ライセンス特定のためのルール自動生成を目的としたライセンス記述パターン抽出手法

Extracting license statement patterns to automate the rule generation for OSS licenses

東　裕之輔[*]　**眞鍋　雄貴**[†]

あらまし　OSS の再利用を支援するため，再利用対象となる OSS に含まれるソースファイルのライセンスを自動的に特定するツール（ライセンス特定ツール）が提案されている．ライセンス特定ツールの一つである Ninka [1] は，様々なルールを用いて，ソースファイル中のライセンスを指定する記述（ライセンス記述）から精度の高いライセンス特定を行う．Ninka が新しいライセンスや既存ライセンスに対応するには，定期的に Ninka のルールが対応しないライセンスに対するルールを作成・追加することが望ましい．しかし，ライセンスが特定されていない数多くのファイルからルールを人手で作成するのは容易ではない．筆者らはこれに対し，ライセンス記述の集合からルール抽出に適したライセンス記述のクラスタを得る手法を提案しているが，得られたクラスタから自動的にルールを抽出するにはクラスタの性質に関して課題が残っている．本研究は，ライセンス記述のクラスタから複数のライセンス記述に出現し，ライセンスルール作成に有用なパターン（ライセンス記述パターン）を系列パターンマイニングを用いて自動的に抽出する手法を提案する．本手法の有用性を検証するために，Debian v7.8.0 から抽出した 2,838 個の Ninka でライセンスが特定できなかったファイルを対象としてケーススタディを行った．その結果，本手法により，ライセンスルール作成に有用な 240 のライセンス記述パターンを抽出することができた．

1　はじめに

開発コストの削減を主たる理由として，オープンソースソフトウェア（OSS）を再利用したソフトウェア開発が多くの組織において行われるようになった．OSS はソースファイルごとに定められたライセンスを遵守することで再利用することができる [1] ため，OSS の再利用に際しては，再利用対象となるソースファイルに適用されているすべてのライセンスを特定し，ライセンスの内容を正しく理解しておく必要がある．ライセンスは，OSS に同梱されるドキュメントで指定される場合もあるが，本研究では，ソースファイルのコメントに記述されるライセンス（ライセンス記述）を対象とする．

再利用対象となるソースファイルが多数存在する場合，すべてのライセンス記述を目視で確認するには相当の労力を要するため，ライセンス特定作業の効率化を目的としたライセンス特定ツールが提案されている [1] [2] [3] [4] [5] [6]．ライセンス特定ツールは，ライセンス記述とライセンスの対応関係を示すルール（ライセンスルール）をあらかじめ用意しておき，再利用対象のソースファイルのライセンス記述とライセンスルールを比較することでライセンスを特定するものである．ライセンス特定ツールの一つに Ninka [1] がある．Ninka は，OSS の分析に基づいて作成した正規表現をライセンスルールとして用いる．ライセンスルールが存在しないライセンス記述を入力とした時，Ninka 以外のツールは何らかのライセンス特定結果（ライセンス名）を出力するのに対して，Ninka は未知のライセンスであることをライセンス特定結果として出力し，誤検出を回避する．そのため，他のツールに比べ Ninka はライセンスの特定精度が極めて高い（適合率が 96.6%）という利点がある．その一方で，未知ライセンスが多数存在する場合，目視すべきソースファイルも多数存

[*]Yunosuke Higashi，和歌山大学システム工学研究科

[†]Yuki Manabe，熊本大学大学院先端科学研究部

在するため，ライセンス特定の効率化という点においてNinkaを用いることは必ずしも適切ではない．Ninkaの高いライセンス特定精度を維持しつつ目視作業を減らすためには，Ninkaがライセンス特定できなかったライセンス記述からライセンスルールを抽出し，Ninkaに適宜追加できることが望ましい．

ライセンスルールの自動抽出へ向けて，これまで筆者らはNinkaが未知ライセンスであると判定したライセンス記述をクラスタリングする手法を提案している [7]．類似する（未知の）ライセンス記述を分類することで，重複するライセンスルールの作成を防ぐことを狙ったものである．本稿では，先行研究のクラスタリング手法により生成されたライセンス記述のクラスタから，ライセンス記述パターンを抽出する手法を提案する．ライセンス記述パターンとは，特定のライセンス記述に頻出する語句の系列を指す．ライセンス記述パターンを適切に抽出することができれば，正規表現としてライセンスルールを作成するための入力とすることが可能になるため，ライセンスルールの自動抽出へ向けた重要なステップになると考えている．

2　準備

2.1　用語

本節では，本論文で用いる用語を定義する．

ライセンス記述　ソースファイル中のコメントにあるライセンスを指定する記述．

ライセンスルール　ライセンスを特定する際，ライセンス記述とライセンス名を対応づけるためのルール．Ninkaはライセンスルールとして正規表現を用いている．

未知ライセンス　Ninkaが対応するライセンスルールを持っていないライセンス．

未知ライセンス記述　未知ライセンスであるライセンスのライセンス記述．

系列パターン　複数の語句の系列に現れる語句の系列．

ライセンス記述パターン　特定のライセンス記述に出現する系列パターン．

2.2　ライセンスルール作成手順

本節では，ライセンスルール作成を自動化する前に，ライセンスルールを手作業で作成した場合，どのような作業が必要なのかを明らかにし，ライセンスルール作成を自動化する意義を説明する．ライセンスルールを自動生成する前提として，手作業によるライセンスルール作成は，以下の3つのステップから成ると想定する．

Step1．検出された未知ライセンスファイルをそのライセンス名によって分類する．

検出された全ての未知ライセンスファイルを手作業で確認し，それぞれのライセンス名を特定する．特定されたライセンス名に基づき分類し，クラスタを得る．

Step2．各クラスタにおける頻出記述と表記ゆれを調査する．

Step1により得られたクラスタからライセンスルール作成に利用できる2つ以上のライセンス記述で出現する記述（頻出記述）を抽出する．加えて，頻出記述からライセンス記述による表記揺れも調査できる．例としてクラスタ内に，“This **software** is free” と “This **program** is free” という頻出記述が得られたとすると，“software” と “program” の表記ゆれを確認できる．

Step3．頻出記述と表記ゆれから正規表現を作成する．

Step2で得られた頻出記述と表記揺れからライセンスルールとなる正規表現を作成する．例えば，“This **software** is free” と “This **program** is free” の両方の頻出記述にマッチする正規表現を作成するには，“This **(software)｜(program)** is free” とすればよい．

以上のステップでライセンスルールを作成した場合，相当な労力がかかることが予想される．まず，Step1では，未知ライセンスが検出されたファイル数に応じて，ライセンスを特定にかかる労力が増加してしまう．German [1] らが行ったNinkaの評価実験では，実験対象となった250のソースファイル中，43ファイル (17.2%) に対し，未知のライセンスが検出されたことを示している．また，Step2では，表記

揺れを多く含んでいるライセンスには，多くの頻出記述が存在する．それらを全て手作業で抽出するのは難しい．そのためには，文字単位の比較を行わなければならず，人為的ミスも発生しやすい．このように，手作業でライセンスルールを作成すると相当な労力を要するため，ライセンスルール作成手順を自動化することは望ましいと考えられる．

3　ライセンス記述階層クラスタリング手法

本研究で提案する手法の入力となるクラスタリング結果を生成する先行研究の手法について説明する．本手法は，単一のライセンスのライセンス記述で構成されているクラスタをできるだけ多く得られるよう，樹形図からクラスタを得る部分が工夫されている．未知ライセンス記述にクラスタリング手法を適用するための前処理と階層クラスタリングの 2 つのステップが存在する．以下に，それぞれについて説明する．

3.1　前処理

まず，ライセンス特定ツール Ninka を用いて，ソースファイルからライセンス記述を抽出する．Ninka は，ソースファイルのコメント部分から，ライセンス記述に頻出する単語が存在する文を抽出することでライセンス記述を抽出することができる．

次に，grep を用いて GPL/BSD 系ライセンスに対応するライセンス記述をソースファイルから抽出されたライセンス記述から除外する．GPL/BSD 系ライセンスは，バージョン間のライセンス記述が酷似しているため，機械学習アルゴリズムで判別することが難しい．そのため，キーフレーズを用いて，バージョンやライセンス名，条項の有無を判別する．

3.2　ライセンス記述の階層クラスタリング

はじめに，前処理により抽出されたライセンス記述から，樹形図を得る．まず，各ライセンス記述の Bag of Words ベクトルを作成する．次に，ウォード法 [8] に基づき，ライセンス記述の Bag of Words 集合に対して階層クラスタリングを行い，その過程から樹形図を得る．このとき，クラスタ間の類似度としてユークリッド距離を用いる．

次に，樹形図から類似したライセンス記述からなるクラスタを得る．一般的な階層クラスタリングでは，閾値となるクラスタ間の距離（高さ）を樹形図から決定し，複数のクラスタに分割する．ただし，未知ライセンス記述の中には，ライセンス名は異なるが酷似しているライセンス記述も含まれる．このようなライセンス記述を別のクラスタにするため，ライセンス記述の階層クラスタリングでは，式 1 を満たす樹形図内のクラスタ C_i をクラスタとして切り出す．ここで，C_j は C_i と次にマージされる対象のクラスタ，C_p は C_i と C_j をマージして得られたクラスタ，SC_i と SC_j を C_i を分割して得られる最も小さいクラスタ，$H(C_p)$ をクラスタ C_i と C_p 間の距離とする．

$$H(C_p) > H(SC_i) \ \ and \ \ H(C_p) > H(SC_j) \tag{1}$$

4　ライセンス記述パターン抽出における技術的課題

Ninka により未知ライセンスと判定された未知ライセンス記述からライセンスルールを自動抽出するために，著者らは以下の工程が必要と考えている．

1. **未知ライセンス記述のクラスタリング**　Ninka が出力した未知ライセンス記述の集合をライセンス毎に分類するためのクラスタを作成する．
2. **ライセンス記述パターンの抽出**　作成した各クラスタの未知ライセンス記述に頻出する語句の系列（ライセンス記述パターン）を抽出する．
3. **正規表現の生成**　抽出されたライセンス記述パターンから正規表現を生成する．

上記の 1 番目の工程については，前章で述べたクラスタリング手法 [7] による自動化を提案している．本研究は 2 番目の工程の自動化を目的とするものである．ライセンス記述パターンの自動抽出を行うためには，以下の二つの技術的課題を解決する必要がある．

課題 1．クラスタリング手法の適用結果をそのまま入力として利用できない

筆者らが提案したクラスタリング手法 [7] によって作成された各クラスタは，同一の未知ライセンス記述のみで構成されるとは限らない．すなわち，複数の未知ライセンスを含んだクラスタが作成される場合がある．異なるライセンスであってもライセンス記述が酷似していれば，テキストの類似性に基づくクラスタリング手法では完全にその差異を区別することができないため，結果的に異なる未知ライセンス記述が同じクラスタに属することを排除できない．複数のライセンス記述を含んだクラスタからライセンス記述パターンを抽出することは技術的に可能であるが，ライセンス特定ツールの本来の目的である「ライセンス記述毎のライセンス特定」のためのライセンスルールを作成できないため無意味である．したがって，ライセンス記述パターンの自動抽出では，[7] の手法を用いて作成される各クラスタを，それぞれ単一のライセンス記述で構成されるクラスタに再構成する必要がある．

課題 2. ライセンスルール作成に不適切な系列パターンが抽出される可能性がある

ライセンス記述には一般の文書でも頻出するような記述が含まれているため，ライセンス記述パターンとしてライセンス特定に不適切なパターンが含まれる場合がある．例えば，“This **software** is free software” と “This **program** is free software” からは，“This [...] is free software” の他にも “This [...] is [...]” というライセンス記述パターンが得られる．“This [...] is [...]” というライセンス記述パターンは，複数のライセンスに対応できることを意味するため,「ライセンス記述毎のライセンス特定」のためのライセンスルールを作成する目的としては不適切である．したがって，ライセンス記述パターンの自動抽出では，課題 1 に加え，ライセンスルールの作成という目的に沿わない不適切なライセンス記述パターンをできるだけ抽出しないようにする必要がある．

5　ライセンス記述パターン抽出手法

本章では，先行研究 [7] のクラスタリング手法を適用し作成した未知ライセンス記述からなるクラスタから，ライセンス記述パターンを自動抽出する手法を提案する．

5.1　提案手法の概要

提案手法の概要を図 5 に示す．図 5 上段は，先行研究 [7] の概要を，図 5 下段は，本稿で提案する手法の概要を示している．先行研究 [7] は，未知ライセンス記述の集

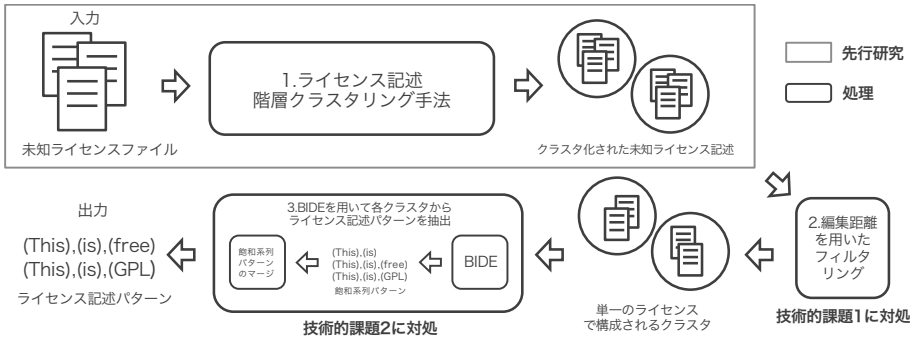

図 1　提案手法の概要

表 1　文章における語句の系列

	文章
1	(This),(program),(is),(free)
2	(This),(library),(is),(free)
3	(This),(library),(is),(OSS)

表 2　BIDE により抽出された飽和系列パターン

	パターン	支持度
1	(This),(is)	3
2	(This),(is),(free)	2
3	(This),(library),(is)	2

合に対して階層クラスタリング手法を適用し，類似する未知ライセンス記述からなるクラスタを作成するものである．本稿で提案する手法は，次の 2 つの処理を行いライセンス記述パターンを自動抽出するものである．まず，第 4 章で述べた技術的課題 1 に対処するために，テキスト間の類似度を用いて各クラスタにおける外れ値となるライセンス記述をフィルタリングし，各クラスタに異なる未知ライセンス記述が混在しないようにする．次に，技術的課題 2 に対処するために，各クラスタに対し系列パターンマイニングアルゴリズムの 1 つである BIDE を適用し，飽和系列パターンを抽出する．また，抽出された飽和系列パターンのマージを行いライセンスルール作成に有用なライセンス記述パターンを生成する．以降では，それぞれの処理の詳細を説明する．

5.2　各クラスタの外れ値となる未知ライセンス記述の除去

先行研究 [7] のクラスタリングの結果としては同じクラスタに属するが同じクラスタ内では外れ値とみなせる未知ライセンス記述が存在すれば，それらを排除することで同じ種類のライセンス記述から構成されるクラスタがなるべく多くなるようにする．

外れ値とみなせる未知ライセンス記述を検出するにあたって，まず，各クラスタにおいてクラスタの中心となる未知ライセンス記述を決定する．クラスタの中心となる未知ライセンス記述とは，ある閾値以上の類似度となるような他の未知ライセンス記述と最も多くのペアとなり得る未知ライセンス記述を指す．そのため本処理では，各クラスタに属するすべての未知ライセンス記述のペアの類似度を求める．類似度は，2 つの未知ライセンス記述間の編集距離 [9] を 2 つのライセンス記述のうち長い方の文字数で割ることで正規化したものを用いる．クラスタの中心となる未知ライセンス記述を決定した後，その記述との類似度が閾値以下となる未知ライセンス記述を外れ値とみなし除去する．

5.3　ライセンス記述パターン抽出手法

5.3.1　各クラスタにおける系列パターン抽出

本処理では，ライセンス毎に分類されたクラスタ毎に頻出する語句の出現順序を抽出するため系列パターンマイニングを行う．本研究では，高速な系列パターンマイニングアルゴリズム BIDE [10] を用いて，飽和系列パターンを抽出する．飽和系列パターンとは，同じ支持度（文章中に出現する回数）の系列パターンの中で他の系列パターンの部分集合にならない系列パターンを指す．例えば，表 1 に示す 3 つの文章があった時，抽出される飽和系列パターンは表 1 のようになる．表 1 では，“(This),(is),(free)” と “(is),(free)” という共に支持度 2 の系列パターンが確認できる．ただし，“(is),(free)” は，“(This),(is),(free)” の部分集合となるため飽和系列パターンではない．また，本研究では，著作権を示す文（“Copyright (c) [yyyy] *** inc. All rights reserved.” 等）をライセンス記述ではないとした．そのため，著作権を示す文を削除後，各クラスタに対し BIDE を適用する．

5.3.2　飽和系列パターンからのライセンス記述パターンの作成

BIDE は飽和系列パターン抽出するが，異なる支持度の飽和系列パターン間においては他の系列パターンの部分集合となることを許容するため，ライセンスルールの作成に特化しない系列パターンを抽出してしまうことがある．例えば，表 2 において，“(This),(is)” は，“(This),(is),(free)” の部分集合である．しかし，“(This),(is)” の系列パターンは，一般的な英文にも対応してしまうためライセンス特定のための

ライセンスルール作成に用いるのは不適切である．本研究では，ライセンス記述以外の記述に対応する系列パターンの出力を防ぐために，各クラスタで抽出された系列パターンに対して，他の系列パターンの部分集合となる系列パターンをマージする．

6 ケーススタディ

提案手法の有用性を確かめるために行ったケーススタディについて述べる．

6.1 データセット

本ケーススタディでは，Debian v7.8.0 の 12,725 個のソースファイルのライセンスを Ninka で特定し，Ninka が未知ライセンスであると判定した 2,838 個のソースファイル中の未知ライセンス記述をデータセットとした．12,725 個のソースファイルは，できるだけ多くのライセンスを収集するために 12,725 個の各ソフトウェアパッケージ [1] から 1 ファイルずつランダムに抽出したものである．2,838 件の未知ライセンス記述を第一著者が目視で確認したところ，194 種類のライセンスが存在することを確認した．ただし，第一著者がライセンスを目視した際，“GPLv2.1” など，本来存在しないはずの（ライセンス記述を書いた開発者が LGPLv2.1 と混同したか，あるいは，意図的に記述した）ライセンス名が書かれているライセンス記述が 20 件見つかった．提案手法は，未知ライセンスファイルを目視することを想定していないため，このような未知ライセンスファイルは従来とは異なるライセンスとして扱いデータセットに含めることとした．

6.2 評価方法

本ケーススタディでは，提案手法の有用性を 4 章で述べた 2 つの技術的課題をどの程度解決できるかで評価する．以下に，それぞれの評価項目について説明する．

課題 1 の評価項目 先行研究 [7] のクラスタリング手法を本データセットに適用した場合，194 個のクラスタが作成される（1 つのクラスタは 1 種類の未知ライセンス記述のみから構成される）のが理想的であるが，表記ゆれ等が原因で実際には複数の種類の未知ライセンス記述が混在したクラスタが作成される [7]．本ケーススタディでは，提案手法により複数の種類の未知ライセンス記述が混在するクラスタを単一の種類の未知ライセンス記述から構成されるクラスタにできたかどうかを評価する．

課題 2 の評価項目 5.3.2 項で示したように，異なる支持度の飽和系列パターン間で系列パターンの部分集合が存在すると，不必要に多くの種類のライセンス記述に適合する，すなわち，ライセンスを一意に特定するためのものではないライセンスルールを作成してしまうことにつながる．また，1 つの未知ライセンス記述が複数のライセンスルールに適合することになるため，その場合は開発者が目視によりライセンスを確認する必要が生じるため，ライセンス特定作業の効率化という本研究の目的からも望ましい状況とは言えなくなる．本ケーススタディでは，1 種類のライセンスにのみ適合するライセンス記述パターンを提案手法がどれだけ多く抽出できたかどうかを評価する．

6.3 実験方法

6.3.1 クラスタの中心となるライセンス記述検出における閾値

5.2 節で述べたように，外れ値とみなせる未知ライセンス記述の除去においては，クラスタの中心となる未知ライセンス記述を判定するための類似度の閾値を定めている．本研究における閾値は，2 つの未知ライセンス記述が同じ種類のライセンスであると判定するための境界を意味する．本ケーススタディで用いる閾値を決定するために，データセットの未知ライセンス記述以外の Debian v7.8.0 のライセンスの種類が異なるライセンス記述のペア 1,000 組の類似度を予備調査した．予備調査の結

[1] http://ftp.riken.jp/pub/Linux/debian/debian-cd/7.8.0/source/iso-dvd/

表 3　クラスタを評価するメトリクス

メトリクス名	概要
C	2 つ以上のライセンス記述が属しているクラスタ数
SLC (Single License Clusters)	単一のライセンスの未知ライセンス記述から成るクラスタ
Non-SLC (Non Single License Clusters)	種類が異なる未知ライセンス記述から成るクラスタ
SE (Single Element)	単一要素（クラスタにならなかった未知ライセンス記述）
RSLC (Ratio of SLC)	作成したクラスタに占める SLC の割合
RNonSLC (Ratio of Non-SLC)	作成したクラスタに占める Non-SLC の割合
#LSLC (The Number of Licenses in SLC)	作成された SLC に含まれるライセンス種類数

表 4　外れ値となる未知ライセンス記述除去の効果

プロジェクト	外れ値除去 (前/後)	C	SLC	Non-SLC	RSLC	RNonSLC	SE	#LSLC
Debian v7.8.0	外れ値除去前	246	170	76	0.69	0.30	48	48
ファイル数 (2,446)	外れ値除去後	126	120	6	**0.95**	0.05	2008	34

果，1000 組のペアの類似度の平均値と中央値は，それぞれ 28.6%と 22.1%であったが，最大値は 93.5%と非常に高い類似度を示すライセンス記述ペアも存在した．原因を詳細に調べた結果，平均に比べて非常に高い類似度を示したライセンス記述のペアは，LGPLv3 と LGPLv2.1 のようにバージョンの値と著作権表示のみしか差異がないような場合であり，互いのライセンス記述が酷似しているものであった．

[7] のクラスタリング手法にはあらかじめ GPL 系および BSD 系ライセンスを別の方法で抽出し取り除く処理が含まれているため，本ケーススタディの結果に対しては大きな影響を与えないはずである．しかし，[7] のケーススタディでは，GPL 系ライセンスのバージョンを選択させるオプションの記述部分に表記揺れが含まれている場合，完全には GPL 系ライセンスを取り除けない場合が存在することも示されている．このため，ライセンス記述が酷似していても区別できるように，本ケーススタディでは閾値を予備実験における類似度の最大値を考慮した 94%に設定する．

6.3.2　課題 1 の評価実験

データセットに [7] のクラスタリング手法を適用して得られたクラスタに，外れ値となる未知ライセンス記述の除去処理を施し，[7] によるクラスタリング結果と，除去処理後のクラスタリング結果を比較する．クラスタリング結果を定量的に比較するために定義したメトリクスを表 3 に示す．本ケーススタディでは，単一のライセンスの未知ライセンス記述からなるクラスタを SLC として示す．RSLC は，作成されたクラスタに占める SLC の割合を示しており，課題 1 は，外れ値となる未知ライセンス記述の除去処理後に RSLC がどの程度向上するかを評価する．理想的には全てのクラスタが SLC になることであり，その場合の RSLC の値は 1 になる．なお，除去された未知ライセンス記述や，類似度が閾値以上となる未知ライセンス記述がクラスタ内に存在しない場合は，クラスタを解体し全て単一要素（SE）として扱う．

6.3.3　課題 2 の評価実験

提案手法により各クラスタから抽出されたライセンス記述パターンが，1 種類のライセンスにのみ適合するかどうかを grep を用いて検証する．grep による検証を行うため，抽出された全ライセンス記述パターンを正規表現化する．例えば，“(This),(program),(is),(free)” の場合，“This.* program.* is.* free” とすることで grep の条件として用いることができる．データセットに対して grep を行い，マッチした未知ライセンス記述のライセンスが全て同一であれば，そのライセンス記述パターンは 1 種類のライセンスのみに対応できたとする．

また，提案手法適用時，BIDE の支持度は 2 に設定した．これは，抽出したライセンス記述パターンが，少数のライセンス記述にしか存在しない表記揺れが含まれている未知ライセンス記述に対応できるようにするためである．

表 5　提案手法によって抽出されたライセンス記述パターン

ライセンス記述パターン	ライセンス名	#Lic
See the License for the specific language governing rights and limitat[...]	AOLserver	6
If you do not delete the provisions above, a recipient may use your ve[...]	AOLserver	1
If you wish to allow use of your version of this file only under the t[...]	AOLserver	1
The Original Code is AOLserver Code and related documentation distribu[...]	AOLserver	1
The Initial Developer of the Original Code is America Online, Inc. Por[...]	AOLserver	1
The contents of this file are subject to the AOLserver Public License [...]	AOLserver	1
Alternatively, the contents of this file may be used under the terms o[...]	AOLserver	1
Software distributed under the License is distributed on an ”AS IS” ba[...]	AOLserver	5
You may obtain a copy of the License at http://aolserver.com/.	AOLserver	1
This software/database is a ”United States Government Work” under the [...]	PublicDomain	1
PUBLIC DOMAIN NOTICE National Center for Biotechnology Information	PublicDomain	1
The National Library of Medicine and the U.S. Government have not plac[...]	PublicDomain	1
The NLM and the U.S. Government disclaim all warranties, express or im[...]	PublicDomain	1
2 - This Source Code Form is subject to the terms of the Mozilla Publi[...]	MPLv2.0	1
If a copy of the MPL was not distributed with this file, You can obtai[...]	MPLv2.0	1
This program is distributed in the hope that it will be useful, but WITH[...]	LGPL	32
You should have received a copy of the GNU Lesser General Public Lic[...]	LGPL	7

系列を示す括弧を外して表示．#Lic はライセンス記述パターンが対応するライセンス種類数を示す．

表 6　1 種類のライセンスに対応したライセンス記述パターン数

	得られた数	1 種類のライセンスに対応した数	割合
飽和系列パターン	1498	363	24.2%
ライセンス記述パターン	371	240	64.7%

6.4　実験結果

6.4.1　課題 1 の評価実験結果

表 4 に，既存手法をデータセットに適用して得られたクラスタリング結果（外れ値除去前のクラスタリング結果）と，各クラスタの外れ値となる未知ライセンス記述を除去した後のクラスタリング結果を示す．外れ値除去前のクラスタリング結果は，246 個のクラスタのうち 170 個が SLC であり，RSLC は 0.69 であった．一方，外れ値除去前のクラスタリング結果は，クラスタが 126 個に減少し，その 95%にあたる 120 が SLC となることが分かった．この結果から，提案手法は全てのクラスタを SLC にすることができない，大部分を単一ライセンスの未知ライセンス記述から構成されるクラスタを生成することができると言える．

6.4.2　課題 2 の評価実験結果

データセットに対して提案手法を適用した結果，126 のクラスタから 371 のライセンス記述パターンを抽出できた．具体例として，4 つの SLC から抽出されたライセンス記述パターンとそのライセンス名を表 5 に示す．表 5 から，PublicDomain と MPLv2.0 に対応するライセンス記述パターンは，全て 1 種類のライセンスの未知ライセンス記述のみにマッチした．これらは，変更を加えずにライセンスルールとして利用できる．一方，AOLserver と LGPL に対応するライセンス記述パターンは，複数のライセンスの未知ライセンス記述にマッチしているものが含まれている．これらは，そのままでは，ライセンスルールとして利用できない．

次に，BIDE によって抽出された飽和系列パターンの中で 1 種類のライセンスにのみに適合した数と，飽和系列パターン間の部分集合をマージする処理を行う提案手法により抽出されたライセンス記述パターンの中で 1 種類のライセンスにのみに適合した数を表 6 に示す．BIDE では，1498 の飽和系列パターンが抽出されその内 363 件（約 24.2%）が 1 種類のライセンスのみに適合していた．一方，提案手法は 371 のライセンス記述パターンを抽出でき，その内 240 件（約 64.7%）が 1 種類のライセンスのみに適合しており，提案手法により抽出パターン数を大幅に削減できると同時に 1 種類のライセンスのみに適合する割合も向上すると言える．

7 考察

7.1 課題 1

課題 1 についての評価実験では，クラスタの外れ値となるライセンス記述を除去することによって，126 のクラスタのうち，95%に当たる 120 のクラスタを SLC にすることができた．一方，6 つの Non-SLC を SLC にすることはできなかった．この Non-SLC に属しているライセンスを確認したところ，CeCILL-B，CeCILL-C が属していた．これらのライセンスのライセンス記述は，記述中に複数回出現する “B” や，“C” などの一文字レベルの差異しかなく，類似度で判別するのは困難である．そのため，本提案手法と合わせて既存手法であるクラスタリング手法 [7] を改善し，これらを事前に grep で取り除けるようにすることで，提案手法が抽出するライセンス記述パターンの品質向上が期待できるといえる．

また，SE が 48 から 2008 まで増加している．これは，外れ値除去のための閾値が高いため，外れ値として検出されるべきではないライセンス記述が外れ値となり，クラスタから除去された多くのライセンス記述が SE として算出されているためである．この問題に対し，1 つのクラスタから 1 つの SLC を生成するのではなく，1 つのクラスタから複数の SLC を生成できるよう今後手法を改善したい．

7.2 課題 2

課題 2 についての評価実験では，126 個のクラスタから，1 種類のライセンスのみに対応するライセンス記述パターンを 240 個抽出できたことが分かった．表 6 から，PublicDomain や，AOLserver ライセンスのような OSI[2] に承認されていないライセンス，つまり OSS のライセンスの中では，著名でないライセンスからもライセンス記述パターンを抽出することができた．

表 6 から，抽出したライセンス記述パターンの中に，MPLv2.0 に対応するものが含まれている．MPLv2.0 は，Ninka にライセンスルールが登録されているにもかかわらず，未知のライセンスとして出力されることが報告されている [11]．著者らが調査したところ，原因は正規表現作成時の人為的ミスであった．このことから本手法で抽出されたライセンス記述パターンは，手作業によるライセンスルール作成における人為的ミスをカバーする可能性がある．

また，LGPL に対応するライセンス記述パターンの中に 32 種類のライセンスに対応するライセンス記述パターンが含まれていた．この記述は，免責条項に関する記述であり，その他多くのライセンスのライセンス記述で，同様の記述が確認できた．このような複数のライセンスに対応するライセンス記述パターンは，それ単体では，1 種類のみに対応できないがいくつかのライセンス記述パターンを組み合わせることで，ライセンスを 1 種類に対応できる可能性がある．そのため，複数のライセンスに対応するライセンス記述パターンは，それ以外のライセンス記述パターンと組み合わせ，一つのライセンスルールとして利用することができるといえる．

8 関連研究

ライセンス不整合に関する研究として，Lokhamn ら [12] や Alspaugh ら [13] がある．ライセンス不整合とは，複数のライセンスにおいて，それぞれのライセンスが持つ条項間に矛盾が発生しており，全てのライセンスを同時に満たせない状態を指す．Lokhamn ら [12] は，ソフトウェアアーキテクチャレベルで，ライセンス衝突を検出するツールを提案している．また，Alspaugh ら [13] は，ライセンスの権利と義務をタプルで表現し，ライセンスの衝突を計算する手法を提案している．これらの研究は，最終的に，ライセンス違反のリスクを減らすことができる点において本研究と関連している．ただし，本研究は，ライセンス特定ツールの精度を上げることでラ

[2] https://opensource.org

イセンス違反リスクを減らすため，これらの研究のアプローチとは異なっている．

パターンマイニングを用いたコーディングパターンの抽出に関する研究として，石尾ら [14] は，複数のコンポーネントに分散した定型的なコードの検出を目的とした系列パターンマイニングによるコーディングパターンマイニング手法を提案している．石尾らの研究は，ソースファイルに対して，系列パターンマイニングを行っている点において本研究と関連している．ただし，本研究は，ライセンスルール作成に必要なライセンス記述パターン抽出するために，系列パターンマイニングを行っている．そのため，石尾らの研究とは目的が異なっている．

9 まとめ

本研究では，ライセンスルール自動生成手法構築のため，[7] によるライセンス記述のクラスタリング結果から，ライセンスルールを作成するためのライセンス記述パターン抽出手法を提案した．提案手法の有用性を確かめるため，Debian v7.8.0 のソフトウェアパッケージ集合から検出された 2,838 の未知ライセンスファイルを対象としたケーススタディを行った．その結果，126 個のクラスタから，371 個のライセンス記述パターンを抽出することができた．そのうち 240 個のライセンス記述パターンは，変更を加えずにライセンスルールとして使えるパターンであった．

今後の課題としては，データセットに存在するライセンスと，ライセンス記述パターンを抽出できたライセンスの関係を明らかにすること．また，複数のライセンスに共通する記述を一つのライセンス記述パターンとしてまとめるために，ライセンス記述の共通部分と独立部分に分解してから本手法を適用できるようにすることが望まれる．

謝辞 本稿の執筆にあたり助言を頂いた和歌山大学の大平雅雄氏に感謝致します．

参考文献

[1] 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, Sep. 2010.
[2] Robert Gobeille. The fossology project. In *Proc. of MSR'08*, pp. 47–50, May 2008.
[3] OSLC. http://forge.ow2.org/projects/oslcv3/.
[4] what license. http://www.what-license.com/.
[5] Ohcount. https://github.com/blackducksw/ohcount.
[6] Timo Tuunanen, Jussi Koskinen, and Tommi Krkkinen. Retrieving open source software licenses. In *Proc. of OSS '06*, pp. 35–46, Jun. 2006.
[7] Yunosuke Higashi, Yuki Manabe, and Masao Ohira. Clustering oss license statements toward automatic generation of license rules. In *Proc. of IWESEP'16*, pp. 30–35, Mar. 2016.
[8] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, Vol. 58, No. 301, pp. 236–244, Mar 1963.
[9] VI Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, Vol. 10, pp. 707–710, 1966.
[10] Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *Proc. of ICDE '04*, pp. 79–90, Mar. 2004.
[11] Yuhao Wu, Yuki Manabe, Tetsuya Kanda, Daniel M. German, and Katsuro Inoue. A method to detect license inconsistencies in large-scale open source projects. In *Proc. of MSR '15*, pp. 324–333, May 2015.
[12] Alexander Lokhman, Antti Luoto, Imed Hammouda, and Tommi Mikkonen. Open source legality compliance of software architecture, a licensing profile approach. In *Proc. of ICSEA'13*, pp. 571–578, Oct. 2013.
[13] Thomas A. Alspaugh, Hazeline U. Asuncion, and Walt Scacchi. Analyzing software licenses in open architecture software systems. In *Proc. of ICSE '09 Workshop on Emerging Trends in FLOSS '09*, pp. 54–57, May 2009.
[14] 石尾隆, 伊達浩典, 三宅達也, 井上克郎. シーケンシャルパターンマイニングを用いた コーディングパターン抽出. 情報処理学会論文誌, Vol. 50, No. 2, pp. 860–871, Feb. 2009.