NAIST-IS-MT1651008

修士論文

オープンソースソフトウェア開発における README 記述項目の分析

池田 祥平

2018 年 3 月 15 日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

池田 祥平

審査委員：

松本 健一 教授	(主指導教員)
飯田 元 教授	(副指導教員)
石尾 隆 准教授	(副指導教員)
伊原 彰紀 助教	(副指導教員)
Raula Gaikovina Kula 特任助教	(副指導教員)
上野 秀剛 准教授	(奈良工業高等専門学校)

オープンソースソフトウェア開発における README 記述項目の分析*

池田 祥平

内容梗概

本論文では，オープンソースソフトウェア（OSS）プロジェクトの開発者に対して README 作成時の方針を与えることを目的として，README を公開するプロジェクトにおいて，多くの README で共通して記述される項目を明らかにする．OSS プロジェクトにおいて README は，インストール方法や利用例等のプロジェクトのソフトウェアを正しく使用するための方法，プロジェクトへの貢献方法等の開発を促進するための方法を共有するために重要なドキュメントである．README の品質を向上させるため，README に記述する項目のガイドラインが存在する．しかしすべての README がガイドラインに準拠しているか否かは明らかではなく，またガイドラインがあらゆるソフトウェア種別の README に適用可能であるかも明らかでない．本論文では，README に記述される項目を明らかにするために，既存の README に記述された項目を抽出し，ソフトウェア種別に記述項目を比較した．43,991 件のプロジェクトを対象に分析した結果，README にはガイドラインに示されているインストール方法，利用方法，ライセンスが最も頻繁に記述されていたが，すべての README に記述されているわけではなかった．またガイドラインに示されている項目を記述するために複数の方法が取られており，ガイドラインに示されていない項目も存在した．ガイドラインに示されていない項目は，README 作成時の方針となる．ソフトウェア種別に記述項目を比較すると，アプリケーションソフトウェアには多数の機能を説明

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, NAIST-IS-MT1651008, 2018 年 3 月 15 日.

する項目が，ライブラリソフトウェアではインストール方法とライセンスが他方に比べ頻繁に記述されていた．ソフトウェア種別に重視する項目の違いは，OSSプロジェクトの開発者が自身のソフトウェア種別に合わせたREADMEを作成する方針となりうる．

キーワード

README，ソフトウェアドキュメント，ソフトウェアエコシステム，オープンソースソフトウェア，アソシエーションルールマイニング

It ' s Written In the README file*

Shohei Ikeda

Abstract

In this thesis, for the purpose of suggest the open source software (OSS) developers guidelines for creating README file, I clarify contents that are commonly provided by software README files. In the OSS project, README is an important document for sharing crucial information, such as installation and example usage of their software. README also contains information to promote development, such as how to contribute to the project. Although there are several guidelines for contents to be described in README, it is not clear whether developers follow them, and it is unknown whether they are applicable to all kinds of projects. In this thesis, in order to clarify README contents, I extracted the existing README contents, and compared contents between two different software types. As a result, there are 26 main contents in README. In particular, the most frequent contents are usage, install, and license. And there are contents not indicated in the guidelines. The contents provide new guideline when creating README. In addition, I clarified that the application software frequently contain contents to explain many functions compared to the library software, and the library software frequently contain installation and license compared to the application software. Differences in contents that focus on software type are guidelines for OSS project developers to write README contents according to their software type.

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1651008, March 15, 2018.

Keywords:

README files, Software documentation, Software ecosystem, Open source software, Association rule mining

目 次

図 目 次

表 目 次

1. はじめに

1.1 背景

オープンソースソフトウェア（OSS）とは，ソースコードが利用可能で，著作権保持者がソフトウェアを変更・配布するための権利を提供するライセンスに基づくソフトウェアである [7]．多くの OSS 開発プロジェクトは，ソースコードをはじめとするファイルを，共有 Web サービス GitHub [1]で公開する．OSS は GitHub を通して，複数の開発者の共同で開発されることも多い．

プロジェクトはソースコードと併せて，利用者とプロジェクトに興味を持つ外部の貢献者候補に向けて，README と呼ばれるドキュメントを公開する．README は OSS に限らず，ソフトウェア利用前の利用者に対してソフトウェアの機能やインストール方法，利用例等のソフトウェアを正しく使用する方法を伝えるためのドキュメントとして利用されている．特に GitHub では，README はプロジェクトのフロントページに公開され，プロジェクトの訪問者がプロジェクトについて知る最初の手がかりとなる重要なドキュメントであるため，他のソフトウェアドキュメントに比べ公開するプロジェクトが多い [2]．初めての利用者に向けたドキュメントであるため，利用方法は最も重要なことのみを簡潔に述べる傾向がある．OSS 開発プロジェクトは，開発を促進するためにプロジェクトへの貢献方法等も README に記述する．README に記述される項目の内，ソフトウェアの機能やインストール方法の説明は，類似プロジェクトの探索やインストールの自動化等の研究に利用されている [5][15]．README は利用者や開発者，研究者に利用される重要なドキュメントである．

しかし，README は内容の不完全さが指摘されている [8]．2017 年に GitHub が開発者 6,000 名に行ったアンケート調査 [2]でも，ソフトウェアの文書化は重要視されているが，頻繁に見落とされている工程であることが指摘された．実に 93 %の回答者が，ほとんどのドキュメントは不完全である，または陳腐化していると述べた．

[1]GitHub: https://github.com/
[2]Open Source Survey: http://opensourcesurvey.org/2017/

README 作成に慣れていない，初心者の OSS プロジェクト開発者に対して README 作成時の方針を与え，README の品質を向上させることを目的として，README の記述項目について説明したガイドラインが提案されている[3][4][5]．しかし README 作成に慣れた，成熟したプロジェクトの開発者がガイドラインを正しいと考え，ガイドラインに準拠した項目を記述しているか否かは明らかではない．またガイドラインが述べる項目が，あらゆる種別のソフトウェアプロジェクトに適用可能かは明らかでない．初心者の開発者に対して README 作成時の方針を与えるために，成熟したプロジェクトの開発者がガイドラインに準拠して README を記述しているか否か，また種別に応じて記述項目を変更しているか否かを調査する必要がある．成熟したプロジェクトの開発者の多くがガイドラインに準拠していない項目を記述していることが明らかになると，初心者の開発者に対して成熟したプロジェクトの開発者が記述している項目を新たな方針として与えることができる．また種別に応じて項目を変更していることが明らかになると，種別に応じた項目を方針として与えることができる．

1.2 目的

本論文の目的は，初心者の OSS プロジェクト開発者に対して，README 作成時の方針を与えることである．そのために，成熟したプロジェクトの開発者の記述する README を理想の README と捉え，ガイドラインに準拠した項目を記述しているか否かを明らかにする．成熟したプロジェクトの開発者がガイドラインに準拠していない場合，成熟したプロジェクトの開発者が記述している項目を初心者の開発者に対して新たな README 作成時の方針として与えることができる．また README に記述される項目は，ソフトウェアの種別に依存して異なると考えられることから，より具体的な作成時の方針のために，ソフトウェアの種別に応じた記述項目の違いを分析する．初心者の開発者に対して，彼らのソフト

[3]GitHub Help -About READMEs-: https://help.github.com/articles/about-readmes/

[4]Making READMEs readable - 18F Open Source Style Guide: https://open-source-guide.18f.gov/making-readmes-readable/

[5]O'Reilly Open Source Convention: OSCON, July 20 - 24, 2015 in Portland, OR https://conferences.oreilly.com/oscon/open-source-2015

ウェアの種別に応じた項目を方針として与えることができる．

本論文で行う 2 つの分析について述べる．

分析 1：README において一般的に記述されている項目の分析

README に記述されている項目がガイドラインに準拠しているか否かを明らかにすることを目的とし，成熟していると考えられる 43,991 件の JavaScript プロジェクトが公開する README を対象に，README に含まれる項目を抽出する手法を定義し，README 作成に慣れた開発者が記述する項目を分析する．

分析 2：ソフトウェア種別の README 記述項目の比較

ソフトウェアの種別に応じた記述項目の違いを明らかにすることを目的とし，成熟したプロジェクトの内，アプリケーションソフトウェアに分類される 2,788 件のプロジェクトと，ライブラリソフトウェアに分類される 1,870 件のプロジェクトが公開する README を対象に，記述される項目の構成の違いを分析する．

1.3 本論文の構成

本論文の構成は次のとおりである．続く 2 章では，本論文が対象とする，OSS プロジェクトにおける README について述べる．3 章では，本論文の分析で必要な情報の説明とその取得方法を述べる．4 章では，分析対象とするデータセットの説明と，分析 1，分析 2 についての分析方法および結果を述べる．5 章では，本論文の結果を補強する考察を述べる．6 章では，本論文が関連する，あるいは貢献できると考えられる研究分野について述べる．最後に 7 章で，本論文のまとめについて述べる．

2. オープンソースソフトウェアプロジェクトにおけるREADME

2.1 READMEの記述項目

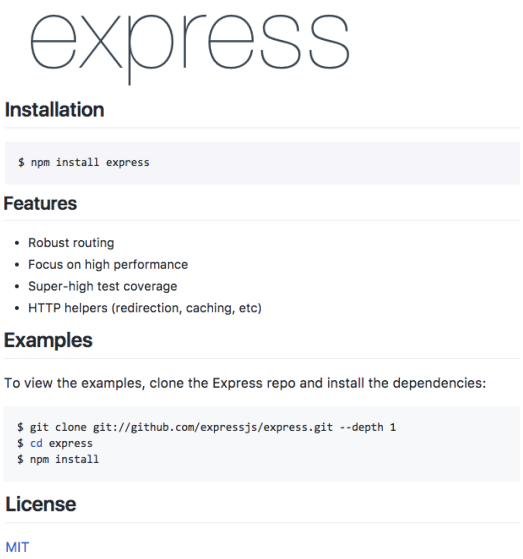
READMEは，ソフトウェアの利用者に向けて，ソフトウェアのインストール方法や利用例といったソフトウェアを正しく使用するための情報を伝えるドキュメントである．特に共同開発を行うOSS開発プロジェクトにおいては，新規開発者に向けて貢献方法を伝えるためにも利用される．

READMEの例として，Webフレームワークソフトウェア`express`[6]の2016年7月の時点におけるREADMEの一部を図1に示す．太字の見出しから，インストール方法を示す“Installation”や機能一覧を示す“Features”，利用例を示す“Examples”，ライセンスの種類を示す“License”といった項目が記述されていることがわかる．“Installation”はインストール方法をコマンドで記述しており，“Examples”はプロジェクトに保存されている利用例を確認する方法を記述している．別の例として，Web用ライブラリ管理支援ソフトウェアBower[7]の2016年7月の時点におけるREADMEの一部を図2に示す．同じく太字の見出しから，インストール方法を示す“Install”と利用方法を示す“Usage”，ライセンスの種類を示す“License”に加え，プロジェクトへの貢献方法を示す“Contributing”といった項目が記述されていることがわかる．“Contributing”は，貢献するための方法として不具合報告や新規機能の要求を挙げている．

READMEをはじめとするソフトウェアドキュメントは，内容が不完全である[8]．OSS開発者がREADMEをどのように作成するか確認するために，JavaScriptエコシステムの1つであるnpmに登録されるプロジェクトを対象にREADMEの変更履歴を確認した．2016年7月の時点でnpmから取得できたプロジェクトの内，READMEを利用していた143,933件のプロジェクトについて更新履歴を分析したところ，プロジェクトはREADMEを平均7回は更新していた．また`express`の

[6]expressjs/express: Fast, unopinionated, minimalist web framework for node.:https://github.com/expressjs/express

[7]bower/bower: A package manager for the web:https://github.com/bower/bower


express

Installation

```
$ npm install express
```

Features

- Robust routing
- Focus on high performance
- Super-high test coverage
- HTTP helpers (redirection, caching, etc)

Examples

To view the examples, clone the Express repo and install the dependencies:

```
$ git clone git://github.com/expressjs/express.git --depth 1
$ cd express
$ npm install
```

License

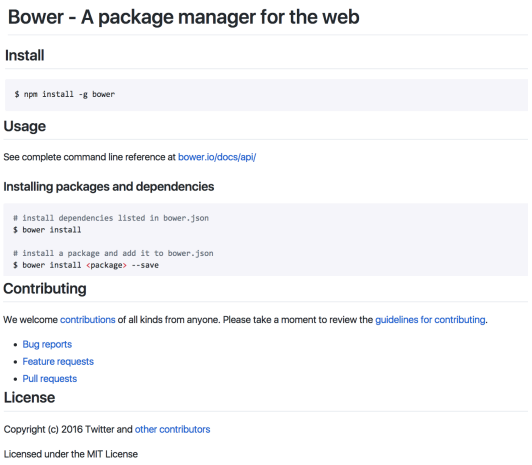
MIT

図 1　README の例（express の場合）

README について，プロジェクトが作成された 2010 年 1 月からデータ取得時の 2016 年 7 月までの更新履歴を調査した．図 3 は，express の README が，時間の経過とともに変更され，ファイルサイズが変化する様子を示している．すべての項目が最初から記述されているわけではなく，ソフトウェアのテスト方法を説明する “Test” を追加している．逆に，ソフトウェアの設定方法を示す “Settings” や貢献者の一覧を示す “Contributor” を README から削除する場合もあった．README は一度の変更では完成せず，試行錯誤されている．README を記述する方針があれば，開発者の負担を軽減できると考えられる．

2.2　README のガイドライン

初心者の OSS 開発者に対して作成支援の方針を与え，README の品質を向上させることを目的として，いくつかの組織が README に記述する項目を説明したガイドラインを公開している．表 1 は，README の項目を示唆する既存のガイドラインをまとめたものである．これらのガイドラインは次の情報源から得た．



Bower - A package manager for the web

Install

```
$ npm install -g bower
```

Usage

See complete command line reference at bower.io/docs/api/

Installing packages and dependencies

```
# install dependencies listed in bower.json
$ bower install

# install a package and add it to bower.json
$ bower install <package> --save
```

Contributing

We welcome contributions of all kinds from anyone. Please take a moment to review the guidelines for contributing.

- Bug reports
- Feature requests
- Pull requests

License

Copyright (c) 2016 Twitter and other contributors

Licensed under the MIT License

図 2 README の例（Bower の場合）

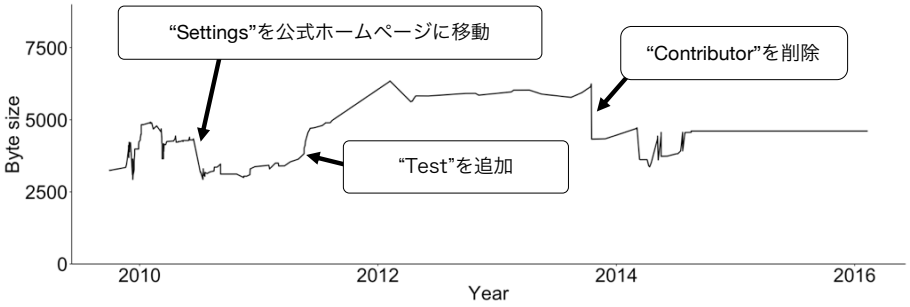


図 3 README ファイルサイズ [bytes] の変化（express の場合）

- GitHub[8]は OSS の共有 Web サービスである．GitHub では README はソフトウェアリポジトリのフロントページに表示するものと捉えられており，一般的な README に記述されているとされる項目を紹介している．

- 18F[9]はアメリカの政府機関の 1 つで，政府機関内における IT サービスの導入を支援する役割を担っている．18F は，読みやすい README は利用者や開発者がソフトウェアを理解することに役立つと考えており，OSS 開発を促進するため，読みやすい README とするために記述すべき項目を説

[8]GitHub Help -About READMEs-: https://help.github.com/articles/about-readmes/

[9]Making READMEs readable - 18F Open Source Style Guide: https://open-source-guide.18f.gov/making-readmes-readable/

表 1 README 記述のためのガイドライン

項目	GitHub	18F	OSCON2015	内容
Purpose	✓	✓	✓	プロジェクトの目的
Install	✓	✓	✓	インストール方法
Usage	✓	✓	✓	利用方法（コード例や初期設定）
Test		✓		プロジェクトのテスト方法
Limit			✓	プロジェクトの動作環境
License		✓	✓	ライセンスの種類
Contribute	✓	✓	✓	プロジェクトへの貢献方法
Support		✓	✓	プロジェクトからサポートを得る方法
Author	✓	✓		作者
Release history			✓	更新履歴

明している．

- OSCON2015[10]は，OSS 開発のための会議である．基調講演の講演者である Mikke Jang 氏は，プロジェクトが利用者を引き付けることができなかったのは，README の品質が悪いためだと説明した．

表 1 に示すように，OSS 開発プロジェクトにおける README には，初心者の開発者に向けたガイドラインが存在している．しかし，README 作成に慣れた，成熟したプロジェクトの開発者がガイドラインを正しいと考え，ガイドラインに準拠した項目を README に記述しているか否か明らかではない．初心者の開発者に対して README 作成の方針を与えるために，成熟したプロジェクトの開発者が README に記述している項目を知る必要がある．

表 1 に示すガイドラインは，すべてのプロジェクトに向けた汎用的な記述項目を提案している．しかし，プロジェクトのソフトウェアには様々な種類があり，種類ごとに必要となる項目は異なるため，すべてのプロジェクトが README に同じ項目を記述するわけではない．例えば再利用が多いプロジェクトは，他のそうでないプロジェクトに比べ，ライセンスに注意して README を作成する．機能の多いプロジェクトは，利用者に向けて機能の一覧を紹介する項目と，機能を

[10]O'Reilly Open Source Convention: OSCON, July 20 - 24, 2015 in Portland, OR https://conferences.oreilly.com/oscon/open-source-2015

素早く実装するために貢献者を募る項目を組み合わせたREADMEを作成する場合がある．ソフトウェアの種別に応じて，READMEの記述項目が異なるのであれば，種別に応じたガイドラインが必要である．

本論文では，初心者のOSS開発者にREADME作成時の方針を与えることを目的として，成熟したプロジェクトの開発者がガイドラインに準拠してREADMEを作成しているか否か分析する．また，ソフトウェア種別の記述項目の違いを明らかにする．

3. README記述項目とソフトウェア種別の取得方法

本論文で行う2つの分析では，README に記述されている項目を分析する．READMEの記述項目を分析するためには，プロジェクトからREADMEを，READMEから記述項目をそれぞれ取得する必要がある．README は，GitHub に公開されているプロジェクトから取得する．README の記述項目は，README における見出しから取得する．

3.1 READMEの取得

GitHub では，プロジェクトはREADME をプロジェクトリポジトリのルートディレクトリに置くことでフロントページに表示できる．GitHub は，文書を記述するための軽量マークアップ言語の1つである Markdown をはじめとして，様々な拡張子の README を利用可能である．本論文では，次節に示す項目抽出のため，Markdown 形式で記述された README を対象とする．

GitHub に公開されている Markdown 形式の README は，プロジェクトリポジトリの URL を入力として，以下の手順で取得できる．

(Step 1) “git clone *@URL*” コマンドを用いて，プロジェクトリポジトリのファイル一覧をダウンロードする．*@URL* はプロジェクトリポジトリの URL 表す．

(Step 2) リポジトリ内から，ファイル名が “README” で，拡張子がMarkdown 形式を表す “md”，“markdown” のファイルを検索する．

取得したREADME から，次節に示す方法で記述項目を取得する．

3.2 READMEの見出しによる項目の特定

開発者はREADME の可読性を向上させるために，各情報を要約した見出しを各情報の本文の前に記述する．図1では，インストール方法，機能一覧，利用例，

ライセンスの各情報の本文を，それぞれ “Installation”，“Features”，“Example”，“License” という見出しで区切って記述している．

本論文では，見出しには README に記述される内容の要約が現れることに着目し，分析 1 および分析 2 では，見出しを README の項目とみなして分析する．ただし，見出しは自然言語であるため，表記ゆれが見られる．例えば，インストール方法の項目を表すために，開発者らは，“How to Install?”，“installing”，“Installation” という見出しを利用する．そこで自然言語処理におけるステミング処理を利用して，これらの表記ゆれを正規化する．見出しに対して，以下の手順でステミング処理を行った際の出力を項目とする．

(Step 1) 英数字以外の記号（例：?，!等）を除去する．

(Step 2) 英文字において，すべての大文字を小文字に置き換える．

(Step 3) 見出しをトークンと呼ばれる自然言語処理における最小単位に分解する．英語においては，1 単語が 1 トークンに相当する．例えば，“how to install” は，“how”，“to”，“install” の 3 トークンに分解される．

(Step 4) ストップワードと呼ばれるそれ単体では意味がないとされる単語（例：what，how，to 等）を除去する．また数字だけで構成される見出しは，意味がないと考えたため，除去する．

(Step 5) 見出し語化という手法によりトークンごとの語形を統一する．例えば “installing” は原型である “install” となる．

上記の手順の結果，同じ形となった見出しを同一の見出しとみなす．例えば，ストップワードを含む “How to Install?”，語形が異なる “installing”，章番号を含む “1. Installation” という見出しは，すべて “install” という 1 つの項目に統合される．

取得した記述項目に基づき，ソフトウェア種別で共通している項目，していない項目を分析する．

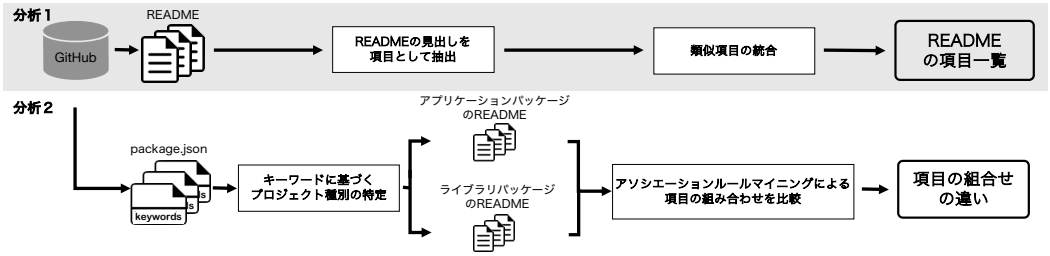


図 4 分析の全体像

4. README 記述項目の分析

本論文の目的は OSS プロジェクトの開発者に対して，README 作成時の方針を与えるために，以下の 2 つの分析から README に記述されている項目を明らかにする．

- 分析 1：README において一般的に記述されている項目の分析
- 分析 2：ソフトウェア種別の README 記述項目の比較

図 4 は，2 つの分析の全体像を示す．本章では，まず分析対象とするデータセットの説明を行い，その後，分析 1，分析 2 のアプローチと結果を述べる．

4.1 データセット

本論文では，OSS プロジェクトが公開する README の記述項目を分析するために，JavaScript エコシステムの 1 つである npm に登録される JavaScript プロジェクトを対象とする．

npm には，ライブラリ（例：`react`），フレームワーク（`express`），コマンドラインツール（例：`browserify` ），プラグインサポートツール（例：`grunt`，`glup`）といった種別のソフトウェアが含まれている．npm に登録されるプロジェクトは，検索で発見されるため，npm 用のメタファイルである package.json に，自身のプロジェクトの機能を説明するキーワードタグを任意の個数追加することができる．利用者は自身の希望する機能を持つプロジェクトを，キーワードを用いて検索でき

```
{
   "name": "express",
   "version": "4.16.2",
   "license": "MIT",
   "repository":    "expressjs/express",
   "keywords": [
          "express",
          "framework",
          "web"
   ]
}
```

図 5　package.json の一例

表 2　データセットの内容

スナップショット取得期間	2016/07/01 - 2016/07/15
抽出したプロジェクト	153,857 プロジェクト
フィルタリング後プロジェクト	43,911 プロジェクト

る[11]．2.1 節で具体的な例として示した `express` における package.json の一部を図 5 に示す．Web 用フレームワークソフトウェアである `express` は，“framework”，“web”，“express” といったキーワードを設定している．キーワードは分析 2 にて，ソフトウェア種別の特定に利用する．

表 2 に示すように，npm のリポジトリ [12]から，2016 年 7 月 1 日から 2016 年 7 月 15 日までの期間でアクセス可能であったプロジェクトのリポジトリ URL をすべて抽出し，153,857 件のプロジェクトを取得した．ただし，この取得方法は 2016 年後期に廃止されている [13]．取得したデータには 2008 年から 2016 年に作成されたプロジェクトまでが含まれていた．

そのうち，3 章で示したように，項目を取得するため，見出し属性を持つ Markdown 形式で記述された README を持つプロジェクトを抽出した．また，本論文では成熟したプロジェクトの開発者が作成した README の記述項目が，初心者の開発者に対する記述方針となると考え，記述項目を明らかにする．成熟した

[11]npm https://www.npmjs.com/

[12]https://registry.npmjs.org/-/all

[13]The npm Blog — Deprecating the /-/all registry endpoint: http://blog.npmjs.org/post/157615772423/deprecating-the-all-registry-endpoint

プロジェクトの開発者が作成したREADMEを対象とするため，次の条件を満たすプロジェクトを成熟したプロジェクトとみなし，本論文の分析対象とした．

1. 現在，試行錯誤している最中だと考えられるREADMEを除外するため，直近で更新のあったREADMEを除去する．本論文では，データセットの入手時点から1年以内にREADMEを更新したプロジェクトを除去した．

2. 未熟なプロジェクトはREADMEを試行錯誤する以前だと考えられるため，READMEの変更回数によって除去する．具体的にはREADMEを作成後一度も更新していないプロジェクトは除去する．

3. 長い時間試行錯誤しているプロジェクトを除去するため，プロジェクトが作成されてからデータセット入手時までの時間によって除去する．本論文では，データセットに入手時点から3年より以前に作成されたプロジェクトを除去した．

4. 分析を容易にするため，英語以外の言語（日本語，キリル文字，またはハングル）で記述されるREADMEを除去する．

5. 分析対象としているため，Markdown形式における見出し表現を利用していないREADMEを除外する．

前処理の結果，43,991件のプロジェクトが対象となった．

4.2 分析1：READMEに一般的に記述されている項目の分析

4.2.1 アプローチ

分析1では，データセットの43,991プロジェクトが公開するREADMEを対象に，記述される項目を明らかにする．

項目の抽出は3.2節に述べた手法を用いて行ったが，プロジェクトは似た内容の項目を別の表記で記述する場合があったため，類似項目を手動で統合した．次の3つの手順で項目を抽出・統合した．

***(Step 1)* README の見出しを項目として抽出**

3.2 節に述べた手法を用いて，43,991 件の README に含まれる見出しから項目を抽出する．対象とした見出しは，ノイズを防ぐため，抽象度の高いレベル 1（$h1$：大見出し），レベル 2（$h2$：中見出し）の見出しである．Markdown 形式では，`#`，`##`，`===`，`---` と記述することで，h1，h2 の見出しを記述することができるため，これらを正規表現により抽出する．ステミング処理には Python の `nltk` ライブラリを使用した．`nltk` ライブラリのトークン化処理は，ソフトウェアエンジニアリングデータセットに対して高い精度で行えることが知られている [12]．

***(Step 2)* 類似項目の統合**

プロジェクトは似た内容の項目を別の表記で記述する場合があったため，類似する内容が記述されている項目を手動で統合する．自然言語で記述されたソフトウェアドキュメントの内容を手動で分類することは，別の研究でも行われる手法である．Maalej らの研究 [9] では，複数の被験者が，API ドキュメントから手動で項目を抽出し，合意を取る方法で統合を行っている．本論文では，自動で抽出した見出しに対して，筆者の他に本論文の審査委員である 2 人を含めた 3 人で，各見出しの内容を手動で確認し，類似すると考えられる項目について 3 人の合意を得た上で統合する．

***(Step 3)* 項目の集計**

README で共通して記述されている項目を明らかにするために，手動による統合で得た項目に基づいて，README ごとの項目の記述の有無を集計する．集計結果は，各項目を記述しているプロジェクトの件数と，その件数のプロジェクト全体に対する割合 PR（Project rate）で表す．PR は，各項目を記述しているプロジェクトの件数 $projects$ と 43,991 件の全プロジェクトを表す $all\ projects$ を用いて，以下の式で計算される．

$$PR = \frac{projects}{all\ projects} \tag{1}$$

4.2.2 結果

Step1 では，3.2 節に述べた手法を用いて，43,991 件の README に含まれる見出しから 69,869 種類の項目を抽出した．Step1 で行った統合の例として，Install に統合した見出しの一部を表 3 に示す．また表 4 で示すストップワードトークンを持つ見出しを，統合後項目に示す項目に例外的に統合した．理由としては，例えば “How to” という見出しはストップワードの除去で失われてしまうが，利用方法を示す見出しであると考えられる．そのため “Usage” に統合した．なお，表 4 で示すストップワードトークンを 2 種類以上持つ見出しは，表 4 のより上行で示される方の項目に優先して統合した．

Step2 では，手動での統合により，69,869 種類の項目から 26 種類の項目を得た．ただし，69,869 種類の項目を全て確認することは困難であったため，69,869 種類の内，共通して記述されていると考えられる 30 件以上のプロジェクトに記述されている項目 222 種類のみを対象として，手動での内容確認・統合を行った．30 件未満のプロジェクトに記述されている項目については，222 種類の項目で使用される単語を含んでいるか否かを条件として自動で統合した．例えば，“Basic Usage” や “Usage Code” は，Usage という単語を含むため，“Usage” に統合された．222 種類で使用される単語を 2 種類以上持つために，統合先が複数存在する項目は，最も長い単語のみを採用して統合している．長い単語を採用する理由としては，より項目の内容を表していると考えられるためである．例えば “Get Benchmark” は，Get という単語から “Install” に統合される一方で，Benchmark という単語から “Benchmark” にも統合されうる．この場合，最も長い単語である Benchmark を採用し，“Get Benchmark” は “Benchmark” に統合する．Get という単語よりも Benchmark という単語のほうが，項目の内容を表していると考えられるためである．222 種類の項目で使用される単語を全く含まないために自動的に統合されなかった，30 件未満のプロジェクトにしか記述されない項目は，共通して記述されている項目ではないと考え，結果から省いた．

結果として得られた 26 種類の項目を，Step3 で集計した結果を表 5 に示す．表中のチェックマークは表 1 のガイドラインで説明されている項目であることを示す．ただし明確に説明されているとはいえない項目は，括弧付きのチェックマー

表 3 見出しの統合例

見出し	統合後項目
How to Install?	Install
installing	
Install it	
Installation	
INSTALL	
1. Installing	

クとした．見出しには，統合される前の見出しの一部を示す．また Step2 において対象とした，30 件以上のプロジェクトに記述されている 222 種類の項目と，手動統合後の項目との対応表を付録に示す．

表 5 に示す 26 種類の項目についてそれぞれ説明する．表 1 に示された項目である “Purpose”，“Install”，“Usage”，“Test”，“Limit”，“License”，“Contribute”，“Support”，“Author” はすべて，記述しているプロジェクトが存在した．ただし，各項目に記述される内容の違いから，いくつかの項目は複数の種類に分類した．

表 1 に示される利用方法は，記述される内容の違いから，“Usage”，“API”，“Option”，“Document”，“Demo”，“Error” の 6 種類が存在した．“Usage” は，ソフトウェアの機能の中でも，最も主要な機能を実行するまでの手順を述べている項目を分類した．“API” には，API やメソッドのような，ソフトウェアに対して行えるアクションについて述べている項目を分類した．“Option” は，Option や Configure のような，ソフトウェアが参照するパラメータの設定方法について述べている項目を分類した．“Document” は，README より詳細なドキュメントを得られる公式ホームページなどへのリンクを示す項目を分類した．“Demo” は，ソフトウェアの動作や出力を，画像や動画を用いて説明している項目を分類した．“Error” は，ソフトウェアが扱わない例外に対する処理を説明している項目を分類した．

同じくプロジェクトの更新履歴は，記述される内容の違いから，“Release history” と “Roadmap” の 2 種類に分類した．“Release history” は，プロジェクトの

表 4 ストップワード除去の例外

ストップワードトークン	統合後項目	見出し例
to do	Todo	To do
how	Usage	How to
who	Author	Who are we?
from		From
more	Document	More....
other		Others
about	Overview	ABOUT
what		What the... ?
that		What is that?
can		What can I do?

アップデートごとの変更点を記述した項目を分類した．“Roadmap” は，プロジェクトにおいて現時点で達成済みの作業と今後達成する予定を説明した項目を分類した．

プロジェクトの目的は，“Purpose”，“Overview” に記述される．“Purpose” は，プロジェクトが解決する目的のみを記述する項目を分類した．“Overview” には，プロジェクトの目的を含め，プロジェクトの機能や有用性などプロジェクトの包括的な説明を記述する項目として分類した．

表 1 に示す項目である “Install”，“Test”，“Limit”，“License”，“Contribute”，“Support”，“Author” は，各項目に記述される内容に大きな違いは見られなかったため，複数の項目には分類しなかった．“Install” には，プロジェクトのダウンロード方法やインストールコマンド，インストールに必要な外部ライブラリを記述した “How to Install?” や “Getting Started” をはじめとして，インストールに用いるツール別に説明した “npm”，“node.js” が該当する．“Contribute” には，プロジェクトへの貢献方法が記述される項目を分類した．プロジェクトが利用する不具合報告サービスへのリンクやプルリクエストのテンプレート等が記述される．“Support” には，プロジェクトの開発者に，質問を行う方法を記述した項目を分

表 5 README における共通項目の分類

Rank	項目	プロジェクト数 (PR)	ガイドラインの有無	内容	見出し
1	Usage	26758 (60.83%)	✓	ソフトウェアの利用方法	Usage, Basic Usage, How to Use, Use Case, Methods, Screenshot, Examples, Quick Examples, Tips, Syntax, Sample, Hello World, Grunt tasks
2	Install	26142 (59.43%)	✓	ソフトウェアのインストール方法	How to Install?, Installation, Getting Started, Get started now!, To setup, Download, Initialization, Instructions, npm, node.js
3	License	15932 (36.22%)	✓	ソフトウェアに適用されるライセンスの種類	LICENSE, MIT License, Unlicense, License, Copyright, Legal
4	API	10675 (24.27%)	(✓)	ソフトウェアの API 一覧	API, API References, API Documents, Command Line, CLI, Build, Events, Constructor, Action, To run, Objects, Interface, Commands, Function, Execute
5	Option	10459 (23.78%)	(✓)	ソフトウェアのオプション一覧	Options, Format, Style, Parameter, Configure, Import, Custom, Compatibility, Browser, Config, Client, Server, Module, Promise, Util, Class, Variables, Router
6	Release history	5874 (13.35%)	✓	プロジェクトの更新履歴	Release History, ChangeLog, Change logs, Version history, Versions, Release Notes
7	Contribute	4938 (11.23%)	✓	プロジェクトへの貢献方法	How to Contribute, Contribution Guides, Development, Donations, CONTACT, Hacking
8	Test	4374 (9.94%)	✓	ソフトウェアのテスト方法	Run test, testing, To test, How to Test
9	Todo	4293 (9.76%)		プロジェクトの残り作業一覧	TODO, Todos, To-Do List, To-do soon, Task, In the future ..., The future, Requirements, Coming soon!
10	Overview	4219 (9.59%)	(✓)	プロジェクトがすること	Overview, Summary, Synopsis, Description, Introduction, Why?, What's this?, About The Name
11	Status	3491 (7.94%)		テストカバレッジなど品質に関する情報	Build Status, Current Status
12	Document	3384 (7.69%)	(✓)	ソフトウェアのより詳細な利用方法	Documentation, Doc, Doe, Notes, Info, Information, TL;DR, Notice, Detail, See also
13	Author	2607 (5.93%)	✓	プロジェクトの作者	Authors, About the Author, who am i?, Credits, Backers, Contributors, Other Contributors, Resources
14	Support	1555 (3.53%)	✓	プロジェクトからサポートを得る方法	Supports, FAQ?, Help, Troubleshooting, Questions
15	Feature	1379 (3.13%)		ソフトウェアの機能一覧	Key Features, Attributions
16	Relate	1297 (2.95%)		プロジェクトに関連するプロジェクトの情報	Related, Related Projects, Link, Inspirations, Alternatives, Source, Other libraries
17	Issue	1039 (2.36%)		プロジェクトが抱えている問題	Issues, Known issues, Problems, Warning, Caveats, Bugs
18	Demo	839 (1.91%)	(✓)	ソフトウェアの出力例	Demo, Example Output, Codepen demo, Result
19	Purpose	821 (1.87%)	✓	プロジェクトの目的	Purpose, Goals, Solution, Motivation, Background, Concepts, Key Ideas, Philosophy, Rationale
20	Refer	772 (1.75%)		プロジェクトが参考にした文献や謝辞	References, Thanks, Special Thanks, Acknowledgements
21	Limit	772 (1.75%)	(✓)	プロジェクトの制約	Limitations, Implementation, Validations, Browser Support
22	Error	456 (1.04%)	(✓)	ソフトウェアが出力するエラーケース	Error, Filters
23	Roadmap	281 (0.64%)	(✓)	プロジェクトの進捗状況	ROADMAP, Road Map, Work Progress
24	Table content	275 (0.63%)		README の目次	Table of Contents, Contents
25	Model	225 (0.51%)		プロジェクトの設計	Models, MVC Model, UI Model
26	Benchmark	200 (0.45%)		ソフトウェアのベンチマーク	Benchmarks, Benchmark / Performance, Performance, Performance wins

類した．開発者のメールアドレスや質問送信用の外部 Web サービスへのリンク等が記述される．他に，“License” には，ライセンスの名前や規約を記述した項目を，

“Limit” には，ソフトウェアが動作可能な環境を記述する項目を “Author” にはプロジェクトの開発に関わる開発者や貢献者，出資者を記述した項目を，“Test” にはテストコマンド等のテスト方法を示す項目を分類した．

表 1 に示されていない項目として，“Todo”，“Status”，“Feature”，“Relate”，“Issue”，“Refer”，“Model”，“Benchmark” が存在した．“Todo” にはプロジェクトの今後の作業を記述した項目を分類した．“Issue” は “Todo” に似ているが，“Issue” には予想外に起きた問題を，利用者への注意喚起を目的として記述している項目を分類した．“Status” には，テストカバレッジ等のプロジェクトの品質を示した項目を分類した．“Feature” には，提供する機能が複数あるプロジェクトが，各機能の要約が記述した項目を分類した．“Relate” には，プロジェクトの開発者が別に開発中のプロジェクトの紹介や，類似プロジェクトと比較したプロジェクトの違いを示した項目を分類した．“Refer” には，プロジェクトが開発中に参考にした文献やプロジェクトについて，紹介あるいは謝辞を述べた項目を分類した．“Model” には，ソフトウェアの設計を述べた項目を分類した．“Benchmark” には，ソフトウェアの処理速度等のベンチマークを記述した項目を分類した．

Hassan ら [5] や Zhang ら [15] は，README に記述される項目の内，ソフトウェアの機能やインストール方法の説明を用いて，類似プロジェクトの探索やインストールの自動化等の研究に利用されているが，README にはソフトウェアの機能やインストール方法以外の項目も記述されている．

README に記述される項目を利用した研究が行われている．Hassan ら [5] は，“Overview” のようなプロジェクトの機能を説明する項目から，特徴的な語句を抜き出し，類似プロジェクトの探索を行った．Zhang ら [15] は，“Install” のようなプロジェクトのインストール方法を説明する項目から，インストールコマンドを抜き出し，インストールの自動化を行った．本結果が示す分類は，README に含まれる項目を明らかにしたものであり，“Overview” や “Install” 以外の項目を利用した研究を支援すると考えられる．また本結果は成熟していると考えられるプロジェクトから得られたものであるため，初心者の OSS 開発者は README を作成する際，本結果が示す分類に基づいて，記述する項目について考えることができる．

README に共通して記述されている項目は 26 種類あり，ガイドラインに準拠した項目が記述されていた．また，ガイドラインに記述されている項目を示すために複数の方法が取られており，ガイドラインに記述されていない項目も存在した．

次に，項目が記述されている割合に着目する．データセットに含まれる README において，最も頻繁に記述される項目は “Usage”，“Install”，“License” であった．項目に該当する見出しの内，最も記述されていた見出しは，それぞれ”Usage”（54.82%），”Installation”（38.23%），”License”（79.61%）であった．ここで括弧内の割合は，各項目を記述しているプロジェクトの内，同じ見出しをしているプロジェクトの割合である．つまり統合した結果，“Usage” と判定された見出しの内，54.82%が”Usage” という見出しを利用して “Usage” を記述していた．

3 つの項目は最も頻繁に記述されているが，すべてのプロジェクトが記述しているわけではない．README に記述しない理由を説明するいくつかの事例を発見した．ひとつは README から独立した外部ファイルに記述している場合である．例えば “License” の内容は，“LICENSE” ファイルとして管理される場合がある．GitHub は，ライセンスの情報は “LICENSE” ファイルとして管理することを推奨している[14]．ライセンスの情報を README には記述せず，“LICENSE” ファイルだけに記述しているプロジェクトは 17,605 件（PR = 40.02%）存在し，README および “LICENSE” ファイルに記述しているプロジェクトは 10,548 件（23.98%）存在した．すなわち，ライセンスの情報は README には記述せず，“LICENSE” ファイルとして管理する方法が多数派であった．

もうひとつはバッジの利用である．バッジとは，サービスの種類によって，ライセンスやテストカバレッジ，プロジェクトのバージョンといった様々な情報を簡潔に示すための画像である [10]．図 6 の例では，プロジェクトに適用されているライセンスが MIT ライセンスであることを示している．このバッジは “License”

[14]Licensing a repository - User Documentation https://help.github.com/articles/licensing-a-repository/

図 6 バッジの一例（ライセンスの種類を示す場合）

の内容に相当するため，このバッジの利用者は README に “License” の項目を記述しない場合がある．ただ，図 6 のライセンス表示用のバッジを利用しているプロジェクトは，157 件（PR = 0.36%）のみであったため，ライセンス表示を利用しているプロジェクトは．“LICENSE” ファイルとして管理するプロジェクトよりも少ない．

> README に最も頻繁に記述される項目は “Usage”，“Install”，“License” である．ただし “License” は README ではなく，外部の “LICENSE” ファイルに記述される方が多い．

最後に，記述しているプロジェクトが少ない項目に着目する．プロジェクトへの貢献方法を述べる “Contribute”，プロジェクトに現状残る作業を説明する “Todo” や “Issue” といった項目は，プロジェクトへの開発者あるいは貢献者候補へ向けた項目だと考えられる．一方で，“Usage” や “Install” は，ソフトウェアを導入し，利用するまでの手順が記述されているため，ソフトウェアの利用者のためのものであると考えられる．“Contribute”，“Todo”，“Issue” は，“Usage” や “Install” に比べ，記述しているプロジェクトが少ない．また，“Contribute” は “License” と同じく，README とは独立した外部ファイル（“CONTRIBUTING” ファイル）として保守されているケースがあったが，1010 件に限られた．多くのプロジェクトが，README はソフトウェアの利用者のために記述するものとして捉えていると考えられる．

“Usage”，“Install” 等の利用者に向けた項目が頻繁に記述されている一方で，“Contribute” 等の貢献者を獲得するための項目は比較的少なく，README はソフトウェアの利用者のために記述される．

他に記述しているプロジェクトが少ない項目として，“Overview”，“Author”，“Support” がある．3 つの項目は，表 1 で示したガイドラインに記載されているにも関わらずほとんど記述されていない．ガイドラインが考えているほど，重要だと考えていないプロジェクトが多いことが考えられる．しかし “Overview” は，見出しを利用して内容を記述しないケースも見られた．見出しを利用していない場合，本分析の手法では項目を記述しているか否かを判別できない．本論文の今後の発展として，見出しを利用しない項目についての分析が挙げられる．

4.3 分析 2：ソフトウェア種別の README 記述項目の違い

4.3.1 アプローチ

分析 2 では，ソフトウェア種別に応じた記述項目の違いを明らかにする．

プロジェクトに設定されているキーワードを用いてソフトウェア種別を特定し，アソシエーションルールマイニングを用いて記述される項目の違いを分析する．次の 2 つの手順でソフトウェア種別の特定と，項目の違いを分析する．

***(Step 1)* キーワードに基づくソフトウェア種別の特定**

43,991 件のプロジェクトが持つメタファイル package.json からキーワードを抽出し，キーワードに基づいてソフトウェア種別を *Application* ソフトウェア，*Library* ソフトウェア，そしてどちらにも属さない *Other* ソフトウェアの 3 つに識別した．3 つの内，分析対象としたソフトウェア種別は，*Application* ソフトウェアと *Library* ソフトウェアの 2 種類である．識別に用いるキーワードは Wittern らの結果を参考にした [13]．Wittern らは，npm で管理されるソフトウェアを分析し，*Application* ソフトウェアと *Library* ソフトウェアだと考えられるプロジェクトが

それぞれ設定しているキーワードを明らかにしている．種別の特定に用いたキーワードを表 6 に示す．

***(Step 2)* アソシエーションルールマイニングによる記述されている項目の比較**

分類した 2 種類のプロジェクトが公開する README において頻繁に記述されている項目を分析した．対象とする項目は，分析 1 の結果である 26 種類の項目を利用する．

記述している項目の特定には，アソシエーションルールマイニング [14, 17, 4] を利用した．アソシエーションルールマイニングは，多数のアイテムの組み合わせから 2 つ以上のアイテムの関連性をアソシエーションルールとして抽出する手法である．アイテムは一般的には商品などを指すが，本論文では記述項目をアイテムとして扱う．アソシエーションルールは，次のように事前条件 C（= Contents）と事後条件 St（= Software type）で表される．C を README の記述項目（例：“Install”，“License”）とし，St はソフトウェア種別（*Application* または *Library*）とする．

$$\{C\} \Rightarrow \{St\} \tag{2}$$

抽出したルールを評価するために，support，confidence，lift という 3 つのメトリクスを利用する．support はすべてのルールに事前条件（C）と事後条件（St）の両方が存在するルールの割合として定義する．

$$support(\{C\} \Rightarrow \{St\}) = \frac{\sigma(C \cap St)}{|St|} \tag{3}$$

confidence は，事前条件（C）と事後条件（St）の両方が事前条件（C）を有するルールに存在するルールの割合である．

$$confidence(\{C\} \Rightarrow \{St\}) = \frac{support(\{C\} \Rightarrow \{St\})}{support(C)} \tag{4}$$

lift は，事後条件（St）を有するルールに事前条件（C）と事後条件（St）が存在するルールの倍率である．

$$lift(\{C\} \Rightarrow \{St\}) = \frac{confidence(\{C\} \Rightarrow \{St\})}{support(St)} \tag{5}$$

アソシエーションルールマイニングは，Python の `Orange` [3] ライブラリを利用して実装した．

入力として与えるのは，項目の有無を表すベクトル集合と，属するソフトウェア種別を表すベクトルである．項目の有無を表すベクトル集合は，*Application* ソフトウェア，*Library* ソフトウェアのいずれかに特定されたプロジェクトが公開する README において，26 種類の項目の有無を “1”（項目が記述されている）あるいは “0”（項目が記述されていない）の 2 値で表現したものである．例えば，プロジェクトに “Usage” が記述されているならば，“Usage” の有無を表すベクトルは “1” となる．属するソフトウェア種別を表すベクトルは，*Application* ソフトウェアに属することを示すベクトルと *Library* ソフトウェアに属することを示すベクトルの 2 種類があり，どちらも “1”（プロジェクトが属している）あるいは “0”（プロジェクトが属していない）の 2 値で表現したものである．例えば，*Application* ソフトウェアに属するプロジェクトであれば，*Application* ソフトウェアに属することを示すベクトルが “1” となる．

出力として得るのは，アソシエーションルールである．ソフトウェア種別に記述項目を比較するために，事前条件が README の記述項目の組み合わせであり，事後条件がソフトウェア種別であるルールのみを抽出する．

抽出したアソシエーションルールは 2 種類の形式で示す．1 つはアソシエーションルールの一覧を表で示す．もう一つの表現は，抽出したルールのグラフ化である．図 7 は，1 つのルールがどのようなグラフに変換されるか示している．この例では，ノードに入る各エッジが “Usage”，“Install”，“License”，“Test” のような同時に記述されている項目の集合を示し，ノードから出るエッジがソフトウェア種別である *Library* を示す．ノードの濃淡は lift の大きさを表す．薄ければ低く，濃ければ高い．例のノードは，本分析の結果抽出したルールにおける最大の lift である 1.49 を示す濃さとなっている．またグラフの描画では `Fruchterman Reingold` アルゴリズムを用いたため，項目はより頻繁に記述されているソフトウェア種別に近づくように描画される．

4.3.2 結果

Step1では，キーワードに基づいて，2,788件の *Application* ソフトウェアと1,870件の *Library* ソフトウェアを抽出した．表6に種別の識別に用いたキーワードと，種別のプロジェクト数を示す．また，ソフトウェア種別を特定した際の例外について説明する．Wittern らの結果では，キーワード “gruntplugin” を持つプロジェクトも *Application* ソフトウェアに分類される．しかし，キーワード “gruntplugin” を持つプロジェクトは，他のプロジェクトとは大きく異なる特有のREADMEの記述方法が発見されたため，*Application* ソフトウェアへは分類せず分析の対象外とした．例えばキーワード “gruntplugin” を持つプロジェクトは，プラグインサポートツール `grunt` で利用されるプラグインである．`grunt` ではプラグインを利用することを “Task” という単語で表している．“Task” は一般的にプロジェクトに残る作業を説明する “Todo” の項目で利用される．このような一般的なプロジェクトとは異なる意味で利用される特有の単語が見られたため，本分析では対象としなかった．また，ノイズを除くため，両方のソフトウェア種別のキーワードを含む305件のプロジェクトを無視した．例えば，`express-session-json`[15] プロジェクトは `express` プロジェクトにおいてjsonファイルの扱いを支援するプロジェクトである．そのためキーワード “express” を持つが，同時にjsonファイルを扱うためキーワード “file” も持つ．

Step2では，アソシエーションルールマイニングを用いて，2種類のREADMEにおいて頻繁に記述されている項目の集合を示す36ルールを抽出した．ただし，頻繁でないルールを除外するために，supportは0.03，confidenceは0.03，liftは1を最低値としてフィルタリングを行っている．36ルールをliftでソートした結果を表7に示す．また，36ルールを描画した結果を図8に示す．

表7の結果から，2種別のプロジェクトが公開するREADMEにおいて記述項目の違いを比較する．

表1に示された項目である “Install” と “License” は，*Library* ソフトウェアが *Application* ソフトウェアに比べて頻繁に記述している．*Library* ソフトウェアは，*Application* ソフトウェアに比べて，他のソフトウェアで頻繁に再利用されるた

[15] express-session-json https://www.npmjs.com/package/express-session-json

表 6 *Application* ソフトウェアと *Library* ソフトウェア

application packages

キーワード	“gulpplugin”, “express”, “react”, “authentication”
プロジェクト数	2,788 プロジェクト

library packages

キーワード	“util”, “array”, “buffer”, “string”, “file”
プロジェクト数	1,870 プロジェクト

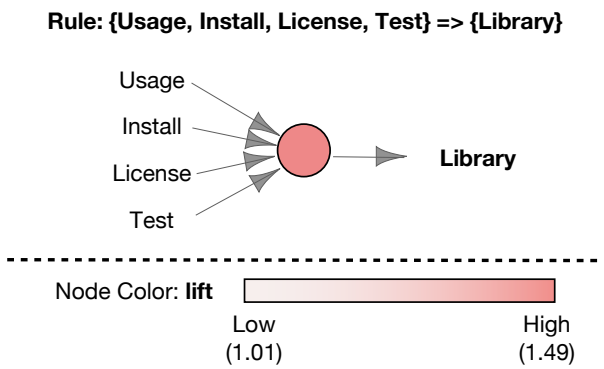


図 7 アソシエーションルールのグラフ化例

め，インストール方法やライセンスの種類の共有を重視しているためだと考えられる．また *Library* ソフトウェアは *Application* ソフトウェアに比べて，プロジェクトの品質を証明するためにテストカバレッジやテストコマンドを説明する項目である “Status”，“Test” を記述している．インストール方法やライセンスと同じく頻繁に再利用されることが理由で，*Library* ソフトウェアが品質を重視していることは十分考えられる．

表 7 アソシエーションルール（application vs. library）

Id	項目	⇒	ソフトウェア種別	support	confidence	lift
1	{Usage,Install,License,Test}	⇒	{Library}	0.04	0.60	1.49
2	{Install,License,Status}	⇒	{Library}	0.03	0.59	1.48
3	{Install,License,Test}	⇒	{Library}	0.05	0.59	1.47
4	{License,Status}	⇒	{Library}	0.04	0.58	1.44
5	{Usage,License,Test}	⇒	{Library}	0.04	0.58	1.43
6	{License,Test}	⇒	{Library}	0.05	0.57	1.42
7	{Install,Status}	⇒	{Library}	0.04	0.56	1.41
8	{Usage,Install,Test}	⇒	{Library}	0.05	0.56	1.39
9	{Usage,Status}	⇒	{Library}	0.04	0.54	1.35
10	{Install,Test}	⇒	{Library}	0.06	0.54	1.35
11	{Install,License,API}	⇒	{Library}	0.06	0.54	1.34
12	{Usage,Test}	⇒	{Library}	0.06	0.54	1.34
13	{Status}	⇒	{Library}	0.05	0.53	1.33
14	{Test}	⇒	{Library}	0.07	0.53	1.31
15	{Usage,Option}	⇒	{Application}	0.14	0.76	1.27
16	{License,Option}	⇒	{Application}	0.08	0.75	1.25
17	{Install,API}	⇒	{Library}	0.09	0.50	1.25
18	{Install,Option}	⇒	{Application}	0.12	0.74	1.24
19	{Option}	⇒	{Application}	0.19	0.73	1.21
20	{License,API}	⇒	{Library}	0.07	0.47	1.18
21	{Install,License}	⇒	{Library}	0.16	0.47	1.17
22	{Usage,Install,Releas History}	⇒	{Library}	0.03	0.46	1.16
23	{Install,Author}	⇒	{Library}	0.03	0.46	1.16
24	{Install,Releas History}	⇒	{Library}	0.04	0.46	1.14
25	{Todo}	⇒	{Application}	0.04	0.68	1.13
26	{API}	⇒	{Library}	0.11	0.45	1.13
27	{Usage,Overview}	⇒	{Application}	0.04	0.67	1.11
28	{License}	⇒	{Library}	0.20	0.44	1.10
29	{Install,Overview}	⇒	{Application}	0.04	0.65	1.09
30	{Overview}	⇒	{Application}	0.05	0.65	1.08
31	{Author}	⇒	{Library}	0.04	0.43	1.07
32	{Install}	⇒	{Library}	0.25	0.42	1.05
33	{Releas History}	⇒	{Library}	0.05	0.42	1.04
34	{Contribute}	⇒	{Library}	0.04	0.41	1.03
35	{Usage}	⇒	{Application}	0.41	0.61	1.02
36	{Document}	⇒	{Application}	0.05	0.61	1.01

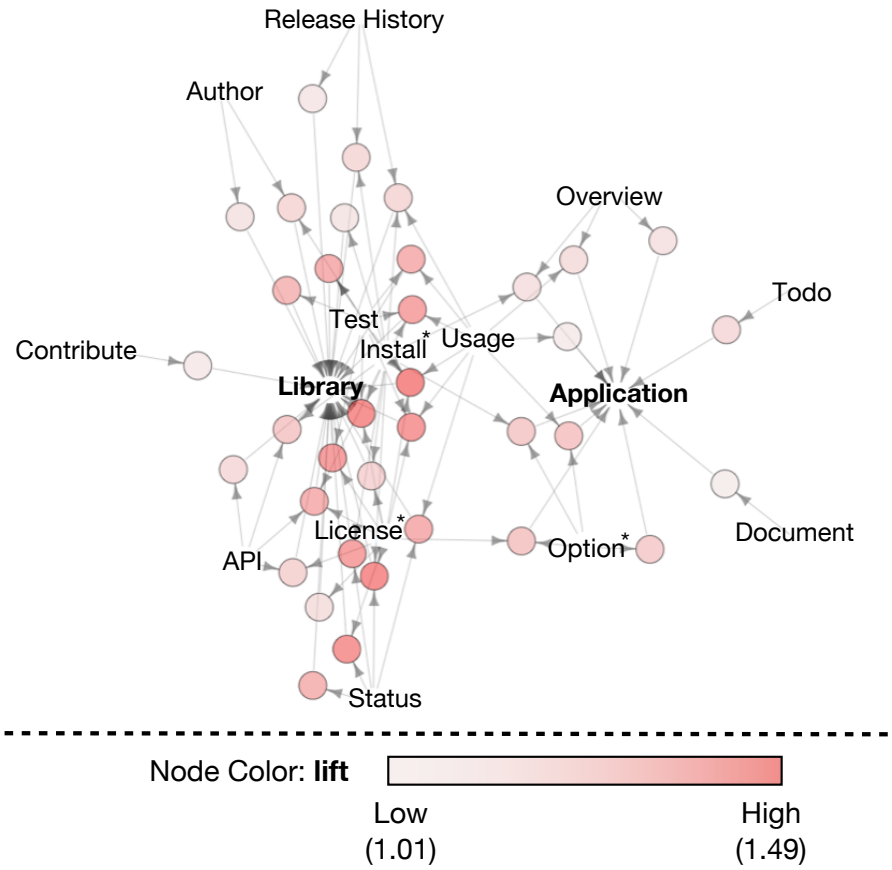


図 8　アソシエーションルールのグラフ

再利用される *Library* ソフトウェアは，*Application* ソフトウェアよりも，インストール方法とライセンスの種類を重視する．

Application ソフトウェアのルールに着目する．*Application* ソフトウェアは，*Library* ソフトウェアに比べ，表 1 に示された項目である “Option” がより頻繁に記述されている．*Application* ソフトウェアは，*Library* ソフトウェアに比べ提供する機能が多いため，可読性向上を目的として，主要な機能以外は “Option” に記述することが理由として考えられる．また *Application* ソフトウェアは *Library*

ソフトウェアに比べ全機能を外部サイトにより説明する “Document” が記述されている．

多機能な *Application* ソフトウェアは，*Library* ソフトウェアよりも，機能一覧を説明する項目を記述する．

本分析で明らかにしたソフトウェア種別に応じた記述項目の違いは，OSS 開発者に自身のソフトウェア種別に応じた README を作成する方針を与える．

4.3.3 評価方法

分析 2 では，ソフトウェア種別に重視している項目が異なると結論づけた．分析 2 の結論の妥当性を評価するために，README 記述項目に基づいたソフトウェア種別の予測を行う．項目に基づいてソフトウェア種別が特定可能であれば，ソフトウェア種別に重視している項目が異なるという結果は妥当であると言える．

予測は次の 3 つの手順で，教師あり機械学習を用いて行う．

***(Step 1)* データセットの準備**

教師あり機械学習を用いて，ソフトウェア種別を予測し，予測精度を評価するために，訓練・評価用のデータセットを準備する．データセットは，分析 2 で用いた表 6 に示す *Application* ソフトウェアと *Library* ソフトウェアを用いる．ただし，記述している項目が少ない README が，*Application* ソフトウェアあるいは *Library* ソフトウェアのどちらに属するプロジェクトが公開する README であるか予測することは困難であると考えられる．例えば，利用方法を記述した項目である “Usage” のみしか記述されていない README は，どちらに属するプロジェクトが公開する README であるか予測することは難しい．予測精度の向上のため，記述している項目の種類が少ない README をデータセットから省く．本評価では，5 種類を閾値とし，5 種類未満の項目しか記述されていない README を除去する．また学習精度に差が出ないよう，データセットに含まれる *Application* ソフトウェアと *Library* ソフトウェアのサンプル数をランダムサンプリングで統

一する．

***(Step 2)* 教師あり機械学習によるソフトウェア種別の予測**

データセットを入力として，*Application* ソフトウェアあるいは *Library* ソフトウェアのどちらに属するプロジェクトが公開する README であるか予測する分類器を用意し，項目に基づいてソフトウェア種別が特定可能か否かを検証する．分類器の訓練および予測精度の評価のために，交差検証を行う．

交差検証は，入力とするデータセットの内，一部を分類器の訓練用として用い，残りを予測精度の評価用として用いる検証方法である．交差検証の中でも，k-fold 法を採用した．k-fold 法は，データセットを k 個に分割し，内 k-1 個が訓練用，1 個が評価用のデータセットとなる．出力は，k 種類の訓練用データセットそれぞれで訓練したときの分類器に，評価用のデータセットを入力した際の予測の正解率 *Accuracy rate* となる．*Accuracy rate* は，評価用のデータセットの個数 *test projects* に対する予測結果が正解であった個数 *accuracy projects* の割合である．

$$Accuracy\ rate = \frac{accuracy\ projects}{test\ projects} \tag{6}$$

***(Step 3)* 予測精度の検定**

ソフトウェア種別に重視している項目が異なるか否か確認するために，分類器の予測精度の検定を行う．

分析 2 では，プロジェクトが *Application* ソフトウェアあるいは *Library* ソフトウェアの 2 種類に分類されるため，二項検定する．

4.3.4 評価結果

Step1 では，455 件の *Application* ソフトウェアと 455 件の *Library* ソフトウェア，合計 910 件のプロジェクトを対象とした．5 種類以上の項目を記述しているプロジェクトは 1096 件あり，内訳は *Application* ソフトウェアが 641 件，*Library* ソフトウェアが 455 件であったため，*Application* ソフトウェアから 455 件をランダムに抽出した．

Step2 では，k=5 とした交差検証を行い，5 つの *Accuracy rate* を得た．5 つの *Accuracy rate* の内，中央値は 62.6%，最大値は 71.4%，最小値は 56.0%であった．

Step3 では，予測結果の中央値に偏りがあるか二項検定を行った．k=5 とした交差検証では，182 回の試行を行っており，各試行は独立である．中央値は 62.6%であるため，182 回中 114 回予測に成功している．結果として，p<0.01 の片側検定で有意な差を確認した．README の記述項目は，ソフトウェア種別によって記述項目が異なると考えられる．

README の記述項目はソフトウェア種別によって，重視している項目が異なると考えられる．

5. 考察

5.1 時系列による記述項目の変化

分析2では，ソフトウェア種別に，データセット取得時に最新であったREADMEを対象に記述項目の違いを分析した．一方で，READMEは試行錯誤されていることを図3で示した．ソフトウェア種別の記述項目の違いがどのような時期に発生したのか知ることは，OSS開発者に今後READMEを作成していく方針を与えられる可能性がある．

ソフトウェア種別の記述項目の違いがどのような時期に発生したか分析することを目的として，プロジェクトが初めて作成したREADME（初期README）とデータセット取得時に最新であったREADME（最新README）の間の記述項目の変化をソフトウェア種別に比較する．本考察における分析の結果は，分析2の結論を補強するものであった．

5.1.1 アプローチ

変更履歴を抽出し，項目ごとに該当する変化のパターンを明らかにする．次の2つの手順で，プロジェクトの履歴から初期READMEを取得し，最新READMEと比較して項目の変化のパターンに分類した．

***(Step 1)* 更新履歴から初期README を取得**

初期READMEから最新READMEの間の記述項目の変化をソフトウェア種別に比較するために，プロジェクトの更新履歴から初期READMEを抽出する．GitHubに公開されるプロジェクトは，バージョン管理システムGitを用いてソースコードやREADMEの変更を管理している．分析2で示した2つのソフトウェア種別に特定されたプロジェクトを対象に，プロジェクトが持つGitの更新履歴から初期READMEを取得する．

初期README時点での記述項目を得るために，取得した初期READMEに対して，分析1と同じアプローチで見出しから記述項目を抽出・分類する．

***(Step 2)* 記述項目の変化パターンの抽出**

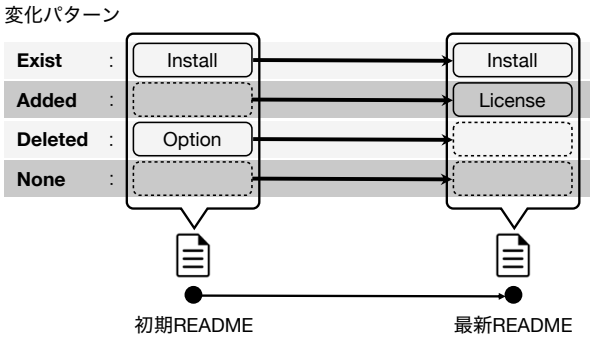


図 9 README 記述項目の変化パターン

図 9 は，本分析で識別する項目の変化パターン 4 つを示す．すなわち，(a) Exist: 初期 README と最新 README の両方に現れる項目，(b) Deleted: 初期 README には記述されたものの，最新 README までに削除された項目，(c) Added: 初期 README には記述されなかったが，最新 README までに追加された項目，(d) None: 初期 README にも最新 README にも現れない項目の 4 つである．対象とする項目は，必ず 4 パターンのいずれかに分類される．

結果を示すために，変化率 *Changing rate* を使用した．各変数について説明する．St は，*application* あるいは *library* のソフトウェア種別を表す．c は，対象とした上位 6 項目のうち，いずれか 1 つの項目を指す．$P_{init}(St)(c)$ や $P_{latest}(St)(c)$ は，ソフトウェア種別や項目に依存して，その値を変化させることを表している．

$P_{all}(St)$ は，ソフトウェア種別における全プロジェクトを表す．$P_{init}(St)(c)$ は，ソフトウェア種別の全初期 README のうち，いずれか 1 つの項目を含む全プロジェクトを表す．$P_{latest}(St)(c)$ は，ソフトウェア種別の全最新 README のうち，いずれか 1 つの項目を含む全プロジェクトを表す．$P_{init}(St)(c)$ および $P_{latest}(St)(c)$ の否定は，各 README のうち，いずれか 1 つの項目を含まない全プロジェクトを表す．4 つの変化パターンにおける *Changing rate* はそれぞれ次のように算出される．

$$Changing\ rate_{exist}(St)(c) = \frac{|P_{init}(St)(c) \cap P_{latest}(St)(c)|}{|P_{all}(St)|} \tag{7}$$

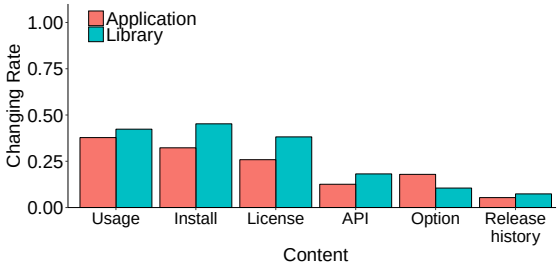


図 10　exist パターン

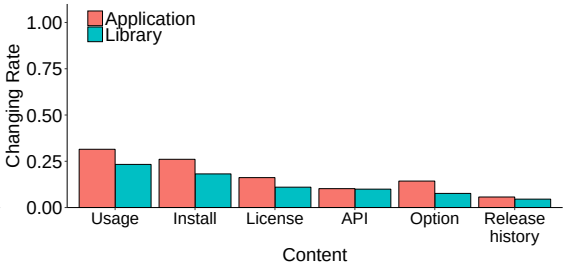


図 11　added パターン

$$Changing\ rate_{added}(St)(c) = \frac{|\overline{P_{init}(St)(c)} \cap P_{latest}(St)(c)|}{|P_{all}(St)|} \tag{8}$$

$$Changing\ rate_{deleted}(St)(c) = \frac{|P_{init}(St)(c) \cap \overline{P_{latest}(St)(c)}|}{|P_{all}(St)|} \tag{9}$$

$$Changing\ rate_{none}(St)(c) = \frac{|\overline{P_{init}(St)(c)} \cap \overline{P_{latest}(St)(c)}|}{|P_{all}(St)|} \tag{10}$$

5.1.2　結果

Step1 では，2,788 件の *Application* ソフトウェアと 1,870 件の *Library* ソフトウェアの更新履歴から，初期 README を取得し，初期 README の記述項目を抽出した．

Step2 では，初期 README と最新 README の項目を比較し，各項目の 4 つの *Changing rate* を算出した．ただし，記述しているプロジェクト数の少ない項目では 1 プロジェクトの変化パターンの有無の比重が大きくなるため，表 5 に示す上位 6 項目（“Usage”，“Install”，“License”，“API”，“Option”，“Release history”）のみを対象とした．

図 10 - 13 に，4 つの変化パターンにおける各項目の *Changing rate* を示す．

Library ソフトウェアに着目すると，図 10 に示すように，*Library* ソフトウェアは *Application* ソフトウェアに比べ，“Option” 以外の項目で $Changing\ rate_{exist}$ が高く，初期の時点から項目を記述している．一方で図 11 に示すように，$Changing\ rate_{added}$ は *Application* ソフトウェアよりも低く，項目を追加することは少ない．図 13 に

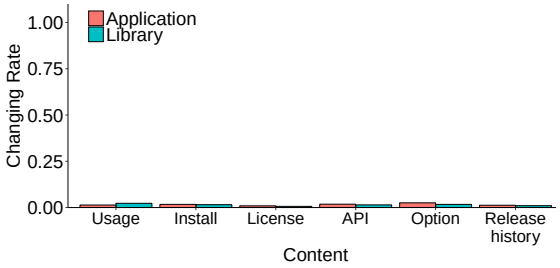


図 12 deleted パターン

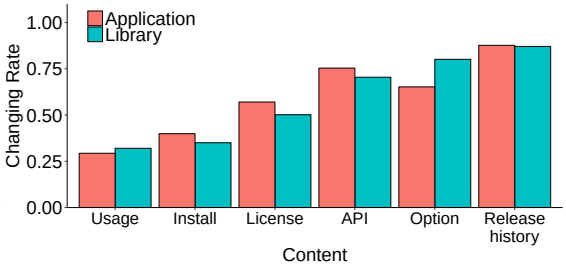


図 13 none パターン

示すように $Changing\ rate_{none}$ が高いことから，*Library* ソフトウェアは *Application* ソフトウェアに比べ “Option” を記述しないことを考えると，*Library* ソフトウェアは README を作成した時点で必要な項目を記述しており，あまり変更を加えないと言える．*Library* ソフトウェアは *Application* ソフトウェアに比べ，単一あるいは少数の機能を提供することが多い．機能は比較的早く実装されるため，*Library* ソフトウェアは *Application* ソフトウェアに比べ大きく機能が変化せず，README も大きく変化しないと考えられる．

Application ソフトウェアに着目すると，*Application* ソフトウェアは *Library* ソフトウェアに比べ，“Option” を除く 5 項目で $Changing\ rate_{exist}$ は低いが，$Changing\ rate_{added}$ は高い．特に “Usage” は，$Changing\ rate_{none}$ が *Library* ソフトウェアより低いことから，最新 README において記述しているプロジェクトの割合が *Library* ソフトウェアを超えており，プロジェクトの成長とともに必要となる項目であることが読み取れる．理由として，*Application* ソフトウェアは *Library* ソフトウェアに比べ，提供される機能が多いため，詳しく説明する必要があることが考えられる．あるいはプロジェクトの成長とともに機能が増え，必要とされるようになったことが考えられる．もし利用者からの指摘により “Usage” が後から追加されている場合は，“Usage” の記述を事前に開発者へ推薦することの必要性を示唆する可能性がある．

本論文の今後の発展として，プロジェクトのコミットコメントやプルリクエストを分析して，項目を追加された理由を探ることが挙げられる．項目の変更内容と，変更した理由を紐付けることは，具体的な記述方針を与えるために有用である．また本考察では初期 README と最新 README のみを対象としたが，更新

一回ごとの粒度で分析することも，具体的な記述方針を与えるために有用であると考えられる．

> 少機能な *Library* ソフトウェアは，*Application* ソフトウェアよりも早く必要な README の項目を記述する．多機能な *Application* ソフトウェアは，機能一覧を説明する項目については初期から記述するが，*Library* ソフトウェアよりも後から利用方法を説明する項目を記述する．

全体的な比率に着目すると，図 12 に示すように，プロジェクトは初期 README に記述した項目を削除する可能性は他の変化パターンに比べ，極端に低いことが読み取れる．$Changing\ rate_{deleted}$ が低いことは，初期に記述される項目は十分必要だと考えられた項目であること，開発者は README へ項目を削除するような大きな変更を行わないことを示唆する可能性があるが，さらなる分析は今後の課題とする．

5.2 結果の妥当性

本項では，本論文の分析における結果の妥当性について考察する．

外的妥当性 - Java エコシステム Maven，Ruby エコシステム Rubygems といった，他のソフトウェアエコシステムに対する本論文での結果の一般化への懸念を指す．分析 1 の結果にて，“Install” が 59.43%のプロジェクトで記述されていることが明らかになった．ただ npm は，“npm install [package name]” というプロジェクトのソフトウェアを容易にインストールするためのコマンドを用意しており，この事実が “Install” の記述の有無に影響を与えている可能性がある．このように README のパターンや傾向が他のシステムでは異なると考えられる．

内的妥当性 - 本論文の内部の懸案事項を指す．本論文の結果に影響を与える恐れのある内部脅威は 2 つ考えられる．1 つは項目の抽出に現れる．分析 1 では，内

容が類似していると考えられる項目を，手作業によって統合している．統合は著者と 2 人の共同研究者間で合意を形成することで行われた．しかし，統合は厳密な反復プロセスに従っており，大きな問題はないと考えられる．もう 1 つの脅威は，README の内容に関連する．すべてのプロジェクトが README に情報を記述しているわけではない．“License” をはじめとして，“Contribute” や “Release history” を外部ファイルに移しているケースを確認している．他のすべてのドキュメントの存在を調査し，開発者らがどの内容をどのドキュメントに記述しているかを明らかにすることは，README に記述すべき項目を明らかにする上で重要な調査だと考えられる．

構成概念妥当性 - README から項目を抽出する際に現れる．分析 1 では，Markdown 形式を利用して，見出しレベル 1 と 2（つまり，*h1* と *h2*）を対象として抽出している．レベル 3（*h3*）を用いてソフトウェアの重要な情報を記述するプロジェクトも考えられる．しかしレベル 3 も含めた抽出を行ったところ，大きく結果が変わらないことを確認している．むしろ誤った統合が増えると考え，レベル 3 の抽出は行わないことにした．

6. 関連研究

本論文で扱ったREADMEは，ソフトウェアドキュメントの1つである．ソフトウェアドキュメントについてはこれまでに多数の研究が報告されている．

ソフトウェアドキュメントの1つにAPIドキュメントがある．APIドキュメントは，ソフトウェアの提供するアプリケーションプログラミングインターフェイス（API）を使用したプログラミング作業を，ソースコードからは明らかではない情報を提供することで支援する．APIドキュメントにおいては，記述内容の分析から，保守の支援，利用の支援のための研究が行われている．Maalejらは JDK 6と.NET 4.0という主要なプラットフォームにおけるAPIドキュメントの記述内容を調査し，記述される項目とその記述傾向の違いを明らかにした [9]．Zhouらは Javaソフトウェアにおいて，APIドキュメントと実装の矛盾を自動検出する手法を提案した [16]．Jiangらは冗長なAPIドキュメントから目的のAPIについての記述を要約する手法を提案した [6]．

同じくソフトウェアドキュメントの1つにリリースノートがある．リリースノートは，ソフトウェアの更新履歴を記述したドキュメントで，利用者のソフトウェアアップデートを支援する．リリースノートにおいては，記述内容の分析から，文書作成支援のための研究が行われている．Abebeらは，リリースノートの記述内容を調査し，記述すべきと考えられる項目を明らかにした [1]．Morenoらは，リリースノートの作成が困難であることを開発者は知っていると報告した上で，リリースノートを自動的に生成する方法を提案した [11]．

READMEに関連した研究としては，READMEを活用した研究がいくつか報告されている．Hassanらは，READMEにはソフトウェアのビルドコマンドが記述されることを利用して，ビルドコマンドを自動的に抽出し，ビルドする手法を提案している [5]．Zhangらは，READMEにはソフトウェアの端的な説明が記述されることを利用して，類似プロジェクトを検出する手法を提案した [15]．本論文が明らかにしたREADMEに記述される項目の一覧は，READMEに記述される項目を活用した研究を支援する．

7. おわりに

本論文では，オープンソースソフトウェア（OSS）プロジェクトの未熟な開発者に対してREADME作成時の方針を与えることを目的として，成熟したプロジェクトのREADMEで共通して記述される項目と，ソフトウェア種別に記述項目の違いを明らかにした．具体的には，READMEに記述される代表的な26種類の分類を発見し，ガイドラインに準拠した項目が含まれていることを明らかにした．特に “Usage”（利用方法），“Install”（インストール方法），“License”（ライセンス）が最も頻繁に記述される項目である．26種類の分類には，ガイドラインに示されていない項目も存在したほか，ガイドラインに示されている項目を記述するために複数の方法が取られていることが明らかになった．またソフトウェア種別に応じて，重視する項目が異なった．アプリケーションソフトウェアでは，機能一覧を説明する “Option” が比較的頻繁に記述され，ライブラリソフトウェアでは “Install” と “License” が頻繁に記述されることを明らかにした．本論文が定義したREADMEの項目の抽出方法と結果として示した分類は，初心者のOSS開発者に対して，README作成時の方針を与えると共に，READMEを活用した研究を支援する．

今後の発展として，データセットを拡張しより一般的な結果を得ることが挙げられる．またバッジや記述内容を分析することで，より具体的な記述方法を示すことが可能になると考えられる．項目の変更履歴を分析し，変更した理由と紐付けることで，記述する時期も考慮した記述方針を与えることが可能になると考えられる．

謝辞

本論文の執筆を進めるに当たり，多くの方々に御指導，御協力を賜りました．ここに御名前を記させて頂くと共に，深く感謝の意を示します．

主指導教員であり，本論文の審査委員を務めて頂いた，奈良先端科学技術大学院大学 情報科学研究科 松本 健一教授には，研究の進め方を始めとして，研究者としての在り方等，あらゆる面で非常に多く重要な御指導・御助言を賜りました．また学外活動や奨学金申請の書類作成においても御協力頂き，二年間の学生生活を非常に有意義にすることができました．心より感謝申し上げます．

副指導教員であり，本論文の審査委員を務めて頂いた，奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア設計学研究室 飯田 元教授には，本論文が捉えるべき問題や意義について貴重なご意見を賜り，本論文の質を向上させることができました．心より感謝申し上げます．

副指導教員であり，本論文の審査委員を務めて頂いた，奈良先端科学技術大学院大学 情報科学研究科 石尾 隆准教授には，本論文の執筆や学内外での発表資料の作成にあたり御指導・御助言賜りました．また分析に重要な手法について，的確な御助言を賜りました．心より感謝申し上げます．

副指導教員であり，本論文の審査委員を務めて頂いた，奈良先端科学技術大学院大学 情報科学研究科 伊原 彰紀助教には，研究の方針や進め方について御指導・御助言賜ったことを始めとして，研究への関わり方や姿勢など生活において重要なことを学ばせて頂きました．先生の支えがあったからこそ，ここまで研究を進めることができました．心より感謝申し上げます．

副指導教員であり，本論文の審査委員を務めて頂いた，奈良先端科学技術大学院大学 情報科学研究科 Raula Gaikovina Kula 特任助教には，研究の方針について重要な御指導・御助言を賜りました．中でも英語論文の執筆にあたり非常に多くの御協力を賜りました．心より感謝申し上げます．

本論文の審査委員を務めて頂いた，奈良工業高等専門学校 情報工学科 上野 秀剛准教授には，奈良工業高等専門学校在学時から長期に渡り，研究の進め方をはじめとして，将来のキャリア形成等の広い範囲に渡り，御指導・御助言賜りました．また奨学金申請の書類作成においても御協力頂き，学びに集中した学生生活

を送ることができました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 畑 秀明助教には，研究の方針について御指導・御助言を賜りました．斬新なアイデアや研究に対する姿勢は，私に非常に強い刺激を与えてくださいました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 ユビキタスコンピューティングシステム研究室 諏訪 博彦助教には，研究計画を立てるにあたって重要な知識や考え方を御指導・御助言賜りました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学研究室の高岸 詔子氏には，学生生活や学外での発表における書類作成や諸連絡等，事務処理の分野で多くの御協力を頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学研究室の OB，先輩，同級生，後輩の皆様には，非常に仲良くしていただき，2 年間の大学院はとても充実したものとなりました．修了後も変わらず，お互いを刺激しあい，切磋琢磨し合う関係であり続けられますよう願っております．

母方の親族の皆様には，奨学金を貸与するにあたり，保証人になっていただきました．私を信用してくださり，本当にありがとうございます．

最後に，私の意思を尊重し，常に支えてくださった家族に深く感謝致します．

参考文献

[1] Surafel Lemma Abebe, Nasir Ali, and Ahmed E. Hassan. An empirical study of software release notes. *Empirical Software Engineering*, 21(3):1107–1142, 2016.

[2] Jailton Coelho and Marco Tulio Valente. Why modern open source projects fail. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 186–196, 2017.

[3] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[4] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the International Conference on Management of Data*, pages 1–12, 2000.

[5] Foyzul Hassan and Xiaoyin Wang. Mining readme files to support automatic building of java projects in software repositories: Poster. In *Proceedings of the 39th International Conference on Software Engineering Companion*, pages 277–279, 2017.

[6] He Jiang, Jingxuan Zhang, Zhilei Ren, and Tao Zhang. An unsupervised approach for discovering relevant tutorial fragments for apis. In *Proceedings of the 39th International Conference on Software Engineering*, pages 38–48, 2017.

[7] Andrew M. St. Laurent. *Understanding Open Source and Free Software Licensing*. O'Reilly Media, August 2004.

[8] Timothy C. Lethbridge, Janice Singer, and Andrew Forward. How software engineers use documentation: The state of the practice. *IEEE Software*, 20(6):35–39, 2003.

[9] Walid Maalej and Martin P. Robillard. Patterns of knowledge in api reference documentation. *IEEE Trans. Softw. Eng.*, 39(9):1264–1282, September 2013.

[10] Samim Mirhosseini and Chris Parnin. Can automated pull requests encourage software developers to upgrade out-of-date dependencies? In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ASE 2017, pages 84–94, 2017.

[11] Laura Moreno, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Andrian Marcus, and Canfora. Arena: An approach for the automated generation of release notes. *IEEE Transactions on Software Engineering*, 43(2):106–127, 2017.

[12] Fouad Nasser A Al Omran and Christoph Treude. Choosing an nlp library for analyzing software documentation: A systematic literature review and a series of experiments. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 187–197, 2017.

[13] Erik Wittern, Philippe Suter, and Shriram Rajagopalan. A look at the dynamics of the javascript package ecosystem. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 351–361, 2016.

[14] Chengqi Zhang and Shichao Zhang. *Association rule mining: models and algorithms*. Springer-Verlag, 2002.

[15] Yun Zhang, David Lo, Pavneet Singh Kochhar, Xin Xia, Quanlai Li, and Jianling Sun. Detecting similar repositories on github. In *Proceedings of the IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*, pages 13–23, 2017.

[16] Yu Zhou, Ruihang Gu, Taolue Chen, Zhiqiu Huang, Sebastiano Panichella, and Harald Gall. Analyzing apis documentation and code to detect directive defects. In *Proceedings of the 39th International Conference on Software Engineering*, pages 27–37, 2017.

[17] Thomas Zimmermann, Andreas Zeller, Peter Weissgerber, and Stephan Diehl. Mining version histories to guide software changes. *IEEE Transactions on Software Engineering*, 31(6):429–445, 2005.

付録

A. 項目の手動統合前後の対応表

分析 1 において行った項目の手動統合における統合前後の項目の対応を表 8 に示す.

表 8: 手動統合前後の項目における対応表

手動統合後の項目	手動統合前の項目
Usage	Usage, Example, Use, Example usage, Basic usage, Usage example, Screenshot, CLI usage, Quick example, Advance usage, Use case, Template, Sample, Syntax, Sample usage, Tip, Simple example, Code example, Grunt task, Basic example, Basic, Command line usage, API usage, Example use, Advance, Simple usage, Programmat usage, Tutorial
Install	Install, Get start, Setup, Quick start, Quickstart, Download, Install usage, Set, Get, NPM, Initialize, Get to know slush, NodeJS, Instruct, Start, Node, Bower, Install NPM, Install dependencies
License	License, Licence, Copyright, MIT license, Disclaim, License MIT, Copyright license, MIT licence, Unlicense, Legal, Lisence, MIT license MIT, License copyright
API	API, Method, Build, CLI, Run, Event, Command, Function, API reference, API document, Constructor, Command line, Interface, Action, Object, Execute

次ページに続く

前ページからの続き

手動統合後の項目	手動統合前の項目
Option	Option, Configure, Depend, Prerequisite, Config, Generation, Compatibility, Plugin, Module, Property, Default config, Browser, Server, Custom, Import, Component, Configure option, Connect, Route, Client, Format, Include, Util, Style, Promise, Structure, Variable, Environment variable, Middleware, Browser compatibility, Read, Class, Prop, Secure, Directory structure
Release history	Release history, Changelog, Version, History, Change log, Release note, Debug, Update, Log, Release, Change, Version history, Deploy
Contribute	Contribute, Develop, Contact, Donate, Hack, Feedback
Test	Test, Run test, Unit test
Todo	Todo, Require, Task, Future
Overview	Note, Overview, Why, Description, Introduction, Synopsis, This, Work, Summary, Intro, Name
Status	Status, Build status, Current status
Document	Document, See also, Doc, Inform, Doe, Doe work, Detail, Info, TL;DR, Notice
Author	Author, Credit, Contributor, Resource, Backer, Maintain
Support	Support, FAQ, Help, Troubleshoot, Commercial support, Question
Feature	Feature, Attribute
Relate	Relate, Relate project, Link, Inspire, Alternate, Source
Issue	Caveat, Known issue, Issue, Warn, Deprecate, Problem, Bug
Demo	Demo, Output, Result, Example output

次ページに続く

前ページからの続き

手動統合後の項目	手動統合前の項目
Purpose	Motivation, Purpose, Goal, Background, Philosophy, Concept, Idea, Rational, Solution
Refer	Refer, Acknowledge, Thank, Special thank
Limit	Limit, Browser support, Implement, Valid, Platform support
Error	Error, Filter, Error handle
Roadmap	Roadmap, Work progress
Table content	Table content, Content
Model	Model
Benchmark	Benchmark, Perform

以上

関連発表論文

国内会議

[1] 池田祥平, 伊原彰紀, ラウラガイコビナクラ, 松本健一, 「GitHub における README 記述項目の分析,」 第 24 回ソフトウエア工学の基礎ワークショップ（FOSE2017）, pages 135–140 2017 年 11 月.