our analysis that many TRSs contain wasteful transitions, which are due to groups' lack of network knowledge of the complexity with the total problem faced by the RSN. Thus, the enterprise deployment questions of real value that have to do with meeting SL in the real world are only partially addressed with previous research.

The first significant difference in our framework herein for achieving SL is that we seek to identify content as *Routine* or *Non-routine* as in Figure 2 top (Routine TRS with sequence <A,B,C,E>) and bottom (Non-routine TRS with sequence <U,V,W,V,X,Z,Y,Z>). This is used to ensure that even when machine recommendations do not perform well, the humans will conduct the problem solving. And, the other difference is that we train on entire *TRSs*, not just on *transitions* which are local. That is we train on <content, TRS> pairs if the content is routine. Hence, in contrast to local single-transition recommendations, we recommend the entire transition sequence - TRS. This strategy addresses the above-identified challenges by preserving needed dependencies using methods detailed below.

Note that we introduce the term 'collective' as a type of collaboration where a specific sequence of groups ***must each apply the specific problem solving knowledge in an order***. That is due to the collective nature of collaboration, sequences rather that single transitions are important. Additionally it is important to note that the SL is achieved collectively by the entire TRS sequence of experts working start-to-finish and not simply by local fast-working groups. In such cases where the expertise order is mandatory, TRSs reflect global network knowledge. Also how frequent that knowledge was used or how often it achieved SL can be identified.

The second unique feature of our research is that we incorporate the 'human-in-the-loop'. This approach requires the machine to *differentiate* between the **Routine (R)** problem solving where it learns and recommends effectively; from the **Non-routine (NR)** where the human experts explore and do better to achieve SL. Here machine learning determines: (1) contextual conditions under which TRS recommendations can be reliable (**Classification of content** in Figure 2). If reliable, (2) recommends the TRS that assists the RSN meet SL (**Path recommnedation** in Figure 2); Otherwise, (3) 'flags' where it cannot do well and increases the reliance on native human problem solving (**Human-in-the-loop** in Figure 2).

Here we validate this two level classification framework making it easier to apply machine learning by alleviating data sparsity. First we establish our two-part main hypothesis:

i. *There is a direct relationship between ticket content as the input for the RSN and the output of resulting problem solving as the TRS*

ii. *The R-TRS that is associated with routine content is less likely to breach service levels.*

The hypothesis is established through analysis of both content and TRS probability distributions. This hypothesis, once established, is exploited by our two-level ***Dynamic RSN Workflow Framework*** (Figure 2) with the following elements:

⟨**Event**⟩**:** Logging of ticket content

⟨**Condition**⟩**:** classification of content as R/NR learnt using TRS history

⟨**Action**⟩**:** If result of classification is 'R' then recommend a Routine TRS (R-TRS); else flag for NR human-in-the-loop transitioning and knowledge base improvement.

The framework, generally applicable to the class of triage problems, leverages key points statistically: (1) Important R-TRS characteristics already exhibited reliably in the history (e.g. resolution, SL met/breached, time to resolve, customer satisfaction) are used to recommend to achieve similar resolution in the future. (2) R-TRSs are selected to be frequent in the training set and thus form a less unbalanced multiclass classification problem.

The experiments on an IT Service Management data set with 62,000 human annotated tickets show this method performs 34% more accurately than the best existing locally applied generative model on content that is routine. That is, it predicts the global R-TRS correctly 77% of the time. Also, 96.2% of R-TRS recommendations are expected to meet their actual SL goals. Finally, content predicted as NR is flagged for both content and transition problem solving by humans.

Next, we present the enterprise context in Section 2. Following this we survey the related interdisciplinary research in Section 3. Then we further explore challenges and address this through the two-phase framework in Section 4. The following Section 5 is primarily devoted to establishing the main hypothesis through heuristics presented. This is followed by model development in Section 6. The experiments and analysis are in Section 7.

2. ENTERPRISE OPERATIONS & DATA

Event and Incident Context: Note that while sophisticated infrastructure monitoring systems are in place, the events generated within the culprit CIs (applications, servers, network components, storage systems, etc.) are at a fine-grained level and the overarching context is not known for most problems to be fixed. Here we address the challenge case where the customers call in reporting a problem, as they perceive it (*user-perceived tickets*). Achieving resolution in these cases require complex discovery-oriented collaboration by the RSN — i.e. the IT service delivery and support organization. The best practice used for this is ISO 20000 (IT Infrastructure Library) [11] with the associated organization, process knowledge and enterprise software.

Any incident communicated (e.g. via a call) is immediately logged as a ticket with mandatory description of the problem in text by the IT Service Desk (ITSD) agents. Then it is either immediately resolved at the service desk or escalated (using generic workflow routing lists in the enterprise software and tacit network knowledge) to RSN groups with the skills to provide collective problem solving. Transitions create a *path* (TRS) ending with a resolver. When the resolution is achieved several ticket attributes become known – SL met or breached, transition history(full TRS), culprit CI(s), problem resolution steps, and time to resolve of the incident.

To enable RSN problem solving there are also some documented procedures in the Incident Management Knowledge Base (IMKB). However, it is observed that about 40% of resolved tickets do not utilize knowledge base because of the following reasons: (1) the problem is new and has not happened in the past, (2) content and transition knowledge are tacit, (3) the problem is complex and cannot get diagnosed. Thus IMKB alone cannot guarantee effective resolution of incidents and consistent SL achievement.

Tickets and Service Levels (SL): The enterprise's operational data analyzed here consisted of 62,000 user-perceived incident tickets with related 111,000 transitions representing the processing by the 900 expert groups. Our analysis also found that tickets may have 1 to 18 associated transitions before resolution. The number of transitions before reaching a resolver is called *TRS length.* For each *type of ticket and service*, the priority is pre-determined in collaboration with the customer. The priority level is set between P1 (highest impact) to P4 (negligible impact) based on the severity and urgency to the customer and the type of RSN problem solving. ***The priority level determines a target time to resolve for the ticket, known as SL goal***. The SL goal is more relaxed for lower priorities. The SL clock runs per ticket, and time is shared among all the expert groups along the path. Observations resulting from analysis are shown in Figure 3. Both the plots have TRS length as x-axis which depicts the number of transitions leading to resolution. In the top plot of Figure 3, we show that tickets are more likely to be resolved in fewer number of transitions. As the TRS length increases, the probability of the tickets being resolved in exact TRS_length transitions drastically drops (shown using the log scale y-axis). Note that the probability of tickets

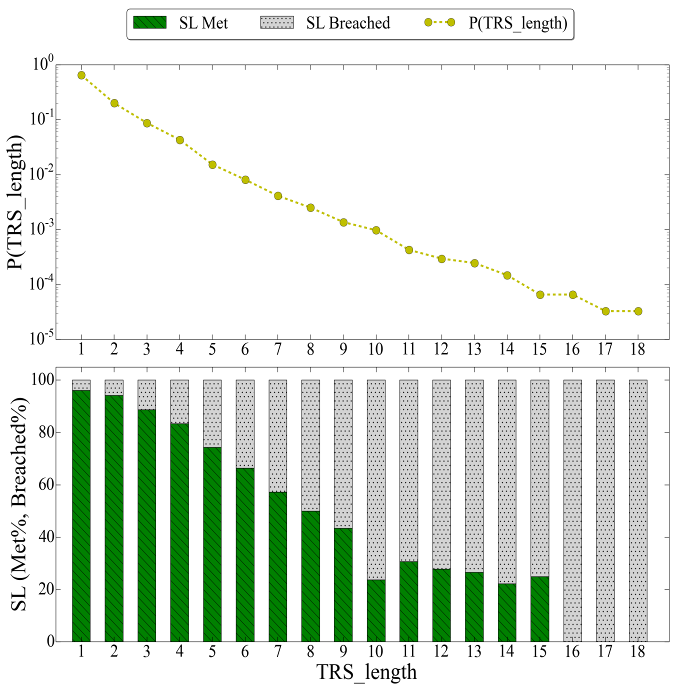


Figure 3: SL and ticket volume versus TRS length.

drops with a factor larger than two at each increment to the TRS length. This plot in linear scale shows an *exponential decay* that is, after execution of each transition there is more than 50% chance of resolving the ticket at that point. This can be represented by a *power law* and it implies that RSN groups try to resolve as many tickets as possible in fewer transitions if possible. The bottom plot in Figure 3 shows that at the same time, the tickets with greater TRS length are also more likely to breach their SL goals.

3. RELATED RESEARCH

Collaborative problem solving is leveraged today in collaboration ecosystems. Question-answer microblogs such as Stack Overflow, Quora, and WebMD have focused on taking advantage of wisdom of the 'qualified crowd' in order to answer questions in respective domains. Other systems go further to exploit content expertise and network knowledge in complex problem solving. Examples include medical systems such as TriageLogic and InXite focus on resolving complex medical cases through ***collective*** collaboration between care providers. Work as a Service (WaaS) research in [10] proposes a hub to achieve responsiveness and address unpredictability. In general, however, the design of statistical models to guide RSN groups to ***collectively achieve SL*** has not been addressed. In the fields of Computer Supported Cooperative Work and Social Networks, coordination mechanisms that address the increasing complexity of collaboration has been extensively studied [1]. More recently the related concept of affordance that is 'individual', 'collective' and 'shared' has also been introduced and discussed extensively [5]. A relevant notion here is the 'collective' network behavior when individuals collectively achieve SL goals that they cannot achieve individually. Thus, it has been pointed out, that shared affordances are essential to the performance improvement. However, again statistical methods for mediating shared affordances have not been researched. In systems engineering, transitions are shown to add inefficiencies. A framework for measurement, traceability and improvement in service-oriented environments is presented in [12]. However in highly dynamic situations, statically defined transitions soon become obsolete. In addition to statistical research mentioned earlier we use 'Resolution Service Network (RSN)' concept first introduced in [3] to analyze which tickets were incorrectly transitioned and to identify reasons for this. The RSN concept is more goal-directed than social networks studied in social science. Other approaches to enhancing the knowledge management through community aware strategies are in [5]. Moreover, the use of event logs to reconstruct the process model as executed has been studied extensively by [17, 18] under the topic of process data mining. The applications explored include process conformance and data provenance. These methods are relevant for extracting span time, queue time, repeating patterns, etc. for better TRS predictions. Recent work also studied expert behaviors in collaborative networks and found that when task transfers happen there is some overlap of knowledge between experts, also authors demonstrated existence of two types of routing patterns, task neutral, and task specific routing [16]. Finally, on-demand real time score and recommendation systems are becoming increasingly popular. These systems are most effective where critical decisions are to be made in massive-scale within limited periods of time, and otherwise can get heavily impacted by constrained and error-prone human performance. Their applications range from intelligent decision support systems [19], to automated response assessment systems [15, 9].

4. ENTERPRISE DEPLOYMENT AND RSN CHALLENGES

Next we discuss as yet unaddressed challenges regarding achieving SL. Also we present the framework overview before proceeding to the details.

4.1 SL challenges

Context and RSN performance: Since SL time is globally achieved by the TRS, it is not enough to recommend locally 'good' or 'most likely' next transition as in existing generative models [7]. Only if a ticket is transitioned to the expert group with the 'correct' content and 'correct' transition the correct future transition occur. So the important goal here is to recommend global TRSs that will avoid 'incorrect' local transitions and thus improve the MTTR (Mean Time To Resolve).

MTTR versus MSTR: The 'time and SL optimal' TRS may or may not meet the 'minimum number of transitions' criteria. For example, a shorter sequence based on higher local probability can end up utilizing groups that take more time (e.g. with novices), thus breaching the SL. While MSTR has been studied for RSNs, MTTR and SL have not. Here we ask: *Can we recommend TRSs that largely have met SL previously?*

Limitations of pure inference models: Existing inference models [14, 7, 3, 8] focus on showing improvement but make assumptions that the real world is static. They do not incorporate a way to implement continuous improvement strategies in an actual enterprise. Here the questions for enterprise deployment we address are: *How can we identify where the RSN is performing well, and therefore can we*

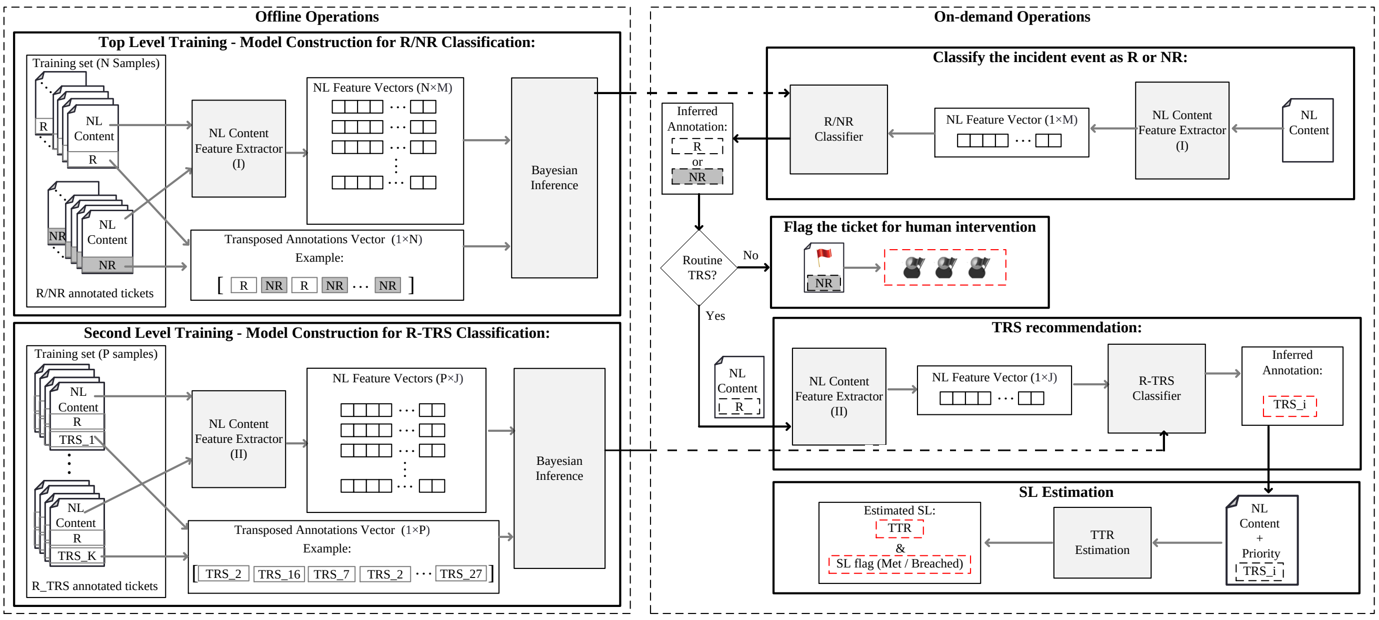


Figure 4: Dynamic RSN recommendation framework

learn from TRS history in those cases? Or, when performing poorly where and how do we need to engage humans in addressing improvement on a continuous basis?

4.2 Overview of Dynamic RSN Workflow

The two-level framework introduced earlier is illustrated with further details in Figure 4. The model developed is divided to offline training (left), and on-demand recommendations (right). Offline training includes computationally intensive operations and they lead to construction of the classification models. Formalized details of training are given in Section 6. On-demand recommendations apply the classifiers on the unlabeled data and recommend actions for achieving SL goal. Formalization details of recommendations and their evaluation are given in Section 7.

Offline Operations – Top Level Training: The goal here is to build a Bayesian binary classifier that takes Natural Language (NL) content, and identifies whether it is associated with highly frequent distinct TRSs (marked as Routine). By establishing the main hypothesis introduced above that highly likely content is strongly associated with frequent distict TRSs, we can demarcate the Routine from the Non-routine. As shown in Figure 4, NL features are extracted from training tickets and are then used along with their R/NR annotations to perform Bayesian inference. Then the top-level R/NR classifier is constructed for on-demand use. The intuition underlying the upfront R/NR classification is that multiclass classifiers work best on data sets with frequent patterns that have causal relationship with the target variable, in this case ***routine content with routine TRS***.

Offline Operations – Second Level Training: The goal is to build a Bayesian multiclass classifier that takes NL content and identifies a Routine TRS that is most likely to resolve the problem. Notation 'R-TRS' is used to refer to a distinct TRS that is pre-labeled as Routine. Labeling strategy of TRSs are discussed in Section 5. Same as above, NL features are extracted from training tickets and are then used along with their R-TRS annotations to perform Bayesian inference. Then the second-level R-TRS classifier is constructed for on-demand use. We also discuss and address the underlying challenges of dealing with skewed class distribution in Section 6.

On-demand Operations: Here we show the two-level application of the method on an unlabeled ticket. First we determine if the NL content of the ticket is associated with either R or NR using the top level R/NR classifier. Secondly, if it is associated with R then the second level R-TRS classifier is applied to provide the TRS recommendation for RSN execution actions. Also SL estimation is performed for the recommended TRS. If the content is associated with NR class then it is flagged and turned over to the RSN for resolving the ticket manually and also for feedback enhancing the knowledge base for future performance improvement. In Figure 4 within the on-demand operations box, all of the dotted boxes are denoting predicted values. In Section 7 we discuss the validation and SL advantages of the framework.

5. ROUTINE VERSUS NON-ROUTINE

We first establish the main hypothesis that the more likely inputs of the RSN are associated with the more likely outputs. To do this we first develop functions that *independently* quantify the regularity of the content (input) and of TRS (output). Note that if content can signal for strong association with frequent TRSs then predicting only among frequent TRSs leads to a more accurate classification outcome as opposed to predicting among all TRSs in the first place. This solution is particularly favored where a small portion of distinct TRSs are used to resolve majority of the tickets in the history. This is discussed further next.

5.1 High Likely Content is associated with R-TRS

Here we first define routineness measures for content and distinct TRSs separately. Then we demarcate R-TRSs from NR-TRSs through the following ***labeling strategy***: we at-

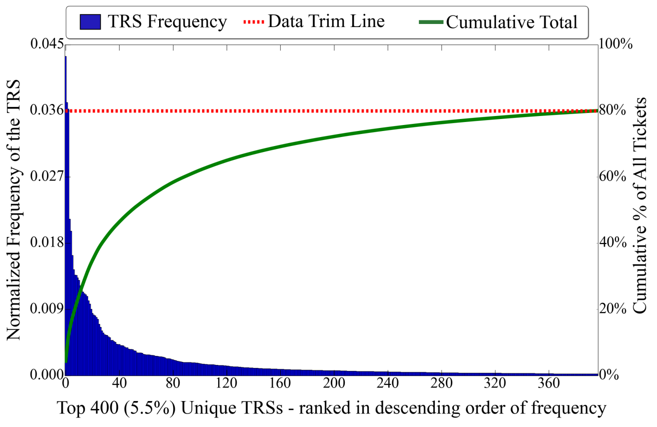


Figure 5: Normalized frequency of distinct TRSs – Pareto Chart

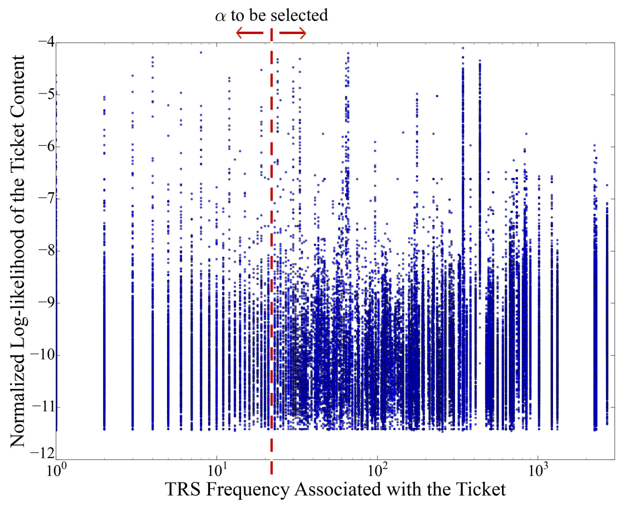


Figure 6: Projected tickets in CLL-TRS_frequency space – High density of tickets in the center right area means tickets with more frequent TRSs are very likely to have regular occurring content.

tempt to find a subset of distinct TRSs that their content likelihood distribution is in maximum distance from that of their complement set. Then members of the set with higher average content likelihood are labeled as R-TRS, and members of its complement are labeled as NR-TRS. To avoid trivial solutions, the aforementioned objective is subject to a constraint that enforces minimum ticket coverage on both subsets.

Routineness of TRS (discrete metric):We characterized regularity of a distinct TRS as the frequency of tickets resolved by that distinct TRS in the training set. The more frequent a distinct TRS is used, the larger its routineness value becomes. Our data illustrates the skewed distribution of tickets over distinct TRSs (Figure 5). This graph is a *Pareto chart* that represents normalized frequency distribution of distinct TRSs in descending order along with cumulative percentage of tickets. In the figure the top 5.5% of the most frequent distinct TRSs (400 distinct resolution sequences out of 7,250) are depicted which are used as resolving paths for 81% of tickets. The other 19% of tickets use 94.5% of remaining less frequent distict TRSs. This follows the well-known *Pareto principle* that implies most of the ticket probability mass is accumulated on a small portion of TRSs. The Pareto principle is an inherent property of RSNs. Also it is acknowledged in the Machine Learning community that more number of classes in a multiclass setup inevitably leads to higher misclassification rate [4]. Here the Pareto principle is leveraged to build a multiclass classifier that ***only trains on a small portion of distinct TRSs denoted as R-TRSs, and can precisely recommend on a large portion of tickets.***

Routineness of content (continuous metric): To characterize the input content of the RSN we treat it as meaningful word sequences with a log-likelihood metric that measures the probability of the word sequence in the content of a ticket as *Content Log-Likelihood*(**CLL**):

$$CLL(t;\lambda) = \frac{1}{|t|}\sum_{w_i \in t} \log \hat{P}(w_i \mid w_{i-1};\lambda) \tag{1}$$

where:

$$\hat{P}(w_i = b \mid w_{i-1} = a;\lambda) = \frac{\#(ab)+\lambda}{\#(a*)+\lambda\,|V|} \tag{2}$$

Here w_i is the ith word token in a ticket t, and λ is a smoothing parameter. Normalization (i.e. division by $|t|$) is needed to establish a fair measurement for significance of words regardless of the number of word tokens in a ticket. Next, the probability of a word w_i in the context of w_{i-1} is computed. We use a *bigram* language model [2] where '#' represents a function that computes the frequency of the given word phrase in the ticket corpus, and $*$ represents any word in the corpus dictionary. $|V|$ is the size of the corpus dictionary. In our data set, the CLL values of different tickets range from -4 to -14, with -4 signifying the most likely content.

Figure 6 projects each ticket into a two dimensional space with the log likelihood of content (y-axis) and the TRS frequency (x-axis which is also on a log scale to accommodate the sparse tail of TRS distribution). As can be seen, there are considerable number of tickets that are having a highly likely content and are resolved by a highly frequent distinct TRS. Therefore, we can now look for an α-split on the x-axis that maximizes the distance between (1) CLL distribution of the tickets that have a TRS frequency less than α (to be labeled as NR) and (2) CLL distribution of the tickets that have a TRS frequency more than α (to be labeled as R). For simplicity, we define distance between two CLL distributions as the difference between the mode values of the two distributions.

Figure 7 illustrates the strategy to identify the optimal α by sliding the α–cut from low-to-high TRS frequencies on Figure 6. In Figure 7 we plot the mode difference of CLL distributions generated by different α–cuts. As we move α from left to right, generally the mode differences increase.

Viewed simply, the mode difference is maximized when α reaches maximum TRS frequency which is a trivial solution. However even though the largest α maximizes the mode difference between R and NR, the volume of R tickets is minimized at that point which is undesirable. This constitutes a trade-off between the ratio of *R_tickets/all_tickets* and the mode difference of the two CLL distributions. We selected the value of $\alpha = 670$ to address this trade-off, and with this the R tickets are 30% of the full data set, where

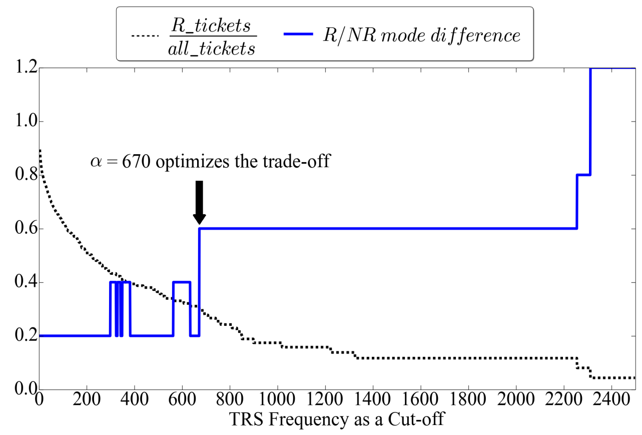


Figure 7: $\alpha = 670$ allows us to separate R from NR effectively.

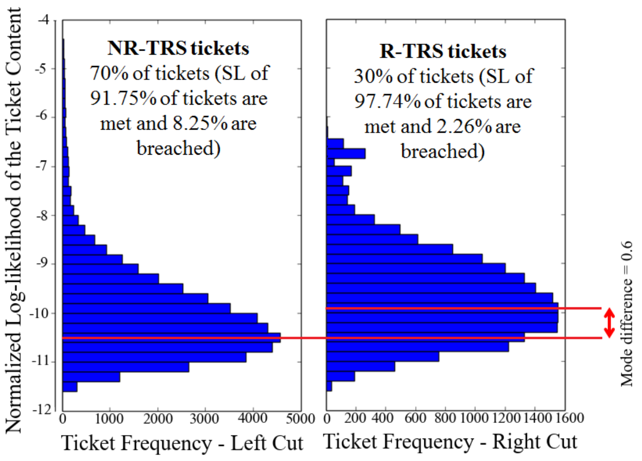


Figure 8: $\alpha = 670$ yielding optimal cut, two CLL distributions as NR-TRS (left) and R-TRS (right)

this α has the mode difference of 0.6 between R and NR.

Establishing the main hypothesis: In Figure 8, for this $\alpha = 670$, we depict the corresponding left and right CLL distributions that are now labeled as NR-TRS and R-TRS respectively. Note that R-TRS distribution is denser in the higher log-likelihood area as compared to NR-TRS.

Part (i): This establishes the hypothesis that a considerable number of tickets with frequent content (higher log-likelihood) are resolved by frequent TRSs. This also identifies the desired α supporting causality between frequent content and R-TRS and allows us to proceed with our two-level classification. Essentially, what we did in this step has introduced heuristics to label a subset of distinct TRSs as R-TRSs and the complement set as NR-TRSs. Later all of the distinct TRSs will be used to train the top-level R/NR classifier, and R-TRSs will be used to train the second level muliclass classifier.

Part (ii): Furthermore, we tie this optimal R/NR split with SL by calculating the SL breach percentage. As shown in 8, the breach ratio of the R-TRS class was found to be a fourth of the NR-TRS class (2.26% to 8.25%). This supports the hypothesis that R tickets are more likely to meet their SL. Also, if R-TRSs are recommended then the recommendations are 97.74% likely to meet enterprise SL goals.

6. EXPERIMENTS USING THE TWO-LEVEL CLASSIFICATION FRAMEWORK

Having proven the main hypothesis and the existence of a strong relationship between frequent content and routine TRS, we proceeded to build the classifiers. The learning algorithm we leveraged is the Transformed Weight-normalized Complement Naïve Bayes (TWCNB) [13] for both top and second level classifiers in the Framework introduced in Figure 4. This algorithm is designed to perform well on skewed training data, and it incorporates effective weight normalization and feature transformations. Further rationale for selecting this method is provided below.

Dampening the effect of skewed data bias: The top level training is a binary R/NR classifier with sufficient number of training instances for both classes. In the second-level training, the distribution of tickets over R-TRS classes is extremely skewed. In other words, there are many more training examples for certain TRSs as compared to others. This causes the classifier to exhibit an inevitable preference over certain classes. In [13] a solution is proposed for document classification by learning the word weights for a class, by using all training data not in that class (training on the complement). By doing this the model uses distinguishing features and is less likely to overfit to content that is labeled with less frequent R-TRS classes. Thus, we leverage the TWCNB classifier, and instead of maximizing $P(NL\_content|class = c)$, minimizes $P(NL\_content|class = \bar{c})$ where $\bar{c}$ denotes all the training data points except those with class label c.

Training (top and second level) and classification (R/NR and TRS recommendation): Here the details of our classification algorithm are presented. Our explanation is on R-TRS training and classification since they represent a multiclass classification, but the same paradigm is applied to the binary case of R/NR. We modified TWCNB for R-TRS classification as follows. Let:

1. $\vec{t}$ be the training set of routine tickets that previously got resolved by going through an R-TRS: $\vec{t} = (\vec{t_1}, \vec{t_2}, ..., \vec{t_n})$ and t_{ij} is the frequency of the j_th word of the dictionary in ticket $\vec{t_i}$.
2. $\vec{RT} = (\vec{rt_1}, \vec{rt_2}, ..., \vec{rt_n})$ be the labels of TRS routes corresponding to each of the training tickets.
3. $C = \{C_1, C_2, ..., C_s\}$ be the set of distinct TRSs.
4. $\vec{test} = (f_1, f_2, ..., f_m)$ be a test ticket where f_j is the frequency of the j_th word of the dictionary in the test ticket.

Then *train* and *predict*:

$$\omega = R\text{-}TRS\_Training(\vec{t}, \vec{RT}) \tag{3}$$

$$Predicted\_label(\vec{test}) = \arg\min_{c \in C} \sum_{j=1}^{m} f_j \cdot \omega(j \mid \bar{c}) \tag{4}$$

For the function call $R\text{-}TRS\_Training(\vec{t}, \vec{RT})$ we use Algorithm 1 that performs training. It uses a set of transforms for term frequencies adapted from [13]. These transforms resolve different poor modeling assumptions of Naïve Bayes classifier including skewed word and class distribution. ω is the transformed weighted normalization function

Algorithm 1 R-TRS_Training ($\vec{r}, \vec{RT}$)

```
1: for $j = 1$ to $m$ do
2:     $IDF_j = \log \frac{n}{\sum_{k=1}^{n} \delta_{kj}}$
3:     for $i = 1$ to $n$ do
4:         $TF_{ij} = \log(t_{ij} + 1)$
5: for $j = 1$ to $m$ do
6:     for $i = 1$ to $n$ do
7:         $NC_{ij} = \frac{TF_{ij} \cdot IDF_j}{\sqrt{\sum_{k=1}^{m} (TF_{ik} \cdot IDF_k)^2}}$
8: for $j = 1$ to $m$ do
9:     for $h = 1$ to $s$ do
10:        $\hat{P}(j \mid \overline{C_h}) = \frac{\lambda + \sum_{k:rt_k \neq c_h}^{n} NC_{kj}}{m\lambda + \sum_{k:rt_k \neq c_h}^{n} \sum_{p=1}^{m} NC_{kp}}$
11:        $\omega(j \mid \overline{C_h}) = \frac{\log \hat{P}(j \mid \overline{C_h})}{\sum_{k=1}^{m} \log \hat{P}(k \mid \overline{C_h})}$
12: Return $\omega$
```

over $P(j \mid \bar{c})$ where j can be the index of any word in the corpus dictionary, and $\bar{c}$ can be complement of any class in the data set (distinct TRSs in this case).

Here necessary details are provided to explain Algorithm 1: Line 2 constructs inverse document frequency transformation where $\delta_{kj} = 1$ if the j_th word of the dictionary is in ticket $\vec{t_k}$, otherwise $\delta_{kj} = 0$. In line 3, n is the number of tickets in the training set. Line 4 constructs term frequency transformation. Line 7, provides the length norm, where m is the size of the corpus dictionary. In line 9, s is the cardinality of the set C. Line 10, builds a smoothed probability function that estimates the probability of j_th word of the dictionary not being in class C_h. Line 11 provides a log weight normalization of $\hat{P}(j \mid \overline{C_h})$.

Experimental process overview: The two-level classification in Figure 4 was applied as follows. For both classifiers we extracted features from NL content of the tickets. For doing so the text was first transformed to vectors with weighted normalized values as discussed in 'dampening the effect of skewed data bias' earlier in section 6. We also dropped the stop words, and removed the words that were less frequent than our minimum cut-off. After applying these constraints the dimensions of our feature vectors reduced to 4623. Next we randomly sampled 80% of <content, TRS> tuples (i.e. 49763 tickets) for end-to-end model training and 20%(i.e. 12441 tickets) for validation. That 80% is fully used to train the top level R/NR classifier, and the routine portion of it (i.e. 14929 tickets or 24% of all tickets) is used to train the second level R-TRS classifier. The training on each level was validated by 10-fold cross validation (rotation on 90%, 10% splits). After tuning parameters of each of the classifiers separately, we observed significant performance in both classifiers in isolation. Then we measured the overall performance of the sequentially combined classifiers by using the 20% validation set. Details follow.

6.1 Performance evaluation with respect to SL

To summarize, the goal of the designed framework is to recommend a TRS *only if* (1) it is well-associated with the ticket content and (2) that TRS is likely to achieve the SL. In addition, because of the early R/NR classification it is still ensured that human-in-the-loop ticket transitioning is enabled for those tickets with no TRS recommendation. Therefore, this framework can be viewed as an enhancement over pure human ticket-transitioning process, which mainly targets to assist a subset of tickets. Although growing the size of the impacted subset is desired, a higher preference is given to more reliable recommendations as they take place. Let us assume that tickets have the following ground truth labels, *actual-R* and *actual-NR*. Human experts can handle (1) all actual-NR tickets, and (2) actual-R tickets that got misclassified as NR. However, it is unfavorable if an acual-NR ticket is misclassified as R, and is further recommended with an R-TRS. Therefore, in this application domain ***the precision of the top-level classifier and the accuracy of the second-level classifier are more important for the overall performance than the coverage***. In particular from SL achievement perspective, it is notable that the recall of the top level classifier is not as important as its precision since false negatives (misclassified routine tickets) still have a second chance to get routed through the RSN by the knowledge of human experts.

The performance of our two-level recommendation framework is evaluated by measuring the proportion of tickets that their TRS got correctly proposed, to all tickets that got proposed as R. This metric is formally defined as:

$$Overall\ R - Precision = \frac{\#(TRS\ correctly\ classified)}{\#(tickets\ predicted\ as\ R)} \tag{5}$$

R/NR Classification – tuning the Precision/Recall trade-off: Increasing the precision of routine class in our R/NR classifier can significantly improve the SL performance overall. Therefore we would like to sacrifice recall (coverage of actual-R tickets by the top-level classifier) while gaining more precision. This is achievable by adjusting the confidence threshold of the classifier. We make our top-level classifier propose R only if it is highly confident, and otherwise it should flag the ticket for human intervention. We define the confidence of the top-level classifier as:

$$Confidence = 1 - \mathbb{H}\Big(\hat{P}(C = \text{`}R\text{'} \mid \tau)\Big) \tag{6}$$

Here the same as equation 4 we have:

$$\hat{P}(C = \text{`}R\text{'} \mid \tau) = 1 - \sum_{j=1}^{m} \tau_j \cdot \omega(j \mid C = \text{`}NR\text{'}) \tag{7}$$

Equation 7 shows $\hat{P}$ as the estimated probability of class 'R' given ticket τ. Also τ_j is the frequency of j_th word of the dictionary in ticket τ. In equation 6, $\mathbb{H}$ denotes the *binary entropy function* as defined in [6] which takes a probability value. $Confidence$ returns a value between 0 and 1. The higher the confidence, the lower the uncertainty in classification. Now for all the tickets that are predicted as R, if the confidence is higher than a cut-off θ then we consider them as R, otherwise we consider them as UR (uncertain R) and flag them to be handled as NR.

Figure 9 illustrates that as we increase θ, we gain more precision at the expense of losing coverage of R class (recall). Precision scales linearly with θ (confidence cut-off), while recall declines polynomially. Figure 9 also depicts how increase in precision equates reduction in coverage of the full data set that can be automated. This is the trade-off between size of

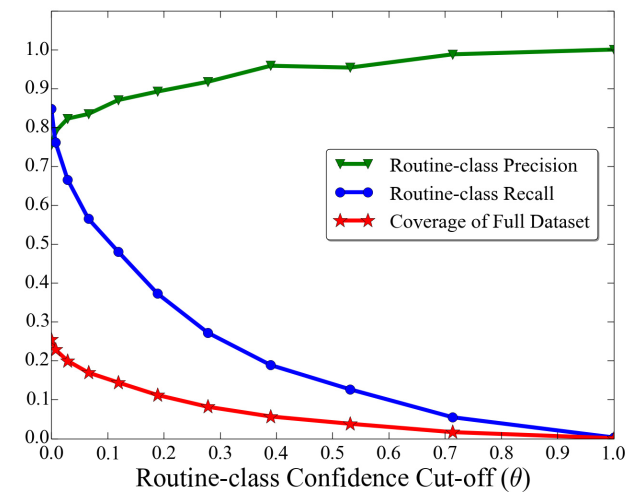


Figure 9: Precision/Recall trade-off for R/NR classifier.

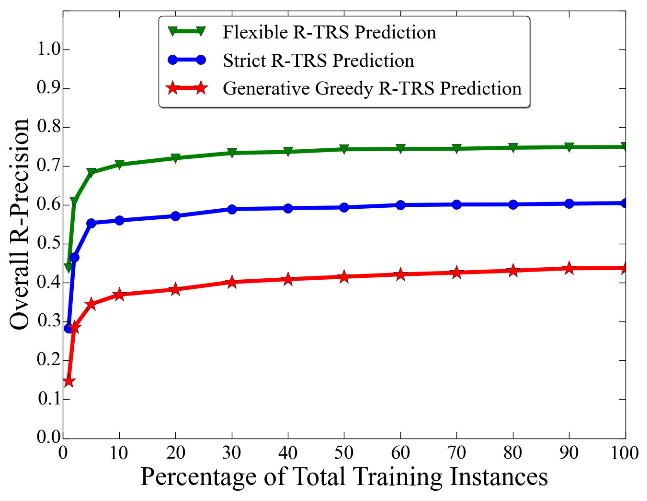


Figure 10: Overall R-Precision of flexible, strict, and greedy models.

the predictable subset of tickets versus accuracy of the TRS predictions. We picked the confidence cut-off as $\theta = 0.02$ which yields 82.2% precision, 66.5% recall and covers 20.2% of the entire tickets. Next we use this θ for the validation of our R-TRS recommendation.

7. EXPERIMENTAL VALIDATION

Next we compare three different R TRS recommendation models applied to the enterprise data, namely: (1) *Strict*, (2) *Flexible*, and (3) *Generative greedy models.*

Flexible and Strict models: For the validation of TRS recommendations we define two different ways of claiming successful classification on a test ticket: (1) *strict* TRS matching: a ticket is called correctly classified if its predicted R-TRS matches exactly with its actual TRS. (2) *flexible* TRS matching: a ticket is called correctly classified if its predicted R-TRS is within the *congruence set* of the actual TRS.

The congruence set of a certain TRS like T, consists other distinct TRSs that are equally eligible to resolve same tickets that historically got resolved by T. Here our subject matter experts established the congruence sets. Such compatibility between TRSs exist in the real world in order to balance the workload among similar groups.

Baseline model —Generative Greedy: The Generative Greedy is considered a robust transition prediction model [7]. This model is designed to make one-step transition predictions and select the most probable resolver next. In our experiment we found that Generative Greedy has shown effectiveness in predicting the resolution sequence for actual-NR tickets with long actual sequence length. To be able to compare our results, we re-defined the 'Overall R-Precision' for Generative Greedy. For any test ticket predicted as R, we let the Generative Greedy also predict n transitions at once where n is the length of the actual TRS. If the Generative Greedy matches the actual TRS, we consider it as correctly classified. The ratio of correctly classified TRSs divided by total number of prediction attempts is considered Overall R-Precision for this method.

Figure 10 shows the overall R-precision of the developed models as the size of the training set grows. All three models converge to a stable precision before reaching to 60% of the size the training set. Many of the misclassification at the strict model are between congruent TRSs. Therefore as can be seen we achieved 17% improvement over strict model by allowing misclassification within congruence sets. Also the flexible model outperforms the baseline by 34%. (flexible:77%, strict: 60%, generative: 43%).

SL and TTR (Time to Resolve) for classified tickets: The SL goal is a time duration set for each priority level and applies to all tickets at that priority. Our estimation model for time to resolve of the tickets after having their R-TRS predicted is illustrated in Figure 11. It shows the four priority zones (bounding squares) with the actual and expected times. The smallest square represents the valid duration for solving P1 tickets, and in the same order the largest square represents the valid duration for solving P4 tickets. The actual TTR is associated with resolved tickets. The Expected TTR is defined for each classified ticket based on its predicted R-TRS which is determined by the Mean value of TTRs associated with the predicted R-TRS. 81.4% of the points in Figure 11 have smaller Expected Time to Resolve than their Actual. This implies that the framework substantially improves TTR for routine tickets. Also, 96.2% of tickets with R-TRS recommendations meet their actual SL goal while this is 90.8% for tickets predicted as NR.

8. CONCLUSIONS

We have introduced a framework that can be deployed to improve collective problem solving within Resolution Service Networks. If a routine resolution path has historically achieved the SL by resolving the tickets on time then it has met the time and customer satisfaction goals. Using this we developed our framework adaptable to enterprise deployment. The first level classifier detects whether the content is associated with a routine path. The second level classifier then recommends a path among the routine ones. If the ticket is not associated with any routine path, it is flagged for increasing human attention and updating the knowledge base.

The path recommendation results are promising as they indicate 77% R-precision for the end-to-end model. Also

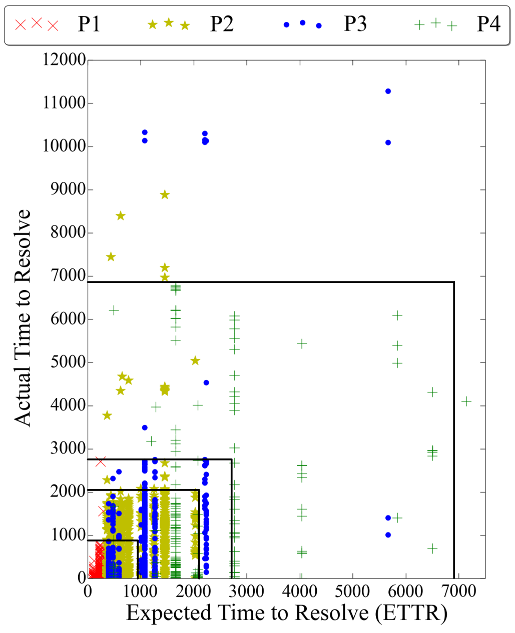


Figure 11: Expected time to resolve vs Actual time to resolve for tickets with a predicted R-TRS

the recommended R-TRSs are above 96% likely to meet the SL goals. The overall two-level classification model has also shown 29% reduction in average SL violation rate mainly by preventing routine content from getting misrouted in the RSN by the experts. More research needs to be conducted in improving the non-routine cases by engaging the human-in-the-loop in production environments. The research here also lays the ground work for understanding how machines can better help RSN problem solving by statistical learning in the routine context; and alerting the humans in the RSN in the case of non-routine.

9. ACKNOWLEDGMENTS

This research is supported by the National Science Foundation under Grant No. 0753710, the CERCS I/UCRC and industry sponsors.

10. REFERENCES

[1] F. Cabitza and C. Simone. Computational coordination mechanisms: A tale of a struggle for flexibility. *Computer Supported Cooperative Work (CSCW)*, 22(4-6):475–529, 2013.

[2] E. Charniak. *Statistical language learning.* MIT press, 1996.

[3] Y. Chen, S. Tao, X. Yan, N. Anerousis, and Q. Shao. Assessing expertise awareness in resolution networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 128–135. IEEE, 2010.

[4] O. Dekel and O. Shamir. Multiclass-multilabel classification with more classes than examples. In *International Conference on Artificial Intelligence and Statistics*, pages 137–144, 2010.

[5] V. Kaptelinin and B. Nardi. Affordances in hci: toward a mediated action perspective. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 967–976. ACM, 2012.

[6] D. J. MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[7] G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, and N. Anerousis. Generative models for ticket resolution in expert networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 733–742. ACM, 2010.

[8] G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, and N. Anerousis. Reliable ticket routing in expert networks. In *Reliable Knowledge Discovery*, pages 127–147. Springer, 2012.

[9] K. Moharreri, M. Ha, and R. H. Nehm. Evograder: an online formative assessment tool for automatically evaluating written evolutionary explanations. *Evolution: Education and Outreach*, 7(1):1–14, 2014.

[10] D. Oppenheim, S. Bagheri, K. Ratakonda, and Y.-M. Che. Agility of enterprise operations across distributed organizations: A model of cross enterprise collaboration. In *SRII Global Conference (SRII), 2011 Annual*, pages 154–162. IEEE, 2011.

[11] B. Orand and J. Villareal. Foundations of it service management with itil 2011: Itil foundation course in a book. *c. August*, 2011.

[12] J. Ramanathan, R. Ramnath, and S. Ramakrishnan. Achieving'handoff'traceability for complex systemimprovement. In *Proceedings of the fifth annual IEEE international conference on Automation science and engineering*, pages 641–646. IEEE Press, 2009.

[13] J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

[14] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis. Efficient ticket routing by resolution sequence mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 605–613. ACM, 2008.

[15] S. Srikant and V. Aggarwal. A system to grade computer programming skills using machine learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1887–1896. ACM, 2014.

[16] H. Sun, M. Srivatsa, S. Tan, Y. Li, L. M. Kaplan, S. Tao, and X. Yan. Analyzing expert behaviors in collaborative networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1486–1495. ACM, 2014.

[17] W. Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes.* Springer Science & Business Media, 2011.

[18] W. M. Van der Aalst. Using process mining to bridge the gap between bi and bpm. *IEEE Computer*, 44(12):77–80, 2011.

[19] L. Yu, S. Wang, and K. K. Lai. An intelligent agent-based fuzzy group decision making model for financial multicriteria decision support: The case of credit scoring. *European Journal of Operational Research*, 195(3):942–959, 2009.