

Who Should Fix This Bug?

John Anvik
Lyndon Hiew
Gail C. Murphy

ICES`06 Proceedings of the 28th international conference on Software engineering
Pages 361~370

背景

- OSS開発ではopen bug repositoryを組み込んでいる
- 報告されたバグが重大な問題または改善がありもしそれを解消するなら報告されたバグは適切な開発者に割り当てなければならない
- バグを解消するのに適切な開発者を推薦するために機械学習アルゴリズムを用いる

背景

- open bug repository (issuetracking system)
 - Bugzilla,GNATS,JIRA
 - 今回はBugzillaを使う
- The Life-Cycle of a Bug Report
 - バグの動きを図に表わしたもの

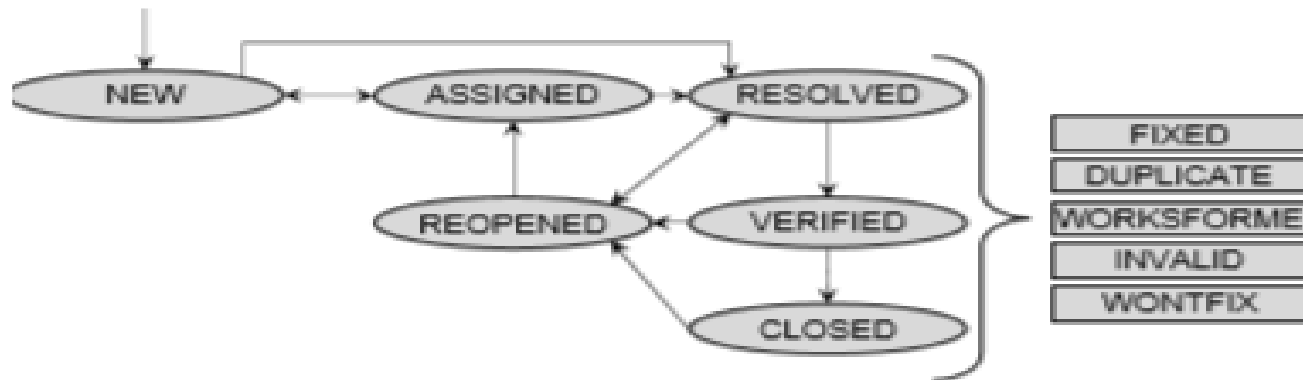


Figure 1: The life-cycle of an Eclipse bug report.

背景

- triageとは
 - 重大なバグ報告を決める
 - 開発者にそのバグ報告の負担(責任)を割り当てる
 - triager
 - triageする人のこと
 - resolver
 - 割り当てられた人のこと

背景

- Machine Learning: text categorization
 - テキスト分類: テキストドキュメントをカテゴリーに分けること
 - Instances: documents
 - Instanceはlabel(category)を持っているのかも
 - Training the classifier
 - Instanceと知られているlabelを入力として機械学習アルゴリズムにいれてclassifier(分類指標)を作ること

A SEMI-AUTOMATED APPROACH TO BUG ASSIGNMENT

- バグ割当ては4stepから成る
 - Step1:バグ報告に特徴づけをする
 - Step2:それぞれの報告にラベルを割り当てる
 - Step3:それらの報告を選んでいく
 - Step4:割当て推薦するための分類指標を作るためにアルゴリズムを用いる
- Eclipse(8655件)とFirefox(9752件)のバグ報告について行う(2004/7/1~2005/3/31)
- Precision(適合率)とRecall(再現率)を用いる

Precisionとrecall

$$\text{Precision} = \frac{\# \text{ of appropriate recommendations}}{\# \text{ of recommendations made}} \quad (1)$$

$$\text{Recall} = \frac{\# \text{ of appropriate recommendations}}{\# \text{ of possibly relevant developers}} \quad (2)$$

適合率: 正確性を示す。

検索範囲は狭いが正確性を上げる

再現率: 網羅性を示す。

検索範囲を広くしてできるだけ正しいものすべて拾ってくる

STEP1

- バグが過去のどのバグと似ているかを判断するために特徴づけをする
- 特徴づけするために...
 - Free-form text を特徴ベクトルへ
 - 機械学習アルゴリズムを用いる

STEP2

- バグ報告にラベルを割り当てる
- assigned-toの値は不適切であった
- Heuristics to label the reports
 - 報告にラベルをつける
- 4つの規定を設ける
 - FIXEDになっているとき(Firefox)
 - 最後に承認されたパッチを提出した人
 - FIXEDになっているとき(Eclipse)
 - 報告に解決されたものとした人
 - DUPLICATEとなっているとき(Firefox,Eclipse)
 - 同じ報告があったとして解決した人
 - WORKSFORMEとなっているとき(Firefox)
 - 分類不可

STEP3

- まず、役に立たないラベルを取り除く
- 次に開発者についても選択していく
 - 活動していない開発者に割り当てても意味がない
 - 3か月で9件以上のバグを解消している開発者を選択

Table 2: The effect of developer profile filtering on recommender accuracy and recall.

		# Dev.		Precision/Recall (%)	
		Firefox	Eclipse	Firefox	Eclipse
No Profile		414	146	23/1	58/7
>1 Fix in 3 mo.		94	82	59/2	57/7
Avg. Fixes Per Month Over 3 mo.	1	66	50	64/2	57/7
	2	33	42	59/2	57/7
	3	26	40	64/2	57/7
	4	21	40	64/2	58/7
	5	18	39	59/2	59/7
	6	13	37	45/1	57/7

STEP4

- Naïve Bayes
 - 確率的分類アルゴリズム
 - 以前、自動バグ割り当てに使われた
- SVM
 - 非線形の分類アルゴリズム
 - テキスト分類に対して効果的である
- C4.5
 - 決定木を生成するためのアルゴリズム
 - 人間のtriagerが決定木を走査する方法に似ている

Table 3: The effect of different machine learning algorithms on recommender accuracy and recall.

Predictions	Naïve Bayes		SVM		C4.5	
	Firefox	Eclipse	Firefox	Eclipse	Firefox	Eclipse
1	59/2	54/6	64/2	58/7	64/2	40/5
2	59/2	49/11	52/3	52/13	41/3	34/9
3	59/2	44/15	57/6	47/16	42/5	31/12

- SVMを今回は使う

VALIDATING WITH GCC

- gccコンパイラプロジェクトで使ってみる
- 分析対象のデータ
 - 194件 {元は2629件 (2004/9～2005/4) }

Table 4: Precision and recall for gcc.

Predictions	Precision/Recall
1	6/0.3
2	18/2
3	18/3

- 原因は...
 - プロジェクトに合わなかった
 - 正確に測定するのは難しい

DISCUSSION

- 50%をこえた適合率で本当に良いのか
 - 従来のtriagerの手にゆだねた方法と比べると根拠を満たしている
 - 再割り当てを行う数を減らすことができ、triageに費やす時間を短縮できる
- 再現率は何故低いのか
 - 適合率と再現率はトレードオフ

Table 5: The size of developer groups per project.

	Min	Mean	Max
Eclipse	1	10	23
Firefox	10	30	53
gcc	2	12	34

- プロジェクトの開発者グループ一つの人数が多い

まとめ

- Bug repositoryの情報を適用した機械学習アルゴリズムを用いた
- Eclipse, Firefoxでは適合率が50%を超えた
 - しかしながら、gccでは6%のものもあった
- 今後の計画
 - 決定のための規範的な手段やさらなる情報源の調査やオープンソースシステムも対象にする

SVM

- サポートベクターマシン
- 教師あり学習を用いるパターン認識モデルの一つである