

Clustering OSS License Statements Toward Automatic Generation of License Rules

Yunosuke Higashi
Wakayama Univ.
Japan

Manabe Yuki
Kumamoto Univ.
Japan

Masao Ohira
Wakayama Univ.
Japan

Hello everyone, I'm Yunosuke Higashi from Wakayama University in Japan.

Today, I'd like to present our study titled **Clustering OSS License Statements** toward automatic generation of license rules.

Background

- Utilizing Open Source Software (OSS) is a means of reducing production costs in the modern software development
- OSS is reusable as part of a proprietary software product if it strictly complies with OSS licenses described in source files
- Identifying OSS licenses manually is a time-consuming task
- In order to help identify OSS licenses, license identification tools have been proposed

First of all, I'll talk about our research background.

Utilizing Open Source Software (OSS) is a means of reducing production costs in the modern software development.

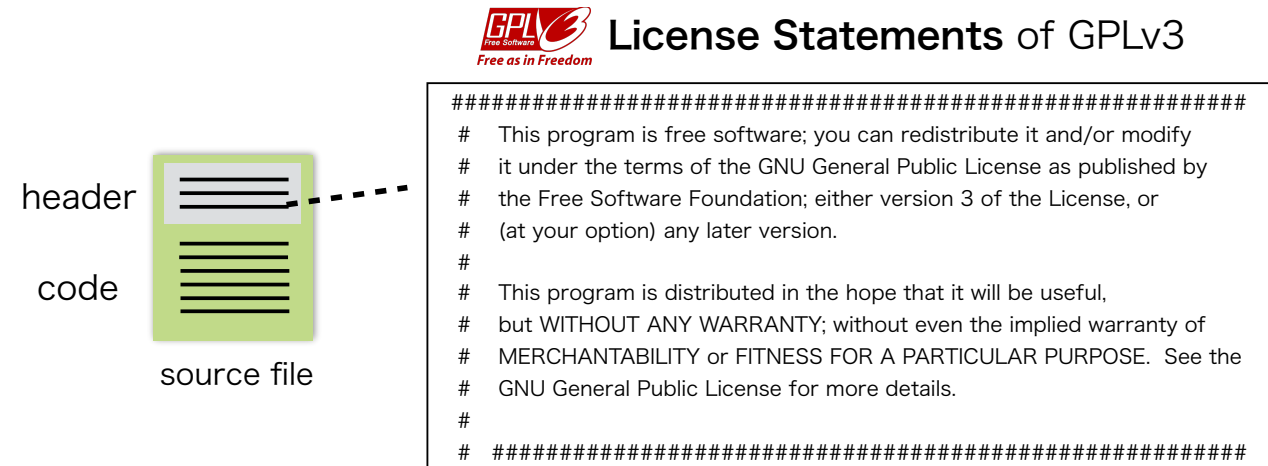
OSS is also reusable as part of a proprietary software product if it strictly complies with OSS licenses described in source files.

Identifying OSS licenses manually is a time-consuming task.

In order to help identify OSS licenses, license identification tools have been proposed.

License Statements of Source Files

- An OSS license is declared in a header part of each source file as license statements



In general, An OSS license is declared in a header part of each source file as license statements.
For example, This figure describes license statements of GPLv3.

License Identification Tools

- License Identification Tools[1][2] automatically identify licenses of each source files



- [1] **Ninka** : Daniel M German, Yuki Manabe, and Katsuro Inoue, "A Sentence-Matching Method for Automatic License Identification of Source Code Files" In Proceedings of ASE, pages 437-446, September 2010
- [2] **Fossology** :R. Gobeille, "The fossology project," in Proc. of MSR'08, 2008, pp. 47–50.

Here, I'd like to describe why Identifying OSS licenses is so time-consuming task.

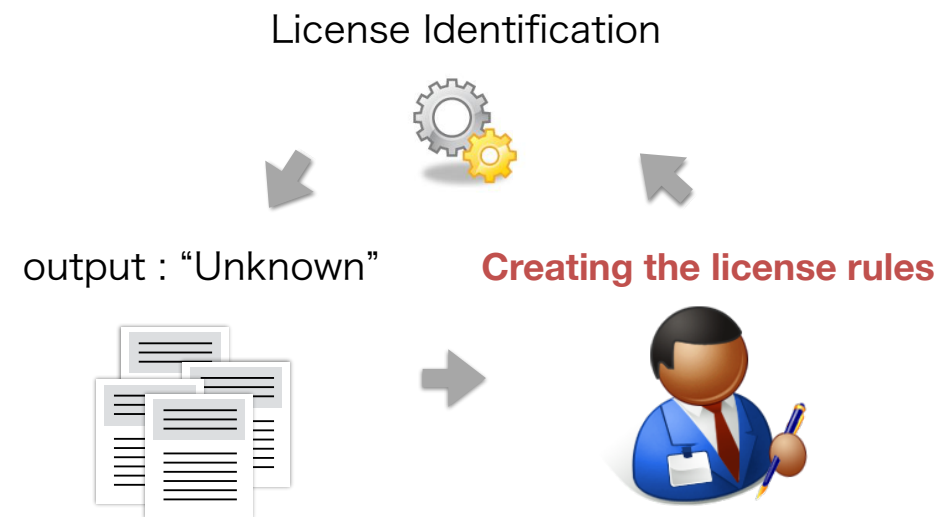
In order to help identify OSS licenses, license identification tools have been proposed.
License Identification Tools automatically identify licenses of each source files by using regular expressions.

When source files are inputted into the tools, it output the license name of source files such as GPLv3.

In case the License Identification tools do not have license identification rules for unknown licenses, the tools reports they are "unknown licenses".

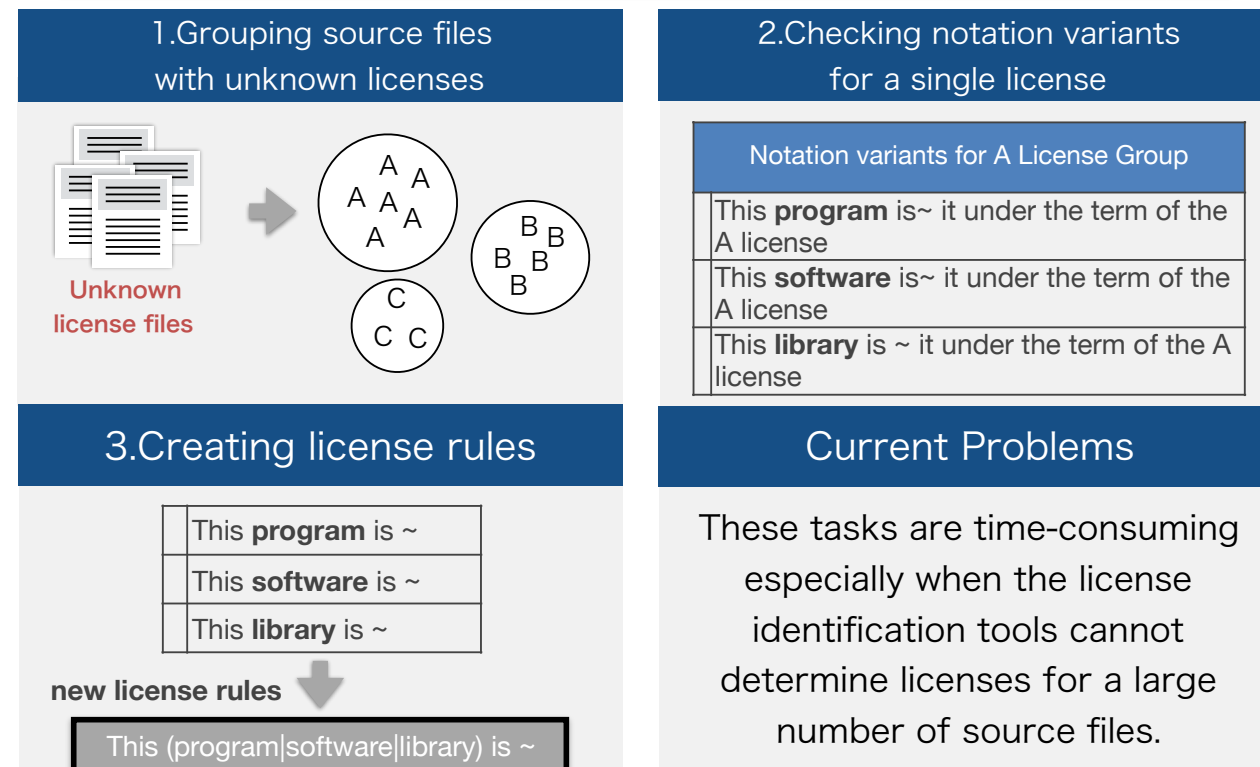
The Weakness of License Identification Tools

- Rule-based license identification tools will require to manually and constantly add new regular expressions to the tools



However, license identification tools will demand to manually and constantly add new regular expressions to the tools to identify new licenses, every time when the tools encounter new OSS licenses or the tools unexpectedly judges some licenses as unknown due to notation variants in license statements.

The Process of Creating License Rules



Clustering OSS License Statements Toward Automatic Generation of License Rules

6

The process of manual creation for new license rules follows the steps 1 to 3 below.
In Step 1, Unknown license files are reviewed and grouped by license manually.

In Step 2, Checking notation variants are checked for a single license.
Even if the source files are grouped by license, there often exist notation variants in license statements for a single license.

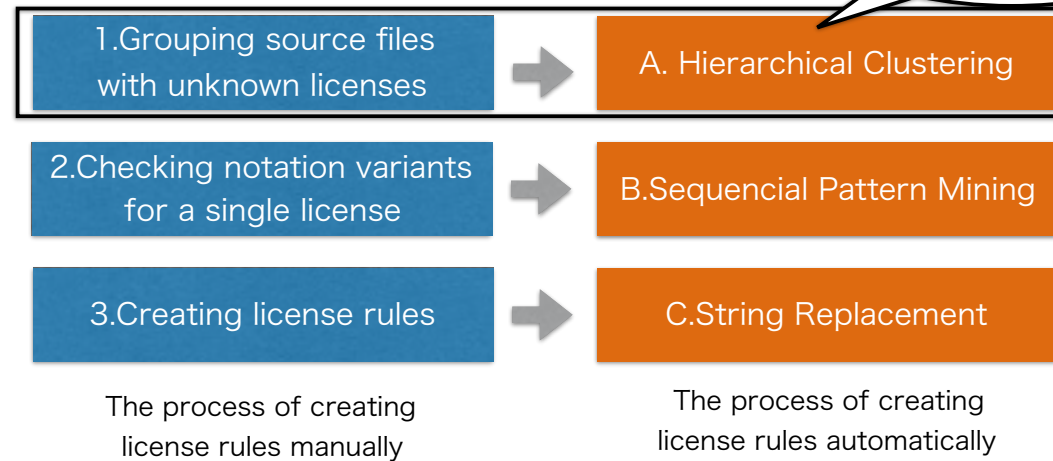
for example, there are three notation variants in license A group, “this program-”, ”this software-”, “this library”

In Step3, Based on the reviews in Step 2, license statements are tokenized as regular expressions and license rules are created to be able to match new licenses.

These tasks are time-consuming especially when the license identification tools cannot determine licenses for a large number of source files. Therefor it is desired to automate the process of creating license rules.

Our Final Goal

- Construct a method to automatically generate candidates of license
 - A method which consists of the following steps



we try to tackle with the issue in Step 1

Two Technical Challenges

- Automating **Step 1** requires to address at least two technical challenges as follows

1. Discriminating GPL/BSD-related licenses

- In our pilot study, we found that Ninka judged many source files as “unknown” licenses even for very popular licenses such as GPLv2.
- we try to correctly discriminate GPL/BSD-related licenses

2. Creating groups of source files with unknown licenses

- The number of groups should correspond to the number of licenses existed in software components being reused

Automating Step 1 requires to address at least two technical challenges as follows.

Challenge 1, Discriminating GPL/BSD-related licenses.

In our pilot study where Ninka was used to detect licenses of source files in Debian v.7.8.0, we found that Ninka judged many source files as “unknown” licenses even for very popular licenses such as GPLv2 ,BSD2.

So, we need correctly discriminate GPL/BSD- related licenses , since GPL/BSD-related licenses have a long history and should be well-known licenses.

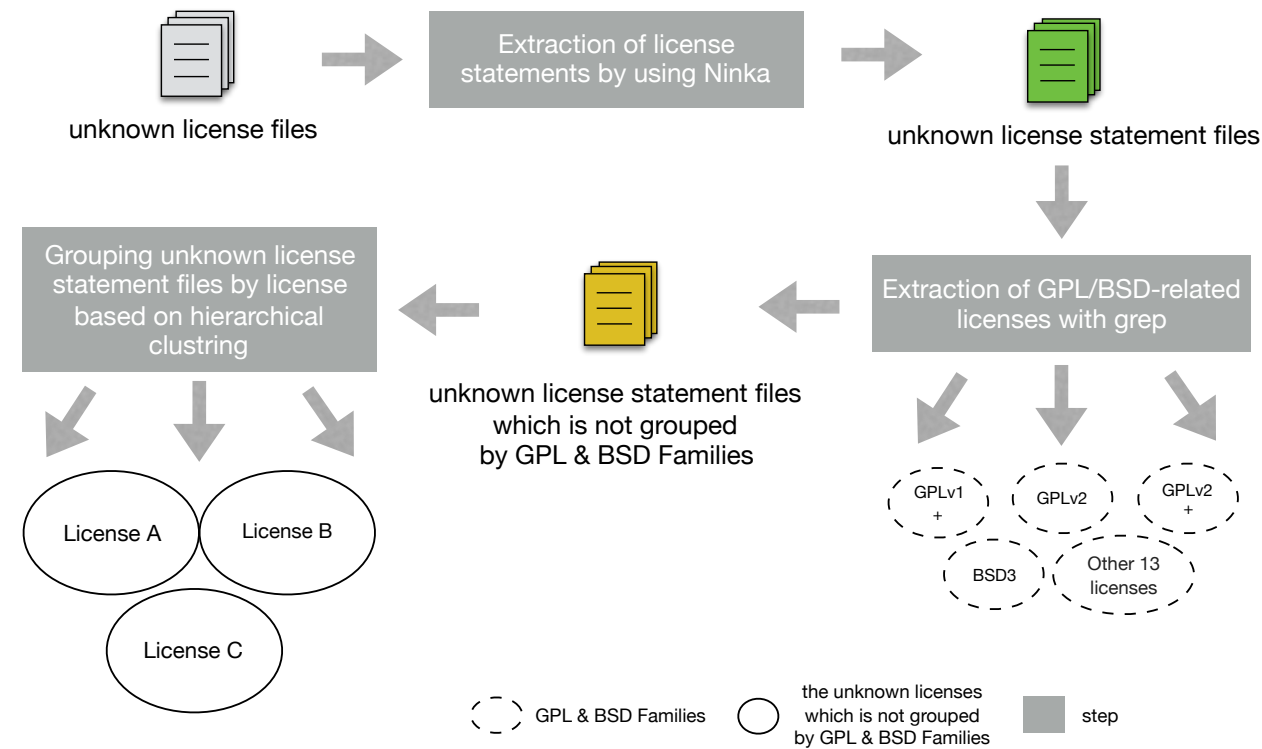
Challenge 2, Creating groups of source files with unknown licenses.

To automate the manual review and grouping in Step 1 described earlier, it is ideal to create groups of source files only with a single license.

The number of groups should correspond to the number of licenses existed in software components being reused.

And, toward Step 2 and Step 3, groups should not be divided due to notation variations for a single license.

An Overview of the Clustering Method



Clustering OSS License Statements Toward Automatic Generation of License Rules

9

This figure, shows an overview of our clustering method.

According to the following procedure, the method creates clusters of unknown license files which are reported by license identification tools.

First, As pre-processing, license statements are extracted from unknown license files by using a license identification tool such as Ninka.

Second, Extract the GPL/BSD-related licenses files are extracted with grep. The extracted license statements are separately grouped so that similar but discriminable licenses such as GPL version 1, 2 and 3 can be correctly identified.

Third, The rest of unknown license files which is not grouped in the previous step are divided into clusters of unknown license files, based on a dendrogram created by hierarchical clustering.

In what follows, we only describe the clustering procedure in detail.

Please allow me to leave out the whole explanation of our method

Grouping of Unknown License Files

1. The unknown license statements files are converted into Bag of Words vectors
2. A dendrogram is created by Ward's method from the unknown license statement files which are not grouped
3. Cutting the tree(dendrogram) at a higher place would yield small clusters
 - in this study the same type of license statements should become a single cluster
 - We introduce two kinds of conditions for cutting the tree in a dendrogram multiple times at higher places

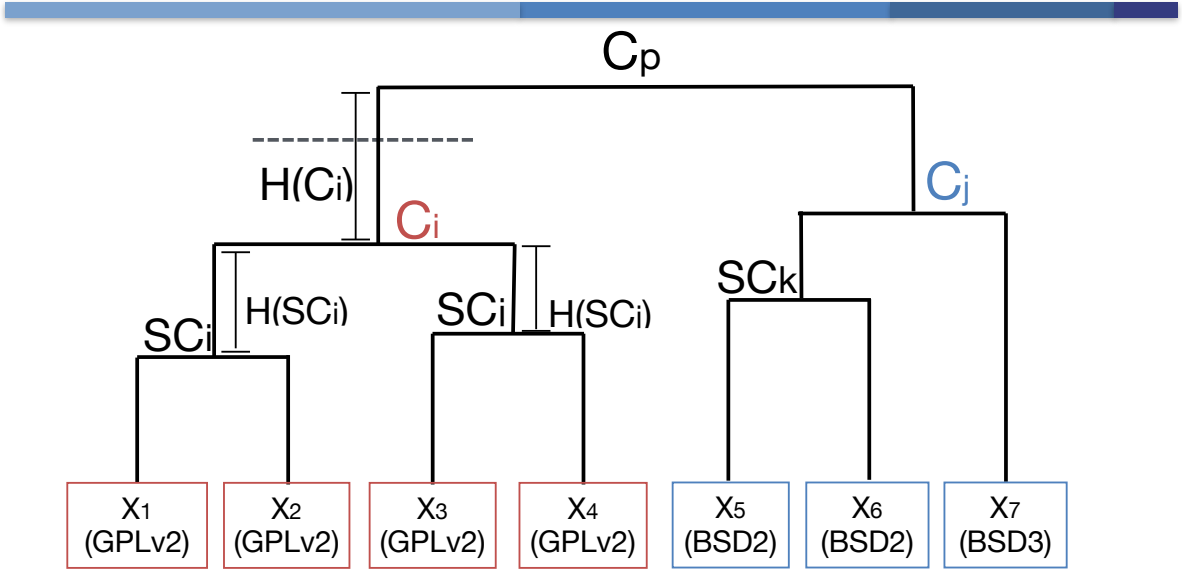
I 'll describe in detail of Creating groups of source files with unknown licenses.
Hierarchical Clustering used in this study.

In Step 1, The unknown license statements files are converted into Bag of Words vectors.
In Step 2, A dendrogram is created by Ward's method from the unknown license statements.
In Step 3, Cutting the tree at a higher place would yield small clusters.

In the general usage of a dendrogram, clusters are obtained only once by cutting the tree at any height.
Cutting the tree at a lower place would yield large clusters which would contain different kinds of license statement files.

Since in this study the same type of license statements should become a single cluster, we introduce two kinds of conditions for cutting the tree in a dendrogram multiple times at higher places.

Condition α



Condition α

$$H(C_i) > H(SC_i) \text{ and } H(C_i) > H(SC_j)$$

This figure show an example of dendrogram.
In this figure, X represents one data and Cx represents one cluster which is created by merging a pair of two data or a pair of two clusters
In Addition, SC represents one samallest cluster which is created using dissimilarity a pair of only two data.

Here, the condition α is used to determine whether to create a large cluster.
For instance, as shown in the Figure , the cluster Ci should be isolated from Cj since Ci includes license statements files with a single license (i.e., GPLv2).
Ci can be created by cutting the tree between Cp and Ci.

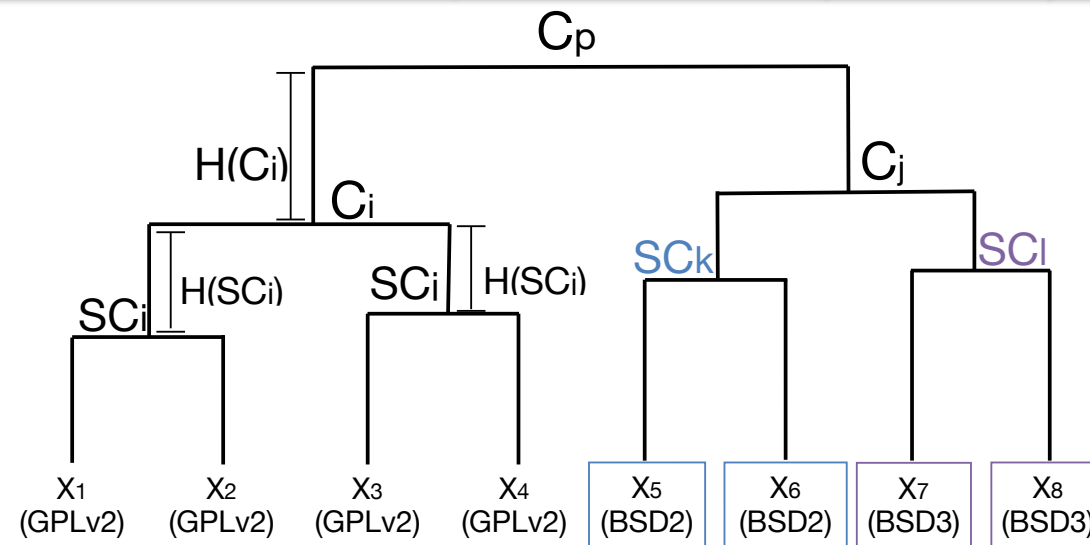
In our study, We focus on the height of a cluster which is show H(Cx) in this figure.
The height of a cluster is also determined by the dissimilarity between the cluster and the large cluster

As the clusters merged, the height of the cluster become longer in dendrogram.
#訳してください。階層クラスタリングでは、非類似度が低いものを順にマージしていくからです。

Ci can be created by cutting the tree between Cp and Ci. In order to do so, the tree will be cut if the following equations are met.

The height of a smallest cluster is uniquely-determined from the dissimilarity between a pair of two data (Cx)
If the equations are not met, the same processing are applied to the next level of clusters.

Condition β



• Condition β

- An extension of the condition α that the cluster is not created by two clusters.

Next, I'll describe of the condition β .

The condition β is an extension of the condition α that the cluster is not created by two clusters.

#訳してください : なぜなら, 2つのクラスタのマージは, それぞれ異なるライセンスで構成されるSLCを一つにしてしまう可能性があるからです.

For instance, as shown in the Figure, the cluster SC_k should be isolated from C_j since SC_k includes three BSD2 licenses but SC_l has three BSD3 licenses.

in only condition α , C_j might be able to created by cutting the tree between C_p and C_j .

However, C_j is not cluster with single license.

Case Study

- Purpose :
 - Created clusters by our method would be better candidates for automating the license rule generation
- Dataset
 - Ninka was used to identify OSS licenses of the sampled 12,735 source files from each software package in Debian v7.8.0.

Project	#Unknown Files	#Licenses
Debian v7.8.0	2,453	118

We conduct a case study to confirm the usefulness of our clustering method.

first, I'll describe the dataset which are applied to our clustering method.

In the case study, We apply our method to 2,453 unknown license files of Debian v7.8.0.

To create the dataset, the license identification tool “Ninka” was used to identify OSS licenses of the sampled 12,735 source files from each software package in Debian v7.8.0. As a result, 2,453 unknown license files were obtained as a set of unknown license statements files.

In order to identify the license of each files in our dataset, We reviewed the dataset manually . As a result, We found the 118 types of licenses in our dataset.

Evaluation of Case Study

- Condition : Ideal (Baseline)
 - Simply cutting the dendrogram at the height which can yield the number of licenses of clusters in dataset
 - 118 licenses exist in our dataset
- Evaluation
 - **We use the metrics** for evaluation of the clustering with condition α, β & ideal

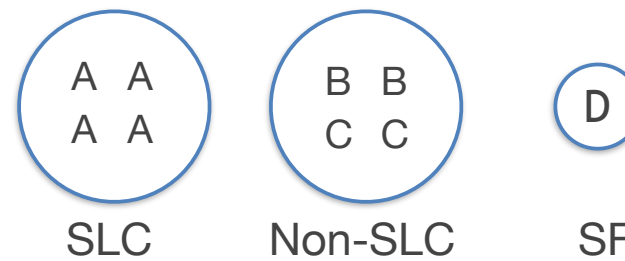
Our hierarchical clustering method are applied to 2,453 unknown license statements files.
We also conduct the case study with an ideal case for baseline.

In the ideal case, We conduct cutting the dendrogram at the height which can yield the number of licenses of clusters in dataset.
#訳してください : Idealでは、単に、118 (データセットに存在するライセンス数) のクラスタに分けることができる高さで、枝刈りを行います。

We use the metrics to evaluate clustering results with condition α, β and ideal.
In what follows, I'll describe the metrics for evaluate our method.

Metrics for Evaluation Created Clusters

- SLC (Single License Cluster)
 - The number of clusters with a single license
 - **Creating more SLC is desired in this study.**
- SF (Single Files)
 - The number of single files which are not clustered
 - **SF are not desired because creating a license rule for each single file is not efficient.**



Clustering OSS License Statements Toward Automatic Generation of License Rules

15

We define two types of metrics.

first, we define the SLC metrics for evaluation of the number of clusters with a single license.
like this figure, it is ideal to contain the same type of license statements files in each cluster.

Second, We define the SF metrics for evaluation of the number of single files which are not clustered.
like this figure, SF are not desired , because creating a license rule for each single file is not efficient.

Results of Evaluation Clusters

	C	SLC	SF	SLC / C	SF / C
cond. α	247	153	48	0.62	0.19
cond. β	719	409	207	0.52	0.29
ideal	81	31	37	0.36	0.46

- SLC
 - The condition α marks higher the ratio of the number of SLC to the number of C than the condition β .
- SF
 - The condition α marks lower the ratio of the number of SLC to the number of C than the condition β .

The condition α is the better for making the cluster with a single license than condition β

The table summarizes the result of the evaluation.
C shows the number of clusters which are created by each condition.

The condition α and β create more SLCs than the ideal.

The condition β creates more SLCs than the condition α .
But, The condition α marks higher ratio of the number of SLC to the number of C than the cond. β .

The ideal case creates less SF than the condition α .
But, The condition α marks lower ratio of the number of SLC to the number of C than ideal case.

Therefore, The condition α is the better for making the cluster with a single license than condition β

The Number of Licenses in SLC

- In addition, We evaluate the number of licenses in SLC of each condition α & β

	SLC	#Licenses in SLC	#Licenses in SLC / SLC
cond. α	153	35	0.23
cond. β	409	42	0.10

- The cond. α marks higher ratio of #Licenses in SLC to SLC than cond. β

The condition α can create the SLCs which contain the different type of license statements files in each SLCs

In addition, We evaluate the number of licenses in SLC of each condition α and β .

it is ideal to contain the different type of license statements files in each single license clusters.

As a result,the condition. β can include more licenses in single license clusters than condition α .
but, The condition. α marks higher ratio of the number of Licenses in SLC to SLC than condition β .

therefore, The condition α can create the single license clusters which contain the different type of license statements files in each single license clusters

Discussion : Result of Step2

- What clustering conditions are suitable?
 - The condition α is more resistant to create a cluster which includes license statements files with different licenses.
 - All the scores of the ratio with the condition α are better if it is compared to the ideal state of clusters.

Clustering with the condition α are suitable
as candidates for generating license
rules automatically

Now, we will discuss what clustering conditions are suitable as candidates for generating license rules automatically.

In Result of Step2, The cond. α creates clusters with the higher ratio of SLC to created clusters than cond β .
This means that the condition α is more resistant to create a cluster which includes license statements files with different licenses.

In Addition, All the scores of the ratio with the condition α are better if it is compared to the ideal state of clusters.

Based on the discussions above, we can conclude that using the condition α for clustering creates clusters which are suitable as candidates for generating license rules automatically.

Conclusion

- We proposed the method to classify unknown license files
 - Our method created clusters which are suitable as candidates for generating license rules automatically.
- Future Work
 - Our clustering method also must be improved in the near future, since many clusters had license statements files with different licenses.

1. Finally, let me summarize what we've done.
2. In this study, We proposed the method to classify unknown license files.
3. Our method created clusters which are suitable as candidates for generating license rules automatically
4. Our future work , our clustering method also must be improved in the near future, since many clusters had license statements files with different licenses.
5. Thank you for your attention. I would be happy to answer any questions, you have now.

End



Discussion : Result of Step1

- The precision in all was 0.89 that is not so high
 - Precision of BSD2 with grep is 0.39
 - Since Apache License v1.1 is created by adding clauses to and modifying BSD2, the discrimination condition for BSD2 mistakenly corresponds to Apache License v1.1.
 - For this case, the precision can be improved by excluding license statements files which include the term “Apache”.

The precision in all was 0.89 that is not so high

In particular, the precision for BSD2 was the worst (0.39)

In order to understand the reason why the precision for BSD2 was worst, we conducted an additional analysis by manually reviewing the unknown license statements files which failed to be discriminated.

Since Apache License v1.1 is created by adding clauses to and modifying BSD2, the discrimination condition for BSD2 mistakenly corresponds to Apache License v1.1.
For this case, the precision can be improved by excluding license statements files which include the term “Apache”.

GPL/BSD-Related Lic. Extraction with grep

- We try to correctly discriminate GPL/BSD related licenses with keyphrase
 - GPL Family Licenses

GPL license family	Keyphrase	version name
GPLv1	GNU General Public License	version 1
GPLv2	GNU General Public License	version 2
GPLv3	GNU General Public License	version 3
LibraryGPLv2	GNU Library General Public License	version 2
LGPLv2.1	GNU Lesser General Public License	version 2.1
LGPLv3	GNU Lesser General Public License	version 3
AGPLv3	GNU Affero General Public License	version 3

- BSD Family Liceses : BSD2, BSD3, BSD4

i 'll describe in detail of GPL/BSD-related licenses extraction with grep.
Using keyphrase to discriminate GPL/BSD-related licenses from others

this table ,show the keypharase for extraction of GPL Family License.
Using a combination of a keyphrase and a version name, one of the GPL-related license is detected by grep.

in the same way, we set the keyphrase which can discriminate versions of BSD license.

These conditions above are created based on examples of license statements which are provided from Open Source Initiative

Case Study : Step2

- Our hierarchical clustering method are applied to unknown license statements files which are not extracted in Step 1.
 - Condition α, β , ideal
 - Ideal Case
 - Based on the ideal case (i.e., 118 licenses exist in Step 3.), clusters created by our clustering method are evaluated
 - Through reviewing all the unknown license statements files in Step 3 for evaluation, 81 clusters and 37 single files

Step 1: Extraction of GPL/BSD

- Evaluation
 - We evaluate that grep can correctly find GPL/BSD related licenses from the dataset with precision
- Result

Table : Accuracy of Grep For GPL/BSD Lics.

Licenses	TP	FP	Precision
GPLv2	170	4	0.98
GPLv3	45	5	0.90
LibraryGPLv2	19	1	0.95
LGPLv2.1	31	2	0.94
LGPLv3	21	3	0.88
AGPLv3	15	0	1.00
BSD2	17	27	0.39
BSD3	16	1	0.94
BSD4	2	0	1.00
all	342	43	0.89

We evaluate that grep can correctly find GPL/BSD related licenses from the dataset with precision. the results are shown in the table.

Of 2,838 unknown license statements files, 385 files were extracted and grouped by GPL/BSD-related license. we can confirm that the discrimination accuracy of grep in all was 0.89 of precision.

Extraction of BSD licenses with grep

- Discriminant conditions and flow for BSD license family

