卒業研究報告書

題目　テキスト分類によるバグレポートの

分類手法の提案

指導教員　水野　修　准教授

京都工芸繊維大学　工芸科学部　情報工学課程

学生番号　11122502

氏名　NAIDANJAV　ZOLBAYAR

平成 27 年 2 月 13 日提出

テキスト分類によるバグレポートの分類手法の提案

平成 27 年 2 月 13 日　　11122502 NAIDANJAV　ZOLBAYAR

概　　要

オープンソースソフトウェア開発においてバグトラッキングシステム(Bug tracking system, BTS)は障害の報告や修正，管理に欠かせない仕組みとなっている．バグトラッキングシステムにはユーザからの報告が寄せられ，その報告が「バグ」についての報告なのか,「機能拡張」への報告なのかが，自己申告で分類として提示される．しかし，この自己申告のバグ報告は誤っていることがしばしばあり，そのバグだと思って修正を始めると実は機能拡張であったり，機能拡張と思ってしばらく後回しにしていると実はバグだったりする．こうした誤申告はソフトウェア開発の生産性に大きな影響を与える．そのため，ユーザからの報告の時点でその報告が実際には何に該当するのかを知るのは重要な課題となる．

Antoniol らは，ユーザから寄せられた状況を記した記述（以降，状況記述と呼ぶ）をテキスト分析することで申告を自動的に振り分ける手法を提案している．彼らの手法ではおよそ 66～82%の精度でバグと機能拡張を振り分けることが可能とされている．一方，本研究室ではスパムフィルタを用いたバグの分類手法を提案しており，ここで用いる汎用のスパムフィルタを利用すれば，さらなる精度を達成できると考えた．そこで，本研究ではスパムフィルタを利用したバグ報告の自動分類手法を提案し，数種類のオープンソースソフトウェアに対して適用する．提案する朱方では Antoniol らの最も良い精度の結果とほぼ同様の値が得られることが分かった．

目次

1. 緒言

近年, ソフトウェアの開発においてはバグの修正が重要な作業と考えられている．多くのソフトウェアプロジェクトは，バグ管理システムを使用してバグを管理している．バグ管理システムはバグに関する問題を管理するために使用されるべきである．しかし，システムの改善や追加機能の要求などの他のソフトウェア活動にもよく使用されている．

オープンソースソフトウェア開発においてバグトラッキングシステム(Bugtracking system, BTS)は障害の報告や修正，管理に欠かせない仕組みとなっている．バグトラッキングシステムにはユーザからの報告が寄せられ，その報告が「バグ」についての報告なのか,「機能拡張」への報告なのかが，自己申告で分類として提示される．しかし，この自己申告のバグ報告は誤っていることがしばしばあり，そのバグだと思って修正を始めると実は機能拡張であったり，機能拡張と思ってしばらく後回しにしていると実はバグだったりする．こうした誤申告はソフトウェア開発の生産性に大きな影響を与える．そのため，ユーザからの報告の時点でその報告が実際には何に該当するのかを知るのは重要な課題となる．

本研究の目的は，こうして報告されたバグレポートを「バグ報告」と「機能拡張」の 2 つに自動分類するシステムの開発である．先行研究で，Antoniol らは，ユーザから寄せられた状況を記した記述（以降，状況記述と呼ぶ）をテキスト分析することで申告を自動的に振り分ける手法を提案している[1]. 彼らの手法では状況記述のテキスト中から重要語を抽出し, その出現を naïve Bayes や分類木などの機械学習器で学習させ，分類を行っている．その結果，およそ 46〜82%の精度でバグ報告と機能拡張を振り分けることが可能と結論づけている．また，grep を用いた振り分けに対する優位性を主張している．

本研究では，バグ報告と機能拡張の振り分けを行うためであれば，一般に利用されているツールを使ったフィルタを作成することでもっと簡単に振り分けができるのではないかと考えた．本研究室ではスパムフィルタを用いた不具合(バグ)の分類手法を提案しており[2-4]，ここで用いるフィルタを利用すれば，さらなる精度を達成できると考えた．そこで，本研究ではスパムフィルタを利用したバグ報告の自動

分類手法を提案し,数種類のオープンソースソフトウェアに対して適用する.また,Antoniol らは交差検証での実験しか実施していないが，本研究では実際の利用局面を考慮できる Training Only Errors(TOE)手法での実験も実施した．その結果，提案する手法では Antoniol らの手法と同様の実験を行ったところでは，彼らの最も良い精度の結果とほぼ同様の値が得られることが分かった.

本報告の以降の構成は次の通りである．2 章では研究の目的と研究設問を述べる．3 章では実験の準備としての理論や手法について説明する.4 章では実験内容について述べる．5 章では実験の結果について述べる．6 章では実験結果から得られる考察を行い，7 章で本報告のまとめと今後の課題を述べる.

2. 研究の目的

2.1. 研究の目的

本研究の目的は，報告されたバグレポートを「バグ報告」と「機能拡張」の 2 つに自動分類するシステムの開発である.

2.2. 研究設問

本研究では次に示す研究設問（Research Question）について，検証を行う.

RQ1：スパムフィルタを用いたバグの分類手法でどの精度までバグ報告と機能拡張を振り分けることができるか？

RQ2：スパムフィルタを用いたバグ報告の分類手法と Antoniol らの手法ではどちらが良い精度を持つのか？

RQ3：提案手法における特徴にはどのようなものがあるか.

以上の研究設問に対する答えを与えるため，本研究ではバグ報告の分類システムを実装した．また，オープンソースのソフトウェア開発プロジェクトよりいくつかのプロジェクトのデータを抽出し，バグ報告を分類する実験を行った.

3. 準備

3.1.バグ管理システム

近年，ソフトウェアプロジェクトの開発においてはバグの修正が重要な作業と考えられている．バグを登録し，修正状況を追跡するシステムがバグ管理システムと呼ばれる．バグ管理システムは，バグの発見日時や発見者，修正担当者，修正履歴，修正方法，重要度などの多くの情報を管理する．

現在，いくつかのバグ管理システムの実装が存在する．なかでも，Eclipse や Mozilla，JBoss，LibreOffice など大きなオープンソースプロジェクトに利用されているため Bugzilla と Jira が最も人気なバグ管理システムである．Bugzilla と Jira の両方ともウェブサーバ上で動作し，ウェブブラウザ経由でアクセスできる．Bugzilla を使用しているプロジェクトはいつでも Jira へ移行できるようになっており，その判断は各プロジェクトに委ねられている．

以下にバグ管理システムの代表として Jira と BugZilla のそれぞれの特徴を説明する．

- **BugZilla**

  Bugzilla は，Mozilla Foundation に開発されたオープンソースバグ管理システムである．極めて初期のバージョンは Tcl で記述されていたが，オープンソースになってからのソースコードは，Perl で記述されている．現在では，オープンソース，プロプライエタリ問わず，数百のプロジェクトでバグ管理ツールとして選択されている．

- **Jira**

  Jira は，Atlassian に開発された企業向けプロプライエタリバグ管理システムであるが，オープンソースプロジェクトに対して無償提供されている．Jira が RSS に対応するため，XML パーサを有する任意な RSS リーダーがデータを抽出できるようになっている．Jira は Java 言語で開発されている．

3.2. テキスト分類フィルタ: CRM114

本研究では，テキスト分類フィルタとして CRM114 を用いた[5]． CRM114（CRM114 Discriminator）とは，テキスト分類フィルタを作成するためのプログラミング言語である．主にスパムフィルタとして開発されおり，メールフィルタの中でも高い予測精度をあげているものの 1 つである．

CRM114 は基本的にはベース識別を利用したテキスト分類フィルタであるが，複数の単語を組み合わせたものをトークンと呼び，学習・分類の単位として利用することが大きな特徴である．従来のテキスト分類フィルタは 1 単語をトークンとしているのに対し，複数単語の組をトークンとすることで，より複雑な学習が可能となっている．

本研究では，CRM114 のデフォルトの分類手法である OSB (Orthogonal Sparse Bigram)を利用する．OSB は任意の連続する 5 単語の組み合わせのうち，2 単語からなるものだけをトークンとする手法である．OSB によるテキスト処理について以下に簡単に示す．

“Houston, we've got a problem”という文章を例として説明する．余白で単語を区切るため，5 つの単語に対して処理を行い，“Houston we've”，“Houston got”，“Houston a”，“Houston problem”の 4 つのトークンが生成される．その 4 つのトークンを学習と分類の対象として抽出する．

3.3. 従来手法

従来手法として，Antoniol らは，ユーザから寄せられた状況記述をテキスト分析することで申告を自動的に振り分ける手法を提案している[1]. 彼らの手法では状況記述のテキスト中から重要語を抽出する. 例えば, “crash”, “critic”, “broken”, “when”などの単語がバグ情報によく使われ，“should”，“implemet”，“support”などの単語が機能拡張によく使われる．また，単語が使われたフィールドによって重要度が異なることも考えている．

その出現を naïve Bayes，分類木とロジスティック回帰の機械学習器で学習させ，分類を行っている．

実験対象として，Mozilla，Eclipse と JBoss の 3 つのオープンソースプロジェクト

を利用している．各プロジェクトから 600 個のバグレポートを任意に選択し，3 人のソフトウェアエンジニアが単純な多数決を使用して，手動分類した．そして，手動分類を行ったデータを用い，十重交差検証により評価を行った．

その結果，およそ 46〜82%の精度でバグ報告と機能拡張を振り分けることが可能と結論づけている．また，grep を用いた振り分けに対する優位性を主張している．

4. 実験

4.1. 実験対象

本研究の実験対象として，Apache ソフトウェア財団に開発されている 5 つのオープンソースプロジェクトを利用する[6]．以下に実験対象としたプロジェクトのそれぞれの特徴を説明する．

- Cayenne

  Java – O/R マッピングツール（object-relational mapping tool）で，Apache Software Foundation で開発／配布が行われている．Cayenne は付属の GUI ツール(CayenneModeler)を用いてテーブ構造の設定から，クラスファイルの作成，データベースへのテーブル作成まで行う．

- MINA

  MINA は Multipurpose Infrastructure for Network Applications の略称であり，Java 言語を用いて開発されているオープンソース•ネットワークアプリケーションフレームワークである．

- James

  James は Java Apache Mail Enterprise Server の略称であり，Apache プロジェクト内の電子メールアプリケーションサーバを開発するサブプロジェクトである．

- Lucene

  あらかじめ蓄積した大量のデータから，指定したキーワードを探し出す機能を持つ全文検索ソフトウェアである．Java のクラスライブラリとして提供される．

•OpenJPA

  OpenJPA は，データベースへのオブジェクトの永続化を単純化するオープンソース実装である．

実験対象としたプロジェクトは全て Java で開発されており，バグが Jira に保存されている．実験対象となる各プロジェクトのバグレポートのバグ数と機能拡張数を表 4.1 に示す．Cayenne，Lucene，OpenJPA のバグ数と機能拡張数が多いため，それ

ぞれのバグの 800 個と機能拡張の 800 個のデータを任意に選択した．なお，800 個以上のデータを使用して実験を行っても，以降で述べる結果にはほとんど影響がなかった．

バグ管理システムからは，バグの発見日時や発見者，修正担当者，修正履歴，内容などの多くの情報をが取得できる．本実験では，分類の対象となるテキスト情報として「バグレポートの内容」と「各レポートに付随するコメント」を用いた．

4.2. 自動分類手法

本研究では，自動分類手法として以下に述べる 3 つの実験を行った．

実験 1 と実験 2 は十重交差検証を用いた実験を行い，その結果として得られる分類の精度を評価する．十重交差検証では，各データを 10 個に分割する．そして，そのうちの 1 つをテストデータとし，残る 9 つを学習データとする．そして，10 個に分割されたデータそれぞれをテストデータとして 10 回検証を行う．そうして得られた 10 回の結果を平均して評価する．各実験の特徴と手順を次に示す．

表 4.1　実験対象

Systems	バグ報告	機能拡張
Cayenne	837	839
James	607	712
Lucene	1659	2209
MINA	439	420
OpenJPA	1338	818

4.2.1. 実験 1

実験 1 では，各プロジェクトごとに TET（Train Every Thing）手法で学習し，十重交差検証を用いて評価する. TET とは, 全ての学習データを学習する手法である. 具体的には次の手順で実験を行った.

1. 全てのバグレポートを「実際にバグに値するかしないか」によってバグ報告と機能拡張に分割する. バグ報告と機能拡張のデータ両方を 10 個に分割し, 十重交差検証を行う.
2. 全ての学習データをバグ報告と機能拡張データの 2 つに分けて学習し, バグ報告コーパスと機能拡張コーパスを作成する.
3. テストデータを作成されたコーパスを用いて分類を行う.
4. クロスバリデーションで得られた 10 回の結果をまとめ, 評価値を計算する.

4.2.2. 実験 2

実験 2 では，各プロジェクトごとに TOE（Train Only Errors）手法で学習し，十重交差検証を用いて評価する．TOE とは，学習データを分類した分類が誤っている場合のみ，このデータを学習する手法である．具体的には次の手順で実験を行った.

1. 全てのバグレポートを「実際にバグに値するかしないか」によってバグ報告と機能拡張に分割する.
2. 全てのデータを時系列順にソートし，古いデータより順に分類を行う．その際，あるデータの分類はその時点での学習コーパスのみを用いて行う．そして，分類した結果が誤っている場合のみ，そのデータをコーパスに学習させる．つまり，実際にはバグ報告であり，分類が機能拡張である場合，そのテキストデータをバグ報告コーパスに学習する. 逆に, 実際に機能拡張であり, 分類がバグ報告であった場合，機能拡張コーパスに学習する．分類が正しかった場合は学習は行わない.

3. 全てのデータを分類し終わった時点で，分類の精度を評価する．

4.2.3. 実験 3

実験 3 では，各プロジェクトに対して，他の全てのプロジェクトを学習データとして分類を行う．ただし，Lucene プロジェクトのデータが学習に向いてないことが実験 1, 2 の結果より判明したため， Lucene を学習には使用しなかった．例えば，MINA プロジェクトを分類するためには，Cayenne，James，OpenJPA の 3 プロジェクトのデータを学習させた．具体的には次の手順で実験を行った．

1. 分類を行うプロジェクトと Lucene の以外のプロジェクトの各データを学習し，バグ報告コーパスと機能拡張コーパスを作成する．
2. 作成されたコーパスを用いて分類を行う．
3. 得られた結果をまとめ，評価値を計算する．

5. 実験結果

5.1. 結果の凡例

本実験において得られる結果は表 5.1 のように得られる．得られる結果の意味は次の通りである．

- True Positive（TP）: TP とは実際にバグ報告であったものをバグ報告と分類したものを指す.
- False Positive（FP）: FP とは実際に機能拡張であったものをバグ報告と分類したものを指す.
- False Negative（FN）: FN とは実際にバグ報告であったものを機能拡張と分類したものを指す.
- True Negative（TN）: TN とは実際に機能拡張であったものを機能拡張と分類したものを指す.

5.2. 評価値

実験結果の評価値としては正解率（Accuracy），再現率（Recall），適合率（Precision），F_1 値の 4 つを用いた．各評価値の概要及び計算式を以下に示す.

（1）正解率（Accuracy）

正解率（Accuracy）は，分類結果全体に対して正解している割合を示す．本実験では，実際にバグ報告であったものをバグ報告，機能拡張であったものを機能拡張と分類できた割合を指す．これを表 5.1 の凡例を用いると，式（5.1）のように定義できる.

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{5.1}$$

正解率は分類全体の傾向を容易にわかるので便利な指標である．しかし，対象データが，負のデータに対して，正のデータが比較的に多い場合などのデータの偏りの影響を受けやすいため，正解率以外にも，次の再現率，適合率，という指標を用いる.

表 5.1　実験結果の凡例

		分類	
		バグ報告	機能拡張
実測	バグ報告	TP	FN
	機能拡張	FP	TN

（2）再現率（Recall）

再現率（Recall）は，実際に正である中で，分類できた割合を示す．本実験では，実際にバグ報告であったものをバグ報告と分類できた割合を指す．これを表 5.1 の凡例を用いると，式（5.2）のように定義できる．

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

バグレポートの分類において，再現率はバグ報告の誤った分類を防ぐ視点から，重要な指標と言える．

（3）適合率（Precision）

適合率（Precision）は，正と分類した中で，対象が実際に正だった割合を示す．本実験では，バグレポートをバグ報告と分類した中で，実際にバグ報告であった割合を指す．これを表 5.1 の凡例を用いると，式（5.3）のように定義できる．

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

バグレポートの分類において，適合率は 1 個のバグ報告を正しく分類するために必要なコストを表す．

（4）F_1 値

再現率と適合率はトレードオフ関係がある．F_1 値は，再現率と適合率の総合的な評価を表す．これを表 5.1 の凡例を用いると，式（5.4）のように定義できる．

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.4}$$

ただし，F_1 値においても実験対象となるデータの偏りの影響を受けることがある．

5.3.実験の結果

5.3.1. Cayenne

表 5.1，5.6，5.11 にそれぞれ実験 1，実験 2，実験 3 を行った結果を示す．また，表 5.1，5.6，5.11 の結果から各評価値を計算した結果を表 5.16 に示す．

5.3.2. MINA

表 5.2，5.7，5.12 にそれぞれ実験 1，実験 2，実験 3 を行った結果を示す．また，表 5.2，5.7，5.12 の結果から各評価値を計算した結果を表 5.17 に示す．

5.3.3. James

表 5.3，5.8，5.13 にそれぞれ実験 1，実験 2，実験 3 を行った結果を示す．また，表 5.3，5.8，5.13 の結果から各評価値を計算した結果を表 5.18 に示す．

5.3.4. Lucene

表 5.4，5.9，5.14 にそれぞれ実験 1，実験 2，実験 3 を行った結果を示す．また，表 5.4，5.9，5.14 の結果から各評価値を計算した結果を表 5.19 に示す．

5.3.5. OpenJPA

表 5.5，5.10，5.15 にそれぞれ実験 1，実験 2，実験 3 を行った結果を示す．また，表 5.5，5.10，5.15 の結果から各評価値を計算した結果を表 5.20 に示す．

5.3.6. 総合結果

表 5.16～5.20 に示した各プロジェクトの評価値から平均値を計算した結果を表 5.21 に示す．この結果は各実験の総合結果となる．

表 5.1　実験 1 の結果 Cayenne

		分類	
		バグ報告	機能拡張
実測	バグ報告	590	210
	機能拡張	81	719

表 5.2　実験 1 の結果 MINA

		分類	
		バグ報告	機能拡張
実測	バグ報告	411	28
	機能拡張	179	241

表 5.3　実験 1 の結果 James

		分類	
		バグ報告	機能拡張
実測	バグ報告	580	27
	機能拡張	316	396

表 5.4　実験 1 の結果 Lucene

		分類	
		バグ報告	機能拡張
実測	バグ報告	127	673
	機能拡張	18	782

表 5.5　実験 1 の結果 OpenJPA

		分類	
		バグ報告	機能拡張
実測	バグ報告	699	101
	機能拡張	280	520

表 5.6　実験 2 の結果 Cayenne

		分類	
		バグ報告	機能拡張
実測	バグ報告	514	286
	機能拡張	142	658

表 5.7　実験 2 の結果 MINA

		分類	
		バグ報告	機能拡張
実測	バグ報告	288	151
	機能拡張	78	342

表 5.8　実験 2 の結果 James

		分類	
		バグ報告	機能拡張
実測	バグ報告	441	166
	機能拡張	139	573

表 5.9　実験 2 の結果 Lucene

		分類	
		バグ報告	機能拡張
実測	バグ報告	525	275
	機能拡張	188	612

表 5.10　実験 2 の結果 OpenJPA

		分類	
		バグ報告	機能拡張
実測	バグ報告	557	243
	機能拡張	240	560

表 5.11　実験 3 の結果 Cayenne

		分類	
		バグ報告	機能拡張
実測	バグ報告	744	56
	機能拡張	409	391

表 5.12　実験 3 の結果 MINA

		分類	
		バグ報告	機能拡張
実測	バグ報告	402	37
	機能拡張	145	275

表 5.13　実験 3 の結果 James

		分類	
		バグ報告	機能拡張
実測	バグ報告	508	99
	機能拡張	141	571

表 5.14　実験 3 の結果 Lucene

		分類	
		バグ報告	機能拡張
実測	バグ報告	695	105
	機能拡張	325	475

表 5.15　実験 3 の結果 OpenJPA

		分類	
		バグ報告	機能拡張
実測	バグ報告	733	67
	機能拡張	389	411

表 5.16　評価値 Cayenne

実験	Accuracy	Recall	Precision	F_1
実験 1	0.8181	0.7375	0.8793	0.8022
実験 2	0.7325	0.6425	0.7835	0.7060
実験 3	0.7094	0.9300	0.6453	0.7619

表 5.17　評価値 MINA

実験	Accuracy	Recall	Precision	F_1
実験 1	0.7590	0.9362	0.6966	0.7988
実験 2	0.7334	0.6560	0.7869	0.7155
実験 3	0.7881	0.9157	0.7349	0.8154

表 5.18　評価値 James

実験	Accuracy	Recall	Precision	F_1
実験 1	0.7400	0.9555	0.6473	0.7718
実験 2	0.7688	0.7265	0.7603	0.7430
実験 3	0.8180	0.8369	0.7827	0.8089

表 5.19　評価値 Lucene

実験	Accuracy	Recall	Precision	F_1
実験 1	0.5681	0.1588	0.8759	0.2688
実験 2	0.7106	0.6562	0.7363	0.6940
実験 3	0.7312	0.8688	0.6814	0.7637

表 5.20　評価値 OpenJPA

実験	Accuracy	Recall	Precision	F_1
実験 1	0.7619	0.8738	0.7140	0.7858
実験 2	0.6981	0.6963	0.6989	0.6976
実験 3	0.7150	0.9163	0.6533	0.7627

表 5.21　総合評価値

手法	Accuracy	Recall	Precision	F_1
実験 1	73%	74%	77%	75%
実験 2	73%	68%	76%	72%
実験 3	76%	90%	70%	78%

6. 考察

6.1. Cayenne に対する考察

表 5.16 を見ると，実験 1 の方が正解率，適合率と F_1 値の精度が他の実験より高いという結果が出た．特に，正解率が 82%となった．実験 2 の結果を見ると，実験 1 より各評価値が下がった結果になった．実験 3 の場合，正解率，適合率と F_1 値の精度が 4%〜13%下がっているが，再現率が 74%から 93%になった．つまり，実験 3 の場合，バグ報告の分類を 93%できたという結果になった．

6.2. MINA に対する考察

表 5.17 を見ると, 実験 1 の方が再現率の精度が他の実験より高い 94%という結果が出た．実験 2 の結果を見ると，適合率の精度が高いが，他の評価値が下がった．特に，再現率の精度が実験 1 より 28%下がった結果になった．実験 3 の場合，実験 1 より再現率が 2%下がったが，正解率が 3%上がった結果になった．つまり，実験 3 の場合，79%の精度でバグ報告と機能拡張を振り分けることができた結果になった．

6.3. James に対する考察

表 5.18 を見ると, 実験 1 の方が再現率の精度が他の実験より高い 96%という結果が出た．実験 2 の結果を見ると，実験 1 より再開率と適合率の精度が上がったが，再現率と F_1 値の精度が下がった結果が出た．特に，再現率の精度が実験 1 より 23%下がった結果になった. 実験 3 の場合, 実験 1 と実験 2 より高い精度の結果が出た．特に，正解率の精度が 82%になった．つまり，実験 3 の場合，82%の精度でバグ報告と機能拡張を振り分けることができた結果になった．

6.4. Lucene に対する考察

表 5.19 を見ると, 実験 1 の方が再現率の精度が他の実験より低い 16%という結果が出た．再現率が低いため，F_1 値の精度も低くなった．実験 2 の結果を見ると，実験 1 より正解率の精度が 14%，再開率の精度が 50%上がった結果が出た．実験 3 の場合, 他の実験より適合率が下がったが, 正解率と再現率が高い精度の結果が出た．特に, 再現率の精度が 89%になった. つまり, 実験 3 の場合, バグ報告の分類を 89%できたという結果になった.

6.5. OpenJPA に対する考察

表 5.20 を見ると，実験 1 の方が正解率，適合率と F_1 値の精度が他の実験より高いという結果が出た．特に，正解率が 76%となった．実験 2 の結果を見ると，実験 1 より各評価値が下がった結果になった．実験 3 の場合，正解率，適合率と F_1 値の精度が 2%〜6%下がっているが，再現率が 87%から 92%になった．つまり，実験 3 の場合，バグ報告の分類を 92%できたという結果になった.

6.6. 先行研究との比較

Antoniol らの手法では，naïve Bayes，分類木とロジスティック回帰の機械学習器で学習させ，分類を行っている[1]．文献[1]に示された学習器ごとに対する平均評価値を表 6.1 に示す．表 6.1 を見ると，Antoniol らの手法の中で，ロジスティック回帰による振り分けの場合が，最も良い分類精度を出したことがわかる．表 5.21 を見ると，本研究では，実験 3 の分類の結果が最も良い精度となったことがわかる.

最も重要な評価値となる正解率と再現率を比較する．本研究での提案手法は Antoniol らの手法より正解率は平均 4%低くなっているが, 再現率は 12%高くなっている．再現率が高くなったということは，バグ報告の誤分類が少なくなったということである.

表 6.1　評価値（状況記述のテキスト分析による振り分け）

手法	Accuracy	Recall	Precision	F_1
naïve Bayes	69%	56%	86%	66%
分類木	74%	67%	78%	72%
ロジスティック回帰	80%	78%	82%	79%

6.7. 研究設問への解答

RQ1：スパムフィルタを用いたバグ報告の分類手法でどの精度までバグ報告と機能拡張を振り分けることができるか？

各プロジェクトの実験結果の評価値をまとめた表 5.16〜5.20 を見ると，正解率，再現率，適合率と F_1 値がそれぞれ 71〜82%，84〜93%，65〜78%，76〜82%の精度で振り分けることができた．つまり，71〜82%の精度でバグ報告と機能拡張を振り分けることができ，バグ報告の分類が 84〜93%できた．

RQ2：スパムフィルタを用いたバグ報告の分類手法と Antoniol らの手法ではどちらが良い精度を示すか？

表 5.21 と表 6.1 の評価値を比較すると，スパムフィルタを用いたバグ報告の分類手法は，Antoniol らの手法よりバグ報告と機能拡張を振り分ける精度が少し低下がされ，バグ報告の分類が比較的に向上された．

RQ3：提案手法における特徴は何か？

本研究で用いたスパムフィルタによる分類では，再現率が高くなる傾向にあることが確認できた．

7. 結言

本研究では，スパムフィルタを利用したバグ報告の自動分類手法を提案した．そして，提案手法を従来手法と比較するために，数種類のオープンソースソフトウェアに対してバグ報告と機能拡張を振り分けの分類実験を行った．実験結果，従来手法よりも高い分類精度が見られた．具体的には，正解率，適合率，F_1 値の精度が少し低下がされ，再現率が比較的に向上された．特に，再現率が向上され，バグ報告の振り分ける分類がより高くなることは，バグレポートの自動分類の性能向上に寄与できる可能性を示している．

今後の課題としては，正解率，適合率，F_1 値が低下した原因の検証と，より多くのプロジェクトに対しての実験の実行が挙げられる．

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学部門 水野 修准教授に厚く御礼申し上げます．本報告書執筆にあたり貴重な助言を多数頂きました，本学情報工学専攻　岡島秀記先輩，胡軼凡先輩，山田晃久先輩，采野友紀哉先輩，河端駿也先輩, 情報工学課程　森啓太君, 藤原剛史君, 山本滉明君をはじめとする，ソフトウェア工学研究室の皆さん，学生生活を通じて著者の支えとなった家族や友人に深く感謝致します．

参考文献

[1] Giuliano Antoniol, Kamel Ayari, Massimiliano Di Penta, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2008. 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* (CASCON '08), Article 23 , 15 pages. DOI=10.1145/1463788.1463819

[2] Osamu Mizuno and Tohru Kikuno, "Prediction of Fault-Prone Software Modules Using a Generic Text Discriminator," IEICE Trans. on Information and Systems, E91-D(4), pp. 888-896, April 2008.

[3] Osamu Mizuno and Tohru Kikuno, "Training on Errors Experiment to Detect Fault-Prone Software Modules by Spam Filter," In The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE2007), pp. 405-414, September 2007. (Dubrovnik, Croatia)

[4] Yukinao Hirata and Osamu Mizuno, "Do Comments Explain Codes Adequately? -- Investigation by Text Filtering --," In Proc. of 8th Working Conference on Mining Software Repositories (MSR2011), pp. 242-245, May 2011. (Honolulu, HI, USA)

[5] William S. Yerazunis, CRM114 -- the Controllable Regex Mutilator, [Online: http://crm114.sourceforge.net/]

[6] ASF Jira – Apache software foundation, [Online: http://issues.apache.org/]