Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 16(4): 228-234, Dec. 2018 Regular paper

Semantic Similarity-Based Contributable Task Identification for New Participating Developers

Jungil Kim[1], **Geunho Choi**[2], **and Eunjoo Lee**[2*], ***Member*, *KIICE***

[1]Department of Software Technology Laboratory, Kyungpook National University, Daegu 41566, Korea
[2]School of Computer Science and Engineering, Kyungpook National University, Daegu 41566, Korea

Abstract

In software development, the quality of a product often depends on whether its developers can rapidly find and contribute to the proper tasks. Currently, the word data of projects to which newcomers have previously contributed are mainly utilized to find appropriate source files in an ongoing project. However, because of the vocabulary gap between software projects, the accuracy of source file identification based on information retrieval is not guaranteed. In this paper, we propose a novel source file identification method to reduce the vocabulary gap between software projects. The proposed method employs DBPedia Spotlight to identify proper source files based on semantic similarity between source files of software projects. In an experiment based on the Spring Framework project, we evaluate the accuracy of the proposed method in the identification of contributable source files. The experimental results show that the proposed approach can achieve better accuracy than the existing method based on comparison of word vocabularies.

Index Terms: Information retrieval, Source file identification, Task recommendation

I. INTRODUCTION

In software development, the quality of a product often depends on whether its developers can rapidly find and contribute to the proper tasks. In particular, rapid influx and contributions of newly participating contributors activates project development and provides new ideas in open-source projects [1]. Developers generally need significant time to become familiar with a software project. For example, a newly participating developer on a software project searches for contributable source files after reading the reference documents and understanding the structures of the software project. However, it is a time-consuming task to manually navigate all source files contained in a large-scale software project [2-8]. To reduce the burden of this task, newly participating developers can rely on experience obtained on previous software projects to which they contributed.

A source file consists of elements such as comments, methods, variables, classes, and identifiers, each of which is represented by words [9]. The vocabulary of words in a source file is usually determined according to the functionality of the source file. Therefore, developers can identify the contributable source files by searching for the words contained in them. However, even the source files that implement the same functionality may be different in vocabulary, because of authors' different coding habits or the different coding conventions of the software projects. The vocabulary difference tends to be influenced by the software projects. When words belong to separate projects, the difference between them is large [10, 11]. This makes it difficult to associate words contained in different software projects. On this basis, the word set in previous projects is unsuitable for

Received 29 June 2018, **Revised** 16 October 2018, **Accepted** 16 October 2018
***Corresponding Author** Eunjoo Lee (E-mail: ejlee@knu.ac.kr, +82-53-950-7548)
School of Computer Science and Engineering, Kyungpook National University, 80, Daehak-ro, Buk-gu, Daegu 41566, Korea.

Open Access **https://doi.org/10.6109/jicce.2018.16.4.228** print ISSN: 2234-8255 online ISSN: 2234-8883

use in finding the proper source files in the ongoing project.

In this paper, we present an approach to reduce the vocabulary difference by identifying the proper source files based on words in external software projects. Spotlight is a tool that links text to real-world concepts based on the DBPedia knowledge base, which can be used to combine different vocabulary words that refer to the same concept meaning [12]. Spotlight discovers linkable words and links the discovered words to the concept entities of DBPedia. We employ Spotlight to convert the words of a source file into semantic words. A semantic word is defined as a word referring to a concept entity in DBPedia. The words referring to the same concept entity are converted into one semantic word in a source file. Thus, a source file is represented as a semantic word collection. We propose a novel identification approach for source files to reduce the vocabulary gap between software projects. The proposed approach forms the semantic word collections of a participating project and external projects into semantic word vectors and a developer contribution vector, respectively, using DBPedia Spotlight. Contributable source files are then determined by computing the similarity between the developer contribution vector and the semantic word vectors using cosine similarity. To investigate the applicability of the proposed approach to identifying contributable source files, we perform an experiment on an open-source project. In the experiment, we compare the proposed approach with the existing method for identifying contributable source files in a project of interest. The experimental result shows that the proposed approach can more effectively identify the actually contributed source files than the existing method. With Top-25 and Top-250 identification, the proposed approach can obtain results improved by 23% and 32%, respectively, compared with the existing method. Therefore, we believe that the proposed approach can support newly participating developers in identifying contributable source files based on their experience in external software project development.

The remainder of this paper is structured as follows. The works related to this study are introduced in Section II. The detailed proposed approach is described in Section III. Experiment results performed on an open-source project are reported in Section IV. Section V concludes this study.

II. RELATED WORKS

In software project development, attraction and settlement of developers increase the flexibility of project development [1]. In particular, because entry and exit of developers often occur in open-source project development, quick settlement of developers is crucial. Mentor recommendation can be a solution for the problem of developer settlement. The major role of a mentor is to help newcomers resolve their works and familiarize the software development process. Newcomers can readily complete their work with the support of mentors [13, 14]. However, because there is usually high influx of newcomers in an open-source project, mentor recommendation is rarely available in specific cases. On the other hand, task recommendation provides another solution to the problem of developer settlement. Newcomers can employ task recommendation to find suitable tasks and increase their contribution rate.

Table 1. Preprocessing results

Preprocessing method	Word	Preprocessed words
CamelCase split	getHttpHeader	get, Http, Header
Lowercase transform	get, Http, Header	get, http, header
Stopword removal	http, header	http, header

Cubranic and Murphy [15] and Malheiros et al. [16] proposed task recommendation methods for the support of new participating developers on a software project. The methods are based on text similarity between presented query keywords and the description text of resolved previous tasks. However, these methods require words to be closely related to the participating project to ensure reliability of recommendation, and they cannot overcome the vocabulary gap when newcomers want to find similar tasks to the works they contributed to other software projects.

In information retrieval, semantic comparison between documents is presented to alleviate the limitation of lexical comparison. This makes document retrieval with the lexical comparison of words ambiguous. Spotlight [17] is a tool to link given words to DBPedia concepts. A DBPedia concept is an online document page describing real-world entities. Spotlight connects words indicating the same entity to an entity concept. Table 1 shows an example of linking words to DBPedia concept entities. The words *synchronizing*, *sync*, *synchronize*, and *synchrony* are all connected to an entity concept of *synchronization*. In this paper, we use Spotlight to alleviate the vocabulary problem in identifying particular source files on a software project, which previous studies have not examined.

III. APPROACH

The overall work flow of our proposed approach is shown in Fig. 1. First, the contributed source files in external projects and the source files of a participating project are passed to Spotlight as inputs. All words contained in the given source files are annotated with semantic words through Spotlight. Semantic word vectors are then generated from the sematic word collections of the source files in the participating project, and a developer contribution vector is generated from the semantic word collections of the contributed source

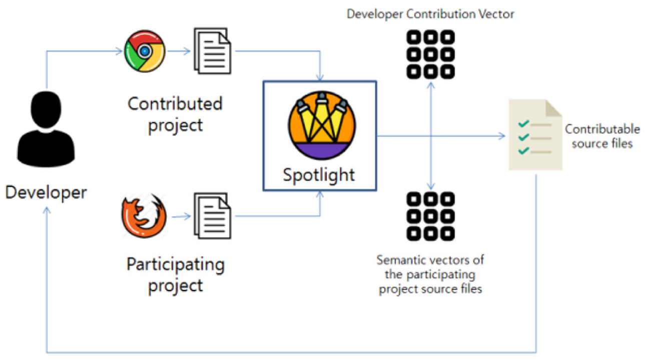
Developer Contribution Vector
Contributed project
Developer
Spotlight
Contributable source files
Participating project
Semantic vectors of the participating project source files

Fig. 1. The work flow of the proposed approach.

files. Finally, Top-N similar source files are determined by comparing the developer contribution vector with the semantic word vectors. In the following subsections, the works are detailed.

A. Preprocessing

Before annotating words of source files with Spotlight, meaningless words must be filtered. For example, English stop words frequently appear in source code but represent no real-world entities. In addition, words comprising two or more constituent words must be split. In this paper, we preprocess the words of a source file with several natural language process techniques. First, a word comprising two or more words is split by using CamelCase splitting. For example, the compound word *wordForm* is split into separate words *word* and *Form* by CamelCase splitting. Uppercases are then transformed into lowercases with lowercase transform. The word *Form* is transformed into *form*. Stopwords are then removed using stopword removal. Stopword removal is used to remove programming language keywords and English stopwords. Words of less than two characters are also excluded because they are often interpreted as meaningless or incorrect abbreviations. Table 1 shows each preprocessing result of the word '*getHttpHeader*'.

B. Annotation

Spotlight is based on the DBPedia ontology comprising more than 5 million concept entities [17]. Given a set of words, Spotlight annotates them through the tasks of named entity recognition and information extraction. An annotation associates a word with a DBPedia concept. Although some words are different in vocabulary, they are given the same annotation if they are associated with the same DBPedia concept.

After preprocessing, we employ Spotlight to reduce the vocabulary gap of words between source files. Spotlight attaches annotations to the words of source files that have been preprocessed. Table 2 shows an example of annotation

Table 2. Example of annotation

Words	Annotation result
synchronizing, sync, synced, synchronize, synchrony, syncs, syncing	Synchronization
parsing, decoding, parsers, parse, decode, parses, parser, parsable	Parsing

results with various similar words in the vocabulary. We term annotations obtained from the annotation task as *semantic words* and formally express the annotations of the words in a source file as a semantic word collection (s_f) as follows.

$$s_f = < t_1, t_2, \dots, t_n > \tag{1}$$

where f is a source file, t_i is a semantic word, and n is the total number of semantic words in f. We produce two sets of semantic word collections, S_c and S_p, from the contributed source files and the source files in the participating project, respectively.

C. Generating Semantic Word Vectors

In the field of information retrieval, term frequency-inverse document frequency (TF-IDF) is used to determine the importance of words [18]. TF-IDF determines the importance of words according to the frequencies of occurrence in a text document and a collection of text documents. High weight is assigned to terms that frequently appear in a text document and infrequently appear in the collection of text documents. On the other hand, low weight is assigned to terms that generally appear in various text documents.

In this paper, we use TF-IDF to determine the importance of the semantic words in S_p. The frequency of a semantic word is determined by counting the occurrences in a semantic word collection. The inverse document frequency of a semantic word is determined by the inverse fraction of the semantic word collections that contain the semantic word. The term frequency and inverse document frequency of a semantic word in a semantic word collection ($TF(s_f, t_i)$ and $IDF(t_i)$) are computed as follows, respectively.

$$TF(s_f, t_i) = log(1 + f_{t_i,s_f}) \tag{2}$$

$$IDF(t_i) = log(1 + |N|/f_{t_i,N}) \tag{3}$$

where N is the total number of semantic word collections, f_{t_i,s_f} is the frequency of a semantic word t_i in a semantic word collection s_f, and $f_{t_i,N}$ is the frequency of a semantic word t_i in all semantic word collections. $IDF(t_i)$ is at least 1. Based on the above equation, the weight of a semantic word t_i (w_{s_f,t_i}) is calculated by multiplying the term frequency and inverse document frequency of t_i and is expressed as follows.

$$w_{s_f,t_i} = TF(s_f, t_i) \times IDF(t_i) \quad (4)$$

Based on the above equation, we compute the weight of all semantic words in all semantic word collections of a participating project S_p and then form each semantic word collection into a semantic word vector. We formally represent a semantic word vector of a semantic word collection s_f (SWV_{s_f}) as follows.

$$SWV_{S_f} = < w_{t_1}, w_{t_2}, \ldots, w_{t_m} > \quad (5)$$

where m is the number of the semantic words in a semantic word collection. w_{t_i} is the abbreviation for w_{s_f,t_i} indicating the weight of a semantic word t_i in s_f and is at least 0.

D. Developer Contribution Vector

The motivation of the approach proposed in this paper is that developers prefer to work with familiar source files when participating in software project development. In general, the source files contributed by a developer reflect his or her preference for contribution to software project development. Therefore, we consider the contributed source files in external software projects as a query to identify contributable source files in a participating project. We compute the contribution rate of each semantic word in the semantic word collections S_c and represent the contribution rate of all semantic words as a vector.

Before computing the contribution rate, we remove the semantic words that are not contained in S_p. We then determine the contribution rate for each semantic word. The contribution rate of a semantic word (c_{t_i}) is computed as follows.

$$c_{t_i} = \frac{\sum_{s_j \in S_c} I(t_i)}{|S_c|} \quad (6)$$

where $I(t_i)$ is 1 if semantic word t_i is contained in s_j; otherwise, it is 0. c_{t_i} indicates the contribution rate at which the developer contributed the source files associated with semantic word t_i and ranges from 0 to 1. Based on the above equation, we compute the contribution rate of all semantic words contained in S_c. All computed contribution rates of the semantic words are represented as a developer contribution vector (DCV_d).

$$DCV_d = \langle c_{t_1}, c_{t_2}, \ldots, c_{t_n} \rangle \quad (7)$$

where d is a developer, and n is the number of semantic words shared between S_p and S_c. A developer contribution vector contains the primary contributed semantic words of a developer. We use the developer contribution vector as a query vector to identify contributable source files in a participating software project.

E. Computation of Semantic Similarity

In this paper, we use cosine similarity to calculate the similarity between a developer contribution vector and a semantic word vector. Cosine similarity is a measurement for computing the cosign angle between two vectors and has been widely used in various areas such as datamining and document clustering [9]. We compute the similarity between a developer contribution vector and a semantic word vector ($SIM(DCV_d, SWV_{s_f})$) as follows.

$$SIM\left(DCV_d, SWV_{s_f}\right) = \frac{DCV_d \cdot SWV_{s_f}}{\|DCV_d\| \left\|SWV_{s_f}\right\|} \quad (8)$$

where $SIM(DCV_d, SWV_{s_f})$ ranges from 0 to 1; the greater the number of shared semantic words between DCV_d and SWV_{s_f}, the closer to 1. On the other hand, the fewer the shared semantic words, the closer to 0.

IV. EXPERIMENTS

A. Experimental Dataset

GHTorrent [19] provides a large dataset of various software projects developed on GitHub. The GHTorrent data have been widely used in several studies for experimental purposes [19-21]. GHTorrent processes obtained data through the public event stream of GitHub and provides the processed data as MySQL and MongoDB dump databases. Metadata of the developmental history of software projects are stored in the Mysql database. Actual repository data of software projects are stored in the MongoDB database. In this study, we downloaded the MySQL database from GHTorrent that contained GitHub data from January to December 2016. We then examined the metadata of the MySQL database and selected Spring Framework because it is popular, it is large, and most of its contributors have contributed to other software projects as well. Spring Framework includes 2,511 source files and 285,200 developers.

B. Experimental Method

The objective of the experiment is to investigate the usefulness of the proposed approach. To achieve the objective, we compare the proposed approach with the existing approach [22] in identifying source files in the experimental project. To identify source files, the existing method makes basic word vectors, whereas the proposed method makes semantic vectors for source files and queries.

To conduct the experiment, we first select developers in the experimental project as subjects who contributed more than five source files to the experimental project and contributed more than 50 source files to external projects. For each subject, we then use 50 recently contributed source files in external projects to generate a query vector. Given a query vector, the existing and proposed methods identify *N* source files. We set the value of *N* to 25 and 250.

C. Evaluation Metric

To quantitatively evaluate the proposed approach, we use Top-N accuracy. Top-N accuracy has been widely used to evaluate the performance of document retrieval methods and recommendation systems [15, 16, 20, 23]. Top-N accuracy indicates the accuracy of the identification result by a given query. It is computed by dividing relevant elements included in an identification result by the number of identified elements (*N*). In this experiment, the relevant element is the source file contributed by the subjects in the experimental project. We compute the Top-N accuracy of an identification result ($Top\text{-}N_d$) as follows.

$$Top\text{-}N_d = \frac{|I|}{|C_d|} \tag{9}$$

where *d* is a developer, *I* is a set of identified source files, and C_d is a set of source files actually contributed by developer *d*. $Top\text{-}N_d$ ranges from 0 to 1. $Top\text{-}N_d$ is 0 if none of the contributed source files $f \in C_d$ are included in the identified source files or 1 if all contributed source files are included in the identified source files. The greater $Top\text{-}N_d$, the more accurate the identification result.

D. Experimental Result

Table 3 shows the Top-N accuracy of the existing method and the proposed approach. The "User ID" column corresponds to the number of unique identifications assigned to the subjects by GHTorrent. With Top-25 and Top-250 identification, the proposed approach obtained better accuracy than the existing method. With Top-25 identification, the proposed approach obtained better accuracy for all but two subjects (5010240 and 599214). On average, the Top-25 accuracy of the proposed approach was 0.29, and the Top-25 accuracy of the existing method was 0.06. With Top-250 identification, the proposed approach obtained better accuracy than the existing method for all subjects. On average, the Top-250 accuracy of the proposed approach was 0.39, and the Top-250 accuracy of the existing method was 0.07.

E. Discussion

The experimental result shows that the proposed approach obtains better identification results than the existing method. In particular, we believe that the proposed approach can alleviate the decreasing identification accuracy due to the vocabulary gap. To identify source files, the existing method is based on vocabulary similarity. Given a set of words as a query, the existing method tries to find the source files that involve words similar to the given query words. It is limited to identifying source files involving words with the same meaning as the query words but different vocabularies.

Table 3. Results of Top-25 and Top-250 identification

User ID	Top-25		Top-250	
	Existing	Proposed	Existing	Proposed
114374	0.12	0.16	0.14	0.16
14645	0.00	0.28	0.00	0.35
206265	0.52	0.76	0.47	0.68
21536	0.00	0.32	0.00	0.31
30576	0.00	0.60	0.00	0.98
5010240	0.00	0.04	0.00	0.10
599214	0.00	0.04	0.11	0.22
64600	0.00	0.36	0.06	0.22
65552	0.00	0.25	0.00	0.75
70068	0.00	0.10	0.00	0.10
73485	0.32	0.32	0.29	0.24
8327481	0.00	0.00	0.00	0.43
85534	0.00	0.20	0.00	0.41
85543	0.00	0.17	0.00	0.50
9686	0.00	0.68	0.00	0.42
Avg.	0.06	0.29	0.07	0.39

This limitation makes the identification of source files difficult when newly participating developers try to find suitable source files. In general, newcomers to a software project often rely on information retrieval to identify their contributable source files [15, 16, 22]. To find contributable source files, newcomers use as queries the words of source files they contributed to external projects. However, if there is a large vocabulary gap between the source files contributed to external projects and the contributable source files in a participating project, the identification may be unsatisfactory. In the result shown in Section IV-D, we clearly revealed that the existing method is limited and confirmed that the proposed approach can reduce the limitation. Therefore, although the relevance of identified source files is determined by subjective judgment, we believe that the proposed approach can be complementary in the vocabulary gap problem in retrieving source files.

F. Limitations

The threat to validity of this study is related to internal validity and external validity. The threat to internal validity concerns the use of the similarity measurement between the developer contribution vector and the semantic word vector and the concept knowledge tool for converting the words of source files to semantic words. We used cosine similarity measurement and the concept knowledge tool Spotlight. However, there are various similarity measurements for source files and several concept knowledge tools. Therefore, when using other similarity measurements and concept knowledge tools, the experimental results may be observed to be different. To reduce this limitation, we will examine several similarity measurements and concept knowledge tools in future works.

The threat to external validity concerns the generalization of the proposed approach. We used only Spring Framework as the experimental project. We also selected only 15 developers as the subjects. This may not ensure the generalization of the proposed approach to other software projects. We will perform additional experiments with various open-source projects to reduce this limitation.

V. CONCLUSION

Words in source files are commonly used as query terms for a developer to find contributable source files in ongoing projects. The words come mostly from external projects to which the developer has contributed. Owing to the vocabulary gap between the participating project and the external projects, accuracy of contributable source files identification is not guaranteed. In this paper, we proposed a novel approach to identify source files based on semantic similarity to reduce the vocabulary gap between software projects. In this approach, the words in the projects are transformed into semantic words generated using DBPedia Spotlight. The semantic word collections of the external projects and the ongoing projects are used to generate the developer contribution vector and the semantic word vectors, respectively. Contributable source files are then determined by computing the similarity between the developer contribution vector and the semantic word vectors using cosine similarity. The experimental results show that the proposed approach can achieve better accuracy than the existing method based on a comparison of word vocabulary. In future works, we will further validate the proposed approach on several open-source projects. We will also apply several similarity measurements and concept knowledge tools to the proposed approach to improve this study.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education (No. 2016R1D1A1B03931276). This study was also supported by the BK21 Plus Project (SW Human Resource Development Program for Supporting Smart Life), funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (No. 21A20131600005).

REFERENCES

[1] R. E. Kraut, M. Burke, J. Riedl, and P. Resnick, "The challenges of dealing with newcomers," in *Building Successful Online Communities: Evidence-Based Social Design*. Cambridge, MA: MIT Press, pp. 179-230, 2002.

[2] G. Chandrika, "Study on software reliability and reliability testing," *Asia-pacific Journal of Convergent Research Interchange*, vol. 1, no. 1, pp. 7-20, 2015. DOI: 10.21742/apjcri.2015.03.02.

[3] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung, "An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks," *IEEE Transaction on Software Engineering*, vol. 32, no. 12, pp. 971-987, 2006. DOI: 10.1109/TSE.2006.116.

[4] M. Zelkowitz, A. Shaw, and J. Gannon, *Principles of Software Engineering and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[5] R. Jones, R. Kumar, B. Pang, and A. Tomkins, ""I know what you did last summer": query logs and user privacy," in *Proceedings of the 16th ACM conference on Conference on Information and Knowledge Management*, Lisbon, Portugal, pp. 909-914, 2007. DOI: 10.1145/1321440.1321573.

[6] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: a study of developer work habits," in *Proceedings of the 28th International Conference on Software Engineering*, Shanghai, China, pp. 492-501, 2006. DOI: 10.1145/1134285.1134355.

[7] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, and D. Redmiles, "The hard life of open source software project newcomers," in *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, Hyderabad, India, pp. 72-78. 2014. DOI: 10.1145/2593702.2593704.

[8] Y. Park and C. Jensen, "Beyond pretty pictures: examining the benefits of code visualization for open source newcomers," in *Proceedings of the 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, Edmonton, Canada, pp. 3-10, 2009. DOI: 10.1109/VISSOF.2009.5336433.

[9] A. Kuhn, S. Ducasse, and T. Girba, "Semantic clustering: identifying topics in source code," *Information and Software Technology*, vol. 49, no. 3, pp. 230-243, 2007. DOI: 10.1016/j.infsof.2006.10.017.

[10] B. Dit, L. Guerrouj, D. Poshyvanyk, and G. Antoniol, "Can better identifier splitting techniques help feature location?," in *Proceedings of IEEE 19th International Conference on Program Comprehension*, Kingston, Canada, pp. 11-20, 2011. DOI: 10.1109/ICPC.2011.47.

[11] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Communications of the ACM*, vol. 30, no. 11, pp. 964-971, 1987. DOI: 10.1145/32206.32212.

[12] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proceedings of the 9th International Conference on Semantic Systems*, Graz, Austria, pp. 121-124. 2013. DOI: 10.1145/2506182.2506198.

[13] I. Steinmacher, I. S. Wiese, and M. A. Gerosa, "Recommending mentors to software project newcomers," in *Proceedings of 2012 3rd International Workshop on Recommendation Systems for Software Engineering*, Zurich, Switzerland, pp. 63-67, 2012. DOI: 10.1109/RSSE.2012.6233413.

[14] G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella, "Who is going to mentor newcomers in open source projects?," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, Cary, NC, pp. 1-11, 2012. DOI: 10.1145/2393596.2393647.
[15] D. Cubranic and G. C. Murphy, "Hipikat: recommending pertinent software development artifacts," in *Proceedings of 25th International Conference on Software Engineering*, Portland, OR, pp. 408-418, 2003.
[16] Y. Malheiros, A. Moraes, C. Trindade, and S. Meira, "A source code recommender system to support newcomers," in *Proceedings of IEEE 36th Annual Computer Software and Applications Conference*, Izmir, Turkey, pp. 19-24, 2012. DOI: 10.1109/COMPSAC.2012.11.
[17] P. N. Mendes, M. Jakob, A. Garcia-Silva, and C. Bizer, "DBpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th International Conference on Semantic Systems*, Graz, Austria, pp. 1-8. 2011. DOI: 10.1145/2063518.2063519.
[18] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513-523, 1988. DOI: 10.1016/0306-4573(88)90021-0.
[19] G. Gousios and D. Spinellis, "GHTorrent: GitHub's data from a firehose," in *Proceedings of 2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, Zurich, Switzerland, pp. 12-21, 2012. DOI: 10.1109/MSR.2012.6224294.
[20] R. Nielek, O. Jarczyk, K. Pawlak, L. Bukowski, R. Bartusiak, and A. Wierzbicki, "Choose a job you love: predicting choices of GitHub developers," in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*, Omaha, NE, pp. 200-207, 2016. DOI: 10.1109/WI.2016.0037.
[21] Y. Zhang, D. Lo, P. S. Kochhar, X. Xia, Q. Li, and J. Sun, "Detecting similar repositories on GitHub," in *Proceedings of IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*, Klagenfurt, Austria, pp. 13-23, 2017. DOI: 10.1109/SANER.2017.7884605.
[22] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY: McGraw-Hill, 1983.
[23] J. Kim and E. Lee, "Understanding review expertise of developers: a reviewer recommendation approach based on latent Dirichlet allocation," *Symmetry*, vol. 10, article no. 114, 2018. DOI: 10.3390/sym10040114.

Jungil Kim
He received his Ph.D. degree at Kyungpook National University, Daegu, Korea, in 2017. He is currently Post-Doc in Department of Software Technology Laboratory at Kyungpook National University. His current research interests are mining software repositories, software change prediction and efficient software project development support method.

Geunho Choi
received his in Computer Enginnering from Yeungjin University, Daegu, Korea, in 2016. He is currently a M.E. student at Kyungpook National University, Daegu, Korea. His research interests are static code analysis, repository mining, and task recommendation for developers.

Eunjoo Lee
received her B.S., M.S., and Ph.D. degrees in Computer Science from Seoul National University, Korea in 1997, 1999, and 2005, respectively. She was a research staff member at Samsung Advanced Institute of Technology from Nov. 2005 to Feb. 2006. Currently, she is an associate professor of School of Computer Science and Engineering at Kyungpook National University. Her current interests include software repository mining, change prediction, and software evolution.