ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAITRISE EN GÉNIE
M.Ing

PAR
TRUONG, Le Hoang

# DÉVELOPPEMENT D'UN SYSTÈME D'IDENTIFICATION DES DIALOGUES PROBLÉMATIQUES DANS LE SYSTÈME DE DIALOGUE PERSONNE-MACHINE

MONTRÉAL, LE 28 AVRIL 2008

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE

M. Pierre Dumouchel, directeur de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Robert Sabourin, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

Mme Sylvie Ratté, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

# REMERCIEMENTS

Le travail de ma maîtrise, dans le cadre du programme de maîtrise en génie concentration avec mémoire de l'École de technologie supérieure (ÉTS), a été effectué au sein du Centre de recherche informatique de Montréal (CRIM).

J'aimerais, en premier lieu, exprimer mes sincères remerciements à mon directeur de recherche Monsieur Pierre Dumouchel, Professeur à l'ÉTS et vice-président scientifique du CRIM, pour son encadrement, sa disponibilité et son soutien pendant tout le long de ce projet. Je n'aurais vraiment pas pu terminer mes études et mes travaux de recherche sans ses aides.

Je voudrais également remercier Madame Narjes Boufaden, post-doctorante à l'ÉTS et chercheur au CRIM, pour ses conseils et ses recommandations productifs.

J'aimerais aussi remercier tous les membres du Jury pour leur évaluation du projet.

Finalement, j'aimerais remercier ma famille, mes amis au Vietnam et à Montréal qui m'ont grandement encouragé durant ces deux dernières années.

# DÉVELOPPEMENT D'UN SYSTÈME D'IDENTIFICATION DES DIALOGUES PROLÉMATIQUES DANS LE SYSTÈME DE DIALOGUE PERSONNE-MACHINE

Truong, Le Hoang

## RÉSUMÉ

Dans ce mémoire, nous proposons un outil de classification automatique de dialogues problématiques dans un contexte d'un système de dialogue personne-machine. Le domaine d'application de cet outil est celui du forage de données (data mining), un sous domaine du domaine de l'apprentissage machine (machine learning). L'architecture de cet outil est modulaire et extensible afin de faciliter l'expérimentation de différents paradigmes de classification. L'outil utilise plusieurs schèmes d'apprentissage machine tels que l'arbre de décision C4.5 et l'arbre de modélisation logistique pour la classification de dialogue et les paramètres utilisés proviennent de la plateforme PARADISE. De plus, nous étudions l'ajout de deux nouveaux paramètres : mots négatives de reconnaissance et répétitions de mots. L'outil est testé selon la technique de validation croisée avec 10 validations croisées sur deux corpus publiquement distribués par le Linguistic Data Consortium (DARPA Communicator 2000 et DARPA Communicator 2001). Les résultats obtenus comparés à ceux-là de l'état de l'art montrent que notre PDI est plus performant et que les deux nouveaux paramètres améliorent la performance globale de l'outil.

**Mots-clés**: dialogue problématique, identificateur de dialogues problématiques, système de dialogue personne-machine, forage de données, machine d'apprentissage, classification de dialogue.

# DEVELOPMENT OF A DIALOG CLASSIFICATION SYSTEM IDENTIFYING PROBLEMATIC DIALOGS IN HUMAN-COMPUTER DIALOG SYSTEM

Truong, Le Hoang

## ABSTRACT

In this thesis, we develop a dialog classification tool containing a Problematic Dialog Identifier (PDI) that helps automate the task of identifying problematic dialogs in a context of a Human-Computer Dialog System (HCDS). This automatic tool is a practical Data Mining application in Machine Learning domain. It is modular and easily extensible. It uses several popular, widely used learning schemes such as C4.5 Tree, Logistic Model Tree for dialog classification. We also study the effect of two new potentially good features, namely negative acknowledgement words and system repetitions, on the performance of PDI. The PDI is tested with 10-fold stratified cross-validation on two publicly distributed corpora DARPA Communicator 2000 and 2001. The obtained results when compared with those of state-of-the-art show that our PDI outperforms and those two features are really good.

**Keywords**: problematic dialog, problematic dialog identifier, human-computer dialog system, data mining, machine learning, dialog classification.

**TABLE DES MATIÈRES**

# LISTE DES TABLEAUX

# LISTE DES FIGURES

# LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

| | |
|---|---|
| HCDS | Human-Computer Dialog System |
| DCS | Dialog Classification System |
| ÉTS | École de technologie supérieure |
| CRIM | Centre de recherche informatique de Montréal |
| PDI | Problematic Dialog Identifier |
| HMIHY | How May I Help You |
| ASR | Automatic Speech Recognition |
| NLU | Natural Language Understanding |
| DM | Dialog Manager |
| DARPA | Defense Advanced Research Projects Agency |
| LSE | Least Square Estimation |
| RSS | Residual Sum of Squares |
| CART | Classification And Regression Tree |
| PARADISE | PARAdigm for DIalog System Evaluation |
| DATE | Dialogue Act Tagging Scheme for Evaluation of HCDS |
| WEKA | Waikato Environment for Knowledge Analysis |
| ERD | Entity Relationship Diagram |
| PK | Primary Key |
| FK | Foreign Key |

# INTRODUCTION

In the context of a call center, spoken dialogue system is offered for different kinds of services to users via telephony. It provides efficient and natural access to information services from any phones or Internet and allows a cost reduction of service operations. Nowadays, its widely used applications such as email, travel planning information, and customer care have moved from research labs into commercial use.

Spoken dialog system is a general term referring to two kinds of dialog systems: Human-Human Dialog System and Human-Computer Dialog System (HCDS). This thesis only focuses on the evaluation of HCDS.

A human-computer dialog is a conversation between a user and an agent (sometimes we may also call a system). A problematic dialog is a dialog in which user is unsatisfied. For example, a user might be unsatisfied because he/she has to repeat the same utterance many times in a row.

To determine a problematic dialog, user satisfaction rating (shortly, user rating) is used. After the dialog is completed, user is asked some questions to assess the agent performance. Different agents may have different set of questions. In this thesis, we use a set of questions defined in DARPA Communicator corpora [18][19]. Those questions are the followings:

- <u>Task Success</u>: Is user's task completed successfully? (Yes / No)
- <u>Task Ease – (A)</u>: In this conversation, it was easy to get the information that user wanted?
- <u>TTSPerf (Text To Speech Performance) – (B)</u>: In this conversation, user found it easy to understand what the system said?
- <u>User Expertise – (C)</u>: In this conversation, user knew what to say or to do at each point in the dialogue?

- Expected Behavior – (D): In this conversation, the system worked the way user expected it to?
- Future Use – (E): In this conversation, based on user's experience using this system to get travel information, user would like to use this system regularly?

For the last five questions, user gives points for each question. The answers to the questions A/ B/ C/ D/ E have value varying from 1 to 5 based on Likert-scale that is a multi-item scale [20]. Likert-scale format is presented in the subsequent section.

By summing up the values of five answers to those questions above, we have the actual user satisfaction score, named UserRating, used to define problematic dialogs:

$$\text{UserRating} < \text{Threshold} \rightarrow \text{Bad dialog}$$

Dialog examples and user's assessments are presented in Chapter 3.

Dialog Classification System (DCS) is a software tool that identifies problematic dialogs (or bad dialogs) from a set of dialogs collected in HCDS to propose new dialog strategies for agent and to provide bad dialogs for the Emotion Detection System as well. In fact, DCS is the first part of "Managing Emotions in Human-Computer Dialogs" project developed by ÉTS (École de technologie supérieure) & CRIM (Centre de recherche informatique de Montréal) in collaboration with Bell Canada Corp. Developing DCS is really essential because it helps automate the task of identifying problematic dialogs that is sometimes overwhelmed for human to accomplish.

An experimentation framework, named Basili's framework [1][2][3], is employed to structure the thesis organization. According to this framework, Chapter 1 gives some definitions about the project; Chapter 2 describes the project planning. Then, Chapter 3 demonstrates the implementation, and Chapter 4 gives the interpretation of the obtained results. Finally, the last section presents the conclusion of the work.

# CHAPITRE 1

# DEFINITION

## 1.1 Motivation & Purpose

The "Managing emotions in Human-Computer Dialogs" project is a practical data mining application of ÉTS & CRIM in collaboration with Bell Canada Corp. It originates from the desire to evaluate user satisfaction in human-computer dialogs in the settings of a call center of Bell Canada Corp. with the purpose of proposing new dialog strategies for HCDS. This results from the finding that the performance of the system based on speech recognition is not perfect. Sometimes clients are upset about the facts that they are not understood and they are ready to withdraw their association with Bell Canada if nobody helps them to solve their problem. Therefore, identifying problematic dialogs is a relatively essential need. However, Bell Canada Corp. records a great number of dialogs everyday. This number will be increased enormously day after day, so the task of identifying problematic dialogs may become overwhelmed for human to costly accomplish. Thus, DCS is a really essential automatic tool to help automate such task.

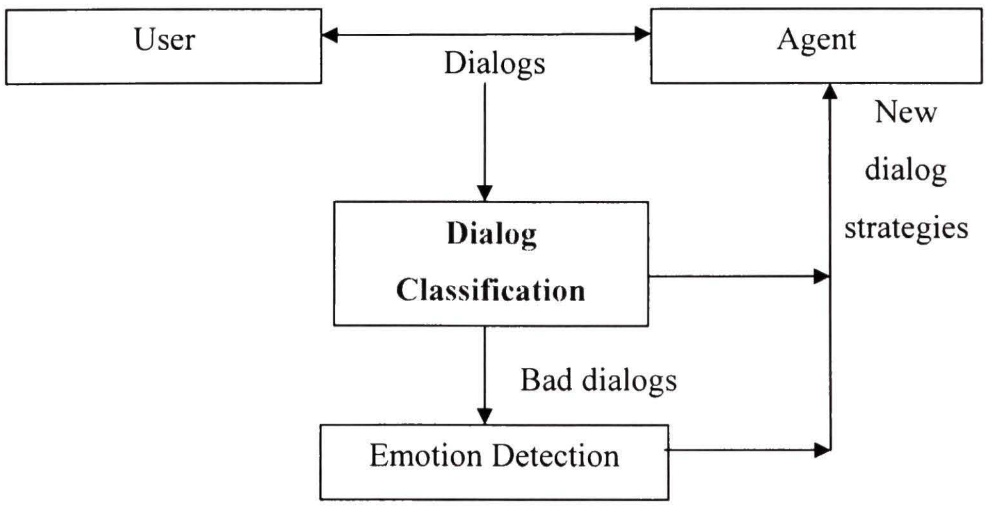


**Figure 1-1** ***Human-Computer Dialog System.***

The project diagram is illustrated in Figure 1.1. We develop DCS in the first stage of the project. The main goal of DCS is to identify problematic dialogs where user is unsatisfied. These bad dialogs are dispatched to the Emotion Detection System – developed by other team and out-of-scope of this research work – to detect user's emotions.

DCS belongs to data mining and machine learning domain (more particularly, pattern classification), so we will take a look at this domain in the literature review section.

## 1.2 Literature Review

### 1.2.1 Data Mining & Machine Learning

Data mining is the extraction of implicit, previously unknown, and potentially useful information from data. Data mining is defined as the process of discovering patterns in data. The process is preferably fully automatic, but it is often semi-automatic due to performance. The patterns discovered must be meaningful in that they lead to some advantage, usually an economic advantage [4].

Machine learning provides the technical basis of data mining. Machine learning is concerned with the design and development of algorithms and techniques that allow computers to "learn". Machine learning has a large number of applications including natural language processing, speech and speaker recognition, pattern classification, to name a few [4].

Now, we consider a simple pattern classification example to know what pattern classification is and how it is applied in practice.

### 1.2.2 A simple pattern classification example [6]

Pattern classification is the act of taking in raw data and making an action based on the category of the pattern. Pattern classification takes decisions based on appropriate probabilistic or non-probabilistic models of the patterns.

It is essential to know several terminologies used in pattern classification. Those are the followings:

- Pattern: a pattern can be an object, a process or an event consisting of both deterministic and stochastic components; a record of dynamic occurrences influenced by both deterministic and stochastic factors. Textures, crystals, weather pattern, speech waveform, dialog pattern are some examples.

- Feature: a feature (also called attribute) is a relevant, intrinsic trait or characteristic that makes a pattern apart from another; data extractable through measurement and/or processing, such as color, age, weight, and aspect ratio.
  There are two main kinds of features: nominal feature (e.g. Sunny, Rainy…) and numeric feature (e.g. 45s, 78°C).

- Pattern class: a pattern class is a set of patterns sharing a set of common features and usually originating from the same source (associated with the generalization or abstraction of patterns).

- Classification: classification is the act of assigning patterns into pattern classes based on their features.

- Noise: noise is a distortion associated with pattern processing (errors in feature extraction) and/or training samples that impact the classification abilities of the system.

Let us consider a simple example of pattern classification as shown in [6]: classify two types of fish (salmon and sea bass). There are some physical differences between salmon and sea bass such as length, width, lightness.

Given that there are differences between the population of salmon and sea bass, we view them as having different models used for feature extraction. For example, somebody tells us that a sea bass is generally longer than a salmon. This gives us a model for the fish: sea bass length is greater than that of salmon → length becomes an obvious feature.

To make classification more accurate, we have to use many features. Suppose we have two features for classifying fish: the length and the lightness. From training samples, we measure their two features and plot a graph as shown in Figure 1.2.

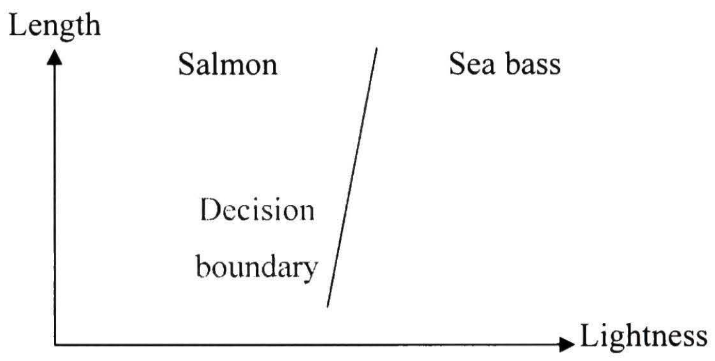


**Figure 1-2** ***Sample space of fish.***

The plot suggests classifying the fish as sea bass if its feature vector falls at the right of the decision boundary and as salmon otherwise. In this example, the decision boundary is a straight line. However, depending on the distribution of the samples, decision boundary could be a curve or something else. It is therefore necessary to choose features carefully to achieve good representation that enables successful pattern classification. This selection could be complicated by noise and errors. Robust features are the ones relatively insensitive to noise and other errors.

Pattern classification is only one of four machine learning styles in data mining. How many machine learning styles are there in data mining? We will describe these styles in the next section.

### 1.2.3 Machine learning styles [4]

Generally, machine learning styles in data mining include:

- Numeric prediction: predicts a target value based on a vector of features.
- Pattern classification: learns a way of classifying unseen patterns into discrete classes from a set of labeled examples.
- Association learning: any association among features is sought, not just ones that predict a particular class value.
- Clustering: seeks groups of samples that belong together.

In these four styles, numeric prediction and pattern classification styles are used more widely than the other two in data mining applications. Moreover, DCS uses pattern classification style, so we only focus on that style along with numeric prediction style because the former is often derived from the latter.

The general problem of numeric prediction and pattern classification can be described in turn in the two sub-sections below.

#### 1.2.3.1 Numeric prediction

Given a column feature vector $X = (X_1, X_2, \ldots, X_d)^T$,
where $X_1, X_2, \ldots, X_d$ are d features of pattern X and d is the number of dimensions of the feature vector X.

The problem is to predict the numeric value $Y = f(X)$, where $f(X)$ is a function with respect to X. For example: $f(X) = X_1 + 2X_2$ or $f(X) = 3X_1 - X_2$.

The numeric prediction system is trained by an n-element dataset, each element has the form (x, y) with x is a d-dimensional vector and y is a numeric value:

$$x_1 = (x_{11}, x_{12}, \dots, x_{1d}), y_1$$
$$x_2 = (x_{21}, x_{22}, \dots, x_{2d}), y_2$$
$$\dots$$
$$x_n = (x_{n1}, x_{n2}, \dots, x_{nd}), y_n$$

where:

$x_1$: 1st observation of feature vector X; $y_1$: corresponding numeric value

$x_2$: 2nd observation of feature vector X; $y_2$: corresponding numeric value

...

$x_n$: nth observation of feature vector X; $y_n$: corresponding numeric value

### 1.2.3.2 Pattern classification

Given a column feature vector $X = (X_1, X_2, \dots, X_d)^T$,
where $X_1, X_2, \dots, X_d$ are d features of pattern X and d is the number of dimensions of the feature vector X.

The problem is to classify X into one of the k classes $\{C_1, C_2, \dots, C_k\}$ .

The pattern classification system is trained by an n-element dataset, each element has the form (x, c) with x is a d-dimensional vector and c is a class:

$$x_1 = (x_{11}, x_{12}, \dots, x_{1d}), c_1$$
$$x_2 = (x_{21}, x_{22}, \dots, x_{2d}), c_2$$
$$\dots$$
$$x_n = (x_{n1}, x_{n2}, \dots, x_{nd}), c_n$$

where:

$x_1$: 1st observation of feature vector X; $c_1$: corresponding class

$x_2$: 2nd observation of feature vector X; $c_2$: corresponding class

…

$x_n$: nth observation of feature vector X; $c_n$: corresponding class

To solve numeric prediction and pattern classification problems simply, we can use one of the three basic learning models which are the ones applied directly on the training dataset.

- Zero-Rule Model
- Linear Regression Model
- Logistic Regression Model

In the next section, we will show how each basic learning model works.

### 1.2.4 Basic learning models

#### 1.2.4.1 Zero-Rule Model [4]

Given a training dataset:

$$x_1 = (x_{11}, x_{12}, \dots, x_{1d}),\ y_1/c_1$$
$$x_2 = (x_{21}, x_{22}, \dots, x_{2d}),\ y_2/c_2$$
$$\dots$$
$$x_n = (x_{n1}, x_{n2}, \dots, x_{nd}),\ y_n/c_n$$

In numeric prediction, Zero-Rule model predicts the average class value in the training data using the following rule: $\hat{y} = \text{average}(y_i)$.

While in pattern classification, Zero-Rule model predicts the majority class in the training data using rule: $\hat{c} = \text{argmax}(\Pr(C_i))$ , where $\Pr(C_i)$ is probability of class $C_i$ in the set of classes $\{C_1, C_2, \dots C_k\}$.

### 1.2.4.2 Linear Regression Model [5][11]

Given a training dataset:

$x_1 = (x_{11}, x_{12},\dots, x_{1d}), y_1/c_1$

$x_2 = (x_{21}, x_{22},\dots, x_{2d}), y_2/c_2$

…

$x_n = (x_{n1},x_{n2},\dots, x_{nd}), y_n/c_n$

In numeric prediction, Linear Regression Model uses a linear function f(x), $x=(x_1,\dots,x_d)^T$ to model the target value y:

$$y = f(x) = b_0 + b^T x = \sum_{i=0}^{d} b_i x_i \tag{1.1}$$

where $b_0$ is an intercept; $b=(b_1,b_2,\dots,b_d)^T$ is a coefficient column vector; and $x_0 = 1$

The problem is to estimate the intercept $b_0$ and vector b using LSE (Least Square Estimation) from the training dataset. LSE tries to minimize the Residual Sum of Squares:

$$RSS = \sum_{i=1}^{n} [y_i - f(x_i)]^2 \tag{1.2}$$

By minimizing the RSS in the equation (1.2), we obtain the values of the intercept $b_0$ and vector b. For new pattern $X=(X_1,X_2,\dots,X_d)^T$, the target value Y is predicted using linear model:

$$Y = b_0 + b_1 X_1 + \dots + b_d X_d \tag{1.3}$$

Linear Regression Model can also be applied to pattern classification by using the following Multi-response Linear Regression procedure:

- Perform k separate linear regression on each class, set the target value $y_i$ to 1 if the instance $x_i$ is in the class that we are doing linear regression and 0 otherwise.
- The regression equation then approximates the membership function for the class (1 for members, 0 for non-members).
- To classify a new instance X, compute the regression value for each membership function $Y_i$, $i = 1 \ldots k$, and assign the new instance the class with the highest value $\hat{Y} = \max(Y_i)$.

### 1.2.4.3 Logistic Regression Model [5]

The Logistic Regression Model is a better alternative solution of Linear Regression Model for classification. It arises from the desire to model the posterior probabilities of the k classes via linear function with respect to x. It is mandatory that those posterior probabilities sum to one and remain in the interval [0, 1].

The model has the form:

$$\log \frac{\Pr(C = C_1 \mid X)}{\Pr(C = C_k \mid X)} = b_{10} + b_1^T * X$$

$$\log \frac{\Pr(C = C_2 \mid X)}{\Pr(C = C_k \mid X)} = b_{20} + b_2^T * X \qquad (1.4)$$

$$\vdots$$

$$\log \frac{\Pr(C = C_{k-1} \mid X)}{\Pr(C = C_k \mid X)} = b_{(k-1)0} + b_{k-1}^T * X$$

where:

$X=(X_1,X_2,\ldots,X_d)^T$: input pattern

$\Pr(C=C_1 \mid X)$: posterior probability of class $C_1$ given X

$\Pr(C=C_2 \mid X)$: posterior probability of class $C_2$ given X

. . .

$\Pr(C=C_k \mid X)$: posterior probability of class $C_k$ given X

$b_{i0}$: intercept

$b_i = (b_1, b_2,\ldots,b_d)^T$: coefficient column vector

Solving the equation system (1.4), we obtain the posterior probabilities of the k classes:

$$\Pr(C = C_i \mid X) = \frac{\exp(b_{i0} + b_i^T * X)}{1 + \sum_{j=1}^{k-1} \exp(b_{j0} + b_j^T * X)}, i = 1 \ldots k - 1$$
$$\Pr(C = C_k \mid X) = \frac{1}{1 + \sum_{j=1}^{k-1} \exp(b_{j0} + b_j^T * X)} \quad (1.5)$$

The result (1.5) shows that those posterior probabilities sum to one and remain in [0, 1]. Now, the problem is to estimate the intercepts and coefficient column vectors. These parameters are usually estimated by using Maximum Likelihood method. This method is mathematically complicated, so we do not present it here.

When k = 2, the model is especially simple because there is only a single linear function:

$$\log \frac{\Pr(C = C_1 \mid X)}{\Pr(C = C_2 \mid X)} = b_0 + b^T * X \quad (1.6)$$

This model is widely used in cases where binary responses (two classes) occur quite frequently, especially in biostatistics' applications. DCS is also a two-class dialog classification system, so it is possibly appropriate to apply Logistic Regression Model with k=2 in DCS.

### 1.2.5 Tree-based learning schemes

In this section, we describe a popular and widely used learning scheme. That is tree-based learning scheme or also known as decision tree which employs three basic learning models presented above.

Tree-based learning scheme or decision tree is a predictive model mapping observations about a pattern to conclusions about its target value. In decision tree, each interior node corresponds to a variable; an arc to a child represents a possible value of that variable. A leaf represents the predicted value of target variable given the values of the variables represented by the path from the root [12]. Figure 1.4 shows an example of decision tree.

Decision tree has three names:

- **Classification tree:** is a term used when the predicted outcome is a categorical class (for example: Play, Don't Play, Good, Bad...)
- **Regression tree:** is a term used when the predicted outcome is a numeric value (for example: CPU time, a patient's length of stay in a hospital...).
- **CART (Classification And Regression Tree)**: is a term used to refer to both of the above trees. It was first introduced by Breiman et al. [10]

To know more details about decision tree, let us consider a practical example about a golf club [12]:

*A manager of a golf club wants to predict customer attendance in his club in order to know if he should hire extra staffs on days when customers play golf or dismiss most of them on days when customers do not play golf.*

The manager observed and measured four features in two weeks:

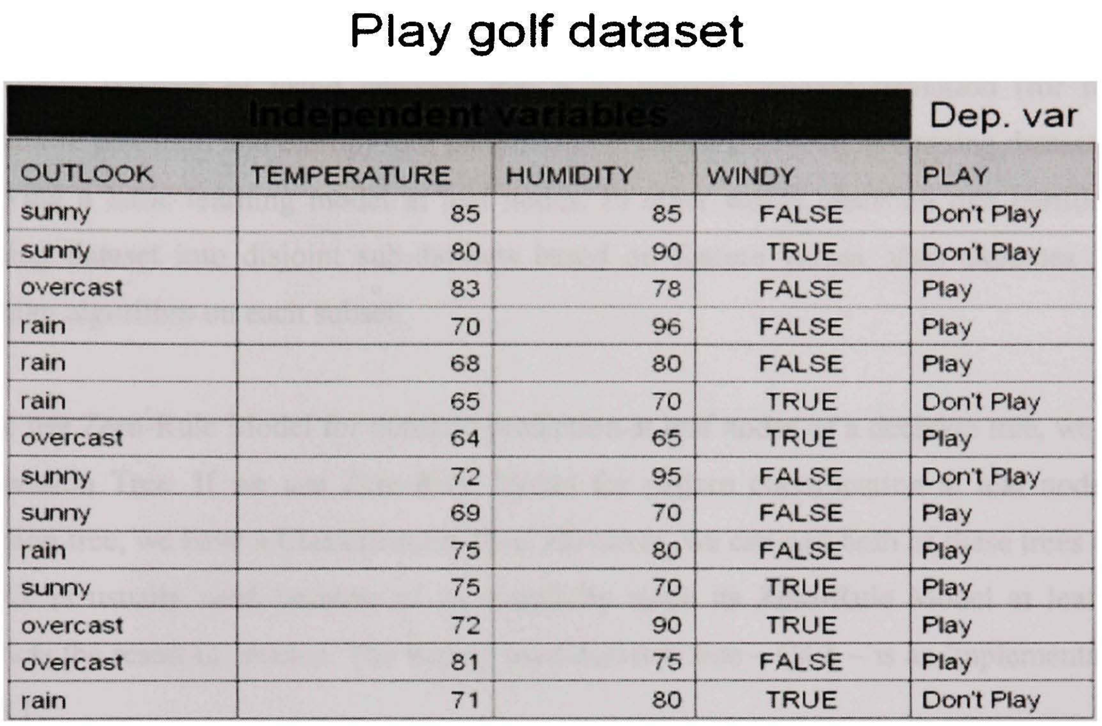
1) Outlook: a nominal feature that has value belonging to a set of possible values (sunny, forecast, rain);
2) Temperature: a numeric feature. The unit of measurement is $^{o}C$;
3) Humidity: a numeric feature. The unit of measurement is %;
4) Wind: a nominal feature that has value belonging to a set of possible values (true, false).

He wants to know if the customers will play or not in a given day, so the target value is a category class belonging to a set of classes: {Play, Don't play}. Then, he made a dataset as shown in Figure 1.3. The decision tree built from this dataset is displayed in Figure 1.4.

According to this decision tree, he can conclude that:

- Users play golf on sunny and non-humid days / on overcast days / or on rainy and non-windy days, then he will hire extra staffs on these days.
- Users don't play golf on sunny and humid days, or on rainy and windy days, then he will dismiss most of the staff on these days.

In this example, the manager has constructed a decision tree using Zero-Rule model at leaf nodes. By using different models at leaf nodes, we have different tree-based learning schemes. There are three popular tree-based learning schemes: CART, Model Tree and Logistic Model Tree. We will describe briefly in turn these learning schemes in the next section.



Play golf dataset

| Independent variables | | | | Dep. var |
|---|---|---|---|---|
| OUTLOOK | TEMPERATURE | HUMIDITY | WINDY | PLAY |
| sunny | 85 | 85 | FALSE | Don't Play |
| sunny | 80 | 90 | TRUE | Don't Play |
| overcast | 83 | 78 | FALSE | Play |
| rain | 70 | 96 | FALSE | Play |
| rain | 68 | 80 | FALSE | Play |
| rain | 65 | 70 | TRUE | Don't Play |
| overcast | 64 | 65 | TRUE | Play |
| sunny | 72 | 95 | FALSE | Don't Play |
| sunny | 69 | 70 | FALSE | Play |
| rain | 75 | 80 | FALSE | Play |
| sunny | 75 | 70 | TRUE | Play |
| overcast | 72 | 90 | TRUE | Play |
| overcast | 81 | 75 | FALSE | Play |
| rain | 71 | 80 | TRUE | Don't Play |

**Figure 1-3** ***Play golf dataset.***

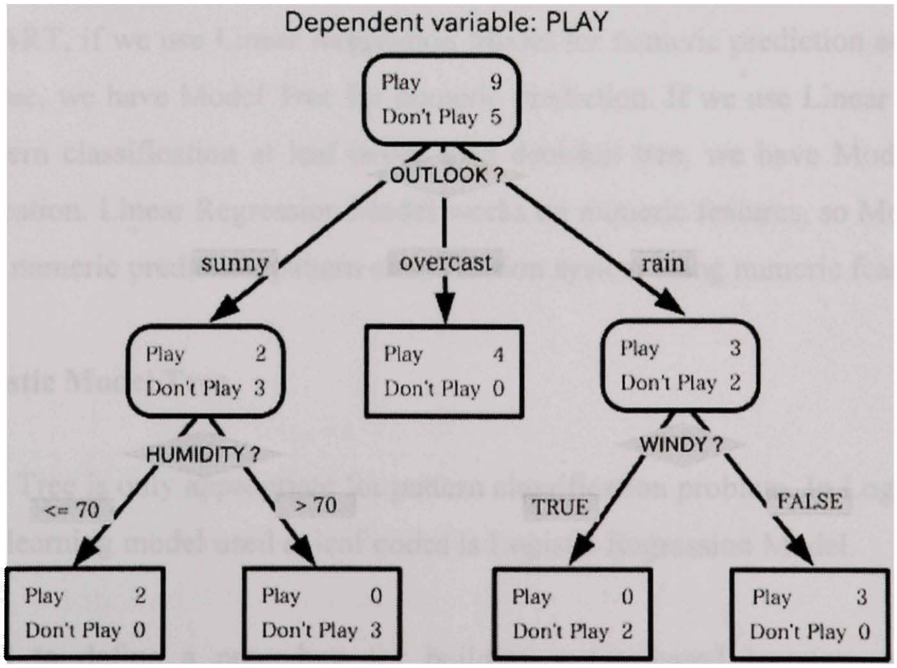


**Figure 1-4** ***Decision tree for play golf dataset.***

#### 1.2.5.1 CART – C4.5

The main purpose of using decision tree is to reduce standard deviation (for numeric prediction problem) and entropy (for pattern classification problem) in training dataset before applying a basic learning model at leaf nodes. In other words, decision tree partitions the training dataset into disjoint sub-datasets based on feature values, then executes a basic learning algorithm on each subset.

If we use Zero-Rule Model for numeric prediction at leaf nodes of a decision tree, we have a Regression Tree. If we use Zero-Rule Model for pattern classification at leaf nodes of a decision tree, we have a Classification Tree. However, we can call both of these trees CART. CART is usually used because of its simplicity since its Zero-Rule Model at leaf nodes predicts the result by chance. The widely used decision tree – C4.5 – is an implementation of CART.

#### 1.2.5.2 Model Tree

Similarly to CART, if we use Linear Regression Model for numeric prediction at leaf nodes of a decision tree, we have Model Tree for numeric prediction. If we use Linear Regression Model for pattern classification at leaf nodes of a decision tree, we have Model Tree for pattern classification. Linear Regression Model works on numeric features, so Model Tree is appropriate for numeric prediction/pattern classification system using numeric features.

#### 1.2.5.3 Logistic Model Tree

Logistic Model Tree is only appropriate for pattern classification problem. In Logistic Model Tree, the basic learning model used at leaf nodes is Logistic Regression Model.

Now, we need to define a procedure for building a tree-based learning scheme. For visualization convenience, the procedure is described by a flow chart as shown in Figure 1.5. We detail each step of the procedure next section.

### 1.2.6 Procedure for building a tree-based learning scheme [6][7][8][9]

The procedure for constructing a tree-based learning scheme consists of the following steps:

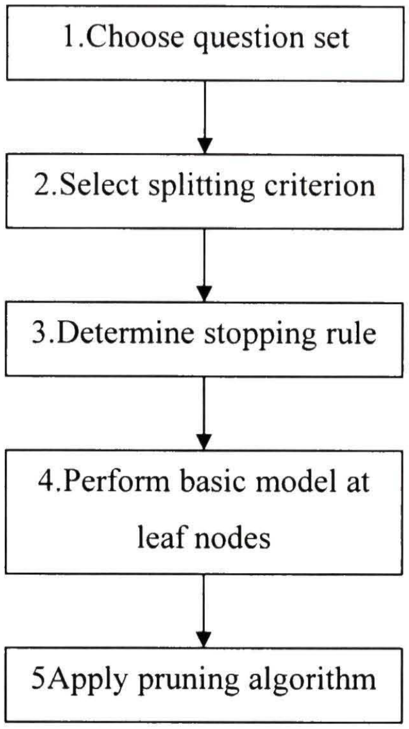

1.Choose question set
2.Select splitting criterion
3.Determine stopping rule
4.Perform basic model at leaf nodes
5Apply pruning algorithm

**Figure 1-5** ***Flow chart for tree building procedure.***

1. <u>Choose a question set used for non-terminal node splitting</u>:

Given a training dataset:

$$x_1 = (x_{11}, x_{12}, \ldots, x_{1d}), y_1/c_1$$
$$x_2 = (x_{21}, x_{22}, \ldots, x_{2d}), y_2/c_2$$
$$\ldots$$
$$x_n = (x_{n1}, x_{n2}, \ldots, x_{nd}), y_n/c_n ,$$

Denote a pattern vector $X = (X_1, X_2, \ldots, X_d)^T$, where: $X_i$ is the ith feature of X

- If feature $X_i$ is a nominal feature, the question has the following form:

$$\text{Is } X_i \in S?$$

where:

S is a subset of a set of possible discrete values of $X_i$. Given **S** is a subset of a k-element set $\{a_1,\dots, a_k\}$, the number of questions will be $(2^k - 2)$.

- If feature $X_i$ is a numeric feature, the question is:

Is $X_i <= C$?

where:

C**:** a threshold value. The number of questions will be (**n**) with:

$$C_j = \frac{V_{(j-1)} + V_j}{2}, j = 1 \dots n, V(0) = 0, V(j) = X_i \tag{1.7}$$

2. Select a splitting criterion that determines which question will be used for splitting:

Assume that we are solving pattern classification problem, the most widely used splitting criterion for a non-terminal node is the Information Gain also called Impurity Reduction or Entropy Reduction. We focus on pattern classification problem because DCS turns out to be a pattern classification system. In numeric prediction case, the splitting criterion is Standard Deviation Reduction.

Suppose we have a binary split that separates a node t into two nodes: left node $t_L$ and right node $t_R$.
The information gain IG is calculated by the following formula:

$$IG = I(t) - [p_L I(t_L) + p_R I(t_R)] \tag{1.8}$$

where:

$t$, $t_L$, $t_R$ : current node, left node, and right node
$p_L$: proportion between samples falling into left node and samples in node t
$p_R$: proportion between samples falling into right node and samples in node t

I(t), I($t_L$), I($t_R$) are the Entropy Impurities of nodes t, $t_L$, $t_R$ and are computed as follow

$$I(t) = -\sum_{i=1}^{d} \Pr(C_i)\log(\Pr(C_i)) \tag{1.9}$$

where $C_i$ is the ith class; and Pr($C_i$) is probability of class $C_i$ .

We deduce a rule for selecting the best question as follow:

*The best question chosen for splitting non-terminal node is the one that maximizes IG in the equation (1.8).*

To demonstrate how to calculate Information Gain, we consider a 3-class problem, given a node t with 10 training vectors:

- 4 vectors belong to class $C_1$;
- 4 vectors belong to class $C_2$;
- 2 vectors belong to class $C_3$.

Suppose node t(4, 4, 2) is split into two nodes: $t_L(3,1,0)$ & $t_R(1,3,2)$. The goal is to compute IG of node t using entropy impurity.

According to the equation (1.9), we have the entropy impurity of each node:

$I(t) = -(4/10)*\log(4/10) - (4/10)*\log(4/10) - (2/10)*\log(2/10) = 1.521$

$I(t_L) = -(3/4)*\log(3/4) - (1/4)*\log(1/4) = 0.815$

$I(t_R) = -(1/6)*\log(1/6) - (3/6)*\log(3/6) - (2/6)*\log(2/6) = 1.472$

Then, applying the equation (1.8), we have the information gain for node t:

$IG = 1.521 - (4/10)*0.815 - (6/10)*1.472 = 0.315$

3. Determine a split-stopping rule for leaf nodes:

We can combine the following conditions to determine the split-stopping rule:

- Zero impurity: all the data samples at leaf node belong to the same class.
- Feature: there's no feature left to split.
- Threshold:
  + The greatest information gain of best question falls below a pre-set threshold $\beta$
  + The number of training samples is small enough.
  + The tree is pretty big.

4. Perform an appropriate basic learning model at leaf nodes:

Zero-Rule Model is used for CART, Linear Regression Model for Model Tree and Logistic Regression Model for Logistic Model Tree.

Once the question set, splitting criterion, split-stopping rule and basic learning model were determined, a greedy algorithm used to build the decision tree is as follow:

- All training samples are placed at the root of the initial tree.
- Create a question set from these training samples.
- The best question is then chosen from the question set to split the root into two nodes.
- The algorithm recursively splits the most promising node with the best question until the stopping rule is satisfied.
- Basic learning model is executed at leaf nodes.

Table 1.1 shows the pseudo-code to build a tree-based learning scheme. After we have built the tree, the algorithm for classifying new data is given in Table 1.2.

Table 1-1 Pseudo-code for building a tree-based learning scheme

```
Given a training set, each pattern $X = (X_1,\ldots,X_d)^T$ is a d-dimensional feature vector
1. Begin with the root node with all training samples.
2. For each new node *t*
_For every nominal feature: generate all its subset and obtain questions in the form
                                Is $X_i \in S$?
_For every numeric feature: calculate $C_j$ and generate all questions of the form
                                Is $Xi <= C_j$?
_For each question Q from the question set above:
        + create $X_{tL}$ (number of samples falling into left node of t) and $X_{tR}$ (number of
        samples falling into right node of t) according to the answer of the question Q.
        + compute Information Gain of node t
_Choose the question $Q_0$ that maximizes Information Gain
_If stopping rule is met, execute appropriate basic learning model;
 else create left node $t_L$ and right node $t_R$ of t based on the answer of question Q0.
```

Table 1-2 Algorithm for classifying new pattern

```
Given a decision tree and an input feature vector of a dialog pattern X
1. Begin at root node.
2. At node t
        _If t is a leaf node, get the target outcome.
        _If t is an interior node, use feature of X to find the answer of the best question of
node t:
                + If the answer is YES, traverse to left node of t
                + If the answer is NO, traverse to right node of t
```

5. Apply a pruning algorithm to obtain an optimal tree:

After building the decision tree, we can apply a pruning algorithm to get an optimal tree. Pruning algorithm is slightly complex. We describe it in a separate part.

### 1.2.7 Pruning algorithm [7]

Given a decision tree, the problem is to prune this tree to obtain a compact and optimal tree. The pruning criterion used to prune the original tree is the cost-complexity measure:

$$R_\alpha(T) = R(T) + \alpha |\bar{T}| \tag{1.10}$$

where:

R(T): misclassification rate of tree T (cost) is computed by the following formula:

$$R(T) = \sum_{t \in \bar{T}} P(t)(1 - \max(\Pr(C_i \mid t)) \tag{1.11}$$

where:

P(t): proportion between data falling in node t and data in root.

$\Pr(C_i \mid t)$: probability of class $C_i$ at node t.

α: complexity parameter

$|\bar{T}|$: number of leaf nodes of tree T (complexity)

Suppose we have a tree like the one in Figure 1.6.

For convenience, we denote $T_t$ as sub-tree starting at node t and {t} as sub-tree containing only node t.

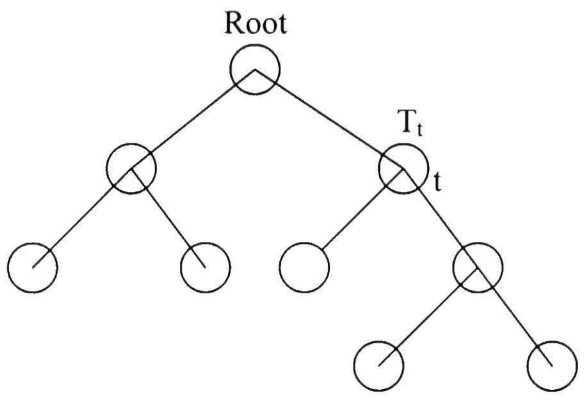


**Figure 1-6** ***Tree example for pruning algorithm.***

Consider node $T_t$ in Figure 1.6, applying the equation (1.10), we have:

$$R_\alpha(T_t) = R(T_t) + \alpha |\overline{T}_t| \quad (1.12)$$

$$R_\alpha(\{t\}) = R(t) + \alpha \quad (1.13)$$

To perform pruning algorithm, we have to find weakest sub-tree that is a tree when we collapse it into a single node, the misclassification rate increases least or is unchanged. Therefore, $T_t$ is considered as a weakest sub-tree if

$$R_\alpha(T_t) \approx R_\alpha(\{t\}) \quad (1.14)$$

Then, substituting the equations (1.12) and (1.13) to the left-hand side and right-hand side of the equation (1.4) respectively, we obtain the formula of the complexity parameter α:

$$\alpha = \frac{R(t) - R(T_t)}{|\overline{T}| - 1} \quad (1.15)$$

Let $\alpha = g(t)$, where t is a non-terminal node and g(t) is a function with respect to t. The cost-complexity pruning procedure is described as follow:

- Start at tree T0 with root r
- Find the sub-tree $T_t$ that minimizes g(t)
- Collapse the sub-tree $T_t$ to node t to obtain a pruned tree $T1 = T0 - T_t$
- Find a pruned tree T2 from T1 using the same way
- Continue the procedure until we obtain the tree containing only root node: {r}

The result of the pruning procedure is a sequence of trees:

$$T0 > T1 > T2 > \dots > \{r\}$$

From the sequence of trees above, we will choose an optimal tree by using one of two methods: Independent Test Set or N-fold Cross Validation.

#### 1.2.7.1 Independent test set method

For this method, the training dataset is divided to two parts as shown in Figure 1.7. We use the first part (about 80% of the training dataset) to create a sequence of trees:

$$T0 > T1 > T2 > \dots > \{r\}$$

Then, we use the other part to estimate the misclassification rate of each tree using the equation (1.11):

$$T0 \rightarrow R(T0)$$
$$T1 \rightarrow R(T1)$$
$$T2 \rightarrow R(T2)$$
$$\dots$$
$$\{r\} \rightarrow R\{r\}$$

The optimal tree Tk is the one that has the minimum misclassification rate:

$$k = \text{argmin}\ (R(Tk))$$

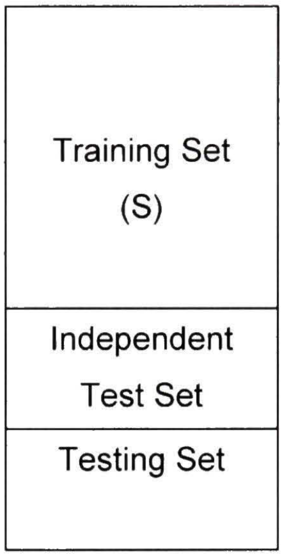


**Figure 1-7** ***Independent test set method.***

### 1.2.7.2 N-fold cross validation method

With the N-fold cross validation method, we use all training samples to create a sequence of trees:

$$T0 > T1 > T2 > \ldots > \{r\}$$

Next, we divide the training dataset into N folds. Each fold contains two parts: training part and independent test part. We use the training part to create a sequence of trees and use the independent test part to estimate misclassification rate for each tree. Figure 1.8 illustrates how we divide the dataset. In summary, we have:

Fold 1 : $T^10 > T^11 > T^12 > \ldots > \{r^1\}$

$R(T^10), R(T^11), R(T^12), \ldots, R\{r^1\}$

Fold 2 : $T^20 > T^21 > T^22 > \ldots > \{r^2\}$

$R(T^20), R(T^21), R(T^22), \ldots, R\{r^2\}$

...

Fold N : $T^N0 > T^N1 > T^N2 > \ldots > \{r^N\}$

$R(T^N0), R(T^N1), R(T^N2), \ldots, R\{r^N\}$

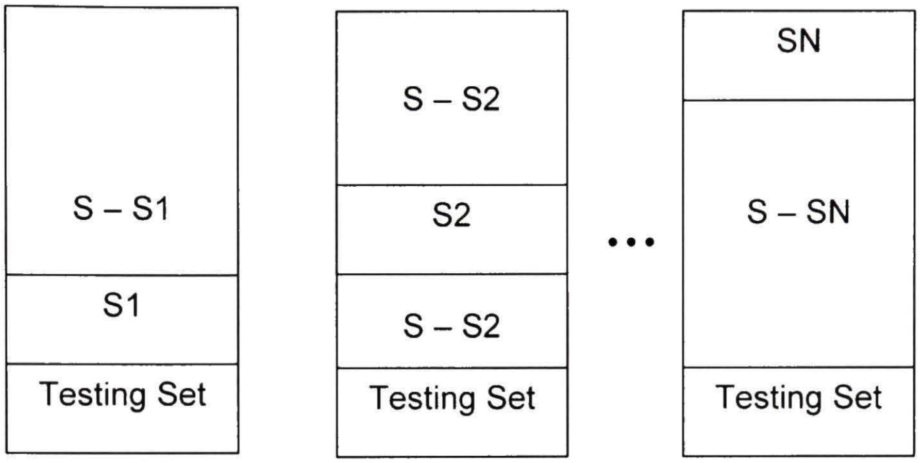


**Figure 1-8** ***N-fold cross validation method.***

We can not directly estimate the misclassification rates for the main sequence of trees, we could approximate them via the misclassification rate $R(T^{i}j)$, since each data sample in the main training dataset occurs in one and only one fold . The N-fold cross-validation estimate can be computed as:

$$R_{cv}(Tk) = \frac{1}{N}\sum_{i=1}^{N} R(T^{i}k) \tag{1.16}$$

So, we have:

$T0 \rightarrow R_{cv}(T0)$

$T1 \rightarrow R_{cv}(T1)$

$T2 \rightarrow R_{cv}(T2)$

...

$\{r\} \rightarrow R_{cv}\{r\}$

The optimal tree Tk is the one that has minimum misclassification rate:

$$k = \text{argmin}\,(R_{cv}(Tk))$$

N-fold cross-validation method is computationally expensive in comparison with the independent test set method. However, it makes more effective use of all training data, so this method is only useful when we have a small dataset. Otherwise, independent test set method is more appropriate.

### 1.2.8 Other learning schemes for pattern classification

In this section, we introduce two more learning schemes for pattern classification that we will use for our experiments. These are one-rule algorithm and boosted decision trees described in the next two subsections.

#### 1.2.8.1 One-rule algorithm [4]

One-rule algorithm for pattern classification generates a one-level decision tree expressed in the form a set of rules that all test one particular attribute, so it's a simple and computer cost-effective method. However, it frequently gives high accuracy in many real-world datasets because the structure underlying those datasets might be quite rudimentary, and just one attribute is sufficient to accurately determine the class of an instance.

The idea of one-rule algorithm comes from the fact that it's always a good plan to try the simplest things first. The pseudo-code for one-rule algorithm is given in Table 1.3.

Table 1-3 Pseudo-code for one-rule algorithm

```
For each attribute {
    For each value of that attribute {
        count how often each class appears;
        find the most frequent class;
        make the rule assign that class to this attribute-value;
    }
    Calculate the error rate of the rules;
}
Choose the attribute that produce rules with the smallest error rate;
```

Let's reconsider the play golf problem in section 1.2.5. Applying the one-rule algorithm on that dataset, we obtain the rules illustrated in Figure 1.9.

| | Attribute | Rules | Errors | Total errors |
|---|---|---|---|---|
| 1 | outlook | sunny → no | 2/5 | 4/14 |
| | | overcast → yes | 0/4 | |
| | | rainy → yes | 2/5 | |
| 2 | temperature | hot → no* | 2/4 | 5/14 |
| | | mild → yes | 2/6 | |
| | | cool → yes | 1/4 | |
| 3 | humidity | high → no | 3/7 | 4/14 |
| | | normal → yes | 1/7 | |
| 4 | windy | false → yes | 2/8 | 5/14 |
| | | true → no* | 3/6 | |

**Figure 1-9** ***One-rule algorithm result on play golf dataset***

From the rule set, we can see that the two attributes "outlook" and "humidity" give the smallest total error rate. Therefore, we may choose one of them. For example, if we choose the attribute "outlook", we have the following classification rules:

| | | |
|---|---|---|
| Outlook: | *sunny → Don't play* | |
| | *overcast → Play* | (One-level decision tree) |
| | *rainy → Don't play* | |

### 1.2.8.2 Boosted decision trees [4]

An approach to making pattern classification more reliable is to combine the outputs of different learning schemes of the same type such as decision trees. Several machine learning techniques do this by learning an ensemble of models and using them in combination. The most prominent technique among these is **boosting**.

The boosting technique is an iterative machine learning method performing pattern classification by mixing various outputs of different learning schemes into a single classifier using weighted vote. Weighting is used to give more influence to the more successful learning scheme.

Now we describe a widely used boosting method called AdaBoost.M1 included in the WEKA [17] machine learning library. The iterative procedure of AdaBoost.M1 method for model generation and pattern classification is as follow:

Model Generation:

- The algorithm begins by assigning equal weight to all instances in the training data.
- It then calls the learning algorithm to form a classifier for this data and reweighs each instance according to the classifier's output. The weight of correctly classified instances is decreased and that of misclassified ones is increased. This produces a set of "easy" instances with low weight and a set of "hard" ones with high weight.

- In the next iteration, and all subsequent ones, a classifier is built for the reweighed data, which consequently focuses on classifying the hard instances correctly. Then the instances' weights are increased or decreased according to the output of this new classifier. As a result, some hard instances might become even harder and easier ones might become even easier; on the other hand, other hard instances might become easier, and easier ones might become harder—all possibilities can occur in practice.

- After each iteration, the weights reflect how often the instances have been misclassified by the classifiers produced so far. By maintaining a measure of "hardness" with each instance, this procedure provides an elegant way of generating a series of learning schemes that complement one another.

- Whenever the error on the weighted training data is 0 or exceeds or equals 0.5, the boosting procedure deletes the current classifier and does not perform any more iteration.

Pattern Classification:

- The outputs of all previously generated classifiers are combined using a weighted vote. A classifier that performs well on the weighted training data from which it was built (*e* close to 0) should receive a high weight, and a classifier that performs badly (*e* close to 0.5) should receive a low one.
- To make a classification, the weights of all classifiers that vote for a particular class are summed up, and the class with the greatest total weight is chosen.

The pseudo-code for the procedure above is displayed in Table 1.4.

Table 1-4 Pseudo-code for boosted decision trees

```
Model Generation:
Assign equal weight to each training instance.
For each of t iterations {
    Apply learning algorithm to weighted dataset and store resulting model;
    Compute error e of model on weighted dataset and store error;
    If e equal to zero, or e greater or equal to 0.5 {
        Delete current classifier;
        Terminate model generation;
    }
    For each instance in dataset {
        If instance is classified correctly by model {
            Multiply weight of instance by $e / (1 - e)$;
        }
    }
    Normalize weight of all instances.
}
```

```
Classification:
Assign weight of zero to all classes.
For each of the t (or less) models {
        Add $-\log(e / (1 - e))$ to weight of class predicted by model.
}
Return class with highest weight.
```

The boosted decision trees use the boosting technique in which several decision trees, C4.5 trees for example, provide outputs for the final single classifier. We will use boosted C.45 trees for our experiments.

### 1.2.9 Related works on identification of problematic dialog

Concerning the problem of identification of problematic dialog (belonging to dialog classification problem), there are several previous works. The summary of these works is given in Table 1.3.

In 1999, Litman et al [27] developed a PDI. In their work, they defined problematic dialogs as the ones that have poor speech recognition performance and used 'percentage of misrecognition' to label those dialogs. They used different kinds of features including Acoustic, Dialog Efficiency, Dialog Quality, Experimental Parameter, and Lexical features. They tested their system called RIPPER (a rule-based algorithm) on 544 dialogs of AT&T spoken dialog system using 25-fold cross validation. The accuracy of their PDI is 77.40%.

After that, Langkilde et al [26] have developed another PDI based on [27] to use for How May I Help You (HMIHY) system. They built their system on 4774 dialogs using different sources of features: Automatic Speech Recognition (ASR) module, Natural Language Understanding (NLU) module, Dialog Manager (DM) component, and Hand-labeled. They defined problematic dialogs in a different way compared with Litman et al [27], i.e. they used 'task success' instead of 'percentage of misrecognition'. They tested their PDI with RIPPER using 5-fold cross validation and obtained 88.50% accuracy with all features (including hand-labeled features), 87.00% with automatic features, and 86.70% with automatic/task-independent features.

In 2000, Walker et al [25] improved the performance of the system in [26] up to 4% on the test with all features adding one more hand-labeled feature, namely "rsuccess".

Table 1-5 Related works on identification of problematic dialog

| | **1** (1999) | **2** (1999) | **3** (2000) | **4** (2001) | **5** (2002) |
|---|---|---|---|---|---|
| **Corpus** | AT&T (544) | HMIHY (4774) | HMIHY (4774) | HMIHY (4692) | DARPA2001 (1242) |
| **Features** | • Acoustic<br>• Dialog Efficiency<br>• Dialog Quality<br>• Experimental Parameter<br>• Lexical | • ASR<br>• NLU<br>• DM<br>• Hand-labeled | • ASR<br>• NLU<br>• DM<br>• Hand-labeled<br>• rsuccess | • ASR<br>• NLU<br>• DM<br>• Hand-labeled<br>• rsuccess<br>• auto-SLU-success | • Task Success<br>• Efficiency<br>• Qualitative |
| **Labelling** | Percentage of misrecognition | Task Success | Task Success | Task Success | User Rating |
| **Classifier** | RIPPER | RIPPER | RIPPER | RIPPER | DT |
| **Test** | 25-fold CV | 5-fold CV | 5-fold CV | 10-fold CV | 10-fold CV |
| Result (Accuracy) | | | | | |
| **All features** | 77.40% | 88.50% | 92.30% | 91.70% | 67% - 89% (Not exactly mentioned) |
| **Automatic features** | None | 87.00% | 87.00% | 84.90% (+autoSLU-success) | None |
| **Auto, Task-Ind features** | None | 86.70% | 86.70% | 85.40% (+autoSLU-success) | None |

On the basis of [25], Walker et al [24] built an "rsuccess predictor" to get a new automatic feature, named auto-SLU-success, by approximating the information provided by hand-labeled feature 'rsuccess'. This work is motivated by the fact that future work should focus on developing automatic features.

In 2002, Walker et al [23] developed a new PDI on DARPA (Defense Advanced Research Projects Agency) Communicator Corpus 2001 [19]. They labeled problematic dialogs using UserRating instead of TaskSuccess because the main goal is to maximize user satisfaction and it is not always the case that user is satisfied when his/her task is successfully completed. For example, users might be unsatisfied although they completed their task due to the fact that they had to repeat themselves many times before the system understood what they said. On the contrary, users might be satisfied although they did not complete their task probably because of database access problems [23]. Walker et al used the features defined in PARADISE framework [15] including Task Success, Efficiency and Qualitative measures.

We published a paper [22] in 2007, on the basis of [23], studying the effect of named entities and acknowledgement words such as YES, NO, OK on the performance of PDI. That work showed that acknowledgement words are good indicators for identification of problematic dialogs. Therefore, this thesis uses acknowledgement words in combination with Task Success measure, Efficiency measure defined in PARADISE framework and one more feature, namely NumRepetitions i.e. number of times the agent repeats the same utterance, for problem of identification of problematic dialog. All of these features are automatically obtainable.

## 1.3 Scope & Outcome

DCS is a single project, i.e. it is developed independently without using outputs of other projects. In contrast, other projects use the results of DCS as their inputs. The outcome of DCS project is a stand-alone system that can identify problematic dialogs of some corpora in a HCDS using several pattern classification algorithms.

**Chapter Summary**

| | |
|---|---|
| **Motivation** | Evaluation of user satisfaction in HCDS |
| **Purpose** | Identification of problematic dialogs |
| **Domain** | _Data mining & Machine Learning<br>_Pattern Classification |
| **Scope** | Single Project |
| **Outcome** | A stand-alone system that can identify problematic dialogs |

# CHAPITRE 2

# PLANNING

## 2.1 Methodology

### 2.1.1 Theory framework [15][16]

PARADISE (PARAdigm for DIalog System Evaluation) is a general framework for evaluating spoken dialogue agents. It is based on the structure of objectives. It posits that system performance can be correlated with a meaningful external criterion such as usability which can be directly measured by user satisfaction.

User satisfaction has been frequently used as an external indicator of the goodness of a dialog. Figure 2.1 illustrates PARADISE diagram. To maximize user satisfaction, one must maximize task success measure and minimize dialog costs including efficiency measures and qualitative measures.

PARADISE uses a decision-theoretic framework to specify the relative contribution of various factors to an agent's overall performance. It capitalizes on maximizing user satisfaction through 3 general factors:

- **Task success measure**: task completion.
- **Efficiency measures**: total time on task, number of turns on task, number of times user and agent speak at the same time, average duration of system turns, average duration of user turns…
- **Qualitative measures**: number of repair utterances, DATE (Dialogue Act Tagging Scheme for Evaluation of HCDS).

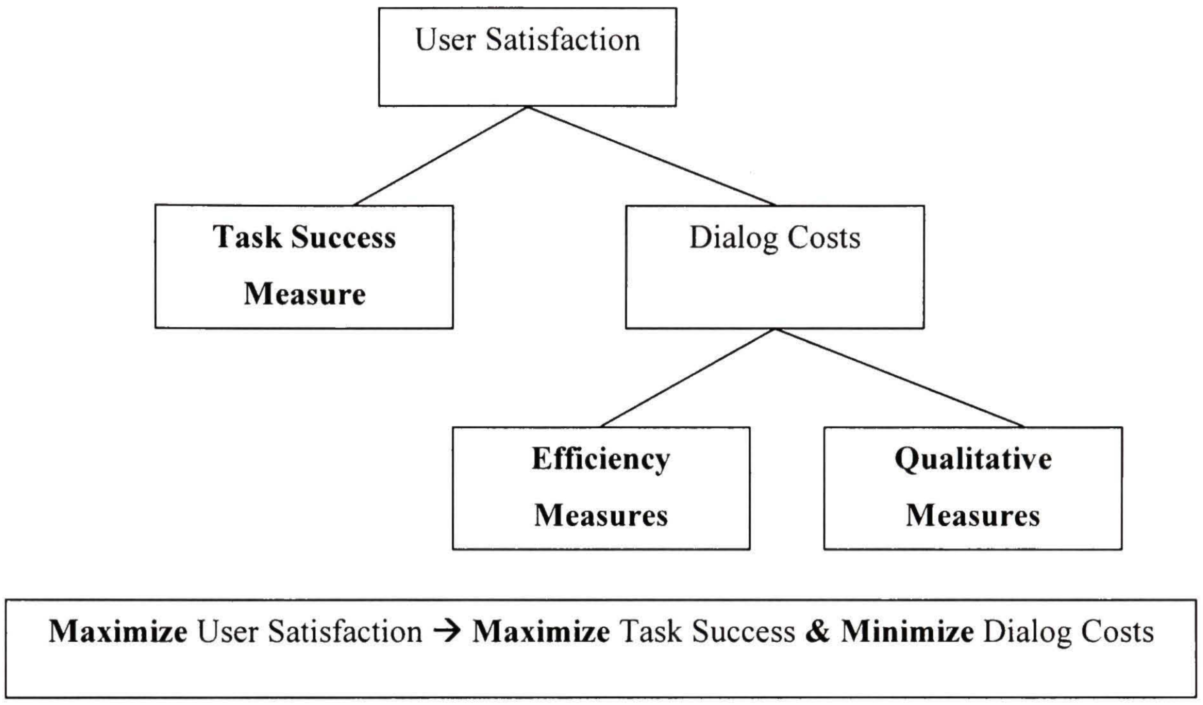


**Figure 2-1** ***PARADISE diagram.***

### 2.1.2 Dialog features for classification task

DCS is a complete and automated system. Therefore, dialog features used for classification must be automatically extractable. Based on PARADISE framework, we define a list of automatically extractable dialog like the following:

1. <u>Task Completion Measure</u>:
   - ***TaskSuccess*** : Task completion of dialog

2. <u>Efficiency Measures</u>:
   - ***TimeOnTask*** **:** Total duration of dialog
   - ***TurnsOnTask*** : Number of turns of dialog including system and user turns.
   - ***NumOverlaps***: Number of times system & user speak at the same time.
   - ***MeanUserTurnDuration***: Average user's turn duration.
   - ***MeanSystemTurnDuration***: Average system's turn duration.

- ***MeanWordsPerUserTurn***: Average number of words of user turn.
- ***MeanWordsPerSystemTurn***: Average number of words of system turn.
- ***Phonetype***: The type of telephone handset that user uses.

Moreover, the problem is to identify problematic dialogs, so the automatically extractable feature "Number of negative acknowledgement words" could be a good one. Negative acknowledgement words include: NO, NOP, FALSE, INCORRECT, WRONG, ERASE. We call that feature ***NumNegativeACKwords***. Plus, we also find that number of times the system repeat the same utterance is a potentially useful feature. We denote this feature as ***NumRepetitions***.

Dialog classification is a 2-class pattern classification problem with set of classes: C = {C1="Good", C2="Bad"}. Hence, a dialog feature vector can be represented in the form:

(***TaskSuccess***, ***TimeOnTask***, ***TurnsOnTask***, ***NumOverlaps***,
***MeanUserTurnDuration***, ***MeanSystemTurnDuration***,
***MeanWordsPerUserTurn***, ***MeanWordsPerSystemTurn***,
***Phonetype***, ***NumNegativeACKwords***,
***NumRepetitions***, ***Label)***

### 2.1.3 Selection of learning schemes [4][5]

It is seldom known in advance which procedure will perform best or even well for any given problem, so the "trial-and-error" approach is always employed in practical data mining application. Particularly, it is tempting to try out different learning schemes with different combinations of their options on a given dataset to select the one that works best.

Among numerous of learning schemes, decision tree has been the most widely used algorithm in practice because it has the following advantages and disadvantage:

- Decision Tree is interpreted so easily because it uses subset of attributes. Practical data mining applications generally require interpretable models.
- Decision Tree efficiently classifies new samples by simply traversing the tree structure without requiring much computation.
- Decision Tree can be applied to any kind of data structure: mixed data type (nominal and numeric data) and data with high dimensionality.
- Decision Tree helps to determine which attribute is the most important for numeric prediction and pattern classification.
- Decision Tree is appropriate for limited dataset.
- It is difficult to design an optimal tree, probably leading to a large tree with poor error rates.

Therefore, decision tree is chosen to be the based learning scheme in our DCS. The learning schemes that we use for DCS are similar to those of state-of-the-art [21] so that we can compare the results.

Those schemes are the followings:

- Zero-Rule Model: This algorithm is useful for determining a baseline performance as a benchmark for other learning schemes.
- Logistic Regression Model: This model is appropriate for 2-class pattern classification problem and usually works well on many datasets.
- One-Rule algorithm: This simple model encourages a “simplicity-first”. Sometimes other learning schemes actually perform worse than this model due to “serious over-fitting”.
- C4.5 Tree.
- Boosted C4.5 Trees.
- Logistic Model Tree.

### 2.1.4 Selection of corpora

We use two publicly distributed corpora from Linguistic Data Consortium. The first corpus is DARPA Communicator 2000 and the second is DARPA Communicator 2001. We choose these corpora because they are available to download and we can compare our results with the state-of-the-art results [23]. The description of DARPA 2000 & 20001 is given below.

#### 2.1.4.1 DARPA 2000 Communicator Corpus [18]

DARPA 2000 Communicator Evaluation was produced by Linguistic Data Consortium (LDC) catalog number LDC2002S56 and ISBN 1-58563-258-9 in the frame of Communicator program. The original goals of the Communicator program were to support the creation of speech-enabled interfaces that scale gracefully across modalities, from speech-only to interfaces that include graphics, maps, pointing and gesture.

The actual research that led to the data collections in 2000 and 2001 explored ways to construct better spoken-dialogue systems, with which users interact via speech-alone to perform relatively complex tasks such as travel planning. During 2000 and 2001 two large data sets were collected, in which users used the Communicator systems built by the research groups to do travel planning.

Nine sites participated in this project: ATT, BBN, Carnegie Mellon University, IBM, MIT, MITRE, NIST, SRI and University of Colorado at Boulder. In 2000, each user called the nine different automated travel-planning systems to make simulated flight reservations. The order in which the users encountered the systems was counterbalanced, for statistical analysis purposes.

#### 2.1.4.2 DARPA 2001 Communicator Corpus [19]

DARPA 2001 Communicator Evaluation was produced by Linguistic Data Consortium (LDC) catalog number LDC2003S01 and ISBN 1-58563-259-7 in the frame of Communicator program. The original goals of the Communicator program were to support the creation of speech-enabled interfaces that scale gracefully across modalities, from speech-only to interfaces that include graphics, maps, pointing and gesture.

The actual research that led to the data collections in 2000 and 2001 explored ways to construct better spoken-dialogue systems, with which users interact via speech-alone to perform relatively complex tasks such as travel planning. During 2000 and 2001 two large data sets were collected, in which users used the Communicator systems built by the research groups to do travel planning.

The following sites participated in this project: ATT, BBN, Carnegie Mellon University, IBM, Lucent Bell Labs, MIT, SRI and University of Colorado at Boulder.

## 2.2 Evaluation measures [4][14]

Different metrics could be used to evaluate the performance of a learning scheme. For a 2-class pattern classification problem such as ours, where we have a positive class (C+) for good dialog and a negative class (C–) for bad dialog, we can use the following metrics:

- Accuracy: reflects the overall correctness of the learning scheme.
- Precision of C+ ($P_+$): reflects the correctness of the learning scheme on C+.
- Precision of C– ($P_-$): reflects the correctness of the learning scheme on C–.
- Recall of C+ ($R_+$): is the accuracy among positive instances.
- Recall of C– ($R_-$): is the accuracy among negative instances.

- F-measure: is a combination of Precision and Recall. F-measure is computed by the following formula:

$$F = \frac{2PR}{P + R} \tag{2.1}$$

These metrics are calculated from a confusion matrix as shown in Table 2.1. Accuracy varies from 0% to 100% whereas Precision, Recall, and F-measure vary from 0 to 1.
The higher the metric is, the better the system performance is.

Table 2-1 Confusion matrix for 2-class problem

| | **Actual C+** | **Actual C-** |
|---|---|---|
| **Predicted C+** | *True Positive (TP)* | *False Positive (FP)* |
| **Predicted C-** | *False Negative (FN)* | *True Negative (TN)* |

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} * 100\% \tag{2.2}$$

$$P_{+} = \frac{TP}{TP + FP} \tag{2.3}$$

$$P_{-} = \frac{TN}{TN + FN} \tag{2.4}$$

$$R_{+} = \frac{TP}{TP + FN} \tag{2.5}$$

$$R_{-} = \frac{TN}{TN + FP} \tag{2.6}$$

$$F_{+} = \frac{2 P_{+} R_{+}}{P_{+} + R_{+}} \tag{2.7}$$

$$F_{-} = \frac{2 P_{-} R_{-}}{P_{-} + R_{-}} \tag{2.8}$$

For dialog classification problem, the best two measures to evaluate DCS are Accuracy and $F_-$ measures due to several reasons:

- Accuracy reflects the overall correctness of the learning scheme but it ignores the difference between error types, so one more measure is needed.
- $F_-$ reflects the precision and recall of class C– and the problem here is to identify problematic dialogs, so $F_-$ is a reasonable choice.

So far we have presented methodology and DCS evaluation measures. In the next section, we describe how to implement and test DCS.

**Chapter Summary**

| | |
|---|---|
| **Methodology** | _Theory Framework: PARADISE<br>_Dialog features for classification task:<br>+ TaskSuccess,<br>+ TimeOnTask,<br>+ TurnsOnTask,<br>+ NumOverlaps,<br>+ MeanUserTurnDuration,<br>+ MeanSystemTurnDuration,<br>+ MeanWordsPerUserTurn,<br>+ MeanWordsPerSystemTurn,<br>+ Phonetype<br>+ NumNegativeACKwords<br>+ NumRepetitions<br>_Learning schemes:<br>+ Zero-Rule Model<br>+ Logistic Regression Model<br>+ One-Rule algorithm<br>+ C4.5 Tree<br>+ Boosted C4.5 Trees<br>+ Logistic Model Tree<br>_Datasets:<br>+ DARPA Communicator 2000 Corpus<br>+ DARPA Communicator 2001 Corpus |
| **Evaluation measures** | _Accuracy<br>_F-measure of bad dialogs |

# CHAPITRE 3

# IMPLEMENTATION

## 3.1 System Implementation

After having described the definition and planning of DCS, in this chapter, we will discuss about system implementation. This chapter is divided in several sections where we will cover system design; detail each component of system, and describe experiments performed on system.

### 3.1.1 System Design

A common pattern classification system usually has three main components as shown in Figure 3.1. Those components include:

- Preprocessing: this component preprocesses the input pattern. There are three kinds of input patterns, each of which can be preprocessed with a different way:
    - **Structural pattern**: this kind of pattern is represented by an image (2D or 3D). For examples: character, face, finger-print. We need to remove noise in patterns by using filling and thinning process, normalize patterns to the same size, correct slant. Then, we can extract Contour or Skeleton for feature extraction.
    - **Time-varying pattern**: is represented by a waveform. For example: speech. We can preprocess the pattern by doing some filtering, normalizing duration, etc.
    - **Abstract pattern**: this kind of pattern can not be represented explicitly. For example: dialog pattern. For this kind of pattern, we can use a parser to parse log files to generate dialog information. Abstract pattern is what we will be facing.
- Feature Extraction: this component extracts features from processed pattern.
- Pattern Classification: this component classifies pattern into a set of categorical classes based on extracted features.

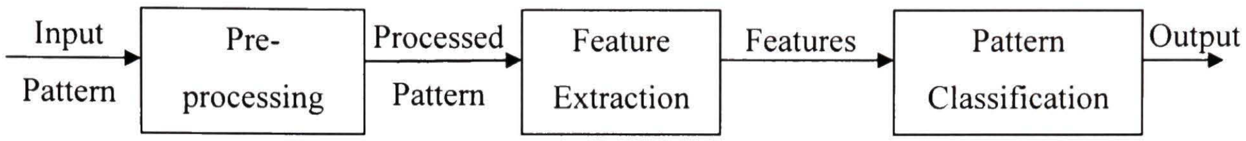


**Figure 3-1** ***Pattern Classification System Diagram.***

On the basis of the common pattern classification system, DCS is designed as shown in Figure 3.2. The input pattern in DCS is represented by log files recording what the agent and the user communicated and time when the dialog took place. These log files are parsed by a parser to produce dialog information. This is the preprocessing phase.
Then, an extractor will extract dialog features from dialog. Finally, classifier component classifies dialog into one of two categorical classes {Good, Bad} using dialog features.

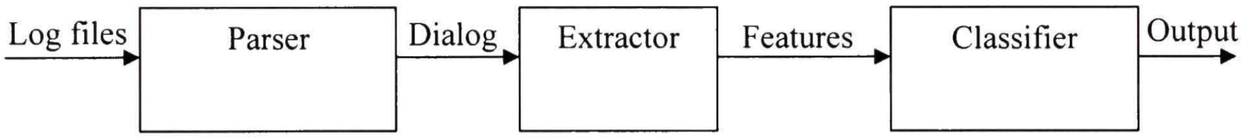


**Figure 3-2** ***Dialog Classification System Diagram.***

### 3.1.2 Parser Component

As mentioned above, parser is the first component in DCS that serves to remove unuseful information or noise from files in order to extract useful dialog information. Normally, a dialog consists of tens of system turns and user turns, each of which has three main properties that the parser needs to extract. Those properties include **start time**, **finish time**, and **utterance**.

In general, the input of the parser is a dialog stored in a form of log files and the output of the parser is a list of system turns and user turns with their three previously mentioned properties.

The log file format of different corpora varies. The parser has to adapt to this reality. Therefore, different method is used for different corpus. In the next section, we will describe how the parser parses log files of two DARPA corpora.

#### 3.1.2.1 DARPA 2000 Communicator Corpus

The format of log file, named "summary file", of this corpus is displayed in Table 3.1. The parser parses this file to extract **start time, finish time,** and **utterance** of system/user turns by seeking their keywords: "System started speaking.", "System finished speaking.", and "System said:"/"User said:" respectively.

Table 3-1 Summary file in DARPA 2000 corpus

```
Fri Jul 7 2000 at 12:20:53.02: New system turn began.
Fri Jul 7 2000 at 12:20:53.02: System started speaking.    → start time of system turn
Fri Jul 7 2000 at 12:21:09.36: System finished speaking.   → finish time of system turn
System said: Welcome. You are logged in as a guest user of AT&T … → system utterance

Fri Jul 7 2000 at 12:21:15.05: New user turn began.
Fri Jul 7 2000 at 12:21:15.05: User started speaking.      → start time of user turn
Fri Jul 7 2000 at 12:21:16.38: User finished speaking.     → finish time of user turn
User said: HONOLULU HAWAII                                 → user utterance

Fri Jul 7 2000 at 12:21:17.13: New system turn began.
Fri Jul 7 2000 at 12:21:17.13: System started speaking.    → start time of system turn
Fri Jul 7 2000 at 12:21:22.32: System finished speaking.   → finish time of system turn
System said: Leaving from Honolulu, And, what city are you flying to? → system utterance
…
```

Next, the parser also parses a "survey file" as shown in Table 3.2 to extract other useful dialog information such as system status, task success, and points of five answers to the previously mentioned questions.

Table 3-2 Survey file in DARPA 2000 corpus

```
AT&T  18  11:32 EDT  2000/07/07  Alive  Yes  3  4  2  3  4
        I was given the wrong option for the flight but it was my fault because I asked for an
option by number and was given the first option that was read to me. I should have started
over again to obtain the preferred departure time.  Also, I thought the names of the cities
were a little hard to understand.
```

In the example in Table 3.2, we have:

- System status=Alive
- Task success=Yes
- Task Ease=3
- TTSPerf=4
- User Expertise=2
- Expected Behavior=3
- Future Use=4

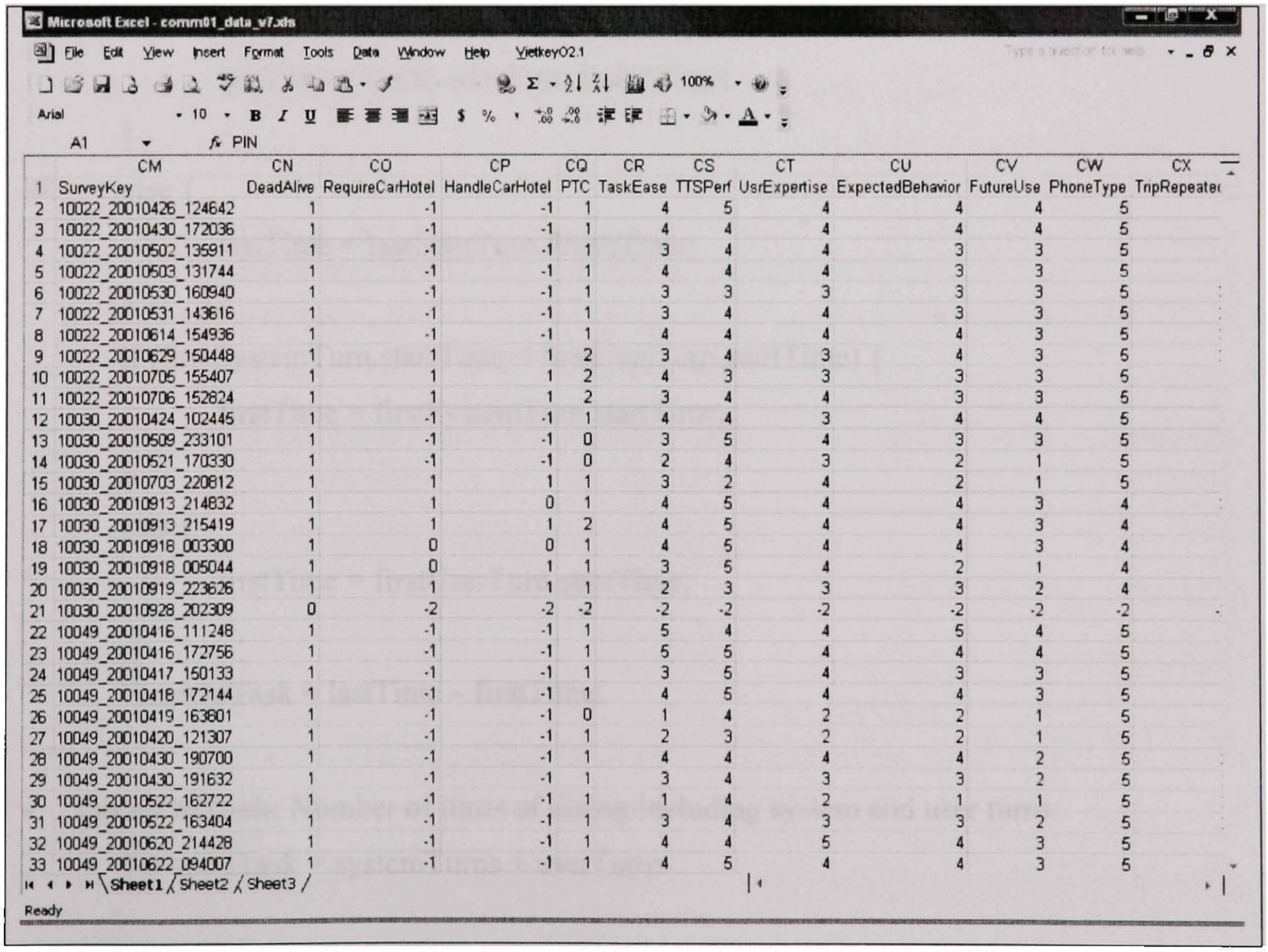
#### 3.1.2.2 DARPA 2001 Communicator Corpus

In DARPA 2001 corpus, the parser parses a log file named "Transcript file" as shown in Table 3.3 to get **start time, finish time,** and **utterance** of system/user turns and other log file named "Communicator file" given in Table 3.4 to get TaskSucess (PTC), Task Ease, TTSPerf, User Expertise, Expected Behavior, Future Use…

Table 3-3 Transcript file in DARPA 2001 corpus

```
Task Start Time 988306633.730
 ...
 Sys: 988306635.820  988306637.850  what is your full name?
    → start time         finish time        utterance
 User: 988306639.280  988306640.930  Asr: adam simons  /  Transcr: adam simons
    → start time         finish time        utterance
 ...
Task End Time 988306878.120
```

Table 3-4 Communicator file in DARPA 2001 corpus



Microsoft Excel - comm01_data_v7.xls

File Edit View Insert Format Tools Data Window Help Vietkey02.1

A1 PIN

|  | CM | CN | CO | CP | CQ | CR | CS | CT | CU | CV | CW | CX |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SurveyKey | DeadAlive | RequireCarHotel | HandleCarHotel | PTC | TaskEase | TTSPerf | UsrExpertise | ExpectedBehavior | FutureUse | PhoneType | TripRepeate |
| 2 | 10022_20010426_124642 | 1 | -1 | -1 | 1 | 4 | 5 | 4 | 4 | 4 | 5 |  |
| 3 | 10022_20010430_172036 | 1 | -1 | -1 | 1 | 4 | 4 | 4 | 4 | 4 | 5 |  |
| 4 | 10022_20010502_135911 | 1 | -1 | -1 | 1 | 3 | 4 | 4 | 3 | 3 | 5 |  |
| 5 | 10022_20010503_131744 | 1 | -1 | -1 | 1 | 4 | 4 | 4 | 3 | 3 | 5 |  |
| 6 | 10022_20010530_160940 | 1 | -1 | -1 | 1 | 3 | 3 | 3 | 3 | 3 | 5 |  |
| 7 | 10022_20010531_143616 | 1 | -1 | -1 | 1 | 3 | 4 | 4 | 3 | 3 | 5 |  |
| 8 | 10022_20010614_154936 | 1 | -1 | -1 | 1 | 4 | 4 | 4 | 4 | 3 | 5 |  |
| 9 | 10022_20010629_150448 | 1 | 1 | 1 | 2 | 3 | 4 | 3 | 4 | 3 | 5 |  |
| 10 | 10022_20010705_155407 | 1 | 1 | 1 | 2 | 4 | 3 | 3 | 3 | 3 | 5 |  |
| 11 | 10022_20010706_152824 | 1 | 1 | 1 | 2 | 3 | 4 | 4 | 3 | 3 | 5 |  |
| 12 | 10030_20010424_102445 | 1 | -1 | -1 | 1 | 4 | 4 | 3 | 4 | 4 | 5 |  |
| 13 | 10030_20010509_233101 | 1 | -1 | -1 | 0 | 3 | 5 | 4 | 3 | 3 | 5 |  |
| 14 | 10030_20010521_170330 | 1 | -1 | -1 | 0 | 2 | 4 | 3 | 2 | 1 | 5 |  |
| 15 | 10030_20010703_220812 | 1 | 1 | 1 | 1 | 3 | 2 | 4 | 2 | 1 | 5 |  |
| 16 | 10030_20010913_214832 | 1 | 1 | 0 | 1 | 4 | 5 | 4 | 4 | 3 | 4 |  |
| 17 | 10030_20010913_215419 | 1 | 1 | 1 | 2 | 4 | 5 | 4 | 4 | 3 | 4 |  |
| 18 | 10030_20010918_003300 | 1 | 0 | 0 | 1 | 4 | 5 | 4 | 4 | 3 | 4 |  |
| 19 | 10030_20010918_005044 | 1 | 0 | 1 | 1 | 3 | 5 | 4 | 2 | 1 | 4 |  |
| 20 | 10030_20010919_223626 | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 2 | 4 |  |
| 21 | 10030_20010928_202309 | 0 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 |  |
| 22 | 10049_20010416_111248 | 1 | -1 | -1 | 1 | 5 | 4 | 4 | 5 | 4 | 5 |  |
| 23 | 10049_20010416_172756 | 1 | -1 | -1 | 1 | 5 | 5 | 4 | 4 | 4 | 5 |  |
| 24 | 10049_20010417_150133 | 1 | -1 | -1 | 1 | 3 | 5 | 4 | 3 | 3 | 5 |  |
| 25 | 10049_20010418_172144 | 1 | -1 | -1 | 1 | 4 | 5 | 4 | 4 | 3 | 5 |  |
| 26 | 10049_20010419_163801 | 1 | -1 | -1 | 0 | 1 | 4 | 2 | 2 | 1 | 5 |  |
| 27 | 10049_20010420_121307 | 1 | -1 | -1 | 1 | 2 | 3 | 2 | 2 | 1 | 5 |  |
| 28 | 10049_20010430_190700 | 1 | -1 | -1 | 1 | 4 | 4 | 4 | 4 | 2 | 5 |  |
| 29 | 10049_20010430_191632 | 1 | -1 | -1 | 1 | 3 | 4 | 3 | 3 | 2 | 5 |  |
| 30 | 10049_20010522_162722 | 1 | -1 | -1 | 1 | 1 | 4 | 2 | 1 | 1 | 5 |  |
| 31 | 10049_20010522_163404 | 1 | -1 | -1 | 1 | 3 | 4 | 3 | 3 | 2 | 5 |  |
| 32 | 10049_20010620_214428 | 1 | -1 | -1 | 1 | 4 | 4 | 5 | 4 | 3 | 5 |  |
| 33 | 10049_20010622_094007 | 1 | -1 | -1 | 1 | 4 | 5 | 4 | 4 | 3 | 5 |  |

Sheet1 / Sheet2 / Sheet3

Ready

### 3.1.3 Extractor Component

Recall that the dialog feature vector which we want to extract is:

( ***TaskSuccess***, ***TimeOnTask,*** ***TurnsOnTask,***
***NumOverlaps,*** ***MeanUserTurnDuration,*** ***MeanSystemTurnDuration,***
***MeanWordsPerUserTurn,*** ***MeanWordsPerSystemTurn,*** ***Phonetype,***
***NumNegativeACKwords,*** ***NumRepetitions,*** ***Label)***

From dialog information (start time, finish time …) provided by the parser, the extractor component computes dialog features as follow:

- **TimeOnTask:** Total duration of dialog

```
if (lastSystemTurn.finishTime > lastUserTurn.finishTime) {
        lastTime = lastSystemTurn.finishTime;
}
else {
        lastTime = lastUserTurn.finishTime;
}
if (firstSystemTurn.startTime < firstUserTurn.startTime) {
        firstTime = firstSystemTurn.startTime;
}
else {
        firstTime = firstUserTurn.startTime;
}
TimeOnTask = lastTime – firstTime;
```

- **TurnsOnTask:** Number of turns of dialog including system and user turns.

```
TurnsOnTask = systemTurns + userTurns
```

- **NumOverlaps:** Number of times system and user speak at the same time.

```
if ( ((usrStartTime < sysFinishTime) && (sysFinishTime < usrFinishTime))
        || ((usrStartTime < sysStartTime) && (sysStartTime < usrFinishTime)) ) {

            numOverlaps ++;
}
```

- **MeanSystemTurnDur:** Average system's turn duration.
  MeanSystemTurnDur = sum (durations of sysTurns) / No. of sysTurns

- **MeanUserTurnDur:** Average user's turn duration.
  MeanUserTurnDur = sum (durations of usrTurns) / No. of usrTurns

- **MeanWordsPerSystemTurn:** Average number of words of system turn.
  MeanWordsPerSystemTurn = sum (numOfWords of sysTurns) / No. of sysTurns

- **MeanWordsPerUserTurn:** Average number of words of user turn.
  MeanWordsPerUserTurn = sum(numOfWords of usrTurns) / No. of usrTurns

- **UserRating =** Task Ease + TTSPerf + User Expertise + Expected Behavior + Future Use

```
For DARPA 2000 corpus:    If  (UserRating > 12) Then Label="Bad";
                          Else Label="Good";

For DARPA 2001 corpus:    If  (UserRating < 17) Then Label="Bad";
                          Else Label="Good";
```

Different thresholds are used for different corpora. Section 3.2.1 presents how these values are selected.

### 3.1.4 Classifier Component

We will use different classifier components which consist of several learning algorithms in order to determine which one is the most appropriate for our tasks. These classifiers are:

- Basic C4.5 Tree (our own implementation)
- Zero-Rule Model
- Logistic Regression Model
- One-Rule Algorithm
- C4.5 Tree
- Boosted C4.5 Trees
- Logistic Model Tree

One of them, the basic C4.5 Tree, is our own implementation. We choose C4.5 tree because it is simple and the most widely used decision tree nowadays. We want to implement it in a basic manner by ourselves so that we can compare our own results with those of other machine learning algorithms that belongs to the library WEKA (Waikato Environment for Knowledge Analysis) developed by many machine learning experts of the University of Waikato [17].

### 3.1.5 System Development

DCS is developed in Java language with Eclipse Java Editor using three-tier software architecture to make the system modular and flexible. The three-tier model is considered to be a software architecture including user interface, functional process logic (business rules) and data storage developed and maintained as independent modules [13].

The three-tier architecture, as its words imply, has the following 3 tiers: Presentation Tier (Layer), Business Logic Layer, and Database Layer. Those layers are connected together and illustrated in Figure 3.3.

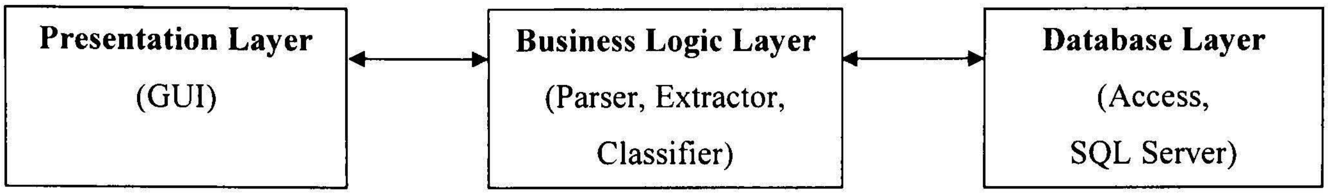
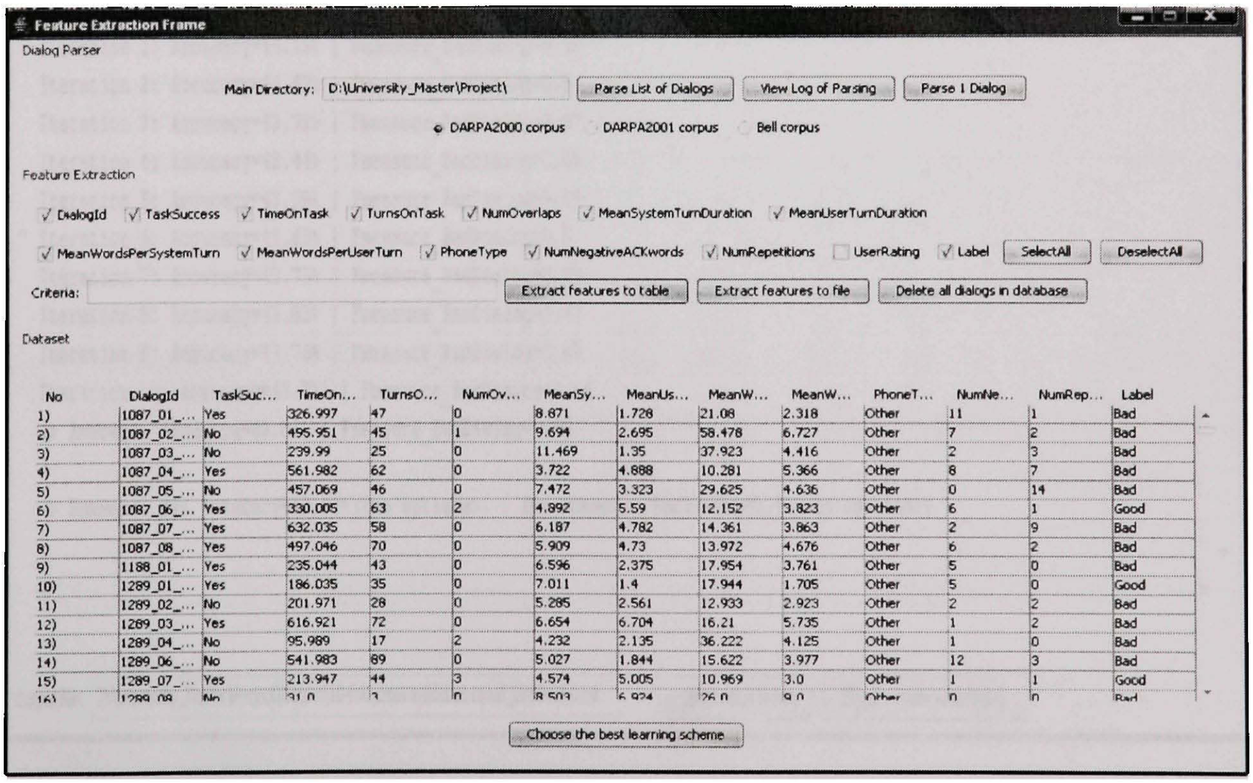


**Figure 3-3** ***Three-tier software architecture.***

### 3.1.5.1 Presentation Layer

Presentation layer contains graphic user interface that helps user interacts visually with DCS. There are three main frames in DCS: Feature Extraction Frame, Learning Scheme Frame, and Dialog Classification Frame. Figures 3.4, 3.5, 3.6 display these frames respectively.

Feature Extraction Frame helps users to parse log files of the corpora and extract dialog features. Users can add or remove a specific feature that they want to extract by checking or unchecking its corresponding check box.



Main Directory: D:\University_Master\Project\

● DARPA2000 corpus ○ DARPA2001 corpus ○ Bell corpus

✓ DialogId ✓ TaskSuccess ✓ TimeOnTask ✓ TurnsOnTask ✓ NumOverlaps ✓ MeanSystemTurnDuration ✓ MeanUserTurnDuration

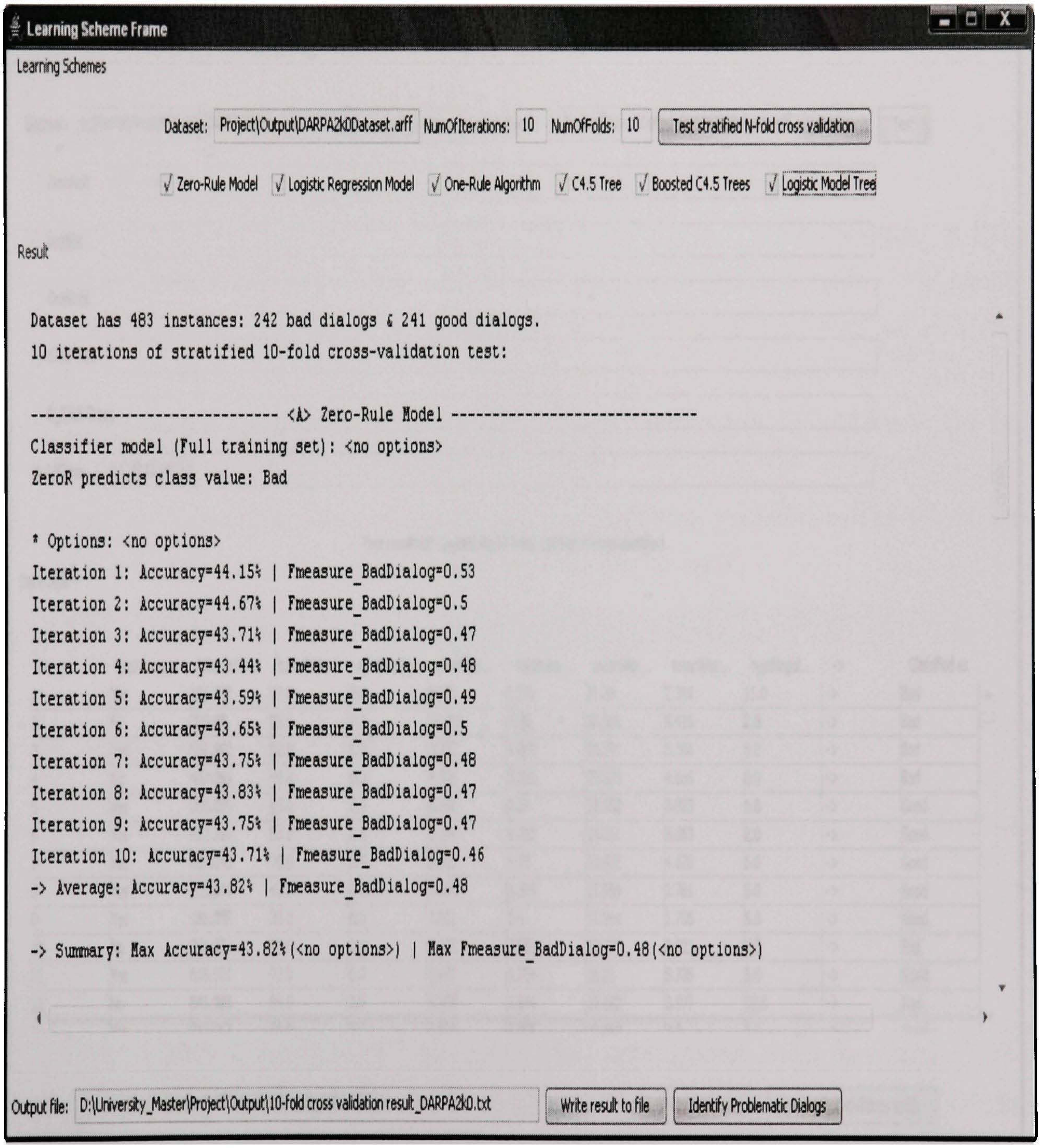
✓ MeanWordsPerSystemTurn ✓ MeanWordsPerUserTurn ✓ PhoneType ✓ NumNegativeACKwords ✓ NumRepetitions ☐ UserRating ✓ Label

Criteria:

| No | DialogId | TaskSuc... | TimeOn... | TurnsO... | NumOv... | MeanSy... | MeanUs... | MeanW... | MeanW... | PhoneT... | NumNe... | NumRep... | Label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1) | 1087_01_... | Yes | 326.997 | 47 | 0 | 8.871 | 1.728 | 21.08 | 2.318 | Other | 11 | 1 | Bad |
| 2) | 1087_02_... | No | 495.951 | 45 | 1 | 9.694 | 2.695 | 58.478 | 6.727 | Other | 7 | 2 | Bad |
| 3) | 1087_03_... | No | 239.99 | 25 | 0 | 11.469 | 1.35 | 37.923 | 4.416 | Other | 2 | 3 | Bad |
| 4) | 1087_04_... | Yes | 561.982 | 62 | 0 | 3.722 | 4.888 | 10.281 | 5.366 | Other | 8 | 7 | Bad |
| 5) | 1087_05_... | No | 457.069 | 46 | 0 | 7.472 | 3.323 | 29.625 | 4.636 | Other | 0 | 14 | Bad |
| 6) | 1087_06_... | Yes | 330.005 | 63 | 2 | 4.892 | 5.59 | 12.152 | 3.823 | Other | 6 | 1 | Good |
| 7) | 1087_07_... | Yes | 632.035 | 58 | 0 | 6.187 | 4.782 | 14.361 | 3.863 | Other | 2 | 9 | Bad |
| 8) | 1087_08_... | Yes | 497.046 | 70 | 0 | 5.909 | 4.73 | 13.972 | 4.676 | Other | 6 | 2 | Bad |
| 9) | 1188_01_... | Yes | 235.044 | 43 | 0 | 6.596 | 2.375 | 17.954 | 3.761 | Other | 5 | 0 | Bad |
| 10) | 1289_01_... | Yes | 186.035 | 35 | 0 | 7.011 | 1.4 | 17.944 | 1.705 | Other | 5 | 0 | Good |
| 11) | 1289_02_... | No | 201.971 | 28 | 0 | 5.285 | 2.561 | 12.933 | 2.923 | Other | 2 | 2 | Bad |
| 12) | 1289_03_... | Yes | 616.921 | 72 | 0 | 6.654 | 6.704 | 16.21 | 5.735 | Other | 1 | 2 | Bad |
| 13) | 1289_04_... | No | 95.989 | 17 | 2 | 4.232 | 2.135 | 36.222 | 4.125 | Other | 1 | 0 | Bad |
| 14) | 1289_06_... | No | 541.983 | 89 | 0 | 5.027 | 1.844 | 15.622 | 3.977 | Other | 12 | 3 | Bad |
| 15) | 1289_07_... | Yes | 213.947 | 44 | 3 | 4.574 | 5.005 | 10.969 | 3.0 | Other | 1 | 1 | Good |
| 16) | 1289_08 | No | 31.0 | 3 | 0 | 8.503 | 6.924 | 26.0 | 9.0 | Other | 0 | 0 | Bad |

**Figure 3-4** ***Feature Extraction Frame.***

Next, Learning Scheme Frame helps users to test performance of different learning schemes on a given dataset to choose the best one for this dataset.


Learning Scheme Frame
Learning Schemes
Test stratified N-fold cross validation
Result
Write result to file
Identify Problematic Dialogs

Dataset: Project\Output\DARPA2k0Dataset.arff NumOfIterations: 10 NumOfFolds: 10

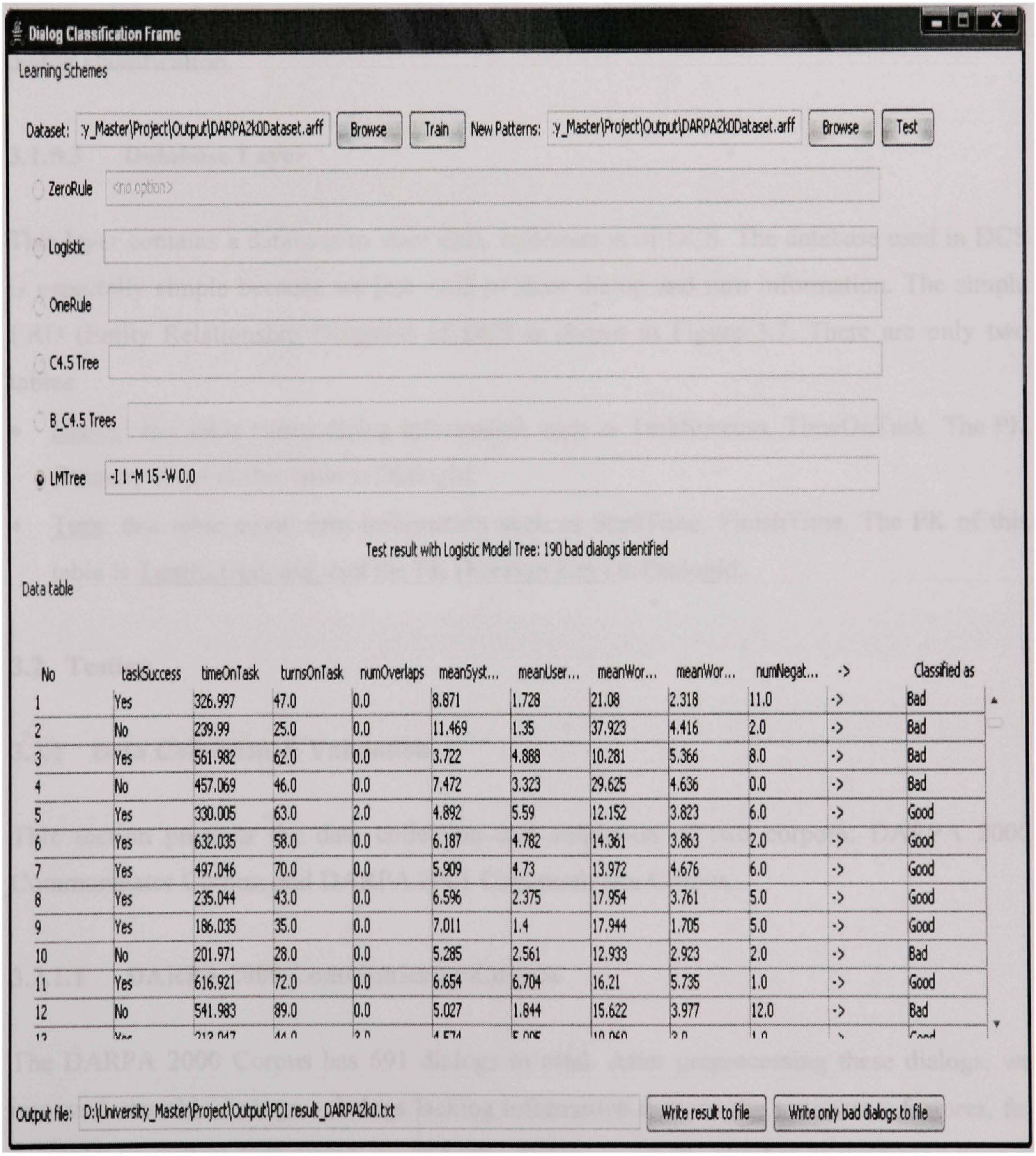
√ Zero-Rule Model √ Logistic Regression Model √ One-Rule Algorithm √ C4.5 Tree √ Boosted C4.5 Trees √ Logistic Model Tree

```
Dataset has 483 instances: 242 bad dialogs & 241 good dialogs.
10 iterations of stratified 10-fold cross-validation test:

----------------------------- <A> Zero-Rule Model -----------------------------
Classifier model (Full training set): <no options>
ZeroR predicts class value: Bad

* Options: <no options>
Iteration 1: Accuracy=44.15% | Fmeasure_BadDialog=0.53
Iteration 2: Accuracy=44.67% | Fmeasure_BadDialog=0.5
Iteration 3: Accuracy=43.71% | Fmeasure_BadDialog=0.47
Iteration 4: Accuracy=43.44% | Fmeasure_BadDialog=0.48
Iteration 5: Accuracy=43.59% | Fmeasure_BadDialog=0.49
Iteration 6: Accuracy=43.65% | Fmeasure_BadDialog=0.5
Iteration 7: Accuracy=43.75% | Fmeasure_BadDialog=0.48
Iteration 8: Accuracy=43.83% | Fmeasure_BadDialog=0.47
Iteration 9: Accuracy=43.75% | Fmeasure_BadDialog=0.47
Iteration 10: Accuracy=43.71% | Fmeasure_BadDialog=0.46
-> Average: Accuracy=43.82% | Fmeasure_BadDialog=0.48

-> Summary: Max Accuracy=43.82%(<no options>) | Max Fmeasure_BadDialog=0.48(<no options>)
```

Output file: D:\University_Master\Project\Output\10-fold cross validation result_DARPA2k0.txt

**Figure 3-5** ***Learning Scheme Frame.***

Finally, Dialog Classification Frame helps user to identify problematic dialogs with the previously chosen learning scheme.



Dataset: :y_Master\Project\Output\DARPA2k0Dataset.arff

New Patterns: :y_Master\Project\Output\DARPA2k0Dataset.arff

ZeroRule <no option>

Logistic

OneRule

C4.5 Tree

B_C4.5 Trees

[✓ LMTree] -I 1 -M 15 -W 0.0

Test result with Logistic Model Tree: 190 bad dialogs identified

| No | taskSuccess | timeOnTask | turnsOnTask | numOverlaps | meanSyst... | meanUser... | meanWor... | meanWor... | numNegat... | -> | Classified as |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Yes | 326.997 | 47.0 | 0.0 | 8.871 | 1.728 | 21.08 | 2.318 | 11.0 | -> | Bad |
| 2 | No | 239.99 | 25.0 | 0.0 | 11.469 | 1.35 | 37.923 | 4.416 | 2.0 | -> | Bad |
| 3 | Yes | 561.982 | 62.0 | 0.0 | 3.722 | 4.888 | 10.281 | 5.366 | 8.0 | -> | Bad |
| 4 | No | 457.069 | 46.0 | 0.0 | 7.472 | 3.323 | 29.625 | 4.636 | 0.0 | -> | Bad |
| 5 | Yes | 330.005 | 63.0 | 2.0 | 4.892 | 5.59 | 12.152 | 3.823 | 6.0 | -> | Good |
| 6 | Yes | 632.035 | 58.0 | 0.0 | 6.187 | 4.782 | 14.361 | 3.863 | 2.0 | -> | Good |
| 7 | Yes | 497.046 | 70.0 | 0.0 | 5.909 | 4.73 | 13.972 | 4.676 | 6.0 | -> | Good |
| 8 | Yes | 235.044 | 43.0 | 0.0 | 6.596 | 2.375 | 17.954 | 3.761 | 5.0 | -> | Good |
| 9 | Yes | 186.035 | 35.0 | 0.0 | 7.011 | 1.4 | 17.944 | 1.705 | 5.0 | -> | Good |
| 10 | No | 201.971 | 28.0 | 0.0 | 5.285 | 2.561 | 12.933 | 2.923 | 2.0 | -> | Bad |
| 11 | Yes | 616.921 | 72.0 | 0.0 | 6.654 | 6.704 | 16.21 | 5.735 | 1.0 | -> | Good |
| 12 | No | 541.983 | 89.0 | 0.0 | 5.027 | 1.844 | 15.622 | 3.977 | 12.0 | -> | Bad |

Output file: D:\University_Master\Project\Output\PDI result_DARPA2k0.txt

**Figure 3-6** ***Dialog Classification Frame.***

#### 3.1.5.2 Business Logic Layer

Business Logic Layer contains business rules, functional algorithms to perform several main functionalities of DCS including parsing, feature extraction, learning scheme testing and dialog classification.

#### 3.1.5.3 Database Layer

This layer contains a database to store data, information of DCS. The database used in DCS is especially simple because we just need to store dialog and turn information. The simple ERD (Entity Relationship Diagram) of DCS is shown in Figure 3.7. There are only two tables:

- Dialog: this table stores dialog information such as TaskSuccess, TimeOnTask. The PK (Primary Key) of this table is DialogId.
- Turn: this table stores turn information such as StartTime, FinishTime. The PK of this table is TurnId,DialogId, and the FK (Foreign Key) is DialogId.

## 3.2 Testing

### 3.2.1 Data Collection & Validation

This section presents the data collection and validation of two corpora: DARPA 2000 Communicator Corpus, and DARPA 2001 Communicator Corpus.

#### 3.2.1.1 DARPA 2000 Communicator Corpus

The DARPA 2000 Corpus has 691 dialogs in total. After preprocessing these dialogs, we removed unusable dialogs – dialogs lacking information that we can not extract features, for example, no-system-turn dialog, no-end-time dialog… – so the number of remaining dialogs is 550. In this corpus, they use inversed Likert-scale as illustrated in Table 3.5.

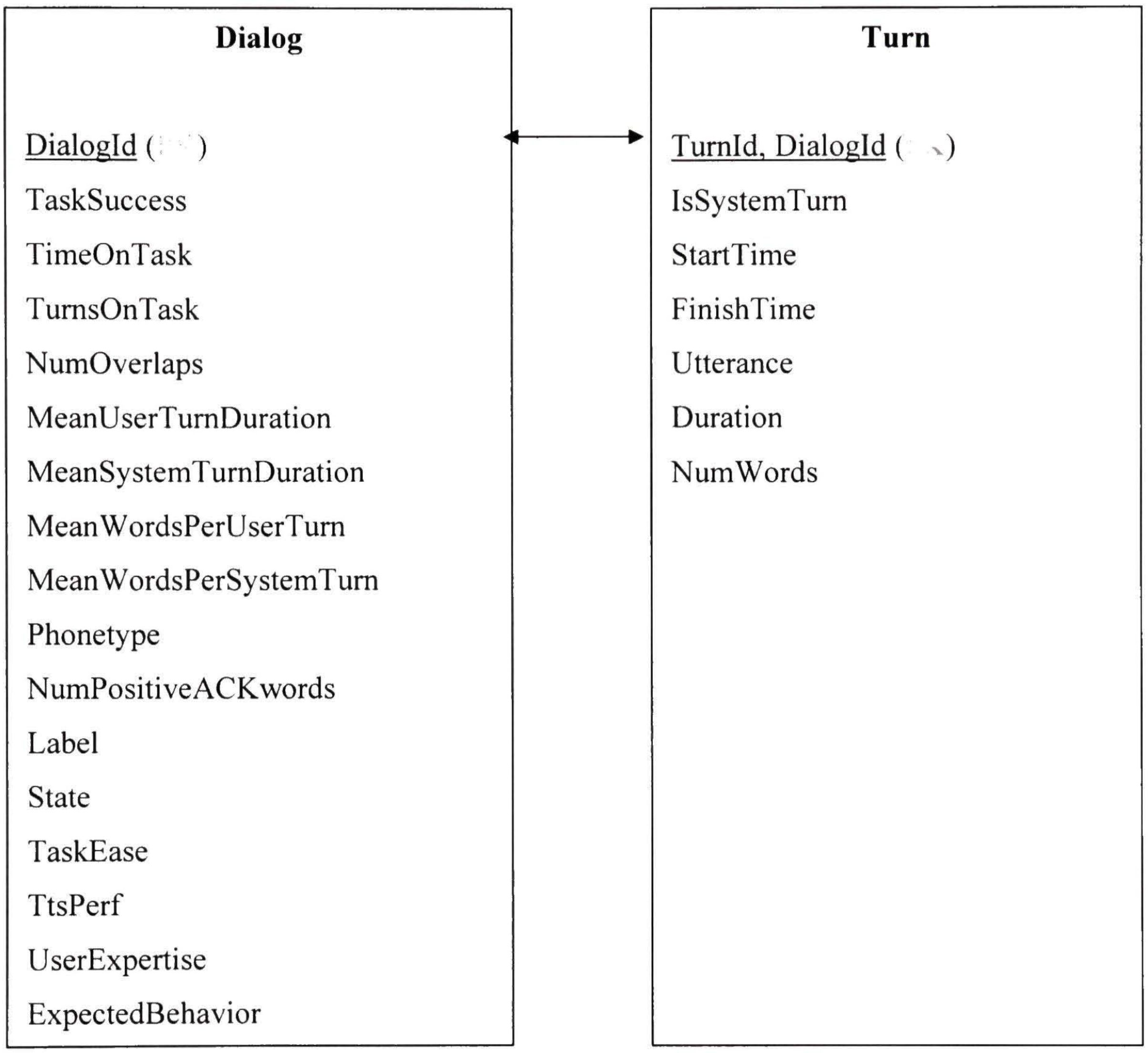


**Figure 3-7** ***ERD of DCS.***

Table 3-5 Likert-scale and Inversed Likert-scale

| Likert-scale | Strongly Disagree | Somewhat Disagree | Neutral | Somewhat Agree | Strongly Agree |
|---|---|---|---|---|---|
| Normal | 1 | 2 | 3 | 4 | 5 |
| Inversed | 5 | 4 | 3 | 2 | 1 |

Recall that there are 5 questions asked, the total value can be varied from 5 to 25. Therefore, we choose 12 as the threshold for labeling dialogs with the meaning that there are at least two questions in which user has "Somewhat Agree".

Hence, Bad dialog in DARPA 2000 is determined by the following condition:

UserRating > 12 → Bad dialog

Applying this condition to the remaining dialogs, we obtain 274 Bad dialogs and 276 Good dialogs. The statistics of data collection and validation of DAPRA 2000 corpus is given in Table 3.6.

Table 3-6 Statistics of DARPA 2000 corpus

| **Total dialogs** | **691** |
|---|---|
| 1. *File-lacking dialogs* | *30* |
| 2. *No-system/user-turn dialogs* | *12* |
| 3. *No-end-time dialogs* | *61* |
| 4. *Dead dialogs* | *15* |
| 5. *Damaged dialogs* | *23* |
| **Remaining dialogs** | **550** (**274** Bad & **276** Good) Inversed Likert-scale |
| → sum (points of 5 questions) > **12** : Bad Dialog | |

### 3.2.1.2 DARPA 2001 Communicator Corpus

The process of collecting and validating data in DARPA 2001 corpus is more complex than that of DARPA 2000 corpus. Table 3.7 summarizes this process. The number of remaining dialogs in DARPA 2001 corpus is 1022.

The Likert-scale shown in Table 3.5 is used in this corpus, so we choose 17 as the threshold for labeling dialogs with the meaning that there are at least two questions in which user has "Somewhat Agree".

Hence, Bad dialog in DARPA 2000 is determined by the following condition:

UserRating < 17 → Bad dialog

Applying this condition to the remaining dialogs, we obtain 472 Bad and 550 Good dialogs.

Table 3-7 Statistics of DARPA 2001 corpus

| | |
|---|---|
| **Total dialogs** (*Transcript file*) | **1684** |
| **No-time** | *113* |
| **Remaining dialogs** | **1571** |
| | |
| **Total dialogs** (*Comm file*) | **1351** |
| **Comm file merged with Excel file** | **1106** |
| **Dead dialogs** | *4* |
| **Remaining dialogs** | **1102** |
| | |
| *Merge (key=id):*<br>**Remaining dialogs (in Transcript file)**<br>vs<br>**Remaining dialogs (in Comm file)**<br>→ | <br>**1571**<br>**vs**<br>**1102**<br>**<u>1022</u>** (**472** Bad **& 550** Good)<br>Likert-scale |
| → sum (points of 5 questions) **< 17** : Bad Dialog | |

### 3.2.2 Experiments

There are two main methods to evaluate a learning scheme: [4]

- Holdout method : This method is used when we have a large dataset. Large independent samples of different data are used for training and large samples are used for testing.

- N-fold stratified cross-validation method : We use this method when we have a limited dataset. This method works as follow: the dataset is divided into N parts, each part is held out in turn for testing and the remaining is used for training learning scheme. Previous tests on numerous different datasets have shown that N=10 is the most appropriate number. In this method, we should employ stratification technique that makes each class properly represented in both training and test sets of each fold.

Our dialog datasets are limited, so 10-fold stratified cross-validation method is used to evaluate DCS performance. However, a single 10-fold stratified cross-validation test might not be enough to get a reliable result, so the standard procedure is to repeat 10-fold stratified cross-validation process 10 times (or 10 iterations). For each iteration, stratified dataset is randomized with seed=1 for 1st iteration, seed=2 for 2nd iteration ..., seed=10 for 10th iteration. Finally, we average the results on these iterations.
The pseudo-code is given in Table 3.8. The whole experiment diagram is displayed in Figure 3.8.

We did several experiments using different set of features. The results and discussion are described in the following chapter.

Table 3-8 Pseudo-code for the experiment

```
stratify dataset (numOfFolds);
For each classifier {
    For each combination of classifier options {
        For seed=1 to numOfIterations {
            randomize stratified dataset (seed);
            do N-fold cross-validation this dataset;
        }
        average the results on these iterations;
    }
    print out the max result of this classifier;
}
select the classifier with the corresponding option that has max result for problematic dialog
identification task;
```

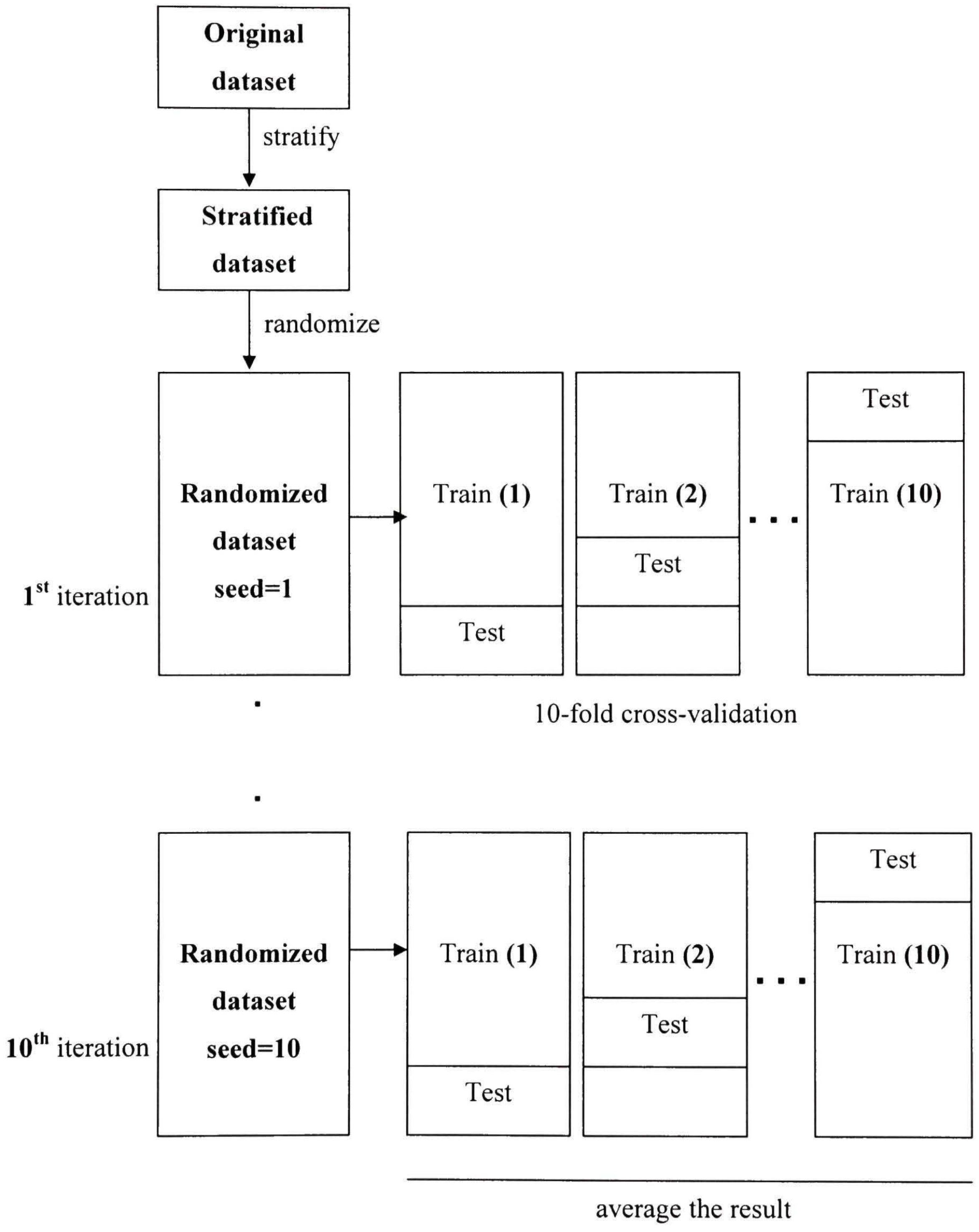


**Figure 3-8** ***Experiment diagram.***

**Chapter Summary**

| | |
|---|---|
| **System Implementation** | _System Design<br>+ Parser Component<br>+ Extractor Component<br>+ Classifier Component<br>_System Development<br>+ Presentation Layer<br>+ Business Logic Layer<br>+ Database Layer |
| **Testing** | _Data Collection and Validation<br>+ DARPA 2000 Communicator Corpus<br>+ DARPA 2001 Communicator Corpus<br>_Experiment: 10 times of 10-fold stratified cross-validation test |

# CHAPITRE 4

# INTERPRETATION

## 4.1 State of the art

In this chapter, we will take a look at the state-of-the-art systems solving the problem of identification of problematic dialog. Since the problem of identification of problematic dialog using user's satisfaction to label bad dialog is a relatively new research topic, we found only one paper specifically related to this problem. Nonetheless, we will review in this section two papers. The first paper is a scientific review article [21] while the other one is a conference paper [23].

At first, we will review the scientific review article [21]. This article is about the comparison of different learning schemes on different tasks. Unfortunately, no task is related to identification of problematic dialog. Nevertheless, we will use their comparisons in order to select a subset of the best learning schemes and apply them on the task of identification of problematic dialog. The conference paper is related to the identification of problematic dialogs. We will use their performance results as a benchmark to achieve. Finally, we will compare our system performance on identification of problematic dialog with the one proposed in [23].

### 4.1.1 Selection of the best learning scheme

In 2003, Niels Landwehr et al [21] tested six learning schemes, namely C4.5 Decision Tree (C4.5), Boosted C4.5 Decision Trees (BC4.5), Logistic Regression Model (LRM), and Logistic Model Tree (LMT), on 32 benchmark datasets from the UCI repository. The results given in Figure 4.1 and Figure 4.2 show that LMT outperforms the other learning schemes on most of the datasets. In particular, LMT outperforms C4.5 on 13 datasets, LRM on 6 datasets, and BC4.5 on 7 datasets. Only BC4.5 outperforms LMT on 6 datasets.

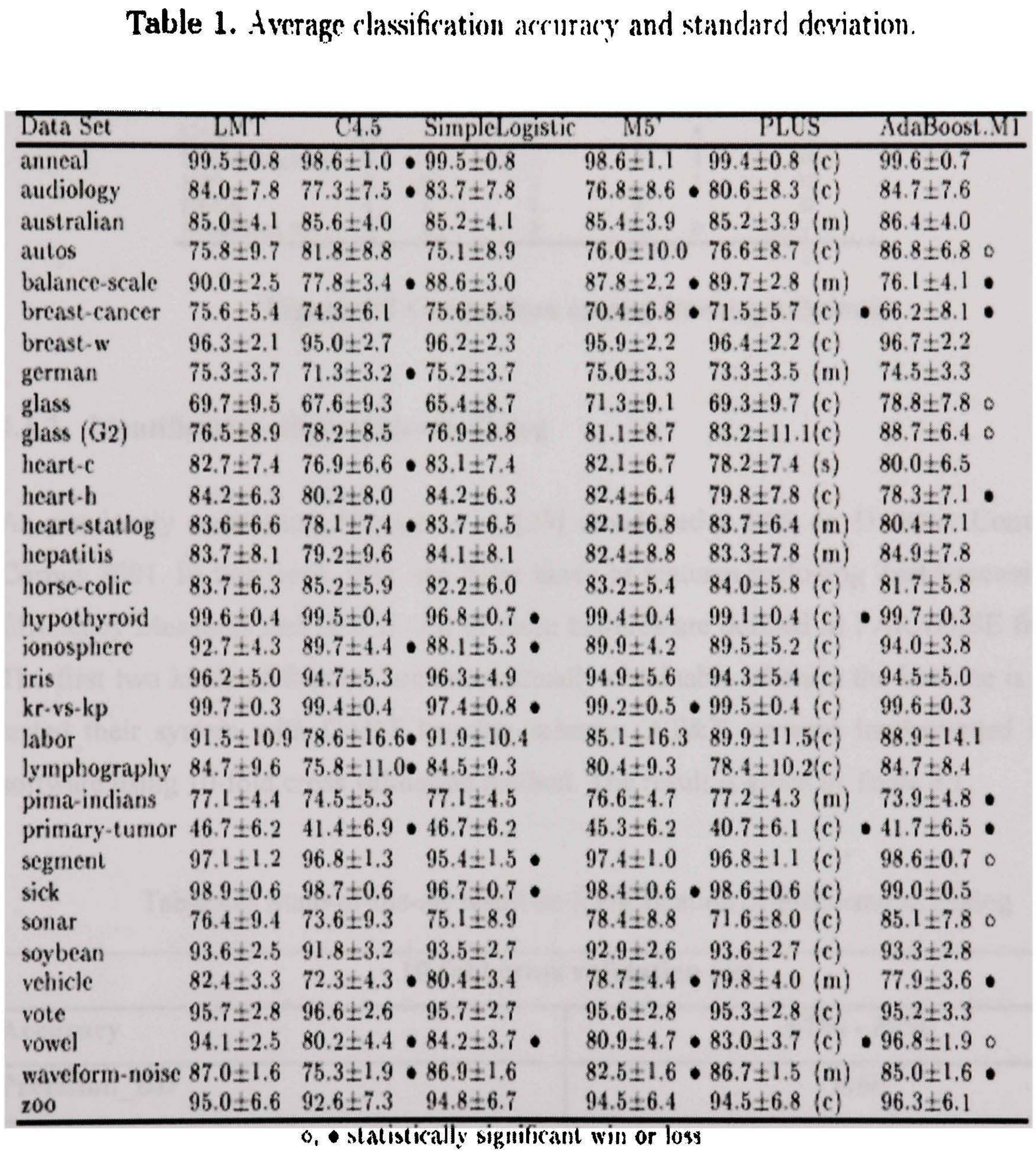


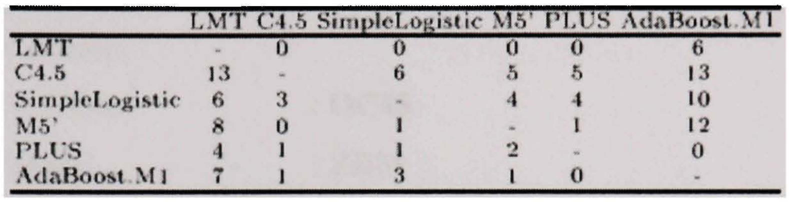
**Table 1.** Average classification accuracy and standard deviation.

| Data Set | LMT | C4.5 | SimpleLogistic | M5' | PLUS | AdaBoost.M1 |
|---|---|---|---|---|---|---|
| anneal | 99.5±0.8 | 98.6±1.0 • | 99.5±0.8 | 98.6±1.1 | 99.4±0.8 (c) | 99.6±0.7 |
| audiology | 84.0±7.8 | 77.3±7.5 • | 83.7±7.8 | 76.8±8.6 • | 80.6±8.3 (c) | 84.7±7.6 |
| australian | 85.0±4.1 | 85.6±4.0 | 85.2±4.1 | 85.4±3.9 | 85.2±3.9 (m) | 86.4±4.0 |
| autos | 75.8±9.7 | 81.8±8.8 | 75.1±8.9 | 76.0±10.0 | 76.6±8.7 (c) | 86.8±6.8 ○ |
| balance-scale | 90.0±2.5 | 77.8±3.4 • | 88.6±3.0 | 87.8±2.2 • | 89.7±2.8 (m) | 76.1±4.1 • |
| breast-cancer | 75.6±5.4 | 74.3±6.1 | 75.6±5.5 | 70.4±6.8 • | 71.5±5.7 (c) • | 66.2±8.1 • |
| breast-w | 96.3±2.1 | 95.0±2.7 | 96.2±2.3 | 95.9±2.2 | 96.4±2.2 (c) | 96.7±2.2 |
| german | 75.3±3.7 | 71.3±3.2 • | 75.2±3.7 | 75.0±3.3 | 73.3±3.5 (m) | 74.5±3.3 |
| glass | 69.7±9.5 | 67.6±9.3 | 65.4±8.7 | 71.3±9.1 | 69.3±9.7 (c) | 78.8±7.8 ○ |
| glass (G2) | 76.5±8.9 | 78.2±8.5 | 76.9±8.8 | 81.1±8.7 | 83.2±11.1(c) | 88.7±6.4 ○ |
| heart-c | 82.7±7.4 | 76.9±6.6 • | 83.1±7.4 | 82.1±6.7 | 78.2±7.4 (s) | 80.0±6.5 |
| heart-h | 84.2±6.3 | 80.2±8.0 | 84.2±6.3 | 82.4±6.4 | 79.8±7.8 (c) | 78.3±7.1 • |
| heart-statlog | 83.6±6.6 | 78.1±7.4 • | 83.7±6.5 | 82.1±6.8 | 83.7±6.4 (m) | 80.4±7.1 |
| hepatitis | 83.7±8.1 | 79.2±9.6 | 84.1±8.1 | 82.4±8.8 | 83.3±7.8 (m) | 84.9±7.8 |
| horse-colic | 83.7±6.3 | 85.2±5.9 | 82.2±6.0 | 83.2±5.4 | 84.0±5.8 (c) | 81.7±5.8 |
| hypothyroid | 99.6±0.4 | 99.5±0.4 | 96.8±0.7 • | 99.4±0.4 | 99.1±0.4 (c) • | 99.7±0.3 |
| ionosphere | 92.7±4.3 | 89.7±4.4 • | 88.1±5.3 • | 89.9±4.2 | 89.5±5.2 (c) | 94.0±3.8 |
| iris | 96.2±5.0 | 94.7±5.3 | 96.3±4.9 | 94.9±5.6 | 94.3±5.4 (c) | 94.5±5.0 |
| kr-vs-kp | 99.7±0.3 | 99.4±0.4 | 97.4±0.8 • | 99.2±0.5 • | 99.5±0.4 (c) | 99.6±0.3 |
| labor | 91.5±10.9 | 78.6±16.6• | 91.9±10.4 | 85.1±16.3 | 89.9±11.5(c) | 88.9±14.1 |
| lymphography | 84.7±9.6 | 75.8±11.0• | 84.5±9.3 | 80.4±9.3 | 78.4±10.2(c) | 84.7±8.4 |
| pima-indians | 77.1±4.4 | 74.5±5.3 | 77.1±4.5 | 76.6±4.7 | 77.2±4.3 (m) | 73.9±4.8 • |
| primary-tumor | 46.7±6.2 | 41.4±6.9 • | 46.7±6.2 | 45.3±6.2 | 40.7±6.1 (c) • | 41.7±6.5 • |
| segment | 97.1±1.2 | 96.8±1.3 | 95.4±1.5 • | 97.4±1.0 | 96.8±1.1 (c) | 98.6±0.7 ○ |
| sick | 98.9±0.6 | 98.7±0.6 | 96.7±0.7 • | 98.4±0.6 • | 98.6±0.6 (c) | 99.0±0.5 |
| sonar | 76.4±9.4 | 73.6±9.3 | 75.1±8.9 | 78.4±8.8 | 71.6±8.0 (c) | 85.1±7.8 ○ |
| soybean | 93.6±2.5 | 91.8±3.2 | 93.5±2.7 | 92.9±2.6 | 93.6±2.7 (c) | 93.3±2.8 |
| vehicle | 82.4±3.3 | 72.3±4.3 • | 80.4±3.4 | 78.7±4.4 • | 79.8±4.0 (m) | 77.9±3.6 • |
| vote | 95.7±2.8 | 96.6±2.6 | 95.7±2.7 | 95.6±2.8 | 95.3±2.8 (c) | 95.2±3.3 |
| vowel | 94.1±2.5 | 80.2±4.4 • | 84.2±3.7 • | 80.9±4.7 • | 83.0±3.7 (c) • | 96.8±1.9 ○ |
| waveform-noise | 87.0±1.6 | 75.3±1.9 • | 86.9±1.6 | 82.5±1.6 • | 86.7±1.5 (m) | 85.0±1.6 • |
| zoo | 95.0±6.6 | 92.6±7.3 | 94.8±6.7 | 94.5±6.4 | 94.5±6.8 (c) | 96.3±6.1 |

○, • statistically significant win or loss

**Figure 4-1** ***Classification result on UCI repository.***

**Table 2.** Number of datasets where algorithm in column significantly outperforms algorithm in row



| | LMT | C4.5 | SimpleLogistic | M5' | PLUS | AdaBoost.M1 |
|---|---|---|---|---|---|---|
| LMT | - | 0 | 0 | 0 | 0 | 6 |
| C4.5 | 13 | - | 6 | 5 | 5 | 13 |
| SimpleLogistic | 6 | 3 | - | 4 | 4 | 10 |
| M5' | 8 | 0 | 1 | - | 1 | 12 |
| PLUS | 4 | 1 | 1 | 2 | - | 0 |
| AdaBoost.M1 | 7 | 1 | 3 | 1 | 0 | - |

**Figure 4-2** ***Comparison among learning schemes.***

### 4.1.2 Identification of problematic dialog

As previously mentioned, Walker et al [23] developed a PDI on DARPA Communicator Corpus 2001. In this work, they use three kinds of features including TaskSuccess Measure, Efficiency Measures and DATE. All of these features are defined in PARADISE framework. The first two kinds of features are automatically obtainable whereas the last one is not. They tested their system with CART learning scheme (AT&T version) implemented in Wagon software using 10-fold cross validation method. The result is given in Table 4.1.

Table 4-1 State-of-the-art result on Identification of Problematic Dialog

| **10-fold cross validation test** | |
|---|---|
| **Accuracy** | **67% - 89%** |
| **Precision_BD** | ***0.66*** |
| **Recall_BD** | ***0.54*** |
| **Fmeasure_BD** | ***0.59*** |

In the next section, we show our results and make a comparison between our system performance and that of the state-of-the-art proposed by [21] and [23].

## 4.2 Result and Interpretation

We perform several experiments on DARPA2000 and DARPA2001 corpora with different set of features. We denote:

| | |
|---|---|
| • Our own C4.5 Tree | : **OC45** |
| • Zero-Rule Model | : **ZRM** |
| • Logistic Regression Model | : **LRM** |
| • One-Rule Algorithm | : **ORA** |
| • C4.5 Tree | : **C45** |
| • Boosted C4.5 Trees | : **BC45** |
| • Logistic Model Tree | : **LMT** |
| • Task Success | : **TS** |
| • Efficiency measures | : **Eff** |
| • NumNegativeACKwords | : **ACK** |
| • NumRepetitions | : **Rep** |

Table 4-2 Experiment Result on DARPA 2000

| **DARPA 2000** (Accuracy / Fmeasure_BD) | | | | |
|---|---|---|---|---|
| | **TS+Eff** | **TS+Eff+ACK** | **TS+Eff+Rep** | **TS+Eff+ACK+Rep** |
| **OC45** | 75.00% / 0.72 | 73.00% / 0.71 | 72.00% / 0.70 | 72.00% / 0.70 |
| **ZRM** | 45.20% / 0.38 | 45.20% / 0.38 | 45.20% / 0.38 | 45.20% / 0.38 |
| **LRM** | 78.28% / 0.74 | 78.32% / 0.74 | 78.96% / 0.75 | 78.60% / 0.75 |
| **ORA** | 77.06% / 0.71 | 77.06% / 0.71 | 77.06% / 0.71 | 77.06% / 0.71 |
| **C45** | 76.35% / 0.71 | 76.20% / 0.71 | 77.69% / 0.75 | 76.87% / 0.74 |
| **BC45** | 73.99% / 0.71 | 72.52% / 0.71 | 73.19% / 0.71 | 73.19% / 0.71 |
| **LMT** | 78.22% / 0.74 | 78.32% / 0. 74 | 78.80% / 0.75 | 78.60% / 0.75 |

Table 4-3 Experiment Result on DARPA 2001

| **DARPA 2001** (Accuracy / Fmeasure_BD) | | | | |
|---|---|---|---|---|
| | **TS+Eff** | **TS+Eff+ACK** | **TS+Eff+Rep** | **TS+Eff+ACK+Rep** |
| **OC45** | 66.00% / 0.60 | 66.00% / 0.60 | 64.00% / 0.60 | 64.00% / 0.60 |
| **ZRM** | 53.86% / 0.00 | 53.86% / 0.00 | 53.86% / 0.00 | 53.86% / 0.00 |
| **LRM** | 69.96% / 0.61 | 69.96% / 0.61 | 69.65% / 0.61 | 69.52% / 0.61 |
| **ORA** | 58.23% / 0.52 | 59.15% / 0.53 | 58.23% / 0.52 | 58.23% / 0.52 |
| **C45** | 68.48% / 0.62 | 68.17% / 0.62 | 68.92% / 0.62 | 68.11% / 0.60 |
| **BC45** | 67.81% / 0.62 | 66.47% / 0.61 | 67.10% / 0.61 | 65.63% / 0.60 |
| **LMT** | 69.91% / 0.61 | 69.96% / 0.61 | 69.45% / 0.61 | 69.52% / 0.61 |

The results given in Table 4.2 and Table 4.3 show that LMT and LRM perform best on DARPA 2000 (~79%/0.75) and 2001 (~70%/0.61). This is the same as the result in previous work [23].

The second article [23] did not mention exactly about the accuracy. They just mentioned that the accuracy varies from 67% to 89%. Our accuracy result is about 70%. Comparing the Fmeasure_BD, we see that our system performs better (0.61 versus 0.59).

However, there are many factors that may cause different results:

1) Kinds of data & data pre-processing.
2) Number and kinds of features used for classification.
3) The way features are used for splitting node (use each feature only once or reuse it)
4) The stop-splitting condition.
5) Heuristics.

In addition, we also look at the deviation in the accuracy results of each method. We consider an experiment with DARPA2000 and set of features are TS+Eff+ACK+Rep. The results of this experiment are displayed in Table 4.4 and deviation graph plotted from this table is shown in Figure 4.3. We see that the deviation of results generated by OC45 is a little bit greater than those of other methods because OC45 is a very basic C4.5 tree we implement to learn how decision tree works. Generally, the deviations of all methods are very small and close together.

Table 4-4 Results of 10-fold CV with DARPA2000 - TS+Eff+ACK+Rep

| Fold | OC45 | ZRM | LRM | ORA | C45 | BC45 | LMT |
|---|---|---|---|---|---|---|---|
| 1 | 76.36% | 44.69% | 79.28% | 76.62% | 76.87% | 72.56% | 79.28% |
| 2 | 67.27% | 44.87% | 78.69% | 77.03% | 76.86% | 72.24% | 78.69% |
| 3 | 78.18% | 45.57% | 78.61% | 77.26% | 76.86% | 73.01% | 78.61% |
| 4 | 69.09% | 45.15% | 78.65% | 77.14% | 76.87% | 73.56% | 78.65% |
| 5 | 63.63% | 44.83% | 78.41% | 77.05% | 76.87% | 73.52% | 78.41% |
| 6 | 78.18% | 45.20% | 78.50% | 77.07% | 76.88% | 73.50% | 78.50% |
| 7 | 72.72% | 45.54% | 78.55% | 77.14% | 76.88% | 73.34% | 78.55% |
| 8 | 76.36% | 45.60% | 78.38% | 77.04% | 76.88% | 73.28% | 78.38% |
| 9 | 70.90% | 45.39% | 78.56% | 77.16% | 76.90% | 73.46% | 78.56% |
| 10 | 67.27% | 45.22% | 78.43% | 77.12% | 76.90% | 73.51% | 78.43% |
| **Average** | **72.00%** | **45.20%** | **78.60%** | **77.06%** | **76.87%** | **73.19%** | **78.60%** |

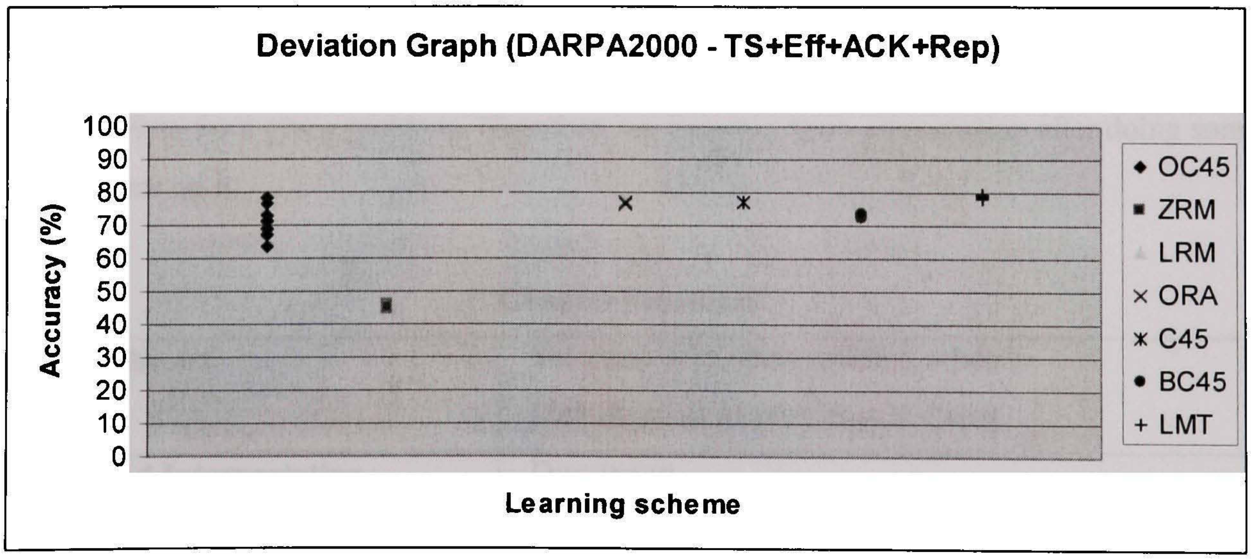


Figure 4-3 *Deviation Graph (DARPA2000 - TS+Eff+ACK+Rep)*

From the obtained results, we can conclude that NumNegativeACKwords and NumRepetitions are two good features because they help improve performance of most of the learning schemes. Particularly, we consider the performance of LMT. In DARPA 2000, NumNegativeACKwords helps improve performance of LMT 0.1%, NumRepetitions 0.58%, and NumNegativeACKwords and NumRepetitions 0.38%. In DARPA 2001, only NumNegativeACKwords helps improve performance of LMT 0.05%.

DCS can be extended easily for new corpora and new learning schemes, i.e. we can add more corpora and learning schemes without changing much the source code. Generally speaking, DCS performance (79% of accuracy / 0.75 of Fmeasure_BD on DARPA 2000 and 70% / 0.62 on DARPA2001) is relatively good for a pattern classification system.

Looking at the Table 4.1, we see that the performance of pattern classification system also depends on dataset itself. For example, for "primary-tumor" dataset, the accuracy of all systems is just about 46.7%, whereas 99.5% accuracy for "anneal" dataset. However, if we have good features, i.e. discriminant features, we can improve system performance.

Therefore, for future work, we could try to find other good features that can improve system performance. Moreover, we could test DCS with other learning schemes than the tree-based ones, such as SVM, kNN. This is one more possibility that we can do. They might give better results or even worse results. As we said before, we do not know in advance which algorithm is appropriate for a given problem. Therefore, we can only have a conclusion after doing some experiments on it.

**Chapter Summary**

| | |
|---|---|
| **State of the art** | _Selection of the best learning scheme<br>_Identification of problematic dialog |
| **Result and Interpretation** | _Discussion<br>_Future work |

# CONCLUSION

Identification of problematic dialog using user's satisfaction to label bad dialog is a relatively new research topic. In this research project, first we have studied this topic based on previous work. After that, we have proposed two new potentially good features and performed some experiments on them to study their effect on the system performance. Our work was motivated by the need of having an automatic system for identifying problematic dialogs in the frame of a practical data mining project, namely "Managing emotions in Human-Computer Dialogs", of ÉTS and CRIM in collaboration with Bell Canada Corp.

The final outcome of our work is the Dialog Classification System (DCS). DCS is a very useful automatic tool for identifying problematic dialogs in Human-Computer Dialog System (HCDS). It is modular and easily extensible in terms of adding new machine learning schemes or new dialog datasets. We can integrate new machine learning schemes or add new corpora into DCS quickly and easily without modifying much the source code. In addition, DCS was developed in a general manner so that any telephone companies using HCDS could employ DCS to evaluate their user's satisfaction in order to propose new strategies for their HCDS.

For the selection of learning scheme, we saw that our results were the same as previous work in that LMT performed best on most of the datasets. This means that when we need to choose a learning scheme for a new dataset in the future, LMT appears to be always the first potentially good choice.

For the identification of problematic dialog, our DCS has outperformed the state-of-the-art system on the DARPA Communicator 2001 corpus.

The performance of a pattern classification system depends not only on dataset but also on the features used for classification. Robust and discriminant features always give good results. Therefore, finding new good features is always the main task in data mining and machine learning field.

We have found two new good features for problem of identification of problematic dialogs, namely negative acknowledgement words (such as NO, NOP) and system repetitions (i.e. number of times that the system said the same utterance) because they helped improve the performance of DCS.

The performance of DCS could be improved if we could find more new good features for it. Moreover, testing DCS with other machine learning schemes such as SVM, kNN could be a possibility because it is very hard to know in advance which machine learning scheme is appropriate for a particular system. That's why the trial-and-error method is always employed in data mining application.

## BIBLIOGRAPHIE

1. École de technologie supérieure. Note de cours. 2007. « *Cadre de Basili* ». In le site du cours "Planification d'un Projet de Recherche en Ingénierie (MTR801)", Professeur Pierre Bourque. En ligne. 4 p. <https://cours.ele.etsmtl.ca/academique/mtr801/Documents-a-acces-restreint/Cadre-de_Basili_H07.pdf>. Consulté 2007.

2. Université du Québec à Montréal. Publications. 1996. « *An example of Basili Framework* ». In le site de l'ÉTS. En ligne. 50 p. <http://www.gelog.etsmtl.ca/publications/pdf/84.pdf>. Consulté 2007.

3. École de technologie supérieure. Note de cours. 2007. « *An example of Basili Framework* ». In le site du cours "Planification d'un Projet de Recherche en Ingénierie (MTR801)", Professeur Pierre Bourque. En ligne. 11 p. <https://cours.ele.etsmtl.ca/academique/mtr801/Documents-a-accesrestreint/Experimentation_in_software_engineering.pdf>. Consulté 2007.

4. Ian H.Witten & Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques,* 2nd edition. The Morgan Kaufmann Series in Data Management Systems Elsevier Inc. 558 p.

5. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The elements of Statistical Learning: Data Mining, Inference, and Prediction,* 1st edition. Springer series in statistics. Springer. 532 p.

6. Richard O.Duda, Peter E.Hart, and David G.Stork. 2001. *Pattern Classification,* 2nd edition. Wiley. 738 p.

7. Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. 2001. *Spoken Language Processing: A guide to theory, algorithm and system development,* 1st edition. Prentice Hall. 965p.

8. Andrew R. Webb. 2002. *Statistical Pattern Recognition,* 2nd edition. QinetiQ Ltd., Malvern, UK. 515 p.

9. Sergios Theodoridis & Konstantinos Koutroumbas. 2003. *Pattern Recognition,* 2nd edition. Elsevier, US. 710 p.

10. Breiman, L., Friedman, J.H., Olshen, R.O. & Stone, C.J. 1984. *Classification and regression trees,* 1st edition. Wadsworth International Group, Belmont, California. 358 p.

11. Universit¨at des Saarlandes. Course note. 2003. « *Linear Models* ». In le site du cours "Connectionist and Statistical Language Processing", Professeur Frank Keller. En ligne. 7 p. <http://homepages.inf.ed.ac.uk/keller/teaching/connectionism/lecture12_4up.pdf>. Consulté 2007.

12. Wikipedia. 2007. *« Decision Tree Learning ».* In le site du Wikipedia. En ligne. <http://en.wikipedia.org/wiki/Decision_tree_learning>. Consulté 2007.

13. Wikipedia. 2007. *« Three-tier architecture».* In le site du Wikipedia. En ligne. <http://en.wikipedia.org/wiki/3-tier#Three-tier_architecture> . Consulté 2007.

14. Data Mining Server. 2001. *« Evaluation of models (discovered knowledge) ».* In le site du DMS. En ligne. <http://dms.irb.hr/tutorial/tut_mod_eval_1.php>. Consulté 2007.

15. Marilyn Walker, Diane Litman, Candace Kamm and Alicia Abella. 1997. « *PARADISE: A framework for evaluating spoken dialog agents* ». In Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics, ACL 97.

16. Marilyn A. Walker, Diane J. Litman, Candace A. Kamm and Alicia Abella. 1998. «*Evaluating Spoken Dialog Agents with PARADISE* ». In Computer Speech and Language, 12-3.

17. WEKA Homepage. <http://www.cs.waikato.ac.nz/~ml/weka>

18. The Linguistic Data Consortium. 2000. *« 2000 Communicator Evaluation ».* In le site du LDC. En ligne. <http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002S56>. Consulté 2007.

19. The Linguistic Data Consortium. 2001. *« 2001 Communicator Evaluation ».* In le site du LDC. En ligne. <http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003S01>. Consulté 2007.

20. Uebersax JS. 2006. *« Likert scales: dispelling the confusion ».* In le site du John S. Uebersax. En ligne. <http://ourworld.compuserve.com/homepages/jsuebersax/likert.htm>. Consulté 2007.

21. Niels Landwehr, Mark Hall, and Eibe Frank. 2003. « *Logistic Model Trees* ». In Proc. of the 14th ECML, pages 241-252.

22. Boufaden, N., Hoang, T. L. et P. Dumouchel. 2007. *«Détection et prédiction de la satisfaction des usagers dans les dialogues personne-machine».* In Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2007), Toulouse, France, 5-8 juin 2007.

23. Helen Wright Hastie, Rashmi Prasad and Marilyn A. Walker. 2002. « *What's the trouble: Automatically Identifying Problematic Dialogs in DARPA Communicator Dialog Systems* ». In Meeting of the Association of Computational Linguistics.

24. Marilyn Walker, Irene Langkilde-Geary, Helen Wright Hastie, Jerry Wright, Allen Gorin. 2001. "*Automatically training a problematic dialogue predictor for a spoken dialogue system*". Journal of Artificial Intelligence Research.

25. Marilyn Walker, Irene Langkilde, Jerry Wright, Alien Gorin, Diane Litman. 2000. "*Learning to predict problematic situations in a spoken dialogue system: experiments with How May I Help You?* ". In North American Meeting of the Association of Computational Linguistics.

26. Irene Langkilde, Marilyn Walker, Jerry Wright, Allen Gorin, Diane Litman. 1999. "*Automatic prediction of problematic human-computer dialogues in 'How May I Help You ?'*". In ASRU99 , To Appear.

27. Diane J. Litman, Marilyn A. Walker and Michael S. Kearns. 1999. "*Automatic detection of poor speech recognition at the dialogue level*". In Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics, ACL99.