An Intelligent Problem Selection Agent for SQL-Tutor Using Artificial Neural Networks

November 8, 2002

Timothy Wang
Supervisor: Tanja Mitrović

Abstract

The use of Intelligent Tutoring Systems (ITS) is necessary to alleviate the teaching shortage that has effected educators in recent years. Structured Query Language Tutor (SQL-Tutor) is an ITS developed at the University of Canterbury and is used for teaching undergraduate database courses. Artificial Neural Networks (ANNs) is a Machine Learning approach that is a simple approximation of the human brain. ANNs are very good at learning in domains where there are no well defined algorithms or the domain is not well understood. The research presented investigates the possibilities of using ANNs for making pedagogical decisions.

The use of ANNs for this project focused on the selection of appropriate problems for students to work with. Two ANNs were used to select a suitable problem. The first network was designed to assess whether a student is struggling with a problem. If it has been determined that the student will have difficulty, the problem selector finds an appropriate problem. Prediction of whether the student will have difficulties with a problem achieved 93% accuracy. The second network selects the problem that is best suited to the student's level of ability. Prediction accuracy achieved with this network is on average 79%.

The first network performed well in assessing whether a student will have difficulty with a problem. The second network was less successful when finding an appropriate problem for a student to attempt. The results do suggest that there is a good basis to use ANNs with SQL-Tutor and other ITSs.

Contents

Chapter 1

Introduction

An Intelligent Tutoring System (ITS) is a tool to help address the growing shortage of educators. ITSs are not meant to replace human experts, such as teachers and lecturers, but are used to offer assistance on a one-to-one basis to teach domain specific lessons in areas such as mathematics, physics, English and programming languages. An ITS observes the behaviour of a student and presents a suitably adapted lesson with appropriate problems. On the successful completion of a given set of tasks, the student may be presented with new concepts or more difficult examples.

The ability to provide assistance for a student at the appropriate level is invaluable in the learning process. Not only does it aid the student's learning process but also prevents problems, such as student frustration and floundering [2]. Selecting a problem at the correct level of difficulty for a student presents many challenges for a tutor [23].

The motivation to develop a problem selection module, using artificial neural networks, to work with the student model of Structured Query Language Tutor (SQL-T) [10], stems from the lack of research into student satisfaction, reduction of stress and frustration when presented with problems. This is applicable especially when being introduced to a new domain. Problem difficulty has been used to determine the knowledge a student has acquired during their learning with an ITS, but there are situations when being presented with increasingly challenging problems does not aid the learning process. The student may need to be presented with easier or similar difficulty problems to boost their confidence and make their experience more enjoyable.

Chapter 2 covers background details for similar endeavours in the area of problem selection and a brief description of SQL-Tutor and neural networks. In Chapter 3, design considerations for the final system are described and broken into two components: error predictor and problem selector. Evaluation of the system is discussed in Chapter 4, and within the same chapter, we present the results and a discussion on our findings. Finally Chapter 5 presents our conclusions and further work that could be undertaken.

Chapter 2

Background

Problem selection in ITSs has been implemented using many different approaches, including neural networks. The following sections review SQL-Tutor, present a brief explanation and history of neural networks and some of the different approaches used for problem selection, as well as comments on their benefits and deficiencies.

2.1 SQL-Tutor

SQL-Tutor is an ITS that helps university-level students to learn Structured Query Language (SQL). The architecture of the stand-alone version of the system is illustrated in Figure 2.1.

SQL-Tutor consists of an interface; a pedagogical module, which determines the timing and content of pedagogical actions; and a student modeller Constraint-Based Modelling (CBM), which analyses student answers. The system contains definitions of several databases, a set of problems and the ideal solutions to them. Each problem is assigned a complexity level, ranging from one (easiest) to nine (hardest). SQL-Tutor does not contains a domain module. In order to check the correctness of the student's solution, SQL-Tutor compares it to the correct solution, using domain knowledge represented in the form of more than 600 constraints. It uses Constraint-Based Modeling [19] to model the knowledge of a student.

When the student submits a solution, the student modeller analyses it and identifies any violated constraints. If the solution is correct, the student is given a congratulatory message. If the student is incorrect, the pedagogical module provides a feedback message at a specific level that determines how much information is provided to the student. There are six levels of feedback in SQL-Tutor: positive/negative feedback, error flag, hint, detailed hint, partial solution and complete solution.

At the lowest level (positive/negative feedback), the message simply informs the student whether the solution is correct or not and, in the latter case, how many errors there are. An error message informs the student about the clause (part of the solution) in which the error occurred. A hint-type message gives additional information about the type of error, in the form of a general description of the error. This description is

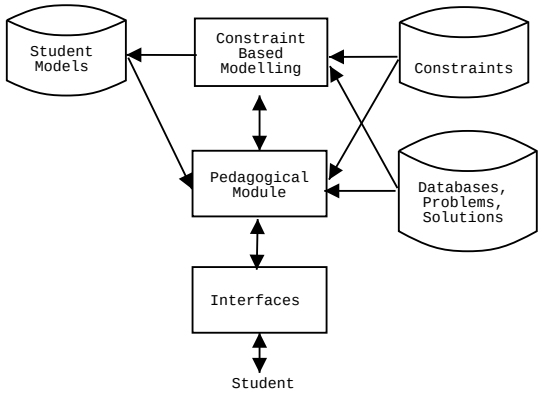

Figure 2.1: Architecture of SQL-Tutor

taken directly from the definition of the violated constraint. A detailed hint provides additional information about the error. Partial solution feedback displays the correct content of the incorrect clause, while the complete solution simply displays the correct solution of the current problem. The student is also assigned a level, ranging from one (novice) to nine (proficient) which increases as a student successfully completes problems indicating proficiency and mastery of SQL.

There are three ways a student may be presented with problems in SQL-Tutor. They may work their way through a series of problems for each database, select a practice problem directly from a menu of options, or turn problem selection over to the system. In the third case, SQL-Tutor examines the student model and selects the problem from a pool of unsolved problems whose level is ± 1 of the student's level, which is relevant for the constraint the student has violated most frequently. The rationale for this rule is that if the student has violated the same constraint several times, it is appropriate to target that constraint for instruction [7].

SQL-Tutor has been evaluated in five studies since 1998, which all provided evidence for the sound psychological foundation of CBM, and showed that students significantly improve their knowledge by using the system [12].

2.2 Neurons and Perceptrons

Before introducing artificial neural networks (ANNs) we must first present the building blocks: artificial neurons [8] and perceptrons [22].

2.2.1 Neuron

An artificial neuron is a simplified model of the human brain's neuron (see Figure 2.2). The earliest neurons were developed by [8] in 1943. Inputs into the neuron (x_i) are the external stimuli from the environment. These values can be from a discrete set, such

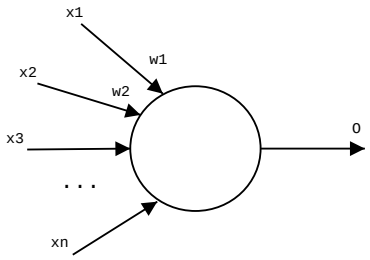

Figure 2.2: Artificial Neuron

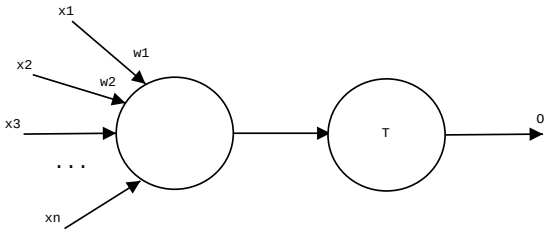

Figure 2.3: Perceptron

as $\{0,1\}$, 0 - inhibitory or 0 - excitatory. Weights (w_i) are real valued numbers that determine the contribution of each input to the neuron's weighted sum and eventually its output. Activation function is a simple step function:

$$\begin{aligned} X &= \sum_i w_i x_i \\ O &= f(X) \\ O &= 1 \text{ if } X \geq 0 \\ O &= 0 \text{ if } X < 0 \end{aligned}$$

These neurons were able to learn logical functions, such as $a \wedge b$, $a \vee b$, and $\neg a$ but could not do anything complex. This discovery was the basis for the development of perceptrons.

2.2.2 Perceptron

Rosenblatt [22] took the work on neurons developed by [8] and extended the concept into perceptrons. The inputs ($x_1,\ldots,x_n$) and weights ($w_1,\ldots,w_n$) are all real valued numbers, they could also accept symbolic values. Perceptrons were able to learn - the process of modifying the values of weights and the threshold (T). The activation function is as follows:

$$\begin{array}{ll} 1 & \text{if } \sum_i w_i x_i > T \\ 0 & \text{otherwise} \end{array}$$

The power of perceptrons came from forming networks of interconnected perceptrons. These networks were able to learn classes that were linearly separable. Unfortunately, not all classes are linearly separable.

2.3 Artificial Neural Networks

An artificial neural network (ANN) is a powerful machine learning method that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Artificial neural networks resemble the human brain in the following two ways:

1. An ANN acquires knowledge through learning.
2. An ANN's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

The true power and advantage of ANNs lie in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the historical data. Traditional linear models are simply inadequate when it comes to modelling data that contains non-linear characteristics.

The most common neural network model is the multilayer perceptron (MLP). This type of neural network is known as a supervised network because it requires labeled examples in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data. Thus, the model can then be used to estimate the output when the desired output is unknown.

The MLP and many other neural networks, such as generalised feedforward (GFF) and probabilistic neural network (PNN), learn using an algorithm known as backpropagation. With backpropagation, the input data is repeatedly presented to the neural network. With each presentation the output of the neural network is compared to the desired output and an error is computed. This error is then fed back (backpropagated) to the neural network and used to adjust the synaptic weights such that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output. This process is known as "training" [18].

Neural networks can be used for many different applications:

- Process modelling and control - Creating a neural network model for a physical plant then using that model to determine the best control settings for the plant.
- Machine diagnostics - Detect when a machine has failed so that the system can automatically shut down the machine when this occurs.
- Portfolio management - Allocate the assets in a portfolio in a way that maximizes return and minimizes risk. For example NetProfit [20].
- Target recognition - Military application which uses video and/or infrared image data to determine if an enemy target is present.

- Medical diagnosis - Assisting doctors with their diagnosis by analyzing the reported symptoms and/or image data such as magnetic resonance imaging (MRI) or X-rays.
- Credit rating - Automatically assigning a company's or individuals credit rating based on their financial condition.
- Targeting marketing - Finding the set of demographics which have the highest response rate for a particular marketing campaign.
- Voice recognition - Transcribing spoken words into ASCII text.
- Financial forecasting - Using the historical data of a security to predict the future movement of that security. For example, Walkrich Investment Advisors [25].
- Quality control - Attaching a camera or sensor to the end of a production process to automatically inspect for defects.
- Intelligent searching - An internet search engine that provides the most relevant content and banner ads based on the users' past behaviour.
- Fraud detection - Detect fraudulent credit card transactions and automatically decline the charge.

2.4 Remedial Mathematics Tutor

An ITS developed to teach remedial mathematics [23] to community college students used a student model based on historical population data. This system tracked students as they solved problems to determine their proficiency on all topics for both procedural and conceptual knowledge. The proficiency and acquisition factor, which determines how quickly a student learns, is calculated by the tutor to help determine topic and problem selection. Problem selection examines a student's proficiency in the selected topic and its subskills (skills required to complete a topic). The greater the proficiency on the topic, the more subskills the student is presented with. Conversely, a student with a lower proficiency will be presented with a problem with fewer subskills to solve. The authors suggest that to aid future predictions (since the student model is not perfect), it is worthwhile to present slightly easier or harder problems than the model determined the student can handle.

2.5 Mixed Numbers, Fractions and Decimals

Mixed numbers, Fractions and Decimals (MFD) [3] is an ITS developed to teach ten and eleven year old children arithmetic concepts. The system adapts its instruction by intelligently selecting topics on which the children should work, providing hints that match the student's level of ability, and dynamically constructing problems that are appropriate for the student. They used a two-component architecture - a population student model (PSM) and per student component - to predict a student's performance.

First, data from the entire student population is used to train a neural network. Second, the system learns how to modify the neural network's output to better fit each individual student's performance.

Each concept to be learned by the students had been represented by a topic neural network. A topic refers to a large unit of knowledge, such as "subtract fractions" or "multiply whole numbers". Each topic has a list of pretopics; before a student can work on a topic, all of its pretopics must be passed. In addition, the ITS knows a topic's subskills, which are all of the substeps required to solve a problem of a given topic.

The PSM network outperformed simple guessing and regression based techniques, and the individual student networks out performed the PSM. Therefore, both components are useful. The PSM is powerful because data can be collected from every user and broad generalisations about student performance can be drawn. The individual student networks are powerful because they can capture information about the student not stored in the student model.

2.6 SQL-Tutor with Bayesian Networks

A probabilistic adaptive decision-making approach for SQL-Tutor using Bayesian probability theory [7] aided the selection of appropriate problems to present to a student. For each problem, simple Bayesian networks are used to make multiple predictions about student performance based on atomic domain elements called constraints. These multiple predictions are then combined heuristically to give an overall measure of the value of the problem. Based on the probabilities the "best" problem is presented to the student.

This approach suffered from inefficiencies in evaluating large Bayesian networks online. Due to the initial probabilities being difficult to determine. Another inefficiency was the sheer number of Bayesian networks necessary to determine an appropriate problem. For each of the constraints that represents domain knowledge, a Bayesian network is required. For a system like SQL-Tutor, there are over 600 constraints. High overhead in memory and computational demands makes this particular implementation too cumbersome for use in fully interactive systems.

2.7 Intelligent Error Prediction

Neural networks were used to determine the number of errors a student would make for a given problem in SQL-Tutor in a study by [26]. This research was a pilot study to determine the feasibility of the use of neural networks with a constraint based ITS.

There were a few fundamental flaws in the evaluation and assessment of the network's accuracy. The trained neural network was tested with the training data, and thus provided a high accuracy level. However, the testing data should not have been the same as the training data. When testing for each simulated student, the same network was used. It is necessary to have the same starting point for each student when testing. As the network was used for each of the students its internal representation changes

and is not a true reflection of the individual student. The synaptic weights would have been changed to adapt to a student the next student tested in the network would have the starting point of the previous student, not the original network that had been generalised from the population model. All of the students needed to be tested from the original trained network.

The study provided the basis for the use of neural networks with SQL-Tutor and is the motivation for further research in the area of problem selection.

2.8 Summary

The domain of problem selection is still in its infancy, many different teaching strategies have been presented and researched. It is the goal of this study to develop a problem selection agent that can determine whether a problem is too difficult for the student and select an appropriate problem to present. We believe that this will aid the student's learning and enjoyment in their interaction with an ITS, in this case SQL-Tutor.

Chapter 3

Design

There are two stages in problem selection. First, we must determine if a student will have difficulty with the current problem. Second, if the student is expected to have difficulty with the current problem we must intervene and select a suitable problem for the student to attempt. If the error predictor determines that a student will not have difficulty with the current problem the student can continue. In this project, both stages are supported by neural networks.

A description of the toolkit used for the project is discussed in Section 3.1. The historical data used for the training of the network is described in Section 3.2. The design and specifications for the error predictor network are discussed in Section 3.3. The design details for the problem selector are discussed in Section 3.4.

3.1 NeuroSolutions

NeuroSolutions version 4.2 developed by NeuroDimension [18] is a toolkit for developing and experimenting with ANNs. It has an icon-based graphical user interface providing the most powerful and flexible development environment available on the market today. The package provides the necessary tools to design, build, train and test all the networks developed for this research. NeuroSolutions runs on the Windows operating systems environment. We were fortunate enough to have an Educators Licence [17] for the package to allow us to access several network architectures; multilayer perceptron (MLP) and generalised feedforward (GFF), and also the ability to use backpropagation. The educator licence restricts the networks to 50 inputs per layer and two hidden layers. However, these restrictions were not a limiting factor for the development of the networks in our research.

3.2 Historical Data

Historical data of the problem domain is required by the network to learn the task. From the historical data we must determine what is necessary, as historical data can

```
...

14:52:17 11/10/2001
Pre-process:

Help Level 5
Feedback Option: Complete Solution
Database: movies
Problem number: 55
Their attempt:
Select: code
From: tape
Where:
Group by: times
Having: having count(times > 10)
Order by:
Two-level-help?: NIL
Mode: 1
-----------
14:52:18 11/10/2001
Post-process:

Satisfied constraints: (458 200 650 65 ... 366 365 364 3 2)
Violated constraints: (154 805 271 471 160 241 514)
-----------
14:52:51 11/10/2001
User logged out
```

Figure 3.1: An example of a problem submission within a student log

be voluminous and difficult to analyse, for the training data set. This can be difficult to determine. A good approach is to include everything that could be useful from the historical data, build and test the network with all the possible data and remove what is not necessary. Unfortunately, there is no set way to do this. Trial and error is a good place to start, eliminating the unnecessary inputs. The data available to us for the investigation into problem selection consisted of log files of previous evaluations of SQL-Tutor in 2001. We had log files for 51 students that have worked with SQL-Tutor. Six of the log files were invalid as there were no problems solved by the student and no data could be extracted from the log file. There were several cases were the student logged themselves onto SQL-Tutor and logged out immediately, thus not completing any problems.

Several deficiencies were addressed with previous work on using neural networks for error prediction [26] in Section 2.7. The time to complete a problem was used to aid the error prediction and was a valuable input. We would not have access to the 'time' variable in an online scenario, as it would be impossible to determine how long it would take the student to complete a problem.

Other information from the student model would be necessary to train the networks.

Training files were generated from these student logs (Figure 3.1). The following information was extracted:

- Problem number, the identification of the problem, ranging from one to *n*.
- Problem level, which shows the problem difficulty. Ranging from one (easiest) to nine (most difficult).
- Student level, which is the current level that the student has achieve. The student level ranges from one (novice) to nine (experienced).
- Problem attempt, whether the student has attempted the problem before. The attempt is equal to zero if the problem is new, and one if they have seen the problem before.
- Coverage SELECT, ratio of coverage of the SELECT clause. This is the percentage of constraints relevant to the SELECT clause that the student has used.
- Coverage FROM, percentage of coverage of the FROM clause.
- Coverage WHERE, percentage of coverage of the WHERE clause.
- Coverage GROUP, percentage of coverage of the GROUP clause.
- Coverage HAVING, percentage of coverage of the HAVING clause.
- Coverage ORDER, percentage of coverage of the ORDER clause.
- Measure SELECT, percentage of measure of the SELECT clause. The percentage of correctly used constraints.
- Measure FROM, percentage of measure of the FROM clause.
- Measure WHERE, percentage of measure of the WHERE clause.
- Measure GROUP, percentage of measure of the GROUP clause.
- Measure HAVING, percentage of measure of the HAVING clause.
- Measure ORDER, percentage of measure of the ORDER clause.
- Number of errors the student made in the last attempt on the current problem.

3.2.1 Training Data

There were 51 log files from a previous evaluation of SQL-Tutor. Six of those student logs were invalid because there were no problem submissions. The students logged onto the system and did not complete any tasks, logged off and did not return. These files were omitted. For training purposes we trained the networks with 21 students and tested the networks with the remaining 24. This split was arbitrarily based on the size of the training and testing set. The training set had 2206 submissions, and the testing set had 2043.

3.2.2 Cross Validation

Cross validation is one of the most successful methods of overcoming the overfitting problem. This is done by simply supplying as set of validation data to the algorithm in addition to the training data [9]. This method monitors the error on an independent set of data and stops training when this error begins to increase. This is considered to be the point of best generalization. It is generally accepted that there should be a 80/20 split of the training data to cross validation ratio if the training set is bigger the more of the training set can be used for cross validation. The most appropriate split for cross validation is dependent on the modelling scenario and on other assumptions. Small split ratios seem to be a safe option, even when the optimal ratio is fairly large [4]. For all the training on the networks developed we used an 80/20 split of the training set. The cross validation set has been taken from the training set file and is handled internally by NeuroSolutions.

3.3 Error Predictor

3.3.1 Representing Errors

We needed to predict whether the student will have difficulties with the problem they have been presented. There were several approaches in determining whether a student is having difficulty:

1. Predict the actual number of errors a student will make. For example, if a student will violate eight constraints, the network must be able to predict that the student will violate eight constraints.

2. Assume that violating one or more constraint is considered having difficulty with the problem. The error must be encoded accordingly. For example, if the student has violated one constraint then the error encoding will be one. If the student does not violate any constraints then their error encoding will be zero.

3. Determine an acceptable threshold of violated constraints, and assume that the student is capable of completing the problem successfully without difficulty.

The third approach has been used as the basis to ascertain whether a student is having difficulty. The decision to use this approach has been made after preliminary experimentation with several network topologies and architectures. For example, both Multilayered Perceptrons (MLP) and Generalised Feedforward (GFF) networks were used to predict the actual number of errors a student will make. These networks performed poorly; the MLP achieved between 50–70% accuracy, not much better than a random coin toss. The GFF performed worse; with 40–65% accuracy. A MLP and a GFF network were built to assess the accuracy with the second approach in the above list. The MLP network performed 15% better than the network built for Approach 1. The GFF network performed 10% better than its first approach counterpart. The final approach to predict whether a student will make errors below a threshold performed the best (Approach 3). The performance of this network is discussed in detail in Chapter 4

(under the results section). We have assumed that a student who will violate five or less constraints is proficient enough to complete the problem without intervention from the problem selector. These students will be allowed to complete the current problem.

3.3.2 Network Architecture

The first network is a two layered multilayer perceptron (MLP) neural network using the linear *tanh* activation function (see Section 3.3.3) and the Delta-Bar-Delta (DBD) (see Section 3.3.4) backpropagation algorithm to update the network weights. There are seventeen inputs into the network. Sixteen of the inputs are problem number and level, student level, attempt, coverage and measure ratios. The seventeenth is the input for the desired output. The network uses this for training and cross validation. Figure 3.2 shows the inputs into the error predictor network. The last input is the desired output, this is necessary for training and cross validation. The desired output is not required when testing or using the network.

The first hidden layer has twelve processing elements or axons. The training parameters for DBD had a step size of 1.000, additive 0.100, multiplicative 0.100 and smoothing of 0.500. The second hidden layer has six processing elements, while the parameters for DBD were the same as for hidden layer one, except for the additive parameter begin set to 0.010. The output layer also uses the DBD step size update function. The parameters are also the same as for the hidden layer one, except the additive parameter is 0.001. The training for the networks stops either when the mean squared error (MSE), reaches 0.01, or 1000 epochs have been reached.

The single output for the network predicts that a student will have difficulty if the value is 1 ± 0.5, thus requiring the intervention of the problem selector network. If the output value is 0 ± 0.5, then it is assumed that the student will make five or less errors on a given problem, will not require the assistance of the problem selector and can complete the current problem. The output value will be an input into the second network as a measure of error for the network to find a suitable problem.

3.3.3 Linear Tanh

The activation function for the axon or processing elements substitutes the immediate portion of the tanh [16], Equation 3.1, by a line of slope β making it a piecewise linear approximation of the tanh. This process element has an input-output map that is discontinuous, so it is not recommended for learning. However, when used with backpropagation it has the ability to learn. It is easier to compute the map and is computationally more efficient than using the tanh function [14].

$$f(x_i, w_i) = \tanh[x_i^{lin}] \tag{3.1}$$

$$f(x_i, w_i) = \begin{cases} -1 & x_i^{lin} < -1 \\ 1 & x_i^{lin} > 1 \\ x_i^{lin} & else \end{cases} \tag{3.2}$$

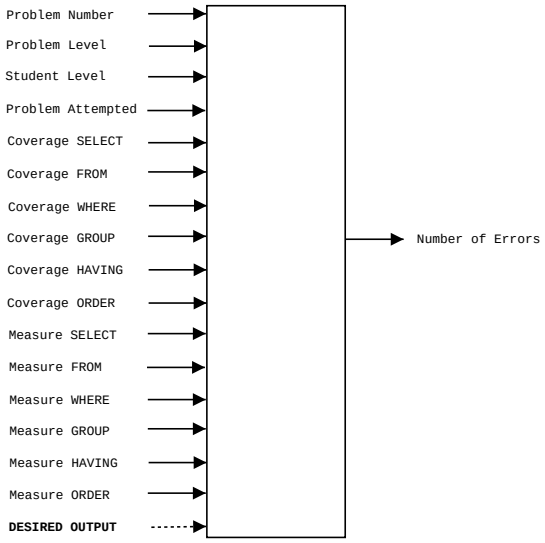

Figure 3.2: The Error Predictor Network Inputs

where $x_i^{lin} = \beta x_i$ is the scaled and offset activity inherited from the linear activation function [13], Equation 3.3.

$$f(x_i, w_i) = \beta x_i + w_i \tag{3.3}$$

3.3.4 Delta-Bar-Delta

Delta-Bar-Delta is an adaptive step-size procedure for searching a performance surface. The step-size and momentum are adapted to the previous values of the error at the processing elements of axons, if the current and past weight updates are both of the same sign. For example, if they are both positive values, the learning rate is increased linearly. The reasoning is that if the weight is being moved in the same direction to decrease the error, then it will get faster with a larger step size. If the sign of the update weights are different, this indicates that the weight has been moved too far. When this occurs the learning rate is decreased geometrically to avoid divergence. Equation 3.4 is the equation for the step size update.

$$\Delta\eta_i(n) = \begin{cases} \kappa & S_i(n-1)\nabla w_i(n) > 0 \\ -\beta\eta_i(n) & S_i(n-1)\nabla w_i(n) < 0 \\ 0 & otherwise \end{cases} \tag{3.4}$$

where:

$$S_i(n) = (1-\lambda)\nabla w_i(n-1) + \lambda S_i(n-1) \tag{3.5}$$

$\kappa =$ Additive constant
$\beta =$ Multiplicative constant
$\lambda =$ Smoothing factor

Equation 3.6 is the equation used to calculate weight updates; this is the momentum function used to find the steepest gradient descent vector. The *momentum* provides the gradient descent with some inertia, so that it tends to move along a direction to the global minimum. The amount of interia is dictated by the momentum parameter, ρ. The higher the momentum, the more it smooths the the gradient estimate and the less effect a single change in the gradient has on the weight change. The major benefit is the added ability to break out of local minima that a step component might get caught in. We must note that oscillations may occur if the momentum is set too high [15].

$$\Delta w_i(n+1) = \eta_i \nabla w_i + \rho \Delta w_i(n) \tag{3.6}$$

3.4 Problem Selector

The problem selector network is used only if we have determined that a student is going to have difficulty with the current problem that has been presented. The activation of the second network is determined by the output from the error prediction network. If the output is 0 ± 0.5 then we do not need to activate the network otherwise; the network determines which problem to present to the student.

The inputs for the second network are, output from the first network as a measure of the number of errors, the level of the current problem, the student level, whether or not the problem the student is attempting has been seen by the student previously (zero if the problem is new and one if they have attempted the problem before), and the twelve coverage and measures ratios. Figure 3.3 illustrates the inputs into the problem selection network. The last input — 'PROBLEM NUMBER' is required for training to compare the network output with the desired output for training and cross validation. This input is not required for testing and using the network. The network selects the next problem for the student to attempt.

Experimentation with several topologies and architectures was used to determine the best network design. Several single hidden layered MLPs were built producing accuracies of 40–55%. These were not satisfactory accuracies for problem selection. Also, several single and double hidden layered Generalised Feedforward (GFF) networks using various learning algorithms: Step, Momentum, Quick Propagation and DBD. These networks performed poorly, producing similar accuracies to the single hidden layered MLPs.

The topology and architecture for the problem selector is a MLP with two hidden layers using the DBD learning algorithm using linear tanh as the activation function. This configuration produce accuracies in the range of 78–85%. This was an adequate level of accuracy to evaluate the performance of the network. All three layers, hidden one, hidden two and output, used DBD. The parameters for the algorithm had the same value for step size of 1.000, multiplicative 0.100 and smoothing of 0.500. The

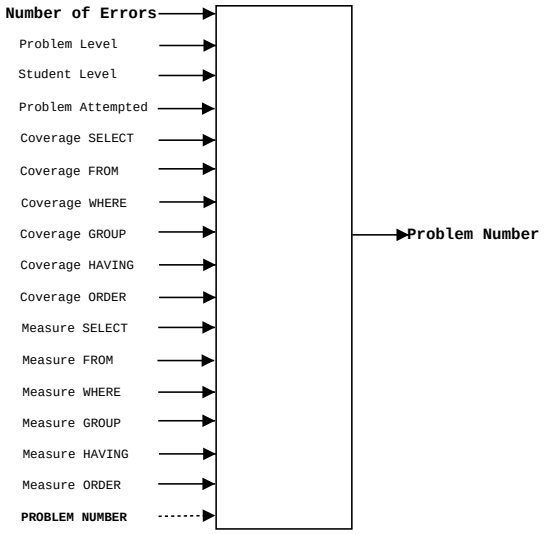

Figure 3.3: The Problem Selector Network

additive values were 0.100, 0.010 and 0.001 respectively. The training of the network terminated when the MSE of the training reached the 0.01 threshold or when 1000 training epochs had been reached.

The output, whether the student will have difficulty 0 ± 0.5, no is intervention necessary, or 1 ± 0.5 the student will make more than five errors, from the error predictor network was needed as input for the problem selector network. In order to train the network with the appropriate values, the output values were collected for each of the 21 simulated students and added to the training set for the error predictor network. Instead of expecting values of zero or one, the network was given values ranging from -0.5 to 1.5.

Figure 3.4 shows how the error predictor is used with the problem selector to find the most appropriate problem for a student. The error predictor is concerned with the current problem and the selector is concerned with the next problem for the student to attempt.

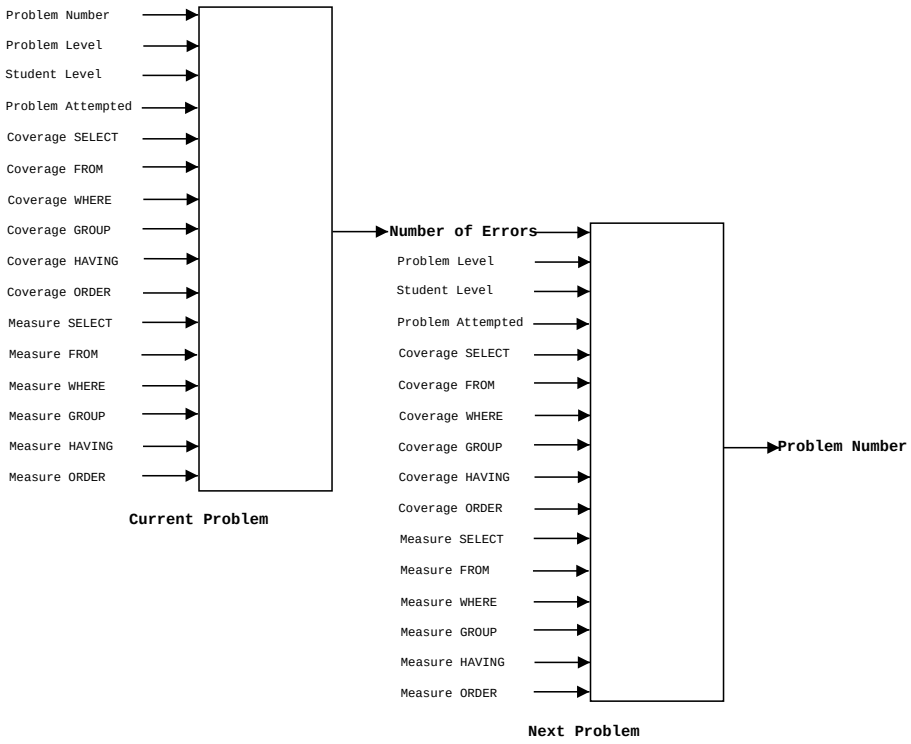

Figure 3.4: Integration of the two networks

Chapter 4

Evaluation

The two possible methods of evaluating the accuracy of the artificial neural networks are online and offline. Either option has advantages and disadvantages.

An online evaluation requires two versions of SQL-Tutor, an existing version and a version modified with the proposed problem selection components. The evaluation would be conducted in a laboratory situation where the students of an undergraduate database course would use the two versions of the ITS. Their student logs would be recorded and a pre and post test would be administered. We could assess whether the new system performed better or worse than the original version of SQL-Tutor. Subjective information about the system and students' experiences could be collected from a suitably constructed questionnaire.

Several earlier evaluations of SQL-Tutor used pre and post testing mechanisms to evaluate the amount of learning achieved by the students [6, 5]. This information is invaluable in assessing the performance of any modification to an ITS.

Unfortunately, the logistics of conducting a controlled evaluation of the two systems, the new problems selection agent and the original SQL-Tutor, are complicated and time consuming for both the student and evaluator. Factors such as having enough students to complete the required tasks and administering and collection of pre and post tests may be out of the evaluator's control.

The alternative option, an offline evaluation, provides a strictly controlled environment in which to evaluate the error predictor and problem selection networks. This involves the random selection of a set of student logs that would be used to test the networks. Logistics of running controlled evaluations with actual students will not be required. Unfortunately, the subjective evaluation information would not be available or the ability to do pre and post tests to determine the amount of learning achieved. NeuroSolutions does provide `C++` source code for a network built, but due to the licence purchased, that option was not available. The overhead and time to develop a network of comparable quality is better spent on trying different configurations and experiments.

With careful consideration and control an offline evaluation of the two networks would be able to test the effectiveness of the error predictor and the problem selector networks. Section 4.1 covers the performance measures available within NeuroSolu-

tions and how they are used to determine the accuracy of the networks. In Section 4.2, results of the performance of both error predictor and problems selector are discussed. In Section 4.3 we discuss the performance of the networks.

4.1 Performance Measures

NeuroSolutions offers six performance values that can be used to measure the performance of the network for a particular data set. The following sections cover all six measures: mean square error (MSE), normalised mean squared error (NMSE), correlation coefficient (r), percentage of error (%Error), Akaike's information criterion (AIC), and Rissanen's minimum description length (MDL). These measures were used to assist training and evaluate the performance of the networks built.

4.1.1 MSE

The mean square error [27] is based on the simple rule of continuously modifying the strengths of the input connections to reduce the difference between the desired output value and the actual output of a processing element. This rule changes the synaptic weights in a way that minimizes the mean square error of the network. The mean square error formula is:

$$MSE = \frac{\sum_{j=0}^{P} \sum_{i=0}^{N} (d_{ij} - y_{ij})^2}{N\,P} \tag{4.1}$$

where:

$P =$ Number of output processing elements
$N =$ Number of exemplars in the data set
$y_{ij} =$ Network output for exemplar i at processing element j
$d_{ij} =$ Desired output for exemplar i at processing element j

4.1.2 NMSE

The normalised mean squared error (NMSE) compares the mean of a series against predicted values. If the NMSE is greater than one then the predictions are worse than the series mean. If the NMSE is less than one, then the predictions are better than the series mean. The normalised mean squared error:

$$NMSE = \frac{P\,N\,MSE}{\sum_{j=0}^{P} \frac{N \sum_{i=0}^{N} d_{ij}^2 - (\sum_{i=0}^{N} d_{ij})^2}{N}} \tag{4.2}$$

where:

$P =$ Number of output processing elements
$N =$ Number of exemplars in the data set
$MSE =$ Mean square error
$d_{ij} =$ Desired output for exemplar i at processing element j

4.1.3 Correlation Coefficient

The size of the mean square error (MSE) can be used to determine how well the network output fits the desired output, but it does not necessarily reflect whether the two sets of data move in the same direction. For instance, by simply scaling the network output, we can change the MSE without changing the directionality of the data. The correlation coefficient (r) solves this problem. By definition, the correlation coefficient between a network output x and and a desired output d is defined in Equation 4.3.

$$r = \frac{\frac{\sum_i (x_i - \overline{x})(d_i - \overline{d})}{N}}{\sqrt{\frac{\sum_i (d_i - \overline{d})^2}{N}} \sqrt{\frac{\sum_i (x_i - \overline{x})^2}{N}}} \tag{4.3}$$

The correlation coefficient is confined to the range [-1,1]. When r = 1 there is a perfect positive linear correlation between x and d, that is, they covary, or vary by the same amount. When r =-1, there is a perfectly linear negative correlation between x and d, that is, they vary in opposite ways (when x increases, d decreases by the same amount). When r =0 there is no correlation between x and d. In that case, the variables are uncorrelated. Intermediate values describe partial correlations. For example, a correlation coefficient of 0.8 means that the fit of the model to the data is reasonably good.

4.1.4 %Error

Denormalised error for overall accuracy of the network. The value calculated is the accuracy of the desired output compared to the predicted output. This value may be misleading. For example, the output value is in the range of 0 to 100. For an exemplar the desired output is 0.1 and the actual output is 0.2. The percent error for this exemplar is 100.

$$\%Error = \frac{100}{N\,P} \sum_{j=0}^{P} \sum_{i=0}^{N} \frac{|dy_{ij} - dd_{ij}|}{dd_{ij}} \tag{4.4}$$

where:

$P =$ Number of output processing elements
$N =$ Number of exemplars in the data set
$dy_{ij} =$ Denormalised network output for exemplar i at processing element j
$dd_{ij} =$ Denormalised desired output for exemplar i at processing element j

4.1.5 AIC

Akaike's information criterion (AIC) [1] is used to measure the tradeoff between training performance and network size. The goal is to minimise this term to produce a network with the best generalisation.

$$AIC(k) = N \ln(MSE) + 2k \tag{4.5}$$

where:

$k =$ Number of network weights
$N =$ Number of exemplars in the training set
$MSE =$ Mean square error

4.1.6 MDL

Rissanen's minimum description length (MDL) [21] criterion is similar to the AIC in that it tries to combine the model's error with the number of degrees of freedom to determine the level of generalisation.

$$MDL(k) = N\ln(MSE) + 0.5k\ln(N) \tag{4.6}$$

$k =$ Number of network weights
$N =$ Number of exemplars in the training set
$MSE =$ Mean square error

4.2 Results

4.2.1 Error Predictor Performance

The initial training measures after 600 epochs are presented in Table 4.1. The network had been trained with the 21 student log files from the 2001 evaluation of SQL-Tutor. The MSE (0.37) measure is not near the termination value of 0.01. The correlation coefficient (0.67) is also lower than the 0.8 threshold. The %Error achieved 88.5% accuracy. Further training of the network, another 400 epochs (Table 4.2), produced a lower MSE (0.34), a higher r (0.71) and a lower %Error, yeilding an accuracy of 88.25%. This trained MLP became the starting network for each student.

Twenty nine student logs were chosen randomly from evaluations performed on SQL-Tutor from 1999 and 2000. Twelve of the logs came from the 1999 evaluation and the remaining seventeen were chosen from the 2000 evaluation. The number of completed problems in the selected logs ranged from one to 163. A total of 1053 completed problems were evaluated with an average of 39 completed problems per student. For each log, we started with a network in its initial state, trained from the 2001 training data. The student logs were supplied to the assigned network and updated accordingly. Twelve of the logs, from the 1999 evaluation, had a range of prediction accuracies from 74.2% to 100% (three cases), with an average prediction accuracy of 93.2%, and an average correlation coefficient $r = 0.92$. The seventeen logs from the 2000 evaluation had a range of prediction accuracies from 100% (four cases) to 48.2%, with an average accuracy of 89.3%. The average correlation coefficient of the seventeen logs is $r = 0.74$. These values were similar to the original base network (Table 4.1 and 4.2). The network had difficulty in predicting errors for two logs from the set of students selected from the 2000 evaluation. The two accuracies were 48.2% and 53.4%. If these students were omitted, the average accuracy becomes 94.6% with a $r = 0.85$. The overall average accuracy of the networks is 91% with a r value of 0.82. These values are a good indication that the network is of high quality and that the network is able to generalise well.

Measure	Value
MSE	0.37
NMSE	0.55
r	0.67
%Error	11.54
AIC	-1165.42
MDL	- 642.29

Table 4.1: Active performance of the Error Predictor Network after 600 epochs

Measure	Value
MSE	0.34
NMSE	0.50
r	0.71
%Error	10.75
AIC	-1308.89
MDL	- 785.77

Table 4.2: Active performance of the Error Predictor Network after 1000 epochs

The accuracy of the networks improved after further problems were supplied to the network. On average, the network settled, and produced correct predictions, after eight problem submissions.

4.2.2 Problem Selector Performance

The initial problem selection network performance measures (Table 4.3) showed a high correlation ($r = 0.93$) and a low mean square error (MSE=0.02). The accuracy of the network is 81%.

The 29 log files used to evaluate the error prediction network have been used to determine the network's accuracy. The prediction accuracy had be determined by the comparision of the network output with the problem that had been selected by SQL-

Measure	Value
MSE	0.02
NMSE	0.13
r	0.93
%Error	19.57
AIC	-5894.13
MDL	-5374.48

Table 4.3: Active performance of the Problem Selection Network after 1000 epochs

Tutor in the log file. The error prediction values from the first network were added to each of the student files and supplied to the problem predictor. All problem submissions were supplied to the problem selecting network, whether the student needed intervention due to having difficulty or not. These were supplied to the network to assess its ability to generalise all cases.

The prediction accuracies of the networks ranged from 10% to 94%. Five of the networks prediction accuracy were under 20%. The average prediction accuracy for all the networks was 79% and a correlation coefficient value $r = 0.77$. The averages for both accuracy and correlation are 84% and $r = 0.8$ if the five networks with their accuracies below 20% were omitted.

The accuracy for the 1999 networks ranged from 10% to 93%, with an average of 61% and an average correlation coeffient of $r = 0.78$. Three of the five networks that performed poorly were in the 1999 set. The average accuracy increased 16% to 77% and coeffiecient $r = 0.88$ if the three networks were omitted.

The accuracy for the 2000 logs ranged from 20% to 94%, with an average of 81.5% and an average correlation $r = 0.76$. Two of the five students that performed poorly were in this set. The average accuracy increased 8% to 89.5% and coeffiecient $r = 0.74$ when the two outliers are omitted. These results suggest the average prediction accuracy for the networks is high, but correlation of the network is reduced. The prediction accuracies of the networks suggests that the network has difficulty in selecting an appropriate problem. The networks that did settled down and produced more correct predictions, were supplied longer logs of greater than 30 completed problems.

The five networks that performed poorly are discussed in Section 4.3.

4.3 Discussion

The error predicting network performed well, achieving an average prediction accuracy of 91%. Two networks out of 29 performed poorly. These networks achieved prediction accuracies of 48% and 53%. Both the student logs indicate that the students may have had difficulty. Several of the problems in both situations have been submitted with an increasing requests for help. After viewing the complete solution, one student had logged out and the other one chose a different problem to work with. This would explain the high error recorded. One problem was submitted eight times and the values were the same, because it seems the student must have been convinced that the solution was correct. They did not change the solution, and therefore no changes were recorded in the measures. If the error predictor got the prediction wrong for the first submission, the proceeding submissions requesting more help will also be wrong.

The average prediction accuracy of the problem selector achieved is 79%. There were five networks that performed poorly, with prediction accuracies ranging from 10% to 20%. This was due to the prediction of the problem number. One of the five student logs only had three completed problems and the network had not been able to generalise well yet. The other four logs had blocks of problem submissions where the exemplars were all identical, due to the student requesting more help for their submission. The network has selected an incorrect problem n times and thus reduced the accuracy considerably.

There was an increase in prediction accuracy for the longer student logs, with more than 30 solved problems. The network was able to make better generalisations over time. Unfortunately, this does not aid a student who is new to a domain. Being presented with seemingly random problems at first will not aid the learning.

The network had difficulty in selecting a problem. This is directly related to the fact that there is no relationship between problems which have numbers in close proximity to one another. For example, problem 10 can have a problem level of two, and problem 11 may have a problem level of six and may in fact be a problem associated with a different database. A student may be best suited to work with problem 48 and the network may output 47.58. Here, the difference is within the acceptable bounds (± 0.5 of problem number) of being correct. However, the problem occurs when the output of the network is 47.34. The network determines the appropriate problem to present is 47. Unfortunately, the problem level between the two may differ, and may be associated with different databases. Therefore, the problem may be too difficult and/or cover a concept the student is not ready to attempt.

It may be possible to adjust the parameters for both the activation function and problem selection network configuration to achieve greater selection accuracies. This and several other approaches including, domain selection and problem level recommendation is presented in further work (Section 5.2).

Chapter 5

Conclusions and Further Work

Our goal has been to produce a problem selection component for SQL-Tutor to aid pedagogical decisions. It is not necessarily the best strategy to present a student with a more difficult problem if they struggled with the last problem completed. It may be more beneficial to present a problem of lesser or equal difficulty. The approach taken used artificial neural networks. The task is broken into two components, an error prediction network and a problem selection network.

5.1 The Domain Being Learned

Neural network design is dependent on the historical data and what is required to be learned. There is no formula for the design of an optimal network. We have found that 'trial and error' in designing a network is required to find a solution. There is no way to quantify that the network produced is optimal, as there are too many parameters to tune. We have settled for a 'good' approximation for both networks.

The error prediction network performed well, achieving prediction accuracies better than 88.5%. Twenty two of the 29 networks (76%) achieved prediction accuracies in excess of the initial base network's prediction accuracy. The performance of the error predictor has indicated that the network produced is robust in its ability to generalise the student logs from three years of evaluations on SQL-Tutor, 1999, 2000 and 2001. There were several cases when the network could not correctly predict the errors. It can be said that in a classroom situation there are always students that are not suited for that style of teaching. There is not a single teaching/learning strategy that can accommodate all students. Overall, the performance of the error predictor is excellent and is well suited for this domain.

Performance of the problem selecting network achieved satisfactory results, with an average problem selection accuracy of 79%. Of the 29 networks, nineteen achieved problem selection accuracies of over 81% (the accuarcy of the initial base network). The problem selection inaccuracies occurred because the network selected problems greater than the ± 0.5 of the problem number. For example, a student should be presented with problem 47 and the network output is 46.03. The network selects problem

46 when the student would be better suited to work with problem 47. Other strategies need to be employed to increase the efficiency of the problem selector. It may be a better strategy to determine the domain of problems the student needs to be presented with.

A limitation of neural networks is the dependency on historical data. Regardless of what the network is trained with, it is going to mimic or generalise existing problem selection strategies. Over time the network may change how it learns, as new examples are learned, but fundamentally the strategy remains the same. Despite which network configuration and activation algorithm used to build an ANN, historical data is needed to train it. It may be possible to produce 'customised' historical data set to reflect a new problem selection strategy, and then train the network.

The error prediction network performed well and is robust in its ability to make pedagogical decisions, determining whether a student is having difficulty. This network is good and further work is not necessary. The performance of the problem selection network is satisfactory, but more work is necessary in order for it to be of practical use. The approach to select problem number has its flaws. However, the domain is well suited for ANNs and further investigation into problem selection strategies can address the inadequacies of the current problem selection network.

5.2 Further Work

This investigation into the feasibility of ANN usage has opened more opportunities into research in the area of problem selection with SQL-Tutor and with other constraint based ITSs such as KERMIT [24] and NORMIT [11].

Our research focused on problem selection. An alternative to this approach would be to to recommend or select a suitable type of problem for a student to work with. Drawing from other students' experiences, from historical data, a student with little experience in a new area can be presented with a problem of a certain problem type or work from a particular database. For example, SQL-Tutor works with the SELECT statement of the SQL database language. There are six clauses: SELECT, FROM, WHERE, HAVING, GROUP BY and ORDER BY. Depending on the progress of the student when a difficult concept has been presented, the recommendation for a suitable type of problem can be offered. Currently, SQL-Tutor does offer the student a choice to select a clause or database to work with. The problem type recommendation could be bypassed if a student does choose to work on a specific clause and/or database. The recommendation of which clause to work with is well suited for ANNs. The historical data and framework is readily available to investigate this possibility. The clause can be recommended and a suitable problem can be selected from a pool of problems for that clause. We could go even further by developing an ANN to do the specific problem selection as the network developed already could be easily used. Another option for selection from the clause pool of problems is the investigation of various heuristics:

- A simple approach, go through the entire problem pool sequentially.
- An intelligent approach, determine the extent of the student's knowledge of the concept.

There is no distinction between two students with the same values for a problem submission. The student can achieve the same results by different means. For example, a student struggling may work slowly to achieve their goal and another adept student may achieve the same goal quickly. There is a difference between the two students, but this is not reflected in the sixteen values used in our research. There is more student information available in the student model that could be used to distinguish the two students, as, for example, the collection of past problems attempted and completed.

A limiting factor in the evaluation of our problem selection agent is that there was no online evaluation with real students. The evaluation of the system was accomplished using historical data and student logs. Student reactions and usability studies would provide great insight into the overall design and functionality of the system. Also, areas of difficulty and/or frustration could be identified, and with this valuable information the system can be improved to provide a better teaching platform. From previous additions and improvements to SQL-Tutor [6, 7], pre and post testing have be utilised to determine the effectiveness of the system. An online evaluation would provide the necessary information for determining the effectiveness of our system.

Acknowledgments

I would like to thank Dr. Antonija Mitrović, for her expertise and patience, Eugénie Sage, for her editing and love, Stacey Mickelbart, for her expert editing skills, and last but not least my fellow honours students, for being able to put up with me for a year. God bless.

Bibliography

[1] AKAIKE, H. Information theory and extension of the maximum likelihood principle. In *2nd International Symposium on Information Proceedings* (1973), pp. 267–281.

[2] ANDERSON, J. *Rules of the Mind*. Laurence Erlbaum Associates, 1993.

[3] BECK, J. E., AND WOOLF, B. P. Using a learning agent with a student model. In *4th International Conference on Intelligent Tutoring Systems* (1998).

[4] GOUTTE, C., AND LARSEN, J. Adaptive metric kernel regression. In *Neural Networks for Signal Processing VIII–Proceedings of the 1998 IEEE workshop* (1998), pp. 184–193.

[5] MARTIN, B., AND MITROVIĆ, A. Evaluating the effects of open student models on learning. In *Proceedings 2nd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems* (Malaga Spain, 2002), P. de Bra, P. Brusilovsky, and R. Conejo, Eds., pp. 294–305.

[6] MAYO, M., AND MITROVIĆ, A. Using a probabilistic student model to control problem difficulty. In *Proceedings ITS'2002* (Springer, 2000), C. F. G. Gauthier and K. VanLehn, Eds., pp. 524–533.

[7] MAYO, M., AND MITROVIĆ, T. Using a probabilistic student model to control problem difficulty. In *Proceedings 5th International Conference ITS' 2000* (2000).

[8] MCCULLOCH, W., AND PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5* (1943), 115–133.

[9] MITCHELL, T. *Machine Learning*. McGraw-Hill, 1997.

[10] MITROVIĆ, A. SQL-Tutor: a preliminary report. Tech. rep., Computer Science Department, University of Canterbury, 1997.

[11] MITROVIĆ, A. NORMIT, a web-enabled tutor for database normalization. Accepted for ICCE 2002, December 2002.

[12] MITROVIĆ, A., MARTIN, B., AND MAYO, M. Using evaluation to shape its design: Results and experiences with sql-tutor. In *User Modeling and User-Adapted Interaction* (2002), vol. 12, pp. 243–279.

[13] NEURODIMENSION INC. Help - Linear transfer function, 2002. NeuroSolutions Help.

[14] NEURODIMENSION INC. Help - LinearTanh, 2002. NeuroSolutions Help.

[15] NEURODIMENSION INC. Help - Momentum, 2002. NeuroSolutions Help.

[16] NEURODIMENSION INC. Help - Tanh, 2002. NeuroSolutions Help.

[17] NEURODIMENSION INC. Level Summary - Educator, 2002. `http://www.neurosolutions.com/products/ns/levels.html`.

[18] NEURODIMENSION INC. NeuroSolutions, 2002. `http://www.nd.com`.

[19] OHLSSON, S. Constraint–based student modeling. In *Student Modeling: the Key to Individualized Knowledge–based Instruction* (1994).

[20] PINNACOR. NetProfit. `http://www.pinnacor.com/`, 2002.

[21] RISSANEN, J. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics 11*, 2 (1983), 416–431.

[22] ROSENBLATT, F. Two theorems of statistical separability in the perceptron. *Symposium on the Machinization of Thought Processes* (1959), 421–456.

[23] STERN, M., BECK, J., AND WOOLF, B. P. Adaptation of problem presentation and feedback in an intelligent mathematics tutor. In *Intelligent Tutoring Systems* (1996), pp. 605–613.

[24] SURAWEERA, P., AND MITROVIĆ, A. Kermit: a constraint-based tutor for database modeling. In *Proc. 6th Int. Conf on Intelligent Tutoring Systems ITS 2002* (Biarritz, France, 2002), G. G. S. Cerri and F. Paraguacu, Eds., pp. 377–387.

[25] WALKRICH INVESTMENT ADVISORS INC. Walkrich Investment Advisors. `http://www.walkrich.com/`, 2000.

[26] WANG, T., AND MITROVIĆ, A. Using neural networks to predict student's performance. Accepted for ICCE 2002, December 2002.

[27] WIDROW, B., AND HOFF, M. Adaptive Switching Circuits. *WESCON Conv Record 4* (1960), 96–104.