修 士 論 文

題　目　　ソースコードにおける不具合出現と
感情推定の関係分析

主任指導教員　　水野　修　准教授

京都工芸繊維大学大学院 工芸科学研究科

情報工学専攻

学生番号　　14622039

氏　　名　　山田　晃久

平成28年2月10日提出

様式 2 号

学位論文の要旨（和文）

平成 28 年 2 月 10 日

京都工芸繊維大学大学院
工芸科学研究科長　殿

工芸科学研究科　　情報工学専攻
平成 26 年入学
学生番号　　14622039
氏　　名　　山田　晃久　㊞

（主任指導教員　　水野　修　㊞　）

本学学位規則第 4 条に基づき、下記のとおり学位論文内容の要旨を提出いたします。

1.　論文題目

ソースコードにおける不具合出現と感情推定の関係分析

2.　論文内容の要旨（400 字程度）

近年，自然言語処理の研究分野において感情推定が注目されている．これは一般的に，文章から筆者や話者の意見の極性を特定することに焦点を当てたテキスト分析である．本論文では，ソースコードから抽出した識別子やコメントを文章とみなして感情推定を行い，開発者の感情極性とソースコードの特徴の間に関係が存在するか否かの調査を行った．3 つのオープンソースソフトウェアプロジェクトに対して実験を適用した結果，プロジェクトによってソースコードの感情極性に差があることを発見した．また，ソースコードのコメント部分が識別子と比較して感情極性の決定に大きく影響していること，及び不具合を含まないソースコードには，不具合を含むソースコードと比較してポジティブな印象を持つ単語を含む識別子が多く使われていることを発見した．これらのことから，ソースコードと感情極性の間には関係が存在し，ソースコードの感情推定は不具合の分析に有効であると考えられる．

An Analysis of Relation between the Appearance of Fault and Sentiment Analysis in Source Code

2016 14622039 *YAMADA Akihisa*

Abstract

Sentiment analysis is becoming popular in research area of natural language processing. Recently, much research applying sentiment analysis to the field of software engineering has been conducted. In this thesis, we analyzed relationship between developers' emotional polarities and characteristics of source code. To do so, we apply sentiment analysis to identifiers and comments extracted from source code. We also conducted experiments with three open source software projects. The results of experiments show that emotional polarities obtained from three projects differ from each other. Our analysis also shows that comments in source code were more influence on emotional polarities than identifiers in source code, and that faulty source code modules have more identifiers including positive words compared with non-faulty source code modules. These results confirm the existence of relation between developers' emotional polarities and characteristics of source code. The relationship between faulty code modules and emotional polarity show a possibility to apply the sentiment analysis to the fault prone module detection.

目　次

1. 緒言

ソフトウェアの開発をより効率的に行うための様々な研究が行われている．その一つとして，リポジトリマイニングという手法がある．リポジトリとは，バージョン管理システムを用いたソフトウェア開発プロジェクトにおけるソースコードなどの情報をまとめて補完することができるデータベースである．このデータベースを分析し，ソフトウェアの開発における有益な知見を発見することがリポジトリマイニングの目的である．

リポジトリマイニングの研究の一環として，ソフトウェアリポジトリに含まれるソースコードなどのテキストデータに対して，テキスト分類やトピック分析などの自然言語処理の手法を適用する研究が進められている．

近年，自然言語処理の研究分野において，感情分析が注目されている．これは一般的に，文章から筆者や話者の意見の極性を特定することに焦点を当てたテキスト分析である．商品のレビューやソーシャルメディアなどに適用され，マーケティングや顧客サービスなどに用いられている．

ソフトウェア開発者の感情とソースコードにおける不具合の間に関係が発見されれば，開発者がどのような感情を持ちながら開発を行っている時に不具合が混入されやすいかを特定し，ソフトウェアへの不具合の混入を未然に防ぐことができるのではと考えられる．そこで本研究では，自然言語テキストではなくソフトウェアプログラムのソースコードに対して感情分析を適用し，ソースコードにおける不具合の出現と感情極性の間の関係の分析を行う．ソースコード中の変数や関数などに付ける名前を識別子といい，コメント文と同様にソースコードの特徴を表す重要な情報となる．本研究では，ソースコードに含まれる識別子やコメント文を自然言語とみなして感情推定を行い，それらが良い印象（ポジティブ）と悪い印象（ネガティブ）のどちらを持っているかを分析する．その結果から，対象とするソフトウェアプロジェクトやコメント文の有無，ソフトウェアに含まれる不具合の有無によって感情極性に差があるか否かを調査する．

本論文の以降の構成を述べる．2 章では研究の基礎となる関連研究について述べる．3 章では本研究において設定した研究設問を述べる．4 章ではソースコードの感情推定の手法の概要について述べる．5 章ではソースコードの感情推定を対象プロ

ジェクトに適用した実験の手順，及び結果について述べる．6 章では実験の結果に対する考察，及び今後の課題について述べる．7 章では本研究のまとめについて述べる．

2. 関連研究

ソフトウェア工学分野では、リポジトリマイニングの隆盛に伴って

バージョン管理システムの変更履歴を用いて不具合を予測する研究が行われている．Nagappan らは，ソースコードの変更履歴から不具合を予測する研究 [1] を行った．また，Kim らは過去の不具合の修正の履歴から未発見の不具合を予測する手法 [2][3] を提案した．(だから何なのか)

感情推定の分野において様々な研究が行われている．Adam らは，SNS である FaceBook を対象に実験 [4] を行った．その結果，ポジティブな投稿の表示を減らした場合，そのユーザ自身のポジティブな投稿が減り，ネガティブな投稿が増えるという結果を示した．高野らは感情推定法の提案を行い，新聞社のアクセス解析へ適用 [5] した．その結果，ポジティブな記事を閲覧した人は連続でポジティブな記事を閲覧しやすく，同様にネガティブな記事を閲覧した人は連続でネガティブな記事を閲覧しやすいという結果を示した．(それでどう役に立つのか)

ソフトウェア工学の分野に感情推定を適用した様々な研究も行われている．Michal は 56 人のソフトウェア開発者を対象に調査を行い，ほとんどの開発者はポジティブな感情を持っている時は生産性が向上し，ネガティブな感情を持っている時は生産性が向上しにくいという結果を示した [6]．Sebastian らはソフトウェア開発者を対象にバイオメトリックセンサーを用いた実験を行った [7]．その結果，開発者の感情と作業の進捗には相関があることを示した．Alessandro らは，バグ管理システムのバグ報告から自動的に感情を分析するツールを作成した [8]．Emitza らはソフトウェア開発プロジェクトのための Web 共有サービスである GitHub[9] を対象に，異なるオープンソースプロジェクトのコミットコメントに対して感情分析を行い，感情と用いられるプログラミング言語やコミットコメントが書かれた時間・曜日などの関係を調査した [10]. (これらと本研究の関わり)

3. 研究設問

本研究は将来的に，開発者がどのような感情を持ちながら開発を行っている時に不具合が混入されやすいかを特定し，開発者の精神的な負担の軽減や，不具合の発生を未然に防止することを目的としている．

これらの目的のために本研究では，開発者の感情極性とソースコードの間に関係が存在するかを調査する．

従って，本研究では以下に設定する研究設問について調査を行う．

RQ1 ソースコードのコメントの有無によって感情極性に差はあるか．

RQ2 異なるプロジェクトのソースコード間に感情極性の差はあるか．

RQ3 ソースコードの不具合の有無によって感情極性に差はあるか．

RQ1 ではソースコードから抽出した識別子を文章と見なして感情推定を行った場合に，識別子にコメントを含む場合と含まない場合とで，感情極性に差があるかを調査する．RQ2 では異なるプロジェクトのソースコードから抽出した識別子を，それぞれ文章と見なして感情推定を行った場合に，感情極性に差はあるかを調査する．RQ3 では不具合を含むソースコードから抽出した識別子と，不具合を含まないソースコードから抽出した識別子を，それぞれ文章と見なして感情推定を行った場合に，感情極性に差があるかを調査する．

4. ソースコードの感情推定

本研究では，以下の手順でソースコードの感情推定を行う．

1. ソフトウェアリポジトリとバグデータベースのバグ情報を統合する．
2. バグが含まれるソースコードファイル，及びバグが含まれないソースコードファイルを抽出する．
3. ソースコードファイルから識別子，及びコメントを抽出する．
4. 抽出した識別子，及びコメントから感情推定を行う．

これらの流れを図 4.1 に示す．以降の節で各手順の詳細を述べる．

4.1 バグ情報の統合

バージョン管理システムで管理されたソフトウェアリポジトリと，バグデータベースに記録されているバグ情報を統合し，バグ混入モジュールを識別する．本研究では，バグ情報の統合には SZZ アルゴリズム [11] を用いる．

4.1.1 SZZ アルゴリズム

SZZ アルゴリズムは，バージョン管理システムとバグデータベースの情報を相互に結びつけることで自動的にバグ混入モジュールを識別するアルゴリズムである．

SZZ アルゴリズムの基本となるアイデアを以下に示す．

- バグデータベースからバグの修正を行っているログを抜き出す．
- バグのログとバージョン管理システムを結びつけて，バグの修正を行ったソースコードの変更を抜き出す．
- バグの報告日以前のソースコードの変更をバグの原因となった変更として決定する．

バグの報告日以前のソースコードの変更が，その後のバグの修正の原因の一つと考えられる．したがって，そのソースコードの変更を識別することがバグ混入モジュールを識別することとなる．

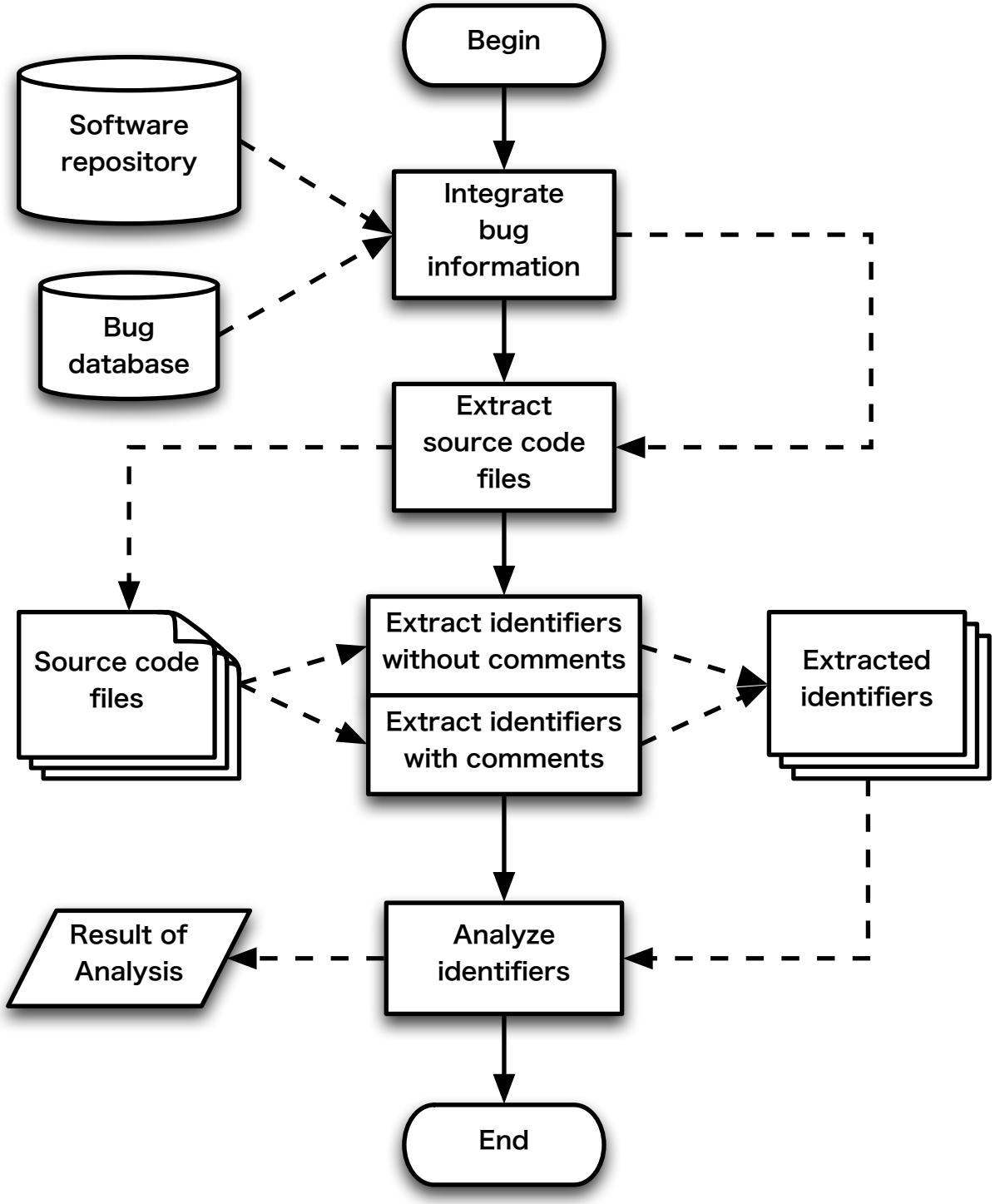
Begin
Software repository
Integrate bug information
Bug database
Extract source code files
Source code files
Extract identifiers without comments
Extract identifiers with comments
Extracted identifiers
Result of Analysis
Analyze identifiers
End

図 4.1　感情推定の手順

4.1.2　利用したツール: szz_tools

本研究では研究室において開発・保守されている，Git リポジトリに SZZ アルゴリズムを適用し，FIX 情報と BUG 情報，及びバグが含まれるモジュールとバグが含まれないモジュールの情報を特定して Git の tag 情報として記録ツールである szz_tools を利用した．

4.2　ソースコードファイルの抽出

手順 1 でバグ情報を統合したソフトウェアリポジトリから，バグが含まれているソースコードファイルと，バグが含まれていないソースコードファイルを抽出する．

4.3　識別子の抽出

手順 2 で得られたソースコードから識別子，及びコメントを抽出する．本研究では，ソースコードのコメントの有無によって感情極性に差があるかを調査する．そのため，バグが含まれているソースコードとバグが含まれていないソースコードのそれぞれについて，識別子のみの抽出を行うとともに，識別子とコメントの両方の抽出も行う．

利用したツール: lscp

本研究ではソースコードからの識別子の抽出に，GitHub で公開されている「lscp」[12] という Perl プログラムを利用した．lscp は A lightweight source code preprocesser の略称であり，ソースコードファイルから識別子名やコメント，文字列リテラルなどの言語データを区別・操作できるツールである．また，lscp は「camelCase」「snake_case」「dot.notation」など，複数の単語を組み合わせた識別子名を単語ごとに分割することができる．本研究では，抽出した識別子やコメントを自然言語と見なして分析を行うため，複数の単語を組み合わせた識別子名はそれぞれの単語に分割して抽出を行う．また，それぞれのソースコードについて識別子のみを抽出する設定と，識別子とコメントの両方を抽出する設定の 2 通りの設定で実行する．

4.4 感情推定

手順 3 で得られた識別子とコメントを文章と見なして感情推定を行う．本研究では，単純ベイズ分類器を用いて感情極性の推定を行う．感情極性とは，その単語や文章が持っている印象がポジティブ (positive) とネガティブ (negative) のどちらであるかを表す二値属性である．

利用した API: Natural Language Sentiment Analysis API

本研究では抽出した識別子やコメントの感情推定に，Web 上で公開されている Natural Language Sentiment Analysis API[13] を利用する．これは，解析したい自然言語の単語や文章のテキストデータを HTTP で送信すると，分析結果として JSON オブジェクトを返す API である．

この API で実行されるプログラムは Python プログラムであり，NLTK[14] を利用している．NLTK は Natural Language Toolkit の略称であり，Python 言語で書かれた自然言語処理のためのツールキットである．NLTK には，テキスト分類，トークン化，ステミング，構文解析などのテキスト処理のためのライブラリがあり，主にテキストマイニングなどに用いられる．

また，分類器の学習には Bo らが作成したデータセットに含まれる映画のレビューのデータセット [15] を用いている．

分析結果として返される JSON オブジェクトは次の 2 つの属性を持つ．

label: テキストデータを分析した結果が，positive, negative, neutral のいずれであるかを示す．

probability: positive, negative, neutral のそれぞれの確率を示す．positive または netagive である確率の合計は 1 であるが，neutral である確率は独立している．neutral である確率が 0.5 より大きければ，label は neutral となり，そうでない場合は，label は positive と negative のうち，確率が大きい方が選ばれる．

分析結果として返される JSON オブジェクトの例を以下に示す．

```
{
    "label": "pos",
```

```
    "probability": {
        "pos": 0.85,
        "neg": 0.15,
        "neutral": 0.4
    }
}
```

5. 適用実験

ここに全体の流れを

5.1 対象プロジェクト

本研究では次の 3 つのオープンソースプロジェクトのリポジトリを対象とする．

- **Apache MINA**[16]

  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．

- **Apache OpenJPA**[17]

  Java Persistence API 仕様のオープンソース実装の一つであり，Java 言語を用いて開発されている．

- **Apache James**[18]

  James は Java Apache Mail Enterprise Server 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・メールアプリケーションフレームワークである．

5.2 バグ情報の統合

4.1 節の手順に従い，研究室で開発・保守されているツール szz_tools を用い，Git で管理された 3 つのリポジトリに対して，バグデータベースの情報とリポジトリの更新情報を結びつける．なお，本実験ではバグ情報は Apache.org が管理するバグデータベース「JIRA」[19] から取得する．

利用したツール: szz_tools.pl

Git の tag コマンドを利用して，バグの除去時点には FIX タグを，バグの混入時点には BUG タグを添付する．そして，バグが含まれるモジュールを特定し，FAULT タグを添付した後，バグが含まれないモジュールを特定し，INNOCENT タグを添付するプログラムである．また，FIX タグと BUG タグには，ソースコードの変更でのバグの原因と考えられるファイルの名前がそれぞれのタグにコメントとして添付する．

添付されたFIXタグ，BUGタグ，FAULTタグ，及びINNOCENTタグの名前は次のようになっている．

- FIX-(issue_ID)-(fix_ID)
- BUG-(issue_ID)-(fix_ID)-(bug_ID)
- FAULT-(issue_ID)-(fix_ID)-(bug_ID)-(file_type)-(blob_hash)
- INNOCENT-(issue_ID)-(fix_ID)-(bug_ID)
  - (issue_ID)：バグデータベースにおける問題番号
  - (fix_ID)：バグ修正の通し番号
  - (bug_ID)：バグの通し番号
  - (file_type)：そのモジュールのファイルタイプ
  - (blob_hash)：そのブロブのハッシュ

利用方法は次の通りである．

```
$ SZZ.pl --target-dir target_directory --csv-dir
  csv_directory
```

–target-dirオプションには対象の.gitリポジトリが存在するパスを示す．–csv-dirオプションにはバグデータベースの情報が記録されているcsvファイルが存在するパスを示す．

5.3 ソースコードファイルの抽出

4.2節の手順に従い，バグが含まれているソースコードファイルとバグが含まれていないソースコードファイルを抽出する．

作成したツール: extract_source-code.pl

Gitで管理されたリポジトリから，FAULTタグ及びINNOCENTタグが添付されたソースコードファイルを抽出し，バグが含まれているソースコードファイルとバグが含まれていないソースコードファイルを区別して保存するプログラムである．

具体的には次の手順で動作する．

1. 対象リポジトリから FAULT タグを取得する．
2. Git の show コマンドを実行し，FAULT タグが付いているソースコードを取得する．
3. 取得したソースコードを FAULTs ディレクトリに保存する．
4. 対象リポジトリから INNOCENT タグを取得する．
5. Git の show コマンドを実行し，INNOCENT タグが付いているソースコードを取得する．
6. 取得したソースコードを INNOCENTs ディレクトリに保存する．

利用方法は次の通りである．

```
$ extract_source-code.pl --target-dir target_directory --
  output-dir output_directory
```

–target-dir オプションには対象の.git リポジトリが存在するパスを示す．–output-dir オプションには結果を出力するディレクトリのパスを示す．

✓ 作成したツールは公開リポジトリにのっけてしれ示す。

5.4 識別子の抽出

4.3 節の手順に従い，前節で抽出したソースコードファイルから lscp を用いて識別子とコメントを抽出する．lscp の実行例を以下に示す．

```
use lscp;

my $lscp = lscp->new;

# The verbosity of the program. Options: "info", "warn", "error",
   "fatal"
$lscp->setOption("logLevel", "fatal");
# The directory containing the input files.
$lscp->setOption("inPath", $IN_DIR);
# The directory for the output files.
$lscp->setOption("outPath", $OUT_DIR);
# Are the input files source code (1), or regular text files (0)?
$lscp->setOption("isCode", 1);
```

```
# If isCode==1, should the program include identifier names?
$lscp->setOption("doIdentifiers", 1);
# Should the program split identifier names,
# such as:camelCase, under_scores, dot.notation?
$lscp->setOption("doTokenize", 1);
# Should the program remove puncuation symbols?
$lscp->setOption("doRemovePunctuation", 1);
# Should the program remove digits [0-9]?
$lscp->setOption("doRemoveDigits", 0);
# Should the program remove small words?
$lscp->setOption("doRemoveSmallWords", 1);
# If doRemoveSmallWords==1, what is the minumum size of words to
  keep?
$lscp->setOption("SmallWordSize", 2);
# Should the program remove programming language keywords?
$lscp->setOption("doStopwordsKeywords", 1);
# If isCode==1, should the program include comments?
$lscp->setOption("doComments", 0);

$lscp->preprocess();
```

ソースコードファイルから識別子を抽出する際に，複数の単語を組み合わせた識別子名はそれぞれの単語に分割するよう設定する．また，句読記号や文字数が 1 文字の単語は除外するよう設定する．

本研究では 3 章に記したように，ソースコードから識別子を抽出する際のコメントの有無によって，感情極性に差があるか否かを調査するため，それぞれのソースコードについて識別子のみを抽出する設定と，識別子とコメントの両方を抽出する設定の 2 通りの設定で実行する．

5.5 感情推定

4.4 節の手順に従い，前節で得られた識別子とコメントを文章と見なして感情推定を行う．

作成したツール: senti-analyze.pl

コマンドライン引数から指定されたディレクトリに保存されているファイルを対象として，ファイルに記録されている複数の単語を 1 つの文章と見なして感情推定を行い，その結果である感情極性が Positive, Negaive, Neutral のいずれであるかを判別し，それぞれの数を集計するプログラムである．感情推定には Web 上で公開されている Natural Language Sentiment Analysis API を利用する．

具体的には次の手順で動作する．

1. コマンドライン引数で指定されたディレクトリ内のファイルの一覧を取得する．
2. それぞれのファイルに記録されているテキスト情報を抽出し，それらをパラメータとして Natural Language Sentiment Analysis API を利用する curl コマンドを実行する．
3. API を実行して返される JSON オブジェクトから label 情報を取得し，label ごとに集計する．
4. 集計の結果を出力する．

また，文字数制限などの理由で API の実行結果としてエラーが返された場合は，そのファイルを結果の集計から除外する．

利用方法は次の通りである．

```
$ senti-analyze.pl --target-dir target_directory --output-
  file output_name
```

–target-dir オプションには対象のファイルが存在するパスを示す．–output-file オプションには結果を出力するファイルのパスを示す．

5.6 実験結果

それぞれのプロジェクトについて，取得した FAULT タグと INNOCENT タグの数を表 5.1 に示す．

表 5.1　取得した FAULT タグと INNOCENT タグの数

プロジェクト名	FAULT タグの数	INNOCENT タグの数
Apache MINA	6314	4078
Apache OpenJPA	12518	13454
Apache James	3794	37378

5.6.1 Apache MINA

表 5.2, 5.3 に Apache MINA プロジェクトに対して，識別子とコメントを抽出したものを文章とみなして感情推定を行った場合と，識別子のみを抽出したものを文章とみなして感情推定を行った場合のそれぞれについて，Positive, Negative, Neutral と判別された数をタグごとに示す.

また，表 5.2, 5.3 について，それぞれのタグにおける Positive, Negative, Neutral と判別された数の割合を表 5.4, 5.5 に示す.

5.6.2 Apache OpenJPA

表 5.6, 5.7 にそれぞれ Apache OpenJPA プロジェクトに対して，識別子とコメントを抽出したものを文章とみなして感情推定を行った場合と，識別子のみを抽出したものを文章とみなして感情推定を行った場合のそれぞれについて，Positive, Negative, Neutral と判別された数をタグごとに示す.

また，表 5.6, 5.7 について，それぞれのタグにおける Positive, Negative, Neutral と判別された数の割合を表 5.8, 5.9 に示す.

5.6.3 Apache James

表 5.10, 5.11 にそれぞれ Apache James プロジェクトに対して，識別子とコメントを抽出したものを文章とみなして感情推定を行った場合と，識別子のみを抽出したものを文章とみなして感情推定を行った場合のそれぞれについて，Positive, Negative, Neutral と判別された数をタグごとに示す.

また，表 5.10, 5.11 について，それぞれのタグにおける Positive, Negative, Neutral と判別された数の割合を表 5.12, 5.13 に示す.

表 5.2　Apache MINA における分析結果（コメントあり）

タグ	Positive の数	Negative の数	Neutral の数
FALUT	3660	344	2310
INNOCENT	2636	174	1249
合計	6296	518	3559

表 5.3　Apache MINA における分析結果（コメントなし）

タグ	Positive の数	Negative の数	Neutral の数
FALUT	334	596	5384
INNOCENT	823	333	2903
合計	1157	929	8287

表 5.4　Apache MINA における各タグの割合（コメントあり）

タグ	Positive の割合	Negative の割合	Neutral の割合
FALUT	0.580	0.054	0.366
INNOCENT	0.649	0.043	0.308
合計	0.607	0.050	0.343

表 5.5　Apache MINA における各タグの割合（コメントなし）

タグ	Positive の割合	Negative の割合	Neutral の割合
FALUT	0.053	0.094	0.853
INNOCENT	0.203	0.082	0.715
合計	0.112	0.090	0.799

表 5.6　Apache OpenJPA における分析結果（コメントあり）

タグ	Positive の数	Negative の数	Neutral の数
FALUT	5736	2036	4196
INNOCENT	8224	1029	3621
合計	13960	3065	7817

表 5.7　Apache OpenJPA における分析結果（コメントなし）

タグ	Positive の数	Negative の数	Neutral の数
FALUT	315	942	10846
INNOCENT	2046	1240	9677
合計	2361	2182	20523

表 5.8　Apache OpenJPA における各タグの割合（コメントあり）

タグ	Positive の割合	Negative の割合	Neutral の割合
FALUT	0.479	0.170	0.351
INNOCENT	0.639	0.080	0.281
合計	0.562	0.123	0.315

表 5.9　Apache OpenJPA における各タグの割合（コメントなし）

タグ	Positive の割合	Negative の割合	Neutral の割合
FALUT	0.026	0.078	0.896
INNOCENT	0.158	0.096	0.747
合計	0.094	0.087	0.819

表 5.10　Apache James における分析結果（コメントあり）

タグ	Positive の数	Negative の数	Neutral の数
FALUT	1740	737	1317
INNOCENT	16924	3213	15831
合計	18664	3950	17148

表 5.11　Apache James における分析結果（コメントなし）

タグ	Positive の数	Negative の数	Neutral の数
FALUT	63	449	3282
INNOCENT	5184	3030	27814
合計	5247	3479	31096

表 5.12　Apache James における各タグの割合（コメントあり）

タグ	Positive の割合	Negative の割合	Neutral の割合
FALUT	0.459	0.194	0.347
INNOCENT	0.471	0.089	0.440
合計	0.469	0.099	0.431

表 5.13　Apache James における各タグの割合（コメントなし）

タグ	Positive の割合	Negative の割合	Neutral の割合
FALUT	0.017	0.118	0.865
INNOCENT	0.144	0.084	0.772
合計	0.132	0.087	0.781

6. 考察

~~6.1~~ 妥当性の検証

まず初めに，実験結果の妥当性について検証を行う．

オープンソースプロジェクトに適用した実験

今回の実験で用いたプロジェクトは全てオープンソースのものである．つまり，商業のプロジェクトに今回の実験で用いた手法を適用しても，同様の結果が得られるとは限らない．

管理システムの完全性

バージョン管理システムやバグ管理システムにはバグの情報が記録されている．しかし，その情報は完全に正確なものではない．プロジェクト中にまだ発見されていないバグが含まれている可能性があるからである．そのため，バグを混入しない変更と扱った変更も，実際はバグを混入する変更である可能性がある．

SZZ アルゴリズムの完全性

今回の実験に用いたデータは SZZ アルゴリズムを用いて作成した．しかし，バージョン管理システムとバグ管理システムを完全に結びつけられるものではない．そのため，解析されないデータが出てくるのは必至であり，それらの解析されなかったデータは扱うことができない．

~~6.2~~ 研究設問の考察

実験結果の妥当性について注意し，研究設問を通して実験結果に対する考察を示す．

RQ1 ソースコードのコメントの有無によって感情極性に差はあるか．

識別子のみを文章とみなして感情推定を行った場合と，識別子とコメントを文章とみなして感情推定を行った場合とで感情極性の Positive, Negative, Neutral の割合を比較する．

Apache MINA について，表 5.4, 5.5 から，全てのタグにおける Positive, Negative, Neutral である割合を比較すると，感情推定にコメントを含んだ場合はそれぞれの割合が 0.607, 0.050, 0.343 であるのに対し，感情推定にコメントを含まない場合はそれぞれの割合が 0.111, 0.090, 0.799 である．このことから Apache MINA において，感情推定にコメントを含まない場合はコメントを含んだ場合と比較して Positive または Negative である割合が低く，Neutral である割合が高いと言える．

Apache OpenJPA についても同様に，表 5.8, 5.9 から，全てのタグにおける Positive, Negative, Neutral である割合を比較すると，感情推定にコメントを含んだ場合はそれぞれの割合が 0.562, 0.123, 0.315 であるのに対し，感情推定にコメントを含まない場合はそれぞれの割合が 0.094, 0.087, 0.819 である．このことから Apache OpenJPA において，感情推定にコメントを含まない場合はコメントを含んだ場合と比較して Positive または Negative である割合が低く，Neutral である割合が高いと言える．

Apache James についても同様に，表 5.12, 5.13 から，全てのタグにおける Positive, Negative, Neutral である割合を比較すると，感情推定にコメントを含んだ場合はそれぞれの割合が 0.469, 0.099, 0.431 であるのに対し，感情推定にコメントを含まない場合はそれぞれの割合が 0.132, 0.087, 0.781 である．このことから Apache James において，感情推定にコメントを含まない場合はコメントを含んだ場合と比較して Positive または Negative である割合が低く，Neutral である割合が高いと言える．

これらのことから，ソースコードの識別子のみを文章とみなして感情推定を行った場合と，識別子とコメントを文章とみなして感情推定を行った場合とで感情極性に差はあると言える．また，いずれのプロジェクトにおいても，感情推定にコメントを含む場合と比較してコメントを含まない場合の方が Neutral である割合が高いことから，ソースコードのコメント部分が感情極性の決定に大きく影響していると考えられる．

RQ2 異なるプロジェクトのソースコード間に感情極性の差はあるか．

それぞれのプロジェクトから抽出した識別子を文章とみなして感情推定を行った結果の，感情極性の Positive, Negative, Neutral の割合に差があるかを確かめるため，それぞれのプロジェクトの組み合わせについて，分布の差の検定を行う．なお分布の差の検定は，統計分析ソフトである「R」を用い，ピアソンの χ^2 検定を行った．ま

た，それぞれの検定において有意水準は 5% とした．

Apache MINA と Apache OpenJPA の組み合わせの場合，コメントを含む場合とコメントを含まない場合のそれぞれについて検定を行った結果，コメントを含む場合の p 値は $p < 2.2 \times 10^{-16}$，コメントを含まない場合の p 値は $p = 1.918 \times 10^{-6}$ となった．どちらの場合も p 値が $p < 0.05$ であるので，コメントを含む場合とコメントを含まない場合のどちらにおいても，2 つのプロジェクトから抽出した識別子の感情極性には有意な差があると言える．

同様に，Apache MINA と Apache James の組み合わせの場合，コメントを含む場合とコメントを含まない場合のそれぞれについて検定を行った結果，コメントを含む場合の p 値は $p < 2.2 \times 10^{-16}$，コメントを含まない場合の p 値は $p = 2.718 \times 10^{-7}$ となった．どちらの場合も p 値が $p < 0.05$ であるので，コメントを含む場合とコメントを含まない場合のどちらにおいても，2 つのプロジェクトから抽出した識別子の感情極性には有意な差があると言える．

また，Apache MINA と Apache James の組み合わせの場合においても，コメントを含む場合とコメントを含まない場合のそれぞれについて検定を行った結果，コメントを含む場合の p 値は $p < 2.2 \times 10^{-16}$，コメントを含まない場合の p 値は $p < 2.2 \times 10^{-16}$ となった．どちらの場合も p 値が $p < 0.05$ であるので，コメントを含む場合とコメントを含まない場合のどちらにおいても，2 つのプロジェクトから抽出した識別子の感情極性には有意な差があると言える．

これらのことから，異なるプロジェクトのソースコードから抽出した識別子を文章とみなして感情推定を行った場合，感情極性に差はあると言える．また，いずれのプロジェクトの組み合わせにおいても，コメントを含む場合だけでなくコメントを含まない場合においても感情極性に差があることから，プロジェクトによって使用される識別子が異なることが理由であると考えられる．

RQ3 ソースコードの不具合の有無によって感情極性に差はあるか．

不具合を含むソースコードと不具合を含まないソースコードのそれぞれから抽出した識別子を文章と見なして感情推定を行った場合に，感情極性の Positive, Negative, Neutral の割合に差があるかを確かめるため，各プロジェクトごとに分布の差の検定を行う．なお，上で行った検定と同様に統計分析ソフトである「R」を用い，ピアソ

ンの χ^2 検定を行った．また，それぞれの検定において有意水準は 5% とした．

Apache MINA について，コメントを含む場合とコメントを含まない場合のそれぞれについて検定を行った結果，コメントを含む場合の p 値は $p = 9.194 \times 10^{-12}$，コメントを含まない場合の p 値は $p < 2.2 \times 10^{-16}$ となった．どちらの場合も p 値が $p < 0.05$ であるので，コメントを含む場合とコメントを含まない場合のどちらにおいても，FAULT タグと INNOCENT タグの感情極性には有意な差があると言える．

同様に Apache OpenJPA についても，コメントを含む場合とコメントを含まない場合のそれぞれについて検定を行った結果，コメントを含む場合の p 値は $p < 2.2 \times 10^{-16}$，コメントを含まない場合の p 値は $p < 2.2 \times 10^{-16}$ となった．どちらの場合も p 値が $p < 0.05$ であるので，コメントを含む場合とコメントを含まない場合のどちらにおいても，FAULT タグと INNOCENT タグの感情極性には有意な差があると言える．

また，Apache James についても，コメントを含む場合とコメントを含まない場合のそれぞれについて検定を行った結果，コメントを含む場合の p 値は $p < 2.2 \times 10^{-16}$，コメントを含まない場合の p 値は $p < 2.2 \times 10^{-16}$ となった．どちらの場合も p 値が $p < 0.05$ であるので，コメントを含む場合とコメントを含まない場合のどちらにおいても，FAULT タグと INNOCENT タグの感情極性には有意な差があると言える．

これらのことから，不具合を含むソースコードと不具合を含まないソースコードのそれぞれから抽出した識別子を文章と見なして感情推定を行った場合，感情極性に差はあると言える．また，いずれの条件においても FAULT タグよりも INNOCSENT タグの方が Positive である割合が高いことから，不具合を含むソースコードと比較して不具合を含まないソースコードには，ポジティブな印象を持つ単語を含む識別子が多く使われていると考えられる．

6.3 今後の課題

以上の考察を踏まえ，今後の課題について考察する．

- 本研究では，5.1 節に示した 3 つのプロジェクトを対象に実験を行った．より多くのプロジェクトを対象として実験を行うことによって，実験結果のさらなる一般化が期待できる．

- 本研究で行った感情推定に用いた分類器の学習には，映画のレビューのデータセットを利用している．より多くのデータセットで分類器を学習することにより，感情推定の精度の向上が期待できる．
- 静的コード解析に本研究の考えを適用することで，開発者の感情の分析によって，ソフトウェアに不具合が混入することを未然に防止できる可能性がある．

7. 結言

本研究では，ソースコードから抽出した識別子やコメントを文章とみなして感情推定を行い，開発者の感情極性とソースコードの間に関係が存在するか否かの調査を行った．3 つのオープンソースソフトウェアプロジェクトに対して実験を適用し，以下の分析結果が得られた．

- ソースコードのコメント部分が感情極性の決定に大きく影響している．
- 異なるプロジェクトのソースコード間に感情極性の差は存在する．
- 不具合を含まないソースコードにはポジティブな印象を持つ単語を含む識別子が多く使われている．

これらの結果から，ソースコードと感情極性の間には関係が存在し，ソースコードの感情推定は不具合の分析に有効であると考えられる．

今後の課題として，より多くのプロジェクトへの実験の適用や分類器の学習に用いるデータセットの拡充，静的ソースコード解析への適用が挙げられる．

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，本学情報工学専攻河端駿也氏，釆野友紀也氏，藤原剛史氏，森啓太氏，Nicolas Fruy 氏，Anais Tournois 氏，本学情報工学課程黒田翔太氏，田中健太郎氏，西浦生成氏，原田禎之氏の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] N. Nagappan and T. Ball, “Use of relative code churn measures to predict system defect density,” Proceedings of the 27th international conference on Software engineering, pp.284–292, St. Louis, USA, May 2005.

[2] S. Kim, K. Pan, and J. E. James Whitehead, “Memories of bug fixes,” Proceedings of 14th ACM SIGSOFT international symposium on foundations of software engineering, pp.34–45, Portland, USA, Nov. 2006.

[3] S. Kim, T. Zimmermann, J. E. James Whitehead, and A. Zeller, “Predicting faults from cached history,” Proceedings of the 29th international conference on Software Engineering, pp.489–498, Minneapolis, USA, May 2007.

[4] A.D.I. Kramer, J.E. Guillory, and J.T. Hancock, “Experimental evidence of massive-scale emotional contagion through social networks,” Proceedings of the National Academy of Sciences of the United States of America, vol.111, no.24, pp.8788–8790, 2014.

[5] 高野憲悟，萩原将文，“感情関連語を用いた感情推定法の提案とニュースサイトのアクセス解析への応用，” Proceedings of the 29th international conference on Software Engineering，vol.11，no.3，pp.495–502，2012.

[6] M.R. Wrobel, “Emotions in the software development process,” Proceedings of the 6th International Conference on Human System Interaction, pp.518–523, IEEE, 2013.

[7] S.C. Müller and T. Fritz, “Stuck and frustrated or in flow and happy: Sensing developers’ emotions and progress，” Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering，pp.688–699，IEEE，2015.

[8] A. Murgia, P. Tourani, B. Adams, and M. Ortu, “Do developers feel emotions? an exploratory analysis of emotions in software artifacts,” Proceedings of the 11th Working Conference on Mining Software Repositories, pp.262–271, ACM, New York, NY, USA, 2014.

[9] GitHub, Inc., GitHub Where software is built, （オンライン），入手先〈https://github.com/〉（参照 2016-02-01）.

[10] E. Guzman, D. Azócar, and Y. Li, “Sentiment analysis of commit comments in github: An empirical study,” Proceedings of the 11th Working Conference on Mining Software Repositories, pp.352–355, ACM, New York, NY, USA, 2014.

[11] J. Śliwerski, T. Zimmermann, and A. Zeller, “When do changes induce fixes?,” Proceedings of the 2005 international workshop on Mining software repositories, pp.1–5, ACM, New York, NY, USA, 2005. St. Louis, Missouri.

[12] S. Thomas, lscp – lightweight source code preprocesser., （オンライン）, 入手先〈https://github.com/doofuslarge/lscp/〉（参照 2016-02-01）.

[13] J. Perkins, Sentiment Analysis text-processing.com API 1.0 documentation, （オンライン）, 入手先〈http://text-processing.com/docs/sentiment.html/〉（参照 2016-02-01）.

[14] Natural Language Toolkit, （オンライン）, 入手先〈http://www.nltk.org/〉（参照 2016-02-01）.

[15] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp.115–124, 2005.

[16] Apache, Apache MINA, （オンライン）, 入手先〈http://mina.apache.org/〉（参照 2016-02-01）.

[17] Apache, Apache OpenJPA –, （オンライン）, 入手先〈http://openjpa.apache.org/〉（参照 2016-02-01）.

[18] Apache, Apache James Project - Overview, （オンライン）, 入手先〈http://james.apache.org/〉（参照 2016-02-01）.

[19] Atlassian, JIRA Software - Issue & Project Tracking for Software Teams — Atlassian, （オンライン）, 入手先〈https://www.atlassian.com/software/jira〉（参照 2016-02-01）.