ABSTRACT
Collecting the data required for quality prediction within a development team is time-consuming and expensive. It is tempting to make predictions using data that crosses from other projects or even other companies.

The literature is contradictory on the value of effort estimates based on such cross data. We hypothesize that data irrelevancies are the root cause of unreliable cross effort estimation. To test that, this study takes three groupings of three projects, and trains effort estimators for project X using either local data (from within the same project) or imported data (that crosses from other projects or companies). This is repeated with and without relevancy filtering.

Our results clearly show that without relevancy filtering, imported data performed significantly worse than using local data. However, with relevancy filtering, imported data performed no worse than local data. Therefore, we recommend the use of relevancy filtering whenever generating estimates using data from another project.

Categories and Subject Descriptors
D.2.9 [Software Engineering]: Effort Estimation

Keywords
Effort estimation, data mining, cross, within

1. INTRODUCTION
Effort estimates for software development are often wrong. Consider NASA’s Check-out Launch Control System, which was canceled when the initial estimate of $200 million was overrun by an additional $200M [25]. This case is not unique: estimates are often wrong by a factor of four or more [1,2,9].

Generating accurate effort estimates requires the use of detailed and accurate historical data. Such a repository of historical records must be large enough to generate smooth, non over-fitted models, but it must also be regularly edited to remove projects based on obsolete technologies and methodologies that no longer fit with the company’s standards.

Constructing such a database is not easy. Project metrics must be tracked and regularly updated, sometimes over the course of months or years. Most companies do not invest the resources required for such book-keeping. In many cases the required data has not been archived at all. For example, after two years we were only able to add 7 records to our NASA wide software cost metrics repository [20].

When data is scare within one project, it is tempting to use data imported from other projects. Such cross-project data exist; for example the PROMISE repository [3] offers a dozen effort estimation data sets for public access. In theory, it is not recommended to use cross data. Effort estimation functions on a theory of locality, i.e. new projects follow similar practices to historical projects and should require a similar amount of effort. As Chen et. al. [4] have shown, inconsistencies in data collection across multiple companies create locality-specific biases in cross data sets. Such biases result in an unacceptable amount of variance in the effort calculations.

In practice, the costs of local data collection is so high, that cross data may be the only option. A recent survey paper has evaluated within or cross data for effort estimation [13]. They concluded that they could not make a conclusion; that the current findings are contradictory about the relative merits of within or cross data.

Effort estimation is not the only field to struggle with the issue of cross data. The same problem exists in the field of defect prediction. Many researchers (e.g. [20] shows that it is possible to apply data miners to local within data to build reasonably accurate predictors for module defects. However, such data mining can fail when applied to cross data [27].
Recently, it has been shown that it is acceptable to use cross data sources for defect prediction, providing that data has been pre-processed by some sort of relevancy filtering [26]. Given a large training set, such relevancy filters select a small subset relevant to the current test case. Such filtering removes training instances that create noise in the estimation process, leaving a body of data that, in theory, follows the principle of locality.

The success of relevancy filtering for defect prediction prompts us to apply it to effort estimation. To the best of our knowledge, this is the first exploration in the effort estimation community of the effects of relevancy filtering when applied to cross and within project data.

Our research is guided by the following research questions:

RQ1 What is the effect of selecting particular cross projects on estimation performance?

RQ2 Is there any evidence that cross data can yield accuracy values as high as within data?

RQ3 Do the characteristics of a particular dataset have an influence on the within versus cross performance?

The rest of this paper is structured as follows. After some background notes on effort estimation and cross vs within data, we will discuss our preferred two-pass relevancy filtering algorithm. Cross and within experiments will then be conducted, with and without relevancy filtering. We show that cross data can usually attain estimation accuracies just as high as those of within data, provided that a relevancy filter is applied to the data, prior to making estimates.

Our conclusion will be that any organization that wishes to utilize effort estimation, but lacks the resources or experience, could make use of publically-available cross data. Filtered cross models can be used to provide accurate estimates until a firm has collected enough historical records to switch to a localized data source.

2. BACKGROUND

2.1 Experts vs. Models

Software effort estimation fall into two groups [22]: Expert judgment and model-based techniques.

Expert judgment methods are one of the most widely used estimation methods [6]. Application of expert judgment may be either explicit (following a method like Delphi [1]) or implicit (informal discussions among such experts). One problem with expert-based estimates is that they may fail victim to competing interests in the sense that a faulty estimation of a senior expert may be taken over the more accurate estimation made by a junior expert within the same company. Another problem, indicated by Jorgensen et. al., is the poor capability of humans to improve their own expert judgment [7].

Model-based techniques are the methods generated by using algorithmic and parametric approaches or by induced prediction systems. The former approach is the adaptation of an expert-proposed model to local data. A well-known example to such an approach is Boehm’s COCOMO method [2]. The latter approach is useful in the case where local data does not conform to the specifications of the expert’s method. A few examples of induced prediction systems are linear regression, neural nets, model trees and analogies [19,23]. All of these systems are built on inherent assumptions. In the case where data violates such assumptions, patches are applied. An example of a patch is taking the logarithm of exponential distributions before linear regression [2,10]. However, choosing the appropriate patch again requires qualified experts.

Different organizations may choose to use an expert judgment, a model-based approach, or some combination of the two in different settings. However, the goal of any estimation model is a common one - to attain high estimation accuracy. On the other hand, estimation accuracy does not depend only on the choice of estimation model. Another critical factor is the possession of detailed and accurate historical data that has been carefully brought up to date.

2.2 Within/Cross Effort Estimation

When a model-based technique is chosen for estimation, historical data is required to learn the inner details of those models. Organizations may choose to collect and maintain information concerning their own past projects in a dataset - this is referred to as a within dataset. Previous studies have suggested using such locality-specific past data for accurate estimates [9,12]. However, collection of within data comes with a cost. As experts on the field, Kitchenham and Mendes have reported three problems likely to occur when an organization wishes to rely on local data [13,16,17]:

1. The time required to accumulate enough local data may be prohibitive.

2. By the time that the local dataset is large enough, the technologies employed by the local projects may have changed and old projects may have become obsolete.

3. For collecting local data in a consistent manner, care is necessary.

On top of the before mentioned reasons, based on our personal experience, we can say that in the result-oriented environment of today’s corporations, the time that must be spent collecting data (and the inevitable postponement of tangible results) is likely to decrease the enthusiasm of managers for estimation practices.

The problems regarding local data collection have motivated researchers to search for alternative solutions like the use of cross datasets (that is, datasets containing data from several different companies) [16,17]. Although software effort estimation is a relatively young research field, the collective effort of many contributors have enabled the building of considerable repositories of software effort datasets. For example, in the PROMISE data repository alone\(^1\), there are currently twelve effort prediction datasets that are available for public access [3]. However, the use of cross data is not

\(^1\)http://promisedata.org
a silver-bullet solution and comes with its own set of problems [13, 16, 17]:

1) It is even more difficult to ensure the consistency of data collection across multiple organizations.

2) Process and practices differ across companies, leading to differing trends in the data.

3) In the absence of a proper sampling strategy, it is difficult to make sure that randomly-selected cross projects truly form a random sample of the defined population.

In a systematic review on the issue, Kitchenham and Mendes tried to find a consensus in the literature about the relative benefits of cross vs within data for effort estimation [11, 13]. Their reluctant conclusion was that no such conclusion exists. Among the ten studies in their review, only seven of them showed independent evidence in their comparisons of accuracies between cross and within prediction models. Amongst those seven valid studies, three studies found that cross performance was not significantly worse than within. The remaining four found that within models significantly outperformed cross models [11]. Therefore, the findings on cross vs. within data usage are currently inconclusive.

2.3 Within/Cross Defect Prediction

Using cross data for the purposes of prediction is not limited to software effort estimation. The subject has also been addressed in the software defect prediction domain. Turhan et. al. questioned the applicability of cross data in defect prediction and found that cross data can be successfully used [26]. Cross data, taken “as is”, is only useful under relatively limited conditions [26]. However, cross data filtered for irrelevancies through the use of a nearest-neighbor technique has yielded surprisingly high accuracy results [26].

Zimmermann et. al. have addressed the same question at the individual project level and have come up with similar conclusions [27]. They agree that building cross-prediction models is a serious challenge. They also report using the cross data as is does not lead to useful predictions [27]. In order to increase these accuracy values, Zimmermann propose an approach that selects projects for cross predictors [27]. This approach can be regarded as an expert-guided relevancy filtering. Taking these findings in defect prediction domain as an impetus, we would like to explore the applicability and performance of applying a relevancy filtering technique to cross data in the effort estimation domain.

3. RELEVANCY FILTERING

Our relevancy filter extends standard analogy-based estimation methods (which we call ABE0).

3.1 ABE0

Analogy-based estimation (ABE), in the simplest terms, generates an estimate for a test project by gathering evidence from the effort values of similar projects in some training set. By analyzing the previous research of experts like Shepperd et. al. [24], Mendes et. al. [18] and Li et. al. [15] on the field of analogy-based estimation, we can come up with a baseline technique:

- Form a table whose rows are completed past projects (this is a training set).
- The columns of this set are composed of independent variables (the features that define projects) and a dependent variable (the recorded effort value).
- Decide on how many similar projects (analogies) to use from the training set when examining a new test instance, i.e. k-values.
- For each test instance, select those k analogies out of the training set.
  - While selecting analogies, use a similarity measure (such as the Euclidean distance of features).
  - Before calculating similarity, apply a scaling measure on independent features to equalize their influence on this similarity measure.
  - Use a feature weighting scheme to reduce the influence of less informative features.
- Use the effort values of the k nearest analogies to calculate an effort estimate.

We can refer to this baseline framework as ABE0. ABE0 uses the Euclidean distance as a similarity measure, whose formula is given in Equation 1.

\[
Distance = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2}
\] (1)

We can see the weighting approach used for project features in Equation 1. In Equation 1, \( w_i \) corresponds to feature weights applied to independent features. For our case we do not favor any features over the others, therefore ABE0 uses a uniform weighting, i.e. \( w_i = 1 \).

The adaptation strategy for the effort estimate is not necessarily a complex process. ABE0 simply returns the median of the effort values of the k nearest analogies. The reason why ABE0 uses median instead of mean in our research is due to the fact that the number of instances that are selected for making the estimate after filtering are quite few in number (2 to 10 instances). In that case, use of mean may let extreme effort values have a very strong influence on the estimation. However, we want our estimates to represent the majority of selected instances and not greatly affected by extreme values, which may or may not be noise. Therefore, we use median instead of mean.

3.2 Two-Pass Filtering

Our relevancy filter is a small variant of ABE0. It is a two-pass system:

- Pass 1 removes the training instances implicated in poor decisions;
- Pass 2 selects those instances nearest the test instance.

In pass 1, the training projects are used to generate a binary tree. The leaves of this binary tree (level zero of the
The leaves of the remaining sub-trees are the survivors of pass one. These move to pass 2 where the survivors are used to build a second binary tree (called BT2). BT2 is generated and traversed by test instances in the same fashion as BT1. This time, while traversing the tree, instead of storing the variances of sub-trees, we use the variance as a decision criterion. If the variance of the current tree is larger than its sub-trees, then continue to move down the subtree; otherwise, stop moving and select the instances in the current tree as the relevant instances and adapt them for estimation. Since the described model is a version of ABE0, the adaptation used for the selected relevant instances of BT2 is the same as ABE0, taking the median. A simple visualization of this relevancy filtering approach is given in Figure 1.

This filter is similar to the NN-filter used by Turhan et al. [26], except that there is no need to pre-specify the number of analogies k to be used for estimation. Each test instance selects its own relevant analogies by traversing to different sub-trees of BT2.

For a detailed discussion on the rationale behind this filter, see [14]. All we need to say here is that this filter is known to generate low errors for ABE0-style effort estimation [14]. Hence, it is a suitable tool for the rest of this study.

4. METHODOLOGY
Previous studies on cross vs. within effort estimation were criticized for their lack of proper statistical analysis or due to the limited datasets used. Therefore, while conducting our research, we have paid particular attention to meeting the requirements of Kitchenham et al [13]. The details regarding dataset selection criteria as well as our statistical analysis methods will be provided in Sections 4.1 and 4.2 respectively.

The reproducibility of results is a critical issue in software engineering [8, 11, 13]. As we want our results to be reproducible in both similar and different settings, we have chosen only publicly available datasets and we have provided as much details as possible regarding our experimental settings.

4.1 Data
In our research, we have used subsets of three commonly-used datasets in software effort estimation research: Nasa93, the original Cocomo81 [2], and Desharnais [5].

We will denote the subsets of Nasa93 as Nasa93c1, Nasa93c2 and Nasa93c5. Nasa93c1, Nasa93c2 and Nasa93c5 contain projects from different NASA development centers around the United States (denoted as development centers 1, 2 and 5 in the complete dataset). In a similar fashion, subsets of Cocomo81 will be denoted as Coc81o, Coc81e and Coc81s and refer to:

- Coc81o’s “organic projects” come from small teams with high experience working with less than rigid requirements.
- Coc81e’s “embedded projects” are developed within tight constraints (hardware, software, operational etc.).
- Co81s’ “semidetached projects” are at an intermediate stage between the organic and embedded modes.

Lastly, the Desharnais dataset will be split into three different subsets based on the three different programming languages used for development projects. We will denote the subsets of Desharnais as DesL1, DesL2 and DesL3 (languages 1, 2 and 3 respectively). The subsets of each dataset are chosen because they form self-consistent and reliable data groupings. Since each of these subsets have certain common criteria (the development center, development mode, or development language), each subset will be treated as a separate within dataset. The details such as feature and instance sizes as well as content of datasets are given in Figure 2. All of the datasets used in this research are available in PROMISE data repository [3].

Kitchenham and Mendes attribute a particular importance to dataset size in their review [13] and state that larger
Figure 2: Nine treatments = three subsets of three data sets. Indentation in column one denotes that indented dataset is a subset of another dataset.

within datasets lead to more reliable comparisons between within and cross models. In order to evaluate the goodness of within datasets, they propose a quality scoring of four values: poor (less than ten projects), fair (between ten to twenty projects), good (between twenty to forty projects) and excellent (more than forty projects) [13]. According to quality criteria indicated by Kitchenham et. al. [13], the nine within datasets used in this study are grouped as follows: one of excellent quality, five of good quality, and three of fair quality.

4.2 Experiments

For each of the three main datasets (Nasa93, Cocomo81 and Desharnais) in our research, we have conducted within and cross experiments. The division of main datasets into its subsets is structured such that the subsets have a self consistent structure according to a dataset-specific characteristic. Therefore, each subset can be considered similar to a within dataset that contains projects sharing the particular characteristics of a single development firm.

To understand the within and cross data formation, assume that a dataset \( X \) with its three subsets \( X_1, X_2 \) and \( X_3 \) is under consideration. For within experiments, the relevancy filtering described in Section 3.2 is applied on each one of \( X_1, X_2 \) and \( X_3 \) separately and the median of the filtered project instances in the training set is stored as the effort estimate for the test instance. For the separation of training and testing sets, the leave-one-out method is used. Leave-one-out selects one instance out of a dataset of \( n \) instances as the test instance and uses the remaining \( n - 1 \) instances as the training set.

For the cross experiments, one of \( X_1, X_2 \) or \( X_3 \) is chosen as the test set and the combination of the remaining two forms the cross dataset for training. This time, the relevancy filtering is applied on the cross dataset, and the estimations for projects in the test set are stored.

Each of the within and cross experiments are repeated twenty times in order to remove any bias that would otherwise be brought by a particular test and training set combination.

4.3 Performance Criteria

In order to compare the performance of within and cross datasets, we have used two measures: the magnitude of relative error (MRE) and win-tie-loss values generated by a statistical rank-sum test. MRE is utilized by the authors because it is the most commonly used performance criterion for software effort estimation [21]. Furthermore, as we can see from Formula 2, MRE gives a per-instance based estimation performance evaluation.

\[
MRE = \frac{|actual_i - predicted_i|}{actual_i} \tag{2}
\]

However, MRE is subject to many pitfalls. When MRE is used as a stand-alone evaluation criterion (i.e. not combined with appropriate statistical tests), it may lead to biased or even false conclusions. To prevent us from falling into MRE-related pitfalls, we use another performance criterion called a win-tie-loss calculation (shown in Figure 3). A win-tie-loss calculation states the fact that comparison between two methods \( i \) and \( j \) makes sense only if they are statistically significant. If they are statistically the same, that could indicate that they are observations coming from the same distribution, therefore they are noted as a tie and their tie values (tie\(_i\) and tie\(_j\)) are incremented. On the other hand, if there is a statistical difference between two methods, then the method with a lower median MRE score, say \( i \), is identified as a “winner” and the one with the lower MRE, say \( j \), is identified as a “loser.” The related values win\(_i\) and loss\(_j\) are incremented by one. As we repeat each treatment twenty times, win-tie-loss calculation provides us with a good perspective on the success of each method across different datasets. The pseudocode for a win-tie-loss calculation is given in Figure 3. For the comparison of methods in win-tie-loss calculation, a non-parametric statistical test

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Features</th>
<th>Projects</th>
<th>Content</th>
<th>Historical Effort Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cocomo81</td>
<td>17</td>
<td>63</td>
<td>NASA projects</td>
<td>months</td>
</tr>
<tr>
<td>Coc81o</td>
<td>17</td>
<td>24</td>
<td>Cocomo81 organic projects</td>
<td>months</td>
</tr>
<tr>
<td>Coc81e</td>
<td>17</td>
<td>28</td>
<td>Cocomo81 embedded projects</td>
<td>months</td>
</tr>
<tr>
<td>Coc81s</td>
<td>17</td>
<td>11</td>
<td>Cocomo81 semidetached projects</td>
<td>months</td>
</tr>
<tr>
<td>Nasa93</td>
<td>17</td>
<td>93</td>
<td>NASA projects</td>
<td>months</td>
</tr>
<tr>
<td>Nasa93c1</td>
<td>17</td>
<td>12</td>
<td>Nasa93 projects from center 1</td>
<td>months</td>
</tr>
<tr>
<td>Nasa93c2</td>
<td>17</td>
<td>37</td>
<td>Nasa93 projects from center 2</td>
<td>months</td>
</tr>
<tr>
<td>Nasa93c5</td>
<td>17</td>
<td>39</td>
<td>Nasa93 projects from center 5</td>
<td>months</td>
</tr>
<tr>
<td>Desharnais</td>
<td>12</td>
<td>81</td>
<td>Canadian software projects</td>
<td>hours</td>
</tr>
<tr>
<td>DesL1</td>
<td>11</td>
<td>46</td>
<td>Desharnais projects developed with language 1</td>
<td>hours</td>
</tr>
<tr>
<td>DesL2</td>
<td>11</td>
<td>25</td>
<td>Desharnais projects developed with language 2</td>
<td>hours</td>
</tr>
<tr>
<td>DesL3</td>
<td>11</td>
<td>10</td>
<td>Desharnais projects developed with language 3</td>
<td>hours</td>
</tr>
</tbody>
</table>

Figure 3: Pseudocode for Win-Tie-Loss Calculation Between Method \( i \) and \( j \)
5. RESULTS
In our experiment, we analyzed 3 datasets * 3 subsets = 9 treatments. We evaluated cross and within performances of each particular dataset, subject to statistical tests, with and without relevancy filtering.

5.1 Without Relevancy Filtering
In this first experiment, we have 9 treatments and for each treatment we observe the estimation performances when within and cross datasets are used. For this purpose we used a linear regression model. Two-pass filtering was not applied.

In cross experiments, for each data set, we selected one of the 3 subsets as the test set and the remaining two as the train set. We then built a linear regression model on the cross data and apply this model on the test set. For the within dataset we also used a linear regression model. The test case selection for within experiment is performed in accordance with leave-one-out method, which picks up one of the instances in the dataset as the test set and uses the remaining instances as the train set. The linear regression model that is built on the train set is then tested on the single test instance. After applying linear regression model on within and cross datasets, we calculated the win-tie-loss values for each treatment.

As shown in Figure 4, we see that in a minority of cases (4/5, see Nasa93c5, Coc81e, Coc81s and DesL2), cross and within data perform just as well as each other. In the majority case (3/5, see Nasa93c1, Nasa93c2, Coc81o, DesL1, DesL3), within performed better than cross data. That is, in the absence of relevancy filtering, the within datasets yield significantly lower MRE values in majority of cases.

5.2 With Relevancy Filtering
This section shows that for each data set, the application of relevancy filtering reverses the conclusion of the previous section; i.e. the cross data becomes useful for estimating the local project.

Figure 4 shows the the win-tie-loss values for the subsets of Nasa93. The greedy clustering algorithm of the two pass relevancy filtering uses some non-determinism (when breaking ties between instances of similar distances), so we repeat these experiments twenty times.

This results shows us that, in all three treatments, the tie values are quite high. This indicates that, for at least 75% of the tests, there is no statistical difference between filtered cross and within results. In short, for Nasa93, the performance of cross data (filtered for relevancy) is indistinguishable from the performance of within data.

Figure 5 shows the win-tie-loss values for the subsets of Nasa93. The greedy clustering algorithm of the two pass relevancy filtering uses some non-determinism (when breaking ties between instances of similar distances), so we repeat these experiments twenty times.

This results shows us that, in all three treatments, the tie values are quite high. This indicates that, for at least 75% of the tests, there is no statistical difference between filtered cross and within results. In short, for Nasa93, the performance of cross data (filtered for relevancy) is indistinguishable from the performance of within data.

Figure 6 shows the win-tie-loss values for the subsets of Cocomo81. In two out of the three treatments the tie values are 19, which tells us that for these treatments, within and cross performance are almost identical. However, the first treatment shows a preference for within data on thirteen of the twenty tests.

The win-tie-loss values for subsets of Desharnais are given in Figure 7. The derived results for the Desharnais subsets are similar to those of Cocomo81 treatments: Two out of the three treatments show identical tie values of 19, which again suggests that the performance of filtered cross datasets is statistically identical to within datasets. However, in one of the treatments, within outperforms cross on sixteen of the twenty trials.
<table>
<thead>
<tr>
<th>Cross Dataset</th>
<th>Test Set</th>
<th>Instances selected in BT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nasa93c2 and Nasa93c5</td>
<td>From Nasa93c1 From Nasa93c2 From Nasa93c5</td>
<td>0.4 2.1 1.8</td>
</tr>
<tr>
<td>Nasa93c1 and Nasa93c5</td>
<td>From Nasa93c1 From Nasa93c2 From Nasa93c5</td>
<td>1.4 0.0 1.8</td>
</tr>
<tr>
<td>Nasa93c1 and Nasa93c2</td>
<td>From Nasa93c1 From Nasa93c2 From Nasa93c5</td>
<td>2.4 1.1 0.0</td>
</tr>
<tr>
<td>Coc81e and Coc81s</td>
<td>From Coc81o From Coc81e From Coc81s</td>
<td>0.2 0.3 1.4</td>
</tr>
<tr>
<td>Coc81o and Coc81s</td>
<td>From Coc81o From Coc81e From Coc81s</td>
<td>3.6 0.3 1.4</td>
</tr>
<tr>
<td>Coc81o and Coc81e</td>
<td>From Coc81o From Coc81e From Coc81s</td>
<td>2.8 0.3 0.0</td>
</tr>
<tr>
<td>DesL2 and DesL3</td>
<td>From DesL1 From DesL2 From DesL3</td>
<td>0.2 0.3 1.4</td>
</tr>
<tr>
<td>DesL1 and DesL3</td>
<td>From DesL1 From DesL2 From DesL3</td>
<td>2.1 0.3 0.1</td>
</tr>
<tr>
<td>DesL1 and DesL2</td>
<td>From DesL1 From DesL2 From DesL3</td>
<td>3.2 1.6 0.0</td>
</tr>
</tbody>
</table>

Figure 8: Mean number of instances used for estimation after filtering in 20 runs. Cross datasets are combinations of two within datasets tested on another within dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Win</th>
<th>Tie</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coc81o</td>
<td>within</td>
<td>13</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Coc81e and Coc81s</td>
<td>cross</td>
<td>0</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>Coc81e</td>
<td>within</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>Coc81o and Coc81s</td>
<td>cross</td>
<td>0</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Coc81s</td>
<td>within</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Coc81o and Coc81e</td>
<td>cross</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 6: MRE win-tie-loss values for Cocomo81 from 20 randomized assessments. In 2 treatments cross data is the same as the within data. However, in the case of Coc81o, within outperforms cross data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Win</th>
<th>Tie</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>DesL1 and DesL3</td>
<td>within</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>DesL2</td>
<td>cross</td>
<td>0</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>DesL1 and DesL3</td>
<td>within</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>DesL3</td>
<td>cross</td>
<td>0</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>DesL1 and DesL2</td>
<td>within</td>
<td>16</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>DesL1 and DesL2</td>
<td>cross</td>
<td>0</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure 7: MRE win-tie-loss values for Desharnais from 20 randomized assessments. In the case of DesL3 the within data is much better than the cross data. For other treatments, within and cross data are statistically the same.

In summary, with relevancy filtering, in the majority case (7 treatments) the cross data performs as well as the within data for effort estimation. There are only two treatments, DesL3 and Coc81o, where within performance was significantly better than cross performance. A possible explanation for those two scenarios may be hidden in the dataset size or in the quality of the within datasets, but the currently-available information makes it difficult to suggest any conclusive reason for the situation.

5.3 Number of Analogies

Figure 8 shows the mean number of instances used for analogy-based estimation by our two-pass relevancy filtering algorithm. Surprisingly, the number of selected analogies is very small: mean value around 3. Further, while exceptions exist, the selected analogies come from multiple other projects. For example, for the Nasa93 dataset, the data relevant to center c1 came from centers c2 and c5 (respectively).

This suggests that we should revisit what we mean by within and cross. Our introduction referenced Chen’s view that estimation works on a principle of locality. However, general software effort studies make use of commonly-explored past datasets such as the ones used in this research (Nasa93, Cocomo81, and Desharnais). This issue of internal validity threaten all effort studies using only past data. We can mitigate this problem by simulating the behavior of a learned theory in new settings. In our study, we make use of the leave-one-out method for all treatments to address such internal validity issues. Leave-one-out selection enables us to separate the training and test sets completely in each experiment.

In our future work section, we will return to this point.

6. THREATS TO VALIDITY

The ideal case for satisfying internal validity would be to learn a theory from a past situation and then to apply the theory to a new setting. However, general software effort studies make use of commonly-explored past datasets such as the ones used in this research (Nasa93, Cocomo81, and Desharnais). This issue of internal validity threaten all effort studies using only past data. We can mitigate this problem by simulating the behavior of a learned theory in new settings. In our study, we make use of the leave-one-out method for all treatments to address such internal validity issues. Leave-one-out selection enables us to separate the training and test sets completely in each experiment.

Another issue is our choice of learning algorithms. In the above, we have apparently compared the results of linear regression (in § 5.1) to analogy-based estimation (in § 5.2). Is such a comparison valid? We believe so since what was con-
stant in the comparison was the cross vs within data used in the analysis. Only the learner was changed, and changed in a very specific way. For the purposes of this discussion, the essential feature of the linear regression study was that it trained from all available cross data. The second study, on the other hand, explored what happens when the training set was restricted to just instances close by the test instance. We considered conducting a third study where we applied relevancy filtering as a pre-processor to linear regression. In that third study, we would have used relevancy filtering to select the training instances for linear regression. However, Figure 8 showed that the number of selected instances selected in this way would be around three- a number that is far too small for effective regression.

A further issue requiring discussion is the two passes used by our relevancy filter. Perhaps that it is an over-elaboration, and a standard simple one-pass Euclidean nearest neighbor algorithm might suffice. This issue has been addressed elsewhere [14]: our two-pass system produces much lower errors than the standard nearest neighbor algorithm. Hence, we recommend it for analogy-based effort estimation.

Lastly, one frequently asked question about this work relates to the “cross” nature of the subsets of our data. Is it fair to describe NASA projects from different centers as “cross”? When writing about defect data sets from NASA, Zimmerman et al. comment:

“For their study Turhan et al. [26] analyzed 12 NASA projects (mostly in C++) which they considered cross-company because they were all developed by contractors under the umbrella of NASA. However, all projects had to follow stringent ISO-9001 industrial practices imposed by NASA, so it is unclear to what extent the data can be actually considered cross-company. [27]”

In reply, we assert that NASA is a much more diverse organization than is commonly appreciated. NASA software is written by layers of contractors working for different companies around the nation. Arguing that all ISO-compliant organizations are the same is like saying the a CMM level3 weapons manufacturer is writing the same software as a CMM level3 finance company. It is true that rigid hardware requirements on flight systems forces a conformity in the final software. However the methods used to create that software can be radically different at different contractor organizations. Also, very little of our NASA data is from flight systems (13 out out 93 records spread evenly across the three centers used to form the NASA subsets). The bulk of our data come from ground systems which are much less constrained in nature than flight systems. In any case, we have empirical support for our claim that all our subsets come from truly different sources. In the results presented above:

- Without relevancy filtering, for the purposes of estimating the current project, the cross-subset data is usually useless.
- With relevancy filtering, the differences disappear. Predictions using the cross-subset data is usually useful.

Those results are enough to recommend the use of relevancy filtering when processing data from other projects.

7. CONCLUSIONS

With these results, we can now address the questions that we proposed to guide this research.

RQ1) What is the effect of selecting particular cross projects on the estimation performance?

We have seen that, in the majority case, cross datasets filtered for relevancy have performed as well as within datasets. Therefore, we can state that selecting particular cross projects, guided by a relevancy filter, has a positive effect on the estimation performance of analogy-based estimation.

RQ2) Is there evidence that cross data can yield accuracy values equal to that of within data?

For seven out of the nine treatments used in our experiments, filtered cross data has performed equally to within data. Our experiments have yielded statistical evidence that cross data can be used to obtain high accuracy values. However, for cross data to attain such values, we cannot use the data “as is”. Rather, we must use a relevancy filtering method to remove the instances that cause a high performance variance.

RQ3) Do the characteristics of particular datasets have an influence on the within versus cross performance?

In two treatments, we have observed that within datasets significantly outperformed cross datasets. Therefore, we cannot say that the raw characteristics of particular datasets have no influence on estimation performance. However, the exact reason behind this is not yet obvious. In our case, the two treatments that yielded higher within performance were datasets with 24 and 10 projects respectively. Based on dataset size alone, it is difficult to observe any connection between number of projects and estimation performance. One potential reason could be the data quality of each set. However, we do not yet have any evidence to back such claims.

8. FUTURE WORK

Given the complex multi-dimensional nature of the software creation process, the geographical dimension may be less important than other factors. The most similar software to what you are writing now may not be in the next office. Rather, it may be in an office on the other side of the world.

Going forward, we would like to learn exactly why an instance is deemed “relevant” or “irrelevant” by our filter. In other words, we would like to know exactly which features are most influential when assigning relevancy. The ability to identify these exact dimensions would make the selection of appropriate projects easier for any institution that uses cross data. It would also lead to (a) more accurate filtering techniques; and (b) a better understanding of the structure of software projects including where to find data most relevant to some current project.
Acknowledgments
This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 60873072 and 90718042.

9. REFERENCES