2012 19th Asia-Pacific Software Engineering Conference

An Empirical Study on Improving Severity Prediction of Defect Reports using Feature Selection

Cheng-Zen Yang, Chun-Chi Hou, Wei-Chen Kao
Dept. of Computer Science and Engineering
Yuan Ze University
Chung-Li, Taiwan 32003
Email: {czyang,angus,wckao}@syslab.cse.yzu.edu.tw

Ing-Xiang Chen
Ericsson Taiwan Ltd.
11F, No. 1, Yuandong Road, Banqiao Dist.
New Taipei City, Taiwan 22063
Email: ing-xiang.chen@ericsson.com

***Abstract*—In software maintenance, severity prediction on defect reports is an emerging issue obtaining research attention due to the considerable triaging cost. In the past research work, several text mining approaches have been proposed to predict the severity using advanced learning models. Although these approaches demonstrate the effectiveness of predicting the severity, they do not discuss the problem of how to find the indicators in good quality. In this paper, we discuss whether feature selection can benefit the severity prediction task with three commonly used feature selection schemes, Information Gain, Chi-Square, and Correlation Coefficient, based on the Multinomial Naive Bayes classification approach. We have conducted empirical experiments with four open-source components from Eclipse and Mozilla. The experimental results show that these three feature selection schemes can further improve the predication performance in over half the cases.**

***Keywords*-Defect Reports, Severity Prediction, Feature Selection, Performance Evaluation**

I. Introduction

As mentioned in many studies, such as [1], debugging is a major concern in software maintenance because it usually costs much time and human resource to make corrections. For large-scale software projects with overwhelming numbers of defect reports, timing requirement is even much tighter due to a huge amount of debugging cost [1]. To effectively utilize limited resource in processing these incoming defect reports, the triaging process becomes an important stage in which there is usually a triager to analyze the incoming defect reports and to decide who should handle the reports and which resorts should be processed urgently.

In most defect tracking systems, the urgency of fixing the reported defects is expressed in a *priority* field. For example, in Tigris there are five priority levels from the most important (P1) to the least important (P5) for programmers to prioritize their work. Although there are many factors influencing the decision of the priority of defect reports, such as the time for the next release as discussed in [2], [3], one critical factor is the *severity* which means the impact of the defect on the execution of the software.

Ideally, a reporter consciously assigns a considered severity level to each defect report according to the defective appearance. However, as reported in [2], [3] many reporters may just use the default severity level in their reports and thus complicate the triaging process. One possible reason is that the reporters do not quite understand the definitions of different severity levels, or they do not have the solid background to decide the severity level [4]. Another possible reason is that the reporters do not pay attention to the severity assessment [2], [3]. Even the worse, the reporters may assign wrong severity levels to reports [5]. For example, a less-experienced reporter may assign the severity levels in an inverse order. If the triager has a tool to predict the severity of the incoming defect reports, especially when the number of daily incoming defect reports is very large as indicated in [1], the timing pressure in the triaging process to decide the priority of the reports can be mitigated [4].

The benefits of having a tool to provide severity predictions are thus twofold. First, it can help the triager to avoid the confusion of these mis-rated severity information by providing auxiliary severity assessment information. Second, it can be used to force the reporters to confirm their severity decisions before sending the defect reports.

Since defect reports contain textual descriptions for the defect appearances, mining these textual descriptions and learning their characteristic features is very straightforward in many previous studies [2], [3], [5]. Ideally, the assessment of the severity should be fairly objective for different reporters because the defect appearances are the same for the same software. However, predicting the severity of defect reports based on the textual content is still a challenging task because reporters may have different interpretive views for various defect situations.

In previous research, one of the studied questions is to investigate whether there are some words to be used as the indicators for severity prediction [2], [5]. In [5], Menzies and Marcus first use a traditional information retrieval technique, tf-idf (term frequency-inverse document frequency), to select top-100 words and then use a well-known feature selection scheme Information Gain (IG) [6], [7] to rank the selected words according to their IG scores for a rule-based learning scheme. They find that despite carrying less domain insights from the viewpoint of human experts, the top IG-scored words can effectively benefit the prediction task. In [2], Lamkanfi et al. use Naive Bayes classifiers to calculate the probability values of each term appearing in severity classes. They find that the top significant terms for severe defects tend to be similar across different software projects and components, but the non-severe indicators do not have such kind of property. These research observations motivate us to discuss whether feature selection schemes generally used for text mining and information retrieval can be used to extract good indicators to benefit severity prediction.

As employed in many text mining applications, feature selection is a common technique to improve the efficiency and accuracy of text classification [8], [9]. The main objective of feature selection is to identify the words in documents with high discriminability for the classification work. In this paper, we study three common feature selection schemes, Information Gain (IG) [6], [7], χ^2 (CHI) [6], [7], and Correlation Coefficient (CC) [6], [8], [10], in severity prediction based on the Multinomial Naive Bayes (NB) classification approach studied in [3]. In this research, we use these feature selection schemes to extract terms that have discriminating effectiveness for classification. These discriminating terms are potential *severe* and *non-severe* indicators for severity prediction. Then the weights of these extracted indicators are adjusted in the following training process of the Multinomial NB classifiers to see whether employing these indicators improves the prediction performance.

The major objective of this research is to investigate the following three research questions.

- *RQ1: Can these feature selection schemes effectively find the potential indicators to improve the performance of severity prediction on defect reports?* Although feature selection has been proven to be an effective approach to improve classification performance, its effectiveness has not been comprehensively studied for the severity prediction task. The work in [5] has used IG for reranking the word order to train the rule-based classifiers. However, the influence of these indicators is not further explored. In this research, we discuss the effectiveness of different feature selection schemes on four software components of two open-source software projects.
- *RQ2: How many identified indicators should be used and how much weight should be applied?* One challenge of using feature selection is to decide an appropriate number of indicators and their weights to benefit the prediction task. If too many terms are considered as the indicators in the prediction model, the prediction performance suffers from introducing false severe indicators. The weights of these terms also have impacts on the prediction performance. In addition, these factors may be closely related to the characteristics of software projects. The word usage of reports plays an important role in deciding the indicator numbers and their weights. In this study, these issues are investigated to discover their relationships with the discussed software projects.
- *RQ3: Can both severe and non-severe indicators be used to improve the prediction performance?* As shown in [2], severe indicators tend to have high specificity across software components and projects, but non-severe indicators are much diversified. Accordingly, will employing many non-severe indicators or giving them much weight in the prediction task improve the prediction performance? We investigate the influences of the *severe* and *non-severe* indicators to show their discriminative power.

The organization of this paper is as follows. In Section 2, we will briefly review the previous work. In Section 3, we describe the problem definition for the severity prediction task. Section 4 presents the feature selection techniques investigated in this research. Section 5 describes the experimental results based on four software components of the Eclipse and Mozilla projects. Section 6 discusses the possible threats to the validity of this research work. Finally, Section 7 concludes the paper.

II. Related Work

In the research areas of software maintenance, severity prediction is an emerging issue since software repository mining grants many attentions recently. In 2008, Menzies and Marcus proposed an automatic severity prediction mechanism by first selecting the most important words according to their *tf-idf* (term frequency-inverse document frequency) scores, then reranking the most informative terms according to their Information Gain (IG) measures, and finally using a rule-based machine learning tool to learn the classification rules [5]. In this study, the prediction task focuses on fine-grained severity categorization with five severity levels for five NASA software projects, and they study the prediction performance of using different numbers of informative terms. Their experimental results show that the performance of the proposed rule-based approach varies dramatically due to the divergent sizes of the report collections. However, their study does not discuss the effectiveness of IG-reranking. The comparison discussion with other feature selection schemes are also absent in their current work.

In 2010, Lamkanfi et al. explored the severity prediction problem further to answer more questions of using text mining techniques [2]. In their study, Naive Bayes (NB) classifiers were used in three open source projects, Mozilla, Eclipse, and GNOME. The NB classifiers provide severity predictions according to probability of the appearance of words in the short summary descriptions. The main objective of their study is to discuss four important research issues in applying text mining techniques to the severity prediction task. First, they found that some terms are good indicators for the severity prediction. Especially in their case-study, severe indicators tend to have high specificity across different software components, but non-severe indicators are much diversified. Second, they found that considering the longer full descriptions of defect reports impairs the prediction performance due to misidentifying many severe defect reports as non-severe reports. The possible reason is that longer descriptions may contain many irrelevant words like stack traces and segments of source code such that the informative words cannot be effectively extracted. In contrast, the words in short summary descriptions are generally more concise to explain the defects. In addition to the above two observations, they discussed the number of the training examples for a good, stable prediction performance. They found that although the numbers are varied in different software components, the prediction performance will be stable if there are enough defect reports. The last observation considers the cross-component prediction. They found that the cross-component expansion cannot effectively help severity prediction in some software components because some problem-specific characteristics exist in these components.

In 2011, Lamkanfi et al. further made a comparison study to discuss the prediction effectiveness of the NB classifiers, the Multinomial NB classifiers, 1-NN classifiers, and Support Vector Machines (SVM) [3]. Their experimental results show that Multinomial NB classifiers can achieve the highest performance in the Area Under Curve (AUC) measures using the Receiver Operating Characteristic (ROC) evaluation. According to the studies in [2], [3], we discuss the effectiveness of feature selection methods based on the Multinomial NB classifiers. Since the components studied in this paper have problem-specific characteristics as indicated in [2], we do not discuss the cross-component prediction in this study.

There are other related studies on the feature selection techniques for software engineering research. For example, Shivaji et al. used Gain Ratio (GR) [11], an improvement of IG, to enhance bug prediction [12]. The improvement of Gain Ratio comes from the fact that it can reduce the biases resulted from IG if the dataset has some attributes having a large number of possible values, or very specific for the data items, such as the report identifier [13]. However, as pointed in [13] in studying the robustness of various feature selection schemes on 16 datasets of software projects, GR has the least stability, but IG and χ^2 (CHI) have the moderate stability performance. Therefore, in this paper we focus on the effectiveness of three common feature selection techniques, i.e., IG, CHI, and CC, based on the Multinomial NB scheme proposed in [3].

III. Problem Definition

The severity prediction problem studied in this paper is defined as a classification problem on incoming defect reports with a classification function f_j to predict the severity level s_i of each incoming defect report r_x:

$$s_i = f_j(r_x)$$

The prediction function f_j can be constructed using different machine learning algorithms. In this study, we use the Multinomial NB classification model to train the classifiers with various feature selection schemes according to its performance superiority in [3].

There are two categories of indicators extracted in the feature selection schemes: *severe* indicators and *non-severe* indicators. The severe indicators are defined as the representative words extracted from the historical severe defect reports with high term frequencies. Similarly, non-severe indicators are extracted from the historical non-severe defect reports. These indicators are expected to have high discriminability for the prediction work.

To evaluate the effectiveness of the feature selection schemes, we use two major severity categories in this study to investigate the effectiveness of feature selection schemes as discussed in [2], [3]. The set S of the severity categories is thus $\{s_0, s_1\}$, where s_0 denotes the *non-severe* class, and s_1 is the *severe* class. Since we also take the same datasets maintained with Bugzilla for the experiments, s_0 is currently ranged from the Bugzilla severity level *trivial* to *minor*, and s_1 is ranged from *major*, *critical*, to *blocker*. For performance evaluation, we adopt the 10-fold cross-validation approach to assess the prediction performance. In the experiments, however, the defect reports labeled *normal* were discarded as discussed in [2] because the *normal* level is the default option and thus many defect reports labeled do not have the correct decisions on the severity levels according to their manual observations. Considering the defect reports labeled *normal* in the experiments needs many human efforts to validate the perdition results. Thus the investigation of the normal-level defect reports is left to our future study.

IV. Severity Prediction Enhancement

In this section, we present the prediction approach with feature selection enhancements. We first provide a brief description for the prediction framework. Then the studied feature selection metrics are elaborated. Finally, other processing techniques and the NB classification model used for the severity prediction task are described.

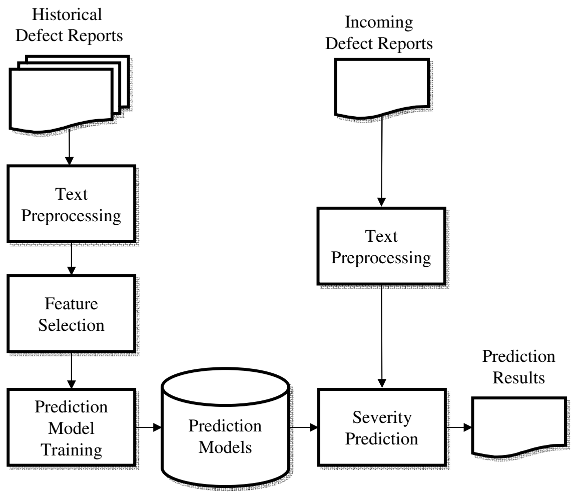


Fig. 1. The process flow of severity prediction for incoming defect reports.

A. General Framework

To predict the severity of the incoming defect reports, classifiers containing the severity knowledge need to be trained first. Figure 1 shows the prediction flow used in our study. The prediction process comprises three main steps: text preprocessing, feature selection and reweighting, and prediction model training. In each step, we employ different mechanisms in order to evaluate the effectiveness of various feature selection schemes.

B. Feature Selection Schemes

In this paper, three common feature selection schemes are investigated: Information Gain (IG) [6], [7], χ^2 (CHI) [6], [7], and Correlation Coefficient (CC) [6], [8], [10]. The characteristics of these feature selection schemes are closely related to the probabilities of each term present or absent in various severity categories. Based on these probabilities, each term can get its IG, CHI, and CC scores with respect to a severity category. For a severity category s_i, $P(t_k, s_i)$ denotes the probability that a term t_k appears in s_i, and $P(\bar{t}_k, s_i)$ denotes the probability that a term t_k is absent in s_i. Similarly, $P(t_k, \bar{s}_i)$ denotes the probability that a term t_k appears in other categories rather than s_i, and $P(\bar{t}_k, \bar{s}_i)$ denotes the probability that a term t_k is absent in other categories rather than s_i. If there are N defect reports, these feature selection schemes can be expressed as follows.

$$IG(t_k, s_i) = \sum_{s \in \{s_i, \bar{s}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, s) \log \frac{P(t, s)}{P(t)P(s)}$$

$$CHI(t_k, s_i) = \frac{N[P(t_k, s_i)P(\bar{t}_k, \bar{s}_i) - P(t_k, \bar{s}_i)P(\bar{t}_k, s_i)]^2}{P(t_k)P(\bar{t}_k)P(s_i)P(\bar{s}_i)}$$

$$CC(t_k, s_i) = \frac{\sqrt{N}[P(t_k, s_i)P(\bar{t}_k, \bar{s}_i) - P(t_k, \bar{s}_i)P(\bar{t}_k, s_i)]}{\sqrt{P(t_k)P(\bar{t}_k)P(s_i)P(\bar{s}_i)}}$$

The main purpose of IG is to find the features that dominate the amount of information of representing a classification category. The most informative term has the largest IG score. In [5], IG has been used to select the top-k informative terms with the benefit of dimensionality reduction of the feature space for severity prediction. However, a word with a large IG score can be either a severe indicator or a non-severe indicator because IG only considers its discriminative power for classification.

As noted in [7], the $CHI(t_k, s_i)$ score reflects the lack of independence between t_k and s_i. If t_k and s_i are independent, the CHI score will be zero. The main benefit of using CHI scores for feature selection is that they are normalized such that the CHI scores of terms in the same category are comparable. However, two shortcomings have been pointed out for the CHI scheme. First, if the term frequency of t_k is very low, its CHI score cannot accurately reflect the lack of independence information [7], [14]. Second, this measure only concerns the independence information. It does not distinguish negatively related terms from positively ones [8], [10]. Therefore, a word with a large CHI score can be either a severe indicator or a non-severe indicator as IG.

The formula of the CC scheme is very similar to the CHI scheme except that CC can express the polarity of a term t_k in the category s_i. That means term t_k with a large positive $CC(t_k, s_i)$ score is the representative term for s_i. Therefore, CC can identify the most informative terms for each category [8], [10]. The indicators of a category can be separated from the indicators of other categories according to their CC scores. As shown in [8], [10], CC outperforms CHI in general classification tasks. For CHI and IG, they have comparable performance in different studies [7], [9], [13]. However, CC and CHI have very similar performance for small feature sets and highly imbalanced datasets [9]. Therefore, this study explores their classification performance for the severity prediction tasks.

C. Text Preprocessing

The reports are studied according to their designated components and the corresponding software packages. Following the approach in [2], we extract terms from the short description (summary) field of the defect reports for severity prediction in the text preprocessing step because short descriptions provide summarized information which usually has good severity indicators, such as "fail", "crash", and "hanging". Then the extracted words are processed with the following steps:

- *Tokenization*: Each individual word is extracted from the textual strings. In the experiments, we used the default settings of WVtool [15] to separate the tokens. At the same time, all words are converted to lower cases.
- *Stopwords removal*: Stopwords are defined as the words containing too less specific meaning to be used for the classification purpose. For example, terms like "the", "to", and "one" are several common stopwords because they do not have specific information for the contextual meaning of the documents. For text mining techniques relying on statistic machine learning approaches, these stopwords are usually removed to enhance the classification performance.
- *Stemming*: Since in natural languages a term may have different forms due to the grammar rules, stemming simplifies the processing complexity by converting terms to their morphological root forms without loss of their specific meanings. For example, "crashes" and "crashed" are stemmed to "crash". In the experiments, we used the Porter stemmer implemented in WVtool to get the base form of a given term because this stemming scheme is widely used in many text mining applications.

D. Feature Selection and Reweighting

After all words are preprocessed, the aforementioned feature selection schemes are used to extract the most informative words as the prediction indicators for s_1 or s_0. In this study, each term receives its IG, CHI, and CC scores for s_0 and s_1, respectively, and finally we obtain six rank lists for all words.

From each rank list, top-m terms are used to study the influences of indicators. The term frequencies of these indicators will be multiplied by a weighting factor α with the following reweighting equation:

$$tf'_l = \alpha \cdot tf_l$$

Various weighting methods are investigated to explore the influences of the different sources of the indicators and the categories of the defect reports to be weighted. The indicators are extracted either from both historical severe defect reports and non-severe defect reports, or historical severe defect reports only. For the reweighting categories, we discuss two situations. The first reweighting experiment is to reweight the terms of all defect reports with all indicators identified from both historical severe and non-severe defect reports. Another is to separately reweight the terms in the severe defect reports with the indicators extracted from the historical severe defect reports only, and the terms in the non-severe defect reports with the indicators extracted from the historical non-severe defect reports. Then we train the Multinomial NB classifiers with these reweighted terms to obtain the enhancement. The testing defect reports are also reweighted accordingly.

E. Prediction Model Training

In this study, we used the Naive Bayes (NB) classification model to investigate the effectiveness of feature selection. The NB model is a well-known classification approach in text mining research due to its simplicity and good performance. Since NB with multinational probability distributions has the best prediction performance among NB, 1-Nearest Neighbor, and Support Vector Machines in [3], the feature selection schemes are studied on the basis of Multinomial NB classifiers.

In the Multinomial NB model, the classification probability of an incoming defect report r_x to the severity class s_i can be defined as $P(s_i|r_x)$ which can be modeled as the posterior probability:

$$P(s_i|r_x) = \frac{P(s_i)P(r_x|s_i)}{P(r_x)}$$

Since $P(r_x)$ is the same for all s_i, it can be ignored. Based on the Naive Bayes assumption that each word t_l in r_x is conditionally independent, the posterior probability $P(s_i|r_x)$ can be thus modeled as

$$\begin{aligned} P(s_i|r_x) &\propto P(s_i)P(r_x|s_i) \\ &= P(s_i)P(t_1, t_2, \ldots, t_n|s_i) \\ &= P(s_i)P(t_1|s_i)P(t_2|s_i) \times \cdots \times P(t_n|s_i) \\ &= P(s_i) \prod_{1 \leq l \leq n} P(t_l|s_i) \end{aligned}$$

The probability $P(s_i)$ is estimated by

$$P(s_i) = \frac{\text{The number of defect reports in } s_i}{\text{Total number of defect reports in the training set}}$$

To estimate $P(t_l|s_i)$ without considering the existence of indicators, we first compute the normalized term frequency tf_l of t_l appearing in the document set of s_i by

$$P(t_l|s_i) = \frac{tf_l}{\sum_{\forall t_v \in V} tf_v}$$

where V is the vocabulary set of all words. However, this equation has the zero-frequency problem. Therefore, we smooth $P(t_l|s_i)$ using Laplace smoothing as [16], [17] and $P(t_l|s_i)$ is estimated by

$$P(t_l|s_i) = \frac{1 + tf_l}{|V| + \sum_{\forall t_v \in V} tf_v}$$

Since top-m indicators have been identified in the feature selection schemes, the reweighted term frequencies tf'_l are used to train the enhanced Multinomial NB models in which the discriminative power of these indicators is considered. The α factor controls the impact of the discriminative power of the severity indicators. With different m and α settings, we discuss the influences of these extracted indicators in different feature selection schemes.

V. Experimental Results and Analysis

A. Datasets

Experiments of this study used defect reports collected from four components of two major open-source projects: Eclipse and Mozilla. These two projects are all maintained by Bugzilla. For Eclipse, the authors of defect reports are expected to have programming experiences because Eclipse is mainly for program developers. Therefore, the defect reports of Eclipse are expected to have indicators in good quality. That means the reporters will consistently use words with similar semantics to describe the severe and non-severe defects. For Mozilla, the reporters may have diversified background for the software development. Therefore, the defect reports of Mozilla may contain fewer indicators or less informative terms. That means the feature selection schemes may have little benefits in extracting informative terms for the prediction task.

Table I describes the details of these components. The defect reports in the experiments all have severity tags to indicate their severity levels.

TABLE I
The numbers of defect reports of the studied datasets

Project	Component	Non-severe DR	Severe DR	Period
Eclipse	UI	1472	3338	2001-2010
Eclipse	JDT-UI	1456	1551	2001-2010
Mozilla	General	2757	10125	1998-2010
Mozilla	Layout	636	1862	1998-2010

The GNOME project studied in [2], [3] were not considered because it has many automatically generated defect reports which may highly facilitate the severity prediction task due to these artificial terms. This preference can be implicitly inferred from the experimental results in [2], [3] because the prediction performance results on GNOME datasets are on average higher than the results on Eclipse and Mozilla.

B. Evaluation Measures

To evaluate the effectiveness of feature selection schemes, we use the Receiver Operating Characteristic (ROC) curves as in [3]. The ROC curves show the variations of the rate of true positives (TPR) against the rate of false positives (FPR) with the following definitions:

$$TPR = \frac{tp}{\text{Total number of positives}} = \frac{tp}{tp + fn}$$

$$FPR = \frac{fp}{\text{Total number of negatives}} = \frac{fp}{fp + tn}$$

where tp (*true positives*) is the number of correctly predicted non-severe reports, fn (*false negatives*) is the number of wrongly predicted severe reports, fp (*false positives*) is the number of wrongly predicted non-severe reports, and tn (*true negatives*) is the number of correctly predicted severe reports.

To make comparisons among classification algorithms, most studies use the Area Under Curve (AUC) measures of ROC curves because AUC can more accurately demonstrate the performance differences than making visual comparisons on ROC curves. The AUC metric shows the probability that a classifier can rank a positive instance higher than a negative one. An AUC value close to 1.0 represents that the classifier makes perfect predictions. If the AUC value is around 0.5, that means the prediction capability of the classifier is equivalent to making random predictions. From the AUC measures, we can clearly perceive the classification effectiveness of various feature selection schemes.

C. Experimental Results

To evaluate the effectiveness of feature selection schemes, we used a 10-fold cross validation approach. All datasets were first equally split into 10 disjoint folds in which 9 folds were used as the historical defect reports for training and the other one was used for testing. This step was repeated 10 times and each time we chose a different fold as the testing set in a rotational manner.

RQ1: Can these feature selection schemes effectively find the potential indicators to improve the performance of severity prediction on defect reports?

To answer this question, we used three feature selection schemes to extract top-m indicators from the short descriptions of defect reports. Table II shows the extracted top-20 indicators in Eclipse UI whose scores are averaged from the 10-fold cross validation experiments. For IG and CHI, the severe indicators and the non-severe indicators are mixed according to their IG or CHI scores. For CC, the severe indicators are separated from the non-severe indicators. In the table we can find that CC effectively identifies both severe indicators and non-severe indicators. For IG and CHI, the informative indicators can be extracted, but severe and non-severe indicators are mixed in the lists. For example, "javadoc" has the highest IG and CHI scores, but it is a non-severe indicator because it appears more frequently in non-severe defect reports.

The ROC results for Eclipse UI are depicted in Figures 2, 3, and 4 while α is 10 and m is ranged from 0 to 100. When $m = 0$, that means no term was reweighted, and the original term frequencies were used to train the baseline Multinomial NB classifiers. In the figures, the baseline ($m = 0$) is represented with a bold line. Figures 2 to 4 show that these feature selection schemes can effectively identify the indicators with good quality. Among three feature selection schemes, CC can highly facilitate the prediction performance than other two schemes for Eclipse UI. The effectiveness of feature selection for all software projects can be further observed from the AUC analysis as studied in RQ2.

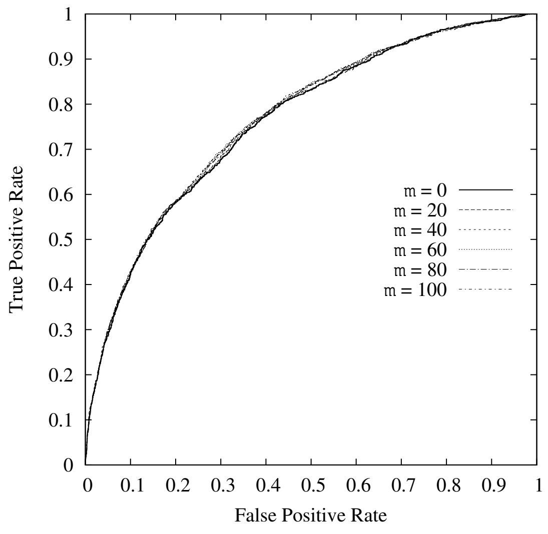


Fig. 2. The ROC curves for IG in Eclipse UI with different numbers of extracted features.

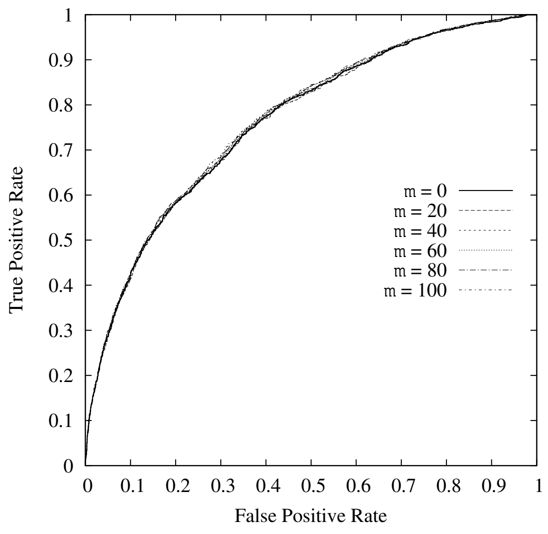


Fig. 3. The ROC curves for CHI in Eclipse UI with different numbers of extracted features.

RQ2: How many identified indicators should be used and how much weight should be increased?

To answer this question, we calculated the AUC values from the ROC curves. Since there are different kinds of weighting methods to be investigated, the indicator source can be either from all historical defect reports (All) or from the historical severe defect reports only (Severe DR). For the reweighting process, we considered two methods: (1) weighting the terms of all defect reports with all indicators identified from both historical severe and non-severe defect reports (All) and (2)

TABLE II
THE EXTRACTED TOP-20 INDICATORS IN ECLIPSE UI FROM BOTH HISTORICAL SEVERE AND NON-SEVERE DEFECT REPORTS.

Rank	IG				CHI				CC					
									Severe			Non-Severe		
	Term	tf in s_1	tf in s_0	IG()	Term	tf in s_1	tf in s_0	CHI()	Term	tf	—CC()—	Term	tf	—CC()—
1	javadoc	0.00028	0.01000	0.00434	javadoc	0.00028	0.01000	143.68113	except	0.00601	6.03170	javadoc	0.01000	11.98454
2	leak	0.00438	0.00000	0.00168	dialog	0.01471	0.02768	47.37992	leak	0.00438	5.82817	dialog	0.02768	6.87657
3	except	0.00601	0.00058	0.00150	typo	0.00006	0.00292	43.14540	editor	0.02212	4.99109	typo	0.00291	6.56861
4	dialog	0.01471	0.02768	0.00136	except	0.00601	0.00058	36.40376	crash	0.00337	4.89282	prefer	0.01908	5.80282
5	typo	0.00006	0.00292	0.00132	leak	0.00438	0.00000	33.99138	dispos	0.00489	4.77262	messag	0.00570	5.30825
6	crash	0.00336	0.00012	0.00107	prefer	0.01000	0.01908	33.81216	regress	0.00303	4.39648	button	0.00779	5.01136
7	prefer	0.01000	0.01908	0.00097	messag	0.00168	0.00570	28.27152	caus	0.00517	4.34856	titl	0.00349	4.94321
8	dispos	0.00489	0.00093	0.00087	button	0.00309	0.00779	25.40695	open	0.00696	4.28877	icon	0.00640	4.91296
9	regress	0.00303	0.00023	0.00082	editor	0.02212	0.01268	25.01120	java	0.00483	4.07798	align	0.00163	4.75962
10	editor	0.02212	0.01268	0.00082	titl	0.00073	0.00349	24.51900	commonnavig	0.00719	3.82561	descript	0.00302	4.47828
11	messag	0.00168	0.00570	0.00079	icon	0.00230	0.00640	24.26724	hang	0.00202	3.67691	wizard	0.00803	4.15914
12	button	0.00309	0.00779	0.00071	crash	0.00336	0.00012	23.79545	us	0.00427	3.59750	vertic	0.00128	4.14227
13	caus	0.00517	0.00140	0.00069	dispos	0.00489	0.00093	22.82974	work	0.01291	3.59241	nitpick	0.00093	3.86127
14	titl	0.00073	0.00349	0.00068	align	0.00006	0.00163	22.53832	deadlock	0.00163	3.55065	mislead	0.00093	3.85504
15	icon	0.00230	0.00640	0.00068	descript	0.00067	0.00302	20.09930	thread	0.00157	3.48835	size	0.00291	3.68584
16	align	0.00006	0.00163	0.00068	regress	0.00303	0.00023	19.33763	mgmt	0.00410	3.44657	keyword	0.00105	3.67206
17	open	0.00696	0.00256	0.00064	caus	0.00517	0.00140	18.93793	broken	0.00275	3.25306	extra	0.00163	3.62755
18	deadlock	0.00163	0.00000	0.00062	open	0.00696	0.00256	18.46787	slow	0.00163	3.24046	page	0.01024	3.62254
19	thread	0.00157	0.00000	0.00060	wizard	0.00387	0.00803	17.48600	coolbar	0.00225	3.15762	window	0.00965	3.57768
20	java	0.00483	0.00140	0.00060	vertic	0.00006	0.00128	17.04694	virtual	0.00124	3.09046	html	0.00140	3.43950

TABLE III
THE AUC MEASURES OF DIFFERENT FEATURE SELECTION SCHEMES FOR ECLIPSE JDT-UI AND UI UNDER DIFFERENT WEIGHTING CONFIGURATIONS.

Training Data		Testing Data		Eclipse JDT-UI, Baseline = 0.7447			Eclipse UI, Baseline = 0.7635		
Indicator Sources	Weighted DR	Indicator Sources	Weighted DR	IG	CHI	CC	IG	CHI	CC
All	All	None	None	0.7531 (15,10)	0.7531 (15,10)	0.7470 (30,10)	0.7686 (20,10)	0.7671 (55,10)	0.7709 (50,10)
All	Separate	None	None	0.7513 (15,10)	0.7513 (15,10)	-	0.7682 (35,10)	0.7666 (60,10)	-
Severe DR	All	None	None	0.7544 (25,10)	0.7535 (30,10)	**0.7677 (35,10)**	0.7699 (65,10)	0.7689 (100,10)	**0.7768 (20,10)**
Severe DR	Severe DR	None	None	-	-	0.7536 (20,10)	-	-	0.7675 (20,4)
All	All	All	All	0.7471 (20,2)	-	-	0.7649 (10,2)	0.7641 (65,2)	-
All	Separate	All	All	0.7463 (20,2)	-	-	0.7646 (10,2)	0.7643 (70,2)	-
Severe DR	All	All	All	0.7471 (20,2)	-	0.7518 (90,4)	0.7644 (10,2)	0.7642 (70,2)	0.7673 (100,2)
Severe DR	Severe DR	All	All	-	-	-	-	-	-
All	All	Severe DR	All	0.7467 (40,2)	0.7455 (35,2)	-	0.7654 (20,2)	0.7645 (20,2)	-
All	Separate	Severe DR	All	0.7465 (40,2)	0.7458 (35,2)	-	0.7653 (20,2)	0.7643 (20,2)	-
Severe DR	All	Severe DR	All	0.7475 (40,2)	0.7460 (35,2)	0.7466 (25,2)	0.7651 (20,2)	0.7641 (100,2)	0.7652 (20,2)
Severe DR	Severe DR	Severe DR	All	-	-	-	-	-	-

separately weighting the terms in the severe/non-severe defect reports with the indicators extracted from the historical severe/non-severe defect reports (Separate). Tables III and IV show the best prediction performance results of three feature selection schemes for various reweighting methods in Eclipse and Mozilla. The numbers in the parentheses show the respective (m, α) values. If there is no better result, a "-" is labeled. The bold numbers are the highest AUC values in these four software components. From the tables, we can find that the classifiers with these feature selection schemes outperform the baseline classifiers in over half the cases.

These two tables also show that CC can achieve the best performance in Eclipse JDT-UI and UI when both sever indicators and non-severe indicators are all from historical severe defect reports, but it has poor performance in Mozilla General and Layout. In contrast, IG and CHI have moderate enhancements for most cases, and they have the highest performance in Mozilla. However, the improvements on Eclipse and Mozilla are not consistent. Employing indicators in Eclipse can get significant improvements but the improvements are very minor in Mozilla. This comes from the fact that although there are much more severe defect reports in Mozilla, few informative severe indicators can be extracted. Most informative indicators are non-severe indicators. In contrast, the numbers of severe indicators and non-severe indicators are comparable in Eclipse.

After our manual inspection, we find that the performance variation is closely related to the characteristics of the software projects. Since the reporters tend to use the comparable numbers of severe indicators and non-severe indicators to describe the severity in Eclipse, CC can highly improve the prediction performance due to its clear separation for term polarities. If both severe and non-severe terms are equally likely reweighted, CC with larger α weights may improve the prediction performance.

The performance degradation of CC in Mozilla is mainly due to the report characteristics of the Mozilla project. Since the words used by the reporters of the Mozilla project are comparatively diversified, the CC scores of most severe indicators

TABLE IV
THE AUC MEASURES OF DIFFERENT FEATURE SELECTION SCHEMES FOR MOZILLA GENERAL AND LAYOUT UNDER DIFFERENT WEIGHTING CONFIGURATIONS.

Training Data		Testing Data		Mozilla General, Baseline = 0.8478			Mozilla Layout, Baseline = 0.7940		
Indicator Sources	Weighted DR	Indicator Sources	Weighted DR	IG	CHI	CC	IG	CHI	CC
All	All	None	None	0.8482 (10,2)	0.8482 (55,3)	-	0.7952 (50,2)	0.7959 (75,3)	-
All	Separate	None	None	0.8483 (10,2)	0.8483 (55,3)	-	0.7953 (50,2)	0.7961 (75,2)	-
Severe DR	All	None	None	0.8482 (10,2)	0.8482 (5,2)	0.8480 (5,2)	-	0.7945 (90,2)	-
Severe DR	Severe DR	None	None	-	-	-	-	-	-
All	All	All	All	0.8481 (30,2)	0.8488 (5,2)	-	0.7979 (70,2)	0.7983 (70,2)	-
All	Separate	All	All	0.8481 (30,2)	0.8487 (5,2)	-	0.7977 (70,2)	**0.7983 (70,2)**	-
Severe DR	All	All	All	0.8482 (30,2)	0.8488 (5,2)	0.8478 (5,2)	0.7971 (70,2)	0.7977 (70,2)	-
Severe DR	Severe DR	All	All	-	-	-	-	-	-
All	All	Severe DR	All	0.8499 (5,2)	0.8488 (10,2)	0.8482 (5,2)	0.7948 (50,2)	0.7963 (85,2)	0.7942 (25,2)
All	Separate	Severe DR	All	0.8499 (5,2)	0.8488 (5,2)	-	0.7952 (85,2)	0.7963 (85,2)	-
Severe DR	All	Severe DR	All	**0.8499 (5,2)**	0.8488 (10,2)	0.8482 (5,2)	0.7947 (90,2)	0.7949 (90,2)	0.7941 (25,2)
Severe DR	Severe DR	Severe DR	All	-	-	-	-	-	-

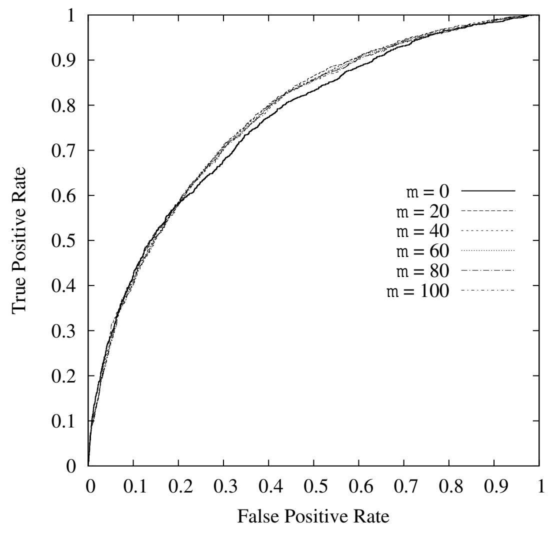


Fig. 4. The ROC curves for CC in Eclipse UI with different numbers of extracted features.

are small. However, equal numbers of severe and non-severe indicators of CC were used in the experiments. Employing too many severe indicators for reweighting hinders the prediction performance. In contrast, the mixed indicator lists of IG and CHI have many informative non-indicators and help IG and CHI to have better performance.

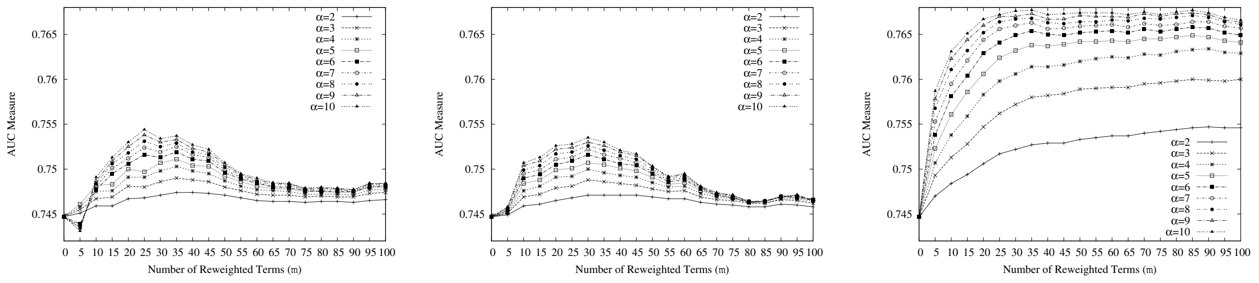
In the tables, we can also notice that if we separately reweight the terms of the training data according to their severity class, no feature selection scheme can have good performance. The main reason is that the probability distributions of words are twisted too much. The Multinomial NB model is thus suffered.

From these tables, we infer that for software projects whose reporters have programming experiences, CC with larger α values may be the best choice to improve the prediction performance. If the reporters of the software project like Mozilla are not so experienced in programming, IG or CHI with smaller α values can be considered because they comparably have improvements in most cases. In Figures 5 to 8, the prediction performance of different software projects with various m and α settings validate the above inference. Figures 5 and 6 illustrate the experiments for Eclipse JDT-UI and UI while the indicators were extracted from the historical severe defect reports and only the training models were reweighted (the *Severe DR-All-None-None* row in Table III). For Eclipse, feature selection schemes with larger α values tend to have more improvements. Figures 7 and 8 illustrate the experiments for Mozilla General and Layout while the training models were reweighted by the indicators extracted from all historical defect reports and the testing reports were reweighted by the indicators extracted from the historical severe defect reports only (the *All-All-Severe DR-All* row in Table IV). They validate the inference that feature selection schemes with smaller α values tend to have improvements for Mozilla.

RQ3: Can both severe and non-severe indicators be used to improve the prediction performance?

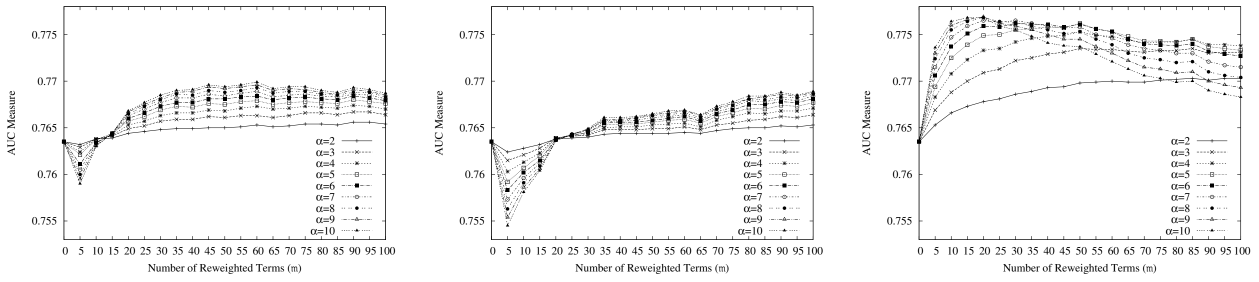
Tables III and IV also show that the answer depends on the characteristics of the projects. For Eclipse, both severe and non-severe indicators can benefit the prediction task. However, non-severe indicators play a more important role in Mozilla. For example, in Mozilla General there are 147 non-severe indicators in the top-200 indicator list of averaged 10-fold scores of IG, and 161 non-severe indicators for CHI. Mozilla Layout also has the similar situation: 128 non-severe indicators for IG and 168 non-severe indicators for CHI. Therefore, using the equal numbers of severe indicators and non-severe indicators in CC will get poor prediction performance. Although employing non-severe indicators can get improvements in Mozilla, the improvements are very minor. This is because although we can extract many informative non-severe indicators, the number of informative severe indicators is not as large as the number of severe defect reports.

The prediction results demonstrate that the reports of Eclipse tend to use more severity-consistent words to describe the defects such that the indicators have high specificity. In



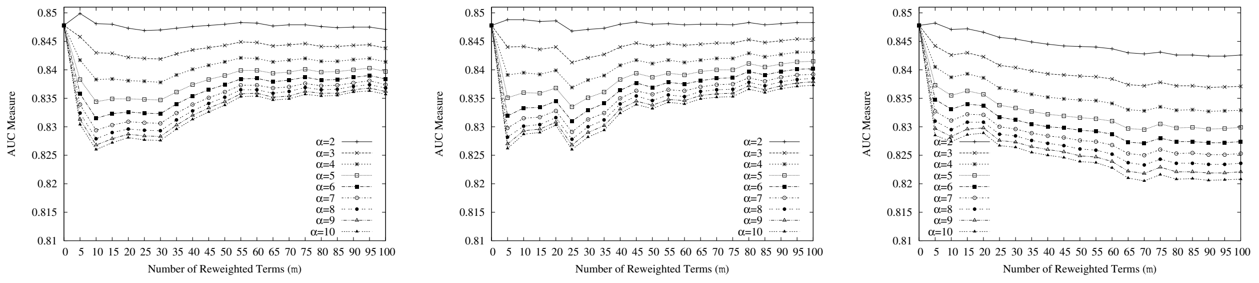
(a) IG (b) CHI (c) CC

Fig. 5. The AUC measures for Eclipse JDT-UI using the indicators extracted from the severe defect reports to reweight the training models only.



(a) IG (b) CHI (c) CC

Fig. 6. The AUC measures for Eclipse UI using the indicators extracted from the severe defect reports to reweight the training models only.



(a) IG (b) CHI (c) CC

Fig. 7. The AUC measures for Mozilla General using the indicators extracted from all defect reports to reweight the training models and the testing defect reports.

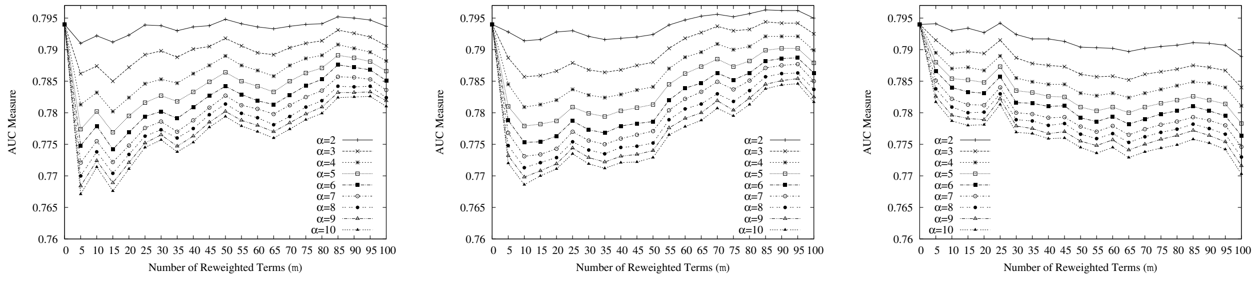
contrast, the reporters of Mozilla tend to use less severity-consistent words to describe the defects. The possible reason is that the reporters of Mozilla may have diversified background for the software development. Therefore, the feature selection schemes can find only few informative terms as good severe indicators in Mozilla.

VI. Threats to Validity

Although this study shows that feature selection can be used to improve the severity prediction, there are some threats to the validity of this study. First, since this study is based on the research work in [2], [3], it faces the same threats as reported in [2], [3]. Therefore, we avoid using latest defect reports and assume that few reports in the datasets are reassigned to other components for the construct validity problem. Since the number of defect reports in each component in our experiments is comparatively large, the risk is mitigated.

For the internal validity, this study additionally faces a threat that the feature selection schemes used in this study do not consider the semantic relationship. Therefore, our current feature selection model has the risk of extracting some words that are irrelevant to severity. For example, in Eclipse UI, "editor" appears in the severe indicator list of CC. In addition, negative sentences are not considered. More advanced natural language processing techniques can be employed to mitigate this threat.

For the external validity, the improvement of feature selection may not be guaranteed if the nature of the software projects is very different with the nature of the projects discussed in this study. In addition, some adaptations may be needed for software projects of different defect tracking systems.



(a) IG (b) CHI (c) CC

Fig. 8. The AUC measures for Mozilla General using the indicators extracted from all defect reports to reweight the training models and the testing defect reports.

VII. Conclusions

In software maintenance, severity prediction on defect reports is an emerging issue for mitigating the considerable triaging cost. In the past research work, several text mining approaches have been proposed to obtain accurate predictions. Although these approaches demonstrate the effectiveness of predicting the severity, they do not discuss the problem of how to find the indicators in good quality.

In this paper, we study three common feature selection schemes, Information Gain (IG), χ^2 (CHI), and Correlation Coefficient (CC), based on the Multinomial Naive Bayes (NB) classification approach. In the study, these feature selection schemes show their effectiveness to extract potential severe and non-severe indicators and thus improve the prediction performance in over half the cases.

Based on the observations of this study, there are still many issues to be discussed in our future plan. First, the semantic relationship is an important problem to decide the implicit severity of a defect report. For this issue, some advanced natural language processing techniques need to be employed. Also, there are other feature selection schemes proposed in past research. A comprehensive study needs to be conducted to examine the effectiveness of different feature selection schemes. Different learning models with feature selection schemes are also in our future study plan to see whether there is a better learning model that can achieve higher prediction performance.

Acknowledgment

This work is supported in part by the National Science Council, Taiwan under Grant NSC 101-2221-E-155-011-MY2. The authors would also like to express their sincere thanks to anonymous reviewers for their precious comments.

References

[1] J. Anvik, L. Hiew, and G. C. Murphy, "Who Should Fix this Bug?" in *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. New York, NY, USA: ACM, 2006, pp. 361–370.

[2] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the Severity of a Reported Bug," in *Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, 2010, pp. 1–10.

[3] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug," in *Proceedings of the 15th European Conference on Software Maintenance and Reengineering (CSMR 2011)*, 2011, pp. 249–258.

[4] I. Herraiz, D. M. German, J. M. Gonzalez-Barahona, and G. Robles, "Towards a Simplification of the Bug Report Form in Eclipse," in *Proceedings of the 2008 International Working Conference on Mining Software Repositories (MSR '08)*, 2008, pp. 145–148.

[5] T. Menzies and A. Marcus, "Automated Severity Assessment of Software Defect Reports," in *Proceedings of the 24th IEEE International Conference on Software Maintenance (ICSM 2008)*, 2008, pp. 346–355.

[6] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, Mar. 2002.

[7] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97)*, 1997, pp. 412–420.

[8] H. T. Ng, W. B. Goh, and K. L. Low, "Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization," in *Proceedings of the 20th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '97. New York, NY, USA: ACM, 1997, pp. 67–73. [Online]. Available: http://doi.acm.org/10.1145/258525.258537

[9] Z. Zheng, X. Wu, and R. Srihari, "Feature Selection for Text Categorization on Imbalanced Data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 80–89, June 2004. [Online]. Available: http://doi.acm.org/10.1145/1007730.1007741

[10] Y.-H. Tseng, C.-J. Lin, H.-H. Chen, and Y.-I. Lin, "Toward Generic Title Generation for Clustered Documents," in *Proceedings of the Third Asia Information Retrieval Symposium (AIRS 2006)*, 2006, pp. 145–157.

[11] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., May 1993.

[12] S. Shivaji, E. J. Whitehead, Jr., R. Akella, and S. Kim, "Reducing Features to Improve Bug Prediction," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (ASE '09)*, 2009, pp. 600–604.

[13] H. Wang, T. M. Khoshgoftaar, and R. Wald, "Measuring Robustness of Feature Selection Techniques on Software Engineering Datasets," in *Proceedings of the 2011 IEEE International Conference on Information Reuse and Integration (IRI 2011)*, 2011, pp. 309–314.

[14] T. Dunning, "Accurate Methods for the Statistics of Surprise and Coincidence," *Computational Linguistics*, vol. 19, no. 1, pp. 61–74, Mar. 1993.

[15] M. Wurst. Word Vector Tool. [Online]. Available: http://sourceforge.net/projects/wvtool/

[16] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," in *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, 1998, pp. 41–48.

[17] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, "Multinomial Naive Bayes for Text Categorization Revisited," in *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, 2004, pp. 488–499.