**Who should pick me up?**

**An approach for identifying suitable source files**

Geunho Choi and Eunjoo Lee\*

School of Computer Engineering, Kyungpook National University

80 Daehakro, Bukgo

Deagu, Republic of Korea

bchoi000@gmail.com, ejlee@knu.ac.kr

**Abstract**

[[1]](#footnote-2)

The quality of developed software is closely related to the contribution of participating developers. When developers participate in a new software project, they must first find artifacts, such as source files, with which they can contribute to the project. As a software project generally incorporates a large number of source files, it is important to quickly and accurately identify suitable source files, in which the developers have experience or interest, to reduce the developers’ search time and effort. In this paper, we describe how to convert a source file to an entity frequency vector using DBpedia spotlight tool and how to calculate the similarity between source files included in the project and source files earlier worked on by a developer. In an experiment with Spring Framework, the similarity of the source files contributed by a single developer was higher than that between other source files. Thus, our approach is applicable to identifying source files that are suitable for a developer who wants to join a particular software development project.

Keywords—repository mining; GitHub; source analysis

**I. Introduction**

TABLE I

Preprocessing

|  |  |
| --- | --- |
| Camel case split | getHttpHeader -> get http headerOrderComparator -> order comparator |
| Additional stopwords | int, package, return, public, throws, case, const, class, interface, authors, param, code, version… |

TABLE II

Annotation

|  |  |
| --- | --- |
| In source file | Annotation result |
| synchronizing, sync, synced, synchronize, synchrony, syncs, syncing, … | Synchronization |
| parsing, decoding, parsers, parse, decode, parses, parser, parsable, … | Parsing |

Understanding source files included in the project is considerably time-consuming for developers [1][2]. When developers participate in a new software project, they must first find artifacts, such as source files, to which they can contribute. Therefore, faster identification of suitable source files in a project based on a developer's past development history can be expected to reduce the search time and effort of developers.

A source file is written by including words about features in the form of the method name, comment, and class name[3] , and is written according to the specific coding convention[4] for each project. Therefore, even if the same feature is implemented in different projects, the composition of words may vary, as shown in Fig. 2. Thus, when a word vector is generated with respect to a source file included in the different project, several similar words representing the same meaning will be included. Based on this, we cannot guarantee the reliability of the calculated similarity. The spotlight tool performs the annotating task of converting several similar words contained in the input text into the same entity defined in DBpedia, and it is possible to merge similar words and identifying meaningful words in the source file.

In this paper, we propose an approach to convert the source files included in the developer’s development history and project them into an entity frequency vector, and identify suitable source files for the developer based on the calculated similarity between the vectors. In an experiment with the Spring Framework, we calculated the similarity between generated developer vectors based on the development history of each developer in Spring Framework and generated file vectors based on the source files included in the Spring Framework. The results show that the similarity of contributed source files are, on average, 30% higher than that of non-contributed source files. Thus, the similarity between the developer vector and the file vector can be used to identify source files that are suitable to a newly participating developer.

 The rest of this paper is structured as follows. Section II introduces related works based on Github and Spotlight. Section III describes our overall approach, how to generate entity frequency vectors, and how to calculate the similarity between developer and file vectors. In Section IV, the result of the experiment is shown on Spring Framework. Finally, Section V concludes the paper.



Fig. 1. An overview of our approach



Fig. 2. Implementations of the same function that removes whitespace

**II. Related works**



Fig. 3. Preprocessing and annotation

**Github** is web-based coding platform that provides various collaborative features related to project development, such as source version control and issue management. Its scale is growing dramatically, and related active research is proceeding. Hauff et al. proposed an approach of matching developers to job advertisements based on the similarity between vectors. They convert job advertisements and the contents of readme files in the project-included developer repository in Github into vectors[5]. Zhang et al. proposed an approach to identify projects that are similar to a query project based on readme files and star information in the Github repository[6], and Visilescu et al. combined and analyzed information from Github and StackOverflow [7]. Zhang et al. proposed an approach recommendation related repository based on user behavior[8]. A study was also performed on the recommendation of suitable reviewers for pull-requests[9][10]. Most of the related works have been based on surface data, such as readme files, commit, star, and filename, but our proposed approach is based on source files included in the development history and the project.

**III. Approach**

 In this chapter, we describe the process of preprocessing and annotation for source files, and then described the generation of an entity frequency vector based on identified entities. Fig .1 shows an overview of our approach.

*A. Preprocessing*

The source file consists of code and comments that are written in keeping with coding conventions, and contains many meaningless words. Therefore, we proposed preprocessing for the reliability of the vector. The two methods of preprocessing are **removing stopwords** and **camel case splitting** for include meaningful words. **Camel case splitting** is used to extract words from the class name and function name, which often contain meaningful words. **Removing stopwords** eliminates stopwords, i.e., java keywords and comment keywords from the source file that are not related to the developer experience. Table I shows the camel case split process and java keywords included as stopwords.

*B. Annotation*

Spotlight is built on DBpedia ontology [12] that contains information on about 5 million named entities, and provides an annotating feature for identifying defined named entities in the input text. It is used for named entity recognition and information extraction. Annotation is performed by merging a set of words generated as result of preprocessing. Table II shows the process of preprocessing and annotating synonyms into one word, and Fig. 3 represents the whole process of preprocessing and annotation.

The entity set $D$ generated by the result from annotation as follows:

 Definition 1. Result of Annotation

|  |  |  |
| --- | --- | --- |
|  | $D= <t\_{1},t\_{2},t\_{3},…,t\_{n}>$, |  |

where

n = number of entities included in D

t = entity generated by the result from annotation

*C. Generate Entity Frequency Vector*

 The frequency calculated by dividing the frequency number on each *t* contained in $D$ by the sum of the frequency number of *t* contained in $D$*,* is used as the word frequency. If we used $f\_{t, D}$(raw count), there is a large difference in element values between the *Developer Vector* generated based on a set of *n* source files and the *File Vector* generated based on a single file, and the normalization is carried out by dividing them into the size of each set $D$.

|  |  |  |
| --- | --- | --- |
|  | $tf\_{D}\_{t}=\frac{f\_{t,D}}{\left|D\right|}$, | (1) |

where

$f\_{t,D}$ = frequency of $t$ in $D$



Fig. 4. Cosine similarity distribution

$D\_{file}$ generated by the preprocessing and annotation each source files included in the project. Calculate entity frequency (2)of each entity included in $D\_{file}$, and generate vector based on calculated entity frequencyanddefined as the *File Vector(FV)*.

Definition 3.1 : File Vector (FV)

$FV\_{file\_{i}} = <tf\_{t\_{1}},tf\_{t\_{2}},…,tf\_{t\_{n}}>$,

where the *File Vector* consists of $tf$ (entity frequency), and $n$ is the number of elements in the vector.

A set of all entities generated by preprocessing and annotating *n* source files extracted from the development history of developer $a (dev\_{a})$ is merged and defined as$ D\_{dev\_{a}}$. Thus, $D\_{dev\_{a}}$ represents the set of all entities included in n source files committed by $dev\_{a}$. The entity frequency vectoris generated based on $D\_{dev\_{a}}$, which is defined as the *Developer Vector (DV)*.

Definition 3.2 : Developer Vector (DV)

$DV\_{dev\_{a}} = <tf\_{t\_{1}},tf\_{t\_{2}},…,tf\_{t\_{m}}>$,

where the *Developer Vector* for $dev\_{a}$ consists of $tf$ (entity frequency), and $m$ is the number of elements in the vector.

*D. Calculate similarity*

The similarity measure between *DV* and *FV* is calculated by using the cosine similarity measure. The cosine similarity is calculated by the following equation:

|  |  |  |
| --- | --- | --- |
|  | $$SIM\left(DV\_{dev\_{a}},FV\_{file}\_{i}\right)=\frac{DV\_{dev\_{a}}∙FV\_{file}\_{i}}{\left‖DV\_{dev\_{a}}\right‖\left‖FV\_{file\_{i}}\right‖}$$ | (2) |

 The higher the similarity value calculated based on (2), the higher the similarity between the source files of *FV* and the source files of *DV.* Thus, source files with higher similarity can be seen as suitable source files for a developer to contribute to the project.

**IV. Experiments**

Part of the aim of this experiment evaluates the reliability of the calculated similarity based on the cosine similarity between the generated *DV* based on the past development history of each developer in the Spring Framework and the generated *FV* based on the sources file included in Spring Framework.

*A. Experimental Dataset*

 GHTorrent[11] is open source project that collects data about public events occurring in Github and provides them in MySQL and MongoDB. Developer- and Project-related data are extracted from 2016 Github data (for 2016.01–2016.12) provided by GHTorrent. Spring Framework, developed by Github, was selected as the experimental project. Spring Framework is a framework written in Java, and is a massive open source project with more than 6000 source files and over 200 developers contributing.

Among developers who participated in Spring Framework, we extracted developers who have entered more than 5 commits to the Spring Framework. A total of 47 developers were extracted, and 9 developers were selected based on more than 5 commits for non-forked projects before Spring Framework participation. Commits for the forked project are often not relevant to the developer experience; therefore, they were excluded.

*B. Experimental method*

The *DV* for each selected developer is generated based on a set of 100 source files extracted from the development history before participating in Spring Framework. In order to compare the similarity between the source files that are actually committed by a developer and the source files that are not committed, the source files included in Spring Framework are classified into a group of committed source files (C) and a group of uncommitted source files (UC). Then, the cosine similarity is calculated for all the source files included in each group, and the results are compared.

*C. Experiment result*

 TABLE III

Experiment Result-Spring Framework

*avg\_cos\_{group}* : $\frac{total similarity}{file count of group}$

*diff* : $(1- {avg\\_UC}/{avg\\_C)} \*100$

|  |  |  |  |
| --- | --- | --- | --- |
| ID | avg\_cos\_C | avg\_cos\_UC | Diff |
| 9686 | 0.151 | 0.112 | 25.8% |
| 14645 | 0.375 | 0.183 | 51.2% |
| 21536 | 0.245 | 0.186 | 24.1% |
| 25509 | 0.323 | 0.166 | 48.6% |
| 64600 | 0.346 | 0.206 | 40.5% |
| 73485 | 0.08 | 0.05 | 37.5% |
| 85534 | 0.151 | 0.126 | 16.6%. |
| 114374 | 0.145 | 0.130 | 10.3% |
| 206265 | 0.245 | 0.187 | 23.7% |

Table III shows the average (*avg\_C*, *avg\_UC*) and difference (diff) of the average for each group similarity, and it was found that *avg\_C* was, on average, 30% higher than *avg\_uc*. Fig. 4 shows the distribution of the cosine similarity between the *DV* of developer 14645 and the *FV* of the source file included in groups C and UC, and this shows the difference in distribution between two groups. Developers who contributed to similar files to files included in past development history (50%-14645) showed higher cosine similarity than developers who did not (10%-114374). Therefore, most of the developers included in the experimental data contributed to source files similar to their own development history before project participation. Thus, identifying files with higher similarity to the *DV* can be applied to identify source files for developers.

**V. Conclusion**

In this paper, we have studied an approach to convert source files to entity frequency vectors using Spotlight and calculated the similarity between vectors, for identifying source files that are suitable to developers who want to join a particular project. The experimental results on the Spring Framework show that the similarity between committed source files is higher than the similarity of uncommitted source files. Therefore, the approach we proposed is appropriate to identify source files that are suitable for developers who want to join the project.

In future work, we plan to experiment on projects other than Spring Framework, and complement the approach to calculate similarity. Moreover, we plan to study a visualization method to effectively display the identified source files.

**Acknowledgment**

This study was 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 (21A20131600005), and also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A3011005).

**References**

[1] R. Minelli, A. Mocci and M. Lanza, "I Know What You Did Last Summer - An Investigation of How Developers Spend Their Time" ICPC, 2015, pp. 25-35.

[2] T. D. LaToza, G. Venolia, and R. DeLine, “Maintaining mental models: a study of developer work habits.” ICSE, 2006, pp. 492-501.

[3] A. Kuhn, , S. Ducasse and T. Gı´rba, “Semantic clustering:

identifying topics in sourcecode.” ISSN, 2007, pp 230–243.

[4] S. McConnell, *Code Complete*, 2nd ed., Microsoft Press, 2004, pp. 844-845

[5] C. Hauff and G. Gousios, “Matching GitHub developer profiles to job advertisements” MSR, 2015, pp. 362-366.

[6] Y. Zhang et al, “Detecting Similar Repositories on GitHub” SANER, 2017, pp. 13-23.

[7] B. Vasilescu, V. Filkov and A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge", 2013, SocialCom, pp. 188-195.

[8] L. Zhang, Y. Zou, B. Xie and Z. Zhu “Recommending relevant projects via user behaviour: an exploratory study on github” CrowdSoft, 2014, pp. 25-30.

[9] Y. Yu, H. Wang, G. Yin and T. Wang, “Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?” ISSN, 2016, pp. 204-218.

[10] Y. Yu, H. Wang, G. Yin and C. X. Ling, "Reviewer Recommender of Pull-Requests in GitHub" ICSME, 2014, pp. 609-612.

[11] G. Gousios, D. Spinellis, "GHTorrent: Github's data from a firehose" MSR, 2012, pp. 12-21.

[12] P. N. Mendes, M. Jakob, A. G. Silva and C. Bizer “DBpedia Spotlight: Shedding Light on the Web of Documents” I-Semantics, 2011, pp 1-8.

1. \* correponding author [↑](#footnote-ref-2)