Aggregation of Development History from Distributed Support Systems

Hiroki Kawai
Department of Information Engineering
Nara National College of Technology
Nara, Japan
Email: h-kawai@info.nara-k.ac.jp

Hidetake Uwano
Department of Information Engineering
Nara National College of Technology
Nara, Japan
Email: uwano@info.nara-k.ac.jp

Soichiro Tani
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, Japan
Email: soichiro-t@is.naist.jp

***Abstract*—This paper proposes a method to recommend the relevant information of the document which recorded in the development support systems such as BTS and VCS. We improve a system which we implemented in previous work with the method proposed in this paper. Our method get a document from the support systems, extract the word, then calculate the feature vector based on the TF-IDF value of each word. In the experiment, we apply the proposal method to the dataset from an open source software projects, and evaluate the recommendation accuracy between the six clustering algorithm. The result of the experiment shows that the proposed method improves the recommendation accuracy compared with the previous work.**

I. Introduction

With increasing outsourcing of software development or open source software projects, developers need to collaborate with geographically distributed co-workers and share information between them. Communications between distributed co-workers are very important for efficient, high-quality software development, however, it is difficult because of distance and the time differential between them [1]. In such the environment, they communicate each other with asynchronous communication tools/systems such as mailing list (ML), bug tracking system (BTS) or version control system (VCS) that we call development support systems (DSS) in this paper [2][3].

These DSSs record the information such as clash report, step to reproduction, assignment of a developer, and source codes after the fault correction. Such records that stored in different DSSs are usually referenced simultaneously to understand the whole context of the fault correction or function enhancement. For example, developer who assigned to a fault reads the clash report in the BTS, understands the plan to fixation discussed on ML, and specify the code to fix from VCS records. Also they examine past records of similar fault or enhancement to refer discussions or source codes.

On the other hand, examine the records which belong to a context from multiple DSSs is difficult. For example, developer who assigned a fault has to read mails stored in ML and tickets in BTS which concern with the fault. To do this, the developer needs knowledge of the fault to search the BTS and ML. However, such knowledge or context is lacking from the record in DSSs, and also difficult to estimate from mails or tickets. Some system integrates multiple DSSs and allows developers input connection between information. Such integrated system support the developers to understand the context, yet difficult to apply small-scale organization and open source software project because they often use a rental hosting service such as SourceForge.net.

For such problem, authors proposed a system that recommends the information belong to same context of an information which developer currently watching [4]. The system serves as a proxy server which detects user's access to DSSs, collects the same information that the user refers to, and make an association with several kinds of the referred information in multiple DSSs. In this paper, we propose an algorithm to improve recommendation accuracy of the system. Our proposal algorithm extracts a characteristic of each information from the words which used in the information by TF-IDF method. Using a cluster algorithm, group of information which divided in same cluster is recommended to the user.

The rest of the paper is structured as follows. Section II explains DSSs and their problems while use them simultaneously. Section III describes our method to integrate the information between multiple DSSs. In Section IV and V, we discuss experiment and the result. Finally, Section VI concludes the paper with future work.

II. Development Support Systems

Development Support Systems (DSS) helps developers share the information to collaborate with other developers on development. Most of software development projects use multiple DSSs in their project. For example, when a developer found a fault in their software, s/he 1) reports the fault at the BTS, 2) discusses correction policy through the ML, and 3) commits a corrected source code to VCS. Each development project selects or creates the DSSs which proper for their projects. On the other hand, small scale projects or organizations such as Open Source Software (OSS) projects employ free hosting DSS services such as Google Code[1] or Source Forge[2] to reduce the management cost.

In case of using multiple DSSs in the project, each developer needs to search several kinds of information in DSSs to

[1]http://code.google.com/hosting/
[2]http://sourceforge.net/

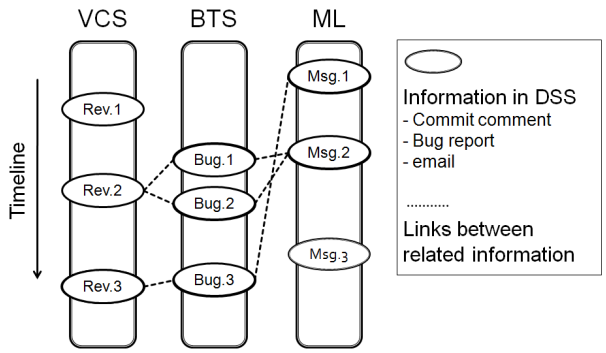

Fig. 1. Development Context in Multiple DSSs

understand the development context. Fig. 1 shows an example of development context exist at multiple DSSs.

Three rounded rectangles represent VCS, BTS, and ML system respectively. Each ellipse in the DSSs show describe the information recorded in each systems such as bug report, mail, and commit comment. Dotted lines between the informations show these informations belong to a same development context. As shown in the figure, each information in same context recorded in the different systems based on the type of the information. Developers search the informations with the context which they understood through the project.

Information search from the DSSs are extremely difficult when the developers have a minor knowledge about the context. For instance, a fault was reported on the BTS with a few explanation of the fault, then discussion and change history were recorded in ML and VCS. Here, a developer who has no knowledge about the fault must review the bug report, then speculate the keywords to search the other DSSs. This is a time-consuming, lead-to-mistake work for the developer. Despite for this, context during the development is not recorded in the DSSs; most of DSS depends on manual recording of links to related information by the developers. Therefore, developers who recently joined the project is hard to understand the context of the project

To resolve the problem explained above, some systems which integrate the multiple DSSs are proposed[5]. Integrated system manages the information such as bug report, mail, commit history, and other information. The system allows the developer searches different informations which belong to different types at a time. However, there are some problems to be solved;

- **Difficult to transfer information from existing DSS**
  Proposed integration systems have an insufficient export/import function of the information and their connection. Replace the system without export/import of the information or with manual re-input is impractical.
- **Cannot connect external hosting services**
  Some of the integrated system allows to connect between DDSs by modify the existing DSSs. However, rental hosting services that used in the OSS projects cannot modify, hence such systems are limited to adopt.
- **Lack of functionality**
  When using the multiple DSSs in the project, developers select the each DSS based on the functionality of each system. On the other hand, when using the integrated system in the project, developers can use only the function which integrated system provide.

To tackle the above problems, authors developed a system that integrate the multiple DSSs into an integrated system without any modification[4]. Our system runs as Web proxy server in the user computer, and detects the access to a DSS through the HTTP. The system captures the information from the user access to the DSS, calculate the related information from other DSS, and then combine them. The combined information is displayed to the user as the result of access to DSS. Fig. 2 shows an example of the system output.

Left side of the figure shows a default output of VCS used in an open source software project. Right side of the figure shows a VCS output with recommendation result by our system. In addition to the default output, lists of related documents recorded in BTS and VCS are displayed. Here, the system recommends the five relevant documents from each DSSs.

The system works as proxy server in the user computer, hence, the user can use the stored information in each DSS without any modification. Current implementation of the system requires specific keywords which represents the context of the each development project. The keywords are specified by the developer who understands the project, although the selection of the keywords affects the recommendation accuracy. A set of documents in a development project is updated frequently, hence the recommendation based on such the static keywords decrease the accuracy. Therefore, our proposal method uses TF-IDF to select the keyword dynamically. In this paper, we propose an algorithm to improve recommendation accuracy of the our system without keywords selection.

III. Proposal Method

Our proposal method extracts a list of words which used in every information from DSSs by word segmentation, then calculates TF-IDF of each word; TF-IDF matrix consisting of the words that contained in the each information is used as feature vector. The method clusters the information with clustering algorithm based on the TF-IDF matrix. Information in the same cluster is recommended as a related information.

A. Word Segmentation

Word segmentation divides written text into meaningful words. We use the word segmentation tool to extract the words from each information and count them. As an implementation, we selected open source morphological analysis tool, MeCab [6].

B. TF-IDF

TF-IDF is a statistical metric which reflects how important a word is for characterize a document in a set of documents [7]. TF-IDF value is calculated from following formula.

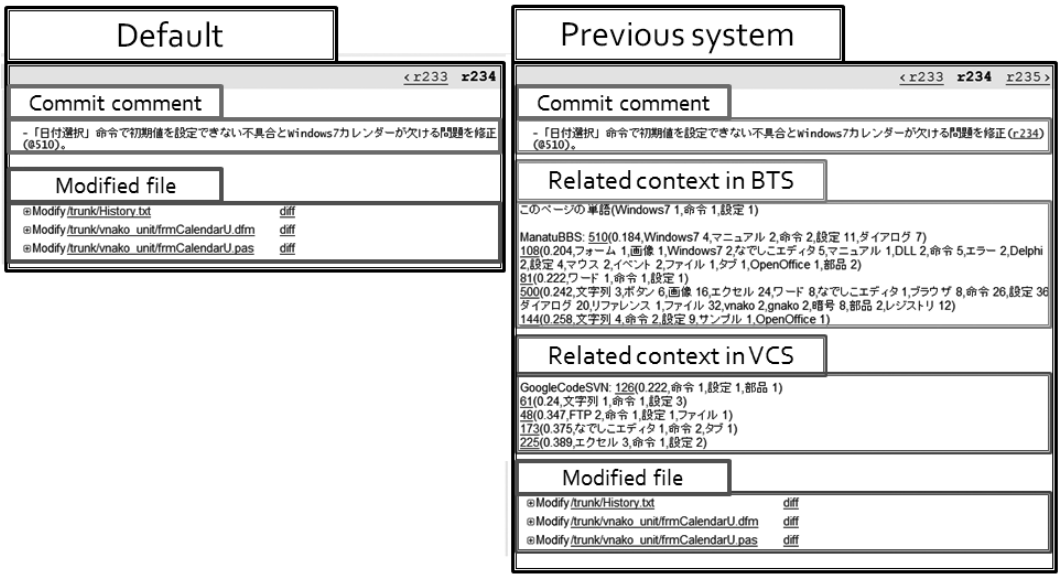

Fig. 2. Screenshot of the proposal system[4]

$$tf_{c,n} = \frac{d_{c,n}}{\sum_{i=1}^{C} d_{i,n}} \quad (1)$$

$$idf_c = log_e \frac{|D|}{|\{d : t_c \in d\}|} \quad (2)$$

$$TF - IDF_{c,n} = tf_{c,n} \cdot idf_c \quad (3)$$

$tf_{c,n}$ describes how many contained a word c into a document n. Here, $d_{c,n}$ means the number of c contained in n. idf_c describes how appear c in a set of document. $|D|$ shows a total number of document in the document set, and $|\{d : t_c \in d\}|$ describes the number of document which include c. The higher value of idf describes the word c appears at only a few documents; that is the document n is characterized by the word c. A set of TF-IDF values derived from the words within a document represents a feature vector of the document. Our method employs the feature vector an array of TF-IDF values as a feature quantity of the information from DSSs.

C. Cluster Analysis

Cluster analysis classifies a set of objects into groups based on the characteristics of each object. We adopt the cluster analysis to recommend a set of information which similar to the information the user is currently browsing from DSSs. The process of clustering in our method is as follow;

1) Divide the each information in the DDSs as different clusters.
2) Calculate distance between clusters with clustering algorithm.
3) Connect two clusters that is nearest than others as one cluster.
4) Continue the step two and three until the every cluster is connected.

We compare the major six algorithms to calculate the distance between clusters described below;

- **Complete Linkage Method**
  The maximum distance between a document in one cluster and a document in other cluster is used as the distance between the clusters.
- **Single Linkage Method**
  The minimum distance between a document in one cluster and a document in other cluster is used as the distance between the clusters.
- **Group Average Method**
  The average distance between documents in one cluster and documents in other cluster is used as the distance between the clusters.
- **Centroid Method**
  Distance between the center of gravity in two clusters is used as the distance between the clusters. The centers of two clusters are calculated based on the number of the document in each cluster.
- **Median Method**
  Distance between the center of gravity in two clusters is used as the distance between the clusters. The centers of two clusters are calculated without the weighted by the number of the document in each cluster.
- **Ward Method**
  Form the cluster as to maximize the ratio between the variance in the cluster and the variance between the clusters.

D. Procedure

The procedure of the proposal method is as follow;

1) **Word Segmentation**
   List words from the every information recorded in DSSs

with MeCab.

2) **Calculate TF-IDF value**
   Calculate the TF-IDF value of every word in each information. Words which used only once within the whole information in the DSSs are removed because cannot use for recommendation.
3) **Clustering**
   Cluster the information by one of the clustering algorithm based on the TF-IDF value. The result of this step makes one cluster which includes every information.
4) **Divide the cluster**
   Divide the result into twenty clusters based on the distance.
5) **Recommendation**
   Recommend a set of information which are divided into a same cluster to the user.

Compared with previous work, the method recommends the information without specific keywords selected by the developer, which is particularly useful for new developers of the project.

IV. Experiment

We experiment with the proposal method to confirm the recommendation accuracy. In the experiment, we adapt the method to a data set of the open source software project, Nadeshiko[3]; a programming language using Japanese sentences on method/variable name and other operations. The data set includes documents from BTS (1,842 tickets recorded from Oct. 2008 to Sept. 2010) and VCS (235 checkins recorded from Aug. 2008 to Sept. 2011) used in the project.

Recommendation accuracy of the method is evaluated with linkage between documents which recorded by developers. We presume that the linkage between documents by the developers is a vital evidence that the documents have a relevant. Therefore, we calculate three metrics from the linkage information for each document; recall, precision, and F1-value. These three metrics are calculated with following four variables;

- True Positive (TP): Number of documents which is included in correct answers and recommended by the method.
- False Positive (FP): Number of documents which is included in correct answers and not recommended by the method.
- False Negative (FN): Number of documents which is not included in correct answers but is recommended by the method.
- True Negative (TN): Number of documents which is not included in correct answers, and is not recommended by the method.

Precision is the percentage of the related documents that is recommended by the method. Precision can be expressed as the following expression. Precision takes a range from zero to one, the higher the value means documents are recommended to developers with lesser misclassification.

[3]http://nadesi.com/

TABLE II
Number of Documents in each Cluster

	VCS		BTS	
	Min.	Max.	Min.	Max.
Complete Linkage Method	2	39	6	142
Single Linkage Method	1	70	1	1818
Group Average Method	4	39	6	145
Centroid Method	1	39	6	145
Ward Method	4	39	56	152
Median Method	1	39	1	196

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \quad (4)$$

Recall is the percentage of the recommended documents in the all related documents. Recall can be expressed as the following expression. Recall takes a range from zero to one, the higher the value means the method recommends the document to developers more correctly.

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \quad (5)$$

F1-value is the harmonic mean of precision and recall, it can be expressed as the following expression. F1-value takes a range from zero to one, indicates high value when the both precision and recall are high.

$$F1-value = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (6)$$

V. Result and Discussion

A. Recommendation Accuracy

Table I shows a result of the experiment on two DSSs. Each value describes Precision, Recall, and F1-value of the method with six clustering algorithm. The results from VCS show the Recall is high on the every clustering algorithm (0.816 on average.) On the other hand, Precision is very low on the every algorithm (0.086 on average.) Therefore, F1-value becomes a very low (0.102) on average. The results from BTS show a similar tendency with VCS. Here, Recall is very high (0.944), however Precision is low (0.040) on average. As a result, F1-value in TBS is also low (0.050.) The result depicts the method recommends lot of documents which belong to irrelevant documents. A main cause of the result is the number of the cluster which decided as fixed value. In this experiment, each data from DSSs were divided into twenty clusters. VCS includes 235 documents, therefore, one cluster from the VCS data consists of 11.75 documents on average. Also BTS includes 1,842 documents, hence, one cluster from the BTS data consists of 92.1 documents on average. In this software projects, links between the documents were connected one by one by developers. As a result, most of document has only one link, that is, true positive (TP) is 1. Therefore, it is considered that Precision of most cluster in VCS became 0.085 (1/11.75) on average.

TABLE I
ACCURACY FOR VCS AND BTS

	VCS			BTS		
	Precision	Recall	F1-value	Precision	Recall	F1-value
Complete Linkage Method	0.090	0.798	0.113	0.035	0.999	0.065
Single Linkage Method	0.074	0.833	0.061	0.072	0.986	0.018
Group Average Method	0.073	0.821	0.105	0.011	0.996	0.022
Centroid Method	0.091	0.798	0.095	0.035	0.996	0.065
Ward Method	0.089	0.845	0.134	0.035	0.999	0.066
Median Method	0.095	0.798	0.104	0.049	0.989	0.061
Average	0.086	0.816	0.102	0.040	0.994	0.050

Table II shows the number of documents that included in each cluster of the VCS and BTS. The table shows that a little difference between clustering algorithms were observed in VCS. However, a difference was observed between the algorithms in BTS. The most characteristic result was shown at Single linkage method in six clustering methods. In the case of Single linkage method in BTS, 1,818 documents were included in one cluster. This means 1,817 documents are recommended in one document to developer. Similarly, in the case of Single linkage method, 70 documents were included in one cluster of VCS.

Fig. 3 shows a tree that is created from clustering result of VCS data with Single Linkage Method. Each number in the tree represents a document or cluster. The length of the horizontal line represents the distance between clusters, and the vertical line represents the clusters which were combined to one cluster. The broken line represents the root that was divided into twenty trees from single tree. The figure shows the most documents has been classified into the single cluster represented at bottom of the figure. Results from other clustering algorithms show a fewer maximum numbers of documents in one cluster, hence, the Single Linkage Method is unsuitable for the target project.

The results suggest that the our proposal method recommend the relevant documents with slight leakage. On the other hand, the method recommends 12 documents from VCS, and 92 documents from BTS on average. This number of the recommended documents bring a much work to the developers. To improve the precision of the method, decrement of the cluster size is required. Improvement of the Precision without decrease of Recall is the our future work.

B. Comparison with the Previous Work

1) Recommendation Accuracy: This section compares the accuracy between the proposal method and our previous result [4]. The system implemented in the previous work recommends the five documents which is most similar to a target document. Accuracy measurement in the previous work is the percentage of relevant documents within the above five document. Here, we compare the accuracy measurement in previous work and Recall of this work.

Fig. 4 shows a comparison between the proposal method and the previous work. Recall of VCS and BTS in the proposal methods are 0.816 and 0.994, respectively. Recommendation accuracy of VCS and BTS in the previous system are 0.533 and 0.579, respectively. The figure suggests the accuracy of the proposal method in this paper is higher than the accuracy in previous work.

2) Applicability: Our previous work used the number of specific keywords that appears in the document as a characteristic of the document. The specific keywords were selected by the contributor of the target projects to represent the characteristics in the project. Because the words have been determined by the developer subjectivity, it is difficult to apply the technique to other development projects. In addition to this, it was impossible to recommend the document which does not include the words.

Compare to the previous work, the proposal method in this paper selects the keyword from the dataset by TD-IDF. Therefore, the method can adapt to the other development projects without the knowledge of the contributor.

VI. CONCLUSION

In this paper, we improved the system which created in the previous work to solve the problem that developer cannot access to the information belong to the same context smoothly. We proposed the recommendation method to describe the relevant information of the information the user currently looking at. Our method gets a document from the DSSs, extract the word, then calculate the feature vector based on the TF-IDF value of each word. Then the method classifies the documents from the degree of similarity of feature vectors, and documents in the same cluster are recommended as relevant information. The result of the experiment showed that the proposed method improves the recommendation accuracy compared with the previous work.

Improvement of the Precision and F1-value is one of the future work. We used fixed number of clusters to be divided in this paper. Effect of dynamical change of the number based on the accuracy metrics such as F1-value should be verified at the future work. Another future work of the study is verification of the our method with different dataset from other development projects. We will examine whether the characteristics of software development dataset affects the recommendation accuracy. Modification of clustering algorithm based on the characteristics of the software development dataset is also the future work.

We used term frequency and inverse document frequency to recommend a document to a developer. However, the high value of idf decreases importance of the word which appears

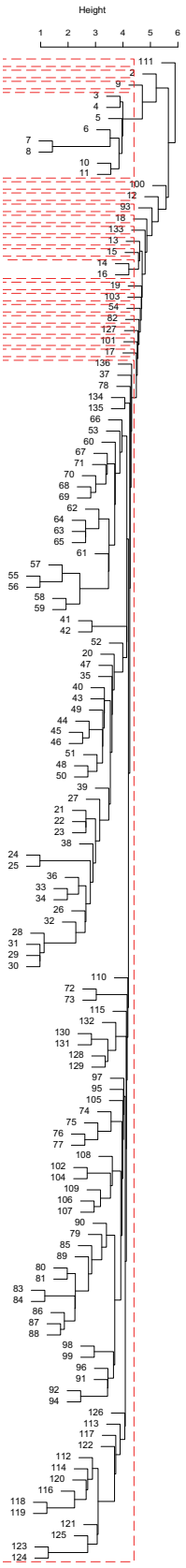

Fig. 3. Tree of VCS with Single Linkage Method

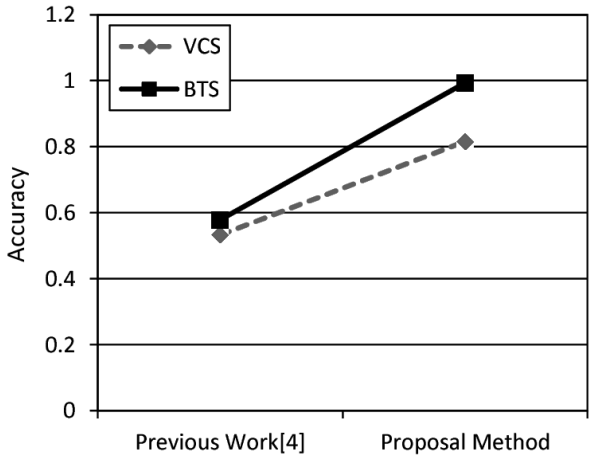

Fig. 4. Comparison of Proposal Method and Previous Work

on many documents. That is, if a developer find a fatal fault in their project and discusses about the fault actively, the importance of the keyword used in the discussion is decreased. To avoid the problem, co-occurrence frequency between keywords is a useful metrics.

ACKNOWLEDGMENT

This work is being conducted as a part of Grant-in-aid for Young Scientists (B), 24700038.

REFERENCES

[1] C. R. B. De Souza, S. D. Basaveswara, and D. F. Redmiles, "Supporting Global Software Development with Event Notification Servers," In Proc. the ICSE 2002 International Workshop on Global Software Development, 2002.

[2] B. Sengupta, S. Chandra, and V. Sinha, "A Research Agenda for Distributed Software Development," In Proc. the 28th International Conference on Software Engineering (ICSE), pp.731-740, 2006.

[3] C. Gutwin, R. Penner, and K. Schneider, "Group Awareness in Distributed Software Development," In Proc. the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW), pp.72-81, 2004.

[4] S. Tani, A. Ihara, M. Ohira, H. Uwano, and K. Matsumoto, "A System for Information Integration between Development Support Systems," In Proc. the 3rd International Workshop on Empirical Software Engineering in Practice (IWESEP), 2011.

[5] M. Ohira, R. Yokomori, M. Sakai, K. Matsumoto, K. Inoue, and K. Torii, "Empirical Project Monitor: A Tool for Mining Multiple Project Data," In Proc. International Workshop on Mining Software Repositories (MSR2004), pp.42–46, 2004.

[6] MeCab, "MeCab: Yet Another Part-of-Speech and Morphological Analyzer," http://mecab.sourceforge.net/ 2011.

[7] G. Salton, and M. J. McGill, "introduction to modern information retrieval," McGraw - Hill, NewTork, 1983.