Augmenting Window Contents with Transfer Learning for Effort Estimation

Sousuke Amasaki[a]

[a]*Okayama Prefectural University, 111 Kuboki, Soja, 719-1197, Japan*

Abstract

BACKGROUND: Some studies showed filtering out old completed projects with a window was effective for preparing a training dataset of an effort estimation model. Other studies showed selecting completed projects similar to a target project was also effective. The application of the similarity-based selection after the windowing approach was failed to synthesize their effects. The shortage of similar projects in the windowed pool was a potential cause of the failure. AIMS: To examine whether augmenting the window pool is effective to improve the estimation accuracy. METHOD: The moving windows approach was used for preparing a window pool. The similarity-based selection was applied to augment the pool. The selection assumes that projects in the pool form a set of virtual target projects. Old projects outside the pool were assumed to form a set of cross-company projects to be selected. The empirical study with a single-company ISBSG data was conducted to evaluate the effect. RESULTS: A positive synergistic effect was observed. The augmented window could synthesize the windowing approach and the similarity-based selection. It could also be combined with the similarity-based selection without performance degradation. CONCLUSIONS: Practitioners should consider adding projects similar to recently completed projects when effort estimation is based on historical data.

Keywords

effort estimation, moving windows, augmenting windows

1. Introduction

The success of software projects relies on many factors. The accuracy of software effort estimation is an serious influential factor at early project phase. Overestimation and underestimation have caused serious consequences for decades. Researchers have studied data-driven software effort estimation models while experts' judgment is still a primary choice in actual. The accuracy of the software effort estimation models is considered insufficient among not a few managers.

Software effort estimation models are affected by the adequacy of historical data from past projects. For instance, an organization's productivity is not stationary nor monotonic due to changes in the environment and the organization itself. Inaccurate effort estimation models would be obtained with the historical data that might not reflect the present productivity. A key to accurate software effort estimation is to prepare historical data that reflect the characteristics of a target project to be estimated.

A past study [1] examined two filtering techniques, namely, chronological filtering and relevancy filtering. The chronological filtering [2] removes too old project data. The relevancy filtering [3] removes dissimilar project data regarding metrics used for estimation. The study found that the combination of those techniques might be worse than the independent application.

The negative synergistic effect can be reasoned, at least, in two aspects. First, the relevancy filtering was applied after applying the chronological filtering. The chronological filtering does not care about feature variables and may select a subset that does not hold enough projects similar to a target project. It would be better to augment the subset with old but resemble projects using the relevancy filtering. Second, the simple average and median were used as effort estimation models as discussed in [1]. The simple models only used the effort variable for estimation and were insensitive to the change in the distribution of feature variables after the relevancy filtering.

This paper proposed an augmented chronological filtering based on the chronological filtering and the relevancy filtering. Its effects were investigated with a software effort estimation model using feature variables, in addition to the simple average and median models. The augmented filtering was also evaluated as alternative chronological filtering in the past combination method. The following questions were asked:

RQ1: Does augmenting moving windows with a relevancy filtering affects the estimation accuracy?

RQ2: Does using the augmentation as a chronological filtering affect the estimation accuracy of the past combination method?

QuASoQ 2020: 8th International Workshop on Quantitative Approaches to Software Quality, December 01, 2020, Singapore
EMAIL: amasaki@cse.oka-pu.ac.jp (S. Amasaki)
ORCID: 0000-0001-8763-3457 (S. Amasaki)

2. Related Work

2.1. Chronological Filtering

Although research in software effort estimation models has a long history, relatively few studies have taken into consideration the chronological order of projects. Therefore, chronological filtering has not been studied well compared with other topics in effort estimation.

To our knowledge, Kitchenham et al. [4] were first to suggest the use of chronological filtering. They built four linear regression models with four subsets, each of which comprised projects from different ranges of time duration. As the coefficients of the models were different from each other, they allowed to drop out older project data. Lokan and Mendes [2] were the first to study the effect of using moving windows in detail. They used linear regression (LR) models and a single-company dataset from the ISBSG repository. Training sets were defined to be the N most recently completed projects. They found that the use of a window could affect accuracy significantly; predictive accuracy was better with larger windows; some window sizes were particularly effective. Amasaki and Lokan also investigated the effect of using moving windows with Estimation by Analogy [5] and CART [6]. They found that moving windows could improve the estimation accuracy, but the effect was different than with LR.

Recent studies showed the effect and its extent could be affected by windowing policies [7] and software organizations [8]. Lokan and Mendes [7] investigated the effect on accuracy when using moving windows of various ranges of time duration to form training sets on which to base effort estimates. They also showed that the use of windows based on duration could affect the accuracy of estimates, but to a lesser extent than windows based on a fixed number of projects [8].

2.2. Relevancy Filtering

Relevancy filtering is a type of transfer learning approach. While many filtering approaches have been proposed for cross-project defect prediction (e.g., [9]), a few studies on cross-company effort estimation have evaluated the effects of relevancy filtering approaches.

Turhan and Mendes [3] applied brings a so-called NN-filter [10] to cross-company effort estimation of web projects. They showed that an estimation model based on raw cross-company data was worse than that based on within-company data but was improved as comparable one by using the NN-filter. Kocaguneli et al. [11, 12, 13] also introduced a transfer learning approach called TEAK for improving cross-company effort estimation. They applied it to transfer old project to a new project and found that TEAK was effective not only for cross-company effort estimation but also for cross-time effort estimation [14].

NN-filter is based on a nearest neighbor algorithm. In that sense, a study by Amasaki and Lokan [5] can be considered an evaluation study of the combination of the relevancy filtering and the chronological filtering. In that study, the combination worked well to improve estimation accuracy for a narrow range of window sizes. While that study used a wrapper approach for feature selection and logarithmic transformation in addition to the nearest neighbor algorithm, our study aims to explore the effects of the combination without such complicated factors. For that purpose, we adopted two simple estimation techniques that were not adopted in [5], described in the next section.

3. Methodology

3.1. Effort Estimation Techniques

In [1], average and median were used as software effort estimation models. The average was adopted because it uses the whole training set and is sensitive to the distribution of effort values in the training set. The median was adopted because it is robust to the distribution and contrasts with the average. These models estimate efforts without adjustments based on feature variables of projects.

To examine the difference in the use of feature variables in software effort estimation, we also adopted Lasso [15] for our experiment. Lasso is a kind of penalized linear regression models. Past studies on the chronological filtering used Lasso and showed that the chronological filtering was effective with it. Our experiment used `LassoLarsIC` of scikit-learn library.

3.2. Chronological Filtering

This study adopted fixed-size moving windows [2] and fixed-duration moving windows [8]. The latest N finished projects were selected as a training set by the fixed-size moving windows. The fixed-duration moving windows selected the latest projects finished within N months. As N influences on the effectiveness of moving windows, we explored various values as well as past studies.

3.3. Relevancy Filtering

This study used a nearest neighbor algorithm as a relevancy filtering approach. It is also called NN-filter [10].

The procedure of NN-filter is as follows:

1. Select k closest instances of history data to each instance of target project data in terms of unweighted Euclidean distance.
2. Combine the selected instances without duplication.

Note that each feature of project data was normalized with min-max normalization before the distance calculation.

As the synergistic effect could be observed with effective filtering, the relevancy filtering had to be configured as effective. For average and mean effort estimation models, we roughly fixed $k = 3$, which is the smallest number which can make average and median estimations give distinct efforts. For lasso, we roughly fixed $k = 10$, half of the minimum of the window sizes we explored. In general, increasing k would lead to worse estimation if NN-filter works well. Hence, these values could not be the best but were expected more reasonable than larger ks.

3.4. Augmentation

The augmentation adds old projects selected by the relevancy filtering into a subset obtained by the chronological filtering as follows:

1. Recently completed projects are selected with the moving windows approach.
2. NN-filter is applied to select projects from the remained old projects. The most similar project to each project of the recently completed projects is selected. The set of selected projects has no duplicate.
3. The selected projects and the recently completed projects are combined.
4. The combined projects are used to train a software effort estimation model.

Note that NN-filter uses the effort variable in addition to feature variables. As efforts of the past projects are known, it is possible to use the effort variable in the augmentation process.

The augmentation shares the same assumption as the chronological filtering that the recently completed projects resemble a target project to be estimated. Results of NN-filter are also expected to pretend to be as fresh as the recently completed projects. Therefore, the selected projects are considered to keep the similarity to the target project.

3.5. Combination

The combination of the chronological filtering and the relevancy filtering was investigated in [1]. The chronological filtering and the relevancy filtering were combined as follows:

1. Recently completed projects are selected with the moving windows approach. The remained old projects are discarded.
2. NN-filter is applied to select projects from the recently completed projects. The selected projects resemble a target project to be estimated.
3. The selected projects are used to train a software effort estimation model.

The combination method was found less effective than each of the filtering methods in [1] with mean and median models.

The augmentation can be considered a variation of moving windows approach while it is a way to combine moving windows and NN-filter. In this paper, this combination was also examined using a subset obtained by the augmentation. As the augmentation has more projects, NN-filter might bring better neighbors from an augmented subset.

3.6. Experiment procedure

As the chronological filtering relies on the time proximity, our experiment needs to assume a situation that a development organization needs to respond to continuously coming new projects. The size of windows influences on where our experiment starts. As same as the past studies, our experiment with a specific window size was conducted as follows:

1. Sort all projects by starting date.
2. For a given window size N, find the earliest project p_0 for which at least $N + 1$ projects were completed prior to the start of p_0 (projects from p_0 onwards are the ones whose training set is affected by using a window, so they form the set of evaluation projects for this window size. For example, with a window of 20 projects, at least 21 projects must have finished for the window to differ from the growing portfolio.)
3. For every project p_i in chronological sequence, starting from p_0, form a training set using moving windows and the growing portfolio (all completed projects).
    - For no filtering, the training set is all projects that finished before p_i started.

Table 1
Summary statistics for ratio-scaled variables in data from single ISBSG organization

Variable	Min	Mean	Median	Max	StDev
Size	10	496	266	6294	699
Effort	62	4553	2408	57749	6212
PDR	0.53	16.47	8.75	387.10	31.42

- For fixed-size moving windows, the training set is the N most recent projects that finished before p_i started. If multiple projects finished on the same date, all of them are included.
- For fixed-duration, the training set is the most recent projects whose whole life cycle had fallen within a window of D months prior to the start of p_i.

4. Estimate an effort of a target project based on past project data.
    - For no filtering, the training set from the previous step is used.
    - For relevancy filtering, a subset selected by a nearest neighbor from the training set is used.
    - For the augment method, an augmented set of the training set with the projects not selected in the previous step is used.
5. Evaluate the estimation results.

This study used the single-company subset of the ISBSG dataset that was analyzed in [2, 7, 8, 5, 6, 16]. Table 1 shows summary statistics. We explored window sizes from 20 to 120 projects for the size-based moving windows and from 12 to 84 months for the duration-based moving windows as well as the past study [17]. No filtering, called the growing portfolio in past studies, was used as a baseline for comparing the filtering methods.

3.7. Performance Measures

The accuracy statistics that we used to evaluate the effort estimation models are based on the difference between estimated effort and actual effort. We used Mean Absolute Error (MAE), which is widely used to evaluate the accuracy of effort estimation models, as it is an unbiased measure that favours neither under- nor over-estimates.

We concentrate first on the statistical significance of differences in accuracy that arise from using the filtering approaches. To test for statistically significant differences between accuracy measures, we use the two-sided Wilcoxon signed-rank test (`wilcoxon` function of the `scipy` package for Python) and set the statistical significance level at $\alpha = 0.05$. The setting of this study is a typical multiple testing, and the p-values of the tests must be controlled. Bonferroni correction is a popular method for this purpose. However, the adoption of this simple correction results in the lack of statistical power, especially for not large effects. We thus controlled the false discovery rate (FDR) of multiple testing [18] with the "`multipletests`" function of the `statsmodels` package in Python. FDR is a ratio of the number of falsely rejected null hypotheses to the number of rejected null hypotheses.

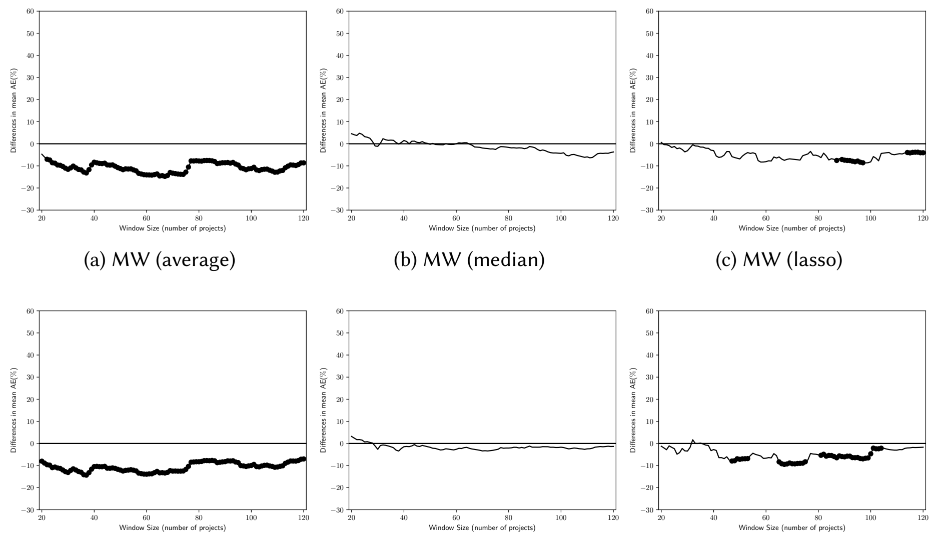
4. Results and Discussion

4.1. Comparisons between Moving Windows and Augmentation

Figure 1 has 6 plots showing the difference in mean absolute error against window sizes using the fixed-size moving windows (baseline) and the augmentation with it. The x-axis of each figure is the size of the window, and the y-axis is the subtraction of the accuracy measure value with the growing approach from that with the moving windows at the given x-value. The moving windows and the augmentation with it were advantageous where the line is below 0. Circle points mean a statistically significant difference, with the moving windows or the augmentation with it, being better than the growing portfolio. At these points, the corresponding FDR-controlled p-value was below $\alpha = 0.05$.

Figure. 1 revealed the effect of using the fixed-size moving windows and the augmentation, compared to always using the growing portfolio as follows:

- With average effort estimation, statistically significant differences were found for almost all window sizes. The augmentation did not bring clear changes except for small window sizes, where additional statistically significant differences were found.
- With median effort estimation, no statistically significant difference was found for all window sizes. The augmentation improved the performance a bit for smaller window sizes but worsened it a bit for larger window sizes. The ef-



(a) MW (average) (b) MW (median) (c) MW (lasso)

(d) Augmentation(average) (e) Augmentation (median) (f) Augmentation (lasso)

Figure 1: The difference in mean absolute error against moving windows (growing portfolio vs. fixed-size MW and Augmentation)

fects never caused a statistically significant difference.

- With lasso, statistically significant differences were found when window size is between 85 and 95 or is more than 110. The augmentation made the advantages in other window sizes statistically significant. The significant differences in larger window sizes disappeared instead. Note that lasso was more accurate than the others even when used with the growing portfolio.

These observations suggested that the augmentation could bring a positive synergistic effect on the estimation accuracy when the augmentation was applied to fixed-size windows with average or lasso.

Figure 2 plotted the same comparisons but using the fixed-duration moving windows. In the figure, square points mean a statistically significant difference, with the fixed-duration moving windows being worse than the growing portfolio. These figures revealed the effects of the fixed-duration moving windows and the augmentation with it, compared to always using the growing portfolio as follows:

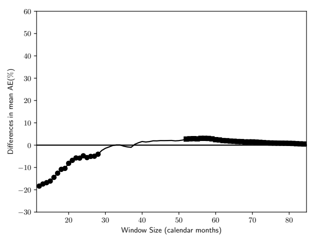
- With average effort estimation, the effective window range was between 20 months and less than 30 months. The growing portfolio got advantageous for more than 53 months. The augmentation extended the advantageous range to more than 40 months. The growing portfolio was no longer advantageous for larger window sizes.
- With median effort estimation, the effective window range was more than 60 months. Disadvantageous window sizes are between 55 months and 60 months. The augmentation made the statistically significant differences disappeared.
- With lasso, there was no significant difference. There was no clear advantage nor disadvantage. The augmentation made no statistically significant difference while the difference got closer a bit.

These observations suggested that the augmentation could improve the estimation accuracy when the augmentation was applied to fixed-duration windows with average effort estimation.

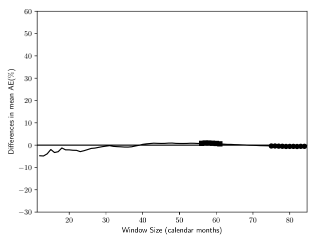
The answer to RQ1 is *yes*: Augmenting moving windows with a relevancy filtering was useful. It did not cause an apparent negative synergistic effect, at least. It sometimes made positive synergistic effects.

4.2. Evaluation of Combination of Augmented MW and NN-filter

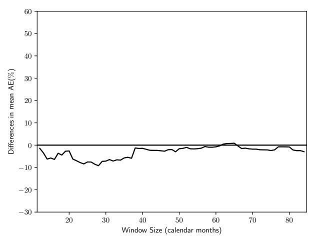
The combination of the augmented moving windows and the NN-filter was evaluated under the same situations. The number of neighbors was set to 3 for av-



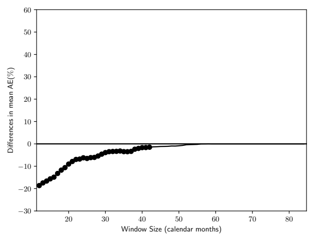
(a) MW (average)



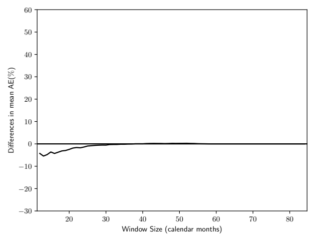
(b) MW (median)



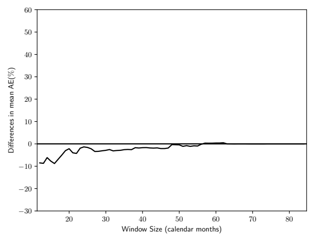
(c) MW (lasso)



(d) Augmentation(average)

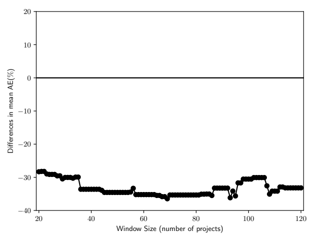


(e) Augmentation (median)

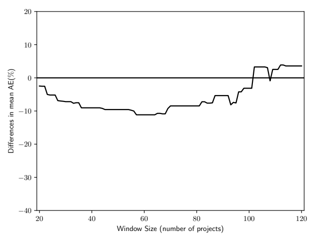


(f) Augmentation (lasso)

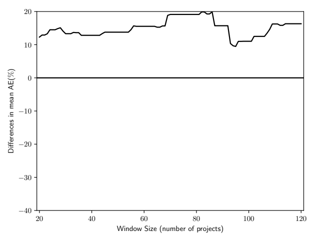
Figure 2: The difference in mean absolute error against moving windows (growing portfolio vs. fixed-duration MW and Augmentation)



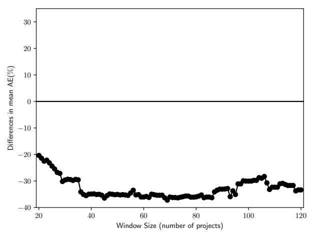
(a) NN (average)



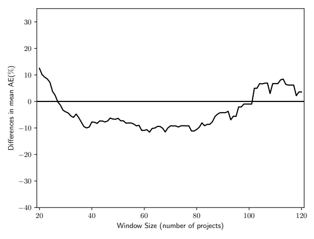
(b) NN (median)



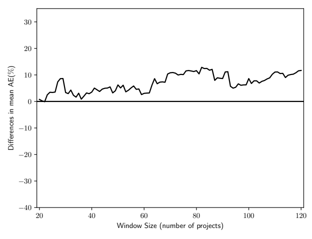
(c) NN (lasso)



(d) AG + NN (average)



(e) AG + NN (median)



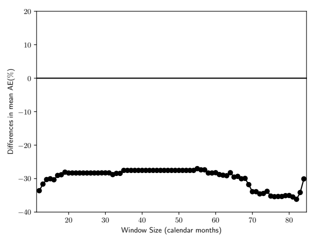
(f) AG + NN (lasso)

Figure 3: The difference in mean absolute error against moving windows (growing portfolio vs. fixed-size MW + Augmentation + NN-filter)

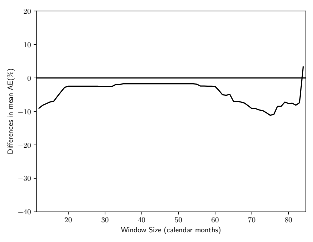
erage and median effort estimation models and 10 for lasso because lasso models, as described in Section 3.3.

Figure 3 has 6 plots showing the difference in mean absolute error against fixed-size window sizes using the NN-filter and using the combination of the augmented windows and the NN-filter. These figures revealed the effects of using the NN-filter and the augmented moving windows with it, compared to always using the growing portfolio as follows:

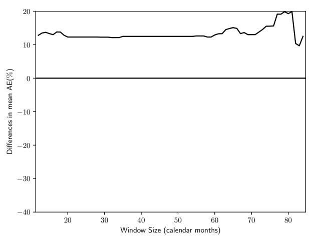
- With average effort estimation, the NN-filter made statistically significant differences for almost all window sizes. Combining the augmented moving windows with the NN-filter made no clear change except for small window sizes, where the



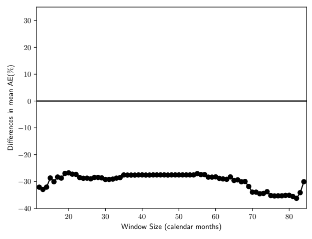
(a) NN (average)



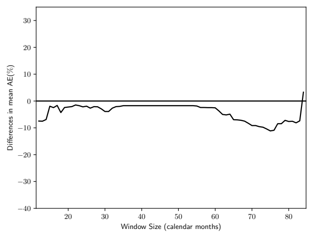
(b) NN (median)



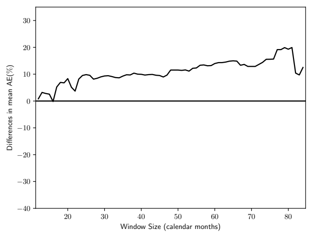
(c) NN (lasso)



(d) AG + NN (average)



(e) AG + NN (median)



(f) AG + NN (lasso)

Figure 4: The difference in mean absolute error against moving windows (growing portfolio vs. fixed-duration MW + Augmentation + NN-filter)

differences got smaller from about -30% to -20%. The significance of the differences was retained, though.

- With median effort estimation, the NN-filter made no clear change while it caused positive effects as depicted by the line running below the zero line for a wide window range. Combining the augmented moving windows with the NN-filter made no clear change except for small window sizes. No statistically significant difference appeared.
- With lasso, the NN-filter made no statistically significant change though it worsened the performance. Combining the augmented moving windows with the NN-filter mitigated the degradation. Note that the augmentation made the significant improvement as shown in Fig. 1(f). The NN-filter canceled the improvement.

In [1], the combination of the moving windows and NN-filter caused a negative synergistic effect. For example, less than half of the window sizes could achieve the improvement of -30% or more where mean effort estimation was applied. The augmentation made the performance improvement of -30% or more for more than a half of the range as shown in Fig. 3(d). Therefore, these observations suggested that the augmented moving windows did not result in a negative synergistic effect caused by the combination of fixed-size moving windows and NN-filter.

Figure 4 plotted the same comparison but using the fixed-duration moving windows. These figures revealed the effects of using the NN-filter and the augmented moving windows with it, compared to always using the growing portfolio as follows:

- With average effort estimation, NN-filter made statistically significant differences for almost all window sizes. Combining the augmented moving windows with the NN-filter made no clear change.
- With median effort estimation, NN-filter made no clear change while it caused positive effects as depicted by the line running below the zero line for a wide window range. Combining the augmented moving windows with the NN-filter made no clear change.
- With lasso, NN-filter made no statistically significant change though it worsened the performance. Combining the augmented moving windows with the NN-filter mitigated the degradation by NN-filter.

Therefore, these observations suggested that the augmentation did not result in a negative synergistic effect caused by the combination of fixed-size moving windows and NN-filter. Rather the degradation by NN-filter could be mitigated.

The answer to RQ2 is as follows: The combination of the augmented chronological filtering and the relevancy filter did not bring a negative synergistic effect except for small window sizes. Rather, the negative effect caused by the relevancy filtering was mitigated by the augmentation.

5. Conclusion

We explored the effects of the augmentation and its combination with a relevancy filtering for effort estimation. We confirmed the augmentation was a useful way to bring a positive synergistic effect of the chronological filtering and the relevancy filtering. Combining the augmented windows with the relevancy filtering, as well as in [1] also diminished the negative synergistic effect caused by the combination of the moving windows and NN-filter found in a past study. We thus concluded that the augmentation can be a good way to combine the two filtering approaches and also a good extension of the moving windows, which can be safely combined with the relevancy filtering.

Further investigation considering other transfer learning approaches is in future work. The NN-filter used for augmentation is a type of transfer learning, it is interesting to examine the effects of other approaches such as [14] for augmentation. Some transfer learning approaches for cross-project defect prediction [19] can also be applied. The threat to external validity can be mitigated with additional project data.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant #18K11246.

References

[1] S. Amasaki, Exploring Preference of Chronological and Relevancy Filtering in Effort Estimation, in: Proc. of Profes 2019, Springer, 2019, pp. 247–262.

[2] C. Lokan, E. Mendes, Applying moving windows to software effort estimation, in: Proc. of ESEM 2009, 2009, pp. 111–122.

[3] B. Turhan, E. Mendes, A Comparison of Cross-Versus Single-Company Effort Prediction Models for Web Projects, in: Proc. of SEAA, IEEE, 2014, pp. 285–292.

[4] B. Kitchenham, S. Lawrence Pfleeger, B. McColl, S. Eagan, An empirical study of maintenance and development estimation accuracy, The Journal of Systems & Software 64 (2002) 57–77.

[5] S. Amasaki, C. Lokan, The Effects of Moving Windows to Software Estimation: Comparative Study on Linear Regression and Estimation by Analogy, in: Proc. of IWSM-MENSURA 2012, IEEE, 2012, pp. 23–32.

[6] S. Amasaki, C. Lokan, The Effect of Moving Windows on Software Effort Estimation: Comparative Study with CART, in: Proc. of IWESEP 2014, IEEE, 2014, pp. 1–6.

[7] C. Lokan, E. Mendes, Investigating the Use of Duration-Based Moving Windows to Improve Software Effort Prediction, in: Proc. of APSEC 2012, 2012, pp. 818–827.

[8] C. Lokan, E. Mendes, Investigating the use of duration-based moving windows to improve software effort prediction: A replicated study, Inf. Softw. Technol. 56 (2014) 1063–1075.

[9] S. Herbold, CrossPare: A tool for benchmarking cross-project defect predictions, in: Proc. of 30th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), IEEE, 2016, pp. 90–95.

[10] B. Turhan, T. Menzies, A. B. Bener, J. Di Stefano, On the relative value of cross-company and within-company data for defect prediction, Empirical Software Engineering 14 (2009) 540–578.

[11] E. Kocaguneli, T. Menzies, How to Find Relevant Data for Effort Estimation?, in: Proc. of ESEM, IEEE, 2011, pp. 255–264.

[12] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, J. W. Keung, When to use data from other projects for effort estimation, in: Proc. of ASE, ACM, 2010, pp. 321–324.

[13] E. Kocaguneli, T. Menzies, A. B. Bener, J. W. Keung, Exploiting the Essential Assumptions of Analogy-Based Effort Estimation, IEEE Transactions on Software Engineering 38 (2012) 425–438.

[14] E. Kocaguneli, T. Menzies, E. Mendes, Transfer learning in effort estimation, Empirical Software Engineering 20 (2015) 813–843.

[15] R. Tibshirani, Regression shrinkage and selection via the lasso, J. Roy. Statist. Soc. Ser. B (1996) 267–288.

[16] S. Amasaki, C. Lokan, Evaluation of Moving Window Policies with CART, in: Proc. of IWESEP 2016, IEEE, 2016, pp. 24–29.

[17] S. Amasaki, C. Lokan, A Replication of Comparative Study of Moving Windows on Linear Regression and Estimation by Analogy, in: Proc. of PROMISE, ACM Press, 2015, pp. 1–10.

[18] Y. Benjamini, D. Yekutieli, The control of the false

discovery rate in multiple testing under dependency, Annals of statistics 29 (2001) 1165–1188.

[19] S. Herbold, Training data selection for cross-project defect prediction, in: Proc. of PROMISE '13, ACM, 2013, pp. 6:1–6:10.