Recommendation of Software Technologies Based on Collaborative Filtering

Tomohiro Akinaga, Naoki Ohsugi, Masateru Tsunoda, Takeshi Kakimoto,
Akito Monden, Ken-ichi Matsumoto
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
+81 743 72 5312
{ tomohi-a, naoki-o, masate-t, takesi-k, akito-m, matumoto } @is.naist.jp

Abstract

Software engineers have to select some appropriate development technologies to use in the work; however, engineers sometimes cannot find the appropriate technologies because there are vast amount of options today. To solve this problem, we propose a software technology recommendation method based on Collaborative Filtering (CF). In the proposed method, at first, questionnaires are collected from concerned engineers about their technical interest. Next, similarities between an active engineer who gets recommendation and the other engineers are calculated according to the technical interests. Then, some similar engineers are selected for the active engineer. At last, some technologies are recommended which attract the similar engineers. An experimental evaluation showed that the proposed method can make accurate recommendations than that of a naïve (non-CF) method.

Keywords: information retrieval, similarity computation algorithms, recommender systems, education

1. Introduction

Today, there are vast amount of software development technologies. A report of Thomson Corporation said that there are 4,700,000 software technologies patented in the world in 2002 [11]. In addition to them, there are many unpatented software technologies, such as Capability Maturity Model for Software (SW-CMM) [8] and eXtreme Programming (XP) [1]. Many new technologies are developed every day, so number of them must increase.

On the other hand, software development engineers have to select some appropriate development technologies to use in the work. There are two main reasons. First, although each technology has possibility to improve development productivity and software quality, its effectiveness depends on context. For example, SW-CMM is a famous software process model used to construct capability for improving the process. This model is more helpful for large organizations than small ones [2]. Therefore, effectiveness of an agile method like XP is higher than SW-CMM for engineers who work in small company. Second, the engineers have not enough time to learn many technologies. Engineers spend a considerable amount of time on work about developing software products, such as discussing, designing, coding and testing. In the 2004's report, Japanese Ministry of Economy, Trade and Industry (METI) said that embedded software engineers spent 160 hours or more for the work per a month; on the other hand, they could use only 200 hours or less for learning per a year [4].

However, engineers sometimes cannot find the appropriate technologies because the number of the technologies is too large. The engineers are unaware of some effective technologies because the technologies are hidden with vast amount of the others. Some lucky engineers may find some names of previously unknown technologies with journals, websites or search engines; yet they have to gather additional information of the found technologies because they do not know effectiveness of the technologies. Finally, they can find some effective technologies if their efforts paid off. Or, all of the found technologies may be useless if things turn out bad.

In order to solve this problem, we propose a recommendation method which helps engineers to find technologies which seem to be suitable for their work. Our recommendation method is based on collaborative filtering (CF) [5], [7], [10]. CF is considered a powerful infor-

	t_1	t_2	$\cdots$	t_j	$\cdots$	t_b	$\cdots$	t_n
u_1	$r_{1,1}$	$r_{1,2}$	$\cdots$	$r_{1,j}$	$\cdots$	$r_{1,b}$	$\cdots$	$r_{1,n}$
u_2	$r_{2,1}$	$r_{2,2}$	$\cdots$	$r_{2,j}$	$\cdots$	$r_{2,b}$	$\cdots$	$r_{2,n}$
$\cdots$	$\cdots$	$\cdots$		$\cdots$		$\cdots$		$\cdots$
u_i	$r_{i,1}$	$r_{i,2}$	$\cdots$	$r_{i,j}$	$\cdots$	$r_{i,b}$	$\cdots$	$r_{i,n}$
$\cdots$	$\cdots$	$\cdots$		$\cdots$		$\cdots$		$\cdots$
u_a	$r_{a,1}$	$r_{a,2}$	$\cdots$	$r_{a,j}$	$\cdots$	$r_{a,b}$	$\cdots$	$r_{a,n}$
$\cdots$	$\cdots$	$\cdots$		$\cdots$		$\cdots$		$\cdots$
u_m	$r_{m,1}$	$r_{m,2}$	$\cdots$	$r_{m,j}$	$\cdots$	$r_{m,b}$	$\cdots$	$r_{m,n}$

Fig. 1. m×n table used for recommendation

mation filtering method, and has been used in *recommender systems* that estimate end-users' preferable items (books, movies, tunes, etc). The system based on CF assumes that, "if users had similar evaluation to same items, they would also have similar evaluation to another items", and the system recommends items which are highly evaluated by similar users.

In our proposed method, we assume "if engineers had similar interest to same technologies, they would also have similar interest to another items". In order to recommend useful technologies, we inquired the engineers' interest levels for known technologies and summarized it as dataset. Using the dataset, our method automatically selects users similar to the active engineer who needs recommendation of technologies. Similar user has similar tendency of interests for technologies. Then, our method recommends technologies which similar users are interested in but are not known to the active engineer.

In what follows, section 2 explains the details of Collaborative Filtering. Section 3 describes recommendation procedure and algorithms of our proposed method. Section 4 shows an experiment to evaluate the recommendation accuracy of proposed method. In the end, section 5 discusses the conclusion and future work.

2. Collaborative Filtering

Collaborative Filtering (CF) is one of the key techniques for implementing a recommender system that recommends to a user a set of candidate items which may be preferable or useful to the user. For example, Resnick et al. [9] developed GroupLens system which recommends interesting Usenet articles to users. It draws on a simple idea; people who agreed in their subjective evaluation of past articles are likely to agree again in the future. Resnick et al. proposed a basic CF algorithm known as a user-based method.

User-based method makes recommendations with the following procedure:

1. After using items (Usenet articles, books, movies, etc.), users explicitly assign numeric ratings to the items.
2. A recommender system correlates the ratings in order to determine which user's ratings are most similar to other ones.
3. The system predicts how much users will like new items, based on the ratings of similar users.
4. If these new items seem to be liked, the system recommends them to the user.

Sarwar et al. [10] proposed another basic CF algorithm called item-based method. Item-based method can make recommendation with extremely sparse dataset whose ratio of rated items to whole items is only 1%. Item-based method makes recommendations with executing 2' and 3' instead of 2 and 3 in the above procedure respectively:

2'. A recommender system correlates the ratings in order to determine which item's ratings are most similar to other item's.

3'. The system predicts how much users will like new items, based on the ratings of similar items already rated by the users.

Intuitively, an idea of the item-based method can be represented as a popular sentence of Amazon.com's recommendations "Customers who bought this book also bought...".

In our proposed method, we regard each user as each software development engineer, and each item as each software development technology. We assume "if engineers had similar interest to same technologies, they would also have similar interest to another items". We use user-based method and item-based method to recommend technologies.

3. Recommendation Method

In our method, we use the database in form of $m \times n$ matrix as shown in Fig. 1 where $u_i \in \{u_1, u_2, ..., u_m\}$ denotes i-th engineer, $t_j \in \{t_1, t_2, ..., t_n\}$ denotes j-th technology, and $r_{i,j} \in \{r_{1,1}, r_{1,2}, ..., r_{m,n}\}$ denotes rating of interest which an engineer u_i has for a technology t_j. Rating of technology is indicated with "not interested" (1) ,…, "very interested" (4) and larger number denotes grater interest. When an engineer does not know a technology, $r_{i,j}$ is set missing value.

In our recommendation method, two collaborative filtering method, user-based method and item-based method is used to recommend technologies. Recommendation with user-based method is made as follows:

1. The similarity between the active engineer u_a and other engineer u_i is calculated by equation (1).

$$sim\left(u_a, u_i\right) = \frac{\sum_{j \in T_a \cap T_i} \left(r_{a,j} - \overline{t_j}\right) \times \left(r_{i,j} - \overline{t_j}\right)}{\sqrt{\sum_{j \in T_a \cap T_i} \left(r_{a,j} - \overline{t_j}\right)^2} \sqrt{\sum_{j \in R_a \cap R_i} \left(r_{i,j} - \overline{t_j}\right)^2}} \quad (1)$$

In this equation, T_a and T_i denote a set of technologies which u_a and u_i has rated. $\overline{t_j}$ denotes average value of ratings for technology t_j.

2. Using the similarity, $\hat{r}_{a,b}$, the predicted value of $r_{a,b}$ (i.e. the rating of the active engineer u_a for the development technology t_b) is calculated by equation (2).

$$\hat{r}_{a,b} = \frac{\sum_{i \in k-neighbors} \left(r_{i,b} - \overline{u}_i\right) \times sim\left(u_a, u_i\right)}{\sum_{i \in k-neighbors} sim\left(u_a, u_i\right)} + \overline{u}_a \quad (2)$$

In this equation, *k-neighbors* is a set of k engineers (similar engineers) who rates technology t_b and have high similarity to the active engineer u_a. Since the size of *k-neighbors* k affects recommendation accuracy, we varied k and adopt it which gave the highest accuracy in an experimental evaluation. $\overline{u_i}$ denotes average value of rating which engineer u_i has.

3. Predicted value is computed for all technologies which active engineer u_a does not rate. Technologies are recommended to the active engineer, ranked by their predicted rating.

Recommendation with item-based method is made as follows:

1. For a technology t_b which are not rated by active engineer u_a, the similarity between the technology t_b and other technologies t_j is calculated by equation (3).

$$sim\left(t_b, t_j\right) = \frac{\sum_{i \in U_b \cap U_j} \left(r_{i,b} - \overline{u_i}\right) \times \left(r_{i,j} - \overline{u_i}\right)}{\sqrt{\sum_{i \in U_a \cap U_j} \left(r_{i,b} - \overline{u_i}\right)^2} \sqrt{\sum_{i \in U_b \cap U_j} \left(r_{i,j} - \overline{u_i}\right)^2}} \quad (3)$$

In this equation, U_b and U_j denote a set of engineers who rate technology t_b and t_j.

2. Using the similarity, $\hat{r}_{a,b}$, the predicted value of $r_{a,b}$ is calculated by equation (4).

$$\hat{r}_{a,b} = \frac{\sum_{j \in k-nearestTechs} r_{a,j} \times sim\left(t_b, t_j\right)}{\sum_{j \in k-nearestTechs} sim\left(t_b, t_j\right)} \quad (4)$$

In this equation, *k-nearestTechs* is a set of k technologies (similar technologies) which are rated by the active user u_a and have high similarity to the technology t_b.

3. Predicted value is computed for all technologies which active engineer u_a does not rate. Technologies are recommended to the active engineer, ranked by their predicted rating.

4. Experimental Evaluation

4.1 Dataset

In the experiment, we made a dataset from questionnaire of 77 people which consist of 29 engineers who engage in software development, 37 graduate students who major in information science, and 11 academic researchers who research software engineering. When answering the questionnaire, an engineer answered whether he knows each technology or not. We regarded the technology is known when an engineer can explain the outline of the technology. When he knew the technology, he rated it based on his interest with 4 grades. Each grade is not interested (1), a little interested (2), interested (3), and very interested (4). Table 1 shows software development technologies written on the questionnaire and engineers' responses. In Table 1, the column of known shows the percentage of engineer who knows each technology, and the column of interested shows the percentage of engineer who rated the technology as more than 3. For example, 97.4% of engineers know Visual Basic, and 25.3% of them are interested in it (i.e. they are rated it as 3 or 4).

4.2 Evaluation Criteria

We used 5 criteria (mean absolute error, recall, precision, F1 value, and NDPM) to evaluate recommendation accuracy of the proposed method. These are often used to evaluate accuracy of CF based system [6]. Except for mean absolute error, the higher these values are, the more accurate evaluated method is.

Mean absolute error (MAE) is the average of absolute

Table 1. Software development technologies written in questionnaire and engineers' responses

Software Development Technology	Known (%)	Interested (%)	Software Development Technology	Known (%)	Interested (%)
eXtreme Programming (XP)	85.7	57.6	Pascal	92.2	12.7
SCRUM	29.9	39.1	Perl	97.4	33.3
Lean Software Development (LSD)	33.8	30.8	Ruby	88.3	32.4
Adaptive Software Development (ASD)	23.4	33.3	Python	63.6	14.3
Crystal Family	14.3	36.4	LISP	77.9	16.7
Future Driven Development (FDD)	24.7	31.6	COBOL	90.9	15.7
eXtreme Modeling (XM)	20.8	31.3	Function Point Method	88.3	50.0
RUP	72.7	41.1	COCOMO	77.9	43.3
ISO9000	83.1	31.3	COCOMOII	70.1	55.6
SW-CMM	84.4	47.7	Agile COCOMO	27.3	47.6
CMMI	84.4	52.3	Goal Question Metric (GQM)	59.7	54.3
Personal Software Process (PSP)	75.3	60.3	eXtensible Markup Language (XML)	100.0	74.0
Team Software Process (TSP)	57.1	52.3	Simple Object Access Protocol (SOAP)	75.3	41.4
UML	100.0	80.5	Web Service	98.7	50.0
PMBOK	61.0	57.4	Service Oriented Architecture (SOA)	70.1	42.6
SWEBOK	57.1	52.3	Structured design	94.8	58.9
Java	98.7	77.6	Object Oriented Design	98.7	73.7
Enterprise Java Beans (EJB)	81.8	44.4	Total Quality Control (TQC)	49.4	52.6
Java Server Pages (JSP)	81.8	42.9	Total Quality Management (TQM)	45.5	51.4
Active Server Pages (ASP)	70.1	25.9	Statistical Quality Control (SQC)	41.6	53.1
PHP Hypertext Preprocessor (PHP)	79.2	32.8	J2SE SDK	83.1	62.5
Common Gateway Interface (CGI)	90.9	37.1	Windows API	88.3	44.1
C/C++	100.0	62.3	Microsoft Foundation Classes (MFC)	64.9	34.0
.NET	96.1	37.8	Visual Component Library (VCL)	33.8	19.2
Visual Basic (VB)	97.4	25.3	Qt	29.9	30.4
K Desktop Environment (KDE)	50.6	28.2			

error. Absolute error is defined as follows:

$$\text{Absolute Error} = |\,\text{Actual Value} - \text{Predicted Value}\,| \quad (5)$$

Precision is a ratio of appropriately recommended technologies to entire recommended technologies, formally defined as (6), where N_r is the number of entire recommended technologies, and N_a is the number of appropriately recommended technologies. We regarded N_r is the number of technologies whose predicted rate is more than a certain threshold., and N_a is the number of technologies which is actually rated more than 3 and whose predicted rate is more than a certain threshold.

$$Precision = \frac{N_a}{N_r} \quad (6)$$

Recall is a ratio of appropriately recommended technologies to entire technologies actually rated more than 3 by the engineer u_i, formally defined as (7), where N_u is the number of entire technologies actually rated more than 3 by the engineer u_i, and N_a is the number of appropriately recommended technologies.

$$Recall = \frac{N_a}{N_u} \quad (7)$$

F1 value is a combined criterion of recall and precision, formally defined as (8).

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (8)$$

NDPM is an evaluation metrics to measure the system performance. NDPM is calculated by comparing the difference between "order of items in ideal recommendation

Table 2. Criteria at peak F1 value

	F1 Value	MAE	Recall	Precision
User-based	0.72	0.62	74%	70%
Item-Based	0.76	0.56	85%	69%
Average-Based	0.65	0.73	88%	52%

for the user" and "order of items in system's recommendation for the user". The lower number of NDPM denotes that system provides better recommendation. NDPM is formally defined as (9)…(11).

$$ndpm(P_a, R_a) = \frac{\sum_{j \in P_a \cup R_a} \sum_{k \in P_a \cup R_a} dpm(p_{aj}, p_{ak}, r_{aj}, r_{ak})}{\sum_{j \in P_a \cup R_a} \sum_{k \in P_a \cup R_a} dpmNormalizer(p_{aj}, p_{ak})} \quad (9)$$

where

$$dpm(p_{aj}, p_{ak}, r_{aj}, r_{ak}) = \begin{cases} 0 & \left(\begin{array}{l} (p_{aj} > p_{ak}) \wedge (r_{aj} > r_{ak}) \\ \vee (p_{aj} < p_{ak}) \wedge (r_{aj} < r_{ak}) \end{array} \right) \\ 1 & \left((p_{aj} \neq p_{ak}) \wedge (r_{aj} = r_{ak}) \right) \\ 2 & \left(\begin{array}{l} (p_{aj} > p_{aj}) \wedge (r_{aj} < r_{ak}) \\ \vee (p_{aj} < p_{aj}) \wedge (r_{aj} > r_{ak}) \end{array} \right) \end{cases} \quad (10)$$

$$dpmNormalizer(p_{aj}, p_{ak}) = \begin{array}{ll} 1 & (p_{aj} = p_{ak}) \\ 2 & (p_{aj} \neq p_{ak}) \end{array} \quad (11)$$

The numerator of the equation (9) is the sum of results of the function *dpm* on the set of ideal recommendation P_a and the set of actual predicted recommendation R_a. We assume that predicted ratings r_{aj} of the active engineer for technologies j that are $(j \in P_a) \wedge (j \notin R_a)$, and ideal ratings p_{aj} of the active engineer for technology j that are $(j \notin P_a) \wedge (j \in R_a)$, are 0. The denominator of the equation (9) normalizes the numerator to the range [0, 1].

4.3 Experimental Procedure

In the experiment, we evaluated both the user-based method and the item-based method. First, we made a dataset from engineers' rating of interest. Rating of a technology unknown to an engineer (i.e. not rated) was treated missing value. Next, we tried several CF computation algorithms and adopted the most accurate algorithms. Then, we conducted an experiment as follows (leave-one-out cross-validation):

1. i-th engineer u_i is regarded as the active engineer, and removed from the dataset.
2. $r_{i,j}$ is regarded as unknown, and predicted rating is computed.
3. Repeat Step 2 for all j.
4. Repeat Step 1, 2, 3 for all i.

We also computed recommendation scores using a naïve (non-CF) method called *average-based method* to compare with proposed methods. In the average-based method, the predicted rating $\hat{r}_{a,b}$ is formally defined as (12), where N is total number of entire engineers.

$$\hat{r}_{a,b} = \frac{\sum_{i=1}^{m} r_{i,b}}{N} \quad (12)$$

4.4 Experimental Result

We conducted pre-examination to set the neighborhood size, and set k as 6 for both user-based method and item-based method, since recommendation accuracy became the highest.

We changed threshold to find the highest F1 value. The highest F1 value in the experiment is shown at Table 2. Other criteria at Table 2 are calculated when F1 value is the highest. Fig. 2 shows the relation between recall and precision, written with changing threshold. From Table 2, we see that item-based method has the highest F1 value and the lowest MAE of three methods. Average-based method has the lowest F1 value and the highest MAE. At Table 2, user-based method has the highest precision and average-based method has the highest recall. But at Fig. 2, the line of item-based method is almost upper position,

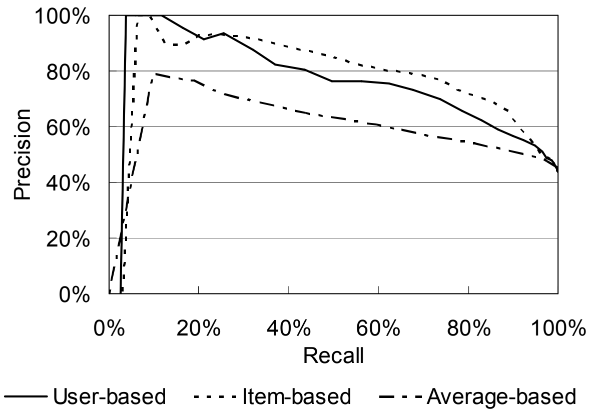

Fig. 2. The relation between recall and precision

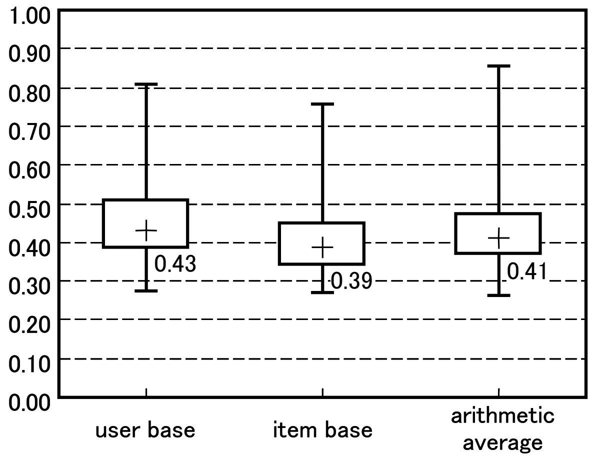

Fig. 3. Boxplots of NDPM of each method

and the line of average-based method is almost lower position. Therefore, from Table 2 and Fig. 2 we conclude that item-based method is the most accurate and average-based method is the most inaccurate.

Fig. 3 shows boxplots of NDPM of each method. From the figure, we see that item-based method made the most appropriate recommendation, but user-based method made the most inappropriate recommendation. This result indicates that user-based method is not appropriate for recommendation of technology. Therefore, user-based method is more suitable for finding similar engineers than recommending useful technologies.

The limitation of our experiment is that the dataset used in the experiment is a part of whole technologies. In our future work, we should make a questionnaire included more technologies and inquire more engineers to make larger dataset and confirm experimental result.

In our experiment, inquired technologies are known to many engineers. However, a new technology would be rated by little engineers. We should investigate the relationship between recommendation accuracy and a number of ratings.

5. Conclusion

In this paper, we proposed a recommendation method based on Collaborative filtering using the engineers' interest levels for technologies. An experimental evaluation showed that the recommendation accuracy of the proposed method was higher than that of a naïve (non-CF) method.

Our future work is to enlarge our dataset and to conduct an experiment using it to confirm our conclusions. Also, we will implement the system based on the proposed method, and examine actual users' evaluation for the system.

References

[1] K. Beck, "Extreme Programming Explained: Embrace Change," Addison-Wesley, New York, 1999.

[2] B. Boehm, and R. Turner, "Balancing Agility and Discipline: A Guide for the Perplexed," Addison-Wesley, 2003.

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

[4] Embedded Software Development Capability Promotion Committee, "Survey of Embedded Software Development Industry Status in FY2004," Ministry of Economy, Trade and Industry, 2004 (in Japanese).

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

[6] J. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, "Evaluating collaborative filtering recommender systems," ACM Transactions on Information Systems (TOIS), Vol.22 , No.1, pp.5-53, 2004.

[7] N. Ohsugi M. Tsunoda, A. Monden, and K. Matsumoto, "Applying Collaborative Filtering for Effort Estimation with Process Metrics," Proc. of the 5th Int'l Conf. on Product Focused Soft. Process Improvement, Springer, Berlin Heidelberg, pp.274-286, 2004.

[8] M. Paulk, B. Curtis, M. Chrissis, and C. Weber, "Capability Maturity Model for Software (Version 1.1)," CMU/SEI-93-TR-024, 1993.

[9] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," Proc. ACM Conf. on Computer Supported Cooperative Work (CSCW'94), pp.175-186, Chapel Hill, North Carolina, U.S.A, Oct. 1994.

[10] B. M. Sarwar, G. Karypis, J. A. Konstan, and J.T. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms," Proc. 10th International World Wide Web Conference (WWW10), pp. 285-295, Hong Kong, May, 2001.

[11] Thomson Corporation, "Thomson Derwent Patent Focus Report 2002-3," http://scientific.thomson.com/, 2003.