An Empirical Study on Issue Knowledge Transfer from Python to R for Machine Learning Software

Wenchin Huang[1,2], Zhenlan Ji[3], Yanhui Li[1,2,*]
1. State Key Laboratory for Novel Software Technology, Nanjing University, China
2. Department of Computer Science and Technology, Nanjing University, China
3. School of Management and Engineering, Nanjing University, China
*Corresponding author: yanhuili@nju.edu.cn

***Abstract*—*Background*: With the blowout of programming languages, developers employ different languages to solve similar problems (e.g., to implement machine learning algorithms) separately and frequently, which gives rise to knowledge transfer across different language development. Since GitHub provides an issue tracking system for developers and users to follow with issues, knowledge about how to deal with issues is a main part of available knowledge on GitHub. Such issue knowledge could be directly transferred to help developers handle new issues on current projects from similar projects in different languages.**
***Aims*: Inspired by a large amount of developed and developing machine learning software written in Python and R on GitHub, we aim to discover how much issue knowledge can be transferred from Python projects to R projects.**
***Method*: We investigate totally 1161 issues from 15 popular machine learning projects in R and 7496 issues from Scikit-Learn in Python on GitHub. After computing the text similarity between issues from R and Python projects, we match top 5 similar Scikit-Learn issues for each R issue and manually judge 1161×5 issue-pairs to label and group them.**
***Results*: We observe that a) 13% (149/1161) of R issues can refer to related Python issues; b) 47% (71/149) of related R issues can be linked to Python issues by the text mining technique BM25 at the very early stage; c) 83% (124/149) of related Python issues support code and description about the similar machines learning problems; d) reference knowledge is considered as the most useful knowledge from Python issues.**
***Conclusion*: We put forward the following suggestions: a) referring to the corresponding cross languages issues is an efficient way for developers, especially there is the lack of related information in current language; b) the text mining technique BM25 is helpful for developers to start earlier for searching similar issues cross languages.**

I. Introduction

With the blowout of programming languages and the widespread use of GitHub, software development has evolved from a single language development to socio-technical ecosystems, within which developers from different language communities solve similar problems separately and frequently [1]. Knowledge gained across different language development assists developers to find out how different languages implement the same requirements, which greatly contributes to knowledge acquisition about current requirements, and subsequently to decisions about design, coding, test and maintenance [2].

DOI reference number: 10.18293/SEKE2020-102

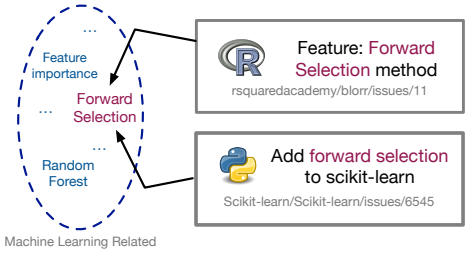

Fig. 1. Two issues with similar topics from R and Python projects

Machine learning software is a category of libraries to implement machine learning algorithms that allows users to accurately predict outcomes without explicit programming (e.g., Scikit-Learn[1]), which consists of algorithm implementations with *similar topics* and *different languages*: (a) machine learning algorithms (e.g., Random Forest) have clear specifications of the functionality regardless of language implementation [3]; (b) for most of machine learning algorithms, there are existing libraries written in popular languages (e.g., Python) with reliable performance for predictive modeling [4]; (c) developers from a different language community (e.g., R) may face the same or very similar requirements, and consequently develop libraries to implement the same or very similar functionality of algorithms in new languages [5]. During the new language implementation of algorithms, knowledge transferred (e.g., from Python to R) would be useful to accelerate current software development.

Issue is a critical way for software projects to track the progress of problems reported by developers and users during developing, maintaining, or using, which could be a bug report, a code document, a feature enhancement request, a task and so on [6], [7]. Since GitHub provides an issue tracking system for developers and users to handle issues, knowledge about how to deal with issues is a main part of available knowledge extracted from current projects on GitHub, which can be transferred into the development of the similar projects in different languages. Here we illustrate an example for a pair of issues sharing the similar topics from the Python project and the R project in Figure 1. From the topics of the issue

[1]Scikit-Learn is one of the most famous machine learning software in Python. http://scikit-learn.github.io.

#6545 in Scikit-learn/Scikit-learn and the other issue #11 in rsquaredacademy/blorr, we can observe that they both aim to implement the specific algorithm of "forward selection". Obviously, when dealing with the new issue in R, developers can gain knowledge from the old issue in Python.

Inspired by a large amount of developed and developing machine learning software written in Python and R on GitHub, we aim to discover how much issue knowledge can be transferred from Python to R. In this paper, we select 15 popular R machine learning repositories and Scikit-Learn in Python to extract their issue contents. By computing the text similarity between 1161 R issues and 7496 Python issues, we manually browse and label the issue-pairs (i.e., the older issue in Python and the new one in R) to check similarity with a top-5 similarity list. Finally, we extract 149 related issue-pairs for further research.

To examine how knowledge can be transferred from Python issues to R issues, we structure our study by addressing the following three research questions (RQs):

RQ1 (Linking issues from R to Python): how can developers link to the related Python issues when R issues are just created? By experimenting three strategies to search related Python issues at the very early stages of R issues, we observe that BM25 search performs the best, which successfully searches over 47% corresponding Python issues for the R issues by only using the initial information (e.g., title and body) from issues.

RQ2 (Types of Knowledge): what kinds of knowledge can developers learn from related Python issues? We observe that related Python issues offer 6 types of knowledge for those R issues: description, reference, outer link, related issue, code and code document. Among them, the most popular kinds of knowledge are description and code.

RQ3 (Helpfulness of Knowledge): which type of knowledge from Python issues is most helpful? Based on the comparison of spearman correspondence and weighted mean helpfulness, we observe that reference knowledge type turns out to be the most helpful knowledge.

Our study makes the following contributions.

- Dimension. This study opens a new dimension in knowledge transfer from Python to R in cross-language software development.
- Study. This study includes an empirical study of cross language knowledge transfer on 1161 issues from 15 repositories in R and 7496 issues from Scikit-Learn in Python.
- Strategy. This paper puts forward BM25 search as an efficient strategy to search for corresponding issues at the early stage for cross language knowledge transfer.

The rest of this paper is organized as follows. Section II describes our research methodology. Section III, Section IV, and Section V present the results of three RQs above. Threats to validity is discussed in Section VI. Finally, the conclusion and future work is put forward in Section VII.

II. Our Approach

In this study, we collect issues from 15 R repositories as the newer language repositories and the famous Python repository Scikit-Learn as the older language repository. Our approach comprises three steps: first we collect the issues from these repositories on the GitHub; after that, we compute the similarity between issue-pairs by using the word embedding technique; finally, we extract the related issue-pairs by manually judging their relationship.

A. Issues Collecting

We choose one of the most representative machine learning packages Scikit-Learn as our studied repository in Python. Besides, we take 15 open sources R packages into consideration, which are all involved in machine learning and highly recommended. We choose these R repositories according to the following four principles:

- they implement widely used algorithms, such as Random Forest, Naive Bayes and K-Nearest Neighbors.
- they are open source repositories on GitHub.
- they are listed on CRAN[2].
- the time of first commit in these repositories should be later than the time of first commit in Scikit-Learn.

Table I illustrates the detail of 15 R repositories we study. The first column shows the names of R repositories. The URL links to the repository are listed in the second column. The numbers of issues and the first commit time are listed in the third and fourth columns, while the algorithms implemented are listed at the last. Though there are some repositories with little stars or forks, even little issues and pulls, we still choose them for the following two reasons: (a) to ensure the variousness of algorithms, (b) to observe issue knowledge transfer for these repositories from the very beginning.

To sum up, we collect 1161 issues from the R repositories, including both open and closed issues, and 7496 issues from Scikit-Learn.

B. Similarity Computing

After choosing the repositories and collecting the issues, we compute the text similarities between 1161 issues from R and 7496 issues from Scikit-Learn by employing the word embedding dataset offered and pre-trained by Google [8]. It is worthwhile pointing out that, we use the **whole issue content**, including topic, question and discussion in each issue for similarity computing. We only consider the natural language text, and exclude the other parts, e.g., code. For issue pairs $\langle \mathcal{I}_P, \mathcal{I}_R \rangle$ from **P**ython and **R** correspondingly, we consider $\mathcal{I}_P$ and $\mathcal{I}_R$ as the sets of words appearing in them, and calculate the similarity of them as follows.

(a) Given two words w_P and w_R appearing in $\mathcal{I}_P$ and $\mathcal{I}_R$, their semantic similarity is defined as the cosine similarity by using their word embeddings:

$$sim\,(w_P, w_R) = \frac{w_P^T w_R}{\|w_P\|\,\|w_R\|}$$

[2]CRAN is the most popular online repository that store up-to-date versions of R packages, including code and documentation. https://cran.r-project.org

TABLE I
AN OVERVIEW OF 15 STUDIED R REPOSITORIES

R repository	URL	#Issues	First Commit	#Watch/#Star/#Fork	Classifier(Algorithms)
Arborist	https://github.com/suiji/Arborist	42	31 Jan 2013	15/68/12	RF
benchm-ml	https://github.com/szilard/benchm-ml	56	28 Mar 2015	155/1734/326	LR/SVM/RF/boosting/...
bigrf	https://github.com/aloysius-lim/bigrf	20	15 Feb 2013	11/91/26	RF
blorr	https://github.com/rsquaredacademy/blorr	71	15 May 2017	2/9/1	LR
classyfire	https://github.com/eaHat/classyfire	17	11 Jul 2014	3/8/0	SVM
edarf	https://github.com/zmjones/edarf	57	4 Sep 2014	12/61/10	RF
forestFloor	https://github.com/sorhawell/forestFloor	33	5 Jul 2015	5/36/7	RF
ggRandomForests	https://github.com/ehrlinger/ggRandomForests	32	4 Jan 2013	8/107/23	RF
grf	https://github.com/grf-labs/grf	332	28 Jul 2014	41/359/99	RF
kknn	https://github.com/KlausVigo/kknn	17	20 Apr 2015	3/15/5	KNN
lumberjack	https://github.com/neurodata/lumberjack	88	15 Feb 2017	9/54/35	RF
naivebayes	https://github.com/majkamichal/naivebayes	5	3 Jun 2017	2/14/4	NB
randomForestSRC	https://github.com/kogalur/randomForestSRC	22	18 Nov 2016	8/48/8	RF
ranger	https://github.com/imbs-hl/ranger	366	28 Jul 2014	42/507/114	RF
TFG	https://github.com/Dani-Basta/TFG	3	31 Oct 2017	5/3/0	KNN
TOTAL		1161			

which is calculated by the Euclidean norm of their vectors using inner product.

(b) In order to compute the similarity between the issues, we introduce the similarity calculation approach proposed by Ye et al. [9], which modified the text-to-text similarity [10]. To calculate the similarity between a word w and the whole context of the issue $\mathcal{I}$, we compute the maximum similarity between w and w' in $\mathcal{I}$:

$$sim\,(w, \mathcal{I}) = \max_{w' \in \mathcal{I}} \{sim\,(w, w')\}$$

(c) Both the words with no word embedding and the words not appearing in the target issues $\mathcal{I}^*$ are ignored in the following calculation. The asymmetric similarity from $\mathcal{I}$ to $\mathcal{I}^*$ can be computed as:

$$sim\,(\mathcal{I} \to \mathcal{I}^*) = \frac{\sum_{w \in P(\mathcal{I} \to \mathcal{I}^*)} sim\,(w, \mathcal{I}^*)}{|P\,(\mathcal{I} \to \mathcal{I}^*)|}$$

where $P(\mathcal{I} \to \mathcal{I}^*) = \{w \in \mathcal{I} | sim(w, \mathcal{I}^*) \neq 0\}$.

(d) The final symmetric similarity $sim(\mathcal{I}_P, \mathcal{I}_R)$ between two issues $\mathcal{I}_P$ and $\mathcal{I}_R$ can be computed as the sum of two asymmetric similarity.

$$sim\,(\mathcal{I}_P, \mathcal{I}_R) = sim\,(\mathcal{I}_P \to \mathcal{I}_R) + sim\,(\mathcal{I}_R \to \mathcal{I}_P)$$

For each R issue $\mathcal{I}_R$, we rank the issue pairs $\langle \mathcal{I}_P, \mathcal{I}_R \rangle$ from large similarity values to small ones in order to find the most related Python issue from Scikit-Learn. We remain the top 5 of the most similar issue pairs (totally 1161×5 issue-pairs), which will be filtered to pick out the real related pairs by the following manual check.

C. Manual Judgement

1) Preprocessing Candidate Issue-Pairs: Among these 1161×5 issue-pairs, some of the issues describe the issue confusedly, some of them are simply notes for recording the updating of repositories, and some of them are just discussing the details in the code (e.g., XX lines in Y documents). These issues are not included in our following consideration.

Before manual judgement, we divide the issue context into two parts, the question part and the discussion part for the following description. The **question part** contains the topic and the body of the issues submitted by issue reporter on the top of the issue content. The **discussion part** is the content followed by the question part. Generally, we classify an issue by browsing the whole issue context, including both question part and discussion part.

2) Checking Related Issues: Our manual judgement is conducted by three members of our research group. Each of them scans the issue-pairs independently, and labels the similarity as "related" or "unrelated". Once we get different results on classification, we will make a double check to ensure whether the issue-pairs are related or not. If the results of the double check are still different, the final label is determined by voting. For example, if two participants vote "related" and one votes "unrelated" for an issue-pair, we judge the issue-pair as a related issue-pair. After manual judgement, there are 149 issue-pairs labeled as related. Notice that we exclude the issue-pairs, in which the creation time of Python issues are later than the creation time of R issues. To ensure the reliability of human labeling, we also calculate the inter-rater (i.e., Cohen's Kappa [11]) for manual labeling, by comparing the final result after voting and the independent result from each member. The results show that for each member, the value of Cohen's Kappa is larger than 0.6, which means the related/unrelated results are substantial [12].

3) Evaluating Related levels: Besides, we label the related levels of these 149 issue pairs. In detail, we group the knowledge transferred between these 149 issue pairs into three levels manually.

- **Direction-related level (88 issue-pairs).** the corresponding Python issues offer knowledge related to the R issues in the same or very similar *directions*.
- **Problem-related level (32 issue-pairs).** In this level, the two issues are probably discussing on the very similar *problems*, however, Python issues have not given direct solutions to R issues.
- **Solution-related level (28 issue-pairs).** The Python issues offer direct *solutions* (e.g., pseudo-code and reference), which are most helpful knowledge to the newer R

TABLE II
THE SUCCESSFULLY SEARCHING RESULT FROM 4 DIFFERENT METHOD IN THE TOP 10 LIST.

Searching Strategies	Successfully Searched Rate
Topic Search	5.37% (8/149)
TF-IDF	20.13% (30/149)
BM25	47.65% (71/149)

issue. This kind of knowledge offered by Python issues can almost be used in dealing with R issues directly.

III. LINKING ISSUES FROM R TO PYTHON

"How can developers link to the related Python issues when facing R issues" may become a critical question of knowledge transfer from Python to R. No matter in which situations, developers desire to gain useful information to solve issues as soon as possible. In this RQ, we focus on the **question part** (see Section II-C1) of the 149 R issues, which contains the topic and the body of issues submitted by issue reporters. The question part of R issues represents the early information we can get at the report time of R issues. Based on the question part of R issues, we employ different strategies to search for their related Python issues, by their topics and two popular text mining methods TF-IDF and BM25[3].

Topic Search. Because of the limitation of searching APIs available on GitHub, we can hardly search the corresponding issues by using all the content of question parts. We select the topics from the issues, which represent the main ideas of issue reporters. We search for the corresponding issues in Scikit-Learn repository by using searching API from GitHub and check the top 10 results on the return list.

TF-IDF search. From two strategies mentioned above, we observe that the available searching tool from GitHub can hardly fulfill our needs. Therefore, we introduce a classic strategy in natural language process, TF-IDF [13]. We abstract word embeddings by TF-IDF with the question part of the R issues and use these word embeddings vectors to match 10 most similar corresponding issues. Different with the text similarity search we used before, we set up word embeddings by TF-IDF instead of using the open source data [8]. In detail, we implement the TF-IDF method by Scikit-learn[4].

BM25 search. Though the strategy above improve a lot, we still consider finding a more suitable way for raising successfully searched rate. BM25 [14] is usually used in evaluating the relationship between query and documents. This strategy mainly computes the similarity by 3 parts, which are the weight of words, similarity between words and documents, similarity between words and queries. We employ BM25 to compute the similarity between question parts from R issues and Python issues. In detail, we reuse an implementation of BM25 algorithm in the Python library Gensim[5] with the default parameter settings.

For each search strategy, we check the top 10 results on the return list. Once the related Python issue from the issue-pair occurs on the top 10 list, we count it as a "successful search" and record the rank. Table II represents the results of the above three search strategies. We can find that in Table II, BM25 performs much better than the other strategies, which can successfully detect almost half of the issue-pairs.

> Answer to RQ1: by conducting three strategies to search related issues at the very early stage of R issues, we observe that BM25 search performs the best, which successfully searches over 47% related Python issues for the 149 R issues by only using the initial information (e.g., title and question body).

IV. TYPES OF KNOWLEDGE

In the RQ above, we start from R issues of the related issue-pairs, and find out how to link to Python issues. In the following two RQs, we will focus on Python issues of the related issue-pairs. Specifically, we are going to discover what type of knowledge we can get from the Python issues in this RQ. In order to make a summary of knowledge in Python related issues, we classify the knowledge into 6 groups, which are **description**, **reference**, **outer link**, **related issue**, **code** and **code document**, as illustrated in Table III with numbers of issues containing such kind of knowledge, brief introduction and typical examples.

We choose an interesting example containing all 6 knowledge types, as shown in Figure 2, which mainly discusses about the implementation of Balanced Random Forest. Next, we will present the description of these 6 kinds of knowledge with help of the example.

Description type (147 Python issues): Almost all of the issues are considered to have description, which offer information in natural language directly. For example in Figure 2, the orange box is labeled as the description part. However, not all issue with natural language are included, we exclude two issues which only contain "thanks" in the content.

Reference type (28 Python issues): This type is the most recognizable in these 6 groups. It offers the research papers, related tutorial, etc. As shown in the green box of Figure 2, the Python issue offers a research paper's link, which mainly talks about using random forest to learn imbalanced data. Though this kind of knowledge may sometimes be complex, it is worth reading for developers, which usually offers the original idea of an algorithm.

Outer link type (97 Python issues): Python issues contain various kinds of url links, which offer related packages or datasets, or connect to different repositories on GitHub and other open source platform. The purple box in Figure 2 gives an example of outer link, which offers dataset.

Related issue type (74 Python issues): Referring to another issue is also quite usual in issues. Those links with different issue numbers in current repository or other repositories are all

[3]Our dataset is constructed via cosine distance and the word embedding set offered and pre-trained by Google (see Section II-B). To avoid the bias, we introduce two different text mining methods TF-IDF and BM25 here.

[4]https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

[5]http://pydoc.net/gensim/3.2.0/gensim.summarization.bm25/

TABLE III
TYPE OF KNOWLEDGE IN 149 RELATED PYTHON ISSUES

Group	#Issues	Brief introduction	Example Issue
Description	147	Natural language description	https://github.com/scikit-learn/scikit-learn/issues/1454
Reference	28	Research paper etc.	https://github.com/scikit-learn/scikit-learn/issues/6545
Outer Link	97	A link to outer website e.g Wikipedia	https://github.com/scikit-learn/scikit-learn/issues/448
Related Issue	74	Another linked issue	https://github.com/scikit-learn/scikit-learn/issues/6473
Code	124	More than 5 lines of code/ Pulls	https://github.com/scikit-learn/scikit-learn/issues/2089
Code Document	58	Tutorial of code	https://github.com/scikit-learn/scikit-learn/issues/3735

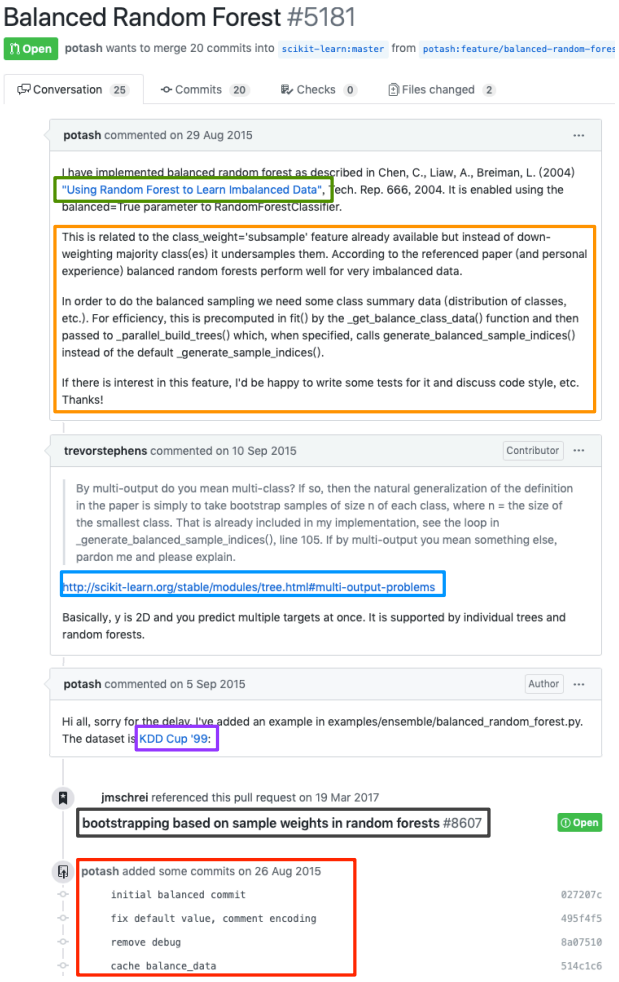

Fig. 2. An example of issues with multiple kinds of knowledge from Scikit Learn Project

involved in this type of knowledge. The black box in Figure 2 is an example of related issue knowledge.

These 4 types of knowledge described above can be learned language-independently, which are more general. The following two kinds of knowledge may need Python skill to understand and apply.

Code type (124 Python issues): Once a code part with more than 3 lines occurs in the issue, we count it as an instance of code knowledge. As illustrated in Figure 2, the red box is a typical example of code part. Due to the different grammar of different languages, this type of knowledge may be more useful for those developers who can skillfully use Python.

Code document type (58 Python issues): Besides code type, code document type will sometimes occur in the issues. As shown in Figure 2, the blue box is an example of code document.

Answer to RQ2: we observe that related Python issue can offer 6 types of knowledge for those R issues: description, reference, outer link, related issue, code and code document. Among them, the most popular kinds of knowledge are description and code.

V. HELPFULNESS OF KNOWLEDGE

In the above RQ, we discuss various types of knowledge from Python issues. "Which type of knowledge from Python issues is most helpful?" is the question we need to solve next. Based on the related levels of the issue-pairs (see Section II-C3), we mark the helpfulness $H(\mathcal{I}_P)$ of corresponding Python issues $\mathcal{I}_P$ in the range from 1 to 3:

$$H(\mathcal{I}_P) = \begin{cases} 1, & \text{if } \langle \mathcal{I}_P, \mathcal{I}_R \rangle \text{ is labeled as direction-related;} \\ 2, & \text{if } \langle \mathcal{I}_P, \mathcal{I}_R \rangle \text{ is labeled as problem-related;} \\ 3, & \text{if } \langle \mathcal{I}_P, \mathcal{I}_R \rangle \text{ is labeled as solution-related.} \end{cases}$$

and use the helpfulness of Python issues to evaluate the most useful knowledge type.

Spearman correspondence: First, we conduct a Spearman corresponding test [15] between whether containing the knowledge type and the helpfulness of Python issues, to select the most useful knowledge type. The larger Spearman correspondence shows the knowledge type with higher helpfulness. As illustrated in Table IV, we can find that the reference type gets the highest Spearman correspondence among 6 types of knowledge. Besides, related issue type gets the second place.

Weighted mean helpfulness: Besides, we judge the helpfulness of knowledge type by the following formulas. Let T ={description, reference, outer link, related issue, code, code document}. For $t \in T$, $S(\mathcal{I}, t)$ implies the set of Python issues containing the knowledge in type t. $H(\mathcal{I}_P)$ implies the helpfulness ($H(\mathcal{I}_P) \in \{1, 2, 3\}$) of the Python issue $\mathcal{I}_P$. The weighted mean helpfulness ($H_w(t)$) of type t are computed as follows.

$$H_w(t) = \frac{\sum_{\mathcal{I}_P \in S(\mathcal{I},t)} H(\mathcal{I}_P)}{|S(\mathcal{I}, t)|}$$

Finally, we rank the value of $H_w(t)$ for each knowledge type in Table IV. We can find that reference type gets the first place again. Also, the related issue type still follows.

TABLE IV
KNOWLEDGE TYPES RANKED BY SPEARMAN CORRESPONDENCE AND WEIGHTED MEAN HELPFULNESS

Knowledge Type	Spearman	$Rank_{sp}$	$H_w(t)$	$Rank_{sc}$
Reference	0.16	1	1.89	1
Related issue	0.11	2	1.70	2
Description	0.09	3	1.60	5
Code document	0.04	4	1.61	4
Outer link	0.01	5	1.67	3
Code	-0.02	6	1.58	6

According to the above results, we conclude that the reference part is the most important part in the related Python issues of the cross language knowledge transferring. Once we search for the cross language issues, scanning for the reference type of knowledge is highly recommended.

Answer to RQ3: based on the comparison of spearman correspondence and weighted mean helpfulness, we observe that reference knowledge type turns out to be the most useful knowledge.

VI. THREATS TO VALIDITY

We select 15 of the open source R repositories from GitHub which all come from machine learning classifiers category. They do not cover the issues in all kinds of R repositories, and new issues submitted after November 2018 are not included. Nonetheless, these 15 repositories are popular and well known which cover several classifiers. Furthermore, it is recommended that more projects with more issues in R and Python should be tested using our approach, and the result may vary.

For each issue-pair, we browse their content, try to understand their ideas, carefully judge the similarity levels and issue groups. We then gather all the result from 3 members and do the double-check job to ensure their correctness. Thus, we believe that all the issue-pairs we extracted are true positive. However, we compute the similarities by only using natural language text, and excluded the code, chart, etc., which may lead to lose some issue-pairs that might also be helpful for cross-language referring.

Text similarity search uses the similar text mining method with dataset extraction. Though we employ two different text mining methods while doing the searching job in RQ1, there might be some coincidence that some issues have no discussion part, which may cause little higher of the successfully searched rate.

VII. CONCLUSIONS AND FUTURE WORKS

Different languages are used to solve similar problems, which gives rise to knowledge transfer across different languages development. In this paper, we discover knowledge transfer between 15 machine learning R repositories and Scikit-Learn by analyzing their issues on GitHub.

We extract 149 related issue-pairs from 1161 R issues and 7496 Scikit-Learn issues manually. We experiment and observe that text similarity search gains high successfully searched rate and the perfect performance on ranking on searching corresponding issues for just created issues. Then, we abstract 6 types of knowledge from cross language issue, which are presented in 149 issue-pairs. Finally, in order to calculate the helpfulness, we rank the knowledge type by two indicators.

In the future, we will extend the study by enlarging the datasets from more different languages to conduct a more complete investigation. At the same time, we will also improve our similarity approach by recording more information (like code), which is considered to be more helpful.

ACKNOWLEDGEMENTS

The work is supported by National Key R&D Program of China (Grant No. 2018YFB1003901) and the National Natural Science Foundation of China (Grant No. 61872177 and 61772259).

REFERENCES

[1] Y. Zhang, D. Lo, P. S. Kochhar, X. Xia, Q. Li, and J. Sun, "Detecting similar repositories on github," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Feb 2017, pp. 13–23.

[2] C. McMillan, M. Grechanik, and D. Poshyvanyk, "Detecting similar software applications," in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 364–374. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337267

[3] S. Athey, J. Tibshirani, S. Wager *et al.*, "Generalized random forests," *The Annals of Statistics*, vol. 47, no. 2, pp. 1148–1178, 2019.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[5] M. N. Wright and A. Ziegler, "ranger: A fast implementation of random forests for high dimensional data in c++ and r," *Journal of Statistical Software*, vol. 077, no. 1, 2015.

[6] T. F. Bissyande, D. Lo, L. Jiang, L. Reveillere, J. Klein, and Y. L. Traon, "Got issues? who cares about it? a large scale investigation of issue trackers from github," in *IEEE International Symposium on Software Reliability Engineering*, 2013.

[7] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining github," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, 2016.

[8] Google, "A pre-trained dataset from google news, google-news-vectors-negative300," https://code.google.com/archive/p/word2vec.

[9] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," in *Proceedings of the 38th international conference on software engineering*. ACM, 2016, pp. 404–415.

[10] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," *Unt Scholarly Works*, vol. 1, pp. 775–780, 2006.

[11] L. M. Hsu and R. Field, "Interrater agreement measures: Comments on kappan, cohen's kappa, scott's π, and aickin's α," *Understanding Statistics*, vol. 2, no. 3, pp. 205–219, 2003.

[12] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[13] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, "Combining word embedding with information retrieval to recommend similar bug reports," in *IEEE International Symposium on Software Reliability Engineering*, 2016.

[14] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in *IEEE/ACM International Conference on Automated Software Engineering*, 2011.

[15] D. J. Best and D. E. Roberts, "Algorithm as 89: The upper tail probabilities of spearman's rho," *Journal of the Royal Statistical Society*, vol. 24, no. 3, pp. 377–379, 1975.