6B-4

Bugzilla 等へ報告される Issue の分類手法の提案

板垣　恭太†　　海尻　賢二†　　海谷　治彦†　　小形　真平†

信州大学大学院工学系研究科†

1. はじめに

大規模なソフトウェアの開発および保守では，Issue Tracking System(ITS)が利用されている．ITS では開発者だけでなくユーザも報告者となり，issue の報告を行うことができる．その際，優先度や重要度も設定でき，開発者はこれらをもとに処理をする順番を決めていく．しかし，これらの項目は始めに報告者が設定するため，報告時は不適切な場合もある．実際に本研究で対象となるプロジェクトに報告された issue の中で報告時から重要度が変更されたものは 33815 個あった．そのため，開発者は全てのレポートに目を通し重要度の精査を行わなければならず，その結果，本当に早急に対処すべき問題への対応が遅くなる可能性が出てしまう．

そこで本研究では報告された issue がバグと Enhancement のどちらであるかを自動で分別する手法の提案を行う．この分類は一般的にバグと Enhancement ではバグの方が緊急性が高いという考えに基づいている．これにより，緊急度の高い問題への対応し遅れを防ぐことが可能になると同時に，開発者が優先度を精査する必要がなくなり時間や労力の浪費も防ぐことができる．本論文では，提案する手法の説明とそれを用いた実験およびその結果について述べる．

2. 提案手法

多くのプロジェクトで利用されている ITS の 1 つに Bugzilla がある．Bugzilla には Severity という重要度に関する項目があり，これはバグの緊急度および再現度による 5 種類に Enhancement を加えた計 6 種類から選択し設定することができる．本研究では，バグについての 5 種類をひとまとめにし，6種類の値をバグ，Enhancement の 2 つのカテゴリにして扱う．また，これらの 2 種類のみにした場合，分類結果の確実性が低い場合にも必ずどちらかに振り分けなければならなくなり，全体的な精度の低下を招く原因となる．そこで，本研究ではバグ，Enhancement の他に分類不可というカテゴリを設け，分類結果の確実性が低いとされた issue はそこに分類する．これにより，全体的な精度を損なうことなく分類を行うことが可能となる．

上述の 3 つのカテゴリに分類するために本研究では Bugzilla におけるテーブル"longdescs"のフィールド"thetext"の項目を利用する．この項目はバグについての詳細な記述を行う部分でテキスト形式での入力が可能である．そこでここに書かれた文章を分析することにより各々の issue の分類を行う．具体的には文章を単語単位で区切り，Feature Selection[1]を利用して単語を選別し，それらの出現頻度を利用して分類を試みる．本研究では分類方法について以下の 2 種類のアプローチを考え，どちらがより高精度な分類が可能かを調べる．1 つは純粋に単語の出現数により分類を行う方法，もう 1 つは Weka により分類器を生成しそれを使用して分類を行う方法である．今回はこれらのアプローチについて，いくつかの実験を行った．

3. 実験

本研究では対象プロジェクトとして Eclipse を選択した．対象データは MINING SOFTWARE REPOSITORIES 2011 によって提供されているデータを使用する．これらのデータから各 Issue の詳細記述を抽出し利用する．また，今回はどちらの実験でも 2 分類と 3 分類の場合を考える．

実験 1 では特徴的な単語の出現数の差を用い分類を行う．実験は以下の手順によって行う．

(1)テーブル"bugs"の"bug_severity"フィールドの値により全ての issue をバグと Enhancement に分ける．

(2)次に各カテゴリから 1000 個ずつランダムに合計 2000 個の issue レポートを選び出し，単語の抽出元とする．

(3)各レポートの詳細記述部を単語単位で区切り，そこから Feature Selection によりカテゴリを特徴付ける単語を選ぶ．

(4)それらの単語がテスト対象となる各々のレポートにどれだけ出現するかを数え上げる．

Proposal for the method of classification of issues reported to Bugzilla

† Kyota Itagaki, Kenji Kaijiri, Haruhiko Kaiya, Shinpei Ogata (Shinshu University)

(5)カテゴリごとに単語の出現数を足しあわせて両カテゴリ間で値を比較し，値が大きい方のカテゴリへ対象となるレポートを分類する．

(4)について，今回はテスト対象を(a)単語抽出元と同じレポートとする場合と(b)各カテゴリから新たに 1000 個ずつランダムに選び出された合計 2000 個のレポートとする場合の 2 通り考える．2 つの場合の精度を比較することで抽出された単語がどの程度一般性を持っているかを調べることができる． 2 分類の場合は単純に値の大小により分類するが，3 分類の場合は 2 つの値の差の大きさに閾値を設け，閾値よりも差が小さい場合は分類不可として扱う．

次に実験 2 では Weka を用いて分類器を生成し，それを利用した分類を行う．途中までは実験 1 と同じ手順であるが，こちらの実験では(4)の後，それらのデータを元に arff ファイルを作成し学習を行い，分類器を生成する．テストに使用する arff ファイルも同様に作成し，分類器にかけることにより分類を行う．3 分類の場合は分類を行った際に得られる probability の値に閾値を設け，この値を下回った時は分類不可として扱うようにする．以上の手順で実験 2 を行うが，Weka を使用する場合，適用するアルゴリズムによって分類器の性能が変わるため，アルゴリズムの選択が重要となる．そこで今回は使用するアルゴリズムを決めるため，各アルゴリズムを使い，3 分類の(b)の単語抽出元レポート数とテスト用レポート数をどちらも 500 個ずつにした場合で実験を行い，最も精度が高いアルゴリズムを選択する方式を採る．

4. 評価

本実験では評価尺度として分類精度を扱う．精度には様々な見方があるが，今回はバグと Enhancement のそれぞれの正答数を足しあわせたものをテストレポートの総数で割り，値を導くというものを採用する．例えば，バグの正答数を A，Enhancement の正答数を B とした場合，(A+B)/2000 で示される．これは 2 分類の場合であるが，始めに述べたように本研究では分類の精度を高めるため 3 分類の場合も扱っている．そこで 3 分類で分類不可に振り分けられたものの数を C とした時，その精度は(A+B)/(2000-C)で表す．以降ではこの定義によって精度を求め，各実験の結果について考察していく．

実験 1 について見てみると表 1 により(a)，(b)どちらも 2 分類より 3 分類の方が，精度が 0.05 から 0.1 ほど上がっていることが分かる．

次に実験 2 について見てみる．今回は使用するアルゴリズムの候補として J48，Naive Bayes，Logistic の 3 つを選んだ．始めにアルゴリズムの精度の比較を行った結果，J48 は 0.379，Naive Bayes は 0.219，Logistic は 0.425 となり，Logistic を使用した場合が最も精度が良くなった．そこで最も精度の良かった Logistic を使用してそれぞれの実験を行った．表 1 を見ると，実験 2 では(a)と(b)のどちらの条件でも 3 分類の場合より 2 分類の場合の方が精度は高くなった．これは分類自体は適切にできているが probability の値が低いケースが 1000 個近くあり，3 分類にした場合，それらが全て分類不可に割り当てられてしまったためだと考えられる．これを改善するために今後は閾値の見直し等を行なっていく必要がある．

最後に実験 1 と 2 を合わせて見てみる．まず，(a)と(b)を比較すると，両実験とも 2 分類と 3 分類のどちらの場合も(a)に比べて(b)の精度が下がっていることから抽出した単語にはあまり一般性がないことが分かる．また，精度を比較するため，(b)の値を見ると，実験 2 の 3 分類を除いてあまり差がないことから，現状では実験 1 で用いた手法の方が安定していると言える．

表 1. 各条件での実験における分類精度

		(a)	(b)
実験 1 (閾値=3)	2 分類	0.651	0.575
	3 分類	0.740	0.629
実験 2 (Logistic, 閾値=0.6)	2 分類	0.891	0.614
	3 分類	0.592	0.431

5. 終わりに

本論文では ITS を通して報告された issue がバグか Enhancement かを自動で分類するための手法を提案した．これにより，開発者の負担軽減が可能になると同時に，開発者は優先度の高い問題から処理を行うことができるようになる．

現在は分類精度があまり高くないため，今後は閾値の見直し等を行い，精度の向上を目指していきたい．

参考文献

[1] Weiqin Zou, et al: Towards Training Set Reduction for Bug Triage, COMPSAC 2011