Evaluating Methods for Effort Estimation Algorithm Ranking

Vincent Rogers, William Sica
CS, WVU, Morgantown, USA
vrogers@mix.wvu.edu,
wsica@mix.wvu.edu

Ekrem Kocaguneli, Tim Menzies
CS & EE, WVU, Morgantown, USA
ekocagun@mix.wvu.edu,
tim@menzies.us

ABSTRACT

Software project planning is a difficult task in which even small errors can lead to large financial consequences. As a result, there has been research conducted to assess new methods to perform software project planning. In this research, there is not a standard error measure or evaluation criteria. This paper seeks to test multiple algorithms from the literature using a combination of algorithms (COMBA) approach. Different error measures are tested on the combined algorithms to see if single error measures are reliable. The results indicate single error measures tend to be reliable, but that the dataset the error was measured on matters greatly, coinciding with a similar study done previously. Finally, future works using a combination of algorithms approach are suggested.

Categories and Subject Descriptors

Software Engineering [**Software Metrics**]: Data Mining

General Terms

Software effort Estimation, Analogy, MMRE, Evaluation Criteria

Keywords

Software Effort Estimation, MMRE, Evaluation Criteria

1. INTRODUCTION

Effort estimation is a crucial part of software project planning. Improper estimates can lead to over budgeting, delays, and project cancellation. In order to address the need for better software project effort estimates, the field of search based software engineering has been developed. [2] Search based software engineering uses data collected from previous projects to make inferences about new projects. Search based software engineering has grown since its inception, but there are still many open research questions. Harman suggests that promising areas of future work include hybridizing existing project planning methods, as well as analyzing the terrain of software project data. [1] This paper seeks to address both concerns, by employing a combination of algorithms. (COMBA)

ICSE 2011 Waikiki, Honolulu, Hawaii USA

COMBA combines different data preprocessors and learners used in search based software engineering, creating a set of combined algorithms which are a hybridization of different techniques. COMBA's approach also generates many different algorithms, allowing the data to be assessed from many view points at once. The behavior of different algorithms on a dataset can give hints as to its appearance and the terrain of the dataset.

Still, the search based software engineering research community does not have a consistent reporting method for the many different algorithms. In order to address this concern, this paper examines frequently used measures to evaluate algorithms for search based software engineering. It examines how different error measures rank algorithms, and how algorithms tend to perform across different datasets. This is to test if a standard is necessary in the literature, or if the results from different error measures are approximately equivalent. If a certain error measure is more indicative of an algorithms ability to make proper estimates then others, this could have implications on previously published research.

A previous study using COMBA indicated that error measures were approximately equivalent, but that certain datasets were better indicators of algorithm performance then others. This experiment is performed on a subset of COMBA from the previous study, with fewer algorithms and datasets. This is to see if the result is applicable on a smaller scale.

In Section 2, the motivation for the experiment and similar projects are discussed.

In Section 3, the experimental procedure is detailed, as well as the different algorithms implemented in COMBA, the error measures used, and the datasets used.

In Section 4, the results of the experiment are given and compared to the previous study this work was based on.

In Section 5, suggestions for future uses of the combination of algorithms approach are given.

2. RELATED WORK

For this section, the main focus is on the conclusion instability problem. Though this project does present many results, there has yet been sturdy evidence that it can give a definitive answer as to which data miner/learner is better than all the rest or maybe even tell what kind of situations. Our secondary source is the opposite of algorithmic learners but more so on the knowledge from a human exert in a field of study, and how the combination of the two yields a higher accuracy then either on an individual basis.

2.1 Sturdy Conclusion Predicament

This program attempts to determine which prediction model would be considered the best among all the others it's compared against.

Finding the right way to use data retrieved from a prediction model to acquire such a certain decision is a tough decision, and there are many ways to try and go about doing such a feat. Shepperd and Kododa [4] did such a study, stating that they think that as the dependence on technology is ever increasing the data sets used for data mining will, of course, also become bigger and thus more complex. Their study only used the prediction systems: regression, rule induction, nearest neighbor, and neural nets, but attempted on synthetically enlarged data sets. From those they hoped to ascertain the relationship of answer accuracy, the choice between prediction models, and characteristics of data sets.
They used various error measures as well, such as MMRE, and studied the information given to try and collect correlations. From their studies they concluded that they couldn't accurately pick a "best" data miner, but they did see a dependency of an estimate's accuracy on the characteristic of a data set and data miner being used.
Currently, a data set repository, called PROMISE, offers a large, growing selection of actual data sets of varying size, complexity, and subject. So, our project doesn't have to worry about artificially enlarging data sets.

2.2 Non-Algorithmic Method

Acquiring an estimate, used to be done only by humans with a vast data base of knowledge and experience in a certain field. But, now estimation over certain data sets is becoming increasingly hard AND that computers have been becoming increasingly better at "learning". A person by the name of Jorgensen [3] did a study only how well the two prediction methods work together. It was found that it was more likely to influence the accuracy of the estimate when the two were used together.
The problem with that kind of human knowledge is that it's not easily passed on, and it's hard to explain how some conclusions are reached. Though, the same problem of sometimes not being to explain how an answer was conjured is also a flaw of algorithmic estimation solutions.

3. EXPERIMENT DESIGN

Numerous methods exist to make estimations on effort. While this paper can not comprehensively assess all of them, it assesses a subsection of algorithms using a combination of algorithms (COMBA) approach. In this way, different data preprocessors are combined with different learner algorithms, so that a larger number of algorithms are generated with the addition of a new preprocessor or learner. The combination done involves sending the data to the preprocessor, and then sending that output to the learner to obtain an estimate.
Multiple methods are used within the literature to assess the accuracy of estimation across a dataset. As this paper seeks to evaluate these different measures, multiple measures of error will be used and compared. In addition, error measures that are a synthesis of multiple error measures will be assessed to see if a trend can be uncovered.
These different error measures will be compared using paired Mann-Whitney Wilcoxian statistical tests to determine which algorithms perform better and have a significantly different distribution. Algorithms can win, tie (if they have similar distributions), or lose with each possible comparison. Algorithms will be ranked on win and loss measures from the comparison.
The ranking will also be done across multiple datasets, to provide a broader reference as well as potentially gaining information about the datasets themselves. Which algorithms perform well on a given dataset could be indicative of that datasets terrain.
This section will discuss the preprocessors, learners, and error measures used as well as the data sets which they were used upon.

3.1 Preprocessors

Before being passed to the learners, the data was run through a preprocessor. Some preprocessors change the values of the data, and others change the shape of the dataset by converting it in to a representative model. The specific preprocessors used will be discussed below.

3.1.1 None

The data is passed to the learner without any preprocessing performed.
Abbreviation in results: none

3.1.2 Logarithmic

The features in the data are replaced with the natural log of their value. This reduces the distance between features, effecting many of the learners the data is sent to.
Abbreviation in results: log

3.1.3 Equal Frequency Discretization

The numeric data is discretized in to a number of bins, with each bin having an equal number of items, or frequency. The data is put in to bins based on numeric value. For example, a 2-bin frequency discretization on the data
$\{1, 4, 2, 8, 3, 9\}$
Would produce a bin $\{1, 2, 3\}$ and a bin $\{4, 8, 9\}$. Equal Frequency Discretization is performed using 3-bins and 5-bins in this experiment
Abbreviation in results: freq3bin, freq5bin

3.1.4 Equal Width Discretization

The numeric data is discretized in to a number of bins, with each width having an equal width of starting and ending values contained. The width of each bin is computed as:
$\frac{MaxValue - MinValue}{Numberof Bins}$
To provide an example,
$\{1, 2, 3, 4, 8, 9\}$
passed through a 2-bin Equal Width Discretization would produce a bin containing $\{1, 2, 3, 4\}$ and a bin containing $\{8, 9\}$. Equal Width Discretization is done in this experiment with 3 and 5 bins.
Abbreviation in results: width3bin, width5bin

3.1.5 Normalization

Each numeric entry in the data is replaced with a normalized value, computed as:
$\frac{Value - MinValue}{MaxValue - MinValue}$
Abbreviation in results: norm

3.1.6 Stepwise Regression

A stepwise regression is performed on the data. This removes values which do not fall with in a certain similarity tolerance in order to remove noise in the data.
Abbreviation in results: SWReg

3.1.7 Principal Component Analysis

Principal Component Analysis reduces the data to a set of features which are not correlated with one another.
Abbreviation in results: PCA

3.1.8 Sequential Filter Sampler

Sequential Filter Sampling filters the data in to a set of relevant instances, by applying a filter to the data, testing the changes, and applying a different filter to sample a subset of the overall data. This sample is then passed on in place of the dataset.
Abbreviation in results: SFS

3.2 Learners

After the data has been preprocessed, it is passed to a learner which makes an estimate on the effort required.

3.2.1 Stepwise Regression

Stepwise Regression is used as a learner as well as a data preprocessor. After making a model, the given project is placed in that model to predict for its effort value.
Abbreviation in results: SWReg

3.2.2 Simple Linear Regression

Simple Linear Regression applies n-dimensional linear regression on the data, attempting to determine a correlation of attributes that generate a given effort value. The instance to be tested is then placed along the regression, to estimate for its effort value.
Abbreviation in results: SLReg

3.2.3 Partial Least Squares Regression

Partial Least Squares Regression project the value being predicted for on to the known values to create a new hyperplane. The project being tested for is then placed on this hyperplane, and the predicted effort value observed.
Abbreviation in results: PlSR

3.2.4 Principal Component Regression

Instead of using all features besides effort to make predictions, Principal Component Regression reduces the space to a set of features with high variance. Regression is then performed on this space, and the unknown project is placed on the new space to predict for its effort value.
Abbreviation in results: PCR

3.2.5 Single Nearest Neighbor

Single Nearest Neighbor finds the project in the dataset that has the closest euclidean distance to the unknown project, and uses that projects effort value for the estimate.
Abbreviation in results: 1NN

3.2.6 Analogy Based Estimation

Analogy Based Estimation finds historical instances similar to the unknown instance. In the COMBA system used, Analogy Based Estimation finds the five nearest neighbors of the unknown project and uses their median as the estimated effort value.
Abbreviation in results: ABE0

3.3 Error Measures

For each dataset, leave one out analysis is performed. In this way, each instance in the dataset is removed to be tested under all preprocessor and learner combinations available, and the estimates stored. Once all estimates have been found, collective error measures are gathered for each combination of preprocessor and learner on each dataset. The error measures used are detailed below.

3.3.1 Mean Absolute Residual (MAR)

The absolute residual error of an estimate is computed as:
$|actual - predicted|$
After the absolute residuals have been calculated across a dataset, their mean is taken and reported for the error value.

3.3.2 Mean Magnitude of Relative Error (MMRE)

The magnitude of relative error is calculated across a dataset, computed as:
$\frac{|actual-predicted|}{actual}$
After the magnitude of relative error for each instance in the dataset is computed, their mean is reported.

3.3.3 Mean Magnitude of Error Relative to the Estimate (MMER)

The magnitude of relative error is computed, but in contrast to MRE it is computed relative to the estimate, as follows:
$\frac{|actual-predicted|}{predicted}$
After the MER is computed for each instance in the dataset, their mean is computed and reported.

3.3.4 Median Magnitude of Relative Error (MDMRE)

As MMRE, but the median of the MRE values across the dataset is computed and reported.

3.3.5 Pred25

The number of instances in a dataset whose predicted value had an MRE score of less than 25% are divided by the number of total instances, and reported as the Pred25 score.

3.3.6 Mean Balanced Relative Error (MBRE)

Balanced relative error is computed as :
$\frac{|actual-predicted|}{dividing}$
Where the dividing term is the smaller of the actual or predicted term. Once the BRE scores have been computed for each instance in the dataset, their mean is computed and reported.

3.3.7 Mean Inverted Balanced Relative Error (MIBRE)

Inverted balanced relative error is computed as :
$\frac{|actual-predicted|}{dividing}$
Where the dividing term is the larger of the actual or predicted term. Once the IBRE scores have been computed for each instance in the dataset, their mean is computed and reported.

3.4 Data Sets

The datasets used in this experiment were obtained from the PROMISE data repository, which provides freely available software engineering data from real world projects. The COMBA software platform used could not handle discrete elements in the data, so discrete elements were removed from the data before being sent to the data preprocessor. The information about each dataset, after removing discrete elements, is provided.

Dataset	Features	Instances
Cocomo81o	17	21
Cocomo81s	17	11
Finnish	8	38
Miyazaki94	8	48
DesharnaisL2	11	25
DesharnaisL3	11	10
Nasa93_center_1	17	12
Albrecht	8	24
Telecom1	3	18

4. RESULTS

This experiment, as stated earlier, utilizes paired combinations of 10 preprocessors and 6 learners on various 10 data sets. To ensure

that we have data with conclusive answers to test on, the leave-one-out method is used on each of its respective data set 7 times. Each repetition represents a different error measure (AR, MRE, MER, MdMRE, MRE, PRED(25), and MIBRE), each explained in the experiment specification section. These measures are used to calculate both wins and loss for both data sets and algorithms (preprocessor, learner combinations). The summed measures are organized with the fewest losses and, inversely, the most wins.

Though the results extracted from this follow up do vary slightly from the previous experiment that we have emulated, the results are essentially the same. All the sorted algorithms' performance can be seen in Figure Figure 2. From our experiment, the SWReg/1NN combination was ranked the best in the terms of fewest losses, and the PCA/SLReg combination was ranked the worst. These correspond to the previous study's results.

rank	preprocessor	learner		preprocessor	learner
1	SWReg	1NN	37	width5bin	1NN
2	none	PLSR	38	log	PCR
3	PCA	PLSR	39	width3bin	1NN
4	SFS	ABE0	40	SFS	SWReg
5	norm	ABE0	41	width5bin	PCR
6	none	ABE0	42	width3bin	PLSR
7	width3bin	SWReg	43	log	SWReg
8	freq5bin	ABE0	44	norm	PCR
9	log	1NN	45	log	PLSR
10	width5bin	PLSR	46	PCA	1NN
11	SWReg	ABE0	47	width3bin	PCR
12	norm	PLSR	48	SWReg	SLReg
13	norm	1NN	49	freq5bin	PCR
14	none	1NN	50	log	SLReg
15	SWReg	PLSR	51	freq5bin	SWReg
16	SWReg	PCR	52	freq3bin	PLSR
17	SFS	1NN	53	freq5bin	PLSR
18	width5bin	SWReg	54	freq3bin	PCR
19	PCA	ABE0	55	width3bin	SLReg
20	width3bin	ABE0	56	freq3bin	SWReg
21	PCA	SWReg	57	width5bin	SLReg
22	SFS	SLReg	58	freq5bin	SLReg
23	SWReg	SWReg	59	freq3bin	SLReg
24	PCA	PCR	60	Top2	MoEMean
25	log	ABE0	61	Top4	MoEMean
26	none	PCR	62	Top8	MoEMean
27	norm	SWReg	63	Top16	MoEMean
28	none	SWReg	64	Top2	MoEMediar
29	width5bin	ABE0	65	Top4	MoEMediar
30	SFS	PLSR	66	Top8	MoEMediar
31	SFS	PCR	67	Top16	MoEMediar
32	freq3bin	ABE0	68	Top2	MoEIrwm
33	none	SLReg	69	Top4	MoEIrwm
34	norm	SLReg	70	Top8	MoEIrwm
35	freq5bin	1NN	71	Top16	MoEIrwm
36	freq3bin	1NN	72	PCA	SLReg

Figure 1: All permutations of data miners and learner pairs, ordered from least amount of loss measures to the most.

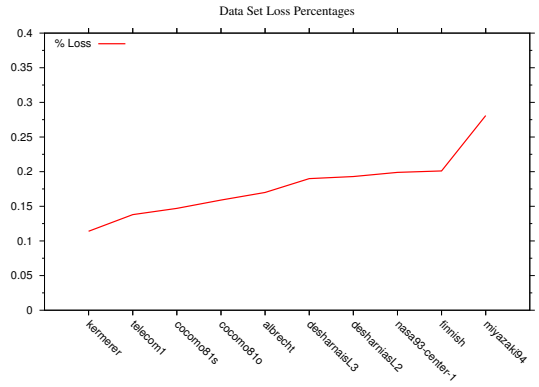

Figure 2: The seventy two algorithms, sorted by their percentage of maximum possible losses (so 100% = 4970).

To get proper percentages of loss over the algorithms, the sum of losses for one algorithm over all error measures is divided by the total possible losses which is the counts of:

$$comparisonsmade * errormeasures * datasets =$$

$$71 * 7 * 10 = 4,970$$

The results of such findings are located in Figure Figure 3. Which resemble the figure of Algorithm Losses provided by the paper ICSE paper.

As for the percentages of Loss over a Data Set, a very similar equation is used where all numbers gathered for a data set are summed together with all data collected for each error measure. That is then divided by the total possible amount of losses for that data set which is the count of:

$$comparisonsmade * errormeasures * algorithms =$$

$$71 * 7 * 72 = 35,784$$

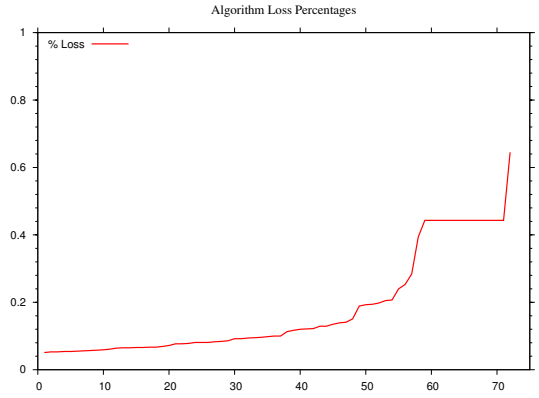

Figure 3: The seventy two algorithms, sorted by their percentage of maximum possible losses (so 100% = 5040).

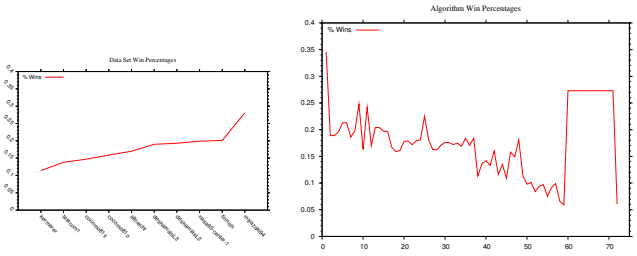

Figure 4: Algorithms and datasets, sorted as per but this time showing their percentage of maximum *wins* over all performance measures.

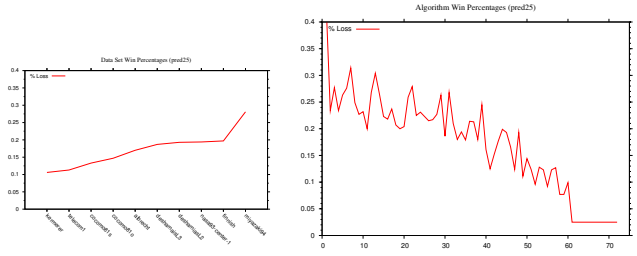

Figure 5: Algorithms and datasets, sorted as per Figure 2 and Figure 3, but this time showing their percentage of maximum *wins* over *just the PRED(25)* performance measures.

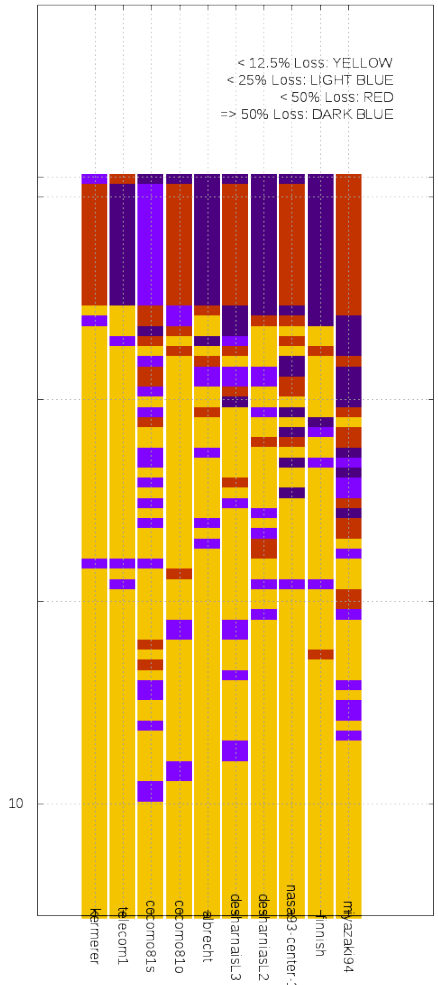

Figure 6: The 20 data sets and 72 methods are expressed on this graph in percentages of maximum possible losses for one algorithm for one data set = # of Error Measures * # of comparisons made. (100% = 497, 50% = 248, 25% = 124, 12.5% = 62) and based on these number both the columns and rows are sorted by their sum.

While the visible trend may seem convincing, it beckons for more evidence. Unfortunately, when the opposite graph to the Losses Figure 2 is constructed in the Wins Figure 4, the trend is very jagged and hard to get an accurate reading. These graphs are are similar to the Wins Figures in the Ekrem paper, and demonstrate the same need for more empirical studies. The Wins Figure 5 demonstrates that there is correlation between the error measure values gathered, if only one.

5. CONCLUSION

The results in the experiment confirm the results found in the previous study using COMBA, even on a smaller subset of the algorithms and data. Certain data sets are better predictors for algorithm success in effort estimation, and error measures tend to produce approximately equal rankings.

Given the repeated success of COMBA, future works using a com-

bination of algorithms approach or improvements that could be made to the existing system will compromise the remaining portion of the paper.

5.1 Future Work

There are still many open questions about evaluation methods for ranking search based software engineering algorithms. A few ways in which the experiment could be expanded are provided below.

5.1.1 More Algorithms, More Datasets

The 72 algorithms covered in this paper are only a small subset of the available algorithms for search based software engineering. While data mining and artificial intelligence are relatively young fields of computer science, they have already generated hundreds of algorithms which can be combined in thousands of ways. The Combination of Algorithms approach could be extended to include a more diverse range of algorithms, and scalability of the results could be assessed. Additionally, there are more datasets available for effort estimation in the PROMISE data repository. These additional datasets could be included and tested, to see if initial results hold across a larger range of datasets.

5.1.2 Open Source Implementation

The experiment was performed using software which referenced proprietary Matlab libraries. It is not portable to machines without Matlab, and Matlab is not freely available. There are many data mining and statistics packages available which are open source, such as R and Weka. An interface could be built which links the available algorithms in the packages together and performs statistical analysis on the results.
R is similar to Matlab in that it is a language for statistical manipulation. A program could be written in R to perform the same function as the current COMBA system.
Weka is an open source Java package with a visual and command line interface. The Weka algorithms can be accessed either through shell script on the command line or imported and accessed in Java code. The advantage of shell script is that if algorithms outside of Weka, possibly written in other languages, can also be accessed. The statistical analysis available in the current Matlab COMBA could be accessed through a shell script as well. The disadvantage is that management becomes more difficult with shell scripting. A program which references several different programming languages can be hard to make modifications for or fix if errors occur.

5.1.3 Incremental Results

Currently, all methods must be run every time analysis is performed. The code could be changed such that the raw results and error measures are saved, and a program links multiple results files and performs Mann-Whitney Wilcoxian tests on results. This has the advantage of shorter turn-around time for each new algorithm. A new algorithm need only run on a subset of the data available, and then could be compared to a database of previous results immediately. The program and results data could also be released as a public resource, if the software was open source as the PROMISE datasets are freely available. Users could create results in a COMBA format, and pass them to the statistical comparison program.
This could even be done on a website, which has the advantage that each set of results benchmarked would contribute to the overall amount of results stored in the system.

5.1.4 Other Fields of Search-Based Software Engineering

This paper only examined the method of effort estimation, while many algorithms exist for other aspects of project planning. The combination of algorithms approach could be applied to tasks such as defect prediction and cost optimization. On a broader scale, the combination of algorithms method could be used to evaluate algorithms from domains other than search based software engineering. It could be assessed as a separate tool for revealing patterns in data.

5.1.5 Discrete Data is Not Handled

The current project did not handle discrete elements in the data, so a different software package could be used which does handle discrete elements in the data. It could be observed whether or not the inclusion of discrete data effected the results or the rankings. How algorithms deal with discrete elements is frequently omitted in academic papers, so if it has a concrete effect this would be a useful result to share.

5.1.6 Performance Analysis

No reason was given as to why the algorithms which performed well did so. A further study could be done in which it was observed for which type of datasets algorithms performed well, and the properties of the data that effect their performance. This has the benefit of not requiring domain specific software engineering data. Any n-dimensional space could be analyzed, as many of the algorithms used in search-based software engineering do not make domain specific assumptions about the data. If COMBA is run on a large number of algorithms and datasets, a large database of results becomes available. This makes possible rapid testing and verification by comparing to existing results stored from previous runs.
These results could be used for suggesting algorithms given a dataset. Similarity measures between datasets could be evaluated, and high-performing algorithms from similar datasets could be suggested without having to test them on the dataset. The results of the recommendation system could then be compared to the actual results of running the possible algorithms. While neural nets and other stochastic optimization processes are typically associated with poor runtimes, a large database provides a terrain for optimization in which no complex algorithms need to be run. The results would already be precomputed, so verification could be done within reasonable time bounds. A study could also be done where results are incrementally added to the database, and the optimization effectiveness is gaged on number of datasets. It would be interesting to see if over fitting occurs with a few datasets, or if that too many datasets could flood the optimizer so that results are too general to be applicable.

5.1.7 Algorithm Tweaking

Many algorithms used within the COMBA system, such as Neural Nets and Stepwise Regression can be run using a variety of different settings. In addition to more algorithms, algorithms can be run with different specifications. Many times, specifications for algorithms are decided using ëngineering judgment ,̈ rather than empirical methods. Sometimes this involves citing another paper in which a particular setting was used, even if that paper did not test other settings. The diverse range of data sets tested for might show that certain settings tend to perform better, and provide a basis for further research using those methods.

5.1.8 Algorithm Building

The results of the COMBA experiment could be used to build a new algorithm whose only goal is to rank well compared to existing algorithms. It could go through iterative optimization to maximize

its ranking, perhaps using a genetic programming approach with a lambda calculus. Datasets could then be introduced that the algorithm did not optimize on, to see if its performance over fitted to the existing data or whether the optimized algorithm uncovered useful information about estimation.

5.1.9 Real Time Readjustment

A combination learning approach could be applied in an incremental fashion on data whose results depend on predictions made by a given algorithm. For example, suppose a real-time simulation of a company is performed using estimates produced by algorithms which performed well on previous data. Changes in performance can be taken in to account, so that multiple time steps are available and can be treated as different datasets. This allows back propagation for algorithm recommendation and evaluation, and for certain applications can uncover more details about the underlying assumptions of data generation.

5.1.10 Synthetic Data Production

While PROMISE provides a collection of software engineering data, there is still a large portion of data which is company specific and private. A problem in creating a synthetic database is whether or not it is representative of the given domain. While data could be randomly generated and algorithms scramble to try and find patterns within that data, perhaps a smarter approach to data production could be produced.

The advantage of doing this within the COMBA system is benchmarking synthetic data with actual data. Synthetic data can be ranked, and patterns seen in the other datasets, such as which algorithms performed well, can be compared. It is possible that some companies with private data have many more entries then the available datasets. Search based software engineering typically relies on datasets with fewer than 100 instances. Synthetic data could be created of a larger size that tries to extrapolate and expand patterns seen in smaller datasets.

5.1.11 Rank On Other Dimensions

It should be noted that only the error measures provided were used to assess the given algorithms. There are many other factors on which an algorithm can be evaluated, such as Big-O notation or empirical runtime results. Future tests could collect the time taken to evaluate each dataset, and provide this along with error measures. This could also be used as a complexity measure, to assess if complex and time-consuming algorithms increase performance or not.

5.1.12 Combination of Data

This paper combined different learners and assessed the combined results. Manipulations can also be performed on the datasets themselves. For example, datasets from different companies could be combined together, and the results of the algorithms on these new combined datasets could be assessed.

Combining datasets introduces the problem that there is not a industrial standard for collecting information about software engineering projects. Design decisions would come up such as how to treat features which are represented in one dataset but not another when combining datasets. Different approached for combination could be evaluated.

5.1.13 Domain Specific Knowledge Acquisition

One aspect of the datasets has a real world value corresponding to it. Other papers have proposed feature weighting, in which certain aspects of a project are more important than others for predicting effort. Domain inspecific ways, such as genetic algorithm optimization, have been proposed to find the best set of feature weights. A broad approach could look at the application of certain features across different datasets. For example, lines of code is generally regarded to be a good indicator of project effort, though it itself often requires estimation to obtain. This null hypothesis could be tested, by stripping lines of code from the dataset and viewing the results. This approach could be made more general by removing features to test their effects on estimation accuracy using different algorithms.

5.1.14 Data Visualization

The human heuristic is often a powerful one for quickly assessing patterns in visual models. One idea would be to create a visual representation of a dataset, and then present it to an expert. The expert could be asked to identify what types of algorithms they think would perform well on a dataset given its shape and attributes. The experts predictions can then be indexed with the actual rankings, to score a visualization technique. There may be a simple visualization that allows an expert to make complex decisions quickly. This also allows new datasets to be visualized, and the expert to make judgments on this new data without consulting all possible algorithms.

This also has the benefit of potentially explaining why certain algorithms or datasets performed well, discussed earlier in this section.

5.1.15 Experiment Verification and Validation

COMBA provides an environment to reproduce results published in other papers. While no algorithm can be shown to be the absolute best, as seen in the No Free Lunch theorem, an algorithm can be shown to be reasonably good. Experiments which relied on a limited number of datasets or algorithms to compare to could be reproduced. A possibility would be reproducing multiple experiments in one paper, and noting if the assumptions in the initially published result held upon more intensive testing.

More ambitiously, if the COMBA system was freely available it could be housed and recommended for use. In this way, researchers in other areas could download COMBA and quickly validate their results. There are more ways to expand this project then are feasible given the time-scale, so this approach provides a system which will build itself. The more people user, the stronger the results produced from it become as new algorithms and datasets become contributed to the COMBA system.

5.1.16 Pattern Matching Tested Algorithms

While this paper directly combined large-scale programming constructs together, the learns and preprocessors used could further be broken down in to their components. There are some programming languages, such as LISP, in which the components of a program are highly transparent and thus modifiable. A potential experiment could be to reprogram algorithms in a language which is transparent, and perform similarity analysis between high-performing algorithms and low-performing algorithms.

This creates interesting problems, such as how to identify when two constructs are doing the same thing but in different ways. This can be solved by pattern matching, or extrapolating the function of a code from a series of commands. The time complexity in these scenarios can be a hindering factor, especially when iterative development and optimization processes are required to obtain a functioning system.

6. REFERENCES

[1] M. Harman. The current state and future of search based software engineering. *Future of Software Engineering*,

0:342–357, 2007.

[2] M. Harman and B. F. Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833 – 839, 2001.

[3] M. Jorgensen and S. Grimstad. Over-optimism in software development projects: The winner#146;s curse. pages 280–285, 2005.

[4] M. Shepperd and G. Kadoda. Comparing software prediction techniques using simulation. *Software Engineering, IEEE Transactions on*, 27(11):1014 –1022, nov 2001.

APPENDIX

The authors of this paper would like to note that it is a continuation of an existing study by Jacky Keung, Ekrem Kocagunelli, and Tim Menzies. The paper describing that study is not yet published or publically available, so no link or reference can be provided at this time.