LETTER

A Machine Learning Method for Automatic Copyright Notice Identification of Source Files

Shi QIU[†a)], **German M. DANIEL**[††b)], ***Nonmembers***, ***and*** **Katsuro INOUE**[†c)], ***Fellow***

SUMMARY For Free and Open Source Software (FOSS), identifying the copyright notices is important. However, both the collaborative manner of FOSS project development and the large number of source files increase its difficulty. In this paper, we aim at automatically identifying the copyright notices in source files based on machine learning techniques. The evaluation experiment shows that our method outperforms FOSSology, the only existing method based on regular expression.

key words: *software maintenance, open source software, software copyright*

1. Introduction

Software copyright grants the copyright owner declared in the copyright notice a legal right to determine under what conditions this software can be redistributed, reused, and modified. Copyright notice is a few sentences mostly placed in the header part of a source file as a comment or in a license document in a FOSS project. Identifying the copyright notices of source files is important for several reasons: a) the copyright owner is allowed to change its license or to grant a commercial one to a third party; b) the copyright owner is allowed to start legal proceedings to enforce its license [1]; c) several FOSS licenses (e.g. the BSD family of licenses) require that the copyright owner of FOSS projects being reused should be acknowledged in the documentation and other materials of the system that reuses it [2].

However, copyright notice identification of source files is difficult. On one hand, different from proprietary software, FOSS projects are developed in a collaborative manner, receiving contributions from a large number of developers who potentially declare the copyright notice. On another hand, a large FOSS project usually consists of a large number of source files in which the copyright notices are buried.

To overcome these difficulties, a tool named FOSSology* has been developed to automatically identify copyright notice [3]. FOSSology identifies copyright notices in the comments of the source files using regular expression-based matching. However, some sentences related to copyright could be wrongly identified as copyright notices by FOSSology. Table 1 shows examples of the identified copyright notices using FOSSology to a source file named bonito64.h in the Linux kernel. We can see that sentence 1 is identified as a copyright notice incorrectly. This is because the target sentence contained a keyword "copyright", and FOSSology's regular expression matches this keyword and all the following words in that sentence. To solve this problem, we propose a machine learning method in this paper. The proposed method is expected to reject the sentences that are not copyright notices such as sentence 1 in Table 1, and identify all actual copyright notices such as sentence 2 and 3 at the same time. The results of experiments suggest that Decision Tree and Random Forest perform best on automatic copyright notice identification of source files, and the proposed machine learning method outperforms FOSSology.

Table 1 Examples of the identified copyright notices using FOSSology for bonito64.h.

1	copyright message in any source redistribution in whole or part.
2	Copyright (c) 1999 Algorithmics Ltd
3	Copyright (C) 2001 MIPS Technologies, Inc. All rights reserved.

2. Machine Learning Method

In this section, we introduce a machine learning method for automatic copyright notice identification of source files. The proposed method consists of four steps.

1) Copyright-related sentence extraction and preprocessing: We first use a keyword-based method to extract copyright-related sentences from the comments in the source files. A sentence is extracted as a copyright-related sentence when it includes any copyright-related keywords and signals such as "copyright", "©", and "(C)". Some other potentially relevant words, such as "authored by", "written by", etc., are included as well. These heuristics are similar to FOSSology. The difference is that FOSSology uses the keywords to construct regular expressions. For each extracted copyright-related sentence, we tokenize it into words, lemmatize and convert each word to lowercase, and then remove punctuation.

2) Vectorization: We use the numbers of words of different categories as features to vectorize the copyright-related sentence. These features are selected based on our observations on the results of using FOSSology to identify

Manuscript received June 17, 2020.
Manuscript revised August 28, 2020.
Manuscript publicized September 18, 2020.

[†]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan.
[††]The author is with the Department of Computer Science, University of Victoria, Victoria, BC V8P 5C2, Canada.
a) E-mail: qiujitsu@ist.osaka-u.ac.jp
b) E-mail: dmg@uvic.ca
c) E-mail: inoue@ist.osaka-u.ac.jp

DOI: 10.1587/transinf.2020EDL8089

*https://www.fossology.org/

Table 2 The word categorization and the tokens we use to replace words.

Category	Token	Example
Copyright-related keyword	COPYRIGHT	copyright, author
Copyright-related signal	SIGNAL	©, (C), (c)
Year	YEAR	1991, 2002, 2013
E-mail address	EMAIL	addr@email.com
Others	OTHER	license, above

copyright notices. We observed that the wrongly detected copyright notices are significantly different from the actual copyright notices in these features. Although there are many other natural language processing methods to vectorize the sentence, we do not use them for two reasons: (1) There is no suitable training and testing dataset and our manually constructed dataset is not large enough to use these methods; (2) Different from natural language, copyright notice has no explicit grammatical rule, they also contain a lot of organization or individual names appearing only once which will make the constructed dataset very sparse. We believe that our selected features are effective and will examine their effectiveness in Sect. 3.

We categorize the words into five categories, i.e. *copyright-related keyword*, *copyright-related signal*, *year*, *e-mail address*, and *others*. For each word in the copyright-related sentence extracted in Step 1, we replace it into a particular token. Table 2 shows the tokens we used to replace the words of different categories. After replacement, we count the number of tokens of each category and then construct a 5-dimension vector. We end up with a list of vectors representing the extracted copyright-related sentences.

3) Training Classifier: Four supervised classifiers are considered, i.e. Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM). These four classifiers are widely used in text classification tasks [4], [5] and related work addressing similar problem [6]. We implemented them with the Scikit-learn library[†] [7]. We train the classifiers with a manually labeled dataset. Each vector in this dataset is manually labeled as actual copyright notice (i.e. positive copyright notice) or false copyright notice (i.e. negative copyright notice). The task of the trained classifier is to classify a vector representing a copyright-related sentence as a positive copyright notice or a negative one. The details of how we train and evaluate the classifiers will be described in Sect. 3.

4) Copyright notice identification: For each vector, the trained classifier predicts whether it represents a copyright notice or not using the trained classifier. A copyright-related sentence is identified as a copyright notice if the corresponding vector is classified as representing a copyright notice. Finally, we identify all copyright notices of a source file.

3. Comparison of Four Supervised Classifiers

In this section, we first describe how we construct the datasets. We then aim to find which supervised classifier performs best by comparing the performance of the four classifiers.

[†]https://scikit-learn.org/stable/

Table 3 Summary of the target version of the Linux kernel.

Version	4.14
Date	Nov 13, 2017
#File	45,477

3.1 Dataset Construction

To achieve this goal, we choose the Linux kernel - the most popular and successful open-source operating system kernel - as the target dataset. The source code of the Linux kernel is downloaded from Github[††]. Table 3 shows a summary of the target version of the Linux kernel.

We first randomly select 2000 source files from 45,477 source files in the Linux kernel and extract copyright-related sentences using the keyword-based method we described in Step 1 in Sect. 2. We end up with 2,297 copyright-related sentences. Note that the duplicate sentences are removed here. It means that even if a sentence exists in two or more source files, we only record it once. For each sentence, we manually inspect and label it as a positive copyright notice or a negative one. As a result, 2,146 sentences are labeled as positive copyright notices, and 151 sentences are labeled as negative ones respectively. The positive copyright notices and the negative ones are unbalanced, so balancing techniques have to be applied [8], [9]. To address the issue of unbalanced data, we manually create the negative copyright notices by randomly replacing words in 151 found negative copyright notices. A similar manual method has been proven effective in handling imbalance datasets in other software engineering tasks [10]. Note that the words used to do replacement are from the words in all 2,297 copyright-related sentences. In this way, we successfully extend the number of negative copyright notices to 2,146. We finally construct a dataset consisting of 2,146 positive copyright notices and 2,146 negative ones.

3.2 Experiment and Results

To evaluate the performance of four supervised classifiers, we first randomly split the dataset into two parts, 15% for the testing dataset and 85% for the training dataset. This method is also widely used in the existing research about the closely related topic [6]. To train the classifiers, 5-fold cross-validation is performed for the training dataset [11]. In 5-fold cross-validation, the training dataset is partitioned into five equal-sized subsets. Each subset has the same percentage of labels. Every time, four subsets are used to train the classifiers and the remaining one is used for validation. This process iterates 5 times until every fold has been used for testing once. Hyperparameter tuning - the process of determining a good set of hyperparameters - is conducted

[††]https://github.com/torvalds/linux

Table 4 Comparison of four classifiers.

Classifier	Label	Precision	Recall	F1-score
NB	Positive	0.99	0.85	0.91
NB	Negative	0.87	0.99	0.92
DT	Positive	1.0	1.0	1.0
DT	Negative	1.0	1.0	1.0
RF	Positive	1.0	1.0	1.0
RF	Negative	1.0	1.0	1.0
SVM	Positive	1.0	0.98	0.99
SVM	Negative	0.98	1.0	0.99

here to achieve the best performance. We train the classifiers with the best hyperparameters and then evaluate their performance with the testing dataset. To evaluate the performance of four classifiers, we use the following metrics: (1) Precision, which refers to the ratio of the number of correct identification to the total number of identifications made of copyright notices; (2) Recall, which refers to the ratio of the number of correct identification to the total number of manually extracted copyright notices; and (3) F1-score, which is the harmonic mean of the precision and the recall. The results are shown in Table 4.

The results suggest that Decision Tree and Random Forest perform best on automatic copyright notice identification of source files. We will use Random Forest to conduct the evaluation experiment in Sect. 4.

4. Comparison to FOSSology

In this section, we first describe how we construct the dataset. We then aim to evaluate the proposed method by comparing the performance of the proposed method and FOSSology, an existing method based on regular expression.

4.1 Dataset Construction

We still use the source files of the Linux kernel as the target to construct the dataset. To achieve this goal, we randomly select 500 source files from the source files in the same Linux kernel. Note that all these 500 source files are unseen in training classifier. For each source file, we manually check the comments to extract the copyright notices. We extract 537 copyright notices from these randomly selected 500 source files. Among them 351 copyright notices are not duplicated. The task of the evaluation experiment is to identify all 537 copyright notices from the source files.

4.2 Experiment and Results

In our evaluation experiment, we first use the proposed method and FOSSology to identify the copyright notices in the selected 500 source files respectively, and then compare their performances. To evaluate our classifier, we use precision, recall, and F1-score as metrics as well.

Table 5 shows the results. It is easy to know that the proposed method outperforms FOSSology. Especially, both the proposed method and FOSSology achieve 100% recall, which suggests that both two methods do not miss any copyright notice. This is important because identifying all copyright notices is an important metric to evaluate the automatic copyright notice identification. However, the proposed method outperforms FOSSology by achieving 100% precision. Specifically, 152 sentences that are not copyright notices are wrongly identified as copyright notices by FOSSology. The results suggest the effectiveness of the proposed method in rejecting the sentences that are not copyright notices.

Table 5 Evaluation of the proposed method.

Method	Precision	Recall	F1-score
Proposed method (RF)	1.0	1.0	1.0
Fossology	0.78	1.0	0.88

Furthermore, we also compare the execution time of the proposed method and FOSSology. For the proposed method, we compute the execution time by the command line. For FOSSology, we compute the execution time by checking the execution report provided by FOSSology itself. Note that FOSSology is a web service-based tool, the execution time also includes the time used for upload, uncompression, etc. We only compute the time used for copyright notice identification here. As a result, it takes 5.11 s by the proposed method to identify all copyright notices from the source files, while it takes 8.25 s by FOSSology. The results suggest that the proposed method is faster and outperforms FOSSology in the execution time.

5. Conclusion

In this paper, we proposed a machine learning method for automatic copyright notice identification of source files. The results of experiments suggest that Decision Tree and Random Forest perform best on automatic copyright notice identification of source files, and the proposed machine learning method outperforms the existing method. Our work highlights the possibility of applying machine learning method to solve software copyright-related issues, and also creates a possibility of studying the copyright notices in FOSS projects, which is overlooked by the software engineering researchers. In our future work, we will implement a tool to provide the copyright notice identification service to end-users, and also test the proposed method on a larger scale. We also plan to study the copyright notice issues in FOSS projects, such as the reliability of the copyright notices in source files.

All data used here can be accessed at https://github.com/QIU10/ML_copyright.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 18H04094.

References

[1] T. Golder and A. Mayer, "Whose IP is it anyway?," Journal of Intellectual Property Law & Practice, vol.4, no.3, pp.165–175, 2009.
[2] A.M.S. Laurent, Understanding open source and free software licensing: Guide to navigating licensing issues in existing & new software, O'Reilly, 2004.
[3] R. Gobeille, "The FOSSology project," Proc. 5th Working Conference on Mining Software Repositories (MSR), Leipzig, Germany, pp.47–50, May 2008.
[4] G. Forman, "An extensive empirical study of feature selection metrics for text classification," Journal of Machine Learning Research, vol.3, pp.1289–1305, March 2003.
[5] F. Sebastiani, "Machine learning in automated text categorization," ACM Comput. Surv. (CSUR), vol.34, no.1, pp.1–47, 2002.
[6] C. Stanik, L. Montgomery, D. Martens, D. Fucci, and W. Maalej, "A simple NLP-based approach to support onboarding and retention in open source communities," 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), Madrid, Spain, pp.172–182, Sept. 2018.
[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, É. Duchesnay, "Scikit-learn: Machine learning in python," Journal of Machine Learning Research, vol.12, pp.2825–2830, 2011.
[8] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," Intelligent Data Analysis, vol.6, no.5, pp.429–449, 2002.
[9] G.E.A.P.A. Batista, R.C. Prati, and M.C. Monard, "A study of the behavior of several methods for balancing machine learning training data," ACM SIGKDD Explorations Newsletter, vol.6, no.1, pp.20–29, 2004.
[10] C. Vendome, M. Linares-Vásquez, G. Bavota, M. Di Penta, D. German, and D. Poshyvanyk, "Machine learning-based detection of open source license exceptions," 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, pp.118–129, Aug. 2017.
[11] S. Ozdemir, Principles of Data Science, Packt Publishing, 2016.