Using Tic-Tac-Toe for Learning Data Mining Classifications and Evaluations

Chen-Huei Chou

***Abstract*—Tic-Tac-Toe game is a popular two-player game played on a three by three grid. The first objective of this study is to examine whether machine learners can successfully classify Tic-Tac-Toe finished games. The second objective is to find out whether novices learning data mining classifications can successfully conduct experiments to evaluate the performance of machine learners using different evaluation methods. Seven machine learners are used and three evaluation methods are applied in this study. The experimental results show that machine classifiers can successfully judge the finished games and novice can correctly conduct evaluations. Also, in-depth instructional pedagogy further improves the correctness of evaluations.**

***Index Terms*—Classification, data mining, evaluation, Tic-Tac-Toe game.**

I. Introduction

Tic-Tac-Toe game is a popular two-player game played on a three by three grid. The player who can place the own marks three-in-a-row wins the game. To judge the winner of the game, it requires the marks on the grid and follows the game rule to find out the winner, either X or O. Such problem is an example of classification. Data mining techniques have been successfully applied to solve problems such as credit evaluation [1], forecasting financial performance [2], assessing risks of prostate cancer patients [3], generating document taxonomies [4], profiling Web usage in the workplace [5], classifying open source software development projects [6], and web query classification [7]. In this study, the first objective is to examine whether machine learners can successfully classify Tic-Tac-Toe games.

Conducting an experiment to evaluate machine learners' performance is normally not a challenge for experienced researchers. However, novice users may run into troubles even a tool is available for use. Since learning data mining concepts and techniques is usually hard for novices, a user friendly context may benefit the learning processes. The Tic-Tac-Toe game has been a commonly played game since their childhoods. The second objective of the study is to examine whether novices can correctly evaluate the performance of machine learners classifying the familiar Tic-Tac-Toe games. Specifically, the study attempts to answer the following research questions:

- *Can machine learning classifiers successfully classify Tic-Tac-Toe games?*
- *What is the best machine learning classifier in classifying Tic-Tac-Toe games?*
- *In learning classifications, can novices correctly evaluate the trained classifiers using different evaluation methods?*
- *What are the common issues from novices' evaluations?*

The rest of the paper is organized as follows. Relevant background is discussed in the next section. Experimental methods are then described. Findings of experiments are reported. Finally, conclusions are drawn and future directions are highlighted.

Manuscript received March 8, 2013; revised June 24, 2013.

C.-H. Chou is with the School of Business, College of Charleston, Charleston, SC 29424 USA (e-mail: chouc@cofc.edu).

II. Literature Review

A. Tic-Tac-Toe Game

The Tic-Tac-Toe, also called noughts and crosses, is originally designed as a paper and pencil game. Two players take turns marking down their own Xs or Os in a three by three grid. A player who successfully place three marks in a row, a column, or a diagonal wins the game. Thus, the rule to win the game is "three-in-a-row." Fig. 1 shows a game won by X due to three X in the diagonal row. There are 255,168 possible games in total. Among the available games, 958 terminal configurations are reached when a winner is found [8]. The terminal configuration could be a game with a determined winner without using all nine cells such as the game in Fig. 1 or with all cells marked.

X	O	X
O	X	O
	O	X

Fig. 1. A Tic-Tac-Toe won by X.

B. Classification

Classification is one of the typical data mining techniques. It is a supervised learning process to build a model for future predictions (classifications). The model, generated from training data, is to associate characteristics of data observations with a desired category. The observations in the training data consist of a set of characteristics (input variables) and a given category (output variable). Tic-Tac-Toe game is a typical binary classification problem. In classifying Tic-Tac-Toe games, there are nine positional input variables and a given winner output variable. Each input variable can carry an "X", an "O", or nothing (blank). The binary winner output variable can be either "X" or "O". The training data is a set of games $\{g_1, g_2, \ldots, g_j\}$ with their output category $\{c_1, c_2\}$. A machine learning algorithm, a classifier, is used to find the model $c_i = f(g_j)$ from correct pairs of $<g_j, c_i>$ consistently, where game g_j is represented by nine positional variables. Once the classifier is built, a correct prediction is made when a game g_j can be assigned to the correct class c_i, either "X" or "O". Various classifiers have been successfully applied in several data mining problems such as credit evaluation [1], forecasting financial performance [2], assessing risks of prostate cancer patients [3], generating document taxonomies [4], profiling Web usage in the workplace [5], classifying open source software development projects [6], and web query classification [7].

C. Evaluation Methods

1) Training set method

The training set evaluation method is to utilize 100% of observations to build a classifier and apply the same set of observations as test set for evaluation. The performance of such evaluation method is normally optimal or overestimated while the built classifier may not perform well when using different test sets.

2) Holdout method

The holdout method, so called percentage split, is to randomly split the observations into two sets, a normally larger training set and a normally smaller test set. In this study, 70% of observations are used for training and 30% of observations are used for evaluation. Good evaluation results may be obtained in some lucky cases randomly allocating easy observations in the test set.

3) Cross validation method

In order to address the issues using training set and holdout method, cross validation provides a more rigorous evaluation procedure by randomly splitting observations into a number of subsets, e.g. n folds. The first n-1 folds are used for training and the left fold is used for testing. Next, it rotates the test set forward while still using the other n-1 folds for training. After repeating the procedure n times, the last procedure uses the last n-1 folds for training and the first fold for testing. Finally, the average of the n evaluation results is the overall performance using the cross validation method. "Extensive tests on numerous datasets, with different learning techniques, have shown that 10 is about the right number of folds to get the best estimate of error, and there is also some theoretical evidence that backs this up" [9]. Therefore, in this study, 10-fold cross validation is applied.

III. Evaluation Methods

A. Data Collection and Data Preparation

In a Tic-Tac-Toe game, nine input cells are available for nine positions. They are upper-left (UL), upper-middle (UM), upper-right (UR), middle-left (ML), middle-middle (MM), middle-right (MR), lower-left (LL), lower-middle (LM), and lower-right (LR). According to Schaeffer [8], 958 terminal configurations are reached when a winner is found assuming X is the first player. Therefore, the 958 games are used for this study. Table I lists 10 games (observations) out of the 958 games. The first nine variables are the positional input variables and last variable—Winner is the output variable. Notation "x" is used for input variables if the player X marks on the cells. Similarly, notation "o" is used if the player O marks on the cells. When a cell is blank without any mark, "b" is used. For the Winner variable, it could be "x" if the player X gets three marks in a row. "o" is stored if the play O is the winner.

TABLE I: Sample Observations of Tic-Tac Toe Games

UL	UM	UR	ML	MM	MR	LL	LM	LR	Winner
x	o	x	o	x	o	o	x	x	x
x	o	x	o	x	o	b	b	x	x
x	o	x	o	x	b	x	o	b	x
x	o	x	o	x	b	x	b	o	x
x	o	x	o	x	b	o	b	x	x
x	o	x	x	o	x	o	o	b	o
x	o	x	x	o	x	b	o	o	o
x	o	x	x	o	o	b	o	x	o
x	o	x	x	o	b	o	o	x	o
x	o	x	x	o	b	b	o	b	o
x	o	x	x	o	x	o	o	b	o

B. Experimental Design

Seven machine learning methods (Naïve Bayes [10], Neural Network, Support Vector Machine [11], [12], Logistic [13], k-Nearest Neighbor [14], and Decision Treess: ID3 [15] and C4.5 [16]) were used for performance evaluations using Weka [17] tool. Two experiments were conducted in order to answer the research questions.

1) Design of experiment 1

In first experiment, for each learning method, 10-fold cross validation was performed 10 times. Accuracy rate of a fold was treated as a performance outcome. Thus, 100 performance results were obtained for each learning method. ANOVA was then applied to determine whether the seven learners have similar performance. When difference was found, Bonferroni post hoc test was applied to differentiate the performance of seven learners.

2) Design of experiment 2

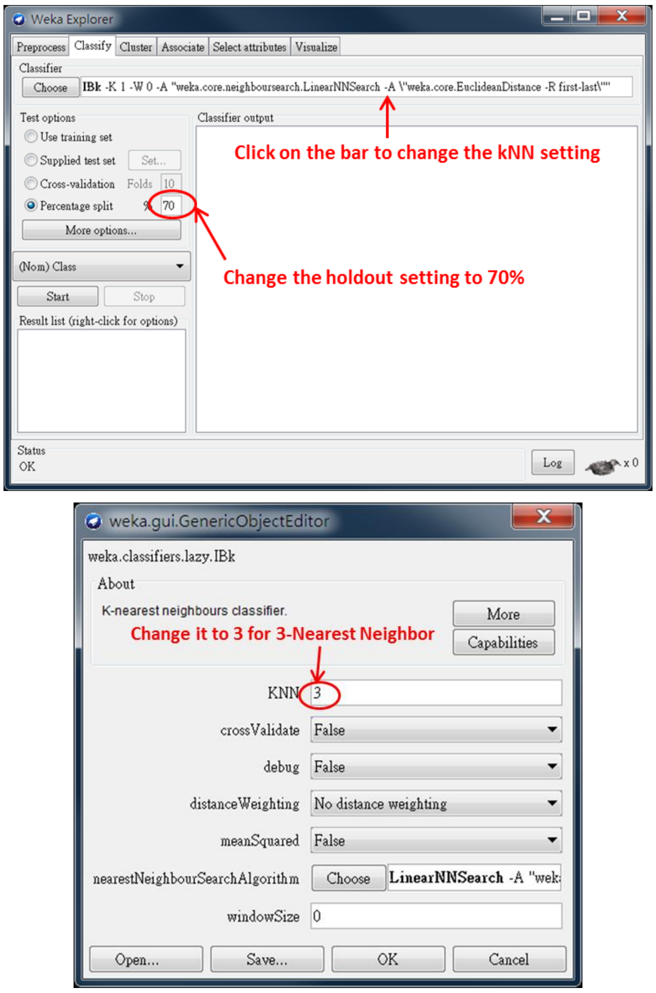
In the second experiment, novice participants were asked to conduct the experiments using Weka. Two pedagogies were used to instruct the use of Weka tool. Through both pedagogies, three evaluation methods (training set, holdout, and cross validation) were used for evaluating the seven learning methods. Finally, the 21 evaluation results from each participant were aggregated and t tests were conducted to determine whether the different pedagogies matter in instructing Weka.

In total, 136 students taking upper-level management information systems course in two different semesters participated in the experiments. After covering the concepts of machine classifiers and evaluation methods, the experiments took place in a computer lab. Different pedagogies were used in the two semesters. In the first semester, textual instructions using Weka tool and settings for the experiments were given (see one example in Fig. 2). Participants followed the printed instructions and conducted the experiments their own. In the second semester, same printed instructions were given with additional screenshots leading the processes. Additional messages highlighting the potential common mistakes were given as well (see one example in Fig. 3). In the beginning the experiment, another hands-on exercise was given to demonstrate the use of Weka tool.

When evaluating the performance of 3-Nearest Neighbor classifier using holdout method, you need to change the classifier setting and holdout setting.

1. After choosing the **IBk** classifier, click on the bar in the **Classifier** section in the top portion of screen and change the value to **3** for the **KNN** in the pop-up window.
2. Next, in the **test options**, choose **Percentage split** for holdout method and specify the percentage to **70**.
3. Finally, you can click on **Start** button to execute the evaluation. The result will be displayed in the **Classifier output** area.

Fig. 2. Sample textual instruction used in both semesters.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Choose
Click on the bar to change the kNN setting
Change the holdout setting to 70%
Log
x 0
weka.gui.GenericObjectEditor
About
More
Capabilities
Change it to 3 for 3-Nearest Neighbor
Choose
Open...
Save...
OK
Cancel

Classifier

```
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
```

Test options
- Use training set   Set...
- Cross-validation   Folds 10
- [✓ Percentage split]   % 70
- More options...

(Nom) Class

Start   Stop

Classifier output

Result list (right-click for options)

Status
OK

weka.classifiers.lazy.IBk

K-nearest neighbours classifier.

KNN	3
crossValidate	False
debug	False
distanceWeighting	No distance weighting
meanSquared	False
nearestNeighbourSearchAlgorithm	LinearNNSearch -A "wek
windowSize	0

Fig. 3. Additional screenshots used in the second semester.

C. *Experimental Environment*

Weka's default parameters were kept for most algorithms. For learning methods, two hidden layers and 50 training epochs were used for Neural Network. Also, three neighbors were set for k-Nearest Neighbor method. The J4.8 decision tree method was Weka's implementation of C4.5. All experiments were conducted in a computer lab equipped with 40 identical personal computers in terms of CPU, RAM, hard drive, operating system, Java runtime, and Weka tool. Each participant used one computer to perform the experiments. Accuracy rate was used for performance evaluations.

IV. Experimental Results

A. *Experiment 1*

Based on ANOVA analysis, there was a significant mean performance difference among the seven machine classifiers (p=.000), Bonferroni post hoc test was performed to conduct pairwise comparisons with a control of overall error rate. The results of pairwise t tests were listed in Table II. Based on the mean performance of 10 runs of 10-fold cross validation results (second column of Table II), 3-Nearest Neighbor outperformed other methods (see second column of Table II). The performance of two decision tree classifiers J4.8 and ID3 were significantly different at .05 level, but not at the .01 level. Pairwisely, no difference was found in a group of four classifiers: Neural Network (NN), Support Vector Machine (SVM), Logistic (LOG), 3-Nearest Neighbor (3NN). Since there was no significant difference found in the four classifiers, they were in the top performer's group. Naïve Bayes (NB) had the poorest performance among the seven classifiers.

TABLE II: Pairwise T Tests Results

Method	Mean Accuracy	Significance Test (p-value)						
		J4.8	ID3	NN	SVM	LOG	NB	3NN
J4.8	85.28	--	.033	.000	.000	.000	.000	.000
ID3	84.05	--	--	.000	.000	.000	.000	.000
NN	98.03	--	--	--	1.000	1.000	.000	.307
SVM	98.33	--	--	--	--	1.000	.000	1.000
LOG	98.28	--	--	--	--	--	.000	1.000
NB	69.64	--	--	--	--	--	--	.000
3NN	98.98	--	--	--	--	--	--	--

B. *Experiment 2*

The objective of the second experiment was to examine whether the novice participants were able to conduct data mining classifications correctly using three different evaluation methods. The same seven machine learners in experiment 1 were used in this experiment. The three evaluation methods used were training set, holdout, and 10-fold cross validation. Each participant was asked to conduct the experiments and report the performance results, accuracy rates, of the classifiers along with the original result generated from the Weka tool (see Appendix). 21 (7 classifiers by 3 evaluation methods) experimental results were collected from each participant. A total of 136 participants joined this experiment and similar size of participants was allocated in the two semesters. The participants came with similar background taking the same course. The mistakes found from participants conducting the classifications were summarized in the Table III. While getting correct results from Weka tool, three participants reported the results incorrectly in the first semester and one was found in the second semester. With such mistake, the results of cross validation method, for example, were reported as results of holdout method. Default settings were used for the classifiers except for NN and 3NN. Four participants in the first semester and one in the second semester left the default setting using NN. Similarly, three participants in the first semester and one participant in the second semester used the 3NN classifier without changing the settings. In addition, the default numerical split for the holdout is 66%. Three from the first semester forgot to change it to the required 70%. Based on the number of participants making mistakes evaluating machine learners, pedagogy in the second semester improved the evaluation processes.

TABLE III: MISTAKES FOUND FROM THE PARTICIPANTS

Mistake	Semester 1 (N=67)	Semester 2 (N=69)
Incorrect reporting	3	1
Incorrect parameter in NN	4	1
Incorrect parameter in 3NN	3	1
Incorrect holdout setting	3	0

Furthermore, to determine whether there was a significant difference between the correct accuracy rate of a classifier and the corresponding mean accuracy rates of the classifier prepared by participants using a particular evaluation method, one *t* test was applied. A total of 42 *t* tests were applied for the two semesters. Table IV, Table V, and Table VI summarizes the performance results of classifiers using training set, holdout, and cross validation evaluation methods, respectively.

TABLE IV: RESULTS USING TRAINING SET EVALUATION METHOD

Classifier	Goal	Accuracy Mean of Semester 1 (N=67)	Accuracy Mean of Semester 2 (N=69)
J4.8	93.7370	93.2148 [**.087**]	93.6039 [.321]
ID3	100.000	99.2307 [**.083**]	99.7579 [.321]
NN	98.4342	98.5297 [**.041**]	98.4539 [.394]
SVM	98.3299	98.3486 [.159]	98.3299 [*]
LOG	98.3299	98.3173 [.159]	98.3299 [*]
NB	69.8330	69.8567 [.220]	69.8300 [.321]
3NN	99.1649	99.1929 [.211]	99.1740 [.471]

TABLE V: RESULTS USING HOLDOUT EVALUATION METHOD

Classifier	Goal	Accuracy Mean of Semester 1 (N=67)	Accuracy Mean of Semester 2 (N=69)
J4.8	80.8362	81.2984 [**.050**]	81.0377 [.321]
ID3	82.5784	82.6785 [.724]	82.8309 [.321]
NN	98.6063	98.4134 [**.004**]	98.5786 [.277]
SVM	98.9547	98.9323 [.176]	98.9456 [.321]
LOG	97.9094	97.9120 [.860]	97.9155 [.321]
NB	70.7317	70.7047 [.358]	70.7187 [.321]
3NN	98.9547	98.9360 [.152]	98.9577 [.321]

TABLE VI: RESULTS USING CROSS VALIDATION EVALUATION METHOD

Classifier	Goal	Accuracy Mean of Semester 1 (N=67)	Accuracy Mean of Semester 2 (N=69)
J4.8	84.5511	84.7997 [.266]	84.4973 [.321]
ID3	83.2985	83.7863 [.168]	83.2881 [.321]
NN	98.2255	98.1876 [.145]	98.2189 [.624]
SVM	98.3299	98.3392 [.321]	98.3390 [.321]
LOG	98.3299	98.3236 [.321]	98.3238 [.321]
NB	69.6242	69.6470 [.185]	69.6403 [.321]
3NN	98.9562	98.9624 [.160]	98.9562 [.321]

First column of the tables indicates the machine classifiers. Second column reports the correct evaluation accuracy rates in percentage. Mean accuracy rates of classifiers prepared by participants in the first and second semesters are listed in the third and fourth columns. P-values of the *t* tests are listed below the accuracy rates. Statistically, no significant differences were found in most participants' evaluations, when compared with correct performance results. When training set evaluation method was used, mean accuracy rates of J4.8 (at .1 level), ID3 (at .1 level), and NN (at .05 level) from the participants in the first semester were significantly different from the corresponding correct results. Using the same evaluation method, all participants from the second semester obtained correct results for SVM and Logistic classifiers. When holdout method was used, the results of J4.8 and NN from the participants in the first semester were significantly different from the correct results at .05 level. No significant differences were found in the second semester.

V. CONCLUSIONS AND FUTURE DIRECTIONS

In the first experiment, 3-Nearest Neighbor classifier outperformed others with accuracy rate close to 99% classifying Tic-Tac-Toe games. Four machine learners, Neural Network, Support Vector Machine, Logistic, and 3-Nearest Neighbor, were the top performers with over 98% accuracy rates. There was no statistically significant difference among these four learners. In the second experiment, two different instructional pedagogies were conducted for novices to learn machine classifications. Both pedagogies showed that most novices can correctly conduct experiments to evaluate machine classifiers using data mining tool Weka. The graphical in-depth instructional pedagogy from the second semester did improve the correctness of evaluations statistically.

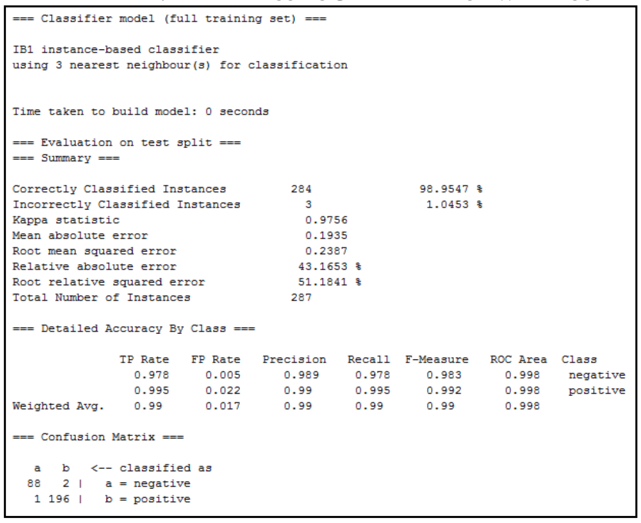
In this study, a familiar game was used for learning machine classifications. Future studies may adopt another familiar scenario—academic admission decision for learning classifications. The decision could be acceptance or rejection based on a set of applicant's characteristics such as admission exam score, years of working experience, grade point average, etc. Another direction of future studies is to explore the Tic-Tac-Toe rules generated from data mining techniques such as decision trees and association rules.

APPENDIX

APPENDIX A: REPORT TEMPLATE PROVIDED TO PARTICIPANTS

Classifier (Brief Setting in Weka)	Evaluation Methods	Accuracy Rate
Decision Tree J4.8 (Tree→J48)	Training Set	
Decision Tree J4.8 (Tree→J48)	10-fold Cross-Validation	
Decision Tree J4.8 (Tree→J48)	Holdout (70%)	
Decision Tree ID3 (Tree→Id3)	Training Set	
Decision Tree ID3 (Tree→Id3)	10-fold Cross-Validation	
Decision Tree ID3 (Tree→Id3)	Holdout (70%)	
Neural Network (Functions→MultilayerPerceptron, hiddenLayers=2, trainingTime=50)	Training Set	
Neural Network (Functions→MultilayerPerceptron, hiddenLayers=2, trainingTime=50)	10-fold Cross-Validation	
Neural Network (Functions→MultilayerPerceptron, hiddenLayers=2, trainingTime=50)	Holdout (70%)	
Support Vector Machine (Functions→SMO)	Training Set	
Support Vector Machine (Functions→SMO)	10-fold Cross-Validation	
Support Vector Machine (Functions→SMO)	Holdout (70%)	
Logistic (Functions→Logistic)	Training Set	
Logistic (Functions→Logistic)	10-fold Cross-Validation	
Logistic (Functions→Logistic)	Holdout (70%)	
Na ïve Bayes (Bayes→NaiveBayes)	Training Set	
Na ïve Bayes (Bayes→NaiveBayes)	10-fold Cross-Validation	
Na ïve Bayes (Bayes→NaiveBayes)	Holdout (70%)	
3-Nearest Neighbor (Lazy→IBk, k=3)	Training Set	
3-Nearest Neighbor (Lazy→IBk, k=3)	10-fold Cross-Validation	
3-Nearest Neighbor (Lazy→IBk, k=3)	Holdout (70%)	

APPENDIX B: PARTIAL RESULTS GENERATED FROM WEKA TOOL



```
=== Classifier model (full training set) ===

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         284               98.9547 %
Incorrectly Classified Instances         3                1.0453 %
Kappa statistic                          0.9756
Mean absolute error                      0.1935
Root mean squared error                  0.2387
Relative absolute error                 43.1653 %
Root relative squared error             51.1841 %
Total Number of Instances              287

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.978     0.005      0.989     0.978     0.983      0.998    negative
                 0.995     0.022      0.99      0.995     0.992      0.998    positive
Weighted Avg.    0.99      0.017      0.99      0.99      0.99       0.998

=== Confusion Matrix ===

   a   b   <-- classified as
  88   2 |   a = negative
   1 196 |   b = positive
```

REFERENCES

[1] A. P. Sinha and J. H. May, "Evaluating and Tuning Predictive Data Mining Models Using Receiver Operating Characteristic Curves," *Journal of Management Information Systems*, vol. 21, no. 3, pp. 249-280, 2005.

[2] S. Walczak, "An Empirical Analysis of Data Requirements for Financial Forecasting with Neural Networks," *Journal of Management Information Systems*, vol. 17, no. 4, pp. 203-222, 2001.

[3] L. Churilov, A. Bagirov, D. Schwartz, K. Smith, and M. Dally, "Data Mining with Combined Use of Optimization Techniques and Self-Organizing Maps for Improving Risk Grouping Rules: Application to Prostate Cancer Patients," *Journal of Management Information Systems*, vol. 21, no. 4, 2005.

[4] S. Spangler, J. T. Kreulen, and J. Lessler, "Generating and Browsing Multiple Taxonomies Over a Document Collection," *Journal of Management Information Systems*, vol. 19, no. 4, pp. 191-212, 2003.

[5] M. Anandarajan, "Profiling Web Usage in the Workplace: A Behavior-Based Artificial Intelligence Approach," *Journal of Management Information Systems*, vol. 19, no. 1, pp. 243-254, 2002

[6] R. E. Vlas and W. N. Robinson, "Two Rule-Based Natural Language Strategies for Requirements Discovery and Classification in Open Source Software Development Projects," *Journal of Management Information Systems*, vol. 28, no. 4, pp. 11-38, 2012.

[7] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, and O. Frieder, "Automatic classification of Web queries using very large unlabeled query logs," *ACM Transactions on Information Systems*, vol. 25, no. 2, 2007.

[8] S. Schaeffer. (2002). Tic-Tac-Toe (Naughts and Crosses, Cheese and Crackers, etc.). *Mathematical Recreations*. [Online]. Available: http://www.mathrec.org/old/2002jan/solutions.html

[9] I. H. Witten and E. Frank, *Data mining: practical machine learning tools and techniques*, 2nd ed. Morgan Kaufmann, 2005.

[10] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. the eleventh conference on uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 1995, pp. 338-345.

[11] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO Algorithm for SVM Classifier Design," *Neural Computation*, vol. 13, no. 3, pp 637-649, 2001.

[12] J. Platt, "Fast Training of Support Vector Machines using Sequential Minimal Optimization," in *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, eds., Cambridge, MA: MIT Press, 1998.

[13] S. le Cessie and J. C. van Houwelingen, "Ridge Estimators in Logistic Regression," *Applied Statistics*, vol. 41, no. 1, pp. 191-201, 1992.

[14] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37-66, 1991.

[15] R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no.1, pp. 81-106, 1986.

[16] R. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[17] Weka 3: Data Mining Software in Java. (2013). [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

Chen-Huei Chou received the B.S. in information and computer engineering from Chung Yuan Christian University, Taiwan in 1996, the M.S. in computer science and information engineering from National Cheng Kung University, Taiwan in 1998, the M.B.A. from the University of Illinois at Chicago, Chicago, Illinois, USA in 2004, and the Ph.D. in management information systems from the University of Wisconsin-Milwaukee, Wisconsin, USA in 2008.

He is an assistant professor of Management Information Systems and Decision Sciences in the School of Business at the College of Charleston, South Carolina, USA. His research has been published in MIS journals and major conference proceedings, including *Journal of Association for Information Systems*, *Decision Support Systems*, *IEEE Transactions on Systems, Man, and Cybernetics,* and *Journal of Information Systems and e-Business Management*. His areas of interests include web design issues in disaster management, ontology development, and data mining.