An Investigation of Dycom's Sensitivity to Different Cross-Company Splits

Leandro L. Minku
Department of Informatics, University of Leicester
University Road
Leicester LE1 7RH, UK
leandro.minku@leicester.ac.uk

KEYWORDS

Software effort estimation, cross-company learning, concept drift, online learning, ensembles

1 INTRODUCTION

The approach Dycom managed to drastically reduce the number of Within-Company (WC) projects used for training while maintaining or slightly improving predictive performance in comparison with WC models [10]. It is the only Cross-Company (CC) approach able achieve that while treating the online learning nature of the SEE problem, i.e., the fact that new projects arrive over time and that SEE models must be able to adapt to any changes that could affect the effort required to develop software projects.

In order to use CC projects for SEE, Dycom splits the CC projects into different subsets according to their productivity. Specifically, pre-defined productivity thresholds are used to decide the number of CC subsets and their composing projects. As each subset contains projects within the same productivity range, each subset is expected to contain projects that are more homogeneous to each other. These subsets are thus used to create different CC SEE models. Mapping functions are then learned to dynamically map the estimations given by the CC models to estimations reflecting the current context of a given company. These mapping functions are learned by using a small number of WC projects received over time.

However, it is unclear how the CC splits affect Dycom's predictive performance. In order to better inform the use of Dycom, it is important to know how sensitive it is to different choices of CC splits. This report explains experiments done to investigate the impact of the CC splits on Dycom's performance. It answers the following research question:

RQ1 – What is the impact of the choice of CC splits on Dycom's predictive performance?

2 DATABASES

The databases used in the experiments for the analysis are the same as the ones used in Dycom's original paper [10]. Part of their descriptions is transcribed here to make this paper more self-contained. The key difference between the use of these databases in Dycom's original [10] and current paper is that the former fixed the CC subsets based on specific thresholds. The current paper will investigate different CC splits, as explained in sections **??** and 3.

Five different databases were used: KitchenMax, CocNasaCoc81, ISBSG2000, ISBSG2001 and ISBSG. These include both data sets derived from the former PROMISE Repository [1] (now SEACRAFT Repository [2]) and the ISBSG Repository [5]. Each database contains a WC data set and a CC data set.

2.1 KitchenMax

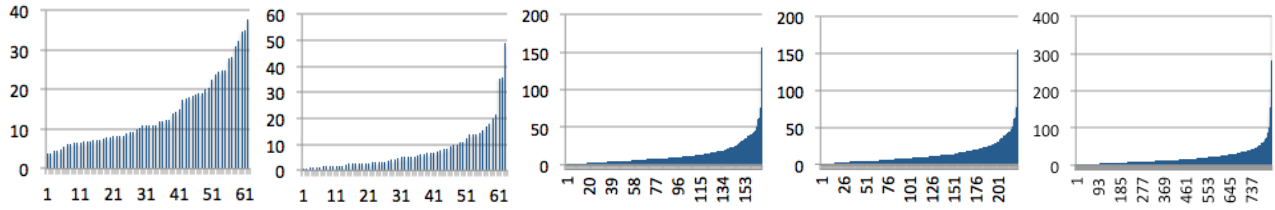
The database KitchenMax is composed of Kitchenham and Maxwell, which are two SEE data sets available from the PROMISE Repository. Kitchenham's detailed description can be found in [6]. It comprises 145 maintenance and development projects undertaken between 1994 and 1998 by a single software development company. Maxwell's detailed description can be found in [12]. It contains 62 projects from one of the biggest commercial banks in Finland, covering the years 1985 to 1993 and both in-house and outsourced development. In order to make these data sets compatible, a single input attribute (functional size) was used. Still, Maxwell uses functional size, whereas Kitchenham uses adjusted functional size. An appropriate mapping function should be able to overcome this problem. There were no functional size attribute values missing. The output attribute is the effort in person-hours.

Kitchenham was considered as the WC data, and was sorted according to the actual start date plus the duration. This sorting corresponds to the exact completion order of the projects. Maxwell's projects were considered as the CC projects. Figure 1a shows the productivity of the CC projects in terms of effort divided by size, which is used by the original Dycom to create the CC subsets.

2.2 CocNasaCoc81

The database CocNasaCoc81 is composed of Cocomo Nasa and Cocomo 81, which are two SEE data sets available from the PROMISE Repository. Cocomo Nasa contains 60 Nasa projects from 1980s-1990s and Cocomo 81 consists of the 63 projects analysed by Boehm to develop the software cost estimation model COCOMO [3] first published in 1981. Both data sets contain 16 input attributes (15 cost drivers [3] and number of lines of code) and one output attribute (software effort in person-months). Cocomo 81 contains an additional input attribute (development type) not present in Cocomo Nasa, which was thus removed. These data sets contain no missing values.

Cocomo Nasa's projects were considered as the WC projects and Cocomo 81's projects were considered as the CC projects. The productivity of the CC projects is shown in figure 1b. Cocomo Nasa provides no information on whether the projects are sorted in chronological order. The original order of the Cocomo Nasa projects was preserved in order to simulate the WC projects' chronology. Even though this may not be the true chronological order, it is still useful to evaluate whether approaches are able to make use of mapped CC models when/if they are beneficial.



(a) KitchenMax (b) CocNasaCoc81 (c) ISBSG2000 (d) ISBSG2001 (e) ISBSG

Figure 1: CC projects' productivities (effort divided by size). Projects are sorted from the most productive to the least productive.

2.3 ISBSG Databases

Three SEE databases were derived from ISBSG Release 10, which contains software project information from several companies. Information on which projects belong to a single company for composing a WC data set have been provided to us upon request. The databases are:

- ISBSG2000 – 119 WC projects implemented after the year 2000 and 168 CC projects implemented up to the end of year 2000.
- ISBSG2001 – 69 WC projects implemented after the year 2001 and 224 CC projects implemented up to the end of year 2001.
- ISBSG – no date restriction to the 187 WC and 826 CC projects, meaning that CC projects with implementation date more recent than WC projects are allowed. This data set can be used to simulate the case in which it is known that other companies can be more evolved than the single company analysed.

Information on how these databases were preprocessed can be found in [8] and is not included here due to space limitations. Four input attributes (development type, language type, development platform and functional size) and one output attribute (software effort in person-hours) were used. The WC projects were sorted based on the implementation date to compose a stream of incoming projects. Figures 1c, 1d and 1e show the productivities of the CC projects of these databases.

3 EXPERIMENTAL SETUP

Dycom algorithm was analysed with different CC splits. A baseline WC approach was also used in order to complement the analysis. The experiments followed the experimental setup from the paper proposing Dycom [10]. In particular, Regression Trees (RTs) were used as Dycom's base learners. As explained in [10], RTs were chosen because they are local approaches, where estimations are based on the projects that are most similar to the project being predicted. This can help dealing with the heterogeneity within each data set, making RT's predictive performance competitive in comparison with several other SEE approaches [9]. The approaches used to answer the research questions were the following:

- **Dycom:** Dycom was used with RTs as the CC and WC models with $p = 10$, i.e., a new WC training project was provided only at every 10 time steps. So, Dycom uses only 10% of the WC projects for training, as in [10]. Whenever a new WC training project was made available, the WC RT used by Dycom was rebuilt from scratch using all WC training projects so far. Minku and Yao [10] suggested that the CC splits are created in such a way that different CC subsets have similar size. Therefore, different CC splits were created in the following way. Given a desired number of CC subsets k, productivity thresholds were set to the values that lead to k CC subsets with the same number of projects. Least productive CC subsets may have one project less than the most productive CC subsets if the total number of CC projects is not a multiple of the number of CC subsets. The experiments used the RT implementation *REPTree* provided by WEKA [4], where splits are created so as to minimise the variance of the targets of the training projects in the nodes.
- **RT [10]:** RTs were created to reflect WC online learning, forming a baseline for the analysis. Whenever a new WC training project was provided, the current RT was discarded and a new RT was trained on all projects so far. This approach considers that all WC completed projects contained known true effort, i.e., every time step received a WC training project. Therefore, it uses ten times more WC training projects than Dycom and Clustering Dycom.

The parameters of all approaches have been set to the same values as in previous work [10], except for the number of CC subsets M, which varied between 1 and 6 in order to investigate the impact of the CC subsets on Dycom's predictive performance.

As in [10], at each time step, the performance on the next ten WC projects was evaluated based on Mean Absolute Error (MAE) and Standardised Accuracy (SA). The equations to calculate these measures are the following, where T is the number of projects used for evaluating the performance, y_i is the actual effort for the project i, and $\hat{y}_i$ is the estimated effort for project i:

$$MAE = \frac{1}{T}\sum_{i=1}^{T}|\hat{y}_i - y_i|;$$

$$SA = \left(1 - \frac{MAE}{MAE_{rguess}}\right) \cdot 100,$$

where MAE is the MAE of the approach being evaluated and MAE_{rguess} is the MAE of 1000 runs of random guess. Random guess is defined as uniformly randomly sampling the true effort over all the WC projects received up to the current time step. Therefore, it has access to the same number of WC training projects as RT, which is 10 times higher than the number of WC projects used by Dycom and Clustering Dycom.

MAE has been recommended by [13] for being unbiased towards under or overestimations. SA is an unbiased measure that allows for interpretability – it is viewed as the ratio of how much better an approach is than random guess [13]. So, it can be used to give a better idea of the magnitude of the differences in performance. Mean Magnitude of the Relative Error (MMRE) and percentage of predictions within 25% of the actual value (PRED(25)) have not been used because they are biased towards underestimations [13] and could lead to misleading conclusions. The comparisons involving MAE have been supported by Wilcoxon Sign-Rank tests and A12 effect size [15], which are both non-parametric. As suggested by Vargha and Delaney, A12 of 0.50 indicates no difference, up to 0.56 indicates a small difference, up to 0.64 indicates medium and over 0.71 indicates large difference. Wilcoxon and A12 have not been used for comparing SA because SA is an interpretable equivalent of MAE.

The original Dycom and Clustering Dycom with Hierarchical Clustering were run a single time for each data set, as they are deterministic when using deterministic RTs. When using K-Means and EM, Dycom was run 30 times, and the MAE obtained at each time step was averaged across these 30 runs.

4 THE IMPACT OF THE NUMBER OF CC MODELS ON DYCOM

This section investigates the impact of the choice of CC splits on Dycom's predictive performance (RQ1). For that, we compare the predictive performance obtained by Dycom using six different CC splits, generated as explained in section 3. Table 1 shows the MAE and SA obtained by the baseline RT and by Dycom with different numbers of CC subsets. Wilcoxon Sign Rank tests were performed to test the null hypothesis H0 that there is no difference between the MAE of a given approach and the MAE of the top ranked approach, and the alternative hypothesis H1 that there is a difference. Holm-Bonferroni corrections were adopted to account for multiple comparisons, at an overall level of significance of 0.05. The effect size A12 for the comparisons is also shown.

As we can see from table 1, the predictive performance usually varies significantly depending on the number of CC subsets used by Dycom. In the cases where the null hypothesis H0 is rejected, the effect size A12 varied from 0.53 (very small effect size) to 0.87 (large effect size). Therefore, a good choice of CC split can considerably improve Dycom's MAE.

It is noteworthy that the default number of three CC subsets used in Dycom's original paper [10] was the best for only one database (ISBSG). This means that the competitive MAE obtained in that paper can be further improved by appropriately tuning the number of CC subsets. In particular, the best number of CC subsets enables Dycom to achieve significantly better MAE than RT. The magnitude of improvements varies from 7.4 units of SA for ISBSG to 30.86 units of SA for ISBSG2001. This means that Dycom can not only enable companies to use much less WC training projects, but can also lead to good improvements in MAE.

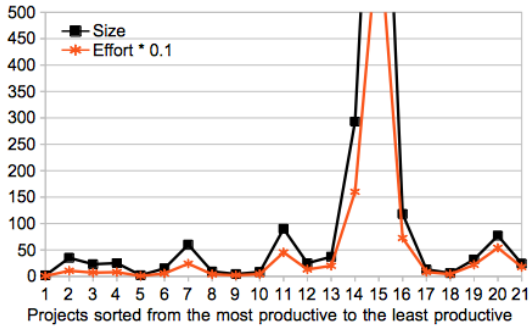
However, in a few cases, a bad choice of CC split led to considerably worse MAE (or SA) than the baseline RT. In particular, the SA of a given choice of CC split was more than 10 units worse than that of RT in 7 cases out of 30 (CocNasaCoc81 with 1, 3, 4, 5 and 6 CC subsets, and ISBSG with 1 and 5 CC subsets).

We have also found that, when using a split of 3 CC subsets herein, Dycom's MAE was 874. This value is much larger than

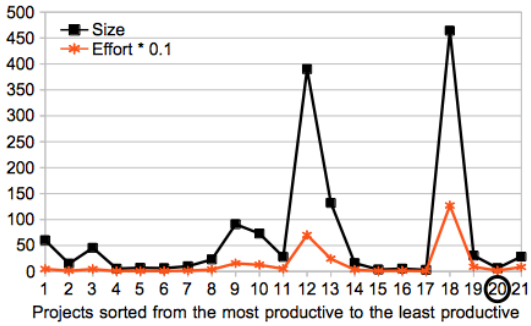
Table 1: Predictive performances obtained by threshold split of CC data based on productivity, averaged across time steps. WC RT's baseline results are also shown.

Database	#CC	MAE	SA	P-value	A12
KitchenMax	1	2176	37.75	1.13E-01	0.51
	2	2316	33.76	1.94E-05	0.61
	3	2218	36.56	1.00E-04	0.54
	4	2319	33.67	5.23E-21	0.59
	5	2465	29.49	4.25E-21	0.67
	6	2165	38.06	–	–
	RT	2441	30.18	3.01E-11	0.62
CocNasaCoc81	1	541	-13.31	1.05E-05	0.67
	2	261	45.34	–	–
	3	874	-82.93	3.37E-09	0.87
	4	552	-15.63	2.09E-06	0.78
	5	390	18.33	9.17E-07	0.68
	6	554	-15.86	1.29E-06	0.73
	RT	319	33.14	1.41E-01	0.52
ISBSG2000	1	2789	36.24	1.39E-15	0.67
	2	2418	44.72	3.38E-04	0.55
	3	2789	36.24	4.21E-09	0.63
	4	2319	46.97	3.37E-01	0.51
	5	2285	47.75	9.61E-03	0.53
	6	2215	49.35	–	–
	RT	2753	37.05	1.11E-05	0.62
ISBSG2001	1	2435	40.80	1.67E-02	0.54
	2	3159	23.19	3.92E-09	0.63
	3	2516	38.83	2.70E-02	0.52
	4	2353	42.79	–	–
	5	2434	40.82	4.47E-02	0.53
	6	2554	37.90	1.69E-03	0.57
	RT	3622	11.93	1.83E-07	0.76
ISBSG	1	4961	18.11	3.96E-28	0.71
	2	3080	49.15	1.11E-09	0.57
	3	2806	53.69	–	–
	4	3059	49.51	3.52E-09	0.54
	5	5871	3.09	1.20E-27	0.77
	6	2854	52.89	1.09E-03	0.56
	RT	3253	46.29	6.37E-06	0.58

The column #CC contains the number of CC subsets used with Dycom, and RT for indicating the baseline RT. The top and bottom ranked MAEs and their corresponding SAs are highlighted in lime (light grey) and orange (dark grey), respectively. The p-values are the results of the Wilcoxon Sign Rank tests to compare the MAE of each approach against the top ranked one for a given database. P-values highlighted in yellow (light grey) represent statistically significant difference at the overall level of significance of 0.05 when using Holm-Bonferroni corrections considering the 6 comparisons made for a given database. A12 effect size represents the probability that the top ranked approach has better MAE than a given approach.



(a) Medium productivity. The y-axis is capped at 500 to avoid hindering visualisation of the projects with lower effort and size.



(b) High productivity. The project with the 20th highest productivity has its identifier circled in the x-axis.

Figure 2: Effort and size of Cocomo 81 (CC) projects with medium and high productivity.

the MAE of 162 obtained in [10] when using slightly different thresholds for the splits into 3 CC subsets. A closer look reveals that Dycom's MAE is worsened when the 20th most productive project is placed together with other highly productive projects, rather than with projects of medium productivity. Figure 1b does not provide any evidence of this project being an outlier among other highly productive projects. Figure 2 shows the separate size and effort of this project in comparison with the medium and highly productive projects. It does not provide any evidence of this project being an outlier either.

The fact that a bad choice of CC split can lead to worse performance than RT's MAE, and that it is difficult to manually identify outlier projects that could significantly affect predictive performance, motives the investigation of whether clustering methods could help to decide how to best split the CC data.

In summary, this section shows that the choice of CC splits can significantly affect Dycom's predictive performance, answering RQ1.

5 CONCLUSIONS

The choice of CC split has a significant impact on Dycom's predictive performance. A good choice can improve Dycom's predictive performance over the results published in [10]. A poor choice can worsen it. It would be desirable to have a method to automatically decide on a good CC split.

ACKNOWLEDGEMENTS

Part of the experiments from this report have been conducted by Siqing Hou during his internship at the University of Birmingham (UK).

REFERENCES

[1] 2015. The Promise Repository of Empirical Software Engineering Data. (2015).
[2] 2017. The SEACRAFT Repository of Empirical Software Engineering Data. (2017).
[3] B. Boehm. 1981. *Software Engineering Economics.* Prentice-Hall, Englewood Cliffs, NJ.
[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA Data Mining Software: An update. *SIGKDD Explorations* 11, 1 (2009), 10–18.
[5] ISBSG. 2011. The International Software Benchmarking Standards Group. http://www.isbsg.org. (2011).
[6] B. Kitchenham, S. L.. Pfleeger, B. McColl, and S. Eagan. 2002. An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software (JSS)* 64 (2002), 57–77.
[7] T. Menzies and M. Shepperd. 2012. Special Issue on Repeatable Results in Software Engineering Prediction. *Empirical Software Engineering (ESE)* 17 (2012), 1–17.
[8] L.L. Minku and X. Yao. 2012. Can Cross-company Data Improve Performance in Software Effort Estimation?. In *International Conference on Predictive Models in Software Engineering (PROMISE)*. Lund, Sweden, 69–78.
[9] L.L. Minku and X. Yao. 2013. Ensembles and Locality: Insight on Improving Software Effort Estimation. *Information and Software Technology (IST)* 55, 8 (2013), 1512–1528.
[10] L. Minku and X. Yao. 2014. How to Make Best Use of Cross-company Data in Software Effort Estimation?. In *ICSE*. Hyderabad, 446–456.
[11] L. Minku and X. Yao. 2017. Which Models of the Past Are Relevant to the Present? A software effort estimation approach to exploiting useful past models. *Automated Software Engineering Journal* 24, 3 (2017), 499–542.
[12] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris. 2005. Software Productivity and Effort Prediction with Ordinal Regression. *Information and Software Technology (IST)* 47 (2005), 17–29.
[13] M. Shepperd and S. McDonell. 2012. Evaluating Prediction Systems in Software Project Estimation. *IST* 54, 8 (2012), 820–827.
[14] L. Song, L.L. Minku, and X. Yao. 2013. The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines. In *PROMISE*. Baltimore, USA, Article No. 9, 10p., doi: 10.1145/2499393.2499394.
[15] A. Vargha and H. D. Delaney. 2000. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. In *Journal of Educational and Behavioral Statistics*, Vol. 25. 101–132.