2-2015

Will this be quick? A case study of bug resolution times across industrial projects

Subhajit DATTA
Singapore Management University, subhajitd@smu.edu.sg

Prasanth LADE

Will this be Quick?

A Case Study of Bug Resolution Times across Industrial Projects

Subhajit Datta*
Singapore University of Technology and Design
Singapore
subhajit.datta@acm.org

Prasanth Lade
Arizona State University
USA
prasanthl@asu.edu

ABSTRACT

Resolution of problem tickets is a source of significant revenue in the worldwide software services industry. Due to the high volume of problem tickets in any large scale customer engagement, automated techniques are necessary to segregate related incoming tickets into groups. Existing techniques focus on this classification problem. In this paper, we present a case study[1] built around the position that predicting the category of resolution times *within* a class of tickets and also the actual resolution times, is strongly beneficial to ticket resolution. We present an approach based on topic analysis to predict the category of resolution times of incoming tickets and validate it on a data-set of 49,000+ problem tickets across 14 classes from four real-life projects. To establish the effectiveness of our approach, we compare topic features with traditional features for both classification and regression problems. Our results indicate the promise of topic analysis based approaches for large scale problem ticket management.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Life cycle*

General Terms

Experimentation

Keywords

problem tickets, bugs, resolution times, service delivery, topic analysis

1. INTRODUCTION

In the global software services industry, millions of *problem tickets* are generated every day. Broadly speaking, a "problem ticket", or simply a "ticket" is a record of an operational difficulty faced by user(s) of an information technology system, that requires intervention from the development or maintenance team for its resolution. A ticket is usually defined by a unique identifier and a brief textual description of the problem, in addition to other optional information.

Ticket resolution drives a major part of post release maintenance and is a source of significant revenue for software development organizations. Thus timely and effective closure of tickets is critical to profitability in the software services industry. Minimizing *resolution time* – the duration between the opening and closure of a ticket – is a key parameter linked to service level agreements (SLA) between a delivery organization and its customers. To get a sense of the factors influencing ticket resolution time, let us briefly outline the typical ticket life-cycle. A ticket is opened when the end user of a software system reports a particular concern by either calling in to a help desk or filling out an online form. For large accounts handling thousands of tickets every day, the support team in charge of ticket resolution is segregated into many sub-teams based on experience and expertise. To enable fastest resolution of a ticket, it is imperative that the ticket gets quickly routed to the sub-team best positioned to resolve it. Thus an important step in the ticket resolution work-flow is an initial classification of tickets into broad functional buckets such as "access control", "performance issues", "printing problems" etc. Given the sheer volume of tickets generated in large service engagements, automated techniques are very useful for this initial classification. As we outline in the Related Work section, there is a large body of existing work on techniques around such classification of tickets into broadly defined buckets.

Automatically classifying incoming tickets into buckets helps the support team assign specific personnel to specific groups of tickets. As mentioned earlier, minimizing resolution time is often the most critical parameter the support team is measured against. Thus, the most skilled personnel needs to be assigned to tickets which are likely to take the longest time to resolve. Once incoming tickets have been classified into broad buckets, how does the support team know which tickets *within* a bucket will take the most time to resolve? To address this question, we need to study the distribution of resolution times of tickets within pre-classified buckets.

In this paper, we study a data-set of 49,811 tickets across 14 classes or buckets (marked $C1$ to $C14$) from four real

*Corresponding author

[1]The study was conducted when the first author was working as a researcher and the second author was an intern at IBM Research, Bangalore.

ISEC '15 Bengaluru, Karnataka, India

http://dx.doi.org/10.1145/2723742.2723744.

Table 1: Resolution times (mins) of tickets in different buckets: Descriptive Statistics

	N	Mean	Median	Std Dev	Skewness	Kurtosis
$C1$	2938	443.94	183.25	722.78	3.76	19.21
$C2$	1063	1671.55	1365.87	1478.77	2.19	7.33
$C3$	5161	147.74	40.98	340.61	6.11	57.15
$C4$	1253	6301.29	1576	18780.57	8.64	104.76
$C5$	1269	12975.29	7260	16220.45	2.84	9.97
$C6$	1642	1461.42	429	3264.16	6.37	61.66
$C7$	3716	365.43	143.85	619.62	3.92	22.31
$C8$	2683	313.93	74.30	600.47	3.92	22.67
$C9$	1032	82.85	63.00	85.81	3.01	14.17
$C10$	2096	630.71	321.53	809.74	2.51	8.77
$C11$	3374	17245.58	12596	15518.73	1.72	3.55
$C12$	1097	15977.46	9775.00	17967.37	2.10	5.30
$C13$	5792	1799.34	151.00	5938.59	7.60	78.60
$C14$	16695	444.02	211.17	645.49	3.43	17.40

world software service delivery projects. Each bucket represents tickets from a particular functional area of a specific project. Out of the four projects considered two projects had three functional areas two others had three functional areas, in total we considered 14 functional areas. In the remainder of the paper, we will use "bucket" to denote these 14 higher level classes. In Table 1 we present the descriptive statistics of the resolution times of tickets in each bucket. As we notice from the skewness values, all the distributions are positively skewed, that is, they have a relatively long right tail. To illustrate this point further, Figure 1 shows the histograms of the two buckets with lowest ($C9$) and highest number ($C14$)of tickets. From the shape of the distributions of all the buckets, it is evident that *there are many tickets which get resolved quickly while few take very long time.* This is a very critical observation. For minimizing resolution times, the support team has to assign its most effective resources to the tickets that are likely to take the longest time to resolve. So, predicting which incoming ticket belongs to which section of the resolution time distribution has significant practical benefits. To address this problem, we present *TAACT – A Topic Analysis based Approach to Categorization of Tickets.* TAACT is validated on our aforementioned data-set and compared with other widely used approaches.

In the next section we motivate our approach, followed by outlines of our research contributions and related work. Subsequently we outline the TAACT, and discuss the results from its application. The paper ends with a discussion of the threats to validity and conclusions.

2. MOTIVATION

Ideally, enough information should be extracted from users at the time of opening a ticket, such that the ticket can be easily classified into a bucket with similar other tickets. But this is seldom, if ever, the case in large projects. Usually a user raising a ticket can only give a textual description of the problem and the circumstances in which it occurs, with few specifics. Details which will lead to a ready identification of the problem's background may be deliberately hidden from the user, such as a detailed stack trace. Even otherwise, in most cases end users will lack the technical sophistication to understand such details even if it was available to them, such as identifying the most relevant part of a stack trace. Thus most often the essential components of a ticket comes down to a unique identifier – which is used to track its states from opening to closure – and a textual description of the problem.

As discussed in the preceding section, existing approaches focus on classifying tickets into broad buckets. The right skewed distributions of resolution times of tickets within a bucket points to the need for finer grained categorization of tickets *within* each bucket. The textual description field of a ticket – which has to be mandatorily filled while raising the ticket – is the richest source of information around the problem faced by the users. But it is essentially unstructured. Thus using only this information for ticket categorization within buckets is far from a trivial problem.

As evident from Table 1, there is wide variation in the resolution times of tickets within each bucket. For the support team, it is helpful to get a sense of whether an incoming ticket will have low (L), medium (M), high (H) or very high (V) resolution times. Accordingly, we divide the resolution time distribution for each bucket into four *bins* – L, M, H, V – each mapping to a quartile of the corresponding distribution of resolution times. After TAACT is trained on a historical set of tickets, it categorizes each incoming ticket into one of these four bins. In the remainder of the paper we will use "bin" to denote these quartile of resolution time distributions, with L, M, H, and V bins indicating the first, second, third, and fourth quartiles respectively.

Given a set of past tickets in buckets with their resolution times, TAACT categorizes incoming new tickets in each bucket into bins within the buckets, using *only* the textual description field of the tickets. Thus, based on the minimal information available for all tickets, TAACT predicts whether an incoming ticket is likely to require low, moderate, high, or very high time for resolution.

In the next section, we highlight our research contributions.

3. RESEARCH CONTRIBUTION

We make the following research contributions in this paper:

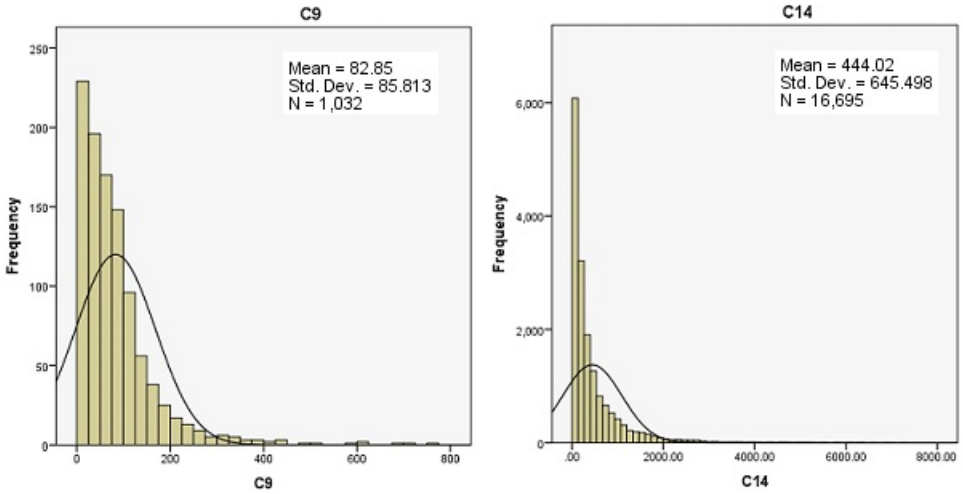

Figure 1: Histograms of buckets $C9$ and $C14$

- As outlined in the Related Work section, automated approaches to classifying problem tickets have been studied at length. Merely classifying an incoming ticket is of limited benefit to the development team unless it helps in the ticket's resolution within service level agreements. We identify the key problem of estimating whether an incoming ticket will need low, medium, high, or very high resolution times. To the best of our knowledge, this problem has not been addressed yet in existing literature.

- We propose TAACT – an approach to address the above problem, and we validate the approach on a very large real life data-set. In contrast to many studies that use open source data (see Related Work section) we use data from actual service delivery engagements. Unlike open source projects, delivery engagements with customers have very stringent service level agreements. Validating our approach on such real life data-set allows us examine TAACT's usefulness for large industrial applications.

- As illustrated in the Results section, our approach compares favorably across a diverse set of metrics with other widely used approaches. Experiments have been conducted to compare features (topics vs tf-idf), classifiers (MCF vs SVM) and regressors (SLDA vs SVR).

4. RELATED WORK

Automated approaches to the processing of problem tickets has been an area of active research interest in recent times, driven largely by the burgeoning demand of the software services industry. We outline some of the work relevant to our study.

The importance of resolving incoming problem tickets as early as possible is recognized by di Lucca in [8] and to address the problem the author seeks to develop an automated approach that can classify tickets with high accuracy. The approach is tested on around 6,000 maintenance tickets from a large, multi-site, software system. In comparison with other existing classification approaches as well as human experts, the approach is able to correctly classify up to 84% of the incoming tickets across eight areas. A classification strategy based on the use of supervised and unsupervised pattern classification and multivariate visualization is presented by Podgurski et al. [11]. They apply the technique on profiles of failed executions in order to group together failures with the same or similar causes. The resulting classification is subsequently used to assess the frequency and severity of failures due to particular defects in three large programs. Cubranic et al., use a supervised Bayesian learning based approach on bug reports from a large open-source project and report that they can correctly predict 30% of the report assignment to developers [7]. Anvik et al. present a semi-automated approach to assign incoming bug reports to developers [2]. They introduce a machine learning based classifier that suggests a small set of developers who are potentially suited to resolve a particular bug. The approach is tested on Eclipse and Firefox projects. Runeson, Alexandersson, and Nyholm present results from using a natural language processing (NLP) based prototype tool to study defect reports from a large telecommunications company [12]. They report that about 2/3 of the duplicate reports can be found using NLP techniques. Wang et al. address the problem of detecting duplicate bug reports in open source projects in [16]. They combine natural language processing with the analysis of execution traces to help a human triager better detect duplicate reports. The approach is calibrated on a subset of the Eclipse bug repository and evaluated on a subset of the Firefox bug repository and the authors report a significant improvement offered by their approach vis-a-vis using natural language information alone. Shao et al. present a Markov model based technique to mine ticket resolution sequences and develop an algorithm to generate ticket transfer recommendations on the model [13]. Incoming change requests for large software systems often contain reports of malfunction or suggestions for improvement. For the requests to be handled effectively, it is important to distinguish between the two types at the time of arrival. Antoniol et al. suggest a text-based approach to classify change requests [1]. It has been reported that a large percent of

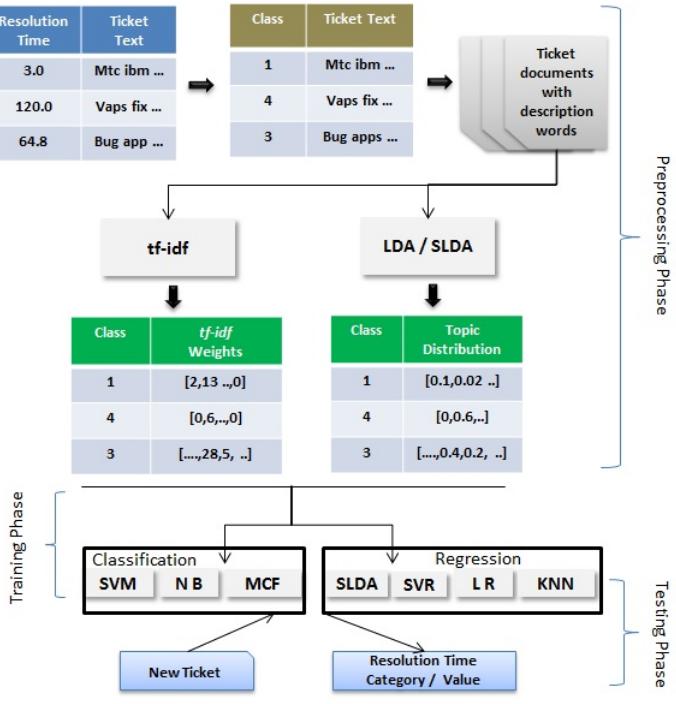

Figure 2: Outline of our approach

defects end up being repeatedly reassigned amongst development team members. To address this problem Jeong et al. propose a "bug tossing graph" based on Markov chains and report that their approach when tested on a large corpus, increased prediction accuracy by about 23%. Somasundaram et al. use a topic analysis based approach to match bug reports with appropriate code components and compare their approach with Support Vector Machines [14].

5. METHODOLOGY

5.1 Overview

As mentioned earlier, we seek to develop an approach that can predict which quartile of resolution times an incoming ticket will belong to. Apparently, it would be ideal to be able to predict the exact resolution time of an incoming ticket. But in this case, we are constrained by the fact that as predictor we can only use the textual description of the tickets. In addition to being unstructured and free-form, often the description field contains too few words to be useful for prediction. Realistically, an indication of which *level* of resolution time an incoming ticket may have – rather than an estimation of the exact resolution time – is of significant help to the support team. Thus resolution time estimation is converted to a classification problem where the resolution time quartile of a ticket is predicted.

Figure 2 outlines our approach. TAACT consists of 3 phases to categorize tickets into respective bins. In the *preprocessing phase*, the text description of each ticket is considered and different features are extracted (as explained in the next subsection). Along with feature extraction, every ticket is assigned to a bin based on its resolution time in the following way. The resolution times of all training tickets that belong to a particular bucket are considered and the 25th (q_1), 50th (q_2), and 75th(q_3) percentiles are computed. The entire range of resolution times is divided into four bins with ranges $[-\infty, q_1], [q_1, q_2], [q_2, q_3], [q3, \infty]$ respectively where each bin corresponds to Low (L), Medium (M), High (H) and Very High (V) resolution times. Each ticket is annotated with L, M, H, V depending on the bin it belongs to. In the *training phase*, the ticket features along with the information as to which bin they belong are passed as input data to train different classifiers (mentioned below). In the *testing phase* the trained classifiers are used to predict the bins which new tickets with unknown resolution times belong to. Even though predicting the category of resolution time is essential, predicting an approximate time for resolution will be very useful. The former problem is related to classification whereas the latter has to dealt with statistical regression where a real number is predicted. In this work we also predict the exact resolution time apart from the four resolution categories.

At the heart of TAACT lie *topic analysis* based classifiers and regressors, which are both explained in the next few subsections. The topic features are extracted using Latent Dirichlet Allocation (LDA) and Supervised LDA (SLDA)models. To evaluate the effectiveness of topic features, we compare them with tf-idf features using MCF, Naive Bayesian (NB) and Support Vector Machines (SVM) classifiers. Also, we evaluate the effectiveness of SLDA based topic features in predicting exact resolution times by comparing to tf-idf features with Linear Regression (LR), Support Vector Regression (SVR) and K-Nearest Neighbor (KNN) regressors. In the training phase of TAACT, all three classifiers – MCF, NB, and SVM, all three regressors LR, SVR and KNN – are trained, and they are compared based on the output of the testing phase.

5.2 Preprocessing Phase

The key element of the preprocessing phase is feature extraction. We have already established how text description is the only universally available field in ticket data. Hence we need to extract features from this field. A common approach to extracting features from text is term frequency inverse document frequency or *tf-idf*. As a benchmark to compare our topic analysis based approach, we will be using TF-IDF in the following way. From the description field of the tickets, all the unique terms are extracted and for each ticket, the frequencies of occurrences of each of the unique terms in the ticket are calculated. Thus each ticket is now represented by a vector of normalized frequencies of terms and the size of the vector is the total number of unique terms in the corpus. Using these term frequency (tf) vectors of all the ticket a ticket-term matrix is constructed where each row corresponds to a document and columns correspond to the frequencies of each of the unique terms. But many terms occur in almost all the tickets and to remove such "noise", inverse document frequencies (idf) – which gives the inverse of the frequency of occurrence of a term in the entire corpus of tickets – are calculated. The product $(tf) * (idf)$ is calculated for each ticket and these *tf-idf* weights are used as the features for each ticket. We believe in our case, there is a more effective approach for feature selection than tf-idf.

Why do tickets have different resolution times? Resolution

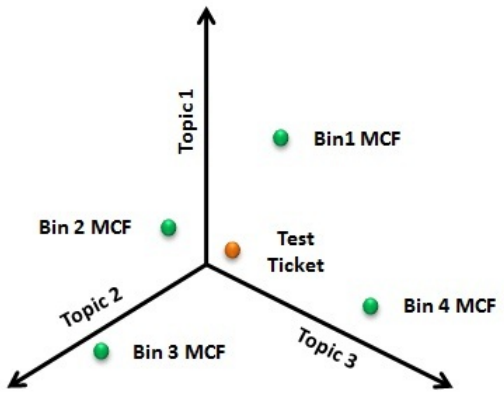

Figure 3: Explanation for Mean Class Feature (MCF) model. The plot shows the mean class features in a topic simplex.

time for a ticket indicates how quickly the support team is able to resolve and close a ticket. Other things being equal, a ticket that reports a more involved problem will take more time to resolve than a ticket describing a simpler problem. Thus resolution times differ due to the nature and intensity of the problem being addressed. This variation in problem intensity is *latent* in the ticket description. We believe understanding this hidden structure in the text description can lead us to more effective feature selection. Two probabilistic topic models called Latent Dirichlet Allocation (LDA) and supervised LDA (SLDA) can help us extract topic based features.

We assume that the factors influencing the resolution times of tickets can be modeled using the latent topics in ticket descriptions. In text mining, a topic is defined as a collection of words that can *co-occur* in a given corpus of documents. Topic models are extremely popular in the domain of text and web mining and are used for query processing and document retrieval [4]. In probabilistic topic models each document is modeled as a multinomial mixture of topics and each topic as a multinomial distribution over words. In this work we consider each ticket raised by users as a text document, and the words in the description of the ticket as the words of the document. From now on, we will use the terms tickets and documents interchangeably. In the following subsection we explain the model used for topic extraction from ticket descriptions.

5.3 Topic Analysis using LDA and Supervised LDA

Probabilistic Latent Semantic Indexing (pLSI) is one of the earliest topic models where each ticket is modeled using a unique identifier and a set of topics [10]. pLSI assigns a topic distribution to each ticket on which it has been trained but it is not a generative model and it *can not* predict the topics on new documents. To overcome these flaws in pLSI, Blei et al. proposed the Latent Dirichlet Allocation (LDA) model which can generalize to previously unseen documents as well [6]. Supervised LDA model [5], is a supervised extension of LDA where the predictor variable (resolution time in our case) is also used to influence the topics extracted. We briefly outline LDA and SLDA graphical models and the inference methodology for extracting topic distributions in our context in the Appendix section.

5.4 Training Phase

As mentioned earlier, we extract topic features using LDA and SLDA. For tickets in the training set, topic based features for each ticket and the corresponding resolution time is supplied as input for training the three classifiers. Naive Bayes is a classifier that learns the conditional probabilities of features for each bin c from the training data, which we denote by CPD_c and prior distributions of each bin c which we denote as P_c. SVM is a discriminative classifier that learns a set of parameters called the support vectors and their weights for each class c versus rest of the classes which we denote as SV_c.

MCF works with the features as normalized vectors or probability distributions. The features extracted from various tickets are grouped according to the resolution time quartile to which the tickets belong to. All features that belong to a bin c, are used to find a *mean class feature* MCF_c for that bin. Unlike the LDA-KL approach outlined in [14] for classifying bug reports to components, MCF uses symmetric Kullback Leibler divergence, which we believe is more effective distance measure between two probability vectors. Figure 3 pictorially explains the MCF model where the mean class topic features are plotted in green and the test topic feature is shown in orange. In this topic simplex, using KL divergence we assign the nearest bin as the prediction for the test ticket.

While classifiers can predict the class of resolution time, regressors can predict the exact resolution times as well. Linear regression fits a hyper plane to a set of data points and estimates the coefficients LR_{coeff} one per feature whereas Support Vector Regression finds support vectors that minimize the error of prediction. Since KNN is a lazy classifier there is no training phase for this model. Below are equations for each of the regressors and we avoid all the derivations for brevity, please refer to [3] for more details. The regression equation in Linear Regression that is used to estimate the coefficients is given by:

$$LR_{coeff} = (Feat_{train}^T Feat_{train})^{-1} Feat_{train}^T Time_{train}$$

where $Feat_{train}$ is $N \times M$ matrix of training features with N sample points and M features. $Time_{train}$ is the $N \times 1$ vector of resolution times for training data. For Support vector regression, the weights of the support vectors SVR_{coeff} are estimated using the following equation:

$$SVR_{coeff} = \sum_{n=1}^{N} (a_n - \hat{a}_n)\phi(Feat_{train})$$

where a_n and $\hat{a}_n$ are Lagrange multipliers which are calculated using training resolution times $Time_{train}$. While we train all these three regressors with tf-idf features, we use SLDA as a topic feature based regression model to predict resolution times. As in linear regression, for SLDA, we learn a set a coefficients $SLDA_{coeff}$ during the training phase.

5.5 Testing Phase

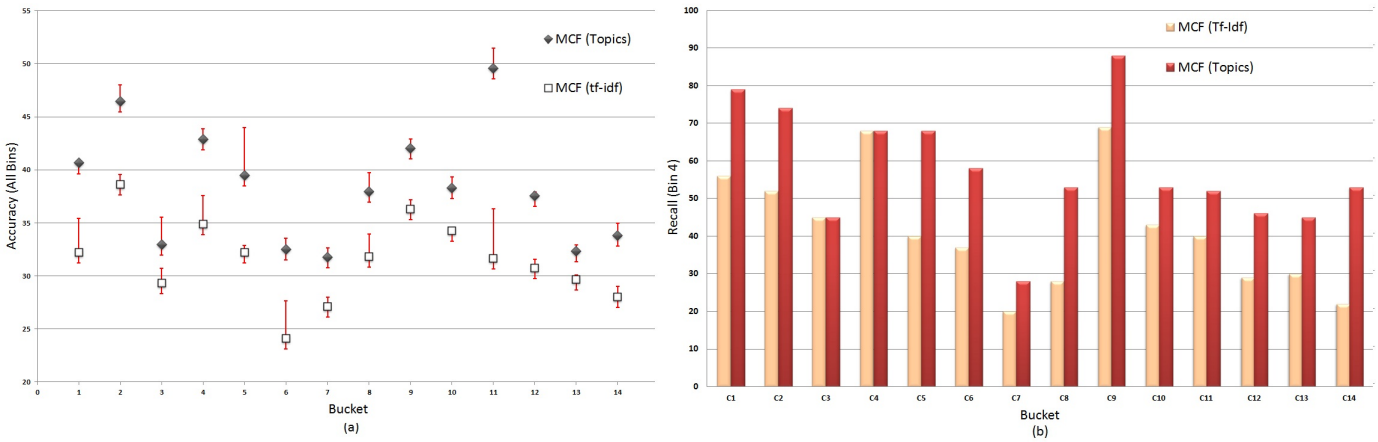

Figure 5: (a) Accuracy for each bucket averaged over all bins for MCF with tf-idf and topics features (b) Recall for fourth quartile of resolution time averaged across all folds for each of the 14 buckets using MCF with tf-idf and topics as features

With reference to Figure 2, after training comes the testing phase where the bins of new incoming tickets as well the exact resolution times are predicted. The feature vector for the new ticket, $Feature_{new}$, is extracted as described earlier. The SVM classifier considers the $Feature_{new}$ and calculates the output given by SV_c that correspond to each bin. The class with maximum SVM output is assigned to the ticket. Naive Bayes takes $Feature_{new}$ as input and estimates the posterior distribution of the ticket for each of the bins c using the CPD_c and P_c distributions. The bin that gives the maximum posterior value is assigned to the ticket.

For the MCF classifier, the bin to which a new ticket belongs to is predicted using the following formula:

$$c_{test} = \min_c KLD(MCF_c, Feature_{new})$$

where KLD is the symmetric Kullback Leibler Divergence, which is a distance measure between two probability distributions given by the following equation, where $\bar{p}$ and $\bar{q}$ are probability distributions or normalized feature vectors:

$$KLD(\bar{p}, \bar{q}) = \sum_i p(i) log(\frac{p(i)}{q(i)}) + \sum_i q(i) log(\frac{q(i)}{p(i)})$$

For Linear regression, the resolution time is predicted as the product $LR_{coeff} * Feat_{new}$. SVR model uses the weights of support vectors, SVR_{coeff} to predict the resolution time for $Feat_{new}$ and since KNN calculates the mean of the resolution times of K neighbors of $Feat_{new}$. Below is the prediction equation for KNN:

$$Time_{new} = \frac{1}{K} \sum_{n=1}^{K} Time_{train}^n$$

where $Time_{train}^n$ are the nearest training samples to the test ticket. The neighbors are selected using Cosine distance between two feature samples $\bar{p}$ and $\bar{q}$ is given by:

$$CD(\bar{p}, \bar{q}) = 1 - \frac{\sum_i (p_i \times q_i)}{\sqrt{\sum_i {p_i}^2} \sqrt{\sum_i {q_i}^2}}$$

To predict the resolution time for a new ticket, we use SLDA to extract the topics feature, $Feat_{new}$ and then predict the time as $SLDA_{coeff} * Feat_{new}$.

6. RESULTS AND DISCUSSION

6.1 Evaluation Data and Metrics

As mentioned, we evaluated TAACT on a data-set of 49,811 problem tickets across 14 buckets from four real-world service delivery projects (Table 1). All the projects were large multi-year engagements between the customer and the service delivery organization, generating large volumes of tickets. The business domains of these projects represent a diverse context for the application of TAACT.

With reference to our earlier discussion, we have extracted each of the two features *tf-idf* and *topics* for each ticket, and used them with classifiers SVM, NB and MCF and regressors SLDA, LR, SVR and KNN. To compare results, we name the classifiers as SVM-*tf-idf*, SVM-*Topics*, NB-*tf-idf*, NB-*Topics*, MCF-*tf-idf*, MCF-*Topics* and regressors as LR-*tf-idf*, SVR-*tf-idf*, KNN-*tf-idf*, SLDA-*Topics*. Topic features for classification are extracted using LDA and those for regression are extracted using SLDA.

All tickets belonging to each bucket are considered separately and divided into three folds in a three-fold evaluation strategy. While data from two folds have been used for training, the third fold is used for testing, to predict whether a ticket belongs to the low (L), medium (M), high (H) or Very High (V) bins. We did 3 fold cross validation to evaluate the model performance for both classification and regression and the mean results are reported here. We have used the following metrics for evaluating the effectiveness of each approach – *recall for 4th quartile*, *accuracy* and *weighted accuracy* for classification, *mean square error* and R^2 for regression. These metrics and reasons for choosing them are explained in the Appendix section.

6.2 Evaluation Analysis

Based on our earlier discussion, we want to establish how well topics vis-a-vis tf-idf are suited as features in SVM, NB,

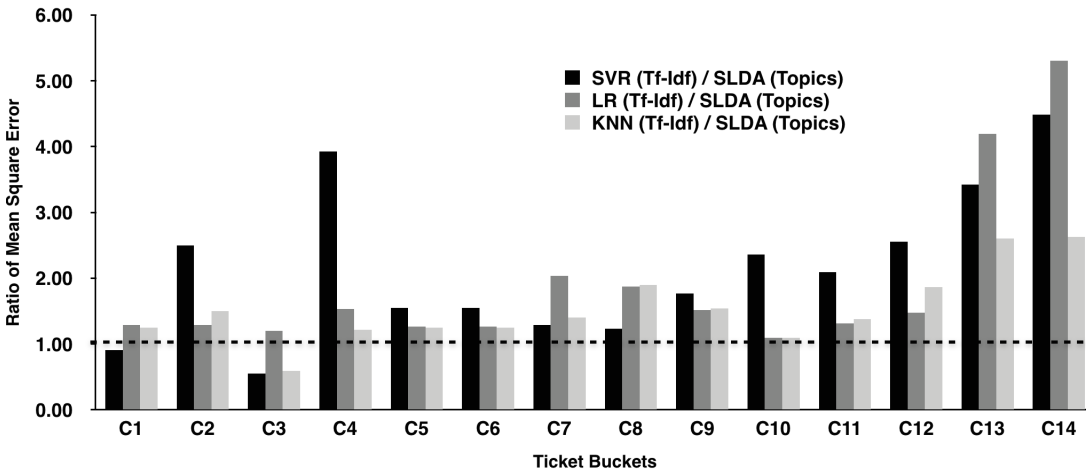

Figure 6: Plots of ratio of Mean Squared Errors using tf-idf features and topic features. Ratio greater than 1 (dotted line) implies topic features have outperformed the tf-idf features

and MCF, and how effective each of these classifiers are in predicting bins of new tickets. Similarly we evaluate topics against tf-idf using LR, SVR and KNN in predicting exact resolution times. For both SVM and SVR we used Radial Basis Function Kernel (RBF Kernel) as the kernel function. All results presented in this section are averaged using 3-fold cross validation where two folds are used for training and the left-out fold is used for testing. For extracting topic features, the optimal LDA and SLDA parameters such as number of topics and α, β and number of iterations have been selected for using 3 fold cross validation.

Classification: To get a general sense of how well each feature-classifier combination is functioning, let us first see the results presented in Figure 4. As evident, topic features give higher weighted accuracies for *all* the classifiers. This indicates that *irrespective of the classifier used, topics are much more discriminative as features than the widely used tf-idf based features.* This corroborates the arguments presented earlier in favor of choosing topics as features for the prediction problem we are addressing. As it is also clear that MCF has higher weighted accuracy than SVM or NB, let us analyze results from MCF in more detail.

Figure 5(a) plots the accuracies for each of the 14 buckets averaged across all bins. We observe that for all the buckets, MCF using topic features gives gives higher accuracy than MCF using tf-idf. The error bar is the standard deviation of accuracies for the three folds. The average standard deviations across all buckets using topic features and tf-idf features are 1.39 and 1.67 respectively, which indicate that topic features are more stable across all folds and all bins when compared to the tf-idf features.

Let us now turn to the most critical metric in our context – recall for the fourth quartile. Figure 5(b) gives plots of the recall for the V bin for tickets in each bucket, when predicted by MCF using topics as well as tf-idf as features. We again observe that topic features give higher recall values than the tf-idf features for all categories except the categories C3 and C4 where both have the same value of recall. The mean and median recall for the fourth quartile (the V bin) is 58.5% and 53.2% and 41.35% and 40.1% respectively for MCF using topics and tf-idf features. The corresponding mean and median recall for the V bin using NB and SVM with topics as features are 34.71% and 30%, and 34.86% and 32.5%, respectively. These results show that *MCF using topic based features are able to correctly identify on average more than half the tickets which have very high resolution times, which is notably higher than the outcomes from NB and SVM.*

Table 2 shows the descriptive statistics of the ratios of the fourth quartile recall values and accuracies across all buckets using MCF (*Topics*) and MCF(*tf-idf*) respectively. We observe that on average MCF with topic features performs 1.117 times better in predicting tickets across all four quartiles and 1.478 times better in predicting very high resolution time tickets when compared to MCF with tf-idf features.

Regression: In order to predict the exact resolution times, we have used both topic and tf-idf features. We used KNN model with different distance metrics and found Cosine Distance to be the optimal metric with K = 10 as the number of neighbors. These parameters are specific to the problem and need not be optimal for every kind of data. We used SLDA model for topic features and LR, SVR and KNN for tf-idf features. The Mean Squared Error for each of the 14 buckets is calculated using the error between actual and predicted resolution times. Since predicting to the scale of minutes is impractical and noisy, we converted resolution times to days for buckets with longer times and to hours for buckets with shorter times.

To evaluate the performance we have plotted the ratio of MSEs from tf-idf vs topic features which have been averaged across three folds. Figure 6 shows the plots for the ratios for 14 buckets where y-axis is the ratio of MSE using tf-idf and MSE using topic features. If a ratio is greater than one then it implies the MSE using tf-idf features is higher than topic features which indicates that the topic features have performed better and vice versa. We observe that for

all categories except C1 and C3, topic features have outperformed tf-idf features irrespective of the regression algorithm. Table 3 shows the actual MSEs using different regression algorithms validated across 3 folds across all 14 buckets. It is also interesting to note that within the regressors used with tf-idf features, KNN outperformed both SVR and LR algorithms parallel to how MCF outperformed SVM and NB algorithms for classification. This indicates that template-based lazy algorithms are able to generalize better than discriminative algorithms. It is to be noted that these topic features have been extracted using SLDA model which implies that latent topics within ticket descriptions that have been extracted using supervised learning do have information that can be used to predict resolution times automatically.

7. THREATS TO VALIDITY AND FUTURE WORK

We now outline the **threats to the validity** of our results, addressing construct validity, internal validity, external validity, and reliability.

Construct validity reflects on the correct measurement of the variables in the study. In areas with significant prior work, it involves demonstrating that the measurements align with existing state of the art or practice. The key variable in this study, resolution time of tickets, is an established measure in the software services industry. As mentioned earlier, resolution time for a ticket is taken as the elapsed time between the opening and closing of a ticket. Our calculation of resolution time is thus dependent on the ticket opening time and closing time as reported by the projects. Errors if any in reporting these time-stamps can influence our results. Also, there is the implicit assumption in our study that the resolution time for a ticket is a reliable proxy for the corresponding resolution effort. This is a generally valid assumption. However, if the closure time of a ticket was recorded – inadvertently or otherwise – as being before or after the time at which the ticket was actually resolved, resolution time will no longer accurately reflect resolution effort. Other measures relating to feature extraction using topics and tf-idf are based on past theoretical work, as discussed earlier. The evaluation metrics used in our study – accuracy, weighted accuracy, recall, and running time per ticket – are also standard measures for similar studies. Sometimes

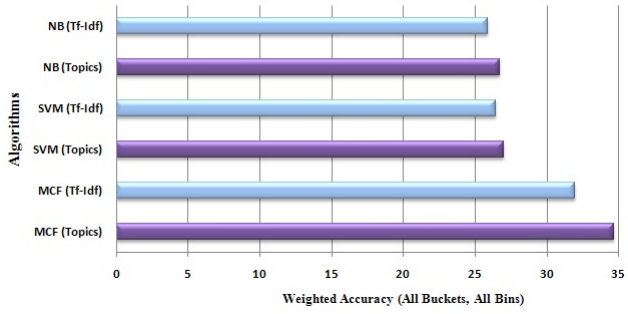

Figure 4: Weighted accuracy averaged across all buckets, folds and bins for SVM, NB and MCF with tf-idf and topics as features

while a ticket is being raised, a *severity* field is attached to indicate the required level of urgency in its resolution. In our validation we have used tickets from top two severity levels. Resolution time distribution of tickets in lower severity levels may have different characteristics. **Internal validity** ensures a study is free from systematic errors and biases. The data we have used for validation in this study has been sourced from several large real life service delivery projects. Since our data is extracted from the ticket tracking systems of these projects, issues that can affect internal validity such as mortality (that is, subjects withdrawing from a study during data collection) and maturation (that is, subjects changing their characteristics during the study outside the parameters of the research) do not arise in our case. As discussed earlier, the projects range across various domains. While our selection is broad, we do not claim the sample to be fully representative. Resolution times of tickets from projects of other domains may show different characteristics. As we have argued earlier, textual description is the only mandatory field in ticket data. Though for some tickets additional information was available (such as who raised the ticket etc), they were ignored in our analysis for the sake of consistency. **External validity** indicates that the results from a study are generalizable. Our results are based on a very large sample of tickets. While we do not claim that the results are generalizable as yet, we believe they establish that topic analysis based approach works well for predicting bins of ticket resolution times. **Reliability** relates to the reproducibility of the results of a study. We arrived at our results using an automated framework for processing and analyzing ticket data. The approach has clearly defined points where researcher judgment has to be applied – such as the selection of parameters for the LDA model. As discussed earlier, the LDA parameters in our case were chosen to give the most effective topic model. The results presented in this paper can be reproduced with the use of the framework and the relevant parameters.

In our **future work** we intend to expand our data-set to include a wider range of projects across different domains. We also intend to run a comparative study where results from our approach are matched with those from one or more human experts who manually predict the resolution time quartiles for incoming tickets. In a typical project, ticket data is accumulated over time. It will be interesting to conduct a longitudinal study using techniques such as dynamic LDA [4], to understand how resolution time characteristics vary over time. Also, often tickets manifest sub-problems of a

Table 2: Comparison of the of performance of 3-fold prediction using topics and tf-idf features with MCF classifier across all categories. The first column is ratio of Recall (R) values for the fourth quartile and the second column is the ratio of accuracies (A) across all buckets.

	R(*topics*)/R(*tf-idf*)	A(*topics*)/A(*tf-idf*)
Median	1.416	1.129
Mean	**1.478**	**1.117**
Max	2.41	1.244
Min	1	0.993
Std Dev	0.363	0.0756

Table 3: Comparison of Mean Squared Errors (averaged across 3 folds) of predicted resolution times using tf-idf and topic features

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
SVR-*tf-idf*	**0.68**	40.03	**0.41**	52.66	43.47	43.47	0.84	0.68	1.20	51.40	28.90	3.73	20.93	9.67
LR-*tf-idf*	0.96	20.65	0.90	20.59	35.56	35.56	1.33	1.03	1.03	23.78	18.19	2.15	25.66	11.43
KNN-*tf-idf*	0.93	24.00	0.44	16.32	35.09	35.09	0.92	1.05	1.05	23.76	19.08	2.72	15.91	5.67
SLDA-*topics*	0.75	**16.05**	0.75	**13.43**	**28.05**	**28.05**	**0.65**	**0.55**	**0.68**	**21.77**	**13.82**	**1.46**	**6.12**	**2.16**

larger problem. Techniques based on hierarchical LDA [4] can be effective in such situations. We plan to try out approaches based on these LDA variants in our future work.

8. CONCLUSION

In this paper we have addressed the problem of predicting the resolution times of incoming tickets in large service delivery engagements. We proposed an approach using on topic analysis based classifier to categorize incoming tickets into low, medium, high, and very high resolution times. We validated our approach on a data-set of close to 50,000 tickets from real world projects and compared topic features with tf-idf features in both classification and regression problems. We found that irrespective of the classifier and regressors used, topics are much more discriminative as features than the widely used tf-idf. Our approach is able to correctly identify more than half the tickets which have very high resolution times. In addition to predicting the discrete category, we were able to predict the actual resolution times using supervised topic models. Topic models outperformed traditional methods in 12 of 14 different ticket buckets that have been considered.

9. APPENDIX

Below are more details about Topic models and Evaluation metrics used in this paper.

9.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation is a generative model that assigns a topic distribution to each ticket where a topic is defined as a distribution over ticket description words. The difference between pLSI and LDA is that the latter assumes Dirichlet priors over the two multinomial distributions described earlier. These priors help in assigning much smoother topic distributions which help to predict the topics on unseen tickets as well. Please see [6] and [9] for further details.

9.2 Supervised Latent Dirichlet Allocation

Supervised LDA (sLDA) is an extension to LDA and it can be used as a supervised learning model where the response variable is also included in the model's structure. Please see [5] for further details on sLDA.

9.3 Evaluation metrics

- **Recall for 4th quartile:** For any classification algorithm the prediction output can be either of these four types – true positive (TP), false positive (FP), true negative (TN), and false negative (FN). In our case for example, TP would mean that a ticket actually belongs to the bin TAACT has predicted it to be, and similarly for FP, TN, and FN. In the classification context, *recall* is defined as the true positive rate or sensitivity and calculated as TP / (TP + FN). In our context, recall for a bin reflects on TAACT's ability to identify the tickets which truly belong to that bin. For the support team handling large volume of tickets, we believe it is *it is most important to be able to correctly categorize tickets in the fourth quartile of resolution time.* A simple example will illustrate this point. The magnitude of earthquakes follow a positively skewed frequency distribution with a long right tail, implying there are many small tremors (first quartile) but few large earthquakes (fourth quartile) [15]. But large earthquakes wreck maximum damage. So, for an ideal earthquake prediction system it is critical to predict a high fraction of the large earthquakes correctly, even if missing out on some of the smaller tremors. Similarly, it is of maximum benefit to the support team if as many of the tickets with the highest resolution times (that is, in the V bin) are correctly identified. Thus a key measure of the effectiveness of TAACT is recall for the fourth quartile. We calculate the recall for the fourth quartile ($Recall_4$) using the following equation.

$$Recall_4 = (\sum_{f=1}^{3} \frac{N^+_{if4}}{N_{if4}})$$

where N_{if4} is the number of tickets in the i^{th} bucket, fold f and fourth quartile (bin V) and N^+_{if4} is the number of correctly classified tickets in i^{th} bucket, f^{th} fold and fourth quartile. $Recall_4$ is calculated separately for each class across all folds and only for the bin corresponding to the fourth quartile.

- **Accuracy:** To get a sense of how well TAACT is categorizing tickets across the four bins in each bucket, we measure accuracy as (TP + TN) / (TP + FN + FP + TN). Accuracy is calculated separately for each class across all folds and bins by the following formula:

$$Accuracy = (\frac{1}{3}\sum_{f=1}^{3} \frac{1}{N_{if}}(\sum_{c=1}^{4} N^+_{ifc}))$$

where i indicates the bucket of the ticket, f is the fold, N_{if} is the number of tickets in bucket i and fold f, c is the bin, and N^+_{ifc} is the number of correctly predicted tickets in i^{th} bucket, f^{th} fold and bin c.

- **Weighted Accuracy:** For an understanding of the overall accuracy of our approach, weighted accuracy is calculated which takes into account the number of tickets in each bucket. Weighted accuracy is calculated across all classes, folds and bins, using the following

formula and the above notation:

$$WeightedAccuracy = \frac{1}{14}(\sum_{i=1}^{14}(\frac{1}{3}\sum_{f=1}^{3}\frac{1}{N_{if}}(\sum_{c=1}^{4}N_{ifc}^{+})))$$

- **Mean Squared Error:** In regression problems, where a real number is predicted, the performance of a model is evaluated by calculating the error between the actual and predicted values. Specifically Mean Squared Error (MSE) is calculated as:

$$MeanSquaredError = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where y_i and $\hat{y}_i$ are actual and predicted values.

10. REFERENCES

[1] Giuliano Antoniol, Kamel Ayari, Massimiliano Di Penta, Foutse Khomh, and Yann-Gail. Is it a bug or an enhancement?: a text-based approach to classify change requests. In *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, pages 304–318, Ontario, Canada, 2008. ACM.

[2] John Anvik, Lyndon Hiew, and Gail C. Murphy. Who should fix this bug? In *Proceedings of the 28th international conference on Software engineering*, pages 361–370, Shanghai, China, 2006. ACM.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[5] David M Blei and Jon D McAuliffe. Supervised topic models. *arXiv:1003.0783*, March 2010.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. M. L. R.*, March 2003.

[7] Davor Cubranic, Gail Murphy, Frank Maurer, and Ginther Ruhe. Automatic bug triage using text categorization. In *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*, pages 97, 92, June 2004.

[8] G. di Lucca. An approach to classify software maintenance requests. In *Proceedings of the International Conference on Software Maintenance (ICSM'02)*, page 93. IEEE Computer Society, 2002.

[9] T.L. Griffiths and M. Steyvers. Finding scientific topics. *National Academy of Sciences, U S A*, 2004.

[10] Thomas Hofmann. Probabilistic latent semantic indexing. In *Procs of 22nd ACM SIGIR Conf on Research and Development in Information Retrieval*, New York, NY, USA, 1999.

[11] Andy Podgurski, David Leon, Patrick Francis, Wes Masri, Melinda Minch, Jiayang Sun, and Bin Wang. Automated support for classifying software failure reports. In *Proceedings of the 25th International Conference on Software Engineering*, pages 465–475, Portland, Oregon, 2003. IEEE Computer Society.

[12] Per Runeson, Magnus Alexandersson, and Oskar Nyholm. Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th international conference on Software Engineering*, pages 499–510. IEEE Computer Society, 2007.

[13] Qihong Shao, Yi Chen, Shu Tao, Xifeng Yan, and Nikos Anerousis. Efficient ticket routing by resolution sequence mining. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 605–613, New York, NY, USA, 2008. ACM.

[14] Kalyanasundaram Somasundaram and Gail C. Murphy. Automatic categorization of bug reports using latent dirichlet allocation. In *Proceedings of the 5th India Software Engineering Conference*, ISEC '12, pages 125–130, New York, NY, USA, 2012. ACM.

[15] Seth Stein and Michael Wysession. *An Introduction to Seismology, Earthquakes and Earth Structure*. Wiley-Blackwell, 1 edition, September 2002.

[16] Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun. An approach to detecting duplicate bug reports using natural language and execution information. In *Proceedings of the 30th international conference on Software engineering*, pages 461–470, Leipzig, Germany, 2008. ACM.