Rule Generation from Neural Networks for Student Assessment

M. J. McAlister S. Wermter
School of Computing, Engineering and Technology,
Informatics Centre, University of Sunderland, St Peter's Campus,
St Peter's Way, Sunderland, SR6 0DD, UK
moira.mcalister@sunderland.ac.uk stefan.wermter@sunderland.ac.uk

Abstract

HyValue is a hybrid electronic submission system which utilizes techniques from natural language processing, neural networks and rule based systems to accept, evaluate and mark work submitted by a student for reading or writing. This paper describes the theory behind the system design and the development of the individual components and their interaction. Issues addressed include the definition of sentence structure, fuzzy rule construction and integration with a knowledge base containing the marking rubrics for reading and writing. An evaluation of the system is provided and conclusions drawn.

Introduction

A central task of all education programmes is assessment. A number of attempts have been made to automate the process [1,2,3]. Several researchers have designed performance-based rubrics [4,5]. However, to date nobody has automated a scoring system for which employs reading and writing rubrics. Since any type of grading is a classification task, the use of a classification model will help to standardize the grading [6]. The major issues lie in the fuzzy nature of a marking procedure which requires linguistic interpretation.

HyValue is a collection of symbolic and neural component systems. It reads in electronically submitted student work and generates statements containing certainty values which relate the accuracy of the answer provided to the question with regard to use of keywords, grammar and spelling. These statements are then compared with an appropriate marking rubric of fuzzy rules; either reading or writing, to produce a mark for the work.

The components of the system are shown in Figure 1.

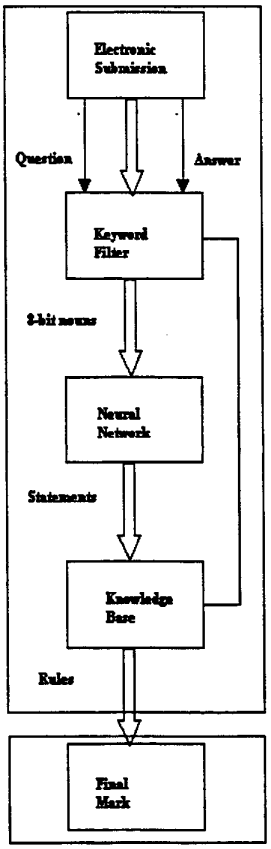

Figure 1: Components of HyValue

The components of the system are required to pass information in specified formats that allow the custom-built software and applications software to interact.

Electronic Submission System

The electronic submission system is a custom-built software system that accepts questions and answers, differentiating between the two. The work is thus

0-7803-5529-6/99/$10.00 ©1999 IEEE

submitted in a template format which is in the form of a plain text file. The template is global and the blank template is used to enter questions and answers as they would be for an exam paper. A typical template is shown in Figure 2.

Q1	What is a .....
A1	Student answer
Q2	What do the variables.....
A2	Student answer

Figure 2: Typical Template

The submission system operates in two modes; training and submission.

Training

The user can choose to train the system manually by typing in cases or by using presaved examples which are stored in template format. In training mode questions and answers (which represent good to bad answers) are provided for the reading and writing type of questions. Two examples are shown in Figure 3 (a and b).

```
(Pseudo code to demonstrate the if statement)
Begin(c5_dpp\ex_o1.c)
        Begin(initize seq.)
        Include libraries for stand i/o
        Set up variables
        End(initize seq.)

        Begin(main seq.)
        Output to screen "is it raining?"
        Read reply
        If reply=yes
        Copy "raincoat" to variable raincoat
        Output to screen "before your leave take your
        raincoat"
        End(main seq.)
End(c5_dpp\ex_01.c)

In the pseudo code what variables are declared at the
start of the code and what are they set to?

reply
garment
```

Figure 3(a): Reading Example Q and A

```
#include <stdio.h>
#include <stlib.h>
void main(int argc, char *argv{})
{
FILE *stream
If(argc !=2)
{
printf("you must give filename");
exit(EXIT_FAILURE);
}
stream = fopen(arg[1], "r");
if stream != NULL)
{
printf (("%s exists", arg[1]);
fclose(stream);
exit(EXIT_SUCCESS);
else
{
printf("%s does not exist", arg[1]);
exit(EXIT_FAILURE);
}

#include <stdio.h> is a preprocessor directive.....
```

Figure 3(b): Writing Example Q and A

The information is input using an editor spawned by the submission system. The training procedure ensures that the template file is split with regard to questions and answers. The question and answer are then dealt with separately by different components of the whole system. Upon completing the task the user then exits the editor and returns to the submission system.

Submission

The submission system takes the prestored template which has been created for the exam and splits the file to produce questions and student answers. The splitting procedure is used within the system to determine the level of accuracy of answers in comparison to the question asked.

Advanced Keyword Filter

The keyword filter system analyzes the structure of a question and answer and breaks them down into its component parts. A basic sentence involves an article, noun, verb and another noun. See Table 1.

Article	Noun	Verb	Noun
The	house	is	blue

Table 1: Simple Sentence Structure

Obviously sentences can be more complex involving the use of prepositions, etc. to connect nouns and articles. See Table 2.

Article	Noun	Prep.	Article	Noun
The	colour	of	the	house

Table 2: More Complex Sentence Structure

Sentences can also involve the use of conditional and logic statements such as "if" and "and". For the system to work effectively it had to incorporate a number of features:

(1) Each sentence must be broken into its component parts to form a pattern which can be passed to the Neural Network.
(2) Using word positions and comparison to a prestored file of words spell checking and grammar features can be introduced.

An algorithm is used to assign numerical values to the component parts of the sentence. Nouns are removed from the sentence and are checked against a prestored database file. All other component parts of the sentence, i.e. articles, verbs and so on are removed. The process is shown in Table 3.

0	2	3	-2
What	house	is	blue

Table 3: Assigning Values

Note, that nouns identified in the database file are assigned the value of 2 and those not in the file are assigned a temporary value of –2 and later added to the database. By assigning values to other components, which are stored in separate database files, the grammatical structure of the sentence can be verified by the system. This is principally necessary for training the system with example sentences.

The sentence structure defined numerically is then compared to a database of sentence structures and a percentage match determined. At this stage, any –2 values for nouns are converted to 2. The nouns database is then updated to include any new nouns. A certainty unit is then calculated for each sentence as follows:

Certainty Unit = Max Certainty/Size of structure
Maximum certainty being 1 and the size of the structure being the length of the sentence.

The nouns in the answer are passed to the Fuzzifier as a text file and the nouns in the question are then assigned an 8-bit binary number and passed to the Neural Network. This transition is necessary for the information to be input to the Neural Network.

Neural Network

A number of Neural Networks were examined with regard to rule generation. Only two Neural Network applications were discovered which generated the needed fuzzy rule format. These were FuneGen and Nevprop. Of these two only Nevprop was available as a complete application tool, FuneGen being available in a limited version only.

Nevprop is a general-purpose back-propagation network written in C for any platform [7]. The application uses the Quickprop algorithm with the inclusion of force gradient descent, added generalization, stopped training, c index and interface enhancements.

The Neural Network is trained to identify question depending on answers. Therefore depending on the keywords detected, the Neural Network generates the corresponding question.

The 8-bit binary number and the corresponding certainty value generated by the Advanced Keyword Filter form the inputs to the network. The network recognizes sequences of 8-bits pattern which represent keywords used in the question. The corresponding output is a sequence of 1s or 0s. These outputs are then stored in a file. The process is visualized in Figure 4.

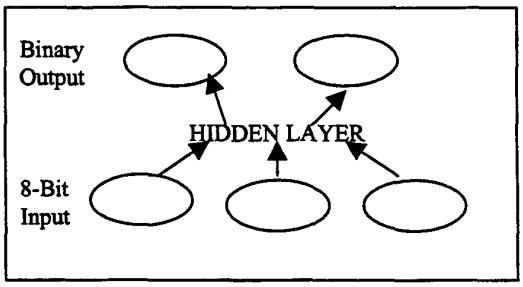

Figure 4: Neural Network Input/Output

Note, that the file is generated with a unique number in binary form which indicates that the keywords found correspond to a similar question in the database file of questions which the Neural Network was trained with. The file with the unique number contains the keywords and is a text file.

Fuzzifier

The Fuzzifier handles the student answer from the split file of the Electronic Submission System. The system is required to compare the strings in the user's answer with those of the file produced by the Neural Network and deduces a match. The Fuzzifier converts the unique binary number to a decimal number and stores it as a text file. As keywords are located in the user's answer the certainty determined in the Advanced Keyword Filter is used to devise statements concerning the quality of the user's response. Features examined include number of keywords and thereby definition grammar and spelling.

The statements produced take the following form for reading:

THE RESPONSE HAS LITTLE RELATION TO THE QUESTION

And for writing:

THE MECHANICS HAVE SERIOUS_ERRORS

The statements are then passed to the Knowledge Base System, FEST.

Knowledge Base System

The application tool used is FEST. It is a fuzzy expert system tool which takes free text statements and then uses its knowledge base and inference engine to come to a conclusion.

The knowledge base system consists of two knowledge bases each containing two rubrics:

(1) Reading.
(2) Writing.

Both rubrics are in the form of free text rules. FEST uses data structures to hold these rules and generates its own rules to improve the efficiency of the system every inference cycle. The system compares the free text rules of the appropriate rubric with the statements generated in the Fuzzifier. Examples are shown in Figure 5 (a and b).

```
IF THE RESPONSE IS LITERAL THEN RESULT IS 4
IF THE RESULT IS 4 AND THE RESPONSE HAS
ONLY_SOME_INFORMATION THEN THE RESULT
IS 5.
```

Figure 5(a): Coded Reading Rubric

```
IF MECHANICS HAVE SERIOUS_ERRORS THEN
THE LMEC IS 1
IF LMEC IS 1 AND THE MECHANICS HAVE
FREQUENT CAPITAL_PUNCT_ERRORS THEN
LMEC IS 2
IF LMEC IS 2 AND THE MECHANICS HAVE
INTERFERES_WITH_MEANING THEN LMEC 3
IF LMEC IS 3 AND THE MECHANICS HAVE FEW
WORDS_SPELLED_CORRECTLY THEN LMEC 4
IF LMEC IS 4 AND THE MECHANICS HAVE
GRAMMAR_SYNTAX_ERRORS THEN LMEC 5
IF LMEC IS 5 THEN THE MECHANICS IS 1 POINT
```

Figure 5(b): Coded Writing Rubric

Note that although rules are fired sequentially if one condition is not met then that rule is not generated.

FEST automatically generates reports on the results which indicate which rules where fired to produce the overall grade.

Experimental Results

Each component of the system was evaluated in the process of development and the results are shown in Table 4.

Component	% Success
ESS	100
AKE	100
NN	75
Fuzzifier	75
FEST	N/A

Table 4: Success Measurements

37 test cases have been run through the system up to the Fuzzifier. 15 are reading (pseudo code) and 22 are writing (articulate description of C source code). The test cases take considerable time to develop given the extensive rubrics used for marking reading and writing.

The Electronic Submission System and the Advanced Keyword Search have now been fully tested. During the testing phase the keyword filter was revised in view of suggestions concerning spelling and grammar. The Neural Network and Fuzzifier have been partially tested and some minor problems have only occurred due to memory overload when handling 256 8-bit file generation.

The Knowledge Based System has not yet been tested but the reading and writing rubrics have been coded.

Conclusions

A combination of symbolic and neural components have been effectively combined to produce a system which is capable of being used within the realm of assessment in a diverse range of areas.

References

1. Forsythe, G. and Wirth, N. Automatic Grading Programs, Communications of the ACM, 8(5), pp 275-278, 1965.
2. Conati, C. and VanLehn, K. POLA: A Student Modeling Framework for Probabilistic On-Line Assessment of Problem Solving Performance, Proceedings of the 5th Conference on User Modeling, pp 75-82, User Modeling, 1996.
3. Roccetti, M. and Salomoni, P. Using Bayesian Belief Networks for the Automated Assessment of Students' Knowledge of Geometry Problem Solving Procedures, Technical Report, University of Bologna, Department of Computer Science, Number BOLOGNA#UBLCS-97-7, p 31, 1996.
4. Brewer, R. Exemplar's A Teacher's Solution, Underhill, VT:Exemplar, 1996.
5. Marzano, R. et al. Assessing Student Outcomes: Performance Assessment Using the Dimensions of Learning Model, Alexandria, VA:ASCD, 1993.
6. Ebert, C. Fuzzy Classification for Software Criticality Analysis, Expert Systems with Applications, 11(3), pp 323-342, 1996
7. Rumelhart, D. et al. Learning Internal Representations by Error Propagation, Parallel Distributed Systems, Ed. Rumelhart and McClelland, MIT Press, Cambridge, MA, 1986.

Acknowledgements

We would like to acknowledge the BSc Information Technology students who have assisted in researching, analyzing and coding the system as part of their Software Project Management module. Without their assistance and advice this system would not have reached such a developed stage in the 8 month period of development.