Quantitative Performance Prediction for Rule-Based Expert Systems*

Valerie Barr
Department of Computer Science
Hofstra University
Hempstead, NY 11550
vbarr@magic.hofstra.edu

Abstract

Often a rule-based system is tested by checking its performance on a number of test cases with known solutions, modifying the system until it gives the correct results for all or a sufficiently high proportion of the test cases. However, the performance on the test cases may not accurately predict performance of the system in actual use. We present a method for making a more accurate performance prediction based on the performance on test cases, coverage of the rule-base by the test cases, how representative the test data is of the population on which the rule-base will be used, and the likelihood of occurrence of different kinds of test cases in the larger population.

Introduction

Over the last 20 years, during which time there has been considerable development and use of knowledge-based systems for medical decision support, there has been a heavy emphasis on functional analysis, addressing, among others, the question: does the system give the results we expect on the test cases? Generally the testing process would lead to a statistic indicating the percentage of the test cases for which the system performed correctly. System performance statistics are then presented as if they apply to the entire rule-base, rather than just to the tested sections, which can lead to false predictions of system performance in actual use. However, in actuality the system performance indicated by the comparison of actual and expected results is relevant only for the tested sections, while performance in the untested sections cannot be predicted.

Functional testing does not guarantee that all parts of the system are actually tested. If some section of the rule-base is not exercised during the functional test then we do not have any information about that section of the system and whether it is correct or contains errors. Furthermore, many performance problems for rule-bases are the result of unforeseen interactions between rules (O'Keefe & O'Leary 1993). A test suite of known cases may never trigger these interactions, though it is important that they be identified in order to correct them before a system is put into actual use.

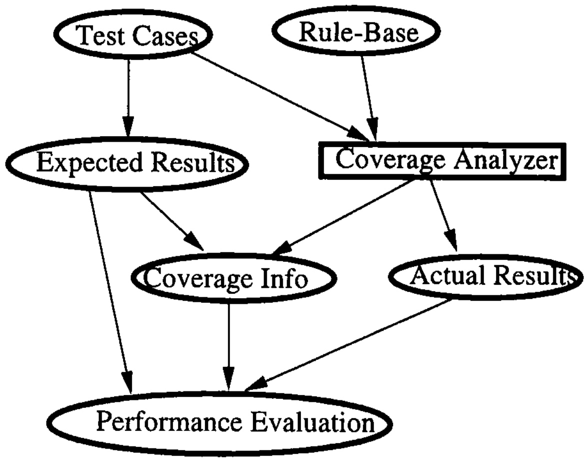

Figure 1: Rule-Base Evaluation with Coverage Analysis.

In order to generate a more accurate performance prediction, we take into account several factors. First, we need to identify which portions of the rule-base are actually exercised, or covered, during the testing process. This can be accomplished by enhancing the functional analysis of the rule-based system with a *rule-base coverage* assessment. In such an approach (Barr 1996) we consider completeness of the test set and coverage of the rule-base by the test data, as indicated in Figure 1.

Completeness of the test set refers to the degree to which the data represents all types of cases which could be presented to the system under intended conditions of use. *Coverage* of the rule-base refers to how extensively possible combinations of inference relations are exercised during test data evaluation. In the trivial case, with a correct rule-base and a complete test suite, the test data would completely cover the rule-base, all actual results would agree with expected results, and we could predict completely correct performance of the rule-base in actual use. In the more usual situation we may have errors and incompleteness in the rule-base, as well as inadequacies in the test data. If we judge the system based only on a comparison of actual and expected results, we could have a situation in which a rule-base performs well on the test data, but actually contains errors which are not identified due to incompleteness of the test data. This could lead to a false prediction of correct performance in actual use, when in fact we cannot make any accurate prediction about performance of the rule-base in those areas for which there is an absence of test data.

Frequently, when testing classification systems, a large population of cases is available. However, many of these cases may represent situations which are easy to classify and similar to each other. Furthermore, running all available cases may be extremely time consuming for a large classification system. A random selection of test cases may give statistical confirmation that the system works properly for the tested situations, but may not cover all types of situations. Our approach carries out structural analysis of the rule-base using five rule-base coverage measures to identify sections not exercised by the test data.

This testing approach allows for clear identification of both incompleteness in the test data and potential errors in the rule-base through identification of sections of the rule-base that have not been exercised during functional test. This can indicate either weaknesses in the test set or sections of the rule-base that may not be necessary or are incorrect. An incomplete test set can be supplemented with additional cases chosen from the available population, guided by a series of heuristics and the coverage analysis information (Barr 1997). This makes it possible to improve completeness of the test suite, thereby increasing the kinds of cases on which the rule-base has been tested. Alternatively, if there is no test data which covers certain parts of the system, it is possible that those sections should not remain a part of the system at all.

Testing with Rule-Base Coverage Measures

Rule-base testing with coverage measures is based on a graph representation of the rule-base, using a directed acyclic graph (DAG) representation. We assume a generic propositional rule-base language (Barr 1996) into which other rule-base languages can be translated. During construction of the DAG, pairwise redundant rules, pairwise simple contradictory rules and potential contradictions (ambiguities) are identified. After DAG construction is complete, static analysis (verification) of the rule-base reports dangling conditions (an antecedent component that is not defined as a finding and is not found as the consequent of another rule), useless conclusions, and cycles in the rule-base. At this point the rule-base can be modified to eliminate or correct any static problems.

The static analysis phase is followed by dynamic analysis of the rule-base using test cases. As test cases are processed, one or more of several rule-base coverage measures (RBCMs) can be reviewed in order to determine the quality of the test data supplied thus far. Additional information about the rule-base and its testing can also be used by the system tester to guide the selection of additional test data. The tester would start by providing sufficient test data to satisfy the simplest functional measure (conclude each class of the system) and proceed to the more difficult structural measures. Finally, if the user is not able to provide sufficient data to attain the desired degree of rule-base coverage (according to the selected criterion), the tester can use the DAG representation to synthesize data, which can then be reviewed by an expert to determine if the data represents a valid case in the problem domain.

This testing approach, described more fully in (Barr 1996), has been implemented in the TRUBAC tool (Testing with RUle-BAse Coverage) (Barr 1996; 1995).

A Metric for Rule-Based Systems

While the coverage measures can be very helpful once we are actually engaged in the process of testing a rule-based system, we would also like to have a quantitative measure of how successful the testing process has been from a coverage standpoint.

The graph representation we employ serves as a suitable foundation for a complexity metric to predict or measure the complexity of the system and the success of the testing process. It imposes no particular execution order on the rules, and it does represent all the logical relations that are inherent within the rule-base. The remaining issue to address is that of a suitable complexity metric over this graph structure. Metrics such as McCabe's cyclomatic complexity metric cannot adequately determine the number of execution paths in a rule-base. The actual number of execution paths is based on the logical relationships in the rule-base, using the following mechanism:

- For each class node, count one path for each edge into the class node. In Figure 2 there are 2 paths based on in-edges at class node G.

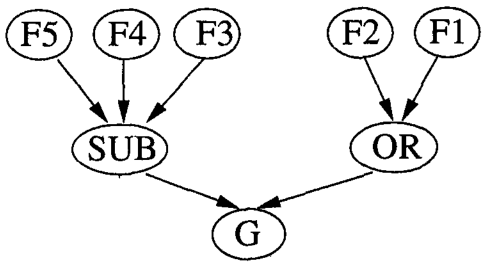

Figure 2: Graph of rule-base with OR and SUB.

- For each OR node, count one additional path. In Figure 2 we count one more path to G at the OR node.
- For each SUB node (representing an intermediate hypothesis or sub-class), consider the number of parent edges. As with an OR node, each parent edge represents a possible path which concludes the subclass. One of these paths will be counted when the classes are considered, and we count additional paths for the additional parent edges. In Figure 2 there are a total of 5 paths to G: 2 at G, 1 more at the OR node, and 2 more at the SUB node.
- for AND nodes no additional paths are added.

This execution path metric can serve a number of purposes in rule-base development and analysis. The total number of execution paths represents the maximum number of test cases needed for complete coverage of the rule-base according to the strongest rule-base coverage measure (**All-edges**). However, usually the actual number of data sets needed will be less than the number of execution paths, since often, particularly in diagnosis systems, one test set may cover a number of execution paths to different diagnoses. For prediction of future performance, this metric allows us to quantify which portion of the system has not been covered during the testing process.

Performance Prediction

Quantitative performance prediction is based on

- performance of the system on test cases,
- a measure of how well the test data covers the rule-base,
- a measure of the degree to which the test set is representative of the population for which the system is intended,
- how likely different test cases are in the population.

Assumptions

In order to clarify and simplify, we first make some assumptions about the system under test, the test data, and the general population. We distinguish between

- GP, the general population,
- TP, the pool of data that is available to us as potential test cases
- TC, the set of test cases that are run through the system under test

Note that $TC \subset TP$, where TP is generally large and repetitive, so that it is not feasible to run all the cases in TP through the rule-base. We then assume that

1. it is possible to arrange the cases in TP into equivalence classes, such that each equivalence class contains one kind of test case, and all test cases in an equivalence class will execute the same path within the rule-base.
2. TC is created by selecting one test case from each of the equivalence classes formed in TP
3. Therefore, each test case in TC corresponds to precisely one path in the rule-base.

We note that the third assumption implies that if the rule-base incorporates logic for all possible scenarios in the general population, then the degree of representativeness and the degree of coverage of the rule-base should be the same.

We also will use the following notation

- R represents the percentage of distinct types of cases in GP which are represented in TP and TC
- P represents the performance of the rule-base on the test data (percentage correct)
- Pr represents the performance predicted in actual use
- C represents the degree of path coverage achieved by the test data (in percentage of paths covered)
- t_i represents the i^{th} test case
- l_i represents the likelihood of occurrence in GP of a case that will use the same inference chain as is used by test case t_i

General Problem

Consider the situation in which we run N test cases through a rule-base and performance is 80% correct. Even if we have complete coverage of the rule-base (every path is traversed during execution of the test data), we can not predict 80% performance in actual use. If we have full path coverage then we presume that the test data was fully representative of the actual population. However, we still must consider the likelihood of occurrence in the population of the cases that were handled correctly by the rule-base and those that were handled incorrectly. If 95% of cases in the actual population will be handled by the part of the system that works correctly, then we could predict performance which will be better than the 80% accuracy achieved by the test data. On the other hand, if only 50% of cases in the actual population will be handled by the part of the system that works correctly, then we could predict performance that will be much worse than the 80% accuracy achieved by the test data.

In general we expect that, while we will not achieve complete coverage of the system, the section that is covered will correspond to the most likely situations in the population. The portions of the system that are not covered during testing will generally correspond to the least likely situations in the population precisely because it is much more difficult to find test data for cases which are rare in the general population.

Next we consider a somewhat more complicated case. Assume we still have 80% correct performance of the system on the test data. However this time the test data covers only 75% of the paths in the rule-base, and the test data is representative of only 75% of the possible cases that can occur in the general population. However, those cases are likely to occur 90% of the time in the general population. (That is to say, if we listed the distinct kinds of possible situations, 75% of them are represented in our test cases. However, if you selected 100 random cases from the general population, 90% of them would correspond to the test cases we have). This implies that in actual use, 10% of the cases that will be presented to the system are from the pool of possible cases (25% of the possible cases) that were not represented by our test data. If we want to generate a safe lower bound on performance, then we have to assume that these cases will operate incorrectly in actual use of the system, since the system was never tested on them. We would like to be able to generate an actual performance prediction for this kind of situation, which we expect to be some function of the performance on the test data, the coverage of the rule-base, the representativeness of the test data and the likelihood of the cases represented by the test data.

Total Path Coverage

We return to the simple situation, with P = 80% correct performance, C = 100% coverage of the rule-base, and R = 100% of the kinds of cases in GP represented by the test data. Let us further assume, for this scenario, that each test case has the same likelihood of occurrence in GP. In this situation we would predict performance in actual use that was the same as the performance on the test data.

If we leave all other conditions the same but assume that the test cases do not have the same likelihood, then the performance prediction changes somewhat. Assume that we have 10 test cases $(t_1 \dots t_{10})$, of which 8 are handled correctly by the system, for P=80%. Further assume that t_1 and t_2 have likelihood of 15% each, t_9 and t_{10} have likelihood of 5% each, and all other cases have likelihood of 10%. We can compute the performance prediction by simply computing the sum

$$\sum_{i=1}^{10} l_i * c_i$$

where l_i is the likelihood of the ith test case and c_i is 1 if the actual result agreed with the expected result for the ith test case and is 0 otherwise. We can easily see that if the system performs incorrectly on t_9 and t_{10} then we can predict performance that is 90% correct although only 80% of the test cases were handled correctly. Similarly, if the system performs correctly on all cases but t_1 and t_2 then we predict performance that is 70% correct in actual use of the system, lower than the 80% performance on the test data.

By assuming a one-to-one correspondence between test cases and paths in the rule-base, we can shift the likelihood figures onto the paths. Then, in the simple scenario in which all paths are executed, we can simply sum the likelihood values for all paths for which the answer given by the system was correct and use the resulting value as the performance prediction. Usually, however, we expect that not all paths will be executed.

Incomplete path coverage

If not all paths are executed by the test data, then the maximum safe performance prediction is based on the assumption that any path not executed during testing will give an incorrect result in actual use of the system.

Consider a scenario in which TC represents 75% of the kinds of cases possible in GP (R = 75%). Given our assumptions above, we then expect coverage of 75% of the paths in the rule-base during testing of the system (C=75%). Let us further assume, as before, that the actual answers generated during testing are correct for 80% of the test cases (P=80%). If all cases in TC

are equally likely then, in actual use of the system, we predict 60% correct performance. This is based on the fact that out of 100 random cases selected from GP, 25 of them will be wrong because they use the untested portion of the rule-base, and an additional 15 will be wrong because they utilize the tested portion of the rule-base which gave incorrect results during testing.

Next we consider that not all cases are equally likely. Assume, with no loss of generality, that there are 100 equivalence classes (types of cases) in GP, and a corresponding 100 paths in the rule-base. Further assume we have only 75 test cases representing 75 of these 100 equivalence classes, and that these 75 cases are more likely and are found in the population 95% of the time. That is, out of 100 random cases from GP, 95 of them will fall into the 75 equivalence classes represented by our test cases, with multiple cases falling into some equivalence classes. Examples of the remaining 25 equivalence classes are found in only 5 out of 100 random cases.

In this situation we can think of the rule-base as being unevenly divided as follows:

- RB_{NC} represents the portion of the rule-base that was not covered during testing, and would presumably handle the 25 kinds of cases that were not included in the test set. We expect this portion of the rule-base to fail in actual use
- RB_C represents the portion of the rule-base that handles the 75 kinds of cases that are included in the test set. This section is further divided into
  - RB_{CC} represents the portion of the rule-base that correctly handles a subset of the 75 kinds of cases included in the test set
  - RB_{CI} represents the portion of the rule-base that incorrectly handles a subset of the 75 kinds of cases included in the test set

Out of 100 cases drawn from GP, 5 would be handled by RB_{NC}, and we expect incorrect answers for those, while 95 would be handled by RB_C. We assume the existence of an oracle that determines if the result given by the rule-base for a test case is correct or that we have *a priori* knowledge of the expected result for each test case. We also assume that we know the likelihood value for each of the paths in RB_C. Given our earlier assumption of a one-to-one correspondence between test cases and paths, this is really a likelihood that each path will be executed.

Let P_c be the number of paths in RB_C. Each path in RB_C has a likelihood l_i based on how popular the corresponding case is in GP. Therefore our performance prediction for RB_C (and prediction of correct performance overall by the system) is

$$\sum_{i=1}^{Pc} l_i * c_i$$

where, as before, c_i is 1 if the actual result equals the expected result along that path and is 0 otherwise.

To demonstrate how this computes a performance prediction, we consider two scenarios.

Scenario 1 We first consider a situation in which R=75%, C=75% and P=80%. Furthermore, we let the total likelihood that a case will be handled by RB_{NC} be 5%, with 95% likelihood that a case will be handled by RB_C. Assume that P_a=100 (there are 100 paths total in the rule-base) and that one path represents an extremely popular case, say 10% of the entire population. That is, 10% of all cases run through the system will execute this particular path. We also assume that the system does give the correct answer for the popular case (it is in RB_{CC}).

Out of 100 random cases from the population at large, GP, 5 run through RB_{NC} (which contains 25 paths) and, presumably, give a wrong answer. The remaining 95 cases run through RB_C (made up of 75 paths). The test cases gave 80% correct answers and were run only through RB_C. Given our assumption of a one-to-one correspondence of test cases and paths, this means that, of all paths in RB_C, 60 will give a correct answer (80% of 75 paths). Therefore, of the 95 random cases handled by RB_C, 77.85 will actually be handled correctly, or we predict 77.85% correct performance of the rule-base in actual use.

We can see this as follows: The 75 paths in RB_C have a total likelihood of execution of 95% out of 100 test cases, of which 10% belongs to one path and 85% is divided equally among the 74 remaining paths. So the likelihood of execution of each of the 74 paths is 1.15%. Another way of viewing this is that of the 60 paths that give a correct result during testing, during actual use one path will handle 10% of cases and each of the remaining 59 paths will handle 1.15% of cases. Therefore, in actual use we expect a correct result in

$$1.15 * 59 + 10 * 1$$

cases, or 77.85 out of 100 cases, or 77.85% correct performance.

Scenario 2 In this case we leave the above scenario unchanged except that we assume that the popular case is tested but is handled incorrectly by the rule-base (the path for the popular case is in RB_{CI}). Therefore we predict correct performance along 60 paths, each of which has a likelihood of execution of 1.15%, for a performance prediction of approximately 69%, significantly lower than the 80% that might be inferred if we simply looked at the 80% correct performance on the test cases.

Conclusions

It is clear from this work that the actual performance we can expect from a rule-based system in real use can be significantly different than its performance on a set of test data. A benefit of the method discussed is that we can get the performance prediction without running repetitive test cases. We avoid the approach of running similar cases in order to get statistical confirmation. If a case executes correctly, then we use its likelihood of occurrence in the general population to generate the performance prediction figure. Using the information about coverage, number of paths, and representativeness in the fashion described allows us to use limited data about the population and the system under test to compute a safe performance prediction figure.

References

Barr, V. 1995. TRUBAC: A tool for testing expert systems with rule-base coverage measures. In *Proceedings of the Thirteenth Annual Pacific Northwest Software Quality Conference.*

Barr, V. 1996. *Applications of Rule-Base Coverage Measures to Expert System Evaluation.* Ph.D. Dissertation, Rutgers University.

Barr, V. 1997. Rule-base coverage analysis applied to test case selection. *Annals of Software Engineering* 4.

O'Keefe, R., and O'Leary, D. 1993. Expert system verification and validation: a survey and tutorial. *Artificial Intelligence Review* 7:3–42.