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 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 two conjectures: (1) instance selection can automatically prune irrelevant instances and (2) 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 & cross experiments, there were few statistically significant differences in (i) the performance of effort estimators; or (ii) the amount of instances retrieved for estimation.

Conclusion: For the purposes of effort estimation, there is little 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. We say that data divided into subsets according to locality(1) can be used for within or cross effort modeling:

- Within studies are localized to one subset;
- A cross study trains from some subsets and tests on others.

As examples of within studies, some authors claim that, for projects in a specific organization, software effort models work best 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 according 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;
- In-house or outsourced development projects;

If locality(1) was true, then any lessons learned from one organization may never apply to another. For example, we might not be able to transfer lessons learned about effort estimation from one company called (say) “Boeing” to another called “Lockheed-Martin”. 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 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. Continuing the above example, this would mean that some of the data from Boeing might apply to some of the projects at Lockheed-Martin.

Note that, if locality(N) was true, then this would simplify effort modeling 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 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 across those divisions.

Recent research offers much support for PREDICTION 1. In a study of 90 effort estimation methods (ten pre-processors × 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:

\[
\text{The probability of retrieving training data from within or across divisions based on single features is the same.}
\]

This result, plus the results in [12], are strong support for locality(N) since they confirm both PREDICTION 1 and PREDICTION 2.

1.1 Practical Implications

These results have three practical implications: Firstly, it would mean that effort estimation is not dependent on some artificial, and arbitrary, division of the training process such as the source organization; the kind of application; or any other single feature division. The complex multi-dimensional nature of the software creation process, divisions of training data based on (say) locational or 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 (or 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.

1.2 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.

Two terms need clarification before continuing this paper:

- **Instance selection** filters data to select the most relevant instances; hence, instance selection and filtering is used interchangeably in the rest of the text.
- **Instance retrieval** is the process of finding the closest neighbors (e.g. in a k-NN method).

That is, in the terminology of this paper, instance selection is a pre-processor to some other process while 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 \( a_1, a_2, \ldots, a_t \). Prediction models generate prediction \( \hat{p}_i \) for instance \( i \). If \( |p_i - a_i| \) is \( AR_i \) (the absolute residual difference between predictions and actuals) then:

\[
\begin{align*}
\text{MAR} & = \frac{\sum_i |p_i - a_i|/t}{t} \\
\text{MRE}_i & = \frac{AR_i/a_i}{AR_i/p_i} \\
\text{PRED}(X) & = \text{the percent of } t \text{ with } MRE_i \leq X\% \\
\text{MMRE} & = \frac{\sum_i MRE_i/t}{t} \\
\text{MMER} & = \frac{\sum_i MRE_i}{t} \\
\text{MdMRE} & = \text{median } MRE_i \text{ value.} \\
\text{The balanced errors are } MBRE_i & = (p_i - a_i)/\min(p_i, a_i) \\
\text{and } MIBRE_i & = (p_i - a_i)/\max(p_i, a_i)
\end{align*}
\]

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, 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 \textit{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 \textit{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 \textit{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 \textit{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 \textit{locality}(N) than for \textit{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 \textit{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 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 \textit{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 \textit{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 \textit{cross-within} experimentation. Those six data sets support the 21 \textit{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).
The goal of our experiment is to find the probability that a learner retrieves training instances from either cross- or within-sources. In order for this analysis to be meaningful, it is essential that we use some state-of-the-art learner (otherwise, a critic of this work could discount our conclusions, saying that our analysis tools were poorly selected). Hence, this section carefully documents TEAK [17], a state-of-the-art instance-based effort estimator.

Since TEAK is an extension to to ABE0 [15, 17], this section will introduce ABE0 before TEAK.

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.

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

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 sub-tree (the performance variance) is then recorded and normalized $\min..\max$ 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
check if two distributions accomplished via the of the top-ranked methods for a particular data set. Ranking was accomplished via the \( \text{win} - \text{loss} \) calculation of Figure 6. We first check if two distributions \( i, j \) are statistically different according to

\[
\begin{align*}
\text{win}_i &= 0, \text{tie}_i = 0, \text{loss}_i = 0 \\
\text{win}_j &= 0, \text{tie}_j = 0, \text{loss}_j = 0 \\
\text{if} &\quad \text{Mann-Whitney}(P_i, P_j) \text{ says they are the same then} \\
\text{tie}_i &= \text{tie}_j + 1; \\
\text{else} &\quad \text{if} \quad \text{mean or median}(P_i) < \text{median}(P_j) \text{ then} \\
\text{win}_j &= \text{win}_j + 1 \\
\text{loss}_j &= \text{loss}_j + 1 \\
\text{else} &\quad \text{win}_i = \text{win}_i + 1 \\
\text{loss}_i &= \text{loss}_i + 1 \\
\text{end if} &\quad \text{end if}
\end{align*}
\]

The black triangles in Figure 5 mark when an estimator was one of the top-ranked methods for a particular data set. 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).

The columns \( k = 1, 2, 4, 8, 16 \) denote variants of standard ABE0 where estimates are generated from the \( k \)-th nearest neighbors.

- The columns \( k = \text{best} \) 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 \( \text{win} - \text{loss} \) calculation of Figure 6. We first check if two distributions \( i, j \) are statistically different according to

- 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 \( k = \text{best} \) 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.

\[
\begin{align*}
\text{MdmRE} & \quad \text{TEAK} \quad \text{LR} \quad \text{NNet} \quad k = \text{best} \quad k = 1 \quad k = 16 \quad k = 2 \quad k = 4 \quad k = 8 \\
\text{Pred(25)} & \quad \text{TEAK} \quad \text{LR} \quad \text{NNet} \quad k = \text{best} \quad k = 1 \quad k = 16 \quad k = 2 \quad k = 4 \quad k = 8 \\
\text{MAR} & \quad \text{TEAK} \quad \text{LR} \quad \text{NNet} \quad k = \text{best} \quad k = 1 \quad k = 16 \quad k = 2 \quad k = 4 \quad k = 8 \\
\end{align*}
\]

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 \( \text{win} - \text{loss} \) from a Mann-Whitney test, 95% confidence). The \( \text{Count} \) rows show the number of times a method appeared as the top performing variant. Results from [17].

\[
\begin{align*}
\text{win}_i &= 0, \text{tie}_i = 0, \text{loss}_i = 0 \\
\text{win}_j &= 0, \text{tie}_j = 0, \text{loss}_j = 0 \\
\text{if} &\quad \text{Mann-Whitney}(P_i, P_j) \text{ says they are the same then} \\
\text{tie}_i &= \text{tie}_j + 1; \\
\text{else} &\quad \text{if} \quad \text{mean or median}(P_i) < \text{median}(P_j) \text{ then} \\
\text{win}_j &= \text{win}_j + 1 \\
\text{loss}_j &= \text{loss}_j + 1 \\
\text{else} &\quad \text{win}_i = \text{win}_i + 1 \\
\text{loss}_i &= \text{loss}_i + 1 \\
\text{end if} &\quad \text{end if}
\end{align*}
\]

Figure 5: 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).
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 regression 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.

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.

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.

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).

\[
\text{Percentage} = \frac{\text{SubsetSizeInAnalogies}}{\text{SubsetSize}}
\]  

(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 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

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.
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 Si” where \(i \in \{1, 2, 3\}\) 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.

<table>
<thead>
<tr>
<th>Test Set</th>
<th>From S1</th>
<th>From S2</th>
<th>From S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1: cocomo81s (28)</td>
<td>1.0 (3.6%)</td>
<td>1.1 (4.8%)</td>
<td>1.6 (14.4%)</td>
</tr>
<tr>
<td>S2: cocomo81o (24)</td>
<td>1.8 (6.6%)</td>
<td>1.3 (5.6%)</td>
<td>1.1 (10.4%)</td>
</tr>
<tr>
<td>S3: cocomo81s (11)</td>
<td>1.4 (5.1%)</td>
<td>1.7 (7.0%)</td>
<td>1.0 (9.4%)</td>
</tr>
<tr>
<td>S1: nas93_center_1 (12)</td>
<td>1.0 (11.4%)</td>
<td>2.9 (7.9%)</td>
<td>1.7 (4.3%)</td>
</tr>
<tr>
<td>S2: nas93_center_2 (37)</td>
<td>1.6 (13.0%)</td>
<td>4.6 (12.4%)</td>
<td>3.8 (9.8%)</td>
</tr>
<tr>
<td>S3: nas93_center_3 (39)</td>
<td>0.8 (6.7%)</td>
<td>2.2 (6.0%)</td>
<td>2.1 (5.4%)</td>
</tr>
<tr>
<td>S1: desharnaisL1 (48)</td>
<td>2.5 (5.5%)</td>
<td>1.7 (7.0%)</td>
<td>0.8 (7.9%)</td>
</tr>
<tr>
<td>S2: desharnaisL2 (25)</td>
<td>2.6 (5.6%)</td>
<td>3.5 (6.1%)</td>
<td>0.7 (6.7%)</td>
</tr>
<tr>
<td>S3: desharnaisL3 (10)</td>
<td>1.9 (4.1%)</td>
<td>1.3 (5.0%)</td>
<td>0.4 (4.0%)</td>
</tr>
<tr>
<td>S1: maxwellHardware1 (8)</td>
<td>1.6 (9.1%)</td>
<td>1.6 (8.8%)</td>
<td></td>
</tr>
<tr>
<td>S2: maxwellAppType1 (17)</td>
<td>1.8 (8.2%)</td>
<td>1.6 (8.8%)</td>
<td></td>
</tr>
<tr>
<td>S3: kemererHardware23456 (8)</td>
<td>0.6 (8.8%)</td>
<td>0.9 (10.9%)</td>
<td></td>
</tr>
<tr>
<td>S1: maxwellHardware2 (17)</td>
<td>0.9 (8.8%)</td>
<td>0.8 (10.8%)</td>
<td></td>
</tr>
<tr>
<td>S2: maxwellAppType2 (29)</td>
<td>0.4 (3.7%)</td>
<td>1.8 (6.2%)</td>
<td>1.0 (5.5%)</td>
</tr>
<tr>
<td>S3: maxwellAppType3 (18)</td>
<td>0.6 (6.3%)</td>
<td>0.9 (3.2%)</td>
<td>1.0 (5.5%)</td>
</tr>
<tr>
<td>S1: maxwellHardware3 (16)</td>
<td>2.5 (6.8%)</td>
<td>0.8 (4.9%)</td>
<td>0.4 (6.0%)</td>
</tr>
<tr>
<td>S2: maxwellAppType5 (7)</td>
<td>2.3 (6.2%)</td>
<td>0.8 (5.0%)</td>
<td>0.3 (4.3%)</td>
</tr>
<tr>
<td>S3: maxwellHardware5 (7)</td>
<td>2.2 (5.5%)</td>
<td>0.8 (5.0%)</td>
<td>0.3 (4.5%)</td>
</tr>
<tr>
<td>S1: maxwellSource1 (8)</td>
<td>0.1 (10.0%)</td>
<td>2.8 (5.2%)</td>
<td></td>
</tr>
<tr>
<td>S2: maxwellSource2 (54)</td>
<td>0.4 (4.6%)</td>
<td>2.8 (5.2%)</td>
<td></td>
</tr>
</tbody>
</table>

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.

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.

#### 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 \(\frac{T}{N}=(7.9, 9)\%\) 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 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 geometric 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 geometries 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 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, while others favor both within and cross.
- Using other ABE or non-ABE methods under similar settings.
- Using different features on different datasets to see if they can define a border between within and cross data.

More generally, this work gives permission to effort analysts to search for data outside of their particular organizational context. One limit to such a search are the ontologies under which data is described at different sites; e.g.

- Is “LOC” the same as “size”?
- One site might use record data using some term like product complexity. Is there any analogous measure to product complexity at other sites?

Hence, one result of this paper might be to spawn a sub-field in effort estimation where researchers try to infer synonyms between data dictionaries at different sites.
Acknowledgements
This work was funded by a National Science Foundation grant CCF-1017330.

8. REFERENCES