H1.9.1 Telecommunications Network Diagnosis

Andrea Pohoreckyj Danyluk
Williams College
Department of Computer Science
Williamstown, MA 01267
andrea@cs.williams.edu

Foster Provost
New York University
New York, NY 10012
fprovost@stern.nyu.edu

Brian Carr
Network Systems–Advanced Technologies
Bell Atlantic
White Plains, NY 10604

December 8, 1999

Abstract

The Scrubber 3 system monitors problems in the local loop of the telephone network, making automated decisions on tens of millions of cases a year, many of which lead to automated actions. Scrubber saves Bell Atlantic millions of dollars annually, by reducing the number of inappropriate technician dispatches. Scrubber's core knowledge base, the Trouble Isolation Module (TIM), is a probability estimation tree constructed via several data mining processes. TIM currently is deployed in the Delphi system, which serves knowledge to multiple applications. As compared to previous approaches, TIM is more general, more robust, and easier to update when the network or user requirements change. Under certain circumstances it also provides better classifications. In fact, TIM's knowledge is general enough that it now serves a second deployed application. One of the most interesting aspects of the construction of TIM is

that data mining was used not only in the traditional sense, namely, building a model from a warehouse of actual historical cases. Data mining also was used to produce an understandable model of the knowledge contained in an earlier, successful diagnostic system.

H1.9.1.1 Project overview

This case study describes a data mining application that has resulted in two deployed network diagnosis systems for Bell Atlantic, combining to make tens of millions of classifications annually, many of which result in automated actions.

As is described in detail below, the Scrubber 2 system diagnosed and acted on faults in the local loop of the telephone network. Our job was to build a knowledge base for the diagnostic system's successor, Scrubber 3. The new system is based on a centralized knowledge server, and covers twice as large a network as the old.

The primary requirement for the system was comprehensibility: the understanding and acceptance by domain experts of how the system makes its decisions. Along with this comprehensibility requirement went a required minimum level of predictive performance—namely, the new system should be at least as good as the prior system. We had additional desiderata, including improving predictive performance, improving extensibility, and reducing dependence on weakly supported cost assumptions. We must stress that these desiderata were seen as far lower priority than the primary requirement.

In fact, to some key players a direct port of the existing system was the optimal solution (to which our desiderata were in direct opposition). However, as we describe below, the knowledge inside the existing system was far from comprehensible itself, so a direct port was not a straightforward solution. A large portion of our effort was in modeling the existing system.

The Scrubber 3 team was made up of about two dozen engineers and managers. Individual members of the team were responsible for one or more of the following tasks: (1) development of the Scrubber 3 software; (2) integration of Scrubber 3 with the Delphi system (which, among other tasks, acts as a knowledge server to multiple applications); (3) development of the knowledge base; (4) evaluation and verification of the encoded knowledge. The team received valuable advice and verification from several local-loop diagnosis domain experts.

The Scrubber 2 system to be ported to Scrubber 3 was the most recent of many generations of local-loop diagnostic systems, the first of which was deployed a decade ago (Rabinowitz et al 1991). Because the diagnostic knowledge in the system had been adapted from earlier systems, and because many knowledge engineers had contributed to it over time (as things changed), no single person could claim a complete understanding of the knowledge base. In order to port the system, the knowledge would have to be understood and then re-coded, which was estimated to take at least one person-year. We will concentrate on the data mining applied here—used to build

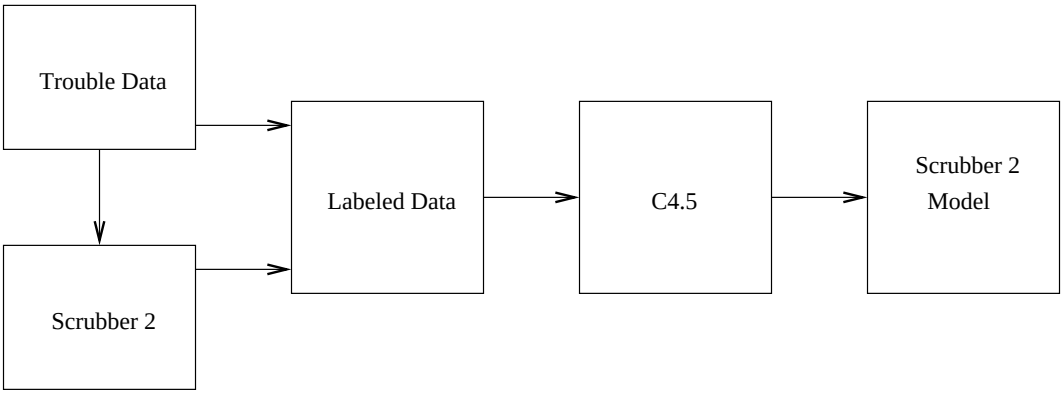

Figure H1.9.1.1: One phase of the KDD process as applied to Scrubber 2.

a model of the knowledge base that had evolved into opacity.

The KDD process applied to this project can be summarized as in Figure H1.9.1.1. What was desired was a model of the classifier as encoded in the Scrubber 2 knowledge base. To build this model, a decision tree classifier learner [link to Sections B2.4, C5.1], C4.5 (Quinlan 1993), was applied to examples of Scrubber 2's diagnostic behavior. (We had performed similar data-mining analyses of the MAX system (Danyluk and Provost 1993; Provost and Danyluk 1999).) The training set consisted of trouble cases analyzed by Scrubber 2, along with Scrubber 2's diagnoses. The learned classifier not only summarized the knowledge used by Scrubber 2 for diagnosis, but (important for our purposes) it identified the attributes used most heavily by the system for diagnosis.

The model indicated that, of a large set of possible attributes, only a small subset were necessary to model its performace accurately. The ability to focus on this smaller set of attributes opened up the possibility of obtaining significantly more data from a database that contained relevant data in abridged form. These additional data were used to refine the model built by the decision tree, in particular to estimate the probabilities at the leaves of the tree.

H1.9.1.2 KDD process

Business problem

Scrubber diagnoses problems in the local loop of the telephone network. The local-loop is that part of the network from the customer's site to the central office. Problems in this part of the telephone network may be due to trouble with customer premise equipment (such as telephones, answering machines, or wiring, for example); trouble with the facilities connecting the customer premise equipment to the cable to the central office; trouble with the cable; or trouble in the central office itself (e.g., in

connecting a cable to a switch). Millions of "troubles" (customer-reported problems) are reported each year. Fixing a trouble can cost hundreds of dollars.

The first expert system to diagnose problems in the local loop, MAX, was deployed in 1990 (Rabinowitz et al 1991). MAX (Maintenance Administrator eXpert) performed the task of a Maintenace Administrator (MA), an employee who studies the relevant data and decides how to resolve the problem. When a customer detects a problem with their line and calls the company to report it, a phone company representative handles the call, recording information reported by the customer. While recording this information, the representative also initiates electrical tests on the line. When MAX was in heavy use, all this information, along with other information, such as the type of switching equipment to which the customer was connected, then was sent either to a human MA or to MAX for diagnosis. The diagnostic task was to determine roughly where the trouble was so that the appropriate type of technician could be dispatched. Over the years the knowledge base of MAX was updated, extended, and migrated to several other systems, including Scrubber and Delphi.

The Scrubber 2 system performed a task at a later point in the diagnostic pipeline. Many troubles reported are not appropriate for immediate technician dispatch. For instance, a customer might have reported a trouble with their phone, only to discover that the trouble was that they had left another phone in their home off the hook. If the customer replaces the phone receiver properly without contacting the phone company, a technician will be dispatched unnecessarily. As another example, heavy rains might cause lines in a cable to become wet, causing problems that then appear to go away as soon as the weather dries up. Finding and fixing such a trouble is a very difficult problem, and a routine dispatch will be fruitless. The job of Scrubber 2 was to survey the queues of troubles waiting to be dispatched, to determine whether the dispatch indeed appeared to be appropriate.

In its first year of operation, the MAX system was estimated to have saved NYNEX approximately six million dollars. Due to the sheer volume of local-loop troubles handled at that time, even a single percentage point increase in accuracy was estimated to be worth three million dollars a year in savings to the company. Given the merger of NYNEX and Bell Atlantic, the volume of local-loop troubles is now significantly higher.

Motivation for a data mining solution

The primary goal of the Scrubber 3 team was to create a new Scrubber knowledge base, appropriate for the new applications platform, that performed as well as Scrubber 2. Because of platform incompatibilities, Scrubber 2's code, developed in a no-longer-supported expert system shell, could not simply be transferred to Scrubber 3. Rather, it would have to be re-coded. Unfortunately, as discussed above, Scrubber 2's knowledge base had evolved over a decade, and various knowledge engineers had worked on it during that time, many of whom had moved on and were not avail-

able for building Scrubber 3. More important, over these years of continual tweaking and updating, a complete understanding of the knowledge base had been lost.

In order to port the system, it would first have to be understood and then re-coded, which was estimated to take at least one person-year. An alternative to understanding the Scrubber 2 knowledge might have been re-engineering the knowledge base "from scratch." This option was dismissed because (1) time estimates for this task were greater than the one person-year required for the port, and (2) the collective knowledge gained over the years of modifying the local-loop knowledge base would have been lost. The evolution of local-loop knowledge had produced a system well regarded from several important perspectives (users, experts, analysts, researchers).

Rather than poring over the knowledge base manually, we used data mining techniques to gain an understanding of the classification rules encoded in Scrubber 2. Training and test sets were built from actual cases diagnosed by the system. Labels on individual examples were based on the diagnoses of Scrubber 2. The task was to use a classifier-learner (in our case, C4.5) to build a classification model that would model Scrubber 2's performance with high fidelity. Our hope was that the model would give us important insights into the knowledge encoded in Scrubber 2, and would automate to a large degree the construction of the knowledge base for Scrubber 3.

We had evidence that such a model could be built, having used the same modeling process for MAX, the first in the series of local-loop diagnostic systems (Danyluk and Provost 1993; Provost and Danyluk 1999). Although MAX was successful and had gained general acceptance, it was also clear that it could be improved. We sought to improve the system in a number of ways including (1) improving the core knowledge base; and (2) building specialized knowledge for different geographic locations. We constructed a database of troubles that had been handled by MAX across the company. For many of these cases we had two sources of labeling: the diagnosis given by MAX and the resolution ultimately reported by the technician who had been dispatched to resolve the trouble. We sought to use the technicians' diagnoses to tune the knowledge already encoded in the system. We applied a number of classifier learning algorithms to our data, including decision tree learners, rule learners, and neural networks. We trained and tested various methods on data labeled with MAX's diagnoses. In all cases we found that we were able to model MAX with a fidelity of at least 0.98 with as few as 10,000 training examples. This led us to believe that we also could model Scrubber with high fidelity, since it was a descendant of MAX, and each operated on a different subset of the same distribution of data.

The available data

The initial data available to us consisted of approximately 26,000 troubles that had been handled by Scrubber 2. These data were extracted from Scrubber's log files. Each trouble was described by over 40 attributes. Of those, approximately half were

continuous-valued attributes and half were symbolic-valued. For each trouble, we had the diagnosis that had been assigned by Scrubber 2. We framed the learning problem [link to Section E4] as a two-class problem, where the trouble either should be dispatched or not. In the former case, the trouble is left in the dispatch queue. In the latter case, Scrubber removes the trouble from the queue and sends it to a voice-response unit, which calls the customer with an automated message and asks the customer to verify whether the problem indeed is no longer apparent.

No background knowledge was encoded explicitly for use by the classifier learner. However, our own background knowledge, developed over years of working with the MAX system and its derivatives, was used in selecting the appropriate learning program (C4.5, in this case) and the appropriate attributes that were given as input to the learner. Of over 40 possible attributes, we selected 17.

Data mining resources

Decision-tree learning was applied to the data in order to construct a high-fidelity model that would also be more understandable than the complex rule base into which the original diagnostic knowledge had evolved. The model produced needed to be understood by the data miners, by the domain experts, and in many cases by the system users. The decision tree learner C4.5 was ultimately used to model the behavior of Scrubber 2. While other algorithms also were applied, among them Ripper (Cohen 1995), we have found over the years that for the local-loop diagnosis problem, C4.5 consistently produces classifiers with high predictive accuracy that are also comprehensible by the domain experts.

As we mentioned above, we had other desiderata for the new knowledge base. For instance, different classification errors have different associated costs. The existing Scrubber 2 model was optimized for a certain set of cost assumptions, but there was continual disagreement as to whether these were best. Furthermore, it became clear that debates over the "best" set of cost assumptions was misplaced. Costs were different depending on location (compare rural Maine to Brooklyn), and costs change based on changes in trouble load and changes in the workforce. Thus there was a tension between the primary need to build a comprehensible model, and the need for flexibility in the face of the diverse and changing environment. Since we had found decision trees to be highly preferable for explanatory purposes, and very accurate for a fixed set of cost assumptions, our solution was to convert the standard classification trees to probability estimation trees. This conversion is described below.

H1.9.1.3 Results

What knowledge has been discovered?

As discussed above, data mining techniques were applied to the Scrubber 2 data in order to model the Scrubber 2 knowledge base, which had evolved into opacity. There were two important criteria for the model constructed: readability and consistency with the behavior of Scrubber 2.

Applying C4.5 to the Scrubber 2 data, we were able to produce high-fidelity replicas of Scrubber 2's knowledge base. With a training set of as few as 10,000 examples, C4.5 produced decision trees with predictive accuracy of over 0.99 (predicting Scrubber 2's diagnosis from its inputs). While the trees produced provided us with a readable model of Scrubber 2's knowledge, we sought to simplify the model where possible in order to gain a better understanding of the system's behavior.

The decision trees produced by C4.5 consistently indicated that a small subset of all the attributes figured prominently in Scrubber 2's diagnoses. Focusing on this subset alone, we used C4.5 to build a new model. While we traded off some fidelity for simplicity of the model produced, the resulting model still fell in a range of acceptable behavior according to guidelines set out at the beginning of the project. These guidelines required not only that overall fidelity of the model be high, but that the number of false positives and false negatives be in appropriate ranges as well.

This model satisfied our criteria for a high-fidelity, yet simple and comprehensible model. This was critical both for our own understanding of Scrubber 2's behavior and for the approval of the domain experts who needed to accept the model for transfer into Scrubber 3. The restriction of the attribute set also had important implications for our construction of the final probability estimation tree.

Scrubber 3

A base requirement for Scrubber 3 was that it be at least as good as the original system. As discussed above, we had additional desiderata as well, including improving the performance of the system over that of Scrubber 2, and designing it for extensibility. We wanted a comprehensible, locally modifiable, extendable, probabilistic classifier, that could accept (naturally) both discrete and continuous attributes. Because of its readability, and because it fit well with the rule-based paradigm used for prior systems, a probabilistic decision tree was the obvious choice.

We modified C4.5 to label examples with class probability estimates (Laplace-corrected class frequencies), rather than with binary decisions. The deployed system reads a probability estimation tree from a file and instantiates it as a lookup table. For a particular problem, it also reads in appropriate decision thresholds, which can be adjusted as needed independently of the learned model.

Even after attribute selection, the resultant tree had over 2500 leaves, some of which were very small. The problem of small disjuncts has received considerable

attention in the machine learning literature as one of the fundamental challenges of data mining and of this domain in particular (Holte et al 1989; Danyluk and Provost 1993). Small disjuncts cause even more problems in building probabilistic decision trees. For example, consider the case where a leaf comprises 5 positive examples and 1 negative. What probability would it be appropriate to assign to that leaf? How confident might we be that that probability was accurate? Notice two things. First, the heuristics used to build decision trees do a great deal of search to find relatively pure leaves (with mostly one class) on the training data. All this search may end up producing example distributions at the leaves that are overly pure simply as an effect of searching too hard. Second, even if the example distributions are more-or-less correct, the corresponding probability estimates may not be precise simply due to the size of the leaves. For simple classification this will not be too much of a problem if the estimate is significantly far from the threshold.

Fortunately, our reduction of the attribute space helped us with both of these problems. We obtained access to a huge database of abridged trouble histories, which did not comprise all the attributes, but did comprise the subset used in the final tree. This database had not only the values for these attributes, but also the resolution reported by the technician for the trouble. After cleaning these data (for example, a resolution can not be considered to be correct if the customer called back shortly thereafter and reported that the trouble in fact had not been resolved), we used them to instantiate the probabilities at the leaves of the tree. The database comprised several million cases; interestingly, the tree still had some small disjuncts.

Applications of the discovered knowledge

Scrubber 3 currently is deployed and is making automated decisions about millions of troubles and saving the company over ten million dollars annually.

The probabilistic model was judged to be general enough to apply to a wider variety of local-loop diagnosis problems. For example, consider the following "front-end" diagnosis task. When a customer calls in with a trouble, the answering CSR (customer service representative) may be able to gather all the relevant data and make an assessment immediately. For example, if a customer calls in from a different phone line, electrical tests on the subject line can be made directly. The Delphi system also acts as a knowledge server to make recommendations to assist with the interaction. Much more knowledge can be gathered by an informed interaction between the CSR and the customer. For example, if the model were to indicate with very high probability that the problem is in the customer's own equipment, the CSR may be able to guide the customer much more aggressively to search for malfunctioning modems or chair legs puncturing extension cords.

By training the probabilities of the tree appropriately for different distributions of troubles, we found that the same tree was effective for different problems. Delphi now uses the probability estimation tree built for Scrubber 3 to serve knowledge also for

assisting the CSRs with the front-end task. Quantifying the benefit of such assistance is more difficult than with a fully automated system like Scrubber. However, the incremental development cost was quite small.

References

Cohen, W. W. 1995. "Fast effective rule induction." In *Proceedings of the Twelfth International Conference on Machine Learning (Tahoe City, Calif.)*, edited by A. Prieditis and S. Russell, pp. 115-123. San Francisco, Calif.: Morgan Kaufmann.

Danyluk, A. P. and Provost F. J. 1993. "Small disjuncts in action: learning to diagnose errors in the local loop of the telephone network." In *Proceedings of the Tenth International Conference on Machine Learning (Amherst, Mass.)*, pp. 81-88. San Mateo, Calif.:Morgan Kaufmann.

Holte, R. C., Acker, L. E. and Porter, B. W. 1989. "Concept learning and the problem of small disjuncts." In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (Detroit, Mich.)*, edited by N. S. Sridharan, pp. 813-818. San Mateo, Calif.:Morgan Kaufmann.

Provost F. J. and Danyluk, A. P. 1999. "Problem definition, data cleaning and evaluation: a classifier learning case study." *Informatica*, **23**:123-136.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, Calif.:Morgan Kaufmann.

Rabinowitz, H., Flamholz J., Wolin E. and Euchner, J. 1991. "Nynex max: A telephone trouble screening expert." In *Innovative Applications of AI 3 (Anaheim, Calif.)*, pp. 213-230, Menlo Park, Calif.: AAAI Press.