Application of Collaborative Filtering for Software Component Retrieval System

Makoto Ichii, Reishi Yokomori, and Katsuro Inoue
Graduate School of Information Science and Technology, Osaka
{m-itii,yokomori,inoue}@ist.osaka-u.ac.jp

Abstract

A search engine for software component helps developers to reuse software component and to understand its behavior. We are studying about a software component search engine and constructs SPARS-J for Java source codes. SPARS-J provides useful information obtained by static analysis of source codes in a repository. However, we also consider that SPARS-J provides more useful information by using the analysis result of its retrieval history. In this paper, we suggest a recommendation method by using collaborative filtering technique, and evaluated the effectiveness of the system implemented of SPARS-J

Keywords: Software Component, Software Retrieve, Collaborative Filtering

1 Introduction

In recently years, the importance of reuse of software components has been widely recognized in actual development. There are many assistance tool of software reuse; In particular, we consider that an effective use of repositories of already developed software is very important.

As the most effective usage of software components, we suggest a search engine for java programs, named SPARS-J(Software Product Archiving and Retrieving System for Java)[3]. SPARS-J provides useful information, such as components using (or used by) the component, similar components, package information, and so on, by static analysis of source codes in the repository. By using SPARS-J, developers can find necessary components easier. However, we can easily imagine that the search efficiency differs among people and multiple users often request a same kind of query for the similar purpose. Therefore, we think that more efficient search can be performed by feeding back the search history to the user.

In this paper, we propose the method of recommending components by applying the technique of collaborative filtering to the search history. Using this method, users can get exact components which satisfy their requirements. We also implement this method as an actual tool on SPARS-J, and evaluate its usefulness by an experimentation.

2 Collaborative Filtering

Collaborative filtering is a recommendation technique based on the idea that "People who agreed in the past will probably agree again" [6]. At first, the systems using this technique obtain the rating of each item from each user and store them up, and predict the fondness of each user. And they recommend the item which is highly rated by the user whose fondness is similar. Such recommendation systems are widely used for Netnews, movies, music, and so on.

Tapestry[1] is an early recommendation system using collaborative filtering. In this system, the user can get a recommendation of E-mail or Netnews by designating the recommender whom (s)he wants to get recommendation from. On the other hand, GroupLens[6] determines the recommender and the information to recommend automatically. A user votes the articles of Netnews based on five-level rating, and GroupLens recommends based on the tendency of the rating.

On the other hand, there are also systems which get rating implicitly. Phoaks[4] acquires and recommends URLs which are automatically collected from Netnews, FAQ, and the reports. Moreover, There is a system using collaborative filtering for the software function[5]; Ohsugi consider the use of software function as implicit vote. The history of software function execution is analyzed and automatic recommendation is performed.

Considering the user's effort, we think it more appropriate for the application to a search engine to collect ratings implicitly from the search history. Our system considers the reference history of software components as implicit vote, and recommends ones automatically.

3 Proposed Method

The processes of our recommendation method are as follows:

1. System records the reference history to a database as user's rating.
2. Based on the rating recorded in the database, system calculates correlation coefficient between each user.

3. By using correlation coefficient and rating of each user, system calculates recommendation value of each component.
4. Based on recommendation values of each component, system recommends components to browse.

Followings are the details of each phase.

1:Recording of search action

In the case of the technique of recommendation based on the user's explicit rating, such as the one of GroupLens, there is a merit which can catch a user's intention certainly. However, there is also a demerit to reduce an absolute number of user's rating because each user isn't forced to rate the all articles they read. Since our system is for an aid to the search engine of software components, we think that there are few users which can consume time and effort. Therefore, we treat a user's reference history itself as rating. Concretely speaking, when a source code of a component is displayed in SPARS-J, we regard that the user votes 1 to the component.

Because search purpose of a user changes, the components which are recommended by using all of his or her history may be irrelevant. Thus, we use only ratings in the user's "session" which is the beginning of use of the system to the end. In other words, we consider a session as an user.

2:Calculation of correlation coefficient

The correlation coefficient method used by GroupLens is not work when the rating scale is binary, such as a track record on HTML. Therefore, we adapt the modification method which proposed by Breese[2], and use the following formulas for calculation of correlation coefficient $c(a,i)$ between the target user a and another user i.

$$v_{i,j} = \begin{cases} 1 & \text{if } j \in I_i \\ 0 & \text{if } j \notin I_i \end{cases}, \ \bar{v}_i = \frac{1}{|I|}\sum_{j \in I} v_{i,j}$$

$$c(a,i) = \frac{\sum_{j \in I}(v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_{j \in I}(v_{a,j} - \bar{v}_a)^2 \sum_{j \in I}(v_{i,j} - \bar{v}_i)^2}}$$

In these formulas, I_i represents the components which the user i has voted, and I is the union set of the components which either user a or i has voted and some components neither a nor i has voted.

3:Calculation of recommendation value

About each user whose correlation coefficient against the target user is positive, the system calculates weighted average from the correlation coefficient. A system considers that the weighted average is a recommendation value for each component.

Recommendation value $p_{a,k}$ is calculated by a below equation, in which a is the target user , k is a component, and U is the set of users i where $c(a,i) > 0$ and $|I_i| > 1$.

$$p_{a,k} = \frac{\sum_{i \in U} c(a,i)^2 v_{i,k}}{\sum_{i \in U} |c(a,i)^2|}$$

4:Recommendation to user

Based on the recommendation value of each component, our system recommends by the following two methods;

1. About all components which have the recommendation value more than a threshold, a system shows a user those in descending order of its recommendation value. It's intended to recommend required components for browsing effectively.
2. For all components which have a use relation with the current- viewing component, a system shows a user them in descending order of its recommendation value. It's intended to recommend useful use-relation for browsing.

4 Implementation of Recommendation System on SPARS-J

Based on the proposed method in previous section, we implemented the recommendation system of software components on SPARS-J, which is a search engine for Java programs. For the implementation, we add the following functions to SPARS-J;

- Recording of search history
  When a searcher requests the display of the source code of a component, system stores it in a database to grasp that the component is seen in the session.
- Calculation of recommendation value
  The system calculates the similarity of the present session and another session, and the recommendation value of each component.
- Display of search history
  The system displays the list of components which the searcher looked at in the session.
- Display of recommended components
  The system displays two ways described previously.

4.1 Experimentation

We evaluated the effectiveness of implemented system on SPARS-J. In the evaluation, we confirm that the recommendation system contributes to improvement in the reference efficiency.

In the experimentation, 8 examinees write Java program by mounting to a skeleton code. We divide 8 examinees into two groups, and prepared two kinds of situations for each 4 subjects; the case that an examinee can use only the

Table 1. Result

	Search Time				Precision			
	P1	P2	P3	P4	P1	P2	P3	P4
G1	34.8	18.5	**21.5**	**15**	0.36	0.18	**0.88**	**0.79**
G2	**12.5**	**3.2**	24.2	26	**0.89**	**0.73**	0.73	0.66

search result of SPARS-J, and the case that an examinee can use both the search result of SPARS-J and output of recommendation system, respectively. For each examinee and each subject, we measured working times to finish, time to make a search, and the precision about components which he browsed. Here, we consider the components, which can be regarded as practically used one, as adequate components.

A procedure of the experimentation is as follows;

1. All 8 examinees try an examples P0 for practice. By this result, they are divided into 2 groups, G1 and G2.
2. The examinees of G1 work against two subjects, P1 and P2, and the examinees of G2 work against two subjects, P3 and P4, respectively. In this phase, all examinees can use only the search result of SPARS-J.
3. The examinees of G1 work against two subjects, P3 and P4, and the examinees of G2 work against two subjects, P1 and P2, respectively. In this phase, all examinees can use both the search result of SPARS-J and output of recommendation system.

Table 1 represents the result of each group G1 and G2. P1, P2, P3, and P4 represent each subject. The cases using a recommendation function are indicated by boldface. From this result, we can confirm that the case using the recommendation function is better than the case using only SPARS-J, for each subject. Although the difference of the subject P3 is small, this is because many examinees had a generally knowledge about the field of P3. These result shows that a recommendation function is useful to improvement in search efficiency.

As a problem which happened during the experiment, some examinees could not get effective recommendation. This is because the system has a policy of "Do not recommend already displayed components", so we should reconsider this policy.

5 Conclusion

In this paper, we suggest a recommendation method by collaborative filtering technique, and evaluated the effectiveness of implemented system on SPARS-J. The experiment result showed that collaborative filtering is also effective for support of software component search.

As future work, we are planning the improvement of accuracy by weighting a history, and the improvement of a user interface.

References

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry: "Using Collaborative Filtering to Weave an Information Tapestry.", *Communications of the ACM*, Vol.35, No.12, pp.61-70, 1992.

[2] J. S. Breese, D. Heckerman and C. Kadie: "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp.43-52, 1998.

[3] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, S. Kusumoto: "Component Rank: Relative Significance Rank for Software Component Search", In *Proceedings of the 25th International Conference on Software Engineering (ICSE2003)*, pp. 14-24, Portland, Oregon, U.S.A., 2003.

[4] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter: "PHOAKS: A System for Sharing Recommendations", *Communications of the ACM*, Vol.40, No.3, pp.59-62, 1997.

[5] N. Ohsugi, A. Monden, and K. Matsumoto : "A Recommendation System for Software Function Discovery". In *Proceedings of the 9th Asia-Pacific Software Enginieering Conference (APSEC2002)*, pp.248-257, 2002.

[6] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl: "GroupLens: An Open Architecture for Collaborative Filtering of Netnews", In *Proceedings of the 1994 CSCW*, pp. 175-186, 1994.