Naoki Ohsugi, Masateru Tsunoda, Akito Monden, Ken-ichi Matsumoto, “Effort Estimation Based on Collaborative Filtering”, In *the 5th International Conference on Product Focused Software Process Improvement (PROFES2004)*, Kansai Science City, Japan, pp.274-286, Springer, Berlin Heidelberg, April, 2004.

Effort Estimation Based on Collaborative Filtering

Naoki Ohsugi, Masateru Tsunoda, Akito Monden, and Ken-ichi Matsumoto

Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City, 630-0192 Japan
{naoki-o, masate-t, akito-m, matumoto}@is.aist-nara.ac.jp

Abstract. Effort estimation methods are one of the important tools for project managers in controlling human resources of ongoing or future software projects. The estimations require historical project data including process and product metrics that characterize past projects. Practically, in using the estimation methods, it is a problem that the historical project data frequently contain substantial missing values. In this paper, we propose an effort estimation method based on *Collaborative Filtering* for solving the problem. Collaborative Filtering has been developed in information retrieval researchers, as one of the estimation techniques using defective data, i.e. data having substantial missing values. The proposed method first evaluates similarity between a target (ongoing) project and each past project, using vector based similarity computation equation. Then it predicts the effort of the target project with the weighted sum of the efforts of past similar projects. We conducted an experimental case study to evaluate the estimation performance of the proposed method. The proposed method showed better performance than the conventional regression method when the data had substantial missing values.

1 Introduction

Many researches have proposed estimation methods of software development effort [1], [2], [6], [7], [17], [18], [20]. They provide systematic procedures to predict the effort (e.g. man-months) of ongoing or future projects based on historical (past) project data. Accurate estimates provide practitioners many advantages in managing the development. These allow project managers to make effective use of valuable resources. Or these give marketers baselines of bid for external contracts; i.e., they can know how many people will be needed to complete the certain works in the particular duration. In fact, estimation methods have not been limited to uses for software development; can be used for any kind of tasks. For instance, a recent study built a method to estimate the effort required to become ISO 9001 certified [13].

These estimation methods usually uses both process and product metrics to characterize each project. Process metrics, which quantify attributes of the development process and of the development environment, are typically collected by hand in forms of progress reports. On the other hand, product metrics, which quantify characteristics of artifacts (e.g. source code) produced during the development process, can be automatically collected by measurement tools.

One of the practical problems in using the estimation methods is that the historical project data usually contain substantial numbers of missing values [4], [9]. Especially, process metrics contain larger numbers of them since they are collected by hand. One reason is that different divisions in an organization may have different policy of data collection, i.e. some project collects a particular metric while other projects do not. And, even if the organization has a unified policy, not all the metrics are collected in each project due to pressing development schedule. Furthermore, the organizational policy to collect a particular set of metrics often changes as the organization's software process matures, thus, older projects tend to have less data than new projects.

Some complementary techniques have been developed for dealing with missing values (henceforth, missing data techniques or MDTs) [10]. Strike et al. [19] conducted experimental evaluation of some MDTs, and found all of the techniques perform well in terms of small biases and high precision. The techniques were: *listwise deletion*, *mean imputation* and some types of *hot-deck imputation*. Listwise deletion is the simplest technique to ignore data sets that have missing values. Mean imputation is a technique to fill the missing values on a variable with the mean of data sets that are not missing. Hot-deck imputation is alternative forms of imputation that are based on estimates of the missing values using other variables from the subset of the data that have no missing values. They concluded that the simplest technique, listwise deletion, is a reasonable choice.

However, MDTs can give bad influences to the accuracy of estimation, according to the amount and distribution of missing values. Kromrey and Hines [12] analyzed the performance of MDTs with a two predictor regression model. Their results showed that listwise deletion technique did not performed well when the level of missing data was more than 30 percent of all data. And the imputation techniques did not perform well when the distribution of missing values is not uniformly. In the context of effort estimation, the ratio of data that have missing values can be frequently larger than 30 percent [5]. And the distribution of missing values is usually not uniform.

In this paper, we propose *Collaborative Filtering* (CF) based effort estimation method, under the assumption that the (historical) predictor data have a large amount of missing values. CF is one of the estimation techniques using defective data having substantial missing values, in information retrieval research domain. The proposed method first evaluates similarity between a target (ongoing) project and each past project, using vector based similarity computation equation. Then, it predicts the effort of the target project with weighted sum of the effort of past similar projects.

Up to date, there is no method that can effectively estimate software development effort with a large amount of missing values, while CF have been developed for estimating user preferences using defective data that have substantial missing values. Estimating a user's preference is to answer a question such as: which items are likely to be preferred by the user? And, to what degree the user seems to prefer the items? As we consider this is an estimation of a value of a metric (an item), we believe CF can be applied for estimating the effort as well as the user preferences, to improve prediction accuracy. However, conventional CF cannot be directly applied to software metrics since value range of each metric is not constant while that of item ratings in CF are constant.

In this paper, we propose a procedure on how to apply CF for estimating software development effort. We also present results of experimental case study conducted for evaluating the performance of the proposed method. Our evaluation criteria focus on the accuracy of prediction with the absolute and relative error. In the case study, we used industrial data, which had been collected in a company, including several kinds of process metrics that have large amount of missing values (approximately 60% of all data), as an evaluating data set. We compared the proposed method, vs. the *Stepwise Multiple Regression Models* enhanced by some different MDTs.

In what follows, Section 2 introduces related works. Section 3 proposes an estimation procedure based on CF. Section 4 describes a procedure of an experimental case study for evaluating the proposed method. Section 5 shows the results of our case study and discusses their implications and limitations. In the end, Section 6 concludes the paper with a summary and some future topics.

2 Related Work

CF has been developed as a technique for realizing *Recommender Systems*, by information retrieval research community. Recommender Systems estimate individual user's preferences, for recommending likely preferred items selected from vast amount of items. Resnick et al. [14] developed the *GroupLens* system for recommending preferred Usenet articles extracted from large amount of news articles, to individual users. GroupLens behave as follows, when it recommends particular user u_a. Before the recommending, each user evaluates of the articles which s/he already read in 5 level scales: 5 (prefer) to 1 (not prefer). The system first finds similar users by computing similarities between u_a and the other users, with the vector operation on the users' evaluations. Then, the system estimates u_a's evaluations to every article i_j which u_a has never read, with weighted sums of the other similar users' evaluation to the article i_j. At last, it recommends the articles that will be likely high evaluated by u_a. Our study provides a specific procedure to apply this algorithm for estimating software development effort. To date, many systems and algorithms have been proposed in the CF researches [3], [8], [16]. It is plausible that these algorithms can estimate development effort as well as user preferences.

CF has challenged for accurately estimating user preferences using defective data that have vast amount of missing values. In a typical Recommender Systems most users do not evaluate more than 1 % of all items since existing items are immensity. (e.g., you can't evaluate all Usenet news in using GroupLens.) Sarwar et al. [16] proposed a CF algorithm to accurately estimate users' preferences when the data are extremely sparsity. This algorithm succeeded on the Amazon.com's website with a famous statement "Customers who bought this book also bought these books". We focus on this feature in CF, which can accurately estimate using defective data. The algorithms can provide improved prediction accuracy using incomplete data when we have instantiated procedure on how to apply CF for estimating software development effort.

3 Estimation Procedure

This section describes our estimation procedure based on CF. In the proposed procedure, we use database in form of $m \times n$ matrix as shown in Fig. 1 where $p_i \in \{p_1, p_2, ..., p_m\}$ denotes i-th project, $m_j \in \{m_1, m_2, ..., m_n\}$ denotes j-th metric, and $v_{i,j} \in \{v_{1,1}, v_{1,2}, ..., v_{m,n}\}$ denotes value of metric m_j observed in project p_i. Note that some elements in the matrix are actually empty (missing values).

	m_1	m_2	…	m_j	…	m_b	…	m_n
p_1	$v_{1,1}$	$v_{1,2}$	…	$v_{1,j}$	…	$v_{1,b}$	…	$v_{1,n}$
p_2	$v_{2,1}$	$v_{2,2}$	…	$v_{2,j}$	…	$v_{2,b}$	…	$v_{2,n}$
…	…	…		…		…		…
p_i	$v_{i,1}$	$v_{i,2}$	…	$v_{i,j}$	…	$v_{i,b}$	…	$v_{i,n}$
…	…	…		…		…		…
p_a	$v_{a,1}$	$v_{a,2}$	…	$v_{a,j}$	…	$v_{a,b}$	…	$v_{a,n}$
…	…	…		…		…		…
p_m	$v_{m,1}$	$v_{m,2}$	…	$v_{m,j}$	…	$v_{m,b}$	…	$v_{m,n}$

Fig. 1. $m \times n$ table used for estimation.

Here we assume a-th project p_a is a target project in which b-th metric $v_{a,b}$ is to be estimated. We denote $\hat{v}_{a,b}$ as an estimated value for $v_{a,b}$. Below describes 3-step procedure to obtain $\hat{v}_{a,b}$.

Step 1 (normalization of metrics): Since each metric has different value range, this first step normalizes values of metrics so that the value range becomes [0, 1]. The normalized value $normalized(v_{i,j})$ of metric $v_{i,j}$ (of project p_i) is calculated by the following equation.

$$normalized(v_{i,j}) = \frac{v_{i,j} - \min(P_j)}{\max(P_j) - \min(P_j)} \quad (1)$$

where P_j denotes a set of projects in which value of metric m_j was observed (collected), $max(P_j)$ and $min(P_j)$ denotes maximum and minimum value in $\{v_{x,j} \mid p_x \in P_j\}$ respectively.

Step 2 (computation of similarity between projects): In this step, similarity $sim(p_a, p_i)$ between the target project p_a and other projects p_i is computed. Many algorithms for computing $sim(p_a, p_i)$ have been proposed in conventional CF researches [3], [8], [14], [16].

We employ a similarity computation algorithm that was proposed in the field of information retrieval to evaluate the similarity between two documents. The similarity is often evaluated by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the two frequency vectors [15]. We can adopt this formalism to compute $sim(p_a, p_i)$, where projects take the role of documents, metrics take the role of words, and values of the metrics take the role of

word frequencies. Formally, we can define the $sim(p_a, p_i)$ between the target project p_a and other projects p_i as:

$$sim(p_a, p_i) = \frac{\sum_{j \in M_a \cap M_i} \left(normalized(v_{a,j}) \times normalized(v_{i,j}) \right)}{\sqrt{\sum_{j \in M_a \cap M_i} \left(normalized(v_{a,j})^2 \right)} \sqrt{\sum_{j \in M_a \cap M_i} \left(normalized(v_{i,j})^2 \right)}} \quad \textbf{(2)}$$

where M_a and M_i denotes a set of metrics observed in project p_a and p_i respectively. The value range of $sim(p_a, p_i)$ is [0, 1]. Note that this computation is suitable for data set containing missing values.

Step 3 (computation of estimation): This step calculates an estimated value $\hat{v}_{a,b}$ of the metric m_b on the target project p_a using $sim(p_a, p_i)$ calculated in previous step. Many algorithms have been proposed to implement this step [3], [8], [14], [16].

We employ weighted sum to compute estimation. The estimated value is computed as the sum of the metrics' values given by the other projects similar to p_a. Each value is weighted by the corresponding the $amplifier(p_a, p_i)$ and the $sim(p_a, p_i)$ between p_a and p_i. Formally, we can define the estimated value $\hat{v}_{a,b}$ as:

$$\hat{v}_{a,b} = \frac{\sum_{i \in k-nearestProjects} \left(v_{i,b} \times amplifier(p_a, p_i) \times sim(p_a, p_i) \right)}{\sum_{i \in k-nearestProjects} sim(p_a, p_i)} \quad \textbf{(3)}$$

where *k-nearestProjects* denotes a set of *k* projects (called *neighborhoods*) that have highest similarity with p_a. The neighborhoods must have m_j as an observed metric. Generally, the neighborhood size *k* affects the estimation accuracy.

To improve accuracy of the estimation, the $amplifier(p_a, p_i)$ calculates an approximate value of the $v_{i,b}$ with comparing the sizes of projects p_a and p_i, i.e. the amplifier indicates what times p_a's value is p_i's value. The *amplifier* derived from the fact that the p_a's value is several times larger (or smaller) than the p_i's value when p_i is a similar to p_a. It's because the similarity is computed by vector operation but not *Euclidean distance*. *Sim*(*pa*, *pi*) is computed by comparing tendencies of the values, whereas *Euclidean distance* is computed by comparing absolute values (cf. [14] for *Euclidean distance*-based similarity) . Formally, we can define the $amplifier(p_a, p_i)$ as:

$$amplifier(p_a, p_i) = \frac{\sum_{j \in M_a \cap M_i} \left(\frac{normalized(v_{a,j})}{normalized(v_{i,j})} \right)}{|M_a \cap M_i|} \quad \textbf{(4)}$$

where $|M_a \cap M_i|$ denotes the number of elements of the product set $M_a \cap M_i$.

4 Case Study

4.1 Goal

The goal of this case study is to evaluate the prediction performance of the proposed method under the situation that available data contain a large amount of missing values. In order to evaluate the prediction performance, we used the absolute and relative error. We compared the proposed method, vs. the stepwise multiple regression models enhanced by the techniques for dealing with missing values.

4.2 Data Source

Table 1 shows the data we used for the case study. The data includes 1081 projects and 14 metrics, were collected from a Japanese software company in a decade. The company has developed some software packages that are the reusable large software components. 36% of all projects in the company tailored them to meet the needs of the individual customers. 13% developed individual software without the software packages. The others (51%) were unknown projects with the missing values.

Table 1 also shows the rates of missing values in each metric. 59.83% of all data were missing originally; however, the rates of missing values were quite different in each metric. In m_3, m_4 and m_{14}, there was no missing value. In m_2 there were few missing values (7.49%), whereas in the other 10 metrics there were at least 70% missing values.

Table 1. Data used in case study

	Metrics	Missing value rate
m_1	Mainframe or not (Mainframe 1, Others 0)	75.76 %
m_2	New development or not (New 1, Others 0)	7.49 %
m_3	Total design cost	0.00 %
m_4	Total coding cost	0.00 %
m_5	Design cost for regular (payroll) staffs of a company	86.68 %
m_6	Design cost for dispatched staffs from other companies	86.68 %
m_7	Design cost for subcontract companies	86.59 %
m_8	Coding cost for regular staffs	86.68 %
m_9	Coding cost for dispatched staffs	86.68 %
m_{10}	Coding cost for subcontract companies	86.59 %
m_{11}	# of faults found in the review of conceptual design	83.53 %
m_{12}	# of faults found in the review of functional design	70.77 %
m_{13}	# of faults found in the review of program design	80.20 %
m_{14}	Testing cost	0.00 %

4.3 Stepwise Multiple Regression Model

To construct a multiple regression model, we used a stepwise method to select a combination of predictor variables that strongly affect the objective variable. The stepwise procedure involves (1) identifying an initial model, (2) repeatedly altering the model at the previous step by adding a predictor variable whose coefficient is considered non-zero ($p < 0.05$), or by removing a variable whose coefficient cannot be considered non-zero ($p < 0.1$), and (3) terminating the search when stepping is no longer possible.

We employed the following three types of MDTs. These techniques are widely used for constructing regression models using data with missing values [10], [12], [19].

Listwise Deletion: This method constructs models with the projects which do not contain any missing values. The data of projects which have some missing values are simply excluded.

Pairwise Deletion: This method uses all available values wherever possible, then, it constructs models using the same method to listwise deletion. For example, all observed values are used regardless of whether the projects have missing other metrics, when it calculates the averages of each metrics or the correlation coefficients between the metrics.

Mean Imputation: This method fills each missing value with the mean of observed values. After all of missing values are filled, it then constructs models using the completed data.

4.4 Evaluation Criteria

In this paper we use the following five evaluation criteria. These criteria are commonly used in estimation methods [6], [11], [19]. In the rest of paper, Y denotes actual (observed) objective variable, $\hat{Y}$ denotes its predicted variable, and "operation" indicates an action to compare Y and $\hat{Y}$. Five criteria are computed after the operation has executed t times.

***Mean Absolute Error* (*MAE*):**

$$MAE = \frac{\sum \left| \hat{Y} - Y \right|}{t} \tag{5}$$

***Variance of Absolute Error* (VAE):**

$$VAE = \frac{\sum \left(\left| \hat{Y} - Y \right| - MAE \right)^2}{t} \tag{6}$$

***Mean Relative Error* (*MRE*):**

$$MRE = \sum \left| \frac{\hat{Y} - Y}{Y} \right| \times \frac{1}{t} \quad (7)$$

***Variance of Relative Error* (*VRE*):**

$$VRE = \sum \left(\left| \frac{\hat{Y} - Y}{Y} \right| - MRE \right)^2 \times \frac{1}{t} \quad (8)$$

Pred25: Ratio of operations whose relative errors are under 25%. Generally, estimation is considered accurate when Pred25 is small.

$$Pred25 = \frac{\sum isAccurate\left(\hat{Y} - Y\right)}{t}$$

$$isAccurate\left(\hat{Y} - Y\right) = \begin{cases} 1 & \left|\hat{Y} - Y\right| < 25 \\ 0 & \left|\hat{Y} - Y\right| \geq 25 \end{cases} \quad (9)$$

4.5 Experimental Procedure

The experimental procedure consists of the following 4 steps.

i. We divided the data into 50-50 two sets randomly: *fit dataset* and *test dataset*. In the case study, we used fit dataset for constructing estimation models, and estimated each testing cost m_{14} on the project in the test dataset. With the same way, we made 10 different pairs of fit and test datasets.

ii. We evaluated performance of CF described in the section 3, using 10 different pairs of fit and test datasets. Every evaluation criterion was measured by the averaged values of 10 results with the different pairs of the datasets, for improving reliability of the experimental results. In order to find an appropriate neighborhood size (the value of k in the equation (3)), we measured MRE of each neighborhood size (from k = 1 up to k = 50) and picked up k having smallest MRE. We then measured MAE, VAE, VRE and Pred25 of CF employing the selected neighborhood size.

iii. We evaluated performance of stepwise multiple regression models with three types of MDTs described in section the 4.3, using 10 different pairs of fit and test datasets. In each MDT, all of the criteria were measured by the average of the results with 10 different pairs of the datasets.

iv. We concluded about which method outperformed the others, with comparing the evaluation criteria resulted by the above ii and iii. And we also considered about the implications and limitation observed from our results.

5 Experimental Results

5.1 Overall Results

Table 2 presents the overall result of our case study. Table 2 includes resulted evaluation criteria with using CF with the neighborhood size 22 and using stepwise multiple regression models with the three types of MDTs. As described in the section 4.4, a lower value indicates better performance than a higher one, in MAE, VAE, MRE and VRE; while, a higher value is better in Pred25. Each evaluation criterion was statistically significant at the 97% or more confidence level, with the t-testing or f-testing.

Our results indicate CF outperformed the others, i.e., it made the most accurate estimations with the lowest variances. All of evaluation criteria consistently indicate CF was the most effective. In our results, listwise deletion performed relatively better performance than the other MDTs, since MAE, MRE and VRE resulted as lower values. MAE, MRE and VRE of mean imputation and pairwise deletion indicate these were ineffective, although these were relatively better in terms of VAE or Pred25.

The detailed results on each method are presented in the following sections.

Table 2. Averaged evaluation criteria resulted with the 10 different experimental dataset

	MAE	VAE	MRE	VRE	Pred25
CF (k = 22)	0.21	2.20	0.82	3.45	36%
Regression (listwise deletion)	0.70	16.45	30.22	287581.18	10%
Regression (pairwise deletion)	52.75	97171.84	6344.27	18937623097.60	12%
Regression (mean imputation)	1.33	5.07	331.69	24208218.49	4%

5.2 CF based Estimation

Fig. 2 presents a line graph of the MREs resulted by applying CF with each neighborhood size (from k = 1 up to k = 50). In the graph, the horizontal axis indicates neighborhood sizes, and the vertical axis indicates MREs corresponding to horizontal axis. This result shows the most accurate estimation was observed at neighborhood size of 22.

Fig. 3 intuitively presents the estimation accuracy of the most effective case of CF. This graph is illustrated as a double logarithmic chart for ease of understanding. The horizontal axis in the graph, indicates the estimated testing costs and vertical axis indicates the actual costs corresponding to horizontal axis; i.e., the points are plotted

near by the line of y = x if the estimations were correct, while, distant point from the line indicates incorrect estimation and a more distant point from the line presents more incorrect estimation than the nearer one. This graph suggests there were more projects having lower testing costs than higher ones. And also shows CF accurately estimated the testing costs regardless of whether the actual costs were high or low.

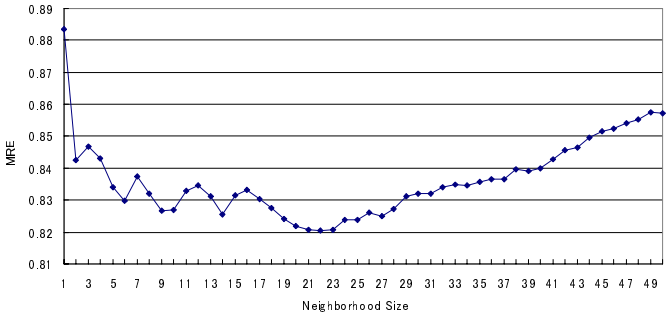

Fig. 2. Accuracy of the CF in each neighborhood size (from k = 1 up to k = 50)

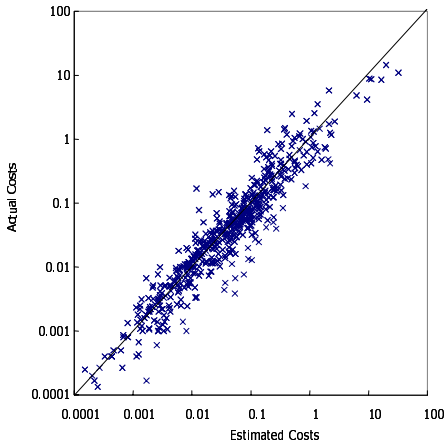

Fig. 3. Relationship between actual and estimated costs using CF (k = 22)

5.3 Stepwise Multiple Regression Model

Fig. 4, Fig. 5 and Fig. 6 present the estimation accuracies of the most effective case of each stepwise multiple regression model with listwise deletion, pairwise deletion and mean imputation. These graphs are illustrated like Fig. 3 described in the section 5.2. The graphs suggest there were more projects having lower testing costs than higher ones. And also show accuracies of the estimation were lower when the actual costs

were lower. Every regression model estimated the lower actual costs incorrectly as bigger ones.

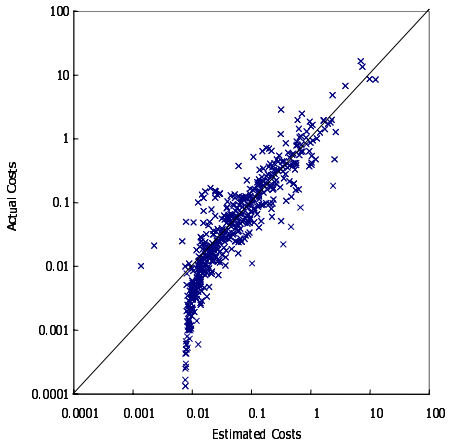

Fig. 4. Relationship between actual and estimated costs using stepwise multiple regression model with listwise deletion

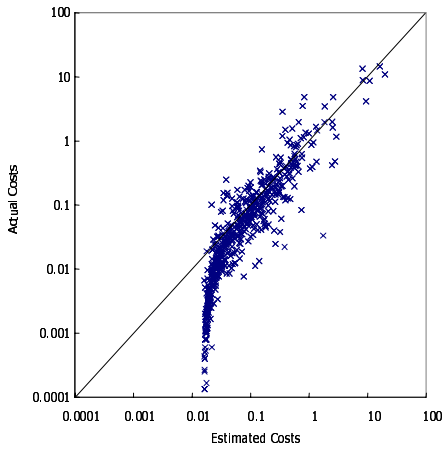

Fig. 5. Relationship between actual and estimated costs using stepwise multiple regression model with pairwise deletion

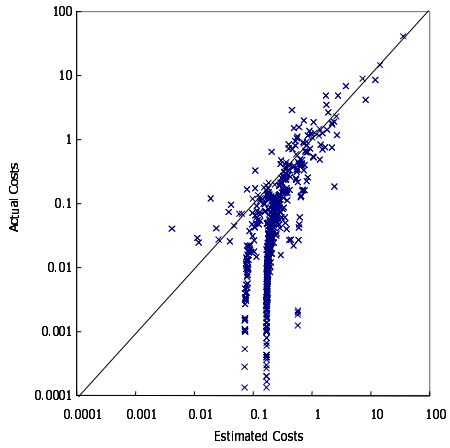

Fig. 6. Relationship between actual and estimated costs using stepwise multiple regression model with mean imputation

5.4 Discussion

Although our results indicate that the proposed method greatly outperformed stepwise regression models with the three types of MDTs, the MRE of CF (0.82) suggests the proposed method needs further improvements. One of the ways for improving is to develop various alternative CF algorithms for computing similarities and estimations, which can be used instead of the equation (2) and (3). Especially, variations of the amplifier in the equation (3) have the possibility for improving accuracies of the estimations. Another way for improving the performances is to develop some methods to select an appropriate combination of predictor metrics that strongly affect the objective metrics, like the stepwise method described in the section 4.3.

One of the limitations of our study, we evaluated just only three types of MDTs for dealing with missing values: listwise deletion, pairwise deletion and mean imputation; however, there still exists more MDTs. For instance, Strike et al. [19] reported that *hot-deck imputation* provided the best performance in his simulation. We must compare the proposed method and the regression models using the other MDTs including hot-deck imputation in order to evaluate these performances.

Furthermore, we evaluated using just only one dataset; however there are many conditions (i.e., various amounts and distributions of the missing values) in the datasets collected by the actual industrial organizations. We must encourage the results of this case study on alternative datasets and conduct experimental simulation with various conditions of data, for improving reliability of the study.

6 Conclusions

This paper proposed a method for estimating software development effort based on collaborative filtering. The proposed method showed better performance than the conventional regression method when the data had substantial missing values.

In the future, we are going to develop CF algorithms for computing similarities and estimations, to improve estimation performances. Furthermore, we will conduct an experiment to compare the proposed method with other MDTs including hot-deck imputation method. we also will encourage the replication of this case study on alternative datasets and conduct experimental simulation with various conditions of data, for improving reliability of the study.

References

1. Albrecht, A., Gaffney, J.: Software Function, Source Lines of Code, and Development Effort Prediction. IEEE Trans. on Software Eng., vol.9, no.6, pp.83-92 (1979)
2. Boehm, B.W.: Software Engineering Economics. IEEE Trans. on Software Eng., vol.10, no.1, 4-21 (1984)
3. Breese, J. S., Heckerman, D., and Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence*, pp.43-52 (1998)
4. Briand, L., Basili, V., and Thomas, W.: A Pattern Recognition Approach for Software Engineering Data Analysis. IEEE Trans. on Software Eng., vol.18, no.11, pp.931-942 (1992)
5. Briand, L., El Eman, K., and Wieczorek, I.: Explaining the Cost of European Space and Military Projects. In Proc. Int'l Conf. Software Eng., vol.1, no.1, pp.61-88 (1996)
6. Conte, S.D., Dunsmore, H.E., and Shen, V.Y.: Software Engineering Metrics and Models. The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California (1986)
7. Finnie, G., and Wittig, G.: A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. Journal of Systems and Software, vol.39, pp.281-289, 1997.
8. Goldberg, D., Nichols, D., Oki, B.M., and Terry, D.: Using Collaborative Filtering to Weave an Information Tapestry. *Comm. of the ACM*, vol.35, no.12, pp.61-70 (1992)
9. Gray, A., and MacDonnell, D.: A Comparison of Techniques for Developing Predictive Models of Software Metrics. Information and Software Technology, vol.3, pp.425-437 (1997)
10. Little, R., and Rubin, D.: Statistical Analysis with Missing Data. John Wiley & Sons, Inc. (1987)
11. Khoshgoftaar, T.M., Munson, J.C., Bhattacharya, B.B., and Richardson, G.D.: Predictive Modeling Techniques of Software Quality from Software Measures. IEEE Trans. on Software Eng., vol.18, no.1, 979-987 (1992)

12. Kromrey, J., and Hines, C.: Nonrandomly Missing Data in Multiple Regression: An Empirical Comparison of Common Missing-Data Treatments. Educational and Psychological Measurement, vo.54, no.3, pp.573-593 (1994)
13. Rahhal, S., and Madhavji, N.: An Effort Estimation Model for Implementing ISO 9001. In *Proc. of the 2nd IEEE Int'l Software Eng. Standards Symp.*, pp.278-286 (1995)
14. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. ACM Conf. on Computer Supported Cooperative Work (CSCW'94)*, Chapel Hill, North Carolina, United States, 175-186 (1994)
15. Salton, G., and McGill, M.: Introduction to Modern Information Retrieval, McGraw-Hill, New York (1983).
16. Sarwar, B.M., Karypis, G., Konstan, J.A., and Riedl, J.: Item-Based Collaborative Filtering Recommendation Algorithms, In *Proc. 10th International World Wide Web Conference (WWW10)*, Hong Kong, 285-295 (2001)
17. Shepperd, M., and Schofield, C.: Estimating Software Project Effort Using Analogies. IEEE Trans. on Software Eng., vol.23, no.12, pp.76-743 (1997)
18. Srinivasan, K., and Fisher, D.: Machine Learning Approaches to Estimating Software Development Effort. IEEE Trans. on Software Eng., vol.21, no.2, pp.126-137 (1995)
19. Strike, K., El Eman, K., and Madhavji, N.: Software Cost Estimation with Incomplete Data. IEEE Trans. on Software Eng., vol.27, no.10, pp.890-908 (2001)
20. Walston, C., and Felix, C.: A Method of Programming Measurement and Estimation. IBM Systems Journal, vol.1, pp.54-73, 1977.