LicenseRec: Knowledge based Open Source License Recommendation for OSS Projects

Weiwei Xu, Xin Wu, Runzhi He, Minghui Zhou†
School of Computer Science, Peking University, Beijing, China
Key Laboratory of High Confidence Software Technologies, Ministry of Education, China
xuww@stu.pku.edu.cn, {blesswoo, rzhe, zhmh}@pku.edu.cn,

***Abstract*—Open Source license is a prerequisite for open source software, which regulates the use, modification, redistribution, and attribution of the software. Open source license is crucial to the community development and commercial interests of an OSS project, yet choosing a proper license from hundreds of licenses remains challenging. Tools assisting developers to understand the terms and pick the right license have been emerging, while inferring license compatibility on the dependency tree and satisfying the complex needs of developers are beyond the capability of most of them. Thus we propose LicenseRec, an open source license recommendation tool that helps to bridge the gap. LicenseRec performs fine-grained license compatibility checks on OSS projects' code and dependencies, and assists developers to choose the optimal license through an interactive wizard with guidelines of three aspects: personal open source style, business pattern, and community development. The usefulness of LicenseRec is confirmed by the consistent positive feedback from 10 software developers with academic and industrial backgrounds. Our tool is accessible at https://licenserec.com. And a video showcasing the tool is available at https://video.licenserec.com.**

***Index Terms*—Open source software, open source license, open source license recommendation**

I. Introduction

Open source software (OSS) is released under an open source license in which the copyright holders grant users the rights to use, modify, and redistribute the software and its source code, and specify the obligations of the users [1]. Open source license provides an effective way to protect the intellectual property rights of OSS and serves as a legal basis for cooperation in open source communities, thus promoting the long-term development of open source software. As Linus Torvalds said, the GPL is one of the defining factors in the success of Linux, because it requires giving back and avoids the risk of fragmentation [2]. As a comparison, the wrong choice of open source license may not only lead to legal problems of using open source software but also cause high financial losses to developers [3].

However, developers find it hard to choose an open source license that is compatible with the licenses of third-party dependencies integrated in their projects and suits their own preferences [4]. One of the reasons accounting for this problem is that the existence of numerous similar open source licenses and the complex legal terms (formal descriptions of the conditions of software use) involved often confuse developers [3], [5]–[7]. Specifically, there are more than 100 licenses certified by OSI[1], of which some terms are extremely difficult to understand without background knowledge, e.g., some parts of GPL-3.0[2] are almost unreadable [8]. Thus it is rather challenging to unravel the compatibility between licenses and use them properly. Another contributing factor to the license-choosing dilemma is the diversity and complexity of developers' needs, e.g. the development of the community and the potential of commercialization [9], [10]. Therefore, developers have a hard time figuring out what type of licenses to choose in accordance with their open source preferences and needs.

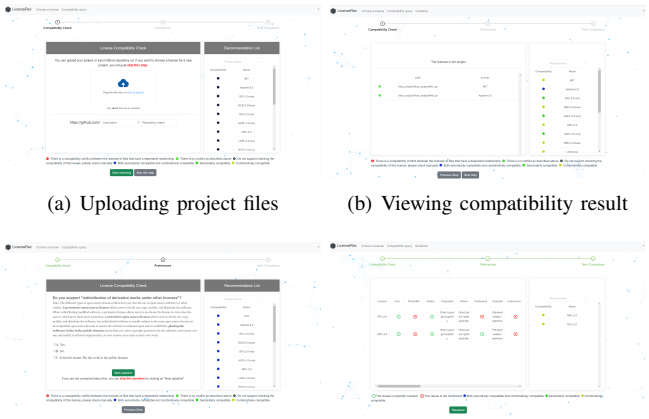
To help developers choose licenses, several tools have been proposed, such as ChooseALicense[3] and findOSSLicense [11]. However, these tools lack a comprehensive consideration of the practical difficulties and actual needs of developers as discussed above, e.g., they both ignore the compatibility with licenses used by third-party dependencies when selecting a license, which may induce financial and legal risks [12], [13].

In this paper, we present LicenseRec, a knowledge-based open source license recommender, to bridge the gap in existing tools for recommending licenses to developers. We construct a knowledge base consisting of term features and compatibility information for 63 popular licenses, along with guidelines essential for developers that are derived from real-world OSS licensing patterns. LicenseRec takes a software project's all files as input. It first detects component licenses (CLs), i.e., licenses declared in the project files and licenses of third-party dependencies. Then it performs compatibility checks to identify candidate licenses that are compatible with all CLs involved in the project. Further, LicenseRec allows users to specify their preferences through an interactive wizard and recommend corresponding candidate licenses. Eventually, LicenseRec presents users with a detailed comparison of terms among recommended licenses and their compatibility with CLs in the project to help users further understand and properly choose licenses in need.

We evaluate the usefulness of LicenseRec by inviting 10 experienced developers (7 Ph.D. students and 3 industry employees) to use it and give feedback. The average "overall rating" of 10 developers is 8.2 out of 10, indicating that

† Corresponding Author

[1]https://opensource.org/
[2]https://www.gnu.org/licenses/gpl-3.0.html
[3]https://choosealicense.com/



(a) Uploading project files (b) Viewing compatibility result

(c) Answering questions in wizard (d) Viewing recommendation result

Fig. 1. LicenseRec interfaces for developers to choose a license.

LicenseRec provides useful support for developers in choosing the right open source license.

The main contributions of this paper are as follows.

- We detect CLs involved in the project from both code files and third-party dependencies via a dependency parser covering three popular languages.
- We make a fine-grained characterization of compatibility, i.e., **secondary compatibility** and **combinative compatibility**, to help users better understand how to properly relicense derivative works with compatible licenses.
- We propose license-choosing guidelines derived from real-world OSS licensing patterns, focusing on three aspects: personal open source style, business pattern, and community development.

LicenseRec is available at https://licenserec.com. Its source code is available on GitHub[4] and the demonstration video is on YouTube[5].

II. Motivating Example based on Related Work

In this section, we explain the idea behind LicenseRec, and demonstrate how LicenseRec could help developers to choose the right license for their projects compared with related work.

Suppose a developer Alice wants to publish an IoT (Internet of Things) library on GitHub. Choosing the right license for her project is non-trivial. There are hundreds of OSI-approved open-source licenses around, and reading through all of them is not a feasible option. Plus, she doesn't know much about what the licenses used by library dependencies of her project are, and how many options are left for her to comply with these licenses. Although it is possible to locate the licenses of dependencies by inspecting the source code and check the compatibility of the licenses with online license comparators, it is a tedious and error-prone task.

Alice tried several license recommendation tools, but none of them could help her. Tldrlegal[6] summarizes the privileges and restrictions of each license into easy-to-read "can", "cannot", and "must" statements. But it doesn't answer the question of which license is the best for the project, as it doesn't elaborate on the advantages and disadvantages of each license. ChooseALicense[7] focuses on a few basic recommendation scenarios, where MIT, Apache-2.0, and GPL-3.0 are the main options provided. OSSWATCH License Diffentiator[8] compares license terms and helps developers narrow down the options through a series of questions. However, their mechanism is not flexible enough to satisfy Alice's needs with more complex requirements, e.g. earning financial benefits and obtaining technical support from the community [14]. Furthermore, all of the above tools are context-free, i.e., they don't consider the characteristics of the project and licenses of the dependencies. FindOSSLicense [11] utilizes rule-based and collaborative filtering techniques to recommend the right license based on the choices of existing repositories and other users. Alice is frustrated to find that none of these tools addresses the license compatibility issue on the dependency tree, which matters because modern OSS projects often depend on hundreds of libraries. Also, none of them mentions whether to relicense or keep the original license of the dependencies.

Here LicenseRec comes to the rescue. After Alice simply uploads the source code of her project to LicenseRec (Figure 1(a)), LicenseRec automatically scans for license files and analyzes the licenses of dependencies. Once the analysis is completed, LicenseRec provides Alice with a list of compatible licenses, as illustrated in Figure 1(b). She is surprised to find some licensing requirements that other tools did not tell her. For example, one of the dependencies uses MIT and another one uses Apache-2.0. Her initial idea was GPL because it's the choice of many well-known projects, but LicenseRec shows that only GPL-3.0 is **secondarily compatible** with both of them, which means 100% of her codebase must be licensed under GPL-3.0.

From the licensing guidelines, Alice learns that user autonomy is critical for a prosperous community, and GPL-3.0 is considered IoT-unfriendly because its "installation information" terms are unfeasible on headless IoT devices. But Alice wants to ensure that her library is used for good purposes, a permissive license is not the answer either. After filling the questions in the license wizard (Figure 1(c)), LicenseRec recommends Mozilla Public License 2.0 (MPL-2.0), a copyleft license that satisfies **combinative compatibility** with both MIT and Apache-2.0 (Figure 1(d)). Alice releases her library under MPL-2.0 and keeps the Apache dependency as it is.

III. LicenseRec Implementation

Figure 2 presents the overview of LicenseRec which mainly contains four parts: License Knowledge Base, License Scanner, Compatibility Checker, and License Wizard.

[4]https://github.com/osslab-pku/RecLicense
[5]https://video.licenserec.com
[6]https://tldrlegal.com/
[7]https://choosealicense.com/
[8]http://oss-watch.ac.uk/apps/licdiff/

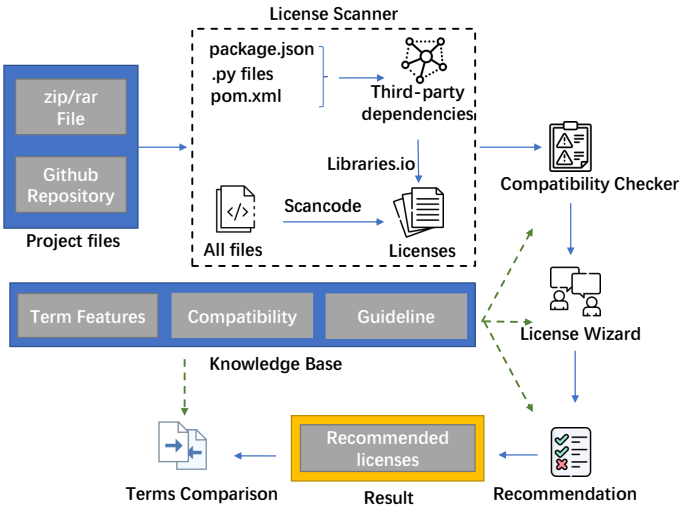


Fig. 2. Overview of LicenseRec.

A. License Knowledge Base

Considering the difficulties of understanding license terms, compatibility between licenses, and impacts of licenses on projects, we design a knowledge base[9] consisting of term features, compatibility, and guidelines.

1) Term Features: License terms refer to the use conditions specified in the license text [11]. Different licenses may conflict on specific terms, e.g. Apache-2.0 explicitly grants the user patent rights but CC0-1.0 explicitly does not. Therefore, we check the compatibility between licenses based on their terms.

We first select representative free/open source licenses that (1) are certified by FSF or OSI, (2) are not outdated versions (e.g., Apache-1.1), and (3) are not restricted to specific areas, software, and authors (e.g., WXwindows). Eventually, we collect 63 licenses. Then, we thoroughly read the 63 licenses and relevant literature [11], [13], [15], and summarize 19 terms that are likely to incur conflicts among licenses such as copyright grants, trademark limitations, patent grants, and so on.[10] Finally, we encode all 63 licenses based on the 19 terms, which we refer to as term features for subsequent compatibility checks and term comparison.

Fig. 3. Secondary compatibility (left) and combinative compatibility (right).

2) License Compatibility Matrix: Open source license compatibility refers to the feasibility of distributing derivative (i.e., modify or combine) works under a different license from the one used by modified or combined works. License compatibility is directed, i.e., the conclusion that License A is compatible with License B cannot be deduced from the fact that License B is compatible with License A.

Figure 3 illustrates two typical scenarios where compatible licenses are used for relicensing. Based on these two scenarios, we summarize two types of compatibility: **secondary compatibility** and **combinative compatibility**. If A satisfies secondary compatibility with B, all parts of the derivative works obtained from works licensed under A can be subject to the constraints of license B. For example, modified software under Apache-2.0 can be relicensed and distributed using GPL-3.0. Unlike secondary compatibility, combinative compatibility means that the part originally subject to license A still needs to be subject to A. For example, software that contains a third-party component licensed under Apache-2.0 can be relicensed and distributed using MPL-2.0 as a whole, as long as the third-party component is still under the constraint of Apache-2.0. Any two licenses need only satisfy either of the above compatibility relationships to be considered compatible.

We develop an algorithm[11] to determine whether any two of the 63 licenses satisfy two types of compatibility based on 19 identified terms and construct a compatibility matrix.

3) Guidelines for License Selection: One of the most common challenges developers encounter when choosing a license stem from the complexity of their needs and the uncertainty of the impact of the license on their project.

Users often have complex needs when they open source their projects, such as taking financial benefits, gaining a wide user base, improving their reputation in the open source community, and so on [14]. We summarize three factors from these needs, i.e., business pattern, community development, and user's personal open source style that influence users' choice of license.

In order to assist with the license selection, we obtain the advantages of various open source licenses on above three factors via extensively reviewing literature and case studies of licenses adopted by OSS maintained by commercial companies. We construct guidelines from the three factors and help users understand what type of license they should choose.

B. License Scanner

Previous work [11] ignores the compatibility between CLs, especially the licenses of dependencies, which makes the recommendation result potentially incompatible. To fill this gap, we design the module to detect CLs in the project for the subsequent compatibility check. CLs include internal licenses, i.e., licenses declared in project files, and external licenses, i.e., the licenses used by its dependencies, so we use two different methods to detect CLs.

In order to obtain internal licenses, we use Scancode[12] to scan all files in the project to detect licenses within

[9]https://github.com/osslab-pku/RecLicense/tree/master/backend/app/knowledgebase

[10]See appendix for more details: https://github.com/osslab-pku/RecLicense/tree/master/appendix

[11]See the appendix: https://github.com/osslab-pku/RecLicense/tree/master/appendix

[12]https://github.com/nexB/scancode-toolkit

files. ScanCode is a Python app using a data-driven approach with good scalability. To obtain external licenses, LicenseRec parses dependency for three languages including Python, Java, and Javascript which are the top three popular languages on GitHub[13]. For Python, since the requirement.txt file is not mandatory in Python projects and the dependencies in $requirement.txt$ may be out of date [16], LicenseRec parse dependencies by analyzing the import statements in all python files. For Java and Javascript, LicenseRec parses the $packages.json$ and $pom.xml$ to obtain the dependencies, respectively. Afterward, LicenseRec retrieves license information of parsed third-party dependencies from libraries.io[14].

C. Compatibility Checker

Compatibility Checker performs compatibility checks on the CLs identified by the License Scanner based on the License Knowledge Base. For files declaring licenses, Compatibility Checker will check whether licenses between interdependent files are compatible. For all CLs in projects, Compatibility Checker detects all incompatibilities between licenses. Finally, it identifies the licenses that meet the compatibility requirements based on the compatibility results.

D. License Wizard and Recommendation

According to the License Knowledge Base, the License Wizard asks the user seven questions to characterize her preferences, including the granting of patent rights, trademark rights, and so on. Then, LicenseRec determines the final recommended licenses that match her preferences. Finally, LicenseRec presents 19 terms of each license to help users understand their differences and make a final decision.

IV. Evaluation

We invite 10 respondents with rich open source experiences (including 7 Ph.D. students and 3 industrial developers) to use LicenseRec and provide feedback. As shown in Table I, the overall rating of LicenseRec by the 10 respondents is 8.2 out of 10. The results show that LicenseRec is a useful tool for developers to understand the license terms, evaluate the compatibility, and choose the license type, which provides strong support for developers to choose suitable open source licenses. It is worth noting that 10 respondents find LicenseRec particularly useful for understanding license compatibility, indicating that the fine-grained characterization of compatibility that we propose is sensible.

TABLE I
User Satisfaction Ratings.

Items	Mean	SD
Overall	8.2	0.87178
helpful in choosing the right open source license	8	1
helpful in understanding open source license terms	7.9	1.04403
helpful in understanding and judging compatibility	8.4	0.8
helpful in distinguishing different licenses	8.3	0.78102
helpful in choosing a proper type of license	8	1

[13]https://octoverse.github.com/2022/top-programming-languages
[14]https://libraries.io/

V. Conclusion and future work

We design a tool, LicenseRec, to assist users to select an open source license for their projects, based on the project's CLs, and users' preferences on personal open source style, business pattern, and community development. The tool is found useful in helping developers choose a license, especially the fine-grained compatibility check that allows developers to understand how to properly relicense derivative works in practice. In the future, we will explore how to use automated methods for term feature extraction, which will greatly increase the scope of the knowledge base. Generating custom licenses to support users' complex needs is also a topic worth exploring.

Acknowledgement

This work is sponsored by the National Natural Science Foundation of China 61825201 & 62142201. We thank Haiqiao Gu for his help in dependency parsing.

References

[1] A. M. S. Laurent, *Understanding open source and free software licensing: guide to navigating licensing issues in existing & new software*. " O'Reilly Media, Inc.", 2004.

[2] Linus Torvalds credits GPL with preventing Linux fragmentation. [Online]. Available: https://www.infoworld.com/article/3112778/linus-torvalds-credits-gpl-with-preventing-linux-fragmentation.html

[3] R. Duan, A. Bijlani, M. Xu, T. Kim, and W. Lee, "Identifying open-source license violation and 1-day security risk at large scale," in *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, 2017, pp. 2169–2185.

[4] D. M. German and A. E. Hassan, "License integration patterns: Addressing license mismatches in component-based development," in *2009 IEEE 31st international conference on software engineering*. IEEE, 2009, pp. 188–198.

[5] Y.-H. Lin, T.-M. Ko, T.-R. Chuang, and K.-J. Lin, "Open source licenses and the creative commons framework: License selection and comparison," *Journal of information science and engineering*, vol. 22, no. 1, pp. 1–17, 2006.

[6] R. W. Gomulkiewicz, "Open source license proliferation: Helpful diversity or hopeless confusion," *Wash. UJL & Pol'y*, vol. 30, p. 261, 2009.

[7] D. A. Almeida, G. C. Murphy, G. Wilson, and M. Hoye, "Do software developers understand open source licenses?" in *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. IEEE, 2017, pp. 1–11.

[8] A. Guadamuz, "Free and open source software," *LAW AND THE INTERNET, 3rd Ed., L. Edwards, C. Waelde, eds., Oxford: Hart*, 2009.

[9] A. Engelfriet, "Choosing an open source license," *IEEE software*, vol. 27, no. 1, pp. 48–49, 2009.

[10] J. Lindman, A. Paajanen, and M. Rossi, "Choosing an open source software license in commercial context: A managerial perspective," in *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2010, pp. 237–244.

[11] G. M. Kapitsaki and G. Charalambous, "Modeling and recommending open source licenses with findosslicense," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 919–935, 2019.

[12] L. Rosen, "Open source licensing," *Software Freedom and Intellectual Property Law*, 2005.

[13] S. Xu, Y. Gao, L. Fan, Z. Liu, Y. Liu, and H. Ji, "Lidetector: License incompatibility detection for open source software," *ACM Transactions on Software Engineering and Methodology*, 2021.

[14] J. Lerner and J. Tirole, "The scope of open source licensing," *Journal of Law, Economics, and Organization*, vol. 21, no. 1, pp. 20–56, 2005.

[15] T. Gordon, "Report on prototype decision support system for oss license compatibility issues," *Qualipso*, vol. 79, p. 80, 2010.

[16] X. Tan, K. Gao, M. Zhou, and L. Zhang, "An exploratory study of deep learning supply chain," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 86–98.