ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FUFILLMENT OF THE REQUIREMENTS FOR
A MASTER'S DEGREE IN ELECTRICAL ENGINEERING
M.Eng.

BY
Hoang Cuong TRUONG

INDENTIFYING PROBLEMATIC DIALOG IN A HUMAN-COMPUTER DIALOG SYSTEM

MONTREAL, DECEMBER 10 2010

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Pierre Dumouchel, Thesis supervisor
Professeur titulaire, département de génie logiciel et des TI à l'École de technologie supérieure

Mrs. Narjes Boufaden, Thesis co-supervisor
Professeure associée à l'ÉTS et chercheure au Centre de Recherche Informatique de Montréal

Mrs. Sylvie Ratté, President of the Board of Examiners
Professeure, département de génie logiciel et des TI à l'École de technologie supérieure

Mr. Stéphane Coulombe, Manager Structural Evaluation
Professeur titulaire, département de génie logiciel et des TI à l'École de technologie supérieure

THIS THESIS HAS BEEN PRESENTED AND DEFENDED

BEFORE A BOARD OF EXAMINERS AND PUBLIC

December 3 2010

AT 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 son aide.

Je voudrais également remercier Madame Narjes Boufaden, ma co-directrice, professeure associée à l'ÉTS et chercheure au CRIM, pour ses conseils et ses recommandations productives.

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.

IDENTIFICATION DES DIALOGUES PROLÉMATIQUES DANS UN SYSTÈME DE PERSONNE-MACHINE

Hoang Cuong TRUONG

RÉSUMÉ

Dans ce mémoire, nous présentons le développement d'un système automatique qui permet d'identifier les dialogues problématiques dans le contexte d'un système de dialogue personne-machine. Le système que nous développons est un type d'application dans le domaine de Reconnaissance de formes et inspection (Pattern Classification). Dans ce travail, nous proposons une approche probabiliste qui permet de prédire la satisfaction de l'usager à chaque tour de parole dans un dialogue parlé. Pour ce faire, toutes les caractéristiques utilisées dans notre système sont automatiquement extraites de l'énoncé. Un modèle de Markov caché (HMM) est utilisé pour construire notre système. Afin d'évaluer la performance du système, nous faisons l'expérimentation sur deux corpus publiquement distribués par le Linguistic Data Consortium (DARPA Communicator 2000 et DARPA Communicator 2001). La validation croisée est utilisée comme méthode d'évaluation. Nos résultats montrent que le système pourrait être appliqué à des problématiques réelles.

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

IDENTIFYING PROBLEMATIC DIALOGS IN A HUMAN-COMPUTER DIALOG SYSTEM

Hoang Cuong TRUONG

ABSTRACT

In this thesis, we present the development of an automatic system that identifies problematic dialogues in the context of a Human-Computer Dialog System (HCDS). The system we developed is a type of application in Pattern Classification domain. In this work, we propose a probabilistic approach that predicts user satisfaction for each turn of dialogue. To do so, all the features used in our system are automatically extracted from the utterance. A robust and fast machine learning scheme, Hidden Markov Model (HMM) is used to build our desired system. In order to evaluate the system performance, we experimented on two publicly distributed corpora: DARPA Communicator 2000 and 2001. We evaluated the system using a 10-fold stratified cross-validation. Our results show that the system could be used in real life applications.

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

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

ASR	Automatic Speech Recognition
CART	Classification and Regression Tree
CRIM	Centre de recherche informatique de Montréal
DARPA	Defense Advanced Research Projects Agency
DCS	Dialog Classification System
DM	Dialog Manager
DT	Decision Tree
ÉTS	École de technologie supérieure
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HMIHY	How May I Help You
HTK	Hidden Markov Model ToolKit
kNN	k-Nearest-Neighbour
LOOCV	Leave-one-out Cross Validation
NLU	Natural Language Understanding
PARADISE	PARAdigm for DIalog System Evaluation
PDI	Problematic Dialog Identifier
PR	Pattern Recognition
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
SLIPPER	Simple Learner with Iterative Pruning to Produce Error Reduction

SLU	Spoken Language Understanding
SVM	Support Vector Machine
WEKA	Waikato Environment for Knowledge Analysis

CHAPTER 1

INTRODUCTION

1.1 Problem

In commercial world, everyone knows that the customer satisfaction is a key element to ensure the survival and development of a company. So many solutions were proposed to assure the customer satisfaction. One of these proposals is the use of call centers to support the customer's demands and especially those who need a remote assistance. Unfortunately, there are many companies whose call centers have to process thousands of phone calls per day and are required to operate 24/7. Therefore, dialog systems in which a machine agent instead of a human operator would answer the incoming calls were suggested as a solution for this kind of problem. However, these systems have their own limitations which usually cause frustration and annoyance for the customer. Such limitations are performance errors related to the natural language understanding (NLU) component and the automatic speech recognition (ASR) component. Thus, there is a question raised that is how we treat these limitations to improve the quality of such systems.

In our research, we're interested in automatically detecting problematic dialogs in real call centers. A dialog is considered problematic if the customer falls into one of two following situations (Langkilde et al., 2000; Hastie et al., 2002):

- He fails in doing the tasks he desires.
- His desired task is successful but he isn't satisfied with his interaction with the machine agent.

Premature detection of a problematic dialog allows us to have various strategies for the management of the failure dialog:

- Process the call as a priority by redirecting to a human operator in case the problem persists.

- Recall the customers who prematurely ended the telephone dialog in order to reduce their unsatisfaction.
- Apologize and appease the customer by a soft human voice.

Our problem is a part of the project named "Managing emotions in Human-Computer Dialogs" which aims to enhance Human-Computer dialog system capabilities to maximize user satisfaction. This is a project of ÉTS and CRIM in collaboration with Bell Canada Corp. The project includes two main parts (Figure 1.1), namely Emotion Detection and Dialog Classification.

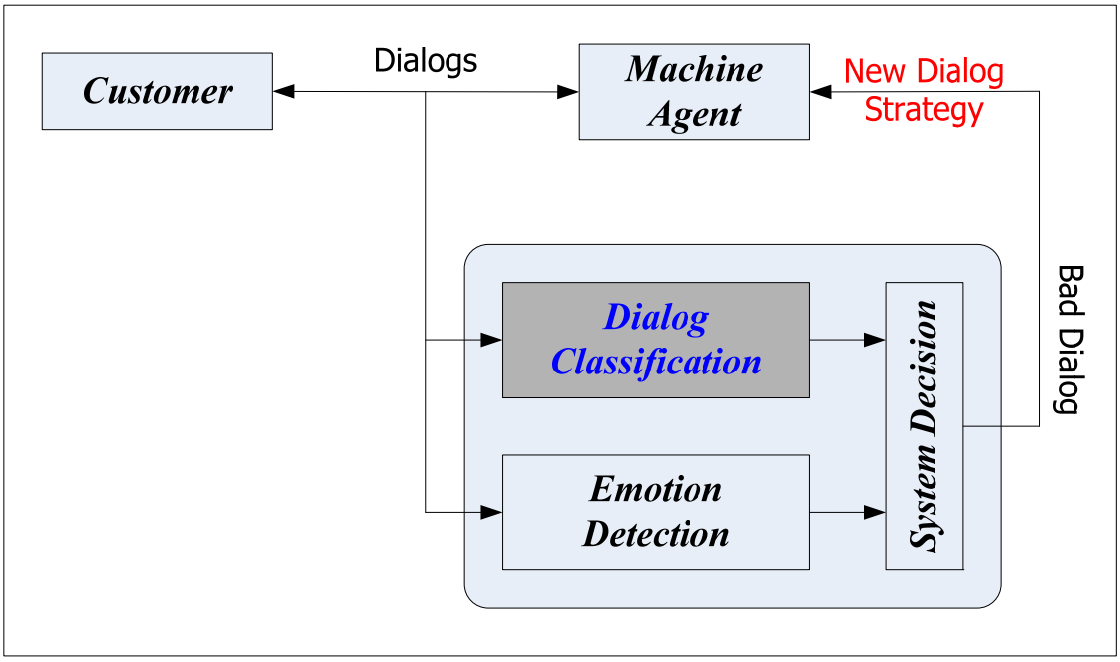


Figure 1.1 Human-Computer Dialog System.

1.2 Objective

In the previous work of our colleague (Truong Le Hoang, 2008), he introduced an approach to automatically classify a spoken human-machine (computer) dialog into problematic dialog (BAD dialog) and non-problematic (GOOD dialog). The approach is based on observing whole-dialog-level information (such as TaskSuccess, TimeOnTask, TurnOnTask…) which

will be described later in the next chapter. His results proved to be useful and could be used in a real application of automatic human-machine dialog classification.

In this thesis, we also have the same purpose that is to automatically classify a spoken human-machine dialog. However, our work will differ from his in two aspects. First, as input of our system, we will work with information extracted at the utterance level instead of the dialog level. By doing so, our system will have the possibility to react more quickly to problematic dialog. Second, we will use a different classification paradigm i.e. Hidden Markov Model (HMM) instead of decision trees (Truong Le Hoang, 2008) to model the dependencies between successive turns.

Our thesis is structured as follows. Chapter 1 is an introduction which depicts the problem and our objectives in this work. Chapter 2 will give some definitions concerning the state of art of the problem and theories used to solve the problem. Chapter 3 describes the methodology used in this project. Chapter 4 demonstrates the design and the implementation of our system. And then, Chapter 5 describes the results and their interpretation. Finally, the last section will be the conclusion and the future work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we introduce a theoretical framework, named PARADISE (Walker et al., 1997), which describes the parameters that could be used to evaluate a spoken dialog. Then, we summarize the latest work related to dialog classification and present the state of art of our problem. We then briefly review Hidden Markov Models and the main algorithms that will be used in our solution to the problem.

2.1 Theoretical framework

PARADISE which stands for ***PARA***digm for ***DI***alog ***S***ystem ***E***valuation is a general framework whose objective is to evaluate spoken dialog agents (in this section, spoken dialog agents indicate human-machine dialog systems). This is a result of the work which was published by Marilyn A. Walker, Diane J. Litman and their colleagues (Walker et al., 1997). This framework has been used to support comparisons among the dialog strategies and allows the calculation of performance over dialogs as well.

The PARADISE model suggests that dialog performance correlates with external criterion which can be measured by user satisfaction. In other words, we can evaluate spoken dialog agents using user satisfaction measurement. The model further suggests that user satisfaction can be measured through two types of factors called “task success” and “dialog costs”. The latest one consists of dialog efficiency and dialog qualitative. Figure 2.1 illustrates PARADISE model.

According to PARADISE, user satisfaction maximization includes both task success maximization and dialog costs minimization. To do so, all the factors have appropriate measures which are:

- Task success measure: Kappa coefficient (Carletta, 1996; Siegel and Castellan, 1988) is used to measure the agreement between two individuals. In PARADISE, the coefficient is calculated from the confusion matrix which summarizes how well a machine agent collects the required information of a particular task.

- Efficiency measure: is calculated from the following feature: number of utterances, dialog time, number of time user and agents talk at the same time, average duration of system turns, and average duration of user turns.

- Qualitative measure: is evaluated through features which could be: agent response delay time, inappropriate utterance ratio, agent repair ratio.

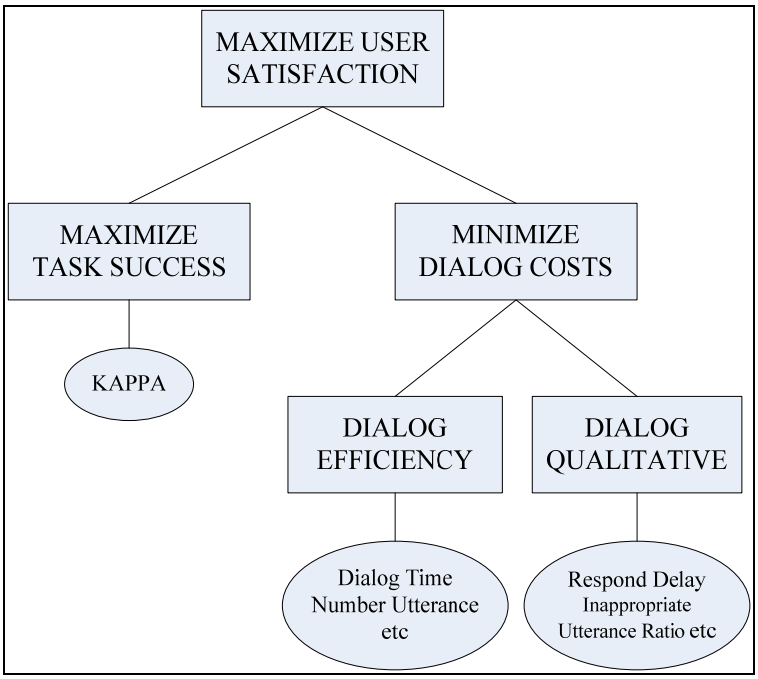


Figure 2.1 PARADISE's structure of objectives for spoken dialog performance.
From Walker (1997)

2.2 Related works on dialog classification

This section will summarize previous works related to classification of telephonic human-

machine dialogs. To do so, we will review several papers chronologically.

Firstly, Litman, Walker and Kearns published a paper (Litman et al., 1999) titled “Automatic Detection of Poor Speech Recognition at the Dialogue Level”, in which they showed that a “good” or “bad” dialog is a result of “good” or “bad” speech recognition performance. Their work was carried on a corpus of 544 dialogs collected from three different systems, namely ANNIE (an agent for voice dialing and messaging), ELVIS (an agent for accessing email) and TOOT (an agent for accessing online train schedules). A set of 23 features based on five types of knowledge sources: Acoustic Features, Dialogue Efficiency Features, Dialogue Quality Features, Experimental Parameter Features and Lexical Features were extracted directly from system logs. Finally, they adopted the learning machine program RIPPER in order to build a classification model which was tested by the method of cross-validation with 25 folds. RIPPER stands for **R**epeated **I**ncremental **P**runing to **P**roduce **E**rror **R**eduction. RIPPER (like other learning programs e.g., C5.0 and CART) is a fast and efficient rule learning system described in more detail in (Cohen, 1995) and (Cohen, 1996). The best result achieved was 77.40% on accuracy rate. They also did several experiments using a subset of features or primitive features to evaluate the features’ role through classification performance.

In the same year, a group from AT&T Labs-Research published a paper (Langkidle et al., 1999) which addressed the automatic prediction of problematic Human-Computer dialogues with their system ‘How May I Help You?’ HMIHY is a spoken dialogue system for customer care at AT&T Labs that provided them a corpus of 4774 dialogues used in their experiments. In this work, Langkidle defined problematic dialogue as a dialog where a customer is unable to complete his desired task. Their feature set was derived from four different sources: ASR component, NLU component, Dialog Manager Component and Hand-Labeled component. They also had two feature sets which are subset of the original set. The first one, called Automatic features, includes all features described above except the Hand-Labeled features and the reverse-order-utterance-id (among the Dialog Manager features). The second subset, called Auto & Task-Independent, keeps only features that are available at runtime and independent of the HMIHY task. They took the machine learning program RIPPER to build a

classification model and used a 5-fold cross-validation in order to evaluate their system's performance. Since they were interested in predicting problematic dialogues, they used only features extracted from exchange 1 or exchange 1 and exchange 2 in the dialogue in order to test their system. Their result is shown in the below table.

Table 2.1 Result from the work done by Langkidle in 1999

	Features Used	**Accuracy**
Exchange 1	Auto features	72.3%
	Auto & Task-Independent features	71.6%
Exchange 1&2	Auto features	79.8%
	Auto & Task-Independent features	78.4%
Full dialog	Auto features	87.0%
	Auto & Task-Independent features	86.7%
	Full features	88.5%

One year later, (Walker et al., 2000) continued to enhance the previous work from Langkidle by adding a new hand-labeled feature, named 'rsuccess'. This feature was used to verify whether the NLU module correctly identifies the task that user was asking HMIHY to perform. Their system's performance was improved by 4% of accuracy (92.3% in comparison with 88.5%) but wasn't useful for real-time application since the new feature is a hand-labeled one.

Then, (Walker et al., 2001) proposed a method to automatically generate the hand-labeled feature 'rsuccess' (auto-SLU-success). Their results are shown in Table 2.2.

Table 2.2 Accuracy % results of Walker in 2001

Features	Exchange 1	Exchange 1&2	Whole dialog
Baseline	67.10	67.10	67.10
Auto (no auto-SLU-success)	70.10	78.10	87.00
Auto + auto-SLU-success	69.60	79.20	84.90
Auto, Task-Independent (no auto-SLU-success)	70.10	78.40	83.40
Auto, Task-Independent + auto-SLU-success	69.20	80.30	85.40
Auto + SLU-success	75.60	85.70	92.90
ALL (Auto + Hand labeled)	77.10	86.90	91.70

In 2007, Narjes Boufaden, Truong Le Hoang and Pierre Dumouchel published a paper (Boufaden et al., 2007) in which they studied the detection and prediction of user satisfaction in Human-Machine spoken dialogs. The features used in their work are a combination of Dialog Efficiency features with two new features, Named Entities and Acknowledgement Words respectively. Three learning machine algorithms including Support Vector Machine, k-Nearest-Neighbour and Decision Tree were applied in this work. Their experiments were tested on the corpus of DARPA 2001 Communicator using the method of 10-fold cross validation. Their results (Table 2.3) show that Named Entities and Acknowledgment Words are a good indicator for prediction of user satisfaction in the beginning of the dialog and they also improved baseline classification performance.

Table 2.3 Result from the work of Boufaden in 2007

Features used	SVM	kNN	DT
Baseline	50.00%	50.00%	50.00%
Dialog Efficiency	61.26%	91.41%	85.75%
Dialog Efficiency + Name Entities + Acknowledgement Words	63.51%	91.80%	87.26%

In 2008, (Schmitt et al., 2008) employed an alternative machine learning method, SLIPPER, which stands for **S**imple **L**earner with **I**terative **P**runing to **P**roduce **E**rror **R**eduction. Their corpus comprises 69,296 calls from a commercially deployed recent call center recorded between Dec 3rd, 2007 and Dec 14th, 2007. The feature set is made up by ASR features, NLU features and Dialog Manager features. In this work, they achieved a very good result which shows that their model can identify problematic calls after only five caller turns with an accuracy of over 90%.

2.2.1 Related works experimented on DARPA 2000 & 2001 Corpora

The comparison of the systems presented earlier is made difficult by the fact that researchers do not use the same data in their study. Therefore, in this subsection, we will review one paper and one thesis which both did the research of spoken dialog classification on the same corpus as ours. We're thus able to compare their results with ours and evaluate our achievement as well.

The first one is a paper by Helen Wright (Helen Wright Hastie et al., 2002) which developed a problematic dialogue identification system and tested it on DARPA 2001 Communicator Corpus. The corpus is composed of 1242 dialogues. All kinds of the features, used in this work, which are defined in PARADISE framework, consist of TaskSuccess Measure, Efficiency Measures and DATE. The two latter kinds of features are automatically extracted, whereas the former is obtained by two methods, namely hand-labelled and automatic. The learning scheme they applied is CART (Classification and Regression Tree) implemented by Wagon software. Their results are given in the following table:

Table 2.4 State of the art results from Hastie in 2002

Task Completion	Dialogue	Recall	Precision	Fmeasure
Hand-labelled	Good	90.00%	84.50%	87.16%
Hand-labelled	Bad (Problematic)	54.50%	66.70%	59.98%
Automatic	Good	88.50%	81.30%	84.74%
Automatic	Bad (Problematic)	66.70%	58.50%	62.33%

The second work presented for comparison is a master thesis (Truong Le Hoang, 2008) which also developed software whose main goal is to automatically identify problematic dialogs. In this work, he used exactly the same corpora as ours (DARPA Communicator 2000 and 2001) in order to test his system performance. The feature set that he took is a combination of Task Success measure, Efficiency measures with two new features, namely NumNegativeACKwords (that number of negative words: NO, NOP, FALSE, INCORRECT, WRONG, ERASE in a dialog) and NumRepetitions (that number of times the agent repeats the same utterance). All the features are automatically extractable. Decision Tree is the learning scheme he chose to build his system. Of many different decision tree algorithms, he developed the Basic C4.5 Tree by himself. The other learning algorithms presented in Table 2.5 are from the library WEKA (Waikato Environment for Knowledge Analysis) which was developed by many machine learning experts from the University of Waikato. The table below shows his best results.

Table 2.5 State of the art result from the master thesis of Truong Le Hoang in 2008

Decision Tree Algorithm	DARPA 2000		DARPA 2001	
	Accuracy	Fmeasure on BAD	Accuracy	Fmeasure on BAD
His own C4.5 Tree	72.00%	70.00%	64.00%	60.00%
Logistic Regression Model	78.60%	75.00%	69.52%	61.00%
One-Rule Algorithm	77.06%	71.00%	58.23%	52.00%
C4.5 Tree	76.87%	74.00%	68.11%	60.00%
Boosted C4.5 Trees	73.19%	71.00%	65.63%	60.00%
Logistic Model Tree	78.60%	75.00%	69.52%	61.00%

Through reviewing seven articles and one master thesis concerning the problem of telephonic human-machine dialogue classification, we found out the following points:

- Most of the works used the rule learning machine schemes and their variants such as: RIPPER, SLIPPER, and Decision Trees except for Boufaden et al,. who tried two other learning machine methods: k-NN and SVM.

- The features used for all the works are extracted from three knowledge sources: Automatic Speech Recognition (ASR) component, Natural Language Understanding (NLU) component and Dialog Manager (DM) component. Most of the features are automatically extractable while a few are hand-labeled. The features are mostly calculated based on the whole dialog level.

- Although the works were tested on many different corpora, their results are on range of 70%-85% of accuracy under the condition that we only consider the results experimented with the automatically extractable features.

Therefore, in this thesis, we not only address to build a dialog classification system using new approaches as described in section 1.2 but also try to reach the performance of the state of art.

2.3 Pattern Classification

Nowadays, information technology becomes very popular and there are a lot of applications in most of the fields such as science and society also. Many of these technologies that were and are studied and developed rapidly in the laboratory in general and the industry in particular are the ones based on machine learning techniques.

2.3.1 Pattern Classification Concepts

Pattern classification (or also known as Pattern recognition) is "the act of taking in raw data taking an action based on the category of the pattern" (Duda, 1999). This definition sounds pretty abstract and somewhat difficult to understand. But before explaining pattern classification more simply and in easy-to-understand way we should know several common terminologies used in pattern classification as follow:

- Pattern: according to the Japanese professor, Satoshi Watanabe (Watanabe, 1985): "*A pattern is opposite of a chaos: it is an entity vaguely defined, that could be given a name*". We can also understand that a pattern could be a process, an event or an abstract object, such as a set of measurements describing a physical object. Figure 2.2 shows some examples of patterns.

- Feature (or attribute): is an intrinsic trait or characteristic of a pattern that we can use to discriminate a pattern from another one. There are generally two kinds of features including nominal feature (e.g. windy, rainy and so on) and numeric feature (e.g. length, age, weight and so on).
- Class or pattern class is a set of patterns that has the same set of common features.
- Training set (or labeled set) is a set of patterns that have been classified or described. Recognition systems usually learn how to recognize a pattern based on these training set.

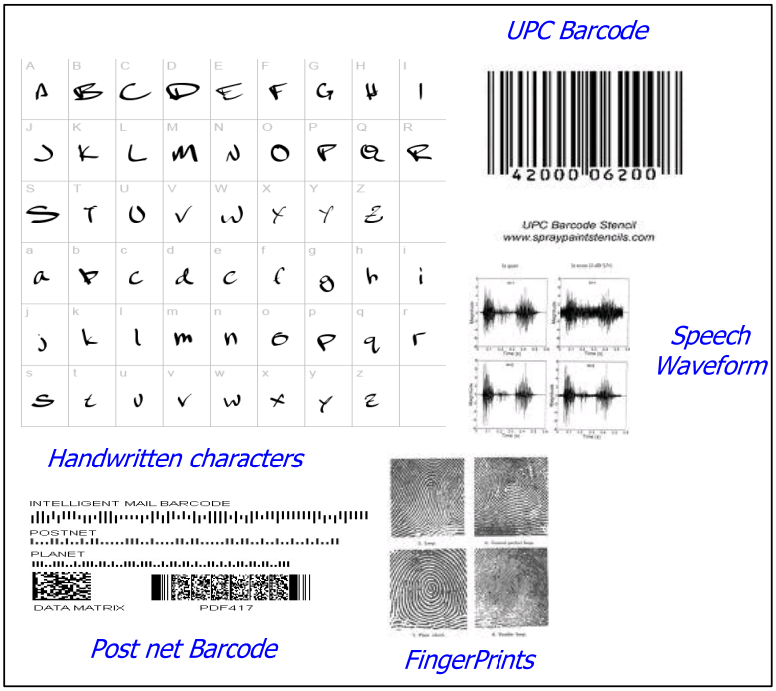


Figure 2.2 Examples of patterns.

Pattern classification generally is a process (illustrated as Figure 2.3) whose principal goal is to classify objects (or patterns) into categories (or classes) based on features extracted from the patterns.

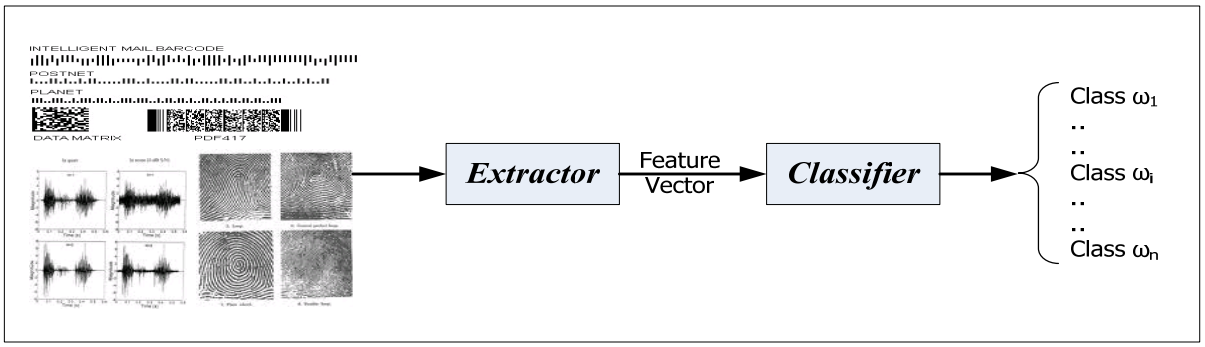


Figure 2.3 Concept of pattern classification.

Pattern recognition is divided into two main methods: supervised learning and unsupervised learning. Supervised learning is a problem in which classes are known beforehand and data samples (or called training data) of each class are also available. Unsupervised learning is a

problem in which we don't have any information about classes as well as number of classes before solving such a problem so that we must infer the information from the data.

A complete pattern classification system usually includes five components (Figure 2.3) in which each part undertake a specific task.

- Sensor is the first component of a pattern recognition system. The only duty of this component is to gather the pattern's observations needed in recognition processing. Depending on the object that the system wants to identify, the observations could be images, digital signals, or waveforms. Thus, a corresponding sensor could be a camera or a signal recording device.

- Pre-processing is used to refine observations collected in the previous phase. This also helps our system to reduce noises that we can meet while collecting data about the pattern. For specific problems concerning image processing, the goal of pre-processing becomes solving the typical sub-problems such as image digitization or segmentation and so on.

- Feature extraction receives data collected and adjusted by the two previous components to create a set of feature values (or called a feature vector).

- Classifier is the most important and essential component of a PR system. We thus need to determine the most suitable PR method among many different ones (e.g. hidden Markov model, neural network, support vector machine or Bayesian decision theory). Building a classifier consists of optimizing its parameters on a training dataset and testing it on a distinct testing dataset.

- System evaluation is an indispensable component of the PR systems. This component evaluates the overall performance of the system and helps us seek effective methods to improve the performance.

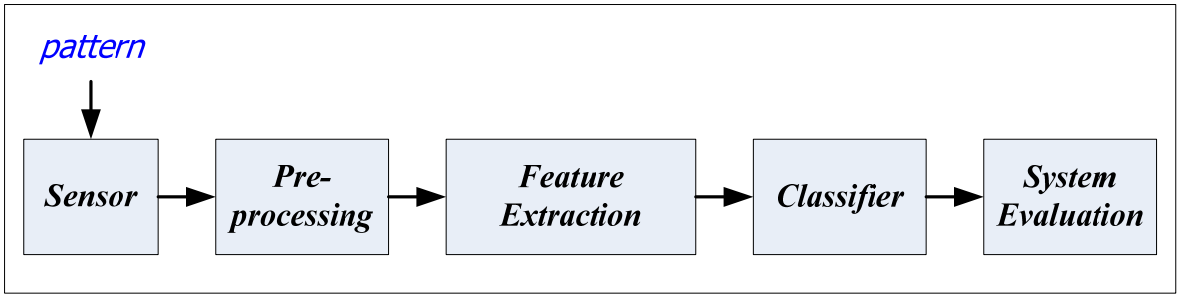


Figure 2.4 Main components of a pattern classification system.

2.4 Hidden Markov Model

HMM (Hidden Markov Model) is a statistical model which can model best the observed data samples of a discrete-time series. HMM has been frequently applied for pattern classification of time-varying data sequences. Thus, HMM is used in different domains such as speech, handwriting, and gesture recognition and gene prediction (Huang et al., 2002).

In the next subsection, we will describe how HMM works and how we train it.

2.4.1 Markov chain (Huang et al., 2002)

HMM is derived from Markov chain. This chain is a discrete random process which has the property that the next state is only dependent on the current state. Andrei Markov, a Russian mathematician, whose best known work is the theory of stochastic processes, invented the Markov chain.

To start the Markov chain, we firstly define $X = X_1, X_2, ...X_n$ as a sequence of random variables and according to the Bayes rule, we have

$$P(X_1, X_2, ..., X_n) = P(X_1)\prod_{i=2}^{n} P(X_i \mid X_1^{i-1}) \tag{2.1}$$

where $X_1^{i-1} = X_1, X_2, ...X_{i-1}$. In the Markov model chain, we have an important assumption (called *Markov assumption*) which states: the probability of the random variable at a given

time depends only on the value at the preceding time. This assumption means that:

$$P(X_i \mid X_1^{i-1}) = P(X_i \mid X_{i-1}) \tag{2.2}$$

Apply the assumption to equation (2.1), we have

$$P(X_1, X_2, ..., X_n) = P(X_1)\prod_{i=2}^{n} P(X_i \mid X_{i-1}) \tag{2.3}$$

Equation (2.3) gives us an idea that we can use the Markov chain to model time-invariant events by ignoring the time index *i*, and considering:

$$P(X_i = s \mid X_{i-1} = s') = P(s \mid s') \tag{2.4}$$

Now, if we assign X_i to a state, the Markov chain becomes a finite state process with transition between states specified by the probability function *P(s|s').* Of course, the Markov *assumption* mentioned in the equation (2.2) is restated to the following: the probability that a Markov chain will be in a particular state at a given time depends only on the state of the Markov chain at the previous time.

Generally, a Markov chain having N distinct states namely {*1,...,N*} fully includes the following parameters (note that the state at time *t* in the Markov chain denoted as s_t):

$$a_{ij} = P(s_t = j \mid s_{t-1} = i) \qquad 1 \le i, j \le N \tag{2.5}$$

$$\pi_i = P(s_1 = i) \qquad 1 \le i \le N \tag{2.6}$$

where:

- a_{ij} is the transition probability from state *i* to state *j*
- π_i is the initial probability that the Markov chain will start in state *i*

Of course, both of parameters must satisfy the basic rule of probability function as below:

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad 1 \le i \le N$$
$$\sum_{j}^{N} \pi_j = 1 \tag{2.7}$$

Let's consider a specific example so as to understand much more about Markov chain. Thereby, we explain why the Markov chain above is also called the observable Markov. This example is a simple three-state Markov chain used to describe the Dow Jones Industrial average. In this model, each state represents a situation of the Dow Jones average at the end of a day which can be one of the following states:

- State 1 – up (in comparison to the index of previous day)
- State 2 – down (in comparison to the index of previous day)
- State 3 – unchanged (in comparison to the index of previous day)

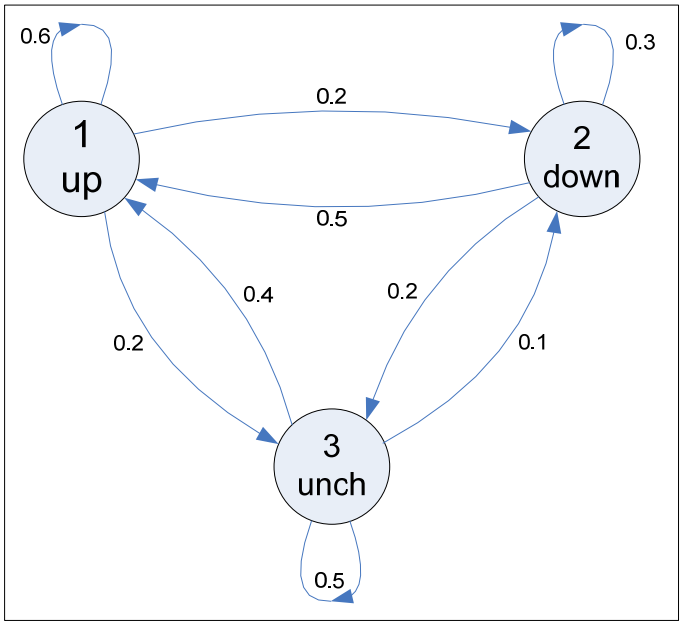


Figure 2.5 Markov chain for the Dow Jones Industrial average.
From Huang (2002, p. 377)

The parameter set for this Dow Jones Markov chain will be:

- A transition probability matrix $A = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$

- An initial probability matrix $\mu = \{\mu_i\} = [0.5 \quad 0.2 \quad 0.3]$

The output of the process at each time instance *t* is a specific event which we can deterministically observe. It means that for each observable event sequence $X = X_1, X_2, ...X_n$, we always determine a correspondent Markov chain state sequence $S = S_1, S_2, ...S_n$. Example: A sequence of events "*up-up-down-unchanged-down*" of Dow Jones has a correspondent sequence of states "S_1-S_1-S_2-S_3-S_2" and the probability will be

P("*up-up-down-unchanged-down*") = P(S_1,S_1,S_2,S_3,S_2)

$$= \mu_1 a_{11} a_{12} a_{23} a_{32} = 0.5 \times 0.6 \times 0.2 \times 0.2 \times 0.1 = 0.0012$$

2.4.2 Hidden Markov Model

In the previous section, each state of the Markov chain corresponds to a deterministically observable event. Before advancing to the definition of the Hidden Markov Model, we should first take a look at Figure 2.6 which is also used to predict the variance of the Dow Jones Industrial average. In comparison with the Markov chain (Figure 2.5), there is a difference that each state doesn't represents a fixed event anymore. In this new model, the output of a state is hidden and depends on the probabilistic function of a correspondent state. In other words, a state in HMM could rather be in one of three modes (up, down and unchanged) based following a probability density function. That's why we call this model Hidden Markov Model which can be viewed as a double-embedded stochastic process with an underlying stochastic process not directly observable.

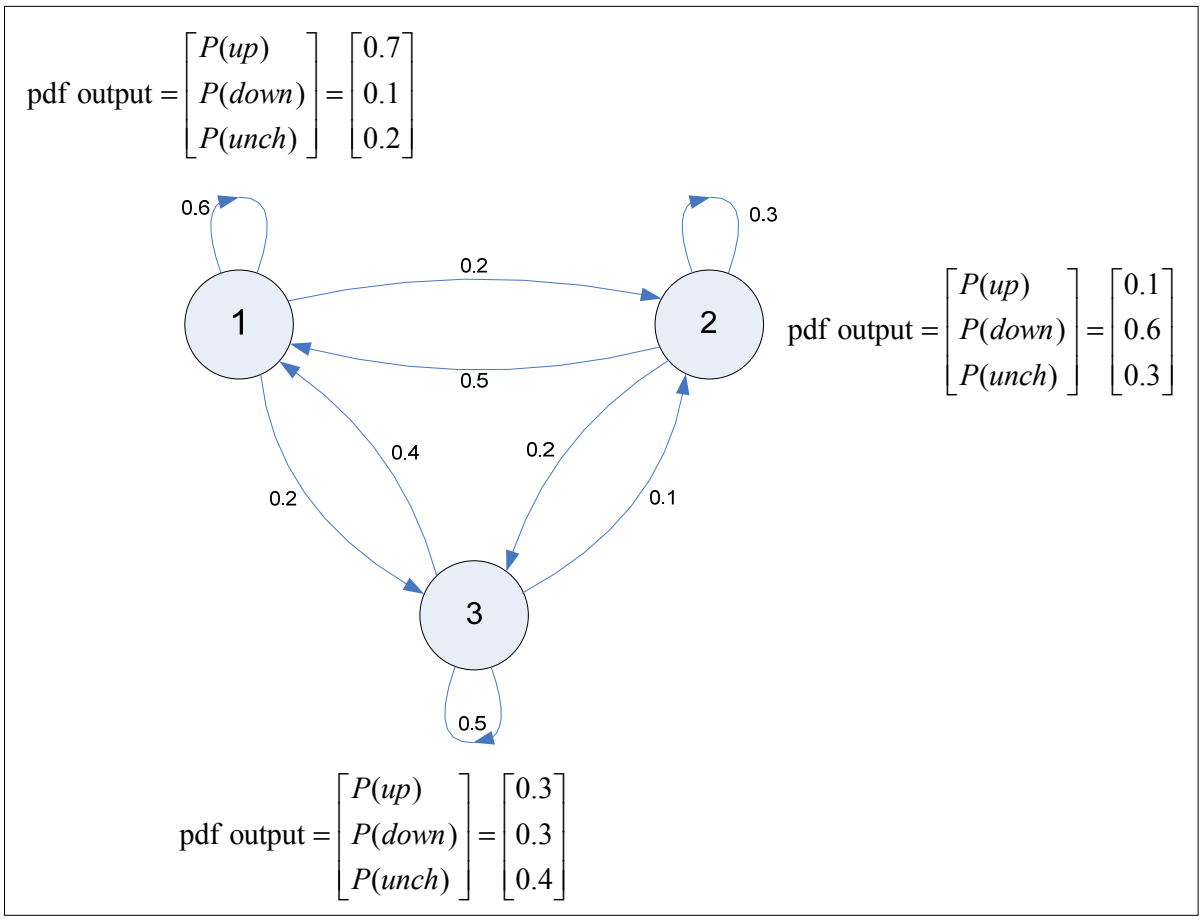


Figure 2.6 Hidden Markov model for the Dow Jones Industrial average.
From Huang (2002, p. 378)

Generally, a hidden Markov model is defined by a set of parameters as below:

- A set of output observations $O = \{o_1, o_2, ..., o_M\}$.
- A set of states $\Omega = \{1,2,..., N\}$.
- A transition probability matrix $A = \{a_{ij}\}$, where a_{ij} is the probability of transiting from state i to state j.
- An output probability matrix $B = \{b_i(k)\}$, where $b_i(k)$ is the probability of emitting symbol o_k at the state i.

- An initial state distribution $\pi = \{\pi_i\}$.

Since a_{ij}, $b_{ij}(k)$ and π_i are all probabilities, they must satisfy the following conditions:

$$a_{ij} \geq 0, \qquad b_i(k) \geq 0, \qquad \pi_i \geq 0 \qquad \forall i, j, k \tag{2.8}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{2.9}$$

$$\sum_{k=1}^{M} b_i(k) = 1 \tag{2.10}$$

$$\sum_{i=1}^{N} \pi_i = 1 \tag{2.11}$$

From now on, note that we will use either the notation $\Phi = \{A, B, \pi\}$ or the symbol Φ only to indicate the whole parameter set of an HMM to avoid ambiguity.

2.4.3 Three basic problems of HMM (Rabiner, 1989)

In order to apply HMM to real-world applications, we firstly need to address three basic problems of HMM and clearly understand their solutions as well:

1. The Evaluation Problem: Given a model Φ and a sequence of observations $X = (X_1, X_2, ..., X_T)$, how to compute the probability $P(X|\Phi)$ that generates the observations?

2. The Decoding Problem: Given a model Φ and a sequence of observations $X = (X_1, X_2, ..., X_T)$, how to find out the best state sequence $S = (s_1, s_2, ..., s_T)$ in the model that generates the corresponding observation sequence?

3. The Learning Problem: Given a model Φ and a set of observation sequences (training data), how to adjust the HMM parameter set to maximize the joint likelihood

probability $\prod_{X} P(X|\Phi)$?

The solutions to these problems are described in annexes I, II, III and IV. In the next subsection, we will briefly mention a tool which implements the HMM algorithms very efficiently.

2.5 HTK Overview

HTK which stands for ***H****idden Markov Model* ***T****ool****k****it* is a tool popularly used to build and manipulate hidden Markov models. The principal idea of HMM is to model any time series event and the core of HTK also have same general purpose. Firstly, HTK was created to address the demands of speech recognition research and a lot of infrastructures in HTK are designed for this task. However, it has been useful for numerous applications such as speech synthesis research, character recognition and DNA sequencing research. Nowadays, HTK becomes very popular and is used at hundreds of sites over the world.

HTK is actually a set of library modules developed by C programming language. The software fully provides the most necessary facilities for speech analysis, HMM training, testing and result analysis. In addition, we can adjust HMM components easily to optimize the HMM's performance. Both continuous density mixture Gaussians and discrete distribution are supported in HTK. Of course, a complex HMM system can be built as expected by using HTK.

For more details about this toolkit, please refer to http://htk.eng.cam.ac.uk/docs/history.shtml

2.5.1 HTK Architecture

HTK is built into the library modules which make sure that every tool can interface to the outside world in exactly the same way. Each module, which consists of many tools, undertakes a specific task. The figure 2.7 illustrates the HTK architecture.

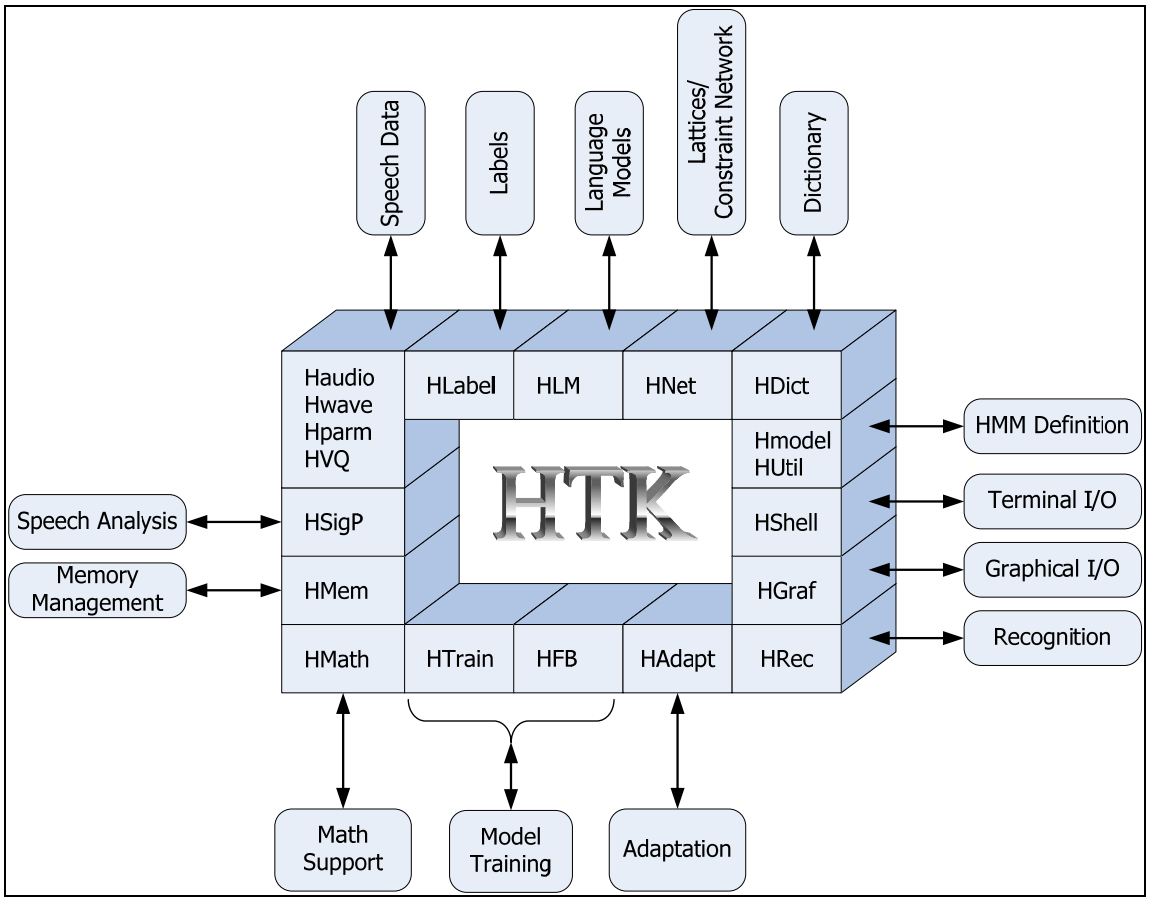


Figure 2.7 HTK Software architecture.
From Young (2006, p. 15)

HTK totally comprises 20 library modules:

1. *HShell* controls user input/output and interaction with operation of HTK.
2. *HMem* undertakes all management of memory.
3. *HSigP* is to process the signal operations needed for speech analysis.
4. *HMath* provides mathematical support.
5. *HLabel* is responsible for interaction with label files.
6. *HLM* is for language model files.

7. *HNet* is used for network and lattice operations.
8. *HDict* is dedicated to dictionary.
9. *HVQ* is for Vector Quantization codebooks.
10. *HModel* specializes in HMM definitions.
11. *HWave* is a library module that processes all speech input and output at the waveform level.
12. *HParm* is similar to HWave but at the parameterized level.
13. *HAudio* is a HTK utility which takes input directly from audio files.
14. *HGraf* has the same utility as the HAudio but is more interactive by using a graphical interface.
15. *HWave* in cooperation with HLabel provide multiple format files that allow data to be imported from other systems.
16. *HUtil* support a number of utilities so as to manipulate HMMs.
17. *HTrain* and *HFB* provide the various HTK training tools.
18. *HAdapt* contains support for the various HTK Adaptation tools.
19. *HRect* processes the recognition operations.

2.5.2 HTK functionalities

HTK fully provides the necessary tools so that we can build a HMM based continuous speech recognizer which usually involves in the processing steps: data preparation, training, testing and analysis. Relationship between these steps and HTK tools is shown in the Figure 2.8.

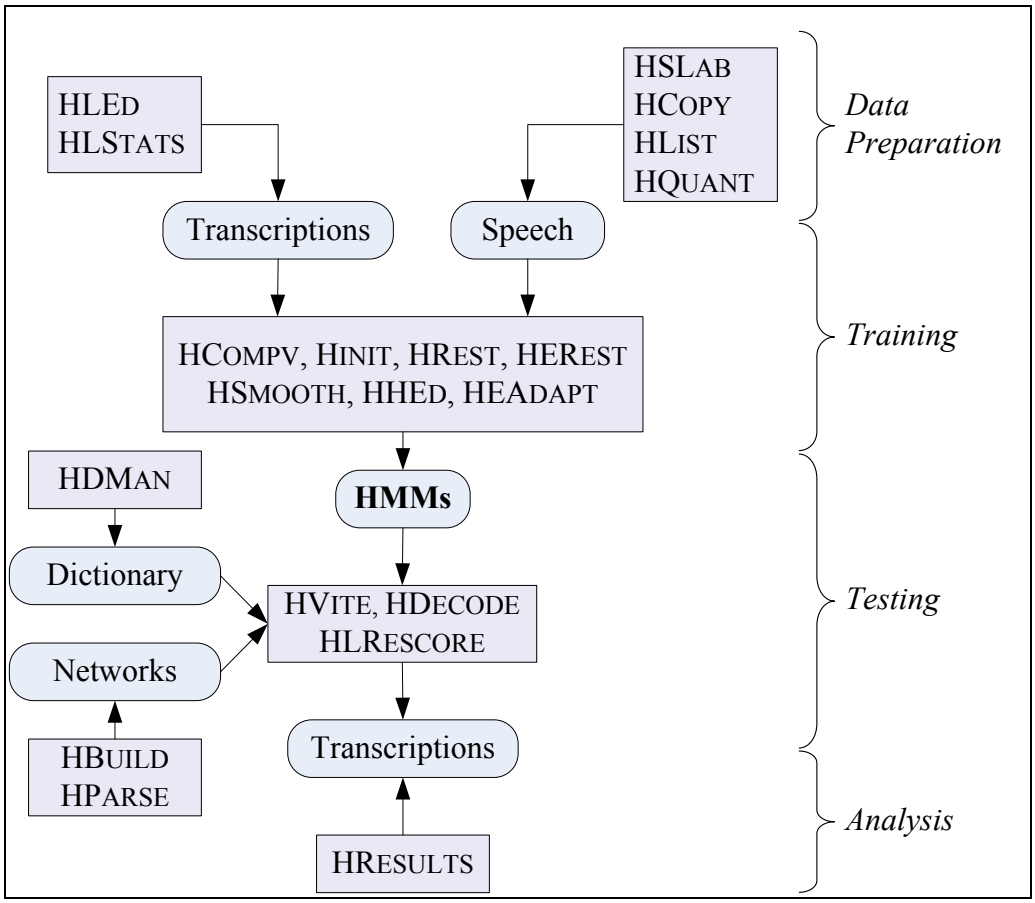


Figure 2.8 HTK Processing stages.

- Data preparation phase: A set of speech data files and their associated transcription are required for training a speech recognizer. But before they can be used in training step, we must convert the speech files into the appropriate parametric form and format its associated transcription in exactly the same way required by HTK. The tools such as HSLAB, HCOPY, HLIST, and HQUANT… are provided to work with speech file whereas the tools such as HLED and HLSTATS are used to annotate transcription.

- Training phase: is the most important step which involves building HMMs for recognizer using Baum-Welch re-estimation. HEREST is tool performing this algorithm. In this

phase, a set of tools are provided by HTK to adjust HMMs training with the purpose of optimizing recognition. These are HCOMPV, HINIT, HREST, HSMOOTH, HHED and HEADAPT.

- Testing phase: is step for speech recognition. For this purpose, HTK provide a tool called HVITE which does recognition using language models and lattices. Another tool, HDECODE, which supports the same task, is also available as an extension to HTK. And HLRECORE is a tool that accompanies HVITE (or HDECODE) to apply a more complex language model. Besides, HTK supplies the dictionary management tool HMAN which helps us construct large dictionaries. The final task in this phase which is to create grammar networks is assisted by the tools HBUILD and HPARSE.

- Analysis phase: Once the HMM based speech recognizer has been built, it is necessary to evaluate its performance by using itself to transcribe some pre-recorded utterances. After that, we will match the recognizer output with the correct reference transcription to see how well the recognizer works. This is done by the tool named HRESULTS which is also provided by HTK.

CHAPTER 3

METHODOLOGY

3.1 Problematic dialogue definition

In Chapter II, we saw that there are several different methods to label a problematic dialogue in situation of a spoken Human-Computer dialogue. The labeling depends on the purpose of dialog management system.

- First, Litman et al correlated a problematic dialogue with the performance of the ASR module (Litman et al., 1999). According to the opinion, a dialogue is considered problematic if the ASR module produces a poor performance on it.

- Then, Langkilde et al made another definition of problematic dialogue (Langkilde et al., 2000). On the basis of task success, problematic dialogue is a dialogue in which user is unable to do what he/she desires.

- Finally, Hastie et al relied on user satisfaction in order to define a problematic dialogue (Hastie et al., 2002). In this definition, problematic dialogue is a dialogue in which user is unsatisfied with the conversation. Otherwise, non problematic dialogue is a dialogue that always assures the user satisfaction although the task that user desires to do would be probably uncompleted.

In this work, we choose the method of labeling a problematic dialogue based on user satisfaction because:

- Our work is a part of the project whose aim is to address enhancing the quality of spoken Human-Computer dialog using emotion management.

- The corpus we use throughout the thesis provides user satisfaction rating which will be described more details in Section 3.5.

3.2 Dialog classification features

This section describes briefly the set of features which has been frequently used in researches concerning dialog classification or problematic dialog identification. The dialog features used in the work that we described in section 2.2 are often divided into four subsets based on their own origination as follow:

- Acoustic/ASR Features: output of the speech recognizer, number of words in the recognizer output, duration in seconds of the input to the recognizer, flag for touchtone input, input modality expected by the recognizer, grammar used the recognizer, and actual modality of the user utterance.

- NLU Features: confidence measure for all possible call types, intra-utterance measure of the consistency between services that user appears to be requesting, measure of coverage of the utterance by salient grammar fragments, measure of the shift in context between utterances, call-type task with the highest confidence score, call-type task with the second highest confidence score, difference in value between the top and the next-to-top confidence scores.

- Dialog Manager Features: number of utterances, number of re-prompts, percent of re-prompt, number of confirmation, percent of confirmation, utterance duration in second, entire dialog duration in second.

- Hand-Labeled Features: human transcripts of each user utterance, age and gender of each user, a cleaned transcript with non-word noise information-removed, the number of words that occurred only in the clean transcript.

We have just taken a look at the overview of different dialog feature types that we can extract from the human-machine dialog system. In the next section, we will introduce the features we chose to use in our research.

3.2.1 Selection of features

This work aims to design a completely automatic system of dialog classification. To achieve this, the dialog features that we use must be automatically extractable. In addition, since we want to give a quick response, we will work on utterance basis instead of a paragraph basis. Therefore, the dialog features here are derived from each utterance. Consequently, we adapted the features defined in the PARADISE framework to make them applicable at the turn level. The set of features used in our system comprises the following:

- Utterance Position: position of each utterance in a whole dialog (e.g. second utterance)
- Utterance Duration: duration in second of the utterance.
- Number of Phonemes: number of phonemes of the utterance.
- Inverse Speech Rate: inversion of measure of the speed that the utterance is spoken in second.
- Num Negative Words: number of negative acknowledgment words such as cancel, wrong, erase and so on.
- Num Positive Words: number of positive acknowledgment words such as yes, thanks, welcome and so on.
- Response Waiting Time: waiting time expressed in second between two consecutive utterances.
- Silence Time: duration of silence of the speaker.
- Repetition Rate: level of similarity between two consecutive utterances from the same speaker.
- ASR Accuracy Rate: measure of precision of ASR component when recognizing each user utterance.

Of these features, the last one (ASR Accuracy Rate) is the only one we are unable to automatically extract because we need corresponding transcript for each utterance in order to compute it. Otherwise, we can realize that all other features belong to the type of dialog costs (dialog efficiency and dialog qualitative) according to PARADISE framework.

In Chapter 4, we will describe how each of these feature parameters is extracted. In the next section, we mention how a HMM can be used to detect the satisfaction of the user. In the next subsection, we will describe the algorithm and the technique used in processing the features extraction.

3.2.2 The dynamic programming algorithm

The dynamic programming is a popular algorithm (Cheriet, 1997) which solves complex problems by breaking them down into simpler sub-problems. This algorithm saves much more time than naive one. Based on algorithm approach, dynamic programming is basically divided into two types as follow:

- Top-down dynamic programming is to store the results of calculations which are re-used after because the same calculation is a sub-problem in a larger calculation.
- Bottom-up dynamic programming is to formulate a complex calculation as a recursive series of simpler calculations.

There are a number of algorithms that use dynamic programming. Here we list some common algorithms:

- The dynamic time warping algorithm for computing the global distance between two time series.
- The algorithms used in bioinformatics, including sequence alignment, structure alignment and RNA structure prediction.
- The Viterbi algorithm in hidden Markov models.
- Fibonacci sequence.

- Sequence alignment.

In our research we use the dynamic programming to measure optimal distance between two given strings of character. Suppose that we have an n-character string $X = X_1, X_2, ..., X_n$ where X_i denotes the i^{th} character in the string X. Similarity, we also have an m-character string $Y = Y_1, Y_2, ..., Y_m$. And the dynamic programming algorithm that we use to solve our own problem is the following:

Step 1: Initialization

D[0,0] = 0

for i = 1,2,…,n {D[i,0] = ∞}

for i = 1,2,…,m {D[0,m] = ∞}

Step 2: Iteration

for i = 1,2,…,n {

for i = 1,2,…,m {

$$D[i,j] = \min \begin{bmatrix} D[i-1,j]+1 & (deletion) \\ D[i-1,j-1] \quad if(X_i = Y_j) & (match) \\ D[i-1,j-1]+1 \; if(X_i != Y_j) & (subtitution) \\ D[i,j-1]+1 & (insertion) \end{bmatrix} \tag{3.1}$$

}

}

Step 3: Termination

The optimal distance between X and Y is ***d(X,Y) = D[n,m]***

3.2.3 The emotional salience (Lee et al., 2002)

The emotional salience is an approach used in emotion detection to improve the recognition performance. This technology is based on the information-theoretic concept of salience to search salient words in each utterance. A salient word with respect to a category is one which appears more often in that category than at in other parts of the corpus. To find the salient words, we need a method to measure relation between the words and emotions in the speech corpus. Therefore, the emotional salience is created as a measure of the amount of information that a specific word implies about the emotion category. The emotional salience $sal_{e_k}(w)$ of a word w for emotion category e_k is defined as mutual information between a specific word and an emotion class

$$\boxed{sal_{e_k}(w) = I(E = e_k; W = w) = P(e_k|w)i(w, e_k)} \tag{3.2}$$

where

$P(e_k|w)$ is the posterior probability that word w implies emotion class e_k.

$i(w, e_k) = \log_2 \frac{P(e_k|w)}{P(e_k)}$ is the self mutual information. (3.3)

$P(e_k)$ is the prior probability of the emotion e_k.

The meaning of the emotional salience is that the bigger the emotional salience is, the higher the correlation between a specific word and an emotion category is.

3.3 Selection of machine learning method

In pattern classification field, it is not easy to know beforehand which learning scheme will work best for any given problem. The approach called “trial-and-error” is usually used to determine an appropriate scheme.

In the section of related work, we saw that two schemes, namely RIPPER and DT (Decision Tree), are often employed in the dialog classification problem. But our research approaches this problem with a different method. We attempted to mine only information exchanged between user and system for dialog classification at the utterance level and then model the user satisfaction through the dialog utterance. So, it is necessary to have an appropriate algorithm for this approach. HMM is our choice due to the following reasons:

- HMM is a robust and strong algorithm used widely in the field of temporal pattern recognition such as speech recognition, gesture recognition, and gene prediction. Hence, we also believe that this model will work well on our problem.

- Besides, we realize that HMM is very suitable to model our problem. Since if we consider an utterance sequence of dialog as an observation sequence in HMM illustrated like the Figure 3.1. This offers us an easier way to process modeling than the conventional algorithms such as RIPPER and DT.

- In addition, the ASR component of the Human-Computer dialogue system also works on a HMM framework. Thus, using HMM in dialogue classification is compatible with the ASP component in order to facility the fusion of both information.

- Finally, HMM provides us many means to improve and refine the algorithm performance. Simply, we can adjust number of states or work on the topology of states in order to improve the system's performance. Using Gaussian Mixture Model for the probability density function of each HMM state could be made to enhance the classification of dialogue. In case of limitation of training data, HMM is still able to adapt to this situation using the parameter tying.

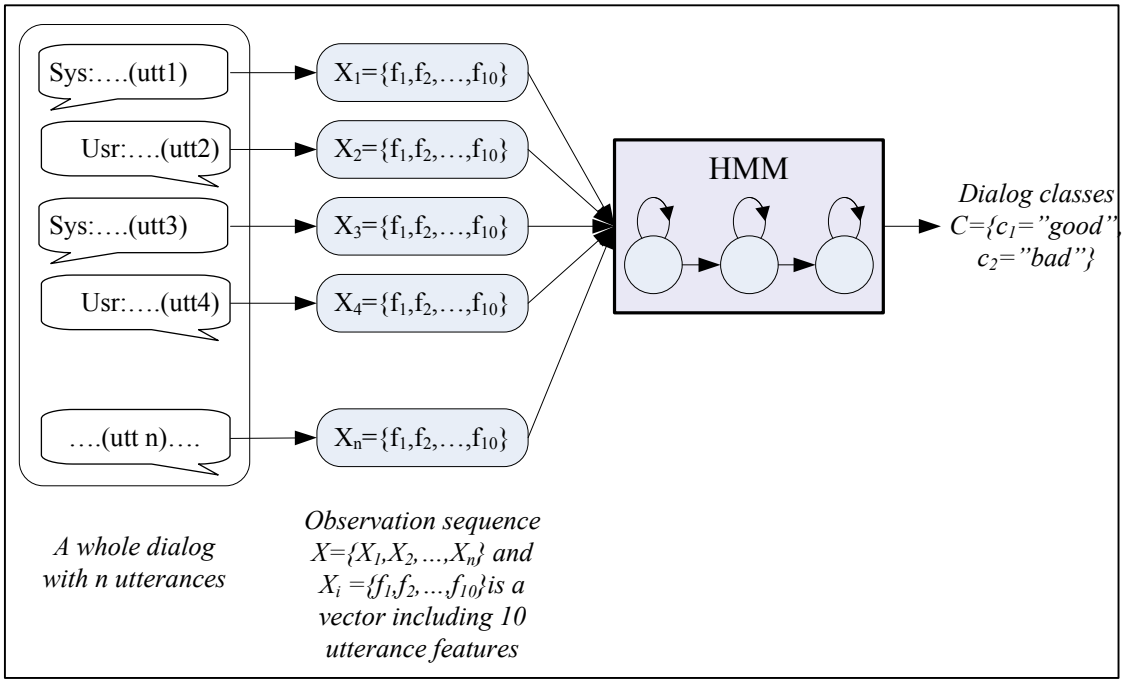


Figure 3.1 Concept of new approach using HMM.

3.4 Corpus Collection

This section briefly presents the data collection used for our experiments. The collection includes two corpora: DARPA 2000 Communication Corpus, and DARPA 2001 Communication Corpus. They were publicly distributed by Linguistic Data Consortium (LDC). Firstly they were collected to serve the DARPA Communication project whose object is to support development of multi-modal speech-enabled dialog systems with advanced conversational capabilities.

During 2000 and 2001, the two data sets were collected, in which users used the Communicator systems built by the research groups to plan their travel trip.

3.4.1 DARPA 2000 Communication corpus (Walker et al., 2002)

DARPA 2000 corpus was collected by nine systems from AT&T, BBN, University of Colorado, Carnegie Mellon University, IBM, Lucent Bell Labs, MIT, MITRE and SRI. The

collection was conducted as a controlled experiment in which the subjects, who were recruited in prior, called each of the nine different automated travel-planning systems over three 3-day periods to make simulated flight reservation.

The experiment had been carried out over nine days at the fixed hours. The subjects talked to the systems in order to complete seven fixed tasks (which included three domestic one way trips, two domestic round trips, and two international round trips) and two open tasks that were to plan an intended business trip and vacation.

As a result, this experiment produced 662 calls in which there were 225 calls for one way trip reservation, 300 calls to make a round trip reservation. And the remaining 137 ones are about the real trip.

3.4.2 DARPA 2001 Communication corpus (Walker et al., 2002)

DARPA 2001 corpus was collected by only eight systems because the system from MITRE didn't participate in this project anymore. In 2001, the collection was more natural than the one in 2000 because they conducted a within-system rather than a within-subject experiment. They did exclude the subject recruitment for the experiment. The object of this is to allow users to learn their new interactive paradigm and allow systems to adapt to their users.

Unlike the experiment in 2000, the one in 2001 had taken place for a long time (6 months) and the user could be continuously accessible to systems via a toll-free number. Besides, the experiment's scenarios were more complex and realistic. Multi flight trip was required to be made in such a corpus. Moreover, they might require user to make car and hotel arrangement as well.

After this experiment, they collected 1242 calls in total. This number consisted of 198 calls for round trip reservation, 350 calls within complex scenario. And the remaining 694 calls are about real trip.

The following table summarizes the different between DARPA 2000 and DARPA 2001.

Table 3.1 Summary of DARPA 2000 and DARPA 2001

	DARPA 2000	**DARPA 2001**
Period	9 days	6 months
Time	Fixed Hours	Continuous
Design	Within subject	Within system
Number of calls	662	1242
Call Types	225 One Way, 300 Round Trip, 137 Real	198 Round Trip, 350 Complex, 694 Real

3.5 Labeling corpus

In order to label the dialogues from the corpora of DARPA 2000 and DARPA 2001, we rely on the points that the user answers to a question set. The questions are given to the user after his dialogue is completed. The answers help us assess the customer satisfaction towards the spoken Human-Computer dialogue system. Below is the question set raised in the DARPA 2000 and 2001 corpus:

- *Task Success*: Is user's task completed successfully? (Yes / No)
- *Task Ease* – (A): In this conversation, it was easy to get the information that user wanted?
- *TTSPerf* (Text To Speech Performance) – (B): In this conversation, user found it easy to understand what the system said?
- *User Expertise* – (C): 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?

The first question is used to determine whether user's task is successfully done while the last

five questions are taken for customer satisfaction rating. In this work, we focus on these five questions because we address problematic dialogue on the basis of the customer satisfaction as mentioned in section 3.1. For each question, user will give a score whose value varies from 1 to 5 based on Likert-scale that is a multi-item scale. The Liker-scale format is presented in the Table 3.2.

Table 3.2 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

After summing up the scores of these five answers, we have a score that expresses the actual user satisfaction. We call such score UserRating, and use it to define problematic dialog as follow:

$$\boxed{UserRating < Threshold \Rightarrow Bad\ Dialog}$$

Since we have two different corpora using two types of Likert-scale, namely Normal Likert-scale and Inversed Likert-scale (Table 3.2), the Thresholds that we choose to label dialogs have a little difference as follows:

- In DARPA 2000, the Inversed Likert-scale is used, so determining problematic dialog (Bad dialog) follows the condition below

$$\boxed{UserRating > 12 \Rightarrow Bad\ Dialog}$$

- Otherwise, in DARPA 2001, the Normal Likert-scale is used, so determining problematic dialog (Bad dialog) follows the condition below

$$\boxed{UserRating < 17 \Rightarrow Bad\ Dialog}$$

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 System design

The dialog classification system that we develop is also a pattern classification system which usually has three main components, namely preprocessing, extraction, and classification. Our system is not an exception and Figure 4.1 illustrates the overall architecture of the dialog classification system.

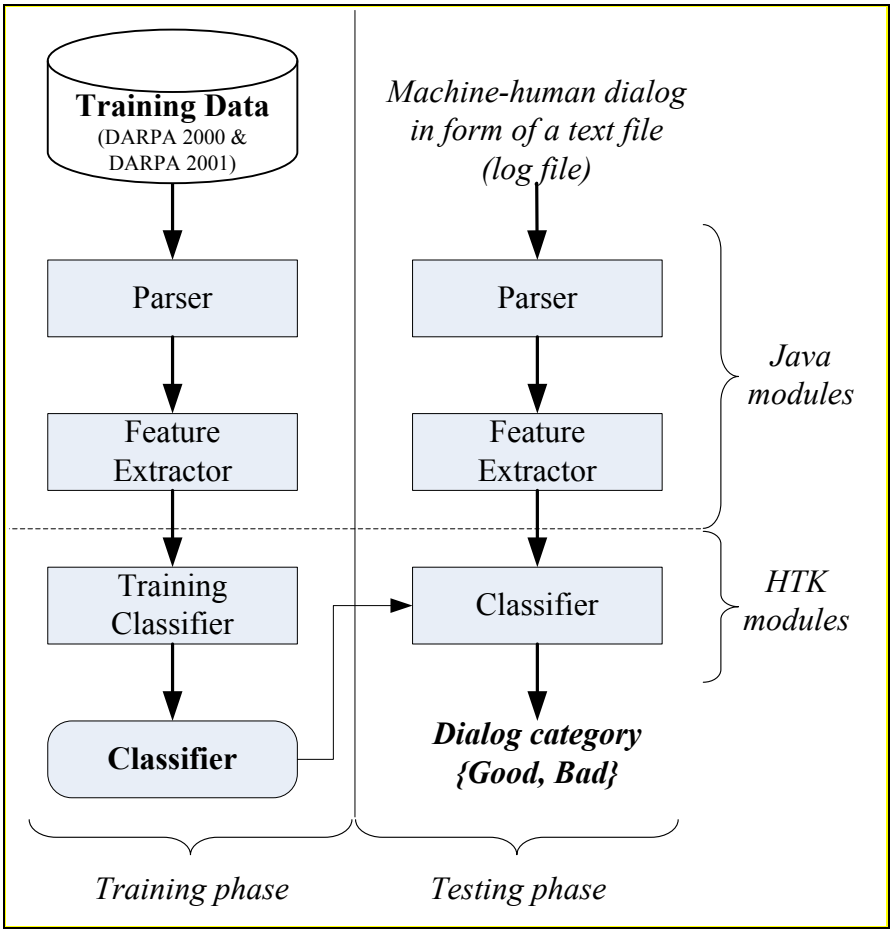


Figure 4.1 Overall architecture of dialog classification system.

As shown in the Figure 4.1, DCS includes three main components as follow:

- Parser: this component plays the role as the preprocessing. Since the input of DCS is a dialog in form of a text file recording what the user and the machine agent communicated and the time when the dialog took place. The task of the preprocessing component is to parse the text file to pick up dialog information. This is why this component is called parser in our system. We develop the parser component using the Java programming language. An example about the task of the parser component is shown in the table below.

Table 4.1 Example of input and output of the parser

Input of parser	Parser	Output of parser
Fri Jul 7 2000 at 12:21:44.71: New user turn began.	=>	User
Fri Jul 7 2000 at 12:21:44.71: User started speaking.	=>	12:21:44.71
User audio file: ASR0_112644.VOX	=>	12:21:47.13
Fri Jul 7 2000 at 12:21:47.13: User finished speaking.	=>	I KNOW THAT'S ROUND
Recognizer heard: I KNOW THAT'S ROUND	=>	UH NO THAT'S WRONG
User said: UH NO THAT'S WRONG		

- Feature Extractor: this component takes input from the output of the parser component in order to produce a set of features used in the next component, namely classifier. Similarity to the parser component, the feature extractor is developed by the Java programming language. Table 4.2 is an example of what the feature extractor component does in our system. In this example, some features (such as Utterance Position, Silence Time and Repetition Rate) are calculated based on the previous utterance.

Table 4.2 Example of input and output of the feature extractor

Input of Extractor	Extractor	Output of Extractor
User 12:21:44.71 12:21:47.13 I KNOW THAT'S ROUND UH NO THAT'S WRONG		Utterance Position : **4** Utterance Duration : **2.42** Number of Phonemes : **10** Inverse Speech Rate : **0.22** Number of Negative Words : **0** Number of Positive Words : **0** Response Waiting Time : **0.2** Silence Time : **0.0** Repetition Rate : **0.0** ASR Accuracy Rate : **0.5**

- Classifier: this component classifies dialog into one of two classes {Good, Bad} based on a set of features. For the component, we use HTK ToolKit to build the classifier.

Each component will be described in detail in Section 4.1.1, Section 4.1.2 and Section 4.1.3.

4.1.1 The parser

Parser is the first component of our system. It receives a dialog transcript in form of a text file as input. The main task of parser is to pick up important information that the feature extractor needs. To do so, the parser must parse the text file and remove noise and unnecessary information from the text file. Only useful information is kept before sending them to the feature extractor. The output of the parser is composed of the following fields for each user turn and system turn in the dialog:

- Is system: the field is used to determine if this is a system turn or user turn.

- Starting time: is the time when the user (or the system) starts their own turn.
- Finishing time: is the time when the user (or the system) finishes their own turn.
- Utterance: here, we have two types of utterance. The first one is the user utterance which is transcribed in the text file by the ASR (Automatic Speech Recognition) component of the human-machine dialog system. The second one is the system utterance. For this utterance, the system simply saves to the text file what it said to the user. We thus realize that the user utterance could be incorrect due to limited capacity of the ASR component.
- Human transcription: this field contains a human transcription of exactly what the user said to the ASR system. This task is done by the human agent who listened to the audio file recording the corresponding dialog and rewrote out what they heard to the text file. Of course, this information only exists in the user turn.

Figure 4.2 summarizes the input and output of the parser.

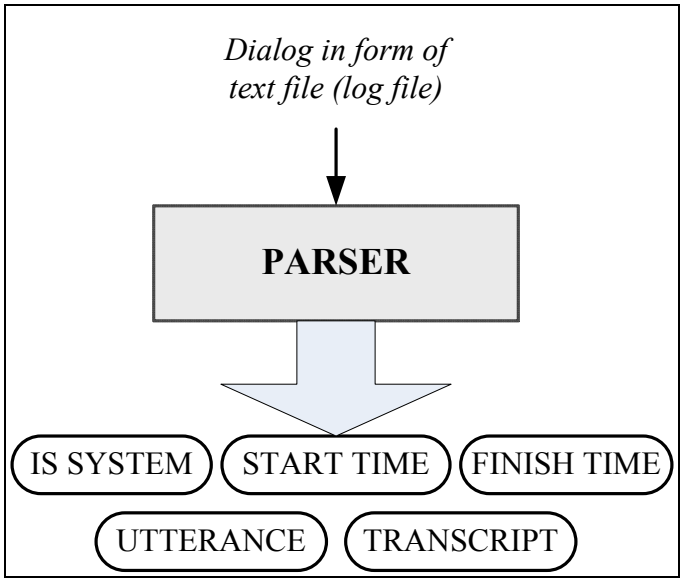


Figure 4.2 Illustration for the parser function.

In our research, we have two different corpora whose text file formats varied. So, we need two parsers, namely the one for the DARPA 2000 and the other one for the DARPA 2001.

We will describe how they parse text files of two DARPA corpora in the next subsections.

4.1.1.1 DARPA 2000 Communicator

The text file of the DARPA 2000 corpus has a format as displayed the Figure 4-3. According to this format, parser will:

- Determine if this is a user turn or system by searching the key word “system turn began” or “user turn began”.
- Extract start time of system or user by searching the key word “System started speaking” or “User started speaking” respectively.
- Extract finish time of system or user by searching the key word “System finished speaking” or “User finish speaking” respectively.
- Extract system utterance by searching the key word “System said”.
- Extract user utterance by searching the key word “Recognizer heard”.
- Extract transcript by searching the key word “User said”.

4.1.1.2 DARPA 2001 Communicator

The text file of the DARPA 2001 corpus has a format as displayed in Figure 4-4. This format is simpler than the one of DARPA 2000. Each turn lies in one line only. And the format of each turn is:

[User/system] [Start time] [Finish time] [ASR: utterance] <Transcr: Transcript: >

The last information is optional and exists in the user turn only. For this corpus, the parser needs to parse text file to pick up line by line, after that it will:

- Determine if this line contains the key word “system” than this is a system turn, whereas this is user turn.
- Extract one by one the remaining fields based on the order of each field as mentioned above.

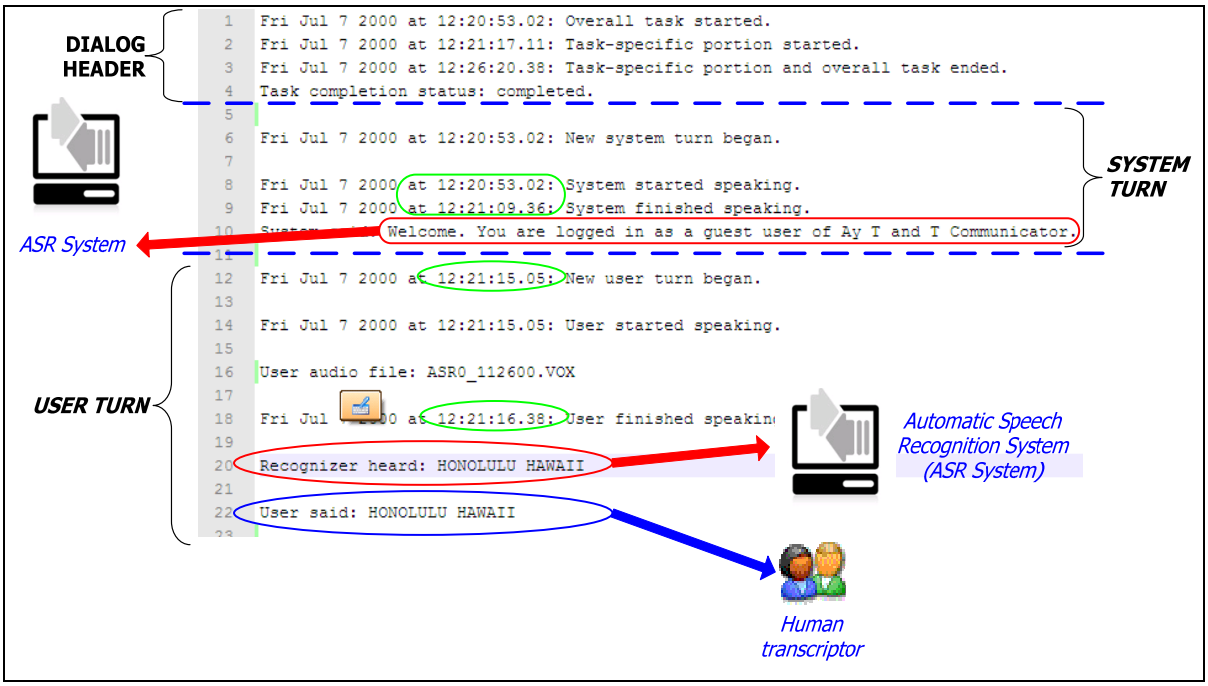


Figure 4.3 Illustration for the text file format in DARPA 2000.

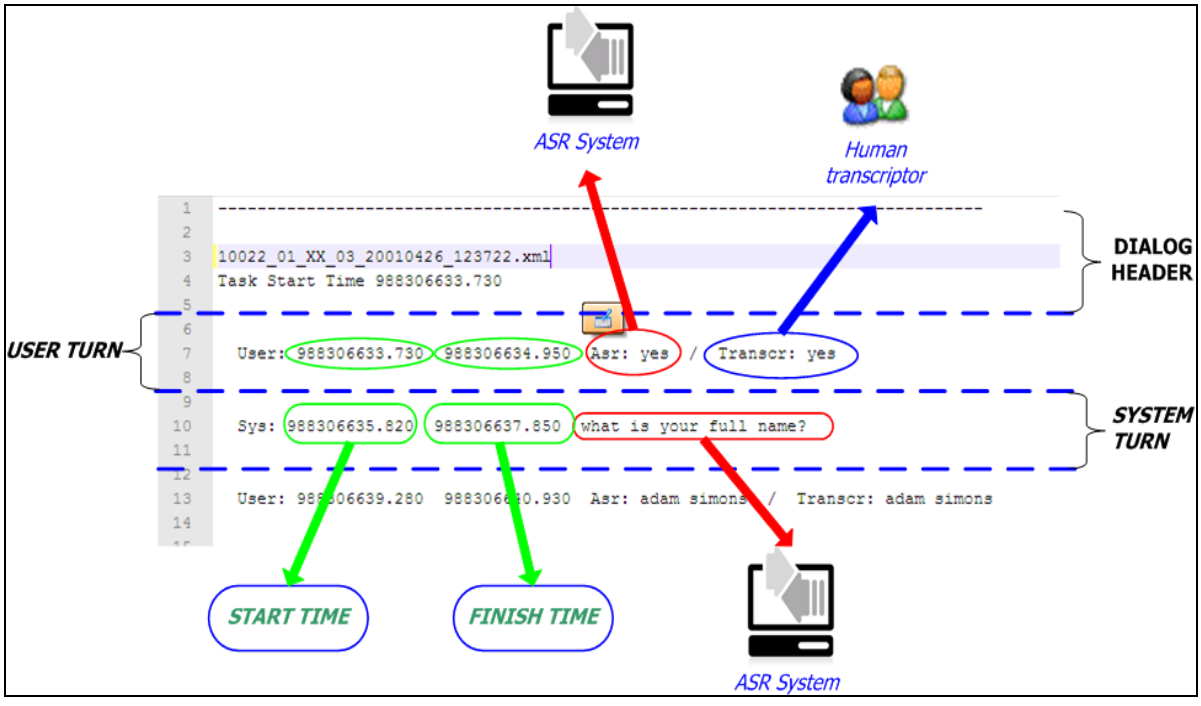


Figure 4.4 Illustration for the text file format in DARPA 2001.

4.1.2 Feature extractor

This component receives information about each turn from the parser. Based on this information, the feature extractor will produce a set of ten features which we call a feature vector. The Figure 4.5 shows the input of this component and the ten features that will be extracted for each turn by the feature extractor.

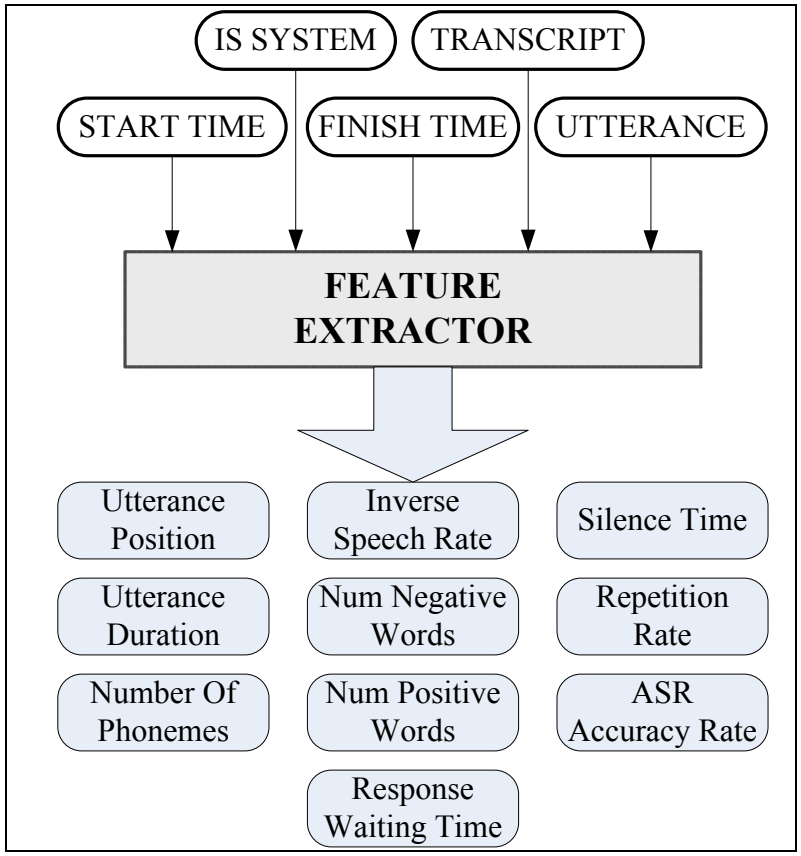


Figure 4.5 Illustration for the feature extractor function.

The next section describes the methods that the feature extractor uses to compute those features for each turn of a dialogue. In addition, we will briefly make a feature definition and explain why the features could be useful to our system.

- **Utterance_Position** is simply the order number of each utterance in the whole dialog.

This feature gives our system the number of times that the user and the machine exchanged turns. We make the assumption that a good dialog is the one in which the user and the machine do not need to talk too much to understand. On the contrary, a dialog which has too few exchanges is not a good one. Our system therefore can evaluate how good a dialog is based on this feature.

$$\boxed{\begin{array}{c} Utterance\_Position = previous\_Utterance\_Position + 1 \\ (First\_Utterance\_Position = 1) \end{array}} \quad (4.1)$$

- **Utterance_Duration** is duration defined as the moment when the user (or the system) starts speaking and the moment when they finish their utterance. This feature always appears in the dialogue-evaluation-related research.

$$\boxed{Utterance\_Duration = Finishing\_Time - Starting\_Time} \quad (4.2)$$

- **Number_of_Phonemes** as the name suggests is used to count how many phonemes are pronounced in each utterance. As for the previous feature (Utterance duration), this is also a basic feature found in the research of dialog classification. Moreover, we think that the number of phonemes provide to our system the amount of information exchanged between the user and the system.

$$\boxed{Number\_of\_Phonemes = countPhonemes(utterance)} \quad (4.3)$$

- **Inverse_Speech_Rate** measures the average speed that each phoneme is emitted in an utterance. As a result, the speech rate is computed by dividing Utterance Duration by Number of Phonemes. In a situation of telephone dialog, there are various methods to reveal sensation and satisfaction, and we make the assumption that the speech rate is one of them.

$$\boxed{Inverse\_Speech\_Rate = \frac{Utterance\_Duration}{Number\_of\_Phonemes + 1}} \quad (4.4)$$

Note that the one at the denominator in the above equation is used to avoid the infinitive value because sometimes, the *Number_of_Phonemes* is equal to zero meaning that nothing was uttered.

- **Number_of_Negative_Words** counts how many times words which are considered as negative ones appear in an utterance. In this definition, a negative word is a one that has a tendency to frequently exist in bad dialogs. We choose this feature under the assumption that the more negative words the dialog has, the higher the probability of a dialog of being a bad dialog is. A method of determining positive and negative words will be defined in the next subsection (Section 4.1.2.1).

$$\boxed{Number\_of\_Negative\_Words = countNegativeWords(utterance)} \quad (4.5)$$

- **Number_of_Positive_Words** is similarity to the feature "number of negative words" but now with positive words instead of negative words.

$$\boxed{Number\_of\_Positive\_Words = countPositiveWords(utterance)} \quad (4.6)$$

- **Response_Waiting_Time** is duration between the moment when the previous speaker finishes his utterance and the moment when the next speaker starts his. A casual conversation between two people will be considered good if no one of them has to take a long time to wait for the response from the other person. And the spoken dialog is not also an exception. This feature thus provides meaningful information to evaluate a dialog.

$$\boxed{\text{Re}sponse\_Waiting\_Time = CurrentTurn.Start - PreviousTurn.Finish} \quad (4.7)$$

Note that Response_Waiting_Time can be less than zero. This could happen in two cases: when user's speech overlaps the system's speech or when system's speech overlaps the user's speech.

- **Silence_Time** is duration of time where the user (or the machine) doesn't say anything to

the machine (or the user) while the dialog between them is still going on. The purpose of using such feature is to emphasize the fact that the dialog is interrupted by the silence of the user (or the machine) when he is supposed to say in his turn.

$$\boxed{\begin{aligned} SilenceTime(of\ user) &= CurrentSystemTurn.Start - previousSystemTurn.Finish \\ SilenceTime(of\ system) &= CurrentUserTurn.Start - previousUserTurn.Finish \end{aligned}} \quad (4.8)$$

According to the feature definition, silence time is always equal or more than zero. The following scenario is an example in which we can calculate the Silence time.

[U1] Machine agent (*from 12:21:15.05 to 12:21:16.38*): Hello, may I help you?

[U2] User (*from 12:21:17.13 to 12:21:22.32*): ….. (**Silence**)

[U3] Machine agent (*from 12:21:24.18 to 12:21:25.39*): May I help you?

[U4] User (*from 12:21:26.25 to 12:21:32.71*): ….. (**Silence**)

[U5] Machine agent (*from 12:21:34.99 to 12:21:36.86*): Do you hear me?

In this scenario, there are two silence utterances, U2 and U4 respectively. The duration of these silence utterances is calculated as follows:

SilenceTime(of U2) = U3.Start – U1.Finish = *12:21:24.18 - 12:21:16.38 = 7.9* (seconds)
SilenceTime(of U4) = U5.Start – U3.Finish = *12:21:34.99 - 12:21:25.39= 9.6* (seconds)

- **Repetition_Rate** is a ratio that shows the similarity level between two consecutive utterances of the same speaker (user or system).

$$\boxed{Repetition\_Rate = 100 - \frac{d(previousUtterance, currentUtterance)}{length(currentUtterance)} \times 100} \quad (4.9)$$

where *d(previousUtterance, currentUtterance)* is to measure the difference between these two utterances and calculated by the dynamic programming algorithm mentioned in Section 3.2.2.

- **ASR_Accuracy_Rate** is a ratio computed to evaluate how well the ASR (which stands for Automatic Speech Recognition) recognizes what the user says to the system.

$$ASR\_Accuracy\_Rate = 100 - \frac{d(utterance, transcript)}{length(transcript)} \times 100 \quad (4.10)$$

where *d(utterance,transcript)* is the distance between utterance and corresponding transcript, calculated by a dynamic programming algorithm described in Section 3.2.2.

Of course, we're only able to extract this feature on user utterance only. For system utterances, a default value (100) will be assigned to the feature.

4.1.2.1 Determination of negative and positive words

This section introduces the method we adopt to identify negative and positive words: the emotional salience technique based on the mutual information. In order to determine whether a word is a negative or positive one, we use a training data which consists of a number of dialogs labeled as GOOD or BAD dialogs.

Here, we assume that a word which appears more frequently in the GOOD dialogs is a positive word; vice versa a word which tends to be in BAD dialogs is a negative word. Firstly, we denote:

- $E = \{e_P, e_N\}$ as the set of emotional classes (e_P: positive emotion, e_N: negative emotion).
- $W = \{w_1, w_2, ..., w_N\}$ as the set of vocabularies derived from the training data.
- N is the overall number of distinct words in both GOOD and BAD dialogs.
- n_i^{GOOD} is number of appearances of the i^{th} word in all the GOOD dialogs.
- n_i^{BAD} is number of appearances of the i^{th} word in all the BAD dialogs.

The following table shows the steps of determining positive word and negative words.

Table 4.3 Method of determining positive and negative word

Compute the prior probability of each emotion in $E = \{e_P, e_N\}$.

$$P(e_P) = \frac{\sum_{k=1}^{N} n_k^{GOOD}}{\sum_{k=1}^{N} n_k^{GOOD} + \sum_{k=1}^{N} n_k^{BAD}} \qquad P(e_N) = \frac{\sum_{k=1}^{N} n_k^{BAD}}{\sum_{k=1}^{N} n_k^{GOOD} + \sum_{k=1}^{N} n_k^{BAD}}$$

for each w_i in W {

$$P(w_i) = \frac{n_i^{GOOD} + n_i^{BAD}}{\sum_{k=1}^{N} n_k^{GOOD} + \sum_{k=1}^{N} n_k^{BAD}}$$

Compute the posterior probability $P(e_P|w_i)$ as follow

$$P(w_i|e_P) = \frac{n_i^{GOOD}}{\sum_{k=1}^{N} n_k^{GOOD}} \qquad P(e_P|w_i) = \frac{P(w_i|e_P)P(e_P)}{P(w_i)} \quad \text{(Bayes' theorem)}$$

Compute the posterior probability $P(e_N|w_i)$ as follow

$$P(w_i|e_N) = \frac{n_i^{BAD}}{\sum_{k=1}^{N} n_k^{BAD}} \qquad P(e_N|w_i) = \frac{P(w_i|e_N)P(e_N)}{P(w_i)} \quad \text{(Bayes' theorem)}$$

Compute the self mutual information is given by

$$i(w_i, e_P) = \log_2 \frac{P(e_P|w_i)}{P(e_P)} \qquad i(w_i, e_N) = \log_2 \frac{P(e_N|w)}{P(e_N)}$$

Compute the emotional salience of i^{th} word for emotion e_P and e_N

$$sal_{e_P}(w_i) = P(e_P|w_i)i(w_i, e_P) \qquad sal_{e_N}(w_i) = P(e_N|w_i)i(w_i, e_N)$$

if $(sal_{e_P}(w_i) \geq \delta_P)$ then w_i is a positive word with threshold $\delta_P : \frac{|W : sal_{e_P}(w_i) \geq \delta_P|}{|W|} = 25\%$

if $(sal_{e_N}(w_i) \geq \delta_N)$ then w_i is a negative word with threshold $\delta_N : \frac{|W : sal_{e_N}(w_i) \geq \delta_N|}{|W|} = 25\%$

}

4.1.3 The classifier

In this study, HMM is the paradigm used to classify dialogs. In this section, we explain how HMM is configured for the purpose of dialog classification.

To implement an HMM based dialog classification, we need to have two distinct HMM models which correspond to two classes C= {“good”, “bad”}. One model is used to recognize good dialogs and the other one to recognize bad dialogs. We call two models Φ_1 and Φ_2 respectively. To classify a given dialog defined by the observation sequence $X = \{X_1, X_2, ..., X_n\}$ that is a set of feature vectors extracted by the Feature extractor, we must do the followings steps:

- Using the Forward algorithm to compute the probability $P(X|\Phi_1)$. (see Annexe I for more detail)
- Using the Forward algorithm to compute the probability $P(X|\Phi_2)$.
- After that, we classify this dialog by choosing the best probability: $class = \arg\max_{i=1,2} \{P(X|\Phi_i)\}$

Of course, before using these models for classification, given a set of training examples corresponding to a particular model, each model must be trained by the Forward-Backward algorithm (Annex IV) to get the best set of parameters $\Phi = \{A, B, \pi\}$ for each model. Figure 4.6 illustrates the process of HMM based dialog classification.

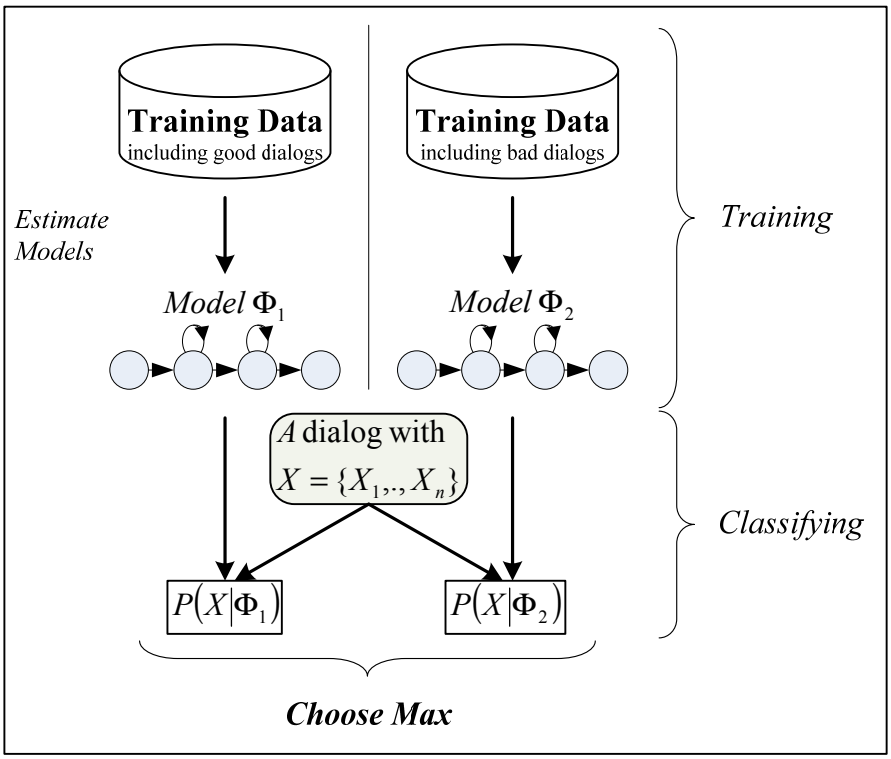


Figure 4.6 Using HMMs for dialog classification.

The HMM topology which we initially use for dialog classification is model with two states. This model can get started and ended at any states. Every state has two transitions. A transition takes from one state to another one including itself. This HMM topology is shown as the following figure.

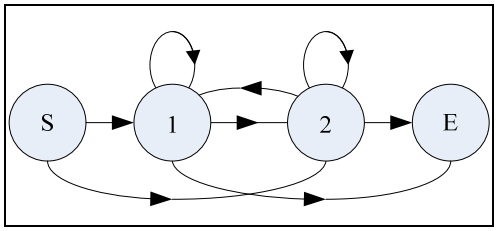


Figure 4.7 HMM topology for dialog classification.

4.2 System Development

As mentioned in the section of system design, DCS has three main components and two of

them, Parser and Extractor, are developed by the Java programming language. The last component, Classifier, is built using HTK. This is also the most important component in DCS. Therefore, this section only focuses on its development.

Building the classifier by HTK includes three phases: data preparation, training and testing. For each phase, HTK always provides corresponding commands which help us implement HMM easily and effectively (Young et al., 2006).

4.2.1 Data preparation tool

Data preparation phase is indispensable to use HTK. The objective of data preparation is to store feature vectors (observation sequence) in form of the HTK format parameter file which is the input of the next phases (training and classifying). The HTK format file includes a header followed by a contiguous sequence of samples. Each sample is a vector of 4-byte floats. And the HTK format header is length of 12 bytes and contains the following information:

- nSamples: number of samples in file (4-byte integer),
- sampPeriod: sample period in 100ns unit (4-byte integer),
- sampSize: number of bytes per sample (2-byte integer),
- parmKind: a code indicating the sample kind (2-byte integer).

Actually, the HTK format header has more other parameters but here we only mention basic parameters which constitute the structure of the HTK format file used throughout our system. The task of this phase is done by Java programming language because HTK only supports creating HTK format parameter file from audio format files.

4.2.2 Training tool

HTK provides us two library modules for training HMMs with many commands such as HRest, HERest to support several training strategies. The strategy that we apply to our system includes the following steps:

- Define the topology required for each HMM by writing a prototype definition. HMM definition can be stored externally as simple text files and hence you can edit them by any text editor.
- Initialize all of the parameters of each model in order to prepare for the step of Baum-Welch re-estimation (Forward-Backward algorithm). HTK provides two commands named HCompV and HInit to do such a step.
- Perform the Baum-Welch re-estimation procedure to estimate a set of parameters as best as possible that each model can obtain. HERest is created to be in charge of doing this task.
- Incrementally refine HMMs obtained after re-estimate procedure thanks to many different methods. These could be using multiple mixture component Gaussian distributions or applying a variety of parameter tying. And HHed is a tool of HTK that help us doing this.

4.2.3 Classifying tool

After the training phase, we have determined the best model for each model. The models will be used in classification phase in order to recognize a dialog as good one or bad one. As previously mentioned, dialog classification is simply a task that involves in computing the probability given by the two models and choosing the best probability to determine which class the dialog belongs to. This task of our system is easily done by the HVite which is one of commands of the library module HRec.

CHAPTER 5

EXPERIMENTATION

5.1 Experiment protocol

This section describes several popular methods used in the experiments to assess a learning scheme. Here, we will mention three methods as follow (Witten et al., 2005).

5.1.1 Holdout method

This method is used when we have a large dataset. The dataset is split into two separate subsets, namely called the training set and the testing set. The size of training set is always much larger than the one of the testing set. The advantage of this method is that it doesn't take too much time for experimentation because it doesn't do any iteration. However, its evaluation couldn't be highly reliable. The cause is that the evaluation depends on which data points end up in the training set and which end up in the testing set. In other words, the evaluation depends on how the division is made.

5.1.2 K-fold cross-validation method

This method is used when our dataset is limited. The method works as follow:

- Firstly, the dataset is divided into *K* subsets.
- Next, the holdout method is repeated *K* times. Each time, one of the *K* subsets is used for testing and the remaining *K-1* subsets are put together to be used as training set.
- Finally, we average the error across all *K* trials.

It's easy to realize that the advantage of the method is to avoid the situation where the evaluation is dependent on how the dataset is divided. Because every data point participates in a testing set exactly once and also gets to be in a training set exactly *K-1* times.

However, the disadvantage is that it takes more computation time because the train/test phase

must be done *K* times. Compared to the holdout method, this method takes *K* times to complete the evaluation with assumption that the proportion between the training set and testing set of the two methods is identical.

5.1.3 Leave-one-out cross-validation

As the name suggests, the leave-one-out cross validation (LOOCV) involves using a single data point from the dataset as the testing data. This is repeated until every data point is used once as the testing data. Hence, this method is a special case of the K-fold cross-validation method as the coefficient *K* is equal to N, number of data point in the dataset. The evaluation given by the LOOCV error rate looks very good, whereas the computation time is very high, especially for the large dataset.

In our study, we decided to choose the second method with K=10 (the K-fold cross-validation method) because our dialog datasets are limited. Moreover, previous experiments made by the other authors were also using K=10.

5.2 Evaluation measure

The previous section is about different methods which are dedicated to the experiments used in the classification problem. This section discusses the metrics that we use to evaluate the performance of a classifier.

Model evaluation is the important step in the process of building a classifier. Depending on the particular problem, different metrics are applied to evaluate the model performance.

First, we will introduce a confusion matrix before mentioning the evaluation measures. The confusion matrix is a matrix which lists the correct classification against the predicted classification for each class. In this case, we have only two classes, namely Good class (Good) and Bad class (Bad). Table 5.1 is a confusion matrix for a 2-class classification problem. In this table, the number of the correct predictions for each class falls along the diagonal of the matrix. Otherwise, the other cells are the number of misclassification error.

Table 5.1 Confusion matrix for 2-class classification problem

	Actual Good	**Actual Bad**
Predicted Good	*True Positive (TP)*	*False Positive (FP)*
Predicted Bad	*False Negative (FN)*	*True Negative (TN)*

Based on the confusion matrix, we have the following metrics:

- Accuracy: reflects the overall correctness of the learning scheme.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \tag{5.1}$$

- Precision of Good class (P_{Good}): reflects the correctness of the learning scheme on positive class.

$$P_{Good} = \frac{TP}{TP + FP} \tag{5.2}$$

- Precision of Bad class (P_{Bad}): reflects the correctness of the learning scheme on negative class.

$$P_{Bad} = \frac{TN}{TN + FN} \tag{5.3}$$

- Recall of Good class (R_{Good}): reflects the accuracy among the Good instances.

$$R_{Good} = \frac{TP}{TP + FN} \tag{5.4}$$

- Recall of Bad class (R_{Bad}): reflects the accuracy among the Bad instances.

$$R_{Bad} = \frac{TN}{TN + FP} \tag{5.5}$$

- F-measure: is a combination between Precision and Recall measures. And we have two types of F-measure as follow:

$$F_{Good} = \frac{2 P_{Good} R_{Good}}{P_{Good} + R_{Good}} \tag{5.6}$$

$$F_{Bad} = \frac{2 P_{Bad} R_{Bad}}{P_{Bad} + R_{Bad}} \tag{5.7}$$

For our system which is also a 2-class classification problem (good dialog and bad dialog), we choose two measures to evaluate our system performance are Accuracy and Recalls of "Good" and "Bad" for the following purposes:

- Accuracy is used to monitor the correctness over the overall performance of the system but we cannot keep track how well our learning scheme work on each class. Therefore, we need one more measure to do this.

- And Recalls of Good class and Bad class are the measures that we choose to track down the system performance in details.

5.3 Data validation

This section presents the data validation of two corpora used in our experiment: DARPA 2000 Communicator Corpus, and DARPA 2001 Communicator Corpus.

First, there are 662 dialogs in DARPA 2000. After preprocessing the corpus by removing dead dialogs, damaged dialogs and unusable dialogs which lack information needed to extract features, only 550 dialogs remained. Then, applying the inverse Likert-scale as mentioned in section 3.5 with the threshold is 12, we obtain 274 Bad dialogs and 276 Good

dialogs. We see that the distribution of Bad dialogs and Good dialogs on the corpus is almost equal to 50% and 50% respectively. Thus, we can use a baseline accuracy of 50.00% for the DARPA 2000 Corpus.

The second corpus has 1139 dialogs from DARPA 2001. As for the DARPA 2000 corpus, after preprocessing these dialogs, number of remaining dialogs was 1022. For this corpus, we applied the Likert-scale instead of inverse one with the threshold is 17. Finally, we got 472 Bad dialogs, and 550 Bad dialogs. So, the distribution of this corpus is 46.18% and 53.82% on the Bad dialogs and Good dialogs respectively. For this corpus, we take 53.82% as baseline accuracy with the assumption that predictor always guesses the majority class.

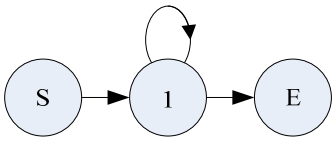
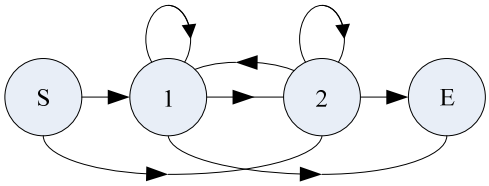
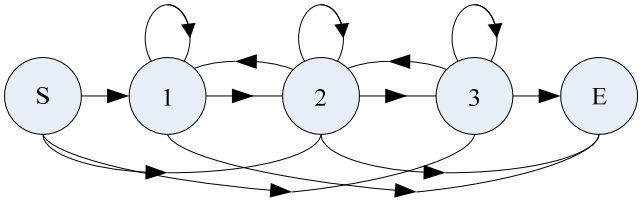
Table 5.2 Statistic on DARPA 2000 and 2001 corpora

Corpus	**Total dialogs**	**Removed dialogs**	**Remaining dialogs**	**Bad dialogs**	**Good dialogs**
DARPA 2000	662	112	550	274 **49.82%**	276 **50.18%**
DARPA 2001	1139	117	1022	472 **46.18%**	550 **53.82%**

5.4 Experimentation

HMM have many different types of configuration, depending on the number of states on each model, the number of transition on each state and the probability density of the observations. Combination of the conditions will provide various classifiers. In our study, we decided to use three different HMM topologies shown in Table 5.3:

Table 5.3 Summarization of three types of HMM used in our system

Alias	HMM Topology	Description
1-state		Number of states: 1 Forward transition
2-state half ergodic		Number of states: 2 Forward transition Backward transition
3-state half ergodic		Number of states: 3 Forward transition Backward transition

Note that we will use the alias as the name of each appropriate HMM in this literature. Of course, these aliases are only valid in the scope of this thesis.

In what follows, we describe the different experimentations we have done. The two first experiments addresses the different topologies of HMM and improving system performance using the Gaussian mixture in HMMs respectively. In the second experiment, we continue to use the same HMM topologies but number of Gaussians at each HMM state will be increased.

Finally, we do experiments to evaluate how important each feature is in our dialog classification system. To do so, we will choose a HMM topology with the best performance and the most reliable. Each feature, in turn, will be taken off and we will use the remaining ones for dialog classification.

5.5 Results and Interpretation

This section presents the results of our experiments that we carried out in this work. We also give interpretation and compare our results to the most recent ones tested on the same corpus as ours: the paper by Helen Wright (Helen Wright Hastie et al., 2002) and the master thesis (Truong Le Hoang, 2008). We have already reviewed both of the works in section 2.2.1.

In this section, in order to facilitate our presentation of the result, we denote:

- R_{Good} : Recall of Good (%).
- R_{Bad} : Recall of Bad (%).
- P_{Good} : Precision of Good (%).
- P_{Bad} : Precision of Bad (%).
- F_{Good} : F-measure of Good (%).
- F_{Bad} : F-measure of Bad (%).
- Avg : Average of ten folds.
- Acc : Accuracy of classification.

5.5.1 Effect of using the different topologies in HMMs

As discussed above, the first experiments are carried on HMMs having different number of states. HMMs with one, two and three states in combination with a normal distribution Gaussian for the probability density function at each state are used and below are the results of the experiments.

Table 5.4 Experiment result on the 1-state HMM

DARPA 2000 CORPUS											
Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
R_{Good}	44.00	65.62	63.33	66.66	51.51	60.00	85.71	48.14	75.00	77.77	**63.77**
R_{Bad}	86.66	78.26	88.00	85.71	81.81	84.00	67.64	75.00	74.19	92.85	**81.41**
P_{Good}	73.33	80.76	86.36	81.81	80.95	81.81	62.06	65.00	69.23	91.30	**77.26**
P_{Bad}	65.00	62.06	66.66	72.72	52.94	63.63	88.46	60.00	79.31	81.25	**69.20**
F_{Good}	55.00	72.41	73.07	73.46	62.96	69.23	71.99	55.31	72.00	83.99	**69.87**
F_{Bad}	74.28	69.23	75.86	78.68	64.28	72.41	76.66	66.67	76.66	86.67	**74.81**
Acc	**67.27**	**70.90**	**74.54**	**76.36**	**63.63**	**70.90**	**74.54**	**61.81**	**74.54**	**85.45**	**72.00**
DARPA 2001 CORPUS											
Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
R_{Good}	80.85	76.78	82.00	98.36	76.36	71.92	89.79	85.71	84.37	81.13	**82.73**
R_{Bad}	52.72	34.78	57.69	56.09	57.44	64.44	41.50	45.65	42.10	73.46	**52.59**
P_{Good}	59.37	58.90	65.07	76.92	67.74	71.92	58.66	65.75	71.05	76.78	**67.22**
P_{Bad}	76.31	55.17	76.92	95.83	67.50	64.44	81.48	72.41	61.53	78.26	**72.98**
F_{Good}	68.46	66.66	72.56	86.33	71.79	71.92	70.96	74.41	77.14	78.90	**74.17**
F_{Bad}	62.36	42.66	65.93	70.76	62.08	64.44	55.00	55.99	50.00	75.78	**61.13**
Acc	**65.68**	**57.84**	**69.90**	**81.37**	**67.64**	**68.62**	**64.70**	**67.64**	**68.62**	**77.45**	**68.92**

Table 5.5 Experiment result on the 2-state half ergodic HMM

DARPA 2000 CORPUS											
Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
R_{Good}	72.00	84.37	80.00	81.48	75.75	80.00	80.95	74.07	79.16	88.88	**79.66**
R_{Bad}	86.66	82.60	88.00	78.57	68.18	76.00	70.58	71.42	70.96	78.57	**77.15**
P_{Good}	81.81	87.09	88.88	78.57	78.12	80.76	62.96	71.42	67.85	80.00	**77.67**
P_{Bad}	78.78	79.16	78.57	81.48	65.21	76.00	85.71	74.07	81.48	88.00	**78.84**
F_{Good}	76.59	85.71	84.21	80.00	76.92	80.38	70.83	72.72	73.07	84.21	**78.65**
F_{Bad}	82.53	80.85	83.01	80.00	66.66	76.00	77.41	72.72	75.86	83.01	**77.99**
Acc	**80.00**	**83.63**	**83.63**	**80.00**	**72.72**	**78.18**	**74.54**	**72.72**	**74.54**	**83.63**	**78.36**
DARPA 2001 CORPUS											
Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
R_{Good}	80.85	75.00	78.00	93.44	72.72	71.92	91.83	82.14	82.81	81.13	**80.98**
R_{Bad}	54.54	34.78	61.53	56.09	63.82	77.77	47.16	58.69	50.00	71.42	**57.58**
P_{Good}	60.31	58.33	66.10	76.00	70.17	80.39	61.64	70.76	73.61	75.43	**69.27**
P_{Bad}	76.92	53.33	74.41	85.18	66.66	68.62	86.20	72.97	63.33	77.77	**72.54**
F_{Good}	69.09	65.62	71.56	83.82	71.42	75.92	73.77	76.03	77.94	78.18	**74.67**
F_{Bad}	63.82	42.10	67.36	67.64	65.21	72.91	60.97	65.06	55.88	74.46	**64.20**
Acc	**66.66**	**56.86**	**69.60**	**78.43**	**68.62**	**74.50**	**68.62**	**71.56**	**70.58**	**76.47**	**70.19**

Table 5.6 Experiment result on the 3-state half ergodic HMM

DARPA 2000 CORPUS											
Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
R_{Good}	56.00	75.00	73.33	81.48	60.60	66.66	76.19	81.48	79.16	74.07	**72.40**
R_{Bad}	83.33	78.26	72.00	67.85	72.72	64.00	70.58	71.42	64.51	64.28	**70.89**
P_{Good}	73.68	82.75	75.86	70.96	76.92	68.96	61.53	73.33	63.33	66.66	**71.40**
P_{Bad}	69.44	69.23	69.23	79.16	55.17	61.53	82.75	80.00	80.00	72.00	**71.85**
F_{Good}	63.63	78.68	74.57	75.86	67.79	67.79	68.08	77.19	70.37	70.17	**71.90**
F_{Bad}	75.75	73.47	70.59	73.07	62.74	62.74	76.18	75.47	71.42	67.92	**71.37**
Acc	**70.90**	**76.36**	**72.72**	**74.54**	**65.45**	**65.45**	**72.72**	**76.36**	**70.90**	**69.09**	**71.45**
DARPA 2001 CORPUS											
Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
R_{Good}	78.72	69.64	84.00	85.24	69.09	64.91	79.59	62.50	62.50	79.24	**73.54**
R_{Bad}	52.72	45.65	57.69	63.41	71.73	73.33	54.71	56.52	57.89	67.34	**60.10**
P_{Good}	58.73	60.93	65.62	77.61	74.50	75.51	61.90	63.63	71.42	72.41	**68.23**
P_{Bad}	74.35	55.26	78.94	74.28	66.00	62.26	74.35	55.31	47.82	75.00	**66.36**
F_{Good}	67.27	64.99	73.68	81.25	71.69	69.81	69.64	63.06	66.66	75.67	**70.79**
F_{Bad}	61.69	50.00	66.66	68.42	68.75	67.34	63.04	55.91	52.38	70.96	**63.08**
Acc	**64.70**	**58.82**	**70.80**	**76.47**	**70.29**	**68.62**	**66.66**	**59.80**	**60.78**	**73.52**	**67.02**

Based on the results obtained, we see that the best HMM topology for both of two corpora is the 2-state HMM. The results are 78.36% and 70.19% on DARPA 2000 and DARPA 2001 respectively. We make some assumptions in order to explain the best results achieved on the 2-state HMM topology:

- Our problem is 2-category classification (GOOD dialog and BAD dialog). So, we think that a 2-state HMM is the most suitable to model our problem because there are two main

types of emotion: negative (or positive) and neutral in context of spoken dialogs (Vidrascu et al., 2005) then each state of 2-state HMM could presents a type of emotion. In other words, of these two states, there might be one state which contains salient emotion whereas the other one might be non salient emotion.

- For 1-state HMM, this model is too simple that we can model the emotions within a dialogue in case of the Gaussian normal distribution used to model the emission probability density function. But if we use the Gaussian Mixture Models instead of normal distribution, we will see that the 1-state HMM works very well (Section 5.5.2).

- For 3-state HMM, there could be too many states in order to model a 2-category classification. This might cause an ineffective classification as we saw the result (Table 5-6).

We also look at the deviation in accuracy of these experiments above. Generally, Table 5.6 gives you the average deviation in accuracy of the experiments on DARPA 2000 and DARPA 2001 using three distinct HMM topologies (1-state HMM, 2-state HMM and 3-state HMM).

Table 5.7 Standard deviation in accuracy

	1-state HMM	**2-state HMM**	**3-state HMM**
DARPA 2000	5.09%	3.28%	3.08%
DARPA 2001	4.37%	4.12%	4.89%

In detail, we plotted a graph (Figure 5.1) which shows all the results in accuracy of six experiments. In such a graph, from left to right, three first columns are DARPA 2000 (1-state HMM, 2-state HMM and 3-state HMM respectively), and three last columns are DARPA 2001 (1-state HMM, 2-state HMM and 3-state HMM respectively).

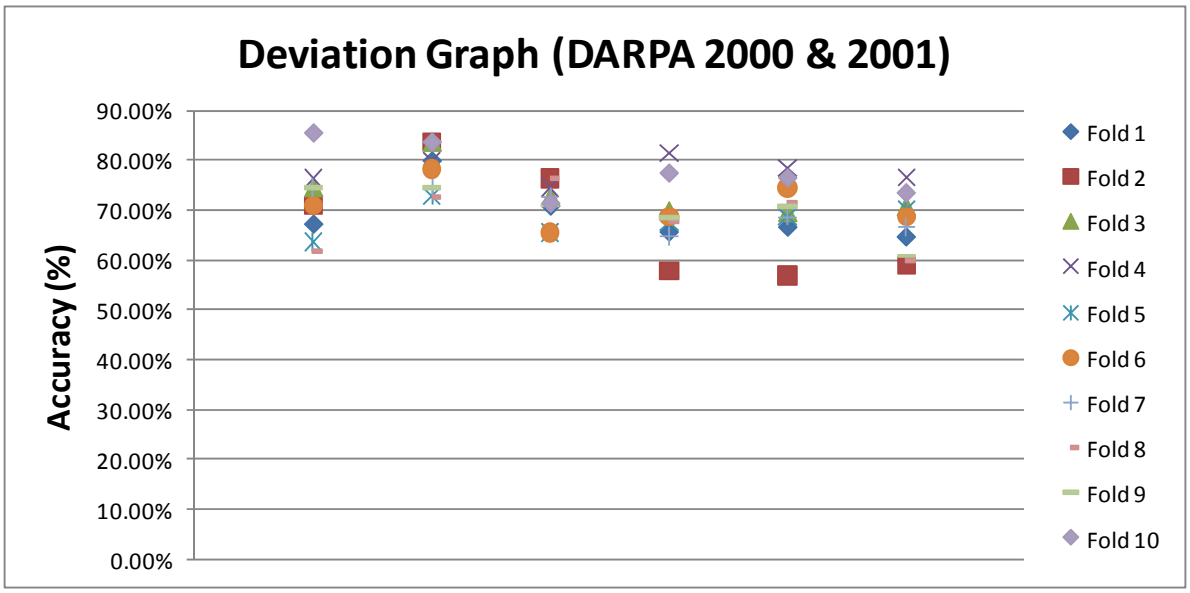


Figure 5.1 Deviation Graph on DARPA 2000 and DARPA 2001.

We can see that the deviations in all the experiments aren't small and close. These deviations show that our system is over fitting because the system makes a good classification on training samples but it cannot do classification well on novel samples. In other words, it means that in some cases, our system cannot learn well enough and then the classification result is worse than average. It seems that this problem is caused by the limitation of the training corpora. This is a limitation of our work that we need to improve in the future work.

5.5.2 Effect of using the Gaussian mixture in HMMs

The experimentation comes from our desire of improving the system performance. And using mixtures of Gaussian (Reynolds et al., 1995; Reynolds, 1995; Reynolds et al., 2000) to model the emission probability density functions in HMMs is the method that we choose in order to do so. The data whose probability density functions is unimodal and symmetric could be modeled by a normal distribution Gaussian. However, in many real problems, there are many kinds of data that cannot adequately be modeled using only one Gaussian. Instead, it is better to use combination of Gaussian distributions. A mixture of Gaussians is simply a weighted sum of K Gaussian densities defined by

$$gm(x) = \sum_{k=1}^{K} w_k \cdot g(\mu_k, \Sigma_k)(x)$$

where the weights are all positive and sum to one:

$$w_k \geq 0 \text{ and } \sum_{k=1}^{K} w_k = 1$$

and $g(\mu,\Sigma)(x)$ is the d-dimension Gaussian probability density function:

$$g(\mu,\Sigma)(x) = \frac{1}{\sqrt{2\pi}^d \sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

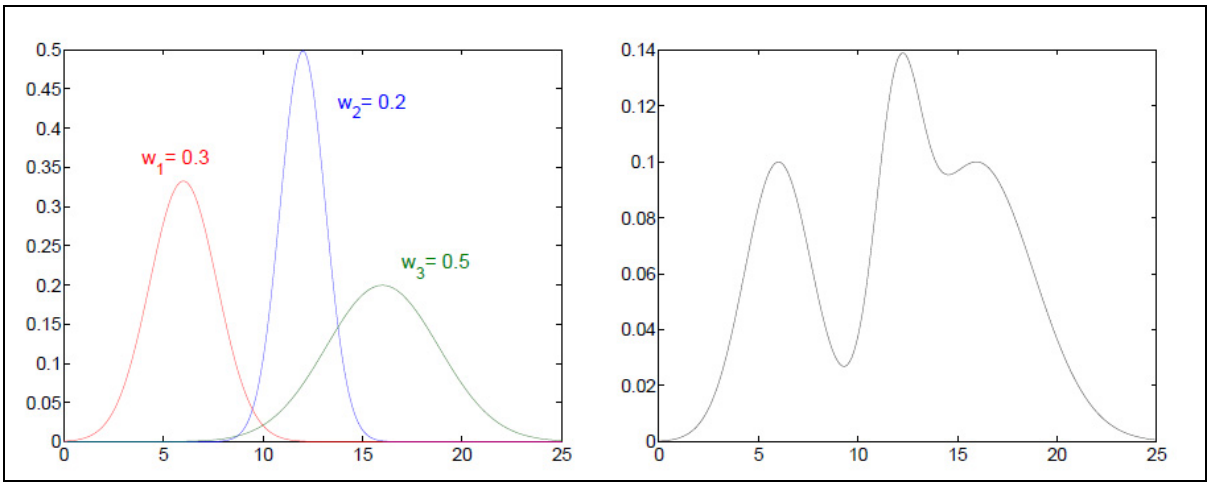


Figure 5.2 Example of 3-Gaussian mixture consisting of three single Gaussians weighted by w1, w2 and w3 respectively.
from Resch (2008)

In this work, we increase the number of Gaussians of each HMM state from 1 to 2, 4, 8, 16 and 32 in turn. These experiments showed better performances on both corpora as follow:

- For DARPA 2000, we obtained a classification accuracy of 79.27% (~1% better than

78.36%) for 3-state HMM with 32 Gaussian on each state.

- For DARPA 2001, we improved classification performance up to 71.37% (more 1% better than 70.19%) for 1-state HMM in combination with 4 mixtures for each state.

In comparison with previous work of Hoang (Truong Le Hoang, 2008), we see that our system is slightly better on both DARPA 2000 and DARPA 2001. In detail, on DARPA 2000, the best result from Hoang's work is 78.96% with F-measure of BAD of 0.75, whereas our best result reaches to 79.27% and F-measure of BAD of 0.77. Similarly, on DARPA 2001, the best result that we obtained is 71.37% with 0.64 of F-measure of BAD against 69.91% (0.61 of F-measure of BAD) from Hoang's work.

The tables 5.8, 5.9 and 5.10 will give the results in more details.

Table 5.8 Result of using Gaussian mixture on 1-state HMM

Corpus	N° of Gaussians	1	2	4	8	16	32
DARPA 2000	R_{Good}	63.77	**78.33**	93.73	92.79	96.33	94.99
	R_{Bad}	81.41	**73.90**	53.39	57.11	48.69	52.39
	P_{Good}	77.26	**74.93**	66.79	68.33	65.26	66.76
	P_{Bad}	69.20	**77.48**	90.13	89.04	93.56	91.61
	F_{Good}	69.87	**76.59**	78.00	78.70	77.81	78.41
	F_{Bad}	74.81	**75.65**	67.06	69.59	64.05	66.66
	Accuracy	**72.00 ±5.09**	**76.00 ±4.00**	**73.45 ±5.45**	**74.90 ±4.00**	**72.36 ±4.43**	**73.63 ±3.45**
DARPA 2001	R_{Good}	82.73	84.62	**82.73**	82.37	88.39	95.58
	R_{Bad}	52.59	52.96	**58.12**	54.87	45.19	29.55
	P_{Good}	67.22	67.78	**69.79**	68.20	65.60	61.27
	P_{Bad}	72.98	74.96	**74.62**	73.17	77.04	87.23
	F_{Good}	74.17	75.27	**75.71**	74.62	75.31	74.67
	F_{Bad}	61.13	62.07	**65.34**	62.71	56.97	44.15
	Accuracy	**68.92 ±4.37**	**70.00 ±4.78**	**71.37 ±5.53**	**69.60 ±4.33**	**68.43 ±4.19**	**65.00 ±4.55**

Table 5.9 Result of using Gaussian mixture on
2-state half ergodic HMM

Corpus	Nº of Gaussians	1	2	4	8	16	32
DARPA 2000	R_{Good}	**79.66**	84.13	95.42	95.56	86.63	96.59
	R_{Bad}	**77.15**	67.73	46.01	49.46	62.42	45.40
	P_{Good}	**77.67**	72.13	63.94	65.50	69.69	63.96
	P_{Bad}	**78.84**	81.11	91.44	91.95	82.84	93.97
	F_{Good}	**78.65**	77.67	76.57	77.72	77.24	76.96
	F_{Bad}	**77.99**	73.82	61.22	64.32	71.19	61.22
	Accuracy	**78.36 ±3.82**	**75.81 ±2.62**	**70.36 ±5.49**	**72.36 ±5.82**	**74.54 ±4.00**	**70.90 ±5.82**
DARPA 2001	R_{Good}	**80.98**	81.43	82.25	77.10	82.65	91.60
	R_{Bad}	**57.58**	56.06	54.13	53.46	49.65	38.17
	P_{Good}	**69.27**	68.50	67.75	65.99	65.80	63.36
	P_{Bad}	**72.54**	73.03	72.44	66.77	71.81	80.62
	F_{Good}	**74.67**	74.41	74.30	71.11	73.27	74.91
	F_{Bad}	**64.20**	63.43	61.96	59.38	58.71	51.81
	Accuracy	**70.19 ±4.12**	**69.80 ±4.86**	**69.21 ±5.96**	**66.26 ±4.86**	**67.45 ±3.76**	**66.76 ±5.59**

Table 5.10 Result of using Gaussian mixture
on 3-state half ergodic HMM

Corpus	N° of Gaussians	1	2	4	8	16	32
DARPA 2000	R_{Good}	72.40	65.64	75.21	83.59	85.05	**87.40**
	R_{Bad}	70.89	73.67	65.31	62.98	67.49	**71.52**
	P_{Good}	71.40	71.56	68.61	69.32	72.66	**75.43**
	P_{Bad}	71.85	67.66	72.44	79.09	81.70	**84.55**
	F_{Good}	71.90	68.47	71.76	75.79	78.37	**80.98**
	F_{Bad}	71.37	70.54	68.69	70.12	73.92	**77.49**
	Accuracy	**71.45 ±2.86**	**69.45 ±4.65**	**69.81 ±3.40**	**73.09 ±4.07**	**76.00 ±4.14**	**79.27 ±4.73**
DARPA 2001	R_{Good}	73.54	71.97	73.21	65.55	68.52	**80.34**
	R_{Bad}	60.10	64.34	60.89	67.50	67.39	**55.96**
	P_{Good}	68.23	70.38	68.92	70.06	71.33	**68.17**
	P_{Bad}	66.36	66.23	66.34	63.11	65.03	**71.86**
	F_{Good}	70.79	71.17	71.00	67.73	69.90	**73.76**
	F_{Bad}	63.08	65.27	63.50	65.23	66.19	**62.92**
	Accuracy	**67.02 ±4.89**	**68.50 ±4.43**	**67.51 ±3.68**	**66.53 ±4.44**	**68.10 ±4.58**	**69.18 ±5.13**

Before going to the next section, we summarize the best results of ours in comparison with those of state of art in Table 5.11. Compared with Hoang's results, our system totally performs better on both corpora. But when compared to Hastie's results, ours just overcome his on identification of BAD dialogs due to F-measure on Bad. Since she didn't mention the accuracy of his experiment, we cannot compare our results to hers. The comparison shows that our system produces a good performance on identification of problematic dialogues.

Table 5.11 Results in comparison with those of state of art

CORPUS	Evaluation measure (%)	Baseline	Our best results	Hoang's results (Hoang, 2008)	Hastie's results (Hastie, 2002)
DARPA 2000	Recall on Bad	-	84.55	-	-
	F-measure on Bad	-	**77.12**	75.50	-
	Accuracy	50.18	**79.27**	78.60	-
DARPA 2001	Recall on Bad	-	58.12	-	66.70
	Recall on Good	-	82.73	-	88.50
	Precision of Bad	-	**74.62**	-	58.50
	Precision of Good	-	69.79	-	81.30
	F-measure on Bad	-	**64.87**	61.00	62.33
	F-measure on Good	-	75.71	-	84.74
	Accuracy	53.82	**71.37**	69.52	-

5.5.3 Selection of feature

As mentioned above, this experiment is run so that we can assess the role of each feature in our system performance. To see how each feature affects classification performance, we will remove each feature in turn and do classification with the remaining ones. Since the feature set has 10 distinct features, we will have totally 10 experiments carried out. The HMM topology we choose to run the experiment is 2-state HMM with one Gaussian for each state. We use this topology because it obtained the best performance on both DARPA 2000 and DARPA 2001 as well. The results of the experiments are shown in the Table 5.12 and can be interpreted as follow.

- The last row is the result tested by the 2-state HMM (one mixture on each state) and using 10-feature set. We put it here with a role as base line so that we can easily compare it to the results that will be tested by the same model but using 9-feature set (one feature will be removed).

- The other rows are results which are achieved by using 9-feature set. (E.g. the second row is an experiment without the feature "Utterance Position").

Table 5.12 Result of evaluating each feature

Corpus	**DARPA 2000**			**DARPA 2001**		
Metric	*Recall of Good*	*Recall of Bad*	*Accuracy (Deviation)*	*Recall of Good*	*Recall of Bad*	*Accuracy (Deviation)*
Utterance Position	75.64%	76.63%	76.00% (-2.36)	68.00%	62.97%	**63.98% (-6.21)**
Utterance Duration	75.85%	79.62%	77.45% (-0.91)	76.56%	60.94%	67.05% (-3.14)
Number of Phonemes	75.05%	77.36%	76.18% (-2.18)	77.38%	62.43%	68.52% (-1.67)
Inverse Speech Rate	77.05%	73.73%	75.09% (-3.27)	73.96%	62.73%	67.64% (-2.55)
Num Negative Words	76.46%	72.04%	**74.00% (-4.36)**	76.91%	59.18%	**65.49% (-4.70)**
Num Positive Words	68.87%	78.31%	**72.36% (-6.00)**	65.26%	71.25%	66.76% (-3.43)
Response Waiting Time	79.66%	75.55%	77.45% (-0.91)	76.42%	59.70%	**66.07% (-4.12)**
Repetition Rate	77.80%	77.42%	77.39% (-0.97)	76.41%	61.36%	67.45% (-2.74)
Silence Time	72.08%	79.40%	**73.63% (-4.73)**	70.40%	66.50%	68.43% (-1.76)
ASR Accuracy	76.69%	76.99%	76.72% (-1.64)	76.27%	61.25%	67.25% (-2.94)
All features	**77.67%**	**78.84%**	**78.36%**	**69.27%**	**72.54%**	**70.19%**

From the result table (Table 5.12), we realize the important points as follows:

- For both corpora and based on the summation shown in Figure 5.3, we could see that all features (including Utterance Position, Num Negative Words and Num Positive Words) are important ones for both corpora.

- Whereas, if we take a look separately on distinct corpus, we will see that the role of each feature in each corpus is different. On DARPA 2000, our system depends much on Number of Negative Words, Number of Positive Words and Silence Time. On DARPA 2001, we see that Utterance Position, Number of Negative Words and Response Waiting Time are the important features for the classification performance. The fact that each feature plays different depending on different corpora because the DARPA 2000 corpus was collected in the experimental context meanwhile the DARPA 2001 corpus was build in the real context. We have mentioned the context of both corpora in section 3.4.

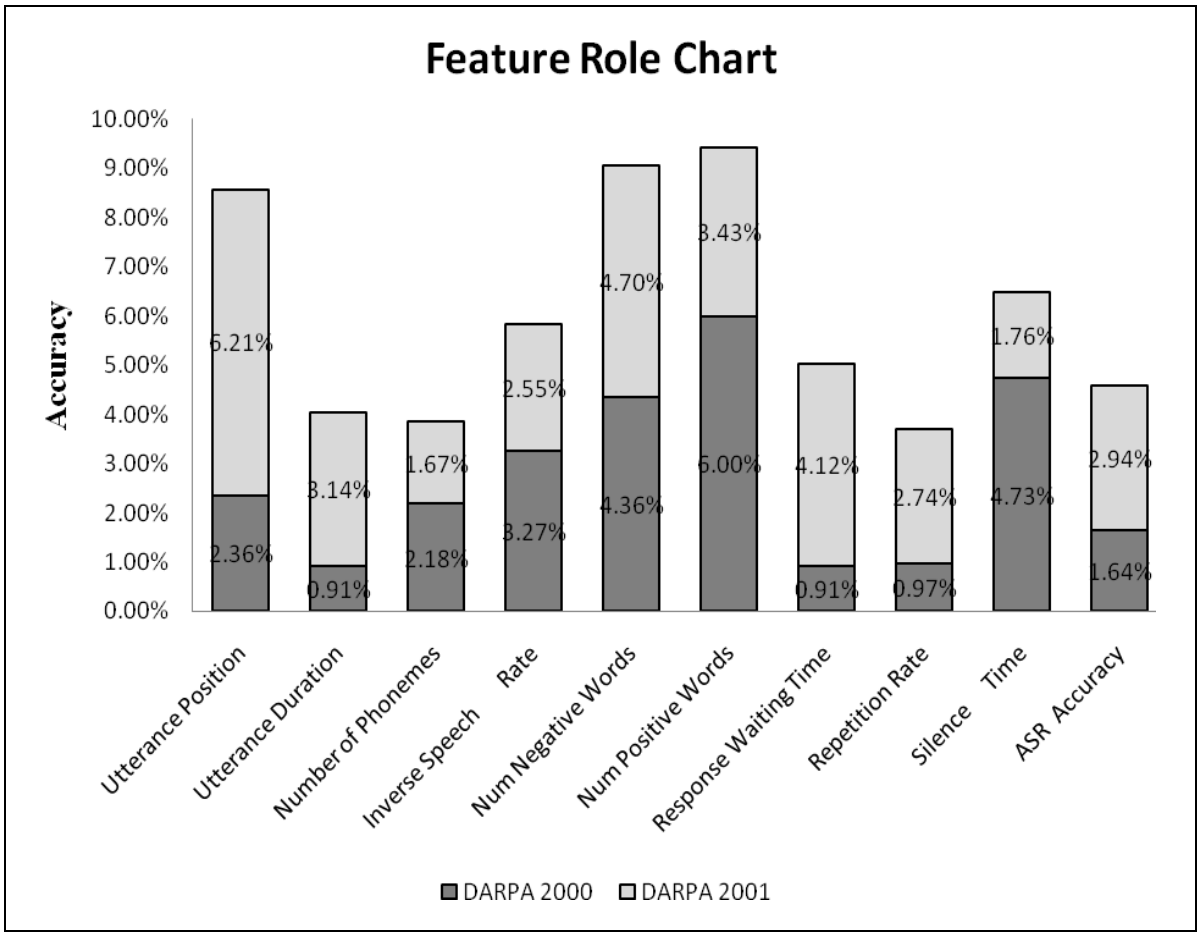


Figure 5.3 Feature role chart.

CHAPTER 6

CONCLUSION

6.1 Conclusion

In this work, we studied the spoken dialogue classification in the real context of telephonic call center. There are two types of dialogue that we try to classify: problematic dialogue (BAD dialog) and non-problematic dialogue (GOOD dialog). Labeling a dialogue is based on the user's satisfaction which was provided by the caller after a dialogue. Our work was motivated by the demand of building an automatic system for detecting problematic dialog in the project, namely "Managing emotions in Human-Computer Dialogs" of ÉTS and CRIM in collaboration with Bell Canada Corp.

Before, this problem was tackled by our colleague, Truong Le Hoang, in his master thesis. In his solution, he used a learning machine scheme named Decision Tree and used features at the dialog level as input to his system. In this work, we proposed a new approach to such a problem. We proposed to use features at the utterance level mined from each utterance in a dialogue as input of our system. We used another machine learning paradigm named Hidden Markov Model to model the user state of satisfaction throughout the dialog in order to identify problematic dialogues.

Our final results are a set of features and a topology of Hidden Markov Model that could be used to classify dialogues efficiently. The feature set includes ten distinct features and most of them can be extracted automatically (except one feature named "ASR Accuracy Rate") and useful for a real application. Our experiments show that a HMM topology with two states is suitable for the dialogue classification for a 2-class problem. Besides, combination of GMM for modeling the observation probability density function for each HMM state is a good idea to improve the classification performance.

Through this work, we also found out that the feature set plays an important role in the

pattern classification system. We also noticed that the performance of pattern classification also depends on dataset. This makes difficult to compare the research's results because they do not often use the same dataset.

6.2 Future work

In the future, we will address the following points:

- Continuing to improve the system's performance by adding new features. Most of the features in this work come from the component of dialog management. Thus, we still have two more components in the Human-Machine Dialog System, namely Acoustic/ASR and NLU component where we're able to discover new interesting features for our system such as the grammar used by the recognizer, a measure of the shift in context between utterances (Walker et al., 2000). We also think about using the Mel-Frequency Cepstral Coefficients (MFCCs) extracted from the acoustic signal of the utterance as new features to improve the system.

- Proposing an approach in order to solve the problem of prediction of problematic dialogues in the situation of Human-Machine Dialog System. Prematurely predicting problematic dialogues prior to the occurrence is a strategy that the Human-Machine Dialog System is interested in because this could be very useful for real applications. Since our current system is quick in responding to each utterance in the dialog, we can develop an automatic system of problematic dialogues based upon this idea.

ANNEX I

FORWARD ALGORITHM

Scoring and Evaluation (Huang et al., 2002; Rabiner, 1989)

In order to calculate the probability $P(X|\Phi)$ of the observation sequence $X = (X_{1,}, X_{2,}, ..., X_{T,})$, given a model Φ, we must sum up the probabilities of all possible state sequences of length T that generate the observation sequence X

$$P(X|\Phi) = \sum_{\text{all S}} P(S|\Phi)P(X|S,\Phi) \tag{A I-1}$$

For a given state sequence $S = (s_1, s_2, ..., s_T)$, where s_1 is the initial state, the probability of the state sequence can be computed by using the Markov assumption

$$P(S|\Phi) = P(s_1|\Phi)\prod_{2}^{T} P(s_t|s_{t-1},\Phi) = \pi_{s_1} a_{s_1 s_2} ... a_{s_{T-1} s_T} \tag{A I-2}$$

We also have the joint output probability can be rewritten by applying the output-independent assumption

$$P(X|S,\Phi) = P(X_1^T|S_1^T,\Phi) = \prod_{1}^{T} P(X_t|s_t,\Phi) = b_{s_1}(X_1) b_{s_2}(X_2) ... b_{s_T}(X_T) \tag{A I-3}$$

Substituting Eq. (A I-2) and (A I-3) to the right-hand side of the Eq. (A I-1), we have

$$P(X|\Phi) = \sum_{\text{all S}} \pi_{s_1} b_{s_1}(X_1) a_{s_1 s_2} b_{s_2}(X_2) ... a_{s_{T-1} s_T} b_{s_T}(X_T) \tag{A I-4}$$

Assume that we have a model with N states and T observations. According to the direct evaluation, there are N^T possible state sequences. In addition, each state sequence requires $2T$ calculations. So we finally reach approximately $2TN^T$ operations required to complete the

evaluation. Clearly, this is an unfeasible calculation, even for a small value of N and T; e.g., for $N = 5$ (states), $T = 100$ (observations), there are on the order of $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ computations. However, we have two efficient methods for computing the probability $P(X|\Phi)$ that are called Forward algorithm and Backward algorithm respectively.

The Forward algorithm (Huang et al., 2002) (Rabiner, 1989)

Let's define forward probability

$$\alpha_t(i) = P(X_1^t, s_t = i|\Phi) = P(X_1, X_2, ..., X_t, s_t = i|\Phi) \quad \text{(A I-5)}$$

as the probability of observation X_1 to X_t with the state sequence terminating in state $s_t = i$ given the model Φ. So we can solve for $\alpha_t(i)$ inductively, as follows:

THE FORWARD ALGORITHM

Step 1: Initialization

$$\alpha_1(i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$$

Step 2: Induction

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(X_t) \qquad 2 \le t \le T; \qquad 1 \le j \le N \quad \text{(A I-6)}$$

Step 3: Termination

$$P(X|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$$

With T observations and N states, this algorithm requires approximately $N^2 T$ operations. It's a considerable difference compared to $2TN^T$ operations as required by the direct calculation.

The key of the Forward algorithm is the induction step which is illustrated in Figure-A I-1. And the next figure (Figure-A I-2) shows the forward probability trellis diagram.

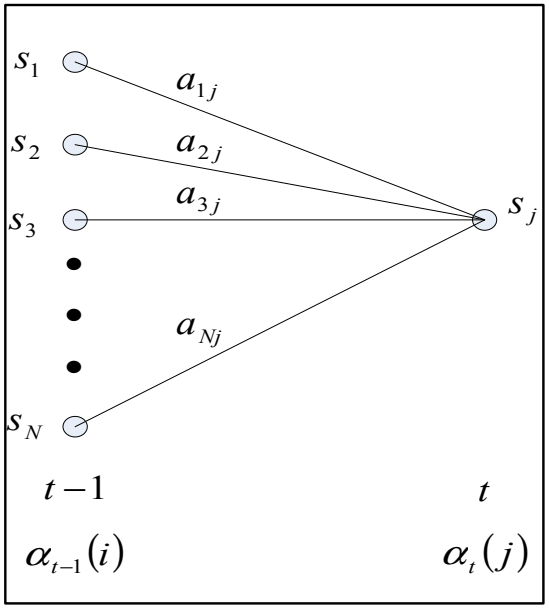


Figure-A I-1 Trellis diagram illustration for calculation of the forward probability $\alpha_t(i)$ at time *t* from the forward probability $\alpha_{t-1}(j), j = 1,2,...,N$.

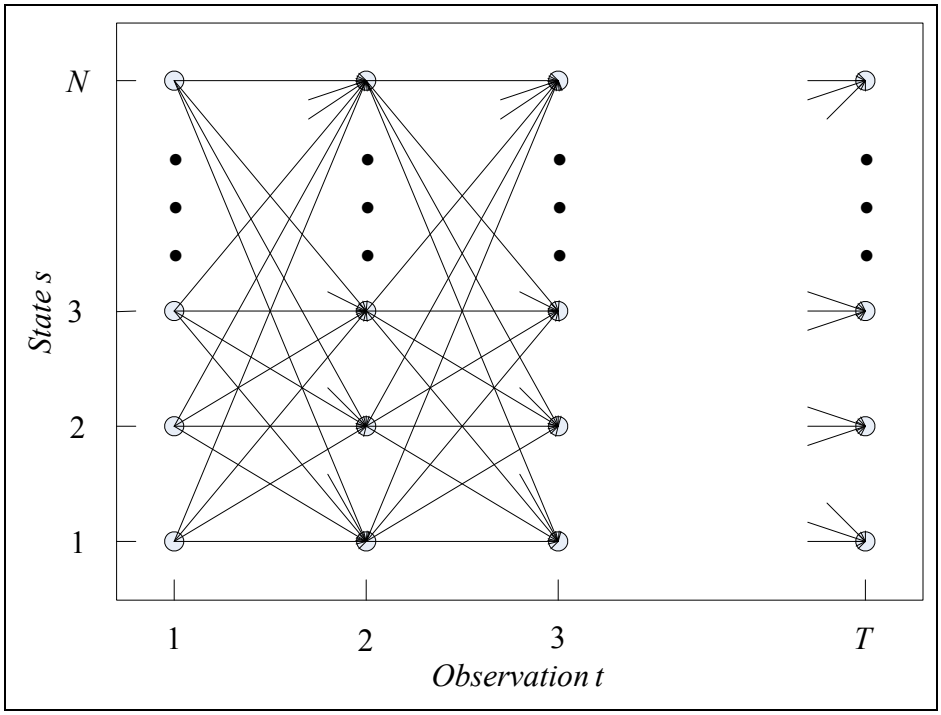
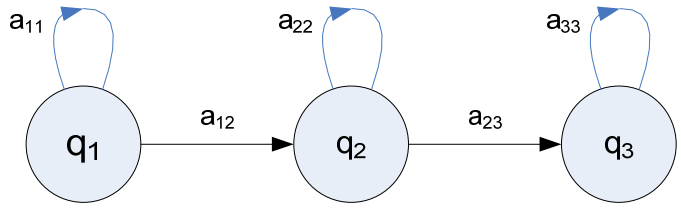


Figure-A I-2 Implementation of the computation of $\alpha_t(i)$ in term of a trellis of observation *t* and state *s*.

In order to understand better the algorithm, let's take a look at the following example that describes how to implement such an algorithm step by step. Consider the three state discrete observation density HMM which emits the colors red, green and blue. The HMM is described by the following parameters:

- The initial state probability vector $\pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$
- The transition probability matrix $A = \begin{bmatrix} 0.6 & 0.4 & 0.0 \\ 0.0 & 0.3 & 0.7 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$
- The output probability matrix $B = \begin{bmatrix} red & red & red \\ green & green & green \\ blue & blue & blue \end{bmatrix} = \begin{bmatrix} 0.80 & 0.65 & 0.20 \\ 0.15 & 0.10 & 0.30 \\ 0.05 & 0.25 & 0.50 \end{bmatrix}$
- And the topology of this HMM is described by the right below figure



Assume that the symbol sequence is observed as follow ***"X=RED, RED, BLUE"***. Using the Forward algorithm, determine the total probability, $P(X \mid \Phi)$, of observing this symbol sequence where $\Phi = (A, B, \pi)$.

The solution is shown as the following diagram.

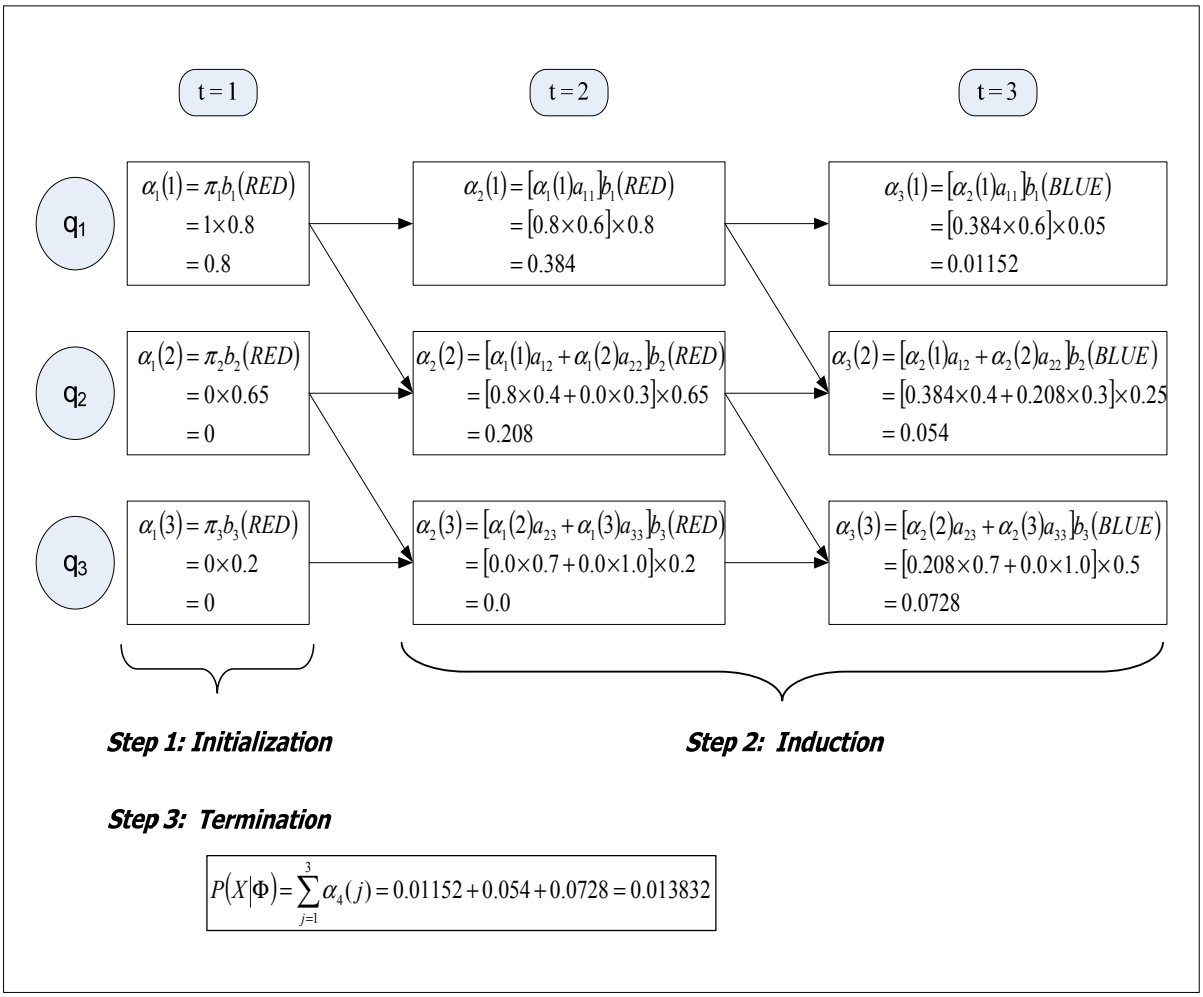


Figure-A I-3 Computing probability $P(X \mid \Phi)$ using the Forward algorithm.

The next annex is the other algorithm to calculate the probability $P(X|\Phi)$ in reverse direction called backward algorithm.

ANNEX II

BACKWARD ALGORITHM

Backward algorithm (Huang et al., 2002; Rabiner, 1989)

Let's define backward probability

$$\beta_t(i) = P\left(X_{t+1}^T, s_t = i \middle| \Phi\right) = P\left(X_{t+1}, X_{t+2}, ..., X_T, s_t = i \middle| \Phi\right) \quad \text{(A II-1)}$$

as the probability of observation X_{t+1} to X_T with the state sequence terminating in state $s_t = i$ given HMM Φ. So we can solve for $\beta_t(i)$ inductively, as follows:

THE BACKWARD ALGORITHM

Step 1: Initialization

$$\beta_1(i) = 1 \qquad 1 \le i \le N$$

Step 2: Induction

$$\beta_t(j) = \sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j) \qquad t = T-1, T-2, ..., 1 \qquad 1 \le j \le N \quad \text{(A II-2)}$$

Step 3: Termination

$$P(X \mid \Phi) = \sum_{i=1}^{N} \pi_i b_i(X_1) \beta_1(i)$$

Similarity to the forward algorithm, the backward requires only N^2T operations and the induction step is the key of the algorithm. Each step in the inductive equation (A II-2) is illustrated by Figure-A II-1.

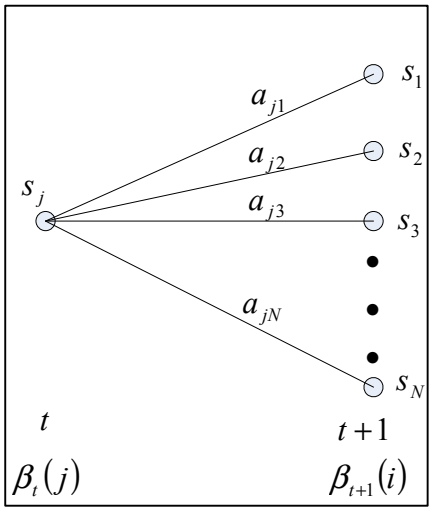


Figure-A II-1: Trellis diagram illustration for calculation of the backward probability $\beta_t(i)$ at time *t* from the back probability $\beta_{t+1}(j), j = 1,2,..., N$.

One more time, we will redo the same example which has been done for the Forward algorithm in the previous annex to see more how the backward algorithm works (Seeing the Figure-A II-2 for the solution using the Backward algorithm). One sure thing is that the Backward algorithm will get the same result as the Forward algorithm.

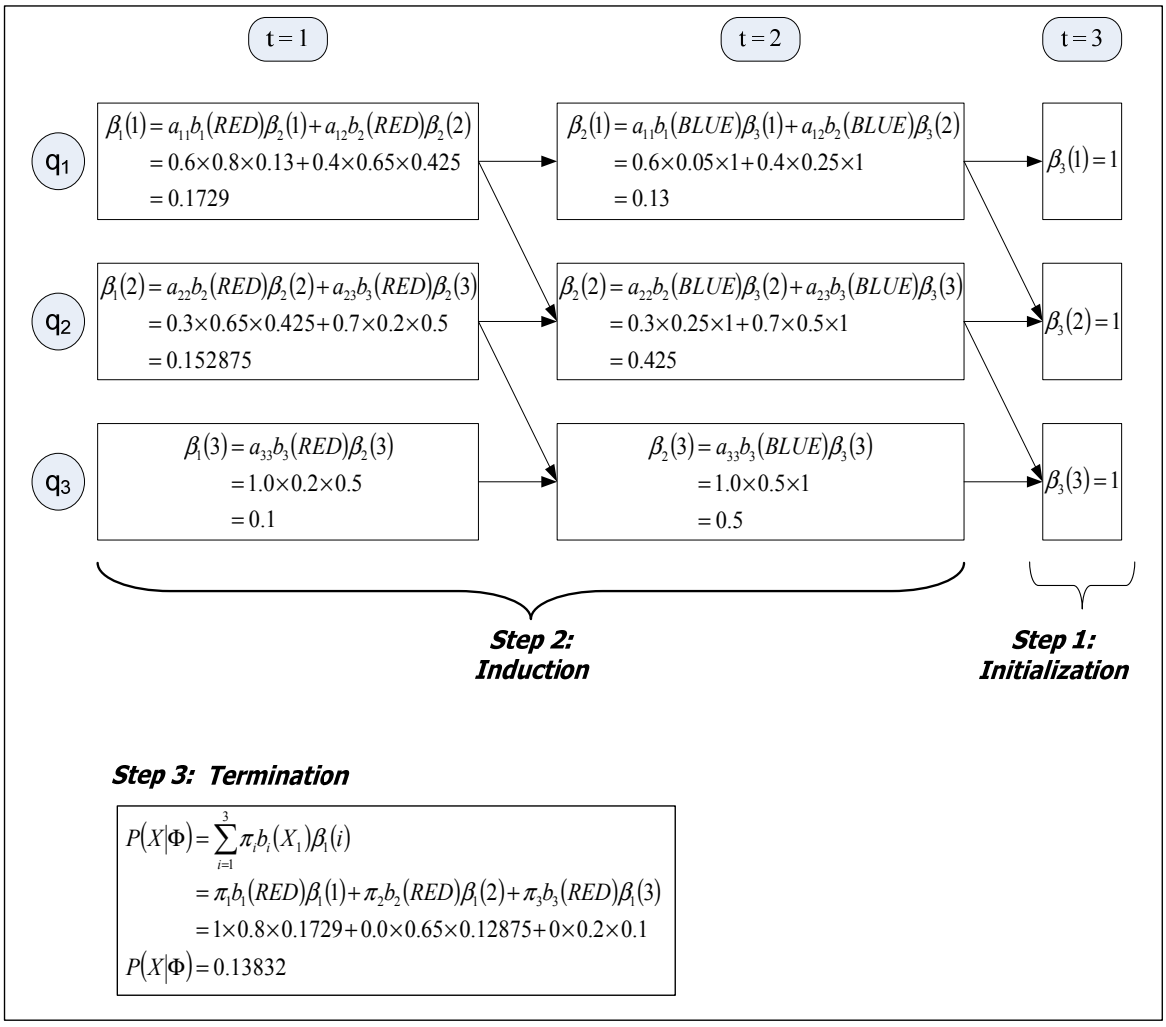


Figure-A II-2 Computing probability $P(X \mid \Phi)$ using the Backward algorithm.

ANNEX III

VITERBI ALGORITHM

Viterbi algorithm (Huang et al., 2002; Rabiner, 1989)

The scoring and evaluation section gives us the solution of computing the probability $P(X|\Phi)$ given the model Φ and the observation sequence $X = (X_1, X_2, ..., X_n)$ by summing up the probabilities of all the possible corresponding state sequences that can generate this observation sequence. But we cannot determine which state sequence produces the best probability. In this annex, we will introduce an algorithm called Viterbi to find the most likely state sequence that generates the observation sequence. In other words, we will look for the state sequence $S = (s_1, s_2, ..., s_T)$ that maximizes the probability $P(S, X|\Phi)$.

The main idea of the Viterbi algorithm is similar to the one of the Forward and Backward algorithms (Figure-A III-1). Instead of summing up probabilities that come from different paths to the same state s_i at time t, the Viterbi algorithm takes and remembers the best path only.

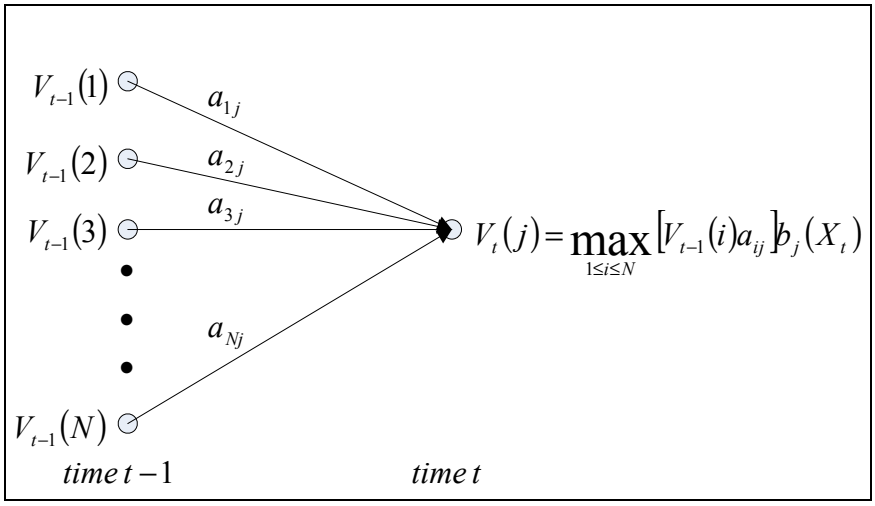


Figure-A III-1 Trellis diagram illustration for calculation of $V_t(i)$ at time *t* from $V_{t-1}(j), j = 1,2,..., N$.

Before going into details about the algorithm, let's define the best-path probability

$$V_t(i) = P\left(X_1^t, S_1^{t-1}, s_t = i \middle| \Phi\right) \qquad \text{(A III-1)}$$

as the probability of the most likely state sequence at time *t*, which generate the observation $X_1^t = (X_1, X_2, ..., X_t)$ (until time *t*) and ends in state *i*. An inductive procedure for the Viterbi algorithm can be described as follows:

THE VITERBI ALGORITHM

Step 1: Initialization

$$V_1(i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$$

$$B_1(i) = 0$$

Step 2: Induction

$$V_t(j) = \underset{1 \le i \le N}{Max}[V_{t-1}(i)a_{ij}]b_j(X_t) \qquad 2 \le t \le T; \qquad 1 \le j \le N \qquad \text{(A III-2)}$$

$$B_t(j) = \underset{1 \le i \le N}{Arg\max}[V_{t-1}(i)a_{ij}] \qquad 2 \le t \le T; \qquad 1 \le j \le N \qquad \text{(A III-3)}$$

Step 3: Termination

The best probability $P^* = \underset{1 \le i \le N}{\text{Max}}\left[V_T(i)\right]$

$$s_T^* = \underset{1 \le i \le N}{Arg\max}\left[V_T(i)\right]$$

Step 4: Backtracking

$$s_t^* = B_{t+1}(s_{t+1}^*) \qquad t = T-1, T-2, ..., 1$$

$S^* = (s_1^*, s_2^*, ..., s_T^*)$ is the best sequence

In this section, we also present an example of carrying this algorithm out step by step. In this example, we reuse the same HMM and parameter set as the previous annexes. Of course, the

question at this time is to determine the most likely sequence that could have generated the symbol sequence mentioned (***"O =RED, RED, BLUE"***) and the corresponding probability by using the Viterbi algorithm. Here is the question's solution.

Step 1: Initialization (at t=1)

$$V_1(1) = \pi_1 b_1(RED) = 1.0 \times 0.8 = 0.8$$
$$V_1(2) = \pi_2 b_2(RED) = 0.0 \times 0.65 = 0.0$$
$$V_1(3) = \pi_3 b_3(RED) = 0.0 \times 0.2 = 0.0$$
$$B_1(1) = B_1(2) = B_1(3) = 0$$

Step 2: Induction (at time $t = 2,3$ ***)***

Computations at time t = 2 and t = 3 are shown in the Table-A III-1.

Table-A III-1 Inductive steps at time t = 2 and t = 3

	t =2	***t =3***
State q_1		$V_2(1)a_{11} = 0.384 \times 0.6 = 02304$ $V_3(1) = \underset{1 \le i \le 1}{Max}[V_2(i)a_{i1}]b_1(BLUE)$ $= 0.2304 \times 0.5 = 0.01152$ $B_3(1) = \arg\max_{1 \le i \le 1}[V_2(i)a_{i1}] = 1$
State q_2	$V_1(1)a_{12} = 0.8 \times 0.4 = 0.32$ $V_1(2)a_{22} = 0.0 \times 0.3 = 0.0$ $V_2(2) = \underset{1 \le i \le 2}{Max}[V_1(i)a_{i2}]b_2(RED)$ $= [V_1(1)a_{12}]b_2(RED)$ $= 0.32 \times 0.65 = 0.208$ $B_2(2) = \arg\max_{1 \le i \le 2}[V_1(i)a_{i2}] = 1$	$V_2(1)a_{12} = 0.384 \times 0.4 = 0.1536$ $V_2(2)a_{22} = 0.208 \times 0.3 = 0.0624$ $V_3(2) = \underset{1 \le i \le 2}{Max}[V_2(i)a_{i2}]b_2(BLUE)$ $= 0.1536 \times 0.65$ $= 0.0384$ $B_3(2) = \arg\max_{1 \le i \le 2}[V_2(i)a_{i2}] = 1$
	t =2	***t =3***

State q_3	$V_1(2)a_{23} = 0.0 \times 0.7 = 0.0$ $V_1(3)a_{33} = 0.0 \times 1.0 = 0.0$ $V_2(3) = \underset{2 \le i \le 3}{Max}[V_1(i)a_{i3}]b_3(RED)$ $= 0.0 \times 0.5$ $= 0.0$ $B_2(3) = \arg\max_{2 \le i \le 3}[V_1(i)a_{i3}] = 2$	$V_2(2)a_{23} = 0.208 \times 0.7 = 0.1456$ $V_2(3)a_{33} = 0.0 \times 1.0 = 0.0$ $V_3(3) = \underset{2 \le i \le 3}{Max}[V_2(i)a_{i3}]b_3(BLUE)$ $= 0.1456 \times 0.5$ $= 0.0728$ $B_3(3) = \arg\max_{2 \le i \le 3}[V_2(i)a_{i3}] = 2$

Step 3: Termination

$$\text{The best probability } P^* = \max_{1 \le i \le 3}[V_3(i)]$$
$$= Max[0.01152, 0.0384, 0.0728]$$
$$P^* = 0.0728$$
$$s_3^* = Arg \max_{1 \le i \le 3}[V_3(i)] = Arg \max[V_3(1), V_3(2), V_3(3)] = 3$$

Step 4: Backtracking

$$s_2^* = B_3(s_3^*) = B_3(3) = 2$$

$$s_1^* = B_2(s_2^*) = B_2(2) = 1$$

$S^* = (s_1^*, s_2^*, s_3^*, s_4^*) = (1,2,3)$ is the best state sequence

The Figure-A III-2 summarizes up main steps in the Viterbi algorithm for this example.

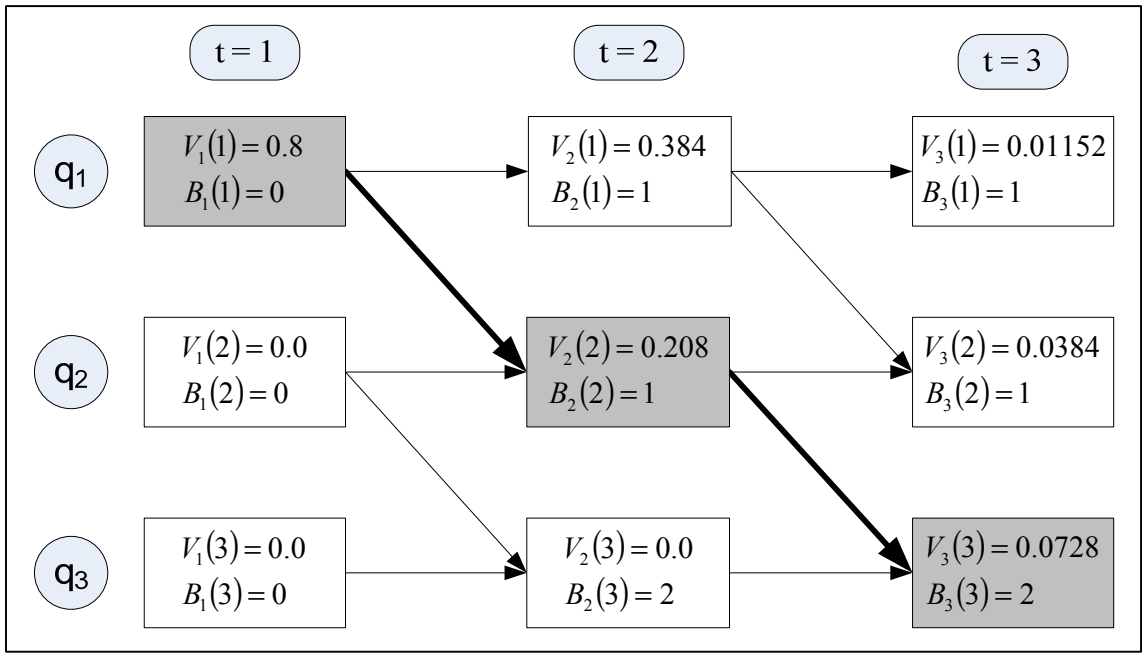


Figure-A III-2 Trellis of Viterbi solution.

ANNEX IV

FORWARD-BACKWARD ALGORITHM

Forward-Backward algorithm (Huang et al., 2002; Rabiner, 1989)

The first two HMM problems have a common hypothesis: given a model $\Phi = \{A, B, \pi\}$ in which all parameter sets are determined. Now, we must face the most difficult problem as well as the most important one in HMM. That is the reverse problem which is stated as: Given a model $\Phi = \{A, B, \pi\}$ in which all parameter sets are ***not*** determined yet and a set of observation sequences (training data), how to determine the HMM parameter set to maximize the joint likelihood probability $\prod_X P(X|\Phi)$.

In fact, we're not able to have a direct method to determine the HMM parameter set which maximizes the probability of the observation sequence. We can, however, optimize the model $\Phi = \{A, B, \pi\}$ using an iterative procedure called Baum-Welch algorithm (or Forward-Backward algorithm). Firstly, we define $\zeta_t(i, j)$ as the probability of being in state s_i at the time t and state s_j at the time $t+1$, given the model $\Phi = \{A, B, \pi\}$ and the observation sequence $X = (X_1, X_2, ..., X_T)$.

$$\xi_t(i, j) = P(s_t = i, s_{t+1} = j | X, \Phi) \qquad \text{(A IV-1)}$$

Figure-A IV-1 illustrates the sequence of events reaching to the conditions required by equation (A IV-1).

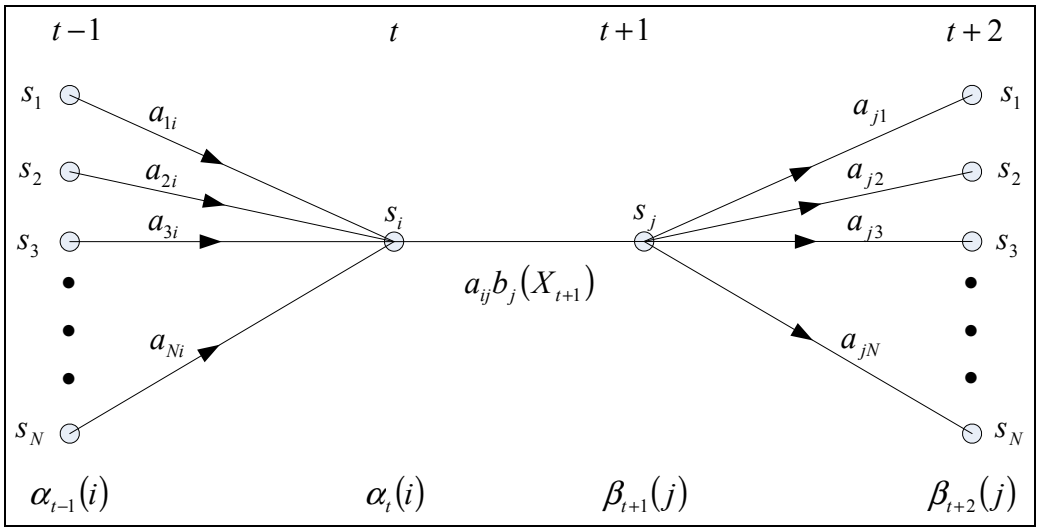


Figure-A IV-1 Illustration of the operations required for computation of $\xi_t(i,j)$ which is the probability of taking the transition from state s_i to state s_j at time $t+1$.

Based on the definition of forward and backward probability, we might rewrite $\xi_t(i,j)$ in the form as follow

$$\xi_t(i,j)=\frac{\alpha_t(i)a_{ij}b_j(X_{t+1})\beta_{t+1}(j)}{P(X|\Phi)}=\frac{\alpha_t(i)a_{ij}b_j(X_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(X_{t+1})\beta_{t+1}(j)} \qquad \text{(A IV-2)}$$

$$\sum_{t=1}^{T-1}\sum_{j=1}^{N}\xi_t(i,j)=\text{Expected number of transition from } s_i \qquad \text{(A IV-3)}$$

$$\sum_{t=1}^{T-1}\xi_t(i,j)=\text{Expected number of transition from } s_i \text{ to } s_j \qquad \text{(A IV-4)}$$

$$\hat{\pi}_i=\text{Expected number of time instances in state } s_i \text{ at time } (t=1)=\sum_{j=1}^{N}\xi_1(i,j) \qquad \text{(A IV-5)}$$

$$\hat{a}_{ij} = \frac{\text{Expected number of transition from } s_i \text{ to } s_j}{\text{Expected number of transition from } s_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N} \xi_t(i,j)} \qquad \text{(A IV-6)}$$

$$\hat{b}_j(k) = \frac{\text{Expected number of time in state } s_j \text{ and observing symbol } o_k}{\text{Expected number of times in state } s_j}$$

$$\hat{b}_j(k) = \frac{\sum_{t:x_t=o_k}^{T-1} \sum_{i}^{N} \xi_t(j,i)}{\sum_{t=1}^{T-1}\sum_{i}^{N} \xi_t(j,i)} \qquad \text{(A IV-7)}$$

Below is the forward-backward algorithm:

THE FORWARD-BACKWARD ALGORITHM

Step 1: Initialization

Choose an initial estimate $\Phi = \{A, B, \pi\}$

Step 2: Likelihood computation

Compute the likelihoods $\alpha_i(t)$ and $\beta_i(t)$ and the posterior probabilities $\xi_t(i, j)$ as defined above with $i, j = 1,2,..., N$ and $t = 1,2,..., T$

Step 3: Parameter update

Using the likelihood and posterior probabilities obtained in the step 2, compute $\hat{\Phi} = \{\hat{A}, \hat{B}, \hat{\pi}\}$ according to the re-estimation equations:

$$\hat{\pi}_i = \sum_{j=1}^{N} \xi_1(i, j) \qquad \hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i, j)} \qquad \hat{b}_j(k) = \frac{\sum_{t:x_t=o_k}^{T-1} \sum_{i}^{N} \xi_t(j, i)}{\sum_{t=1}^{T-1} \sum_{i}^{N} \xi_t(j, i)}$$

Step 4: Iteration

Set $\Phi = \hat{\Phi}$, repeat from step 2 until convergence

BIBLIOGRAPHY

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, Juin 5-8 2007).

Cheriet, Mohamed. 1997. SYS-821: Reconnaissance des formes et inspection: Méthodes syntaxiques et structurelles: notes du cours SYS-821. Programme de Maîtrise en genie de la production automatisée. Montréal: École de technologie supérieure.

Cohen, W. 1995. «Fast effective rule induction». In *Proceeding of Twelfth International Conference on Machine Learning*. (Tahoe City, California, USA, July 9-12 1995), p. 115-123. Morgan Kaufmann Publishers.

Cohen, W. 1996. «Learning trees and rules with set-valued features». In *Fourteenth Conference of the American Association of Artificial Intelligence*. (Portland, Oregon, August 4-8 1996), p. 709-716.

Duda Richard O., Peter E. Hart and David G. Stork. 1999. *Pattern Classification*, 2nd ed. Wiley-Interscience, 654 p.

Hastie, Helen Wright, Rashmi Prasad and Marilyn Walker. 2002. « What's the Trouble: Automatically Identifying Problematic Dialogues in DARPA Communicator Dialogue Systems ». In *Proceeding of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. (Philadelphia, June 2002), p. 384-391. Morristown (NJ): Association for Computational Lingustics Publishers.

Huang, Xuedong, Alex Acero and Hsiao-Wuen Hon. 2001. « Hidden Markov Models ». In *Spoken Language Processing*, 1st edition, p. 375-412. Upper Saddle River (NJ): Prentice Hall PTR.

Langkidle, Irene, Marilyn Walker, Jerry Wright, Allen Gorin, Diane Litman. 1999. « Automatic Prediction of Problematic Human-Computer Dialogue in 'How May I Help You?' ». In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, ASRUU99*. P. 369-372.

Lee, Chul Min, Shrikanth S. Narayanan and Roberto Pieraccini. 2002. « Combining Acoustic and Language Information for Emotion Recognition ». In *7th International Conference on Spoken Language Processing (IC SLP-2002)*. (Denver, Sept. 16-20 2002), p. 873-876.

Litman, Diane J., Marilyn A. Walker and Michael S. Kearn. 1999. « Automatic Detection of Poor Speech Recognition at the Dialogue Level». In *Proceedings of the 37th Annual*

Meeting of the Association for Computational Linguistics. (Maryland, June 20-26 1999), p. 309-316. Morristown (NJ): Association for Computational Lingustics Publishers.

Rabiner, Lawrence. 1989. « A Tutorial on Hidden Markov Models and Selected Application in Speech Recognition ». *Proceedings of the IEEE*, vol. 77, n° 66, February, p. 257-86.

Resch, B. Mixture of Gaussians: A tutorial for the course computational intelligence. http://www.igi.tugraz.at/lehre/CI. Consulted August 7th, 2010.

Reynolds, Douglas A. and Richard C. Rose. 1995. « Robust Text-Independent Speaker Identification using Gaussian Mixture Speaker Models ». *IEEE Transaction on Speech and Audio Processing*, vol. 3, n° 1, January, p. 72-83.

Reynolds, Douglas A.. 1995. « Speaker identification and verification using Gaussian mixture speaker models ». *Speech Communication*, vol. 17, n° 1-2, August, p. 91-108.

Reynolds, Douglas A., Thomas F. Quatieri and Robert B. Dunn. 2000. « Speaker Verification Using Adapted Gaussian Mixture Models ». *Digital Signal Processing,* vol. 10, n° 1-3, p. 19-41.

Schmitt, Alexander, Carolin Hank and Jackson Liscombe. 2008. « Detecting Problematic Dialogs with Automated Agents ». In *Perception in Multimodal Dialogue Systems. 4th IEEE Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems, PIT 2008*. (Kloster Irsee, June 16-18 2008), p. 72-80. Berlin: Springer-Verlag Publishers.

Truong, Le Hoang. 2008. «Développement d'un système d'identification des dialogues problèmatiues dans le système de dialogue personne-machine ». Mémoire de maîtrise en genie, Montréal, École de technologie supérieure, 76 p.

Vidrascu, Laurence. and Laurence Devillers. 2005. Real-Life Emotion Representation and Detection in Call Centers Data. Affective Computing and Intelligent Interaction. J. Tao, T. Tan and R. Picard, Springer Berlin / Heidelberg. 3784**:** 739-746.

Walker Marily A., Alexander I. Rudnicky, John Aberdeen, Elizabeth Owen Bratt, John S. Garofolo, Helen Hastie, Audrey N. Le, Bryan Pellom, Alex Potaminaos, Rebecca Passonneau, Rashmi Prasad, Salim Roukos, Gregory A.Sanders, Stephanie Seneff and David Stallard. 2002. « DARPA Communicator Evaluation: Progress from 2000 to 2001 ». In *7th International Conference on Spoken Language Processing (IC SLP-2002)*. (Denver, Sept. 16-20 2002), p. 273-276.

Walker, Marilyn A., Diane J. Litman, Candace A. Kamm and Alicia Abella. 1997. « PARADISE: A Framework for Evaluating Spoken Dialogue Agents ». In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics.* (Madrid July 7-12, 1997), p. 271-280. Morristown (NJ): Association for Computational Lingustics Publishers.

Walker, Marilyn A., Irene Langkilde-Geary, Helen Wright Hastie, Jerry Wright and Allen Gorin. 2001. « Automatically Training A Problematic Dialogue Predictor for a Spoken Dialogue System». *Journal of Artificial Intelligence Research*, vol. 16, nº 34, January/June, p. 293-319.

Walker, Marilyn A., Irene Langkilde, Jerry Wright, Allen Gorin and Diane Litman. 2000. « Learning to Predict Problematic Situations in a Spoken Dialogue System: Experiments with How May I Help You? ». In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics*. (Seattle, April 29 – May 04), p. 210-217. San Francisco (Ca.): Morgan Kaufmann Publishers Inc.

Watanabe, Satoshi. 1985. *Pattern Recognition: Human and Mechanical,* 1st Ed. John Wiley & Sons, Inc., 520 p.

Witten, Ian H. and Eibe Frank. 2005. « Credibility: Evaluating what's been learned ». In *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition, p. 143-186. The Morgan Kaufmann Series in Data Management Systems Elsevier Inc.

Young, Steve, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xuying Liu, Grareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev and Phil Woodland. 2006. *The HTK Book*, 3.4 ed. Cambridge (UK): Cambridge University Press, 359 p.