

ライセンスルール自動生成に向けた 未知ライセンスにおける 頻出記述パターンの抽出

オープンソースソフトウェア工学研究室
60160087 東 裕之輔

1

OSSの再利用

- 企業では、開発コスト削減のため、オープンソースソフトウェア(OSS)を再利用した派生開発が行なわれている

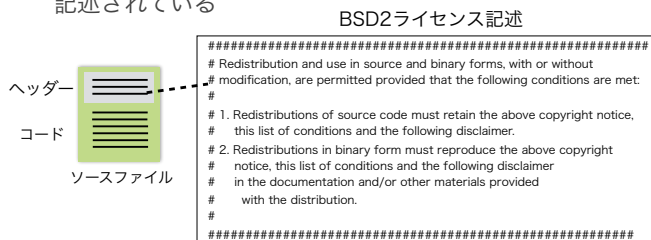


- OSSは、ソースファイルごとに決定されているOSSライセンスを遵守することで再利用することができる

2

ソースファイルのライセンス

- OSSライセンスは、ソースファイルのヘッダー部分に記述されている



- OSSのライセンスを遵守するには、全てのソースファイルのライセンスを確認しなければならない

3

3

OSSに含まれるソースファイル数

- OSSには、多くのソースファイルが含まれている

ドメイン: OS	ドメイン: データベース	ドメイン: サーバー
ソースファイル数: 22,310	ソースファイル数: 1,452	ソースファイル数: 291

- 再利用するOSSに含まれる、全てのソースファイルのライセンスを手作業で確認するのは、労力がかかる
 - Linux, MySQL, Apacheを再利用すると24,000ファイル以上のライセンス記述を確認しなければならない

4

4

ライセンス特定ツール[1][2]

- ライセンスルール（正規表現）を用いて、自動的にソースファイルのライセンスを特定する



[1] Ninka : Daniel M German, Yuki Manabe, and Katsuro Inoue, "A Sentence-Matching Method for Automatic License Identification of Source Code Files" In Proc. of ASE'10, pp. 437-446
[2] Fossology : Robert Gobeille, "The fossology project," In Proc. of MSR'08, 2008, pp. 47-50.

5

5

ライセンス特定ツールの課題

- 新しいライセンスや、未知の表記ゆれは、年々増加する
- ライセンス特定ツールは、**コンスタント**に新しいルールを追加し続けていくことが重要

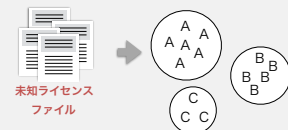


6

6

手作業でのライセンスルール作成手順

STEP1.未知ライセンスファイル集合を ライセンス毎に分類



STEP3.ライセンスルール作成



STEP2.各グループにおいて 表記ゆれを調査

ライセンスグループAの表記ゆれ

This program is~ it under the term of the A license
This software is~ it under the term of the A license
This library is ~ it under the term of the A license

問題点

大量の未知ライセンスファイルが検出
された場合、大変な労力がかかる

7

7

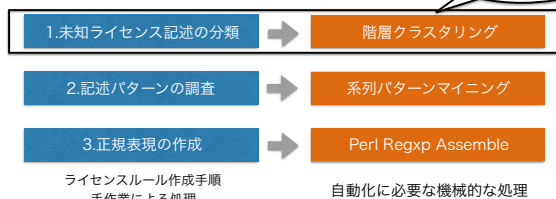
本研究のゴール

＊ ゴール：ライセンスルール自動生成手法の構築

＊ アプローチ

ライセンスルール作成手順における各ステップを、機械的
処理に置き換える

まず、STEP1の自
動化を行う



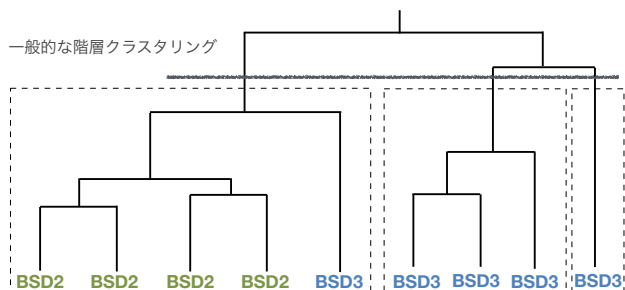
8

8

階層クラスタリング

- データの類似関係から、デンドログラムを生成
- 任意の高さでデンドログラムの枝刈りを行う

一般的な階層クラスタリング

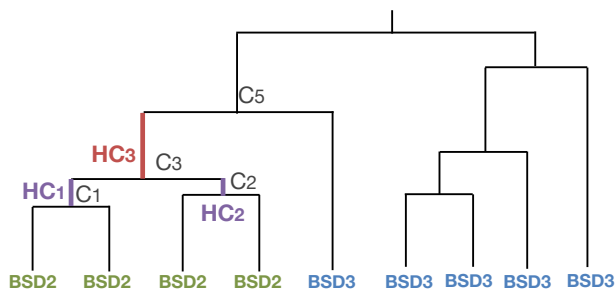


9

9

提案手法

- 次式が成り立つとき、枝刈りを行う



10

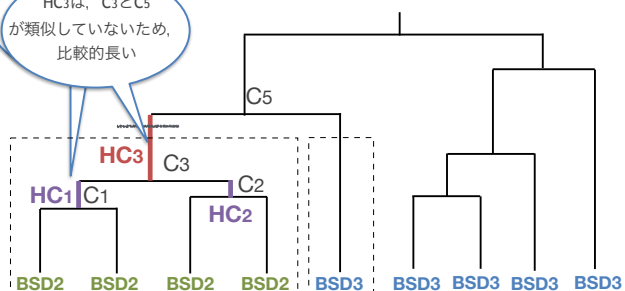
10

提案手法

- 次式が成り立つとき、枝刈りを行う

＊ C5を作成しない：((HC3 > HC1) & (HC3 > HC2)) == 1

HC3は、C3とC5
が類似していないため、
比較的長い



11

11

ケーススタディ

- 実験手順

- 未知のライセンスが検出されたファイルから、ライセンス記述を抽出
- ライセンス記述をBag of Wordsベクトルに変換
- 提案手法と階層クラスタリングを適用し、結果を比較

- データセット



未知ライセンス：2453ファイル 未知ライセンス：2801ファイル 未知ライセンス：1230ファイル

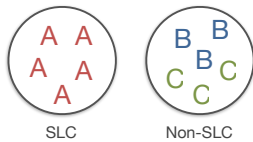
データセットのライセンスは、目視で特定

12

12

評価メトリクス

- SLC (Single License Cluster)
 - 2ファイル以上属していて、かつ1種類のライセンスから成るクラス
 - 本研究では、SLCを多く作成できることが望ましいとする
 - SLCからは、適切にライセンスルールを生成することができる



- SLC作成率 (SLC/クラスタ数)
 - 作成したクラスタのうち、どれくらいがSLCだったかを示す

13

13

評価方法

- 評価方法
 - 提案手法と比較手法を適用し、SLCの作成率を比べる
- 比較手法：一般的な階層クラスタリング
 - クラスタ間の距離尺度：ワード法
 - 事前に、デンドログラムをカットする位置（クラスタ数）を変更しながら実験を行い、SLCの作成率が最大になる結果と比較する

14

14

実験結果

Debian-7.8.0	クラスタ数	SLC (作成率)	Non-SLC (作成率)	単一ファイル
提案手法	295	178(60%)	69(23%)	48(16%)
一般的な階層クラスタリング	20	9(45%)	10(50%)	1(5%)
手作業	121	24(86%)	0	4(14%)

Linux-4.4.6	クラスタ数	SLC (作成率)	Non-SLC (作成率)	単一ファイル
提案手法	393	322(81%)	33(8%)	38(10%)
一般的な階層クラスタリング	280	217(78%)	40(14%)	23(8%)
手作業	28	24(86%)	0	4(14%)

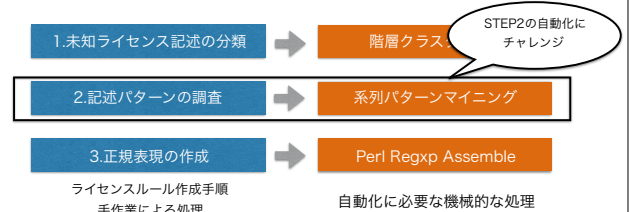
freebsd-10.3.0	クラスタ数	SLC (作成率)	Non-SLC (作成率)	単一ファイル
提案手法	148	121(82%)	11(7%)	16(11%)
一般的な階層クラスタリング	89	67(75%)	20(22%)	2(2%)
手作業	62	47(76%)	0	15(24%)

15

15

本研究のゴール

- ゴール：ライセンスルール自動生成手法の構築
- アプローチ
 - ライセンスルール作成手順における各ステップを、機械的処理に置き換える

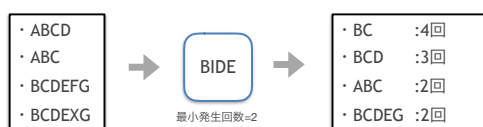


16

16

系列パターンマイニング

- 大量のアイテム列から頻出する部分アイテム列パターンを抽出するマイニング手法
 - 適用例：大量の購入パターン履歴から、よく同時に購入される商品を発見する
- BIDEアルゴリズム
 - 発生回数が同じで、かつ他の抽出したパターンの部分集合となるパターンを抽出しない
 - 計算コストが低い



17

17

STEP2自動化における技術的課題

- 1. Non-SLCからルールを作成することができない
 - Diagram showing SLC (A, A, A, A) generating a rule 'Aにマッチするルール' (Rule matching A). Non-SLC (B, B, C, C) cannot generate a rule 'B,Cにマッチするルール' (Rule matching B,C) because it is not a single license cluster.
- 2. 1種類のライセンスにのみに存在する記述を抽出しなければならない

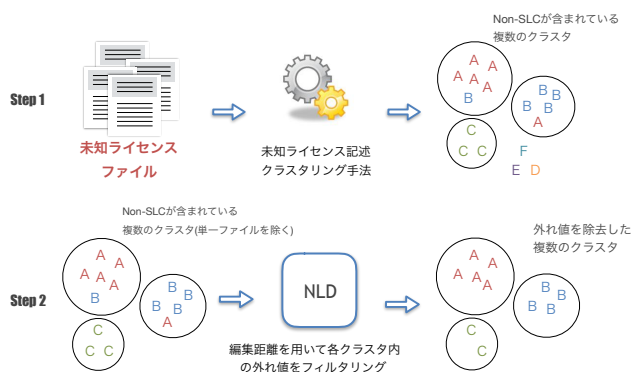
系列パターンマイニングで生成されたパターン	発生回数	ライセンス
of the	8	MPLv2.0
This Source Code Form is subject to the terms of the Mozilla Public License, v<dot>2<dot>0.	4	MPLv2.0
2 - This Source Code Form is subject to the terms of the Mozilla Public License, v<dot>2<dot>0.	3	MPLv2.0
If a copy of the MPL was not distributed with this file, You can obtain one at http://colorado.org/mozilla.org/MPL2.0/ .	4	MPLv2.0

どこにでもあるパターン (ノイズ)
ノイズが多いと、ノイズじゃない
パターンを確認するのに労力がかかる

18

18

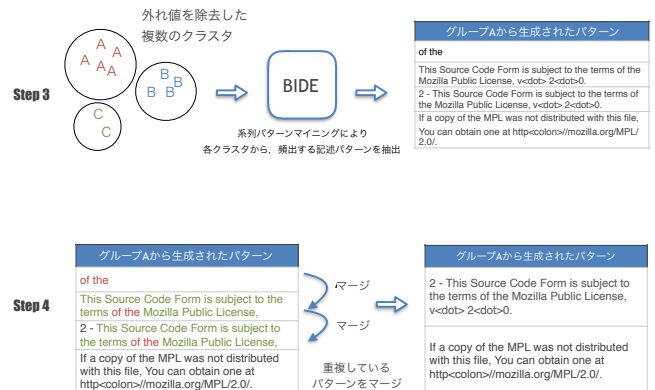
提案手法の概要(1/2)



19

19

提案手法の概要(2/2)



20

20

編集距離とは

- 編集距離とは、ある文字列Aから、文字列Bにするまで、何回の操作(削除、置換、挿入)が必要となるか
 - 例 kittenとsitting 距離=4
 - 1 「kitten」
 - 2 「sitten」 (「k」を「s」に置換)
 - 3 「sittin」 (「e」を「i」に置換)
 - 4 「sitting」 (「g」を挿入して終了)
- 本手法では、距離を最大文字数で正規化した編集距離類似度を用いる

$$NLD = 1 - \frac{LD(A, B)}{\maxlength(A, B)}$$

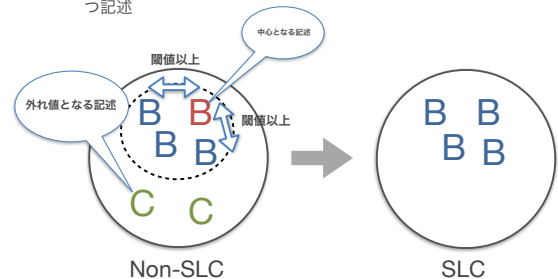
NLD : 正規化された編集距離
LD : 編集距離
maxlength : 最大文字数

21

21

外れ値の検出方法

- クラスタ内の外れ値をフィルタリング
 - クラスタの中心となる記述グループを抽出
 - クラスタの中心となる記述 : 類似度が一定以上ある記述を最も多く持つ記述

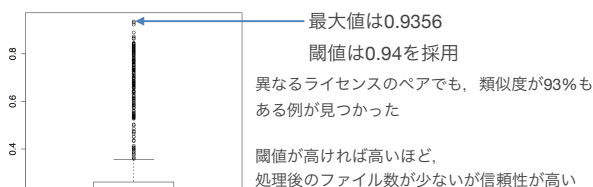


22

22

編集距離の閾値決定

- 予備調査
 - Debian v7.8.0から、ランダムに取り出した、異なるライセンス記述のペアにおける類似度を1000組調査



23

23

実験

- 実験目的 : 技術的課題をクリアできるかどうかを確認める
 - どれくらいのクラスタをSLCにすることができるか
 - 記述パターンからどれくらいのライセンスルールを生成できるか
- データセット
 - Debian v7.8.0の未知ライセンスファイルをクラスタリングした結果を利用

Debian-7.8.0	ファイル数	クラスタ数	SLC	Non-SLC	単一ファイル
提案手法	2,453	295	178(60%)	69(23%)	48(16%)

24

24

実験方法

- どれくらいのクラスタをSLCにすることができるか
 - 1. 予備実験で決定した閾値で外れ値を除去する
 - 2. SLC, Non-SLCの数をカウントする
- ライセンスルールを生成できる記述パターンをどれだけ作成できるのか
 - 3. BIDEによって出力された記述パターン(マージ前)が, 1種類のライセンスのみに存在するものか, データセットを使って調査
 - 4. 記述パターンをマージ後, 同じ調査を行い, 1種類のライセンスのみに存在する記述パターンの割合を比較する

25

25

実験結果1

- どれくらいのクラスタをSLCにすることができるか
 - 閾値を変化させることにより, SLC, Non-SLCがどう変化するかを調べた

閾値	クラスタ数	#SLC	#Non-SLC	ファイル数	閾値	クラスタ数	#SLC	#Non-SLC	ファイル数
0	247	178	69	2405	0.91	247	141	4	517
0.50	233	189	44	1102	0.92	233	134	4	494
0.55	228	192	36	1013	0.93	228	130	4	475
0.60	224	196	28	949	0.94	224	127	2	444
0.65	219	194	25	886	0.95	219	119	1	413
0.70	210	190	20	827	0.96	210	114	1	390
0.75	194	177	17	762	0.97	194	104	1	363
0.80	183	170	13	706	0.98	183	93	1	322
0.85	170	162	8	644	0.99	170	74	0	264
0.90	145	141	4	542					

結果: 閾値0.94で, 69あったNon-SLCを2つに減らすことが出来た.

また, 閾値を0.99に設定すると, Non-SLCを0にすることも可能

26

26

実験結果2

- 系列パターンマイニングを各クラスタに適用した結果

	生成されたパターン数	1種類のライセンスのみに存在したパターン数	割合
マージ前	1498	375	25%
マージ後	371	247	67%

- ライセンスルールとして適切でないパターンを減らすことができた
- しかし, 課題が残る結果に, . . .
 - 割合が低い
 - 1種類のライセンスのみに存在したパターン数の数が128減ってしまっている

27

27

まとめと今後の課題

- まとめ
 - 複数の未知ライセンスクラスタから, 記述パターンを抽出する手法を提案した
 - ケーススタディで分かったこと
 - 外れ値除去により, クラスティング結果から, ほとんどのNon-SLCをなくすことができた
 - ノイズとなる記述パターンの出力を減らすには, もっと工夫が必要
- 今後の課題
 - 文字数, 単語数でフィルタリングしてからマージする
 - 1種類のライセンスのみに存在したパターンの調査を, 既知ライセンスを含めて行う

28

28