How to Find Relevant Data for Effort Estimation?

Ekrem Kocaguneli, Tim Menzies
Lane Department of Computer Science and Electrical Engineering
West Virginia University,Morgantown, USA
ekocagun@mix.wvu.edu, tim@menzies.us

ABSTRACT

Background: Building effort estimators requires the correct selection of training data. How can we find that data? It is tempting to *cross* the boundaries of development type, location, language, application and hardware to use existing datasets of other organizations. However, prior results caution that using such *cross* data may not be useful.
Aim: We test the conjecture that *instance selection* can automatically prune irrelevant instances and retrieval from the remaining examples is useful for effort estimation, regardless of their source.
Method: We selected 8 *cross-within divisions* (21 pairs of *within-cross* subsets) out of 19 datasets and evaluated these divisions under different analogy-based estimation (ABE) methods.
Results: Between the *within* and *cross* experiments, there were very few statistically significant differences in (a) the performance of effort estimators; or (b) the amount of instances retrieved for estimation.
Conclusion: There is no practical difference between *cross* and *within* data. After applying instance selection, the remaining examples, be they from *within* or from *cross* source divisions, can be used for effort estimation.

Categories and Subject Descriptors
H.4 [**Software Cost Estimation**]: *k*-NN; D.2.8 [**Software Engineering**]: Cost—*within resource, cross resource*

1. INTRODUCTION

A recurring problem in effort estimation is finding training data that is relevant to some local problem. When we cannot find enough local training data, it is tempting to try and import data from other sources. However, it is not clear that this approach is useful: many studies report that using imported data degrades estimation efficacy, perhaps because the imported data is not relevant to the local context (e.g. see the Kitchenham et al. [14] and Zimmermann et al. [31] studies discussed later in this paper).

In this paper, we offer one solution to the problem of importing relevant data from other sources in order to make estimates about local models. Our solution is based on a fresh look at what it means to say that examples are *local* or *imported*. Many publications [2, 6–8, 19, 28, 29] including several of our own [21, 22] either explicatively or tacitly assume "$locality(1)$"; i.e. clumps of similar projects can be discovered *using a single feature*. For example, some authors claim that, for a project in a specific organization, software effort models work better when calibrated with local data collected within that same organization. Proponents of such a *within source* approach assume that it is best to retrieve training data for examples divided acccording to:

- The *project type* being developed: e.g. embedded, etc;
- The *development centers* of the different developers;
- The *development language* of the projects;
- The *application type* (management information system; guidance, navigation, and control; etc);
- The *targeted hardware platform*;
- Just the *in-house or outsourced* development projects;

It would be useful if $locality(1)$ were false since, if true, this means that any lessons learned from one organization may never apply to another. If so, then our ability to make general conclusions about software engineering (SE) would be confined to small, highly specialized, sub-groups (e.g. just one company).

The opposite to $locality(1)$ is "$locality(N)$"; i.e. the assumption that effort estimation data forms a complex multi-dimensional space that can only be usefully divided *using multiple features*. If this were true, then this would be very good news since that would mean that relevant data for effort estimation does not come just from small sub-groups within one organization. Rather, useful data could be collected from many projects including *cross sources*. This would simplify the construction of effort models for new projects: just search other contexts for the right data for the new project. Also, it could lead to conclusions about SE that are general to many development contexts. This paper argues against $locality(1)$ and for $locality(N)$ using two predictions that would support $locality(N)$ and would contradict $locality(1)$:

PREDICTION 1: *Effort models built from training data divided on a single feature will perform no better (and, perhaps, even worse) than those that divide the data using multiple features.*

PREDICTION 2: *Consider project data that was grouped into divisions w.r.t. to the value of a feature. If training data is retrieved from within and across those divisions, then it would be equally as probable to find useful data* within *as* cross *those divisions.*

Recent research offers much support for **PREDICTION 1**. In a study of 90 effort estimation methods (ten pre-processors $\times$ nine learners), we found that the best methods were those that divided the training data according to multiple features [12]. That result is detailed in our *Related Work* section.

For **PREDICTION 2**, we test if divisions based on different *single* features change the effort estimation process. We allow an *instance selection & retrieval* algorithm to find which instances are best for training. That algorithm is given access to all the training data, or just the data divided via a single feature. It will be shown that:

> *The probability of retrieving training data from within or across divisions based on single features is the same.*

This result, plus the results in [12], are strong support for $locality(N)$ since they confirm both **PREDICTION 1** and **PREDICTION 2**. These results have three practical implications: Firstly, it would mean that effort estimation is not dependent on some artificial, and trite, division of training data such as the source organization; the kind of application; or any other single feature division. Given the complex multi-dimensional nature of the software creation process, divisions of training data based on (say) geographical dimensions 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 (and automatic *instance selection & retrieval algorithms* can find that data).

Secondly, if $locality(N)$ was true, it would be useful to build effort estimation models from data taken from multiple contexts. Hence:

- Building estimation models is less expensive since generating local models need not wait for an elaborate (and expensive and time-consuming) collection process from local data.
- There are effects in SE that transcend our current divisions of data (e.g. data from company1 or company2). This is a very exciting result since it promises future generalizable SE results that apply to many organizations.

Thirdly, $locality(N)$ offers a strong business case for collecting SE data in some sharable repository. Such repositories can now be trusted to provide, at least some, relevant historical examples for building effort estimators for some new project.

1.1 Terminology

Here, we offer some terminology clarification. Other papers such as Kitchenham et al. [14] and Zimmermann et al. [31] discuss the impact of dividing data according to a single feature (either the *organization* or *application* name). This paper explores those single-feature division *as well as* other single feature divisions such as the *project type* being developed; the *development centers* of the developers; the *development language* of the software; the *application type*; the *targeted hardware platform*; and *in-house or outsourced* development. So whereas (e.g.) Kitchenham et al. explore cross-vs-within company data, we explore cross-vs-within data "*sources*" where the data is divided by the values of *any* single feature. As shown below, we find no examples where single feature division improves a state-of-the-art effort estimation method.

Other two terms that need clarification are *instance selection* and *instance retrieval*. Instance selection refers to a filtering method, where data is filtered to a subset of more *relevant* instances; hence, *instance selection* and *filtering* is used interchangeably in the rest of the text. On the other hand, *instance retrieval* is the process of finding the closest neighbors (e.g. in a k-NN method). In that sense, *instance selection* is a pre-processor to a second process; whereas, *instance retrieval* is the part of a single estimation process.

2. RELATED WORK

2.1 Evidence for PREDICTION 1

Keung et al. [12] built 90 effort estimators using 10 pre-processors and 9 learners. Pre-processors included normalization, various discretization methods and feature selection (PCA, stepwise, sequential forward). Learners included $k = 1$ and $k = 5$-nearest-neighbor, linear and stepwise regression, CART, neural nets and PCR.

The 90 estimators were assessed via multiple accuracy statistics. Let t instances have actuals $a_1, a_2, .., a_t$. Prediction models generate prediction p_i for instance i. If $|p_i - a_i|$ is AR_i (the *absolute residual* difference between predictions and actuals) then:

- MAR is the *mean absolute residual* $(\sum_i p_i - a_i)/t$.
- MRE_i and MER_i are the ratios AR_i/a_i and AR_i/p_i.
- $PRED(X)$ is the percent of t with $MRE_i \leq X\%$.
- $MMRE = \sum_i MRE_i/t$
- $MMER = \sum_i MER_i/t$.
- $MdMRE$ is median MRE_i value.
- Finally, the *balanced errors* are $MBRE_i = (p_i - a_i)/min(p_i, a_i)$ and $MIBRE_i = (p_i - a_i)/max(p_i, a_i)$

The 90 estimators were used in a leave-one-out study on twenty data sets from http://promisedata.org/?cat=14, and were compared via a Mann-Whitney test (95% confidence). As it might have been predicted by Shepperd et al. [25], the ranking of the estimators varied across different data sets and the different accuracy estimators.

However, Keung et al. found a small group of 13 estimators that were consistently the best performers across all data sets (measured according to all of MAR, MRE, PRED(25), MMRE, MMER, MdMRE, MBRE, and MIBRE). In terms of this paper, the major result of Keung et al. is that all these 13 estimators used CART or $k = 1$ nearest neighbor. This is significant since both these estimators use multiple features to sub-divide the training data:

- k-th nearest neighbor algorithms use all project features (perhaps, weighted by some feature) to determine related projects [26];
- Tree-based algorithms like CART [4] divide data into multiple branches, where each branch tests and divides that data on multiple features.

Hence, this result is strong support for **PREDICTION 1**.

2.2 Other Evidence

Other results in the literature are also inconclusive about $locality(N)$. In their review of papers building effort models using data from within one company or across multiple companies, Kitchenham et al. [14] found equal evidence for and against the value of building effort models based on a single feature division (specifically, they found four studies favoring the use of *within* company data, and another three reporting that using *cross* data performance is not significantly worse than *within*). In other work, in the field of defect prediction, Zimmermann et al. [31] found that predictors performed worse when trained from cross-application data than from

within-application data. The evidence for their conclusion is quite emphatic: *within* defeated *cross* in 618 out of 622 comparisons.

On the other hand, support for $locality(N)$ comes from the work of Turhan et al. [30], and Kocaguneli et al. [15]. Turhan et al. compare defect predictors learned from cross or within resource data. Like Zimmermann, they found that using *all* cross resource data lead to poor predictor performance (very large false alarm rates). However, after *instance selection* pruned away irrelevant cross resource data, they found that the cross resource predictors were equivalent to the predictors learned from within resource data [30]. Inspired by [30], Kocaguneli et al. [15] used *instance selection* as a pre-processor for a study on cross-vs-within resource effort estimation. In a limited study with three data sets, they found that after instance selection, the performance differences in the predictors learned from cross or within data were statistically insignificant.

2.3 Resolving the Evidence

The results in [30] and [15] support $locality(N)$ but the other results discussed above are inconsistent or unsupportive. How can we reconcile this conflicting evidence? One way is to note that:

- Studies supporting $locality(N)$ all used a filtering method (instance selection).
- Instance selection is *not* seen in the Kitchenham, Zimmermann et al. studies.

An *instance selection* method uses every feature (perhaps, with some feature weighting) to find relevant training examples. Hence, the studies with *instance selection* [15, 30] offer more support for $locality(N)$ than for $locality(1)$; however, they are hardly conclusive, since they do not collect the information required to comment on **PREDICTION 2**. What is required is a well-controlled instance selection and retrieval experiment over data divided by some single feature. **PREDICTION 2** would be supported if the instance selection & retrieval method (TEAK) retrieved as much data *within* as *across* the filtered single feature divisions.

3. METHODOLOGY

Figure 1 illustrates how this study differs to prior work. Most effort estimation research falls into Figure 1.a where estimation models are applied *within* one source set to learn an estimator. Examples of this approach include [1, 18, 20, 21, 26].

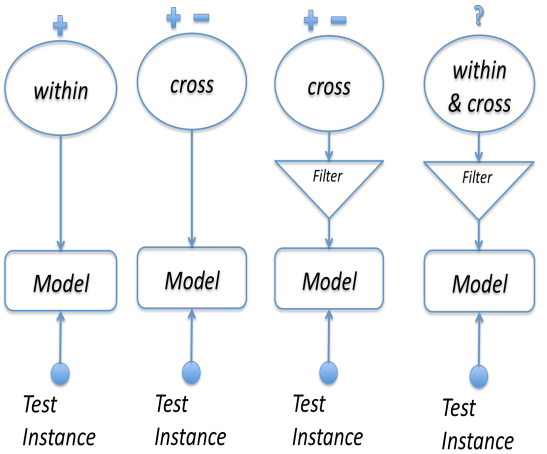
A smaller number of papers, such as those surveyed by Kitchenham et al. [14], explore building models using data that falls *across* many sources (see Figure 1.b). Fewer still are the papers like [15, 30]) that, prior to learning, apply some instance selection to cross resource data sources (see Figure 1.c).

To the best of our knowledge, this paper is the first that allows an effort estimator to select (filter) training data from either *cross* or *within* different sources, then checks what data was retrieved from which source (see Figure 1.d).

3.1 Datasets

There are 2 fundamental factors that were considered for selection of the datasets used in this research:

- Public availability: For reproducibility purposes
- Cross-within divisibility: For enabling *cross* vs. *within* experimentation



(a) *Within* (b) *Cross* (c) *Filter* (d) *Within & Cross Filter*

Figure 1: The problem types in *within* vs. *cross* data comparison and our conclusions so far. "+" and "-" signs on top of each approach mean positive and negative results respectively. A "?" sign means that the approach has not previously been investigated.

A critical issue in SE is the ability of the proposed results to be reproducible [10, 14] and use of proprietary data is a major obstacle towards this goal. Therefore, all our datasets are publicly available through PROMISE data repository [3].

We define *cross-within division* as the subset(s) of effort data that are formed through division of one feature: instances having the same value for that feature form a subset. Such features are plausible candidates for generating a *cross* source experiment, i.e. the features should be likely to change from one source to other. Accordingly, this study began by exploring what PROMISE effort data can be divided via a single feature. After manually inspecting more than 20 datasets, six were found to be suitable for *cross-within* experimentation. Those six data sets support the 21 *cross-within divisions* shown in Figure 8. The selected division criteria include:

- project type: embedded, organic and semidetached (*cocomo81*),
- center: geographical development center (*nasa93*),
- language type: programming language used for development (*desharnais*),
- application type: on-line service program, production control program etc. (*finnish* and *maxwell*),
- hardware: PC, mainframe, networked etc. (*kemerer* and maxwell),
- source: whether in-house or outsourced (maxwell).

We will use the following nomenclatures: If a subset name is followed by a set of numbers, they correspond to values of the feature used to form the subset. If a name has multiple numbers at the end (e.g. finnishAppType2345) then all instances with these values are combined in a single subset.

3.2 Instance Selection and Retrieval

The goal of our experiment is to determine at what probability a learner retrieves training instances from either cross- or within-sources. The learner used in this study is TEAK [17], which is best explained as an extension to ABE0 [15, 17].

Dataset	Criterion	Subsets	Subsets Size
cocomo81	project type	cocomo81e	28
		cocomo81o	24
		cocomo81s	11
nasa93	development center	nasa93_center_1	12
		nasa93_center_2	37
		nasa93_center_5	39
desharnais	language type	desharnaisL1	46
		desharnaisL2	25
		desharnaisL3	10
finnish	application type	finnishAppType1	17
		finnishAppType2345	18
kemerer	hardware	kemererHardware1	7
		kemererHardware23456	8
maxwell	application type	maxwellAppType1	10
		maxwellAppType2	29
		maxwellAppType3	18
maxwell	hardware	maxwellHardware2	37
		maxwellHardware3	16
		maxwellHardware5	7
maxwell	source	maxwellSource1	8
		maxwellSource2	54

Figure 2: 6 datasets are selected from 20+ candidates. Then selected datasets are divided into subsets according to a criterion that can define a *cross-within division*. The datasets, subset sizes as well as the selection criteria are provided here.

3.2.1 ABE0

Analogy-based estimators (ABE) generate an estimate for a *test* project by retrieving similar past projects (a.k.a. analogies) from a database of past projects and adapting their effort values into an estimate. We use ABE methods in this study since 1) they are widely investigated methods in the literature [5, 11, 13, 15, 17, 18, 20], 2) they are particularly helpful for *cross* source studies as they are based on distances between individual project instances.

There are various design options associated with ABE methods such as the distance measure for nearness [20], adaptation of analogy effort values [20], row processing [5, 13], column processing [13, 18] and so on. Elsewhere we show that these options can easily lead to more than 6000 ABE variants [12]. Here we define ABE0 that is a *baseline* ABE method that combines the tools used in Kadoda & Shepperd [11], Mendes et al. [20], and Li et al. [18]:

- Input a database of past projects
- For each test instance, retrieve k similar projects (analogies).
  - For choosing k analogies use a similarity measure.
  - Before calculating similarity, scale independent features to equalize their influence on the similarity measure.
  - Use a feature weighting scheme to reduce the effect of less informative features.
- Adapt the effort values of the k nearest analogies to come up with the effort estimate.

ABE0 uses the Euclidean distance as a similarity measure, whose formula is given in Equation 1, where w_i corresponds to feature weights applied on independent features. ABE0 framework does not favor any features over the others, therefore each feature has equal importance in ABE0, i.e. $w_i = 1$. For adaptation ABE0 takes the median of retrieved k projects.

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

3.2.2 TEAK

TEAK is a *variance-based* instance selector that discards training data associated with regions of high estimation variance. It augments ABE0 with instance selection and an indexing scheme for filtering relevant training examples. Detailed description of TEAK can be found in [17]. In summary, TEAK is a two-pass system:

- Pass 1 prunes training instances implicated in poor decisions (instance selection);
- Pass 2 retrieves closest instances to the test instance (instance retrieval).

In the first pass, training instances are combined using greedy-agglomerative clustering (GAC), to form an initial cluster tree that we call GAC1; e.g. Figure 3. Level zero of GAC1 is formed by leaves, which are the individual project instances. These instances are greedily combined into tuples to form the nodes of upper levels. The variance of the effort values associated with each subtree (the performance variance) is then recorded and normalized

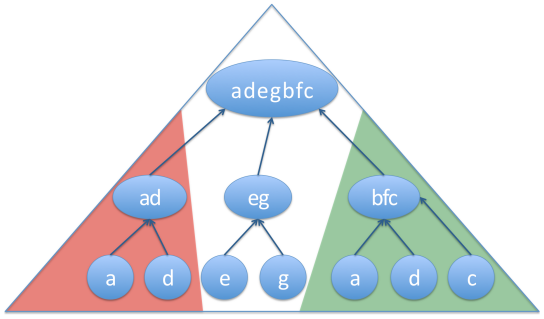


Figure 3: A sample GAC tree with regions of high variance (red) and low variance (green). GAC trees may not always be binary. For example here, leaves are odd numbered, hence node "c" is left *behind*. Such instances are pushed *forward* into the closest node in the higher level. For example, "c" is pushed forward into the "b+f" node to make "b+f+c" node.

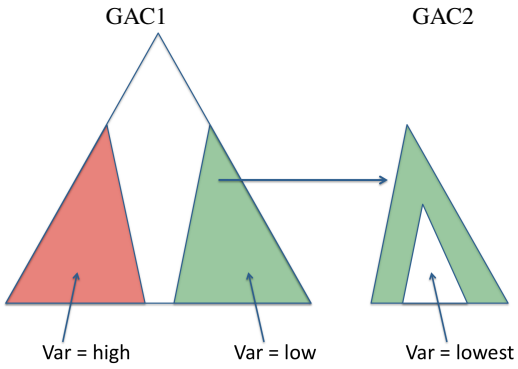


Figure 4: Execution of TEAK on 2 GAC trees, where tree on the left is GAC1 and the one on the left is GAC2 (i.e. lower variance sub-tree of GAC1). The instances in the low variance region of GAC1 (green region) are selected to form GAC2. Then test instance traverses GAC2 until no decrease in effort variance is possible. Wherever the test instance stops is retrieved as the subtree to be used for adaptation (white region of GAC2).

					Simple ABE0				
	TEAK	LR	NNet	*k=best*	k=1	k=16	k=2	k=4	k=8
MdMRE									
Cocomo81	▲								
Cocomo81e	▲								
Cocomo81o	▲								
Nasa93		▲							
Nasa93c2		▲							
Nasa93c5	▲								
Desharnais		▲							
Sdr	▲								
ISBSG-Banking	▲								
Count	6	3	0	0	0	0	0	0	0
Pred(25)									
Cocomo81	▲								
Cocomo81e			▲						
Cocomo81o	▲								
Nasa93		▲							
Nasa93c2		▲							
Nasa93c5	▲								
Desharnais		▲							
Sdr	▲								
ISBSG-Banking	▲								
Count	5	3	1	0	0	0	0	0	0
MAR									
Cocomo81	▲								
Cocomo81e	▲								
Cocomo81o	▲								
Nasa93		▲							
Nasa93c2		▲							
Nasa93c5	▲								
Desharnais		▲							
Sdr	▲								
ISBSG-Banking	▲								
Count	6	3	0	0	0	0	0	0	0

Figure 5: Results from 20 repeats of a leave-one-out experiment, repeated for the performance measures of MdMRE, Pred(25) and MAR. Black triangles mark when an estimator was one of the top-ranked methods for a particular data set (where ranking was computed via $win - loss$ from a Mann-Whitney test, 95% confidence). The $Count$ rows show the number of times a method appeared as the top performing variant. Results from [17].

$min..max$ to 0..1. The high variance sub-trees are then pruned, as these are the sub-trees that would cause an ABE method to make an estimate from a highly variable instance space. Hence, pass one prunes sub-trees with a variance greater than $\alpha\%$ of the maximum variance seen in any tree. After some experimentation, we found that $\alpha = 10$ lead to estimates with lowest errors.

The leaves of the remaining sub-trees are the *survivors* of pass one. They are filtered to pass 2 where they are used to build a second GAC tree (GAC2). GAC2 is generated and traversed in a similar fashion to GAC1, then test instances are moved from root to leaves. Unlike GAC1, this time variance is a decision criterion for the movement of test instances: If the variance of the current tree is larger than its sub-trees, then continue to move down; otherwise, stop and retrieve the instances in the current tree as the analogies. TEAK is a form of ABE0, so its adaptation method is the same, i.e. take the median of the analogy effort values. A simple visualization of this approach is given in in Figure 4.

We use TEAK in this study since, as shown by the leave-one-out experiments of [17], its performance is comparable with other commonly-used effort estimators including neural networks (NNet) and linear regression (LR). A summary of those performance results are given in Figure 5 (for a complete analysis and for definitions of datasets please refer to Figure 7 of [17]). That figure shows the results of a statistical comparison of various performance indicators (defined in [12]) for nine effort estimators and nine data sets from http://promisedata.org/?cat=14:

$win_i = 0, tie_i = 0, loss_i = 0$
$win_j = 0, tie_j = 0, loss_j = 0$
if Mann-Whitney(P_i, P_j) says they are the same **then**
  $tie_i = tie_i + 1$;
  $tie_j = tie_j + 1$;
else
  if mean or median(P_i) $<$ median(P_j) **then**
    $win_i = win_i + 1$
    $loss_j = loss_j + 1$
  else
    $win_j = win_j + 1$
    $loss_i = loss_i + 1$
  end if
end if

Figure 6: Pseudocode for win-tie-loss calculation between methods *i* and *j* w.r.t. performance measures P_i and P_j. Note here that only for Pred(30) the comparison is based on actual values ($Pred(30)_i$, $Pred(30)_j$) rather than mean or median values of performance measure arrays (median(P_i), median(P_j)).

- The columns $k = 1, 2, 4, 8, 16$ denote variants of standard ABE0 where estimates are generated from the k-th nearest neighbors.
- The column $k = best$ denote a variant of ABE0 where k was chosen by an initial pre-processor that chose a best k value after exploring the training data.
- The columns LR and $NNet$ refer to linear regression and neural nets.

The black triangles in Figure 5 mark when an estimator was one of the top-ranked methods for a particular data set. Ranking was accomplished via the $win - loss$ calculation of Figure 6. We first check if two distributions i, j are statistically different according to the Mann-Whitney test. In our experimental setting, i, j are arrays of performance measure results coming from two different methods. If they are not statistically different, then they are said to *tie* and we increment tie_i and tie_j. On the contrary, if they are different, we updated win_i, win_j and $loss_i, loss_j$ after a numerical comparison of performance measures. The related pseudocode is given in Figure 6. To get rid of any bias due to a particular experimental setting, for every experiment 20 runs are made. The key feature of Figure 5 is that TEAK always performed better than the other ABE0 methods, and usually performed better than neural nets. TEAK's only near-rival was linear regresison but, as shown in the LR columns, TEAK was ranked top nearly twice as much as linear regression.

3.3 Experimentation

The experimentation of this research has two different goals:

- The *performance comparison* of a state-of-the-art effort estimation method (TEAK) when trained from *within* and *cross* source data.
- The *retrieval tendency* goals question the tendency of a *within* test instance to retrieve *within* or *cross* data. In other words, given the chance that a test instance had access to *within* and *cross* data at the same time, what percentage of every subset would be retrieved into *k* analogies used for estimation?

3.3.1 Performance Comparison

For performance comparison we have two settings: *Within* and *cross*. In *within* data setting, only *within* one source is used as

the dataset and a testing strategy of leave-one-out cross-validation (LOOCV) is employed. LOOCV works as follows: Given a *within* dataset of T projects, 1 project at a time is selected as the test and the remaining $T-1$ projects are used for training, so eventually we have T predictions. The resulting T predictions are then used to compute 4 different performance measures defined in §2: PRED(30), MAR, MMRE, MdMRE.

Cross data setting uses *within* data as the test set and the *cross* data as the training set. In this setting LOOCV is used as follows: each *within* source is selected as the test instance and TEAK derives an estimate for that instance by adapting *cross* analogies. Ultimately we end up with T predictions adapted from a *cross* dataset. Finally the performances under *within* and *cross* data settings are compared. For that purpose we use both mere performance values as well as win-tie-loss statistics.

3.3.2 Retrieval Tendency

For retrieval tendency, we select test instances according to LOOCV. For each test instance, we are left with training sets of $T-1$ *within* data and the subsets of *cross* data. After marking every *within* and *cross* instance, we combine the two datasets into a single training set and let the test instance choose analogies from the unified training set (note that analogies are retrieved after filtering in pass #1 of TEAK). In this setting our aim is to see what percentage of *within* and *cross* subsets would appear among retrieved *k* analogies. The retrieval percentage for a subset is the ratio of instances retrieved in analogies from that subset to its total size (see Equation 2).

$$Percentage = \frac{SubsetSizeInAnalogies}{SubsetSize} \quad (2)$$

4. RESULTS

4.1 Performance Comparison

For performance comparison 4 different performance measures are employed: MAR, MMRE, MdMRE and Pred(30). The actual performance values are also evaluated subject to Mann Whitney statistical test at 95% confidence and this evaluation is summarized by win-tie-loss statistics.

Figure 7 shows *within* and *cross* data performance when TEAK is used as the estimation method. For each performance measure win-tie-loss statistics (abbreviated with **W**, **T**, **L** respectively) of *within* performance when compared to *cross* over 20 runs as well as actual performance measure values are reported.

The gray lines in Figure 7 show the experiments where the $within$ results "dominate"; i.e. win in more than half the comparisons. Note that there are only two gray lines. In the remaining $\frac{19}{21}$ cases, the *within* data does not provide an advantage over *cross* data. In fact, in one case (kemererHardware1) the *within* data is far worse than *cross* with an **L** value of 20. These results are confirmation of previous conclusions [15, 30] in a much larger scale with 4 error measures and 21 different cases: instance selection on *cross* sources improves its performance to an extent where it is no worse than *within* data.

4.2 Retrieval Tendency

To explore retrieval tendency, LOOCV is used to choose single test instances one by one from a *within* dataset of size T. The remaining $T-1$ *within* instances are combined with the *cross* subsets. Prior to combination, every training instance is marked with the source that it belongs to (cross vs within). Then the test instance is allowed to choose *k* analogies from a training set of *within* and *cross* data.

The rig lets us check the percentage retrieval of analogies from each one of the *within* and *cross* subsets. Those results are shown in Figure 8. Each *cross-within division* is represented with a row of 2 or 3 subsets; columns named *"From* S_i*"* where $i \in \{1, 2, 3\}$

Dataset	MAR					MMRE					MdMRE					Pred(30)				
				Median Results					Median Results					Median Results					Median Results	
	W	T	L	within	cross	W	T	L	within	cross	W	T	L	within	cross	W	T	L	within	cross
cocomo81e	0	20	0	1.0E+3	1.1E+3	0	16	4	2.4	0.9	4	16	0	0.7	0.9	4	16	0	0.1	0.1
cocomo81o	0	20	0	8.2E+2	8.1E+2	2	18	0	0.8	2.7	2	18	0	0.8	0.9	2	18	0	0.1	0.2
cocomo81s	18	2	0	3.6E+1	1.8E+2	15	5	0	1.0	8.6	15	5	0	0.5	1.7	13	5	2	0.2	0.1
nasa93_center_1	0	20	0	1.4E+2	1.3E+2	0	20	0	1.2	2.0	0	20	0	0.8	0.8	0	20	0	0.6	0.5
nasa93_center_2	4	16	0	1.8E+2	2.1E+2	2	18	0	1.3	2.8	2	18	0	0.7	0.8	2	18	0	0.2	0.2
nasa93_center_5	0	20	0	6.9E+2	8.9E+2	0	12	8	0.9	0.7	8	12	0	0.6	0.8	8	11	1	0.2	0.2
desharnaisL1	11	9	0	9.9E+2	2.0E+3	9	11	0	0.6	2.4	9	11	0	0.4	1.7	9	11	0	0.4	0.3
desharnaisL2	0	20	0	2.8E+3	2.8E+3	0	20	0	0.5	0.6	0	20	0	0.5	0.5	0	20	0	0.2	0.3
desharnaisL3	0	20	0	2.8E+3	3.2E+3	2	18	0	0.5	0.5	2	18	0	0.4	0.5	2	18	0	0.2	0.2
finnishAppType1	0	20	0	3.2E+3	3.8E+3	0	20	0	1.1	1.0	0	20	0	0.5	0.6	0	20	0	0.3	0.2
finnishAppType2345	0	20	0	7.1E+3	5.4E+3	0	17	3	2.2	0.9	0	17	3	0.8	0.7	0	17	3	0.1	0.2
kemererHardware1	0	0	20	1.4E+2	5.5E+1	0	0	20	1.3	0.3	0	0	20	1.1	0.3	0	0	20	0.4	0.5
kemererHardware23456	0	20	0	2.0E+2	2.0E+2	0	20	0	0.7	0.7	0	20	0	0.6	0.5	0	20	0	0.1	0.1
maxwellAppType1	6	14	0	1.4E+3	3.2E+3	1	19	0	0.8	1.9	1	19	0	0.4	0.7	0	19	1	0.3	0.3
maxwellAppType2	0	18	2	6.6E+3	5.4E+3	0	19	1	1.2	0.9	0	19	1	0.5	0.4	0	19	1	0.3	0.3
maxwellAppType3	0	20	0	5.6E+3	6.6E+3	1	19	0	1.0	1.0	1	19	0	0.5	0.6	1	19	0	0.2	0.2
maxwellHardware2	0	20	0	5.6E+3	5.3E+3	0	20	0	0.8	1.0	0	20	0	0.5	0.5	0	20	0	0.2	0.2
maxwellHardware3	0	20	0	5.3E+3	5.9E+3	0	20	0	0.9	0.7	0	20	0	0.4	0.5	0	20	0	0.3	0.4
maxwellHardware5	0	20	0	3.6E+3	3.6E+3	0	20	0	3.7	2.8	0	20	0	0.7	0.8	0	20	0	0.1	0.1
maxwellSource1	6	14	0	1.5E+3	3.3E+3	1	19	0	0.3	0.4	1	19	0	0.1	0.4	1	19	0	0.8	0.8
maxwellSource2	0	20	0	6.0E+3	6.0E+3	0	20	0	1.2	1.9	0	20	0	0.6	0.7	0	20	0	0.2	0.2

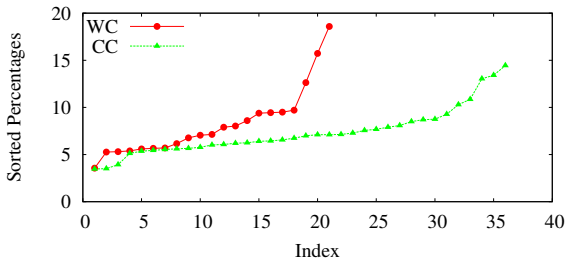
Figure 7: Results of TEAK: Comparison of performance between *within* and *cross* data w.r.t. 4 different performance measures (median of MAR, MMRE, MdMRE, Pred(30) over 20 runs) as well as *W, T, L* statistics. Highlighted rows are the cases, where *within* data is "*dominantly*" better than *cross*, i.e. wins more than half the time. Under the columns of *within* and *cross* the actual performance values associated with *within* and *cross* source datasets are provided respectively.

represent the subsets of the rows:

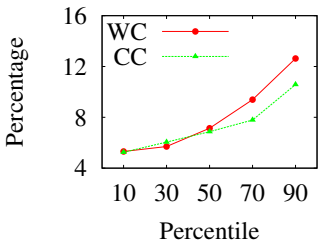
- The highlighted diagonal entries of each cell show the amount of instances retrieved from *within* subset.
- The off-diagonal values are the amount of instances retrieved from *cross* datasets.

To better see the percentages of *within* and *cross* subsets, we sorted and plotted them in Figure 9. Figure 9(a) shows the sorted percentage values, where the $within$ percentages are shown with circles, whereas the $cross$ percentages are represented by triangles. Observe how the *cross* percentage values are shifted versions of *within* percentages (this shift-effect comes from the fact that there are more *cross* subsets than *within* subsets).

The percentiles from 10^{th} to 90^{th} with increments of 20 are given in Figure 9(b). When we plot the percentiles, the shift-effect due to subset number disappears and we are able to observe that within and cross retrieval tendencies at the indicated percentile values are very close. A statistical test (Mann-Whitney, 95% confidence) confirms this: the distributions of Figure 9 are *not* statistically significantly different.



(a) Percentages



(b) Percentiles

Figure 9: Percentages and percentiles of instances retrieved by TEAK from within and cross datasets. The cross percentages are very similar to shifted version of within percentages, the shift-effect is due to different number of subsets. The percentile graph removes the shift-effect and we see that *within* test instances retrieve very close percentages of *within* and *cross* instances.

5. DISCUSSION

5.1 Implications

We have shown above that, for boundaries defined by a single feature:

- There was usually no difference in the performance of effort estimators; learned from *within* or from *across* those boundaries;
- There was usually no difference in the probability of retrieving instances for those estimates from *within* or from *cross* those boundaries.

That is, it was not useful to divide the data by *any* of single feature boundaries shown in Figure 8; i.e. by project type, geographical location of the development center, language type, application type, hardware, or source. Hence, at least for the purposes of selecting and retrieving relevant examples for effort estimation, there is no information gain in dividing data using a single feature.

Test Set	From S1	From S2	From S3
S1: cocomo81e (28)	1.0 (3.6%)	1.1 (4.8%)	1.6 (14.4%)
S2: cocomo81o (24)	1.8 (6.6%)	1.3 (5.6%)	1.1 (10.4%)
S3: cocomo81s (11)	1.4 (5.1%)	1.7 (7.0%)	1.0 (9.4%)
S1: nasa93_center_1 (12)	1.0 (8.1%)	2.9 (7.9%)	1.7 (4.3%)
S2: nasa93_center_2 (37)	1.6 (13.0%)	4.6 (12.4%)	3.8 (9.8%)
S3: nasa93_center_5 (39)	0.8 (6.7%)	2.2 (6.0%)	2.1 (5.4%)
S1: desharnaisL1 (46)	2.5 (5.5%)	1.7 (7.0%)	0.8 (7.9%)
S2: desharnaisL2 (25)	2.6 (5.6%)	1.5 (6.1%)	0.7 (6.7%)
S3: desharnaisL3 (10)	1.9 (4.1%)	1.3 (5.0%)	0.4 (4.0%)
S1: finnishAppType1 (17)	1.6 (9.1%)	1.6 (8.8%)	
S2: finnishAppType2345 (18)	1.4 (8.2%)	1.6 (8.8%)	
S1: kemererHardware1 (7)	0.6 (8.8%)	0.9 (10.7%)	
S2: kemererHardware23456 (8)	0.5 (7.3%)	0.8 (10.6%)	
S1: maxwellAppType1 (10)	0.7 (7.1%)	1.7 (5.9%)	1.0 (5.8%)
S2: maxwellAppType2 (29)	0.4 (3.7%)	1.8 (6.2%)	1.0 (5.5%)
S3: maxwellAppType3 (18)	0.6 (6.3%)	0.9 (3.2%)	1.0 (5.6%)
S1: maxwellHardware2 (37)	1.7 (4.6%)	0.8 (4.9%)	0.4 (6.0%)
S2: maxwellHardware3 (16)	2.5 (6.8%)	1.1 (6.8%)	0.3 (4.3%)
S3: maxwellHardware5 (7)	2.3 (6.2%)	0.8 (5.0%)	0.3 (4.5%)
S1: maxwellSource1 (8)	0.1 (1.6%)	2.8 (5.2%)	
S2: maxwellSource2 (54)	0.4 (4.6%)	2.8 (5.3%)	

Figure 8: The average amount of analogies (k) retrieved from within and cross resource datasets by TEAK. In parenthesis the percentage of retrieved instances out of the actual *within* source dataset is given. The diagonal entries that are highlighted with gray are the within source retrieval amounts and percentages.

5.2 Small Retrieval Sizes

The median values of the percentiles (i.e. 50^{th} percentile) in Figure 9 is 7%. Initially, this low value troubled us but after a review of the relevant literature we found that our results are consistent with prior results:

- Chang's prototype generators [5] replaced training sets of size $T = (514, 150, 66)$ with prototypes of size $N = (34, 14, 6)$ (respectively).
- That is, prototypes may be as few as $\frac{N}{T} = (7, 9, 9)\%$ of the original data. Note that these values are close to how many instances were retrieved in the above results.

5.3 Geometric Implications

Our results imply something about the location of training data in instance space. Geometrically, $locality(1)$ assumes that project data lines up along one dimension; e.g. as shown by the circles in Figure 10. This figure displays projects described in terms of a two dimensional instance space (labeled here as x and y). Note that (a) the circles are arranged (approximately) parallel to the y axis and that (b) the longer projects (indicated with larger circles) occur at

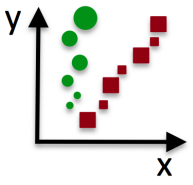


Figure 10: Projects described in two dimensions.

higher y values. The space of the circle
examples in Figure 10 could be processed by $locality(1)$ since a single feature (in this case, the vertical y-axis) usefully divides the examples with higher effort from those with lower effort.

From a geometic perspective, $locality(1)$ is improbable. Given the idiosyncrasies of software development, we find it highly unlikely that naturally occurring project examples will all line up in a row parallel to one axis. What is more likely, we believe, are geometrics like those shown as squares in Figure 10. As before, the size of the shapes indicates the effort associated with each project. Note these examples do not run parallel to any feature and the longer and shorter projects are not easily separated by either axis. The space of small squares and larger squares cannot be divided by any simplistic $locality(1)$ assumption.

5.4 Threats to Validity

External validity questions whether the results can be generalized outside the specifications of a study [23]. For the purpose of external validity, we use of 21 within-cross dataset pairs. Among 10 studies investigated by Kitchenham et al. in [14], 9 of them used single within-cross dataset pairs, and 1 study used 6 pairs. In terms of external validity, this report has higher validity than a standard *within* vs. *cross* data comparison effort estimation study.

Another consideration for external validity is the employed methods. There are thousands of possible ABE variants and there is no way that this study covers them all. There is obviously need for future research that repeats these experimentations with different ABE variants. However, experiments reported here include a filtering based variant (TEAK) built on a base variant (ABE0) and run on 21 within-cross pairs. Therefore, the extent of the experimentation in this research offers enough support for the claims that 1) *cross* data performs no worse than *within* data and 2) a *within* test instance tends to retrieve equally from *within* and *cross* projects.

Construct validity (i.e. face validity) asks if we are measuring what we actually intended to measure [24]. Previous studies have concerned themselves with the construct validity of different performance measures for effort estimation (e.g. [27]). So as not to bias our conclusions due to a limited number of measures, we used 4 different performance measures aided with win-tie-loss statistics.

In terms of *internal validity* of our results, there is one dimension of experimental conditions not explored. We are making use of LOOCV, whose a possible alternative would be N-Way cross-validation. In N-Way cross-validation, data is randomly divided into B bins and each bin is tested on a model learned from the combination of other bins (typical values for B are 3 or 10). From a theoretical point of view, not controlling the stability of our results across different testing strategies is a threat to validity, as different testing strategies entails different bias and variance conditions [9]. Elsewhere [16], we show that there is very little difference in the bias and variance values generated for LOOCV and N-way cross-validation. Since two testing strategies have similar bias-variance characteristics for effort datasets, we opted for LOOCV due to the fact that LOOCV is a deterministic procedure that can be exactly repeated by any other researcher with access to a particular data set. N-way cross-validation on the other hand requires a random number generator and a stratification heuristic (to maintain same class distribution in each bin). Without access to exact same random number generator and stratification heuristic, it would be difficult for a researcher "A" to reproduce results of researcher "B".

6. CONCLUSION

We have shown that when using a state-of-the-art effort estimator (TEAC), then after instance selection:

1. The $cross$ performance results are no worse than $within$ (see Figure 7);
2. The probability that the estimator retrieves a training instance from $cross$ or $within$ is the same (see Figure 9.b).

Result #1 grants us permission to compare cross-vs-within results (since there is no performance delta between them). Result #2 shows that the single-feature divisions have no bearing on effort estimation. Coupled with the results of [12], these results are strong support for $locality(N)$ since we have confirmed both **PREDICTION 1** and **PREDICTION 2**.

This means that (to repeat a comment made in our introduction), 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. As shown here, using instance selection tools like TEAK, it is possible to automatically find that relevant training data.

7. FUTURE DIRECTIONS

Some of the most likely future directions to this research are:

- Reproduction of this work on proprietary data.
- Investigating why particular subsets (cocomo81s, desharnaisL1) favor *within* data, whereas the rest favors both *within* and *cross*.
- Using different ABE or non-ABE methods under similar settings.
- Experimenting if limited *within* data can be supplemented with *cross* data.
- Using different features on different datasets to see if they can define a border between *within* and *cross* data.

8. REFERENCES

[1] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl. Optimal Project Feature Weights in Analogy-Based Cost Estimation: Improvement and Limitations. *IEEE Transactions on Software Engineering*, 32(2):83–92, 2006.

[2] B. Boehm. Safe and Simple Software Cost Analysis. *IEEE Software*, pages 14–17, 2000.

[3] G. Boetticher, T. Menzies, and T. Ostrand. PROMISE.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees, 1984.

[5] C.-l. Chang. Finding Prototypes for Nearest Classifiers. *IEEE Transactions on Computer*, C(11), 1974.

[6] S. Chulani, B. Boehm, and B. Steece. From Multiple Regression to Bayesian Analysis for Calibrating {COCOMO} {II}. *Journal of Parametrics*, 15(2):175–188, 1999.

[7] D. Ferens and D. Christensen. Calibrating Software Cost Models to {D}epartment of {D}efense {D}atabase: A Review of Ten Studies. *Journal of Parametrics*, 18(1):55–74, Nov. 1998.

[8] H. Habib-agahi, S. Malhotra, and J. Quirk. Estimating Software Productivity and Cost for {NASA} Projects. *Journal of Parametrics*, pages 59–71, Nov. 1998.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2 edition, 2008.

[10] N. Juristo and S. Vegas. Using Differences among Replications of Software Engineering Experiments to Gain Knowledge. In *ESEM*, pages 356–366, 2009.

[11] G. Kadoda, M. Cartwright, and M. Shepperd. On configuring a case-based reasoning software project prediction system. In *UK CBR Workshop, Cambridge, UK*, pages 1–10. Citeseer, 2000.

[12] J. Keung, E. Kocaguneli, and T. Menzies. A Ranking Stability Indicator for Selecting the Best Effort Estimator in Software Cost Estimation. *Automated Software Engineering (submitted)*, 2011.

[13] J. W. Keung, B. Kitchenham, and D. R. Jeffery. Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation. *IEEE Trans. Softw. Eng.*, 34(4):471–484, 2008.

[14] B. Kitchenham, E. Mendes, and G. H. Travassos. Cross versus Within-Company Cost Estimation Studies: A Systematic Review. *IEEE Trans. Softw. Eng.*, 33(5):316–329, 2007.

[15] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. W. Keung. When to use data from other projects for effort estimation. In *ASE'10*, pages 321–324, 2010.

[16] E. Kocaguneli and T. Menzies. The Effects of Test Set Selection on Effort Estimation (in preperation), 2011.

[17] E. Kocaguneli, T. Menzies, A. Bener, and J. Keung. Exploiting the Essential Assumptions of Analogy-based Effort Estimation. *To Appear in IEEE Trans. Softw. Eng*, 2011.

[18] Y. Li, M. Xie, and T. Goh. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*, 82(2):241–252, Feb. 2009.

[19] K. Lum, J. Powell, and J. Hihn. Validation of Spacecraft Software Cost Estimation Models for Flight and Ground Systems. In *ISPA Conference Proceedings, Software Modeling Track*, May 2002.

[20] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[21] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting Best Practices for Effort Estimation. *IEEE Transaction on Software Engineering*, 32(11):883–895, 2006.

[22] T. Menzies, D. Port, Z. Chen, and J. Hihn. Simple software cost analysis: safe or unsafe? In *PROMISE '05*, pages 1–6, New York, NY, USA, 2005. ACM.

[23] D. Milic and C. Wohlin. Distribution Patterns of Effort Estimations. In *Euromicro*, 2004.

[24] C. Robson. *Real world research: a resource for social scientists and practitioner-researchers*. Blackwell Publisher Ltd, 2002.

[25] M. Shepperd and G. Kadoda. Comparing Software Prediction Techniques Using Simulation. *Software Engineering, IEEE Transactions on*, 27(11):1014–1022, 2002.

[26] M. Shepperd and C. Schofield. Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering*, 23(12), Nov. 1997.

[27] E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. An empirical validation of the relationship between the magnitude of relative error and project size. *Eighth IEEE Symposium on Software Metrics*, pages 3–12, 2002.

[28] S. Stukes and H. Apgar. Applications Oriented Software Data Collection: Software Model Calibration Report, {TR}-9007/549-1, Management Consulting and Research, Mar. 1991.

[29] S. Stukes and D. Ferens. Software Cost Model Calibration. *Journal of Parametrics*, 18(1):77–98, 1998.

[30] B. Turhan, T. Menzies, A. Bener, and J. Di Stefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 14(5):540–578, 2009.

[31] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction. *ESEC/FSE'09*, page 91, 2009.