國 立 交 通 大 學

電信工程學系

碩 士 論 文

語音對話系統和對話策略之研究

A Study on Speech Dialogue System and Dialogue Strategy

研究生：蔡金翰

指導教授：陳信宏

中 華 民 國 九十四 年 七 月

語音對話系統和對話策略之研究

A Study on Speech Dialogue System and Dialogue Strategy

研 究 生：蔡金翰　　Student : Chin-Han Tsai

指導教授：陳信宏 博士　　Advisor : Dr. Sin-Horng Chen

國立交通大學

電信工程學系

碩士論文

A Thesis

Department of Communication Engineering

College of Electrical Engineering and computer Science

National Chiao Tung University

In Partial Fulfillment of Requirements

for the Degree of

Master of Science

in Electrical Engineering

July 2005

Hsinchu,Taiwan,Republic of China

中華民國九十四年七月

語音對話系統和對話策略之研究

研究生：蔡金翰　　指導教授：陳信宏 博士

國立交通大學電信工程學系碩士班

中文摘要

在此論文中首先研究架設對話系統的方法，注重於對話管理員和對話策略的設計方面。為了減輕設計者設計對話策略的負擔，以及避免人為主觀設計出來的並非最佳的對話策略，所以改採用強化學習的方式，讓系統透過跟使用者互動的方式學習最適合的對話策略。之後更設計了一連串的實驗來觀察對話策略學習的情況和效率，證實強化學習的確能用在設計對話策略上，以及觀察辨認環境對策略學習的影響。最後再將研究的方法用來架設實際的對話系統—『智慧型口語對話汽車導航系統』。

A Study on Speech Dialogue System and Dialogue Strategy

Student： Chin-Han Tsai Advisor： Dr. Sin-Horng Chen

Department of Communication Engineering
National Chiao Tung University

Abstract

In this thesis, we study the way of building a dialogue system at first; focus on the design of dialogue manager and dialogue strategy. In order to abate the load of designing a dialogue strategy off designer, and avoid that a dialogue strategy devised subjectively is not the optimal one, we apply the reinforcement learning to design it; let system learns the optimal strategy via interaction with users. Afterward we make a series of experiments to observe the circumstance and efficiency of strategy learning; prove that reinforcement learning really can be implemented on strategy learning, and to observe the effect of environment of speech recognition. Finally, we use the method studied to construct real dialogue system, “intelligent transportation system”.

致謝

研究所兩年的碩士生活真的過得很快，感覺昨天才剛進實驗室幫忙打掃，沒想到現在已經畢業，要被拿掃把趕出實驗室了，不過這兩年卻是我求學生涯中學到最多的階段，不論是人際關係、做研究的方法、解決問題的方式等等都獲益良多。

首先要感謝陳信宏老師和王逸如老師的指導，以及廖元甫老師的協助，老師們的教誨是對我在研究和做事領域上最大的推力和拉力，讓我確確實實感受到出去外面工作時所會遭遇到的困難，以及學會如何解決這些困難的方法，這些的確不是以前大學畢業的我所能擁有的能力。再來就是要感謝學長們毫不藏私的領導我們走進並熟悉語音這塊領域，尤其是俊良學長和智合學長，還有振宇學長在程式撰寫方面的各種協助。還有同屆的各位戰友們，隆勳、佩穎、順哥、lubo、希群，因為能有大家一起打哈、一起埋怨、一起努力的關係，才能讓我在這兩年順利的完成學業，感謝順哥在工作站方面的各種協助，雖然後來較少用到工作站；也感謝佩穎問我程式的問題，讓我能短時間找回撰寫程式的感覺，甚至從妳那獲得語音合成的寶貴知識。當然也感謝活潑的學弟們，因為有你們，實驗室整個充滿了活力。

最後當然也得感謝家人的支持，以及「下班」時經常跑來找我的老弟，陪我一起打哈回寢室，使得一天的疲倦瞬間消失殆盡。再來，爸媽，我會完成你們交代的任務的！盡量啦。

目錄

表目錄

圖目錄

第一章 緒論

1.1 研究動機

從工業革命以來，人跟機器之間已有著密不可分的關係，各種人機介面也相應而生，發展至今，新的人機介面概念就是想仿照人與人之間的溝通方式，以建立最自然且最有效的溝通管道，於是乎有關語音辨認和合成的研究開始蓬勃發展起來。然而儘管語音辨認和合成技術的發展如何進步，沒有一套方法將這些技術整合運用在人機介面上仍是惘然，所以本論文要研究的就是如何有效地建立語音的人機介面－對話系統。

此外也有鑒於汽車駕駛在使用汽車導航等相關系統時，如果仍以傳統的人機介面操作，明顯對駕駛員非常不便，甚至會影響駕駛員的行車安全，所以會利用本論文介紹的架設方式架設一套『智慧型口語對話汽車導航系統』(Intelligent Transportation System，ITS)。

1.2 研究方向

一套對話系統所包含到的技術著實不少，例如即時語音辨認、語音合成、自發性語音研究、資料搜尋、會話分析等等，每項技術對整個對話系統的重要性都是不可忽視的。然而將所有技術整合起來，控制著整個對話系統的對話流程以及系統功能的是對話系統中的對話管理員，而對話管理員該如何控制著這些流程則記載在對話策略中，所以本論文的研究方向將

是注重在對話管理員的架設以及對話策略的設計上。

1.3 章節概要

本論文共分為五章：

第一章 緒論：介紹本論文之研究動機與方向。

第二章 對話系統設計和對話策略學習：研究架設對話系統的方式以及運用強化學習來設計對話策略。

第三章 對話策略學習實驗：設計一套簡單的對話系統，利用虛擬使用者和系統互動來觀察對話策略學習的情況。

第四章 實際對話系統之實現：利用第三章介紹的方式實際架設智慧型口語對話汽車導航系統。

第五章 結論與未來展望。

第二章 對話系統設計和對話策略學習

對人們來說，最便利的溝通管道便是透過聲音，架設對話系統的目的就是建立一個透過語音操作的人機介面。要想達到此目標就得使電腦能直接透過語音理解使用者的語言和語義，並且能準確地完成使用者交代的工作。

2.1 對話系統簡介

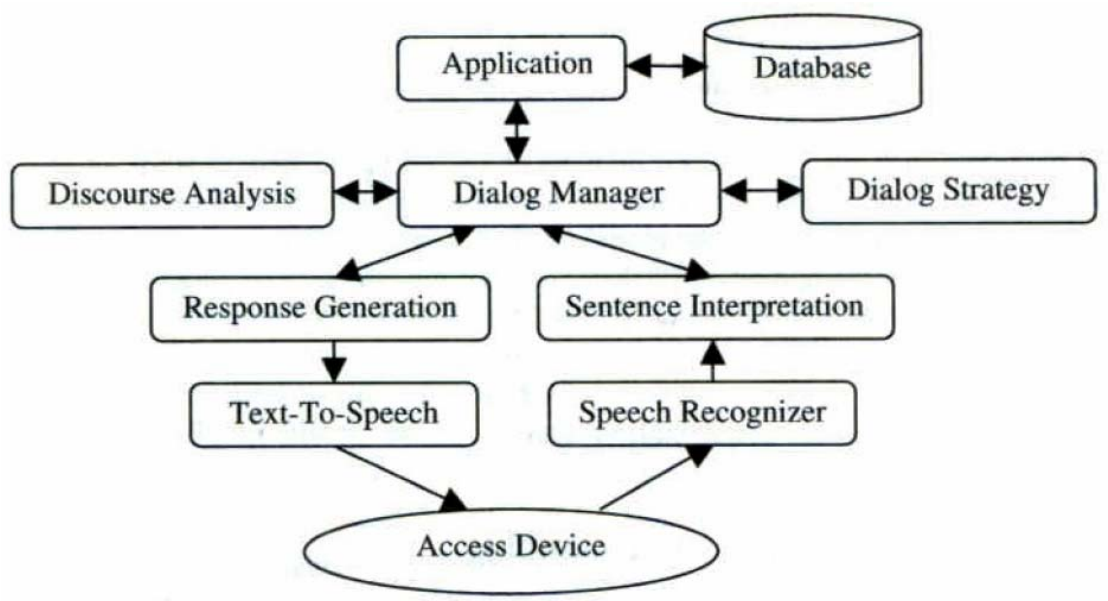

圖 2.1 對話系統的系統圖

整個系統的核心就是對話管理員（Dialog Manager），它整合了由即時語音辨認器辨認出來的結果，透過會話分析（Discourse Analysis）來分析語義，再由得到的語義和系統目前的狀態從對話策略（Dialog Strategy）裡尋得系統下一步該做的動作，並且可藉由搜尋外部的資料庫（Database）來完成任務，再將需要告知使用者的訊息藉由語音合成的技術（Text-To-Speech）用語音通知使用者。

2.1.1 對話策略

對話策略是用來決定系統如何和使用者互動，裡面定義了系統在所處的各種狀態下（State），應該回應使用者的方式（Action）。由於對話策略正是決策系統運行的動作，所以對話策略的好壞不但決定系統是否能如期完成使用者交代的任務，而且也對系統整體的效能影響很大。

2.2 對話系統設計[1]

設計一套對話系統所需的技術不少，例如即時語音辨認、語音合成、自發性語音研究、資料搜尋、會話分析等等，每一項技術都值得做深入的研究，不過由於對話策略在整個對話系統中佔著舉足輕重的地位，所以本論文注重在最佳對話策略的設計上。

2.2.1 最佳化問題（Optimization problem）

為了設計出最佳的對話策略，首先定義一個客觀量測函式（Objective measure function）：$C = \sum W_i \langle C_i \rangle$，其中$W_i$是比重係數（Weight），$\langle C_i \rangle$則是代價期望值（Expected cost），包含離最終目標（例如完成使用者交付的工作）的距離，以及對話系統和使用者互動的效率。如此就能將對話系統的設計看作是一個解決最佳化的問題，其中能使此函式得到最佳值的策略就是最佳的對話策略。

2.2.2 序列決策程序（Sequential decision process）

舉一個簡單的對話系統為例：day-and-month dialogue，這套系統的目的就是經由最有效率的對話方式，取得使用者想設定的詳細日期，此時的客觀量測函式就能寫成：$C = W_I \langle N_I \rangle + W_E \langle N_E \rangle + W_F \langle N_F \rangle$。

N_I：系統和使用者間完成一段對話的互動次數

N_E：系統最後得到的資訊有錯誤的個數（0～2）

N_F：系統最後離最終目標的距離（完成：0，差月或日的資訊：1，完全沒得到資訊：2）。

系統為了達到目標能夠採取的行動包括詢問幾號、詢問幾月、詢問幾月幾號以及關閉對話，送出得到的日期。再將系統所需獲得的資訊整合成數種狀態，以便判斷何時要採取何種回應。在此範例中可以分成五種狀態：初始狀態（$m=0, d=0$）、只得知月份（$m=1\sim12, d=0$）、只得知幾號（$m=0, d=1\sim31$）、得知完整日期（$m=1\sim12, d=1\sim31$）和最終狀態（$m=-1, d=-1$）。如此就能設計出如下圖三種對話策略的對話系統：

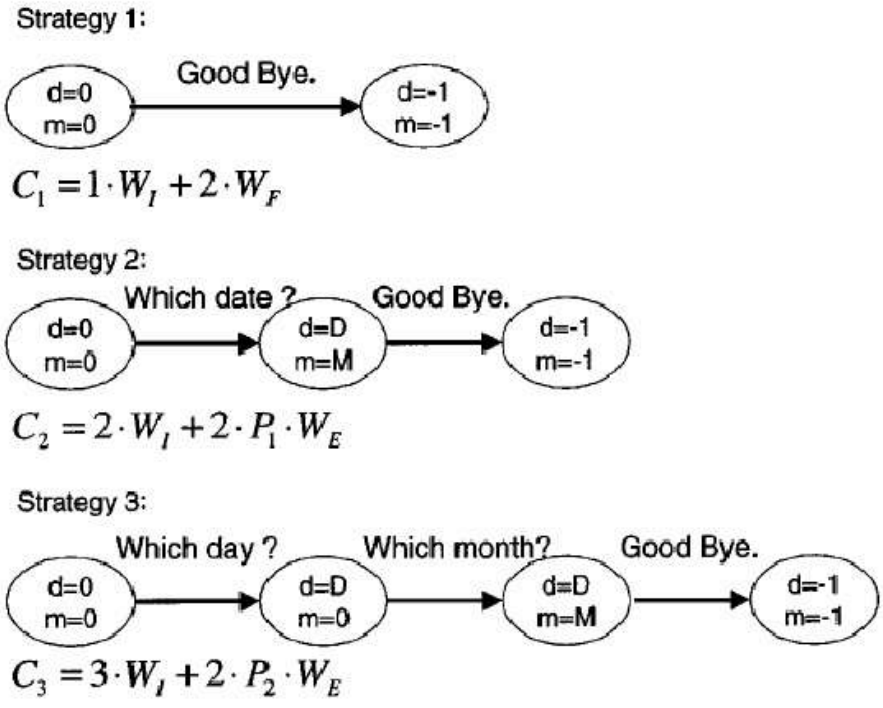

圖 2.2 在 day-and-month dialogue 系統中，三種可行的對話策略

如上圖所示，便可以將對話系統的設計看作是一個序列決策程序，而

這三種對話策略的客觀函式也如上圖中的C_1、C_2、C_3所示，其中P_1表示在同時問月和日時，系統得到的月或日發生錯誤的機率；P_2表示一次只問月或日時，系統得到的值發生錯誤的機率，所以理論上P_1>P_2。這樣我們就能由誰的客觀函式的值最小，來判定它是這三種策略中最好的一個，例如當 $P_2 > (W_F - W_I)/W_E$ 時， $C_1 < C_3, C_1 < C_2$ （因為P_1>P_2），此時最好的策略是 Strategy 1；當 $P_1 - P_2 > W_I/2W_E$ 時，$C_3 < C_2$，則換成Strategy 3 是最好的策略。

2.2.3 馬可夫決策程序（Markov decision process）

雖然可透過上述的方式來研判提供的策略中哪個較好，但是卻無法得知真正最好的策略，所以為了得知所有策略中，哪個才是真正的最佳策略，於是將機率的觀念套到系統設計中。只要將系統中各個狀態的狀態轉換機率（State transition probability）套入，就能計算所有對話策略的客觀函式，由此便可判定出最佳策略。不過為了簡化此方法的複雜度，於是做了兩個假設：

- 狀態轉換機率符合馬可夫特性

$$T(s,a,s') = \Pr(s_{t+1} = s' \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = \Pr(s_{t+1} = s' \mid s_t, a_t) \qquad (2.1)$$

$T(s,a,s')$：在狀態 s 採取行動 a 而下一個狀態會是 s'的機率。

- 付出的代價（Cost）或獲得的獎勵（Reward）只跟目前所在的狀態和採取的行動有關，亦即

$$\Pr(c_t \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots s_0, a_0) = \Pr(c_t \mid s_t, a_t) \qquad (2.2)$$

如此就可將對話系統的設計看作是馬可夫決策程序（Markov decision process, MDP）。為了解此問題，定義個狀態 s 的最佳值函式（Optimal value function）：

$$V^*(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) = \max_{a}\left(R(s,a) + \gamma \sum_{s' \in \mathbf{S}} T(s,a,s') V^*(s')\right), \forall s \in \boldsymbol{S} \quad (2.3)$$

其中π代表使用的對策，即為在各種狀態下會採取的行動，γ為 discount 參數，r_t為在未來 t 步之後得到的獎勵，$R(s,a)$為在狀態 s 採取行動 a 所得到的獎勵。則最佳策略就可以經由下面的式子求得：

$$\pi^*(s) = \arg\max_{a}\left(R(s,a) + \gamma \sum_{s' \in \mathbf{S}} T(s,a,s') V^*(s')\right) \quad (2.4)$$

雖然上述的方法的確能求得真正的最佳策略，但是當整個系統的狀態空間（State space）非常龐大時，上述的式子就會變得非常難解，此外當所能使用的資訊有限，無法完全得知 MDP 所有的參數時，更是無法用上述的方法求取最佳策略。

2.3 對話策略學習

為了解決上述設計中的困難，於是改採用學習的方式，讓系統經由多次和使用者互動的經驗來學習適合的對話策略。

2.3.1 強化學習（Reinforcement Learning）簡介[2,3]

讓系統和外在環境互動，藉由 trial-and-error 的方式學習最佳的行為模式，就是利用統計學的方法，以及 dynamic programming 的方式，讓系統在所有可行的行動裡，探索出最好的一個。下圖就是一個以對話系統為例子的強化學習標準模型：

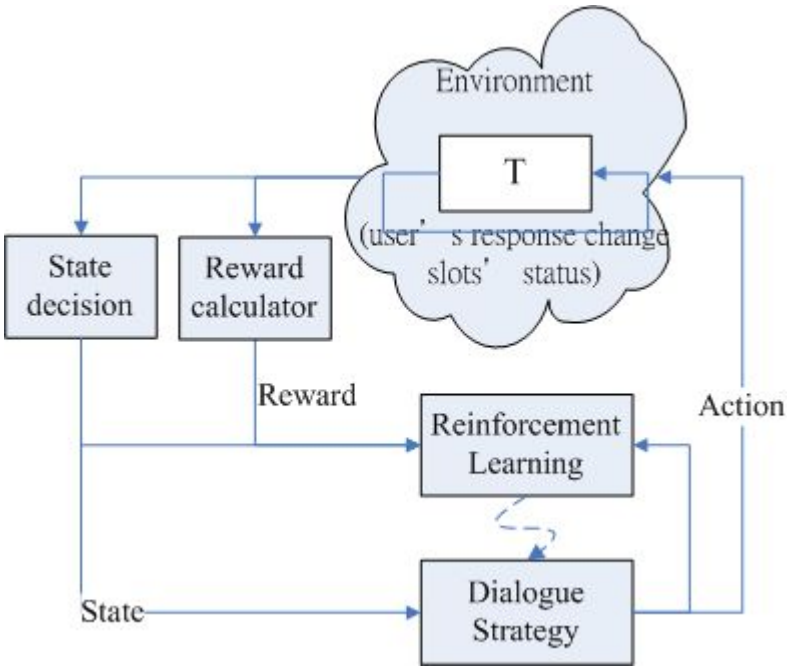

圖 2.3 標準的強化學習模型

將使用者的回應和系統目前所處的狀態視為外在環境，利用強化學習來學習最佳的對話策略（在哪種狀態要採取哪種行動）。其主要的想法就是以之前提到的 MDP 模型，運用學習的方法使得到的結果漸漸趨近於此目標，所以如上圖所示，需要得知『目前系統所處的狀態』以及採取的『行動』，再加上『之後系統所處的狀態』和獲得的『獎勵』。

由於對話系統的獎勵是要在採取行動之後的下一個狀態才能得到，所以是屬於延遲型獎勵（Delayed reward）的情況，在此情況下的強化學習可以分成兩大類：

- 模型式（Model-based）：所謂的模型就是指 MDP 最佳值函式中的狀態轉換機率（T）和獎勵函式（R）。此方式就是透過學習 T 和 R 來讓算出來的值函式（Value function）能趨近最佳值函式。例如 certainly equivalent methods [2,11]、Dyna method [2,15,16]、prioritized sweeping [2,12]（queue-Dyna [2,13]）等等。
- 無模型式（Model-free）：不再使用 T 和 R，直接透過學習值函式

的方式，使其能趨近於最佳值函式。例如 adaptive heuristic critic[2,3,10] 和 temporal difference methods（$TD(\lambda)$）[2,3,14]，以及 Q-learning [2,3,4]等等。

雖然 model-free 做法的收斂速度比 model-based 的做法還慢，但是所花的計算時間卻相對少很多，而且也較容易套用在實際系統上。在對話系統上，由於採用即時語音辨認器，在辨認使用者的語音輸入上勢必會花不少時間，尤其若要求一定水準的辨認率，那需要花的辨認時間就更久了，所以在目前許多使用 reinforcement learning 來設計對話系統的設計者，幾乎都是採用 Q-learning 的方式，或者在原本的 Q-learning 上加點改良。

2.3.2 Q-learning

定義一個 optimal Q（Quality）value function：

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a') \tag{2.5}$$

所以

$$V^*(s) = \max_a Q^*(s,a) \tag{2.6}$$

由上式可求得最佳策略為

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{2.7}$$

而 Q-learning 就是透過直接學習 Q value 的方式，使 $Q(s,a)$ 趨近 $Q^*(s,a)$，使用的 Q-learning rule 如下：

$$\begin{aligned} Q(s,a) &:= (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a')\right) \\ &= Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) \end{aligned} \tag{2.8}$$

其中 α 為 learning rate，s 為目前的狀態，a 為在狀態 s 採取的行動，s'為下

一個狀態，a'為在狀態 s'採取的行動。

Q-learning 不但容易實現在實際系統上，而且也已經被證明過在 discrete case 下，learning rate α 的範圍只要在 $0 \le \alpha \le 1$ 且隨著學習的過程逐步減少的話，Q-learning 就可收斂到最佳策略[4]。

2.3.3 探索行動空間的時機和方式

在使用強化學習時，有個非常重要的問題需要解決，那就是何時該依循目前已得的策略行動（Exploit），何時該探索其它可能更好的行動（Explore），這個部份對強化學習的效率和結果影響很大。最為人知也最容易的方式就是 ε-greedy method [3]，平時都依循得到的策略來採取行動，只有ε的機率會隨機選一個來當目前要採取的行動。雖然此方式可行，但是效率卻非常不好，如此隨機選擇能找到更好的行動機會不高；而且也可能在已經找到最佳策略時，卻還一直探索其它行動而降低學習效率。所以從開始有人使用強化學習到現在，仍有不少有關探索時機和方式被發表出來。

另一種較佳的方法是 SA-Q learning [7]，想法是將 SA（Simulated annealing）演算法套用在 Q learning 裡。SA 演算法是定義目前所處的狀態（i）轉換到新的狀態（j）的機率為 $P(i \Rightarrow j)$，在處理最大化問題的情況下其值定義成：

$$P(i \Rightarrow j) = \begin{cases} 1, & \text{if } f(j) > f(i) \\ e^{\frac{f(j)-f(i)}{t}}, & \text{otherwise} \end{cases} \tag{2.9}$$

其中 $f(i)$和 $f(j)$相當於最佳化問題中的代價函式，t是temperature，運用到Q learning的方式就是以此決定是否探索其它行動。假設在狀態s下，

依照策略選擇出來的行動為a_p，隨機選擇出來的行動為a_r，這時決定採取探索（將原本要採取的行動a_p換成a_r）的機率為$P\left(a_p \Rightarrow a_r\right)$，將SA演算法套用進來可得到：

$$P\left(a_p \Rightarrow a_r\right)=\begin{cases}1, \text{ if } Q\left(s,a_r\right)>Q\left(s,a_p\right)\\ e^{\frac{Q\left(s,a_r\right)-Q\left(s,a_p\right)}{t}}, \text{ otherwise}\end{cases} \tag{2.10}$$

由於$Q\left(s,a_r\right)\le Q\left(s,a_p\right)$，所以$P\left(a_p \Rightarrow a_r\right)=\exp\left(\frac{Q\left(s,a_r\right)-Q\left(s,a_p\right)}{t}\right)$，當系統到達目的狀態（Goal state）時，再利用 temperature-dropping criterion 更新 t：$t_{k+1}=\lambda t_k, k=0,1,2,\ldots,\lambda\in\left(0.5,1\right)$。此方法可以視為仍是採用 ε-greedy method，不過會依照學習的狀況，動態調整ε。

第三章 對話策略學習實驗

為了將 Q learning 套用在對話策略設計上，以及觀察其收斂的情況，於是著手設計一個專門用來模擬學習對話策略的對話系統。由於只用來觀察對話策略學習的狀況，所以只須要設計對話系統中對話管理員和對話策略的部份即可，因此大大減低設計的難度。

3.1 實驗一設定

對話系統的實驗設定如下：

3.1.1 對話管理員設計

- Slots：將系統需要得知的資訊設計成一個需要填值的表單（Keyword-Value pair），例如目的地（Keyword）：交大（Value）
- States：表示系統目前所處的狀態，由目前 slots 的狀態決定
- Actions：系統在跟使用者互動中，可以採取的行動

可模擬 N 個 slots，K 種功能（Task），允許各個功能間共用某些 slots，對話系統的目的設定為『以最有效率的方式得知使用者想使用的功能以及有關的 slots 資訊』。

例如有一個對話系統，它提供了 5 種功能，共使用 13 個 slots，各個功能所使用的 slots 分佈如下：

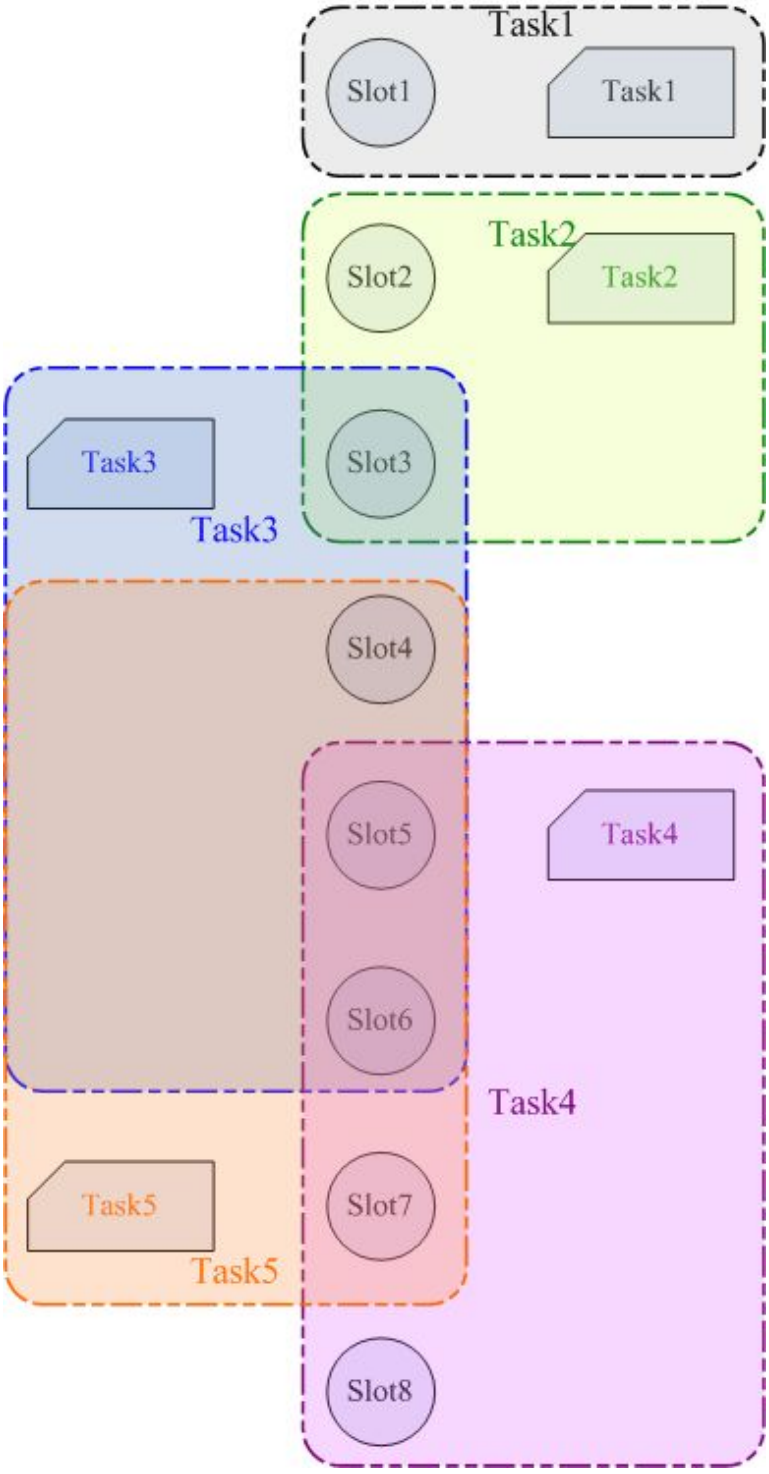

圖 3.1 各種功能使用的 slots 分布（實驗一）

圖 3.1 中圓形是表示要完成此功能所需的必要 slots；多邊形則是非必要 slots。在此次模擬中，將系統設計成只需從使用者那得知必要的 slots 資訊即可完成任務，而不需一定得要使用者表明他想使用何種功能（Task1

～Task5）。此時代表功能的 slots 就是所謂的非必要 slots。接著將介紹在對話系統中所制定的 state 和 action。

3.1.1.1 States

為了模擬允許使用者主控整個對話流程（User initiative），所以 states 考慮了已知或未知目前使用者想使用何種功能的資訊，此外為了先簡化模擬的複雜度，所以假設即時語音辨認器的辨認率是 100%，而且暫不學習確認 slots 的行動（當辨認出來的結果可信度不高時，系統再次跟使用者確認結果是否正確的動作），因此各個 slot 可能的情況只有『已填值』和『未填值』，依此建立的 states 如下：

- 和功能相依的states（$2^{\text{num. of slots in task}}$）：task1《$2^1+2$（ready & goal）》、task2《$2^2+2$》、task3《$2^4+2$》、task4《$2^4+2$》、task5《$2^4+2$》

ready：當要完成此任務所需的必要 slots 都得知後，顯示所得資訊，跟使用者做最後整體確認以便啟動功能完成任務。goal：此任務完成。

- 未知使用者想使用何種功能的state（$2^{\text{num. of total slots}}$）：$2^8$
- 其它 state：初始、系統結束…等等

共有 329 個 states。

3.1.1.2 Actions

至於 actions 的部份則完全跟功能有關，包含：

- Greeting（只有系統問候語，並不詢問特定 slots）
  - 歡迎使用 xxx，請問需要什麼服務？
- 詢問想使用何種功能
  - $2^{\text{num. of tasks}}-1 \rightarrow 2^5-1$
- 詢問跟功能有關的 slots

- $\sum_{taski}\left(2^{num.\ of\ slots\ in\ taski}-1\right)-\sum_{\substack{taski\cap taskj\\ i\neq j}}\left(2^{num.\ of\ slots\ both\ in\ taski\ and\ taskj}-1\right)$
  $+\sum_{\substack{taski\cap taskj\cap taskk\\ i,j,k互異}}\left(2^{num.\ of\ slots\ also\ in\ taski,\ taskj\ and\ taskk}-1\right)-\ldots$

$$\rightarrow 2^1-1+2^2-1+3\times\left(2^4-1\right)-2^1+1-2^2+1-2\times\left(2^3-1\right)+2^2-1=34$$

- 最後確認是否啟動功能
  - 功能個數→5
- 詢問是否確定要關閉系統（有模擬使用者放棄使用系統的行為，所以系統需要跟使用者確認是否確定要關閉系統）

所以共有 72 個 actions。

3.1.1.3 對話策略

有了定義好的 states 和 actions 後，就能以此建立對話系統的對話策略，也就是決定系統在各個 state 所要採取的 action。為了之後要使用 Q learning 來學習對話策略，所以將所有的 Q 值（$Q(s,a)$）建立成一個二維矩陣（329×72），一維代表 states；一維代表 actions，這樣系統會採取的策略就是在此 state 下，Q 值最大的 action。此外為了避免系統和使用者互動卻一直無法完成任務而使用者也不打算放棄使用的情況發生，定義對話次數大於某個數之後，若系統現在的 state 仍和上次的相同時，系統就會自動關閉對話。

3.1.2 Q Learning 設定

要讓系統使用 Q learning 來學習對話策略，除了要定義好上述的 states 和 actions 外，還得定義在學習過程中，依情況給予系統的各種獎勵（Reward）函數，以及決定探索的時機和方式。

3.1.2.1 獎勵函數

獎勵的部份目前與下列參數相關：

- 目前跟各個功能最終目標的距離（尚未填值的 slots 數）：D

$$D = \sum_{taski} W_1 \times UV_1 + W_2 \times UV_2$$

UV_i：尚未填值的slots數，i是 1 時代表是必要slots；2 則代表非必要slots。

- 對已確定的 slot 重新填值：RV

RV = 對已確定的slot重新填值的個數

- 系統詢問的 slots 和使用者回答的 slots 之間的符合度：Mis

Mis=兩者不符合的個數

- 系統在未完成任務就被迫關閉：C

$$C = \begin{cases} 1, & \text{系統強制關閉or使用者放棄使用} \\ 0, & \text{otherwise} \end{cases}$$

- 任務成功與否：G

$$G = \begin{cases} 1, & \text{完成任務} \\ 0, & \text{未完成任務} \end{cases}$$

- 對話次數：每對話一次就對獎勵函數加 T

T=1

而獎勵函數則定義如下：

$$Reward = W_D D + W_R RV + W_M Mis + W_{cancel} C + W_G G + W_T T$$

在後面模擬中的比重參數各為：

$W_1 = 1, W_2 = 0.5$

$W_D = 1, W_R = -3, W_M = -3, W_{cancel} = -60, W_G = 53, W_T = -1$。

所以對話策略就是在此 state s 裡採取使 Q value 值最大的 action：$\pi^*(s) = \arg\max_a Q^*(s,a)$。

上述所定的獎勵函數中，比較特別的函數是 *Mis* 和 *C*。

1. *Mis*：由於設計只需將必要的 slots 填完，不必得知使用者想要使用的功能就能完成任務，所以在系統未知使用何種功能的資訊下，同樣是詢問 slots，一個只詢問使用者想用的功能的 slots；另一個則是多問了不相干的 slots，這兩種問法對系統而言都能拉近跟任務完成之間的距離，所以無法分辨兩者哪個較適當。因此才會加上此種計算獎勵的方式，以便判斷較佳的策略。
2. *C*：詳細觀察上述定義的獎勵會覺得這部份似乎跟參數 *G* 重複了，之所以會再使用一項 *C*，是因為如果只使用 *G*，通常會將 *G* 定的很大，使得系統在學習過程中會更偏向往達成目標的方向走，但是這也會提高學習結果收斂到區域性最佳解（Local optimal）的風險。以之前所設計的對話系統為例，當只有 slot3 的值是確定的情況下（如圖 3.2），使用者要使用的功能可能是 task2 或 task3，此時最佳的策略應該是先詢問使用者想使用的功能是 task2 還是 task3（Optimal strategy），但是因為 *G* 很大而且只差 slot2 的資訊就能完成 task2（Local optimal strategy），所以此時學習到的策略很容易會收斂到區域性最佳策略（Local optimal strategy）（如圖 3.3）。如果此時使用者是想使用 task3 的話，此系統就無法完成使用者交代的任務。為了解決此問題只好將 *G* 調低，讓系統即使暫時進入區域性最佳策略，也還能跳回最佳策略（Optimal strategy），但是這樣勢必也會無法拉開最佳策略和其它路徑的距離，於是才引入 *C* 以降低其它路徑的 Q value（如圖 3.4）。

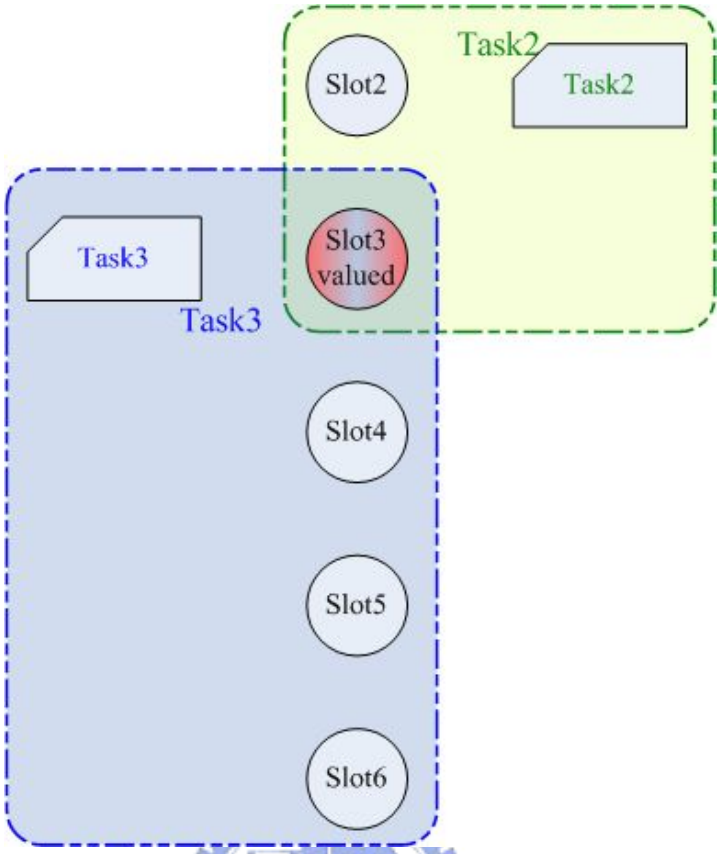

圖 3.2 當只有 Slot3 的值確定時

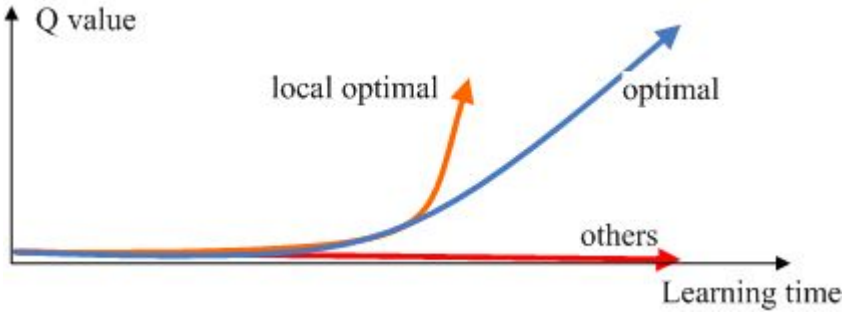

圖 3.3 只使用值很大的 G

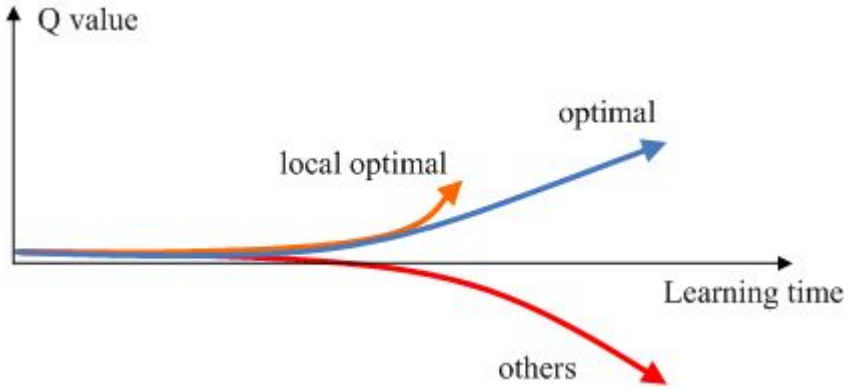

圖 3.4 引入 C，G 的值就可以不必太大

3.1.2.2 SA-Q Learning 參數設定

在決定何時該探索其它可能更好的行動的部份我採用 SA-Q learning[7] 的方式，目前採用的參數值為 $t_0 = 33.5598, \lambda = 0.924$ ，使系統在剛開始時大約有 0.8 的機率會探索；當此功能成功完成 10 次後，大約有 0.6 的機率會探索；當此功能成功完成 30 次後，大約只剩 0.1 的機率會探索。

3.2 使用者模型（User Model）之模擬

完全經由跟實際的使用者互動來學習對話策略雖然最後是可以得到最佳策略，但是在系統剛開始學習時，由於此時幾乎可說是沒有對話策略，所以此時的使用者需要很有耐心應付跟系統雞同鴨講的情況，此外也需要大量的互動經驗才能得到真正的最佳策略。所以先用虛擬使用者跟系統互動，先讓系統學習到初步不錯的對話策略，再讓系統直接和實際的使用者互動，進一步學到最佳的對話策略。此外由於是虛擬的使用者，所以可以將互動媒介退化到單純的傳輸語意，因此可以很輕易地架設個測試用的對話系統。經由語意的方式互動可以快速得知對話系統的效能，藉此能輕易調整系統的各個參數來增加效能，也可以由此測試系統是否正常運作，幫助系統設計者在設計早期就可以修改大部分的設計缺失，不至於到整個系

統都完成了才發現許多設計上的錯誤。

使用虛擬使用者代替真實的使用者的方式如下圖：

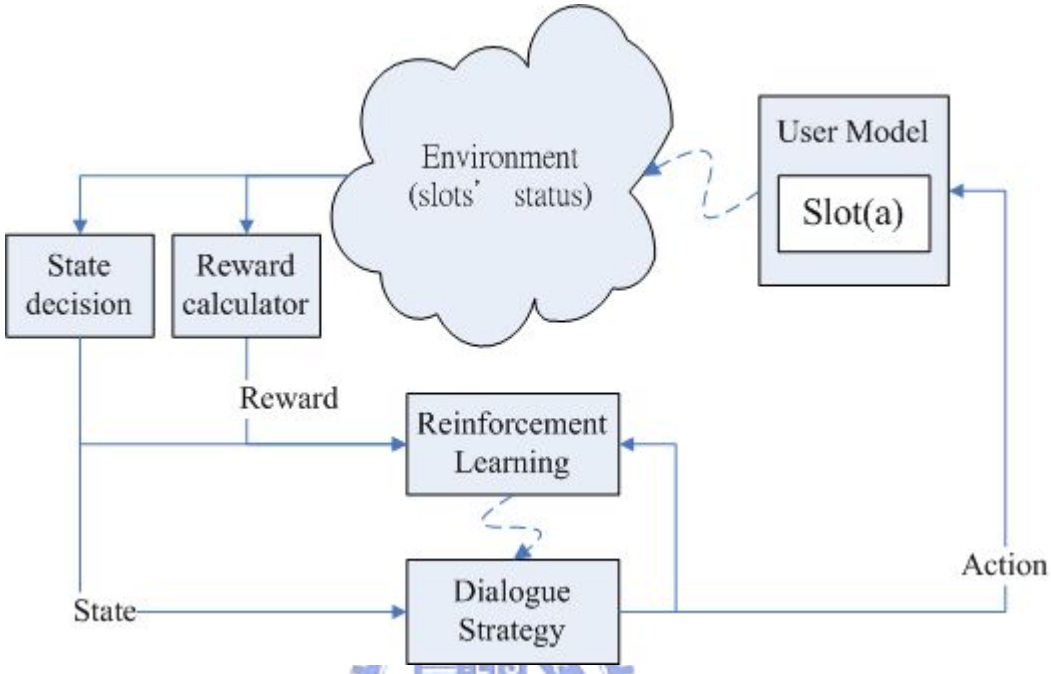

圖 3.5 使用虛擬使用者的對話系統架構

將外在環境中的使用者用使用者模型代替，此虛擬使用者『只會對系統給予的行動產生相對的回應，而跟系統的狀態無關』。回應的方式就是依照想要使用的功能所需，配合系統給的提示，用機率的方式決定是否對系統所詢問的問題給予答案。

此使用者模型包含的參數（使用者提供各個 slot 資訊的機率）如下：

- 遇到系統 greeting 時
  - $P_{greet}(s_i \mid task_k)$：在想使用第 k 個功能時，會在系統 greeting 時回答第 i 個 slot 的機率
- 其它時候
  - 由系統提示的複雜度分成三種，每一種各有一個填 slots 的機

率，以同時提到的 slots 越多，回答每個 slot 的機率就越低為原則

- ◆ 系統提示很簡單：T_1
- ◆ 系統提示不會太難：T_2
- ◆ 系統提示很複雜，不易懂：T_3

■ $P(s_{answer} \mid s_{ask}, T_i)$：當系統提示屬於$T_i$（i＝1, 2, 3），而被問到slot s_{ask} 時，回答slot s_{answer} 的機率

以上都是用程式隨機產生 0 或 1，設其出現 1 的機率等於上述的參數，產生 1 時即判斷成回答此 slot，0 時則否。原本使用者參數的部份應該統計使用者在所有系統提示下回應的方式，但是考慮在系統架設之初，收集的語料量極少，所以才把系統提示只分三大類，以減少所需的語料量。

虛擬使用者回應的流程圖如下：

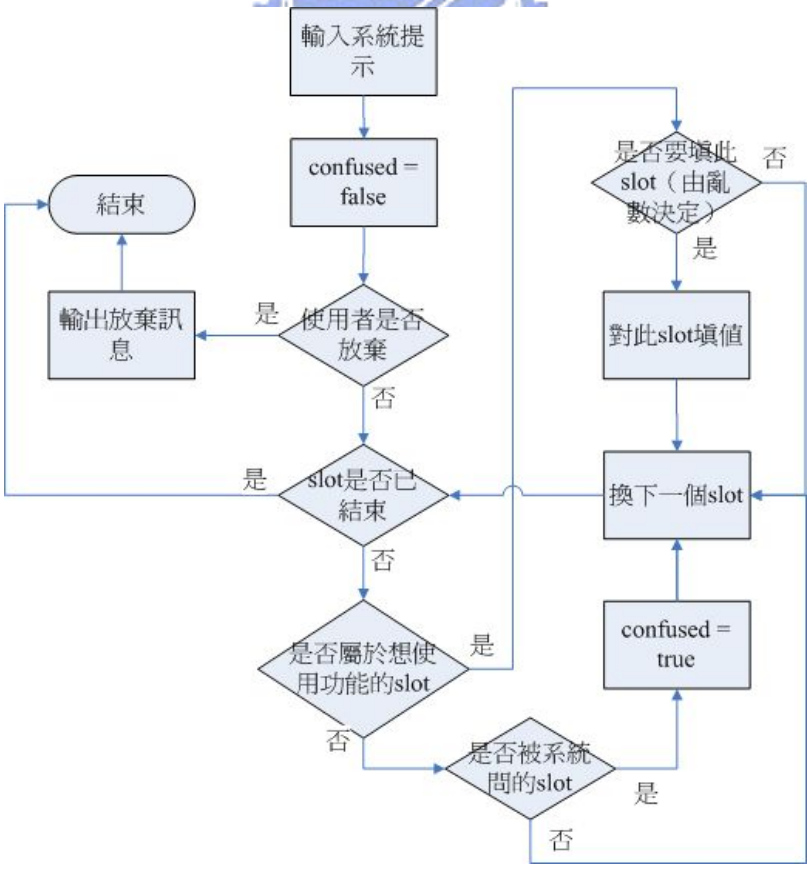

圖 3.6 User model 回應系統提示流程圖

整個模擬程式的流程圖如下圖所示：

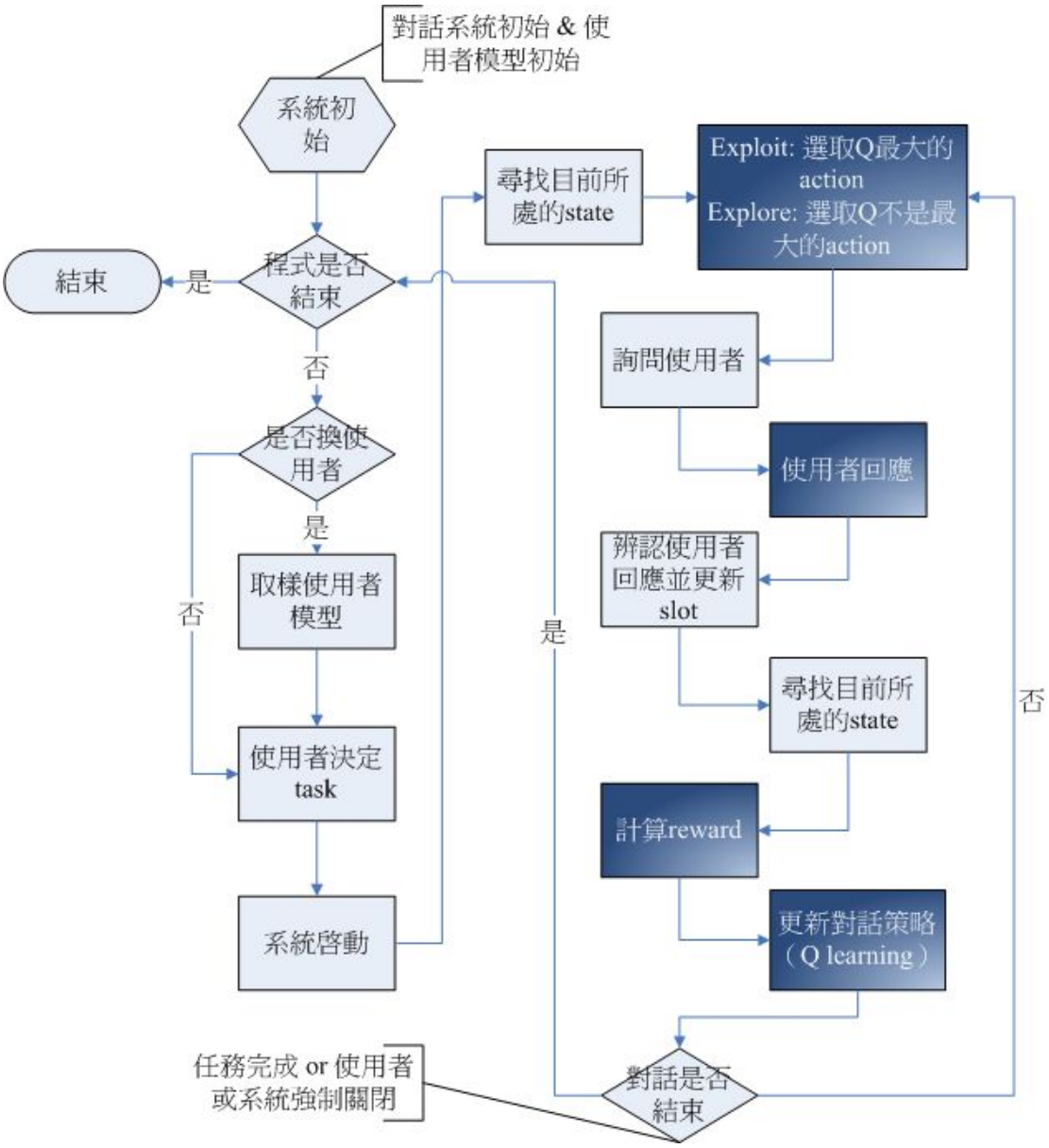

圖 3.7 模擬程式流程圖

3.3 實驗一結果

將程式模擬出來的結果整理分析之後得到以下結果。

3.3.1 學習到的最佳策略

將系統最後得到的對話策略加以分析整理之後，可將系統採取的策略用圖 3.8 表示：

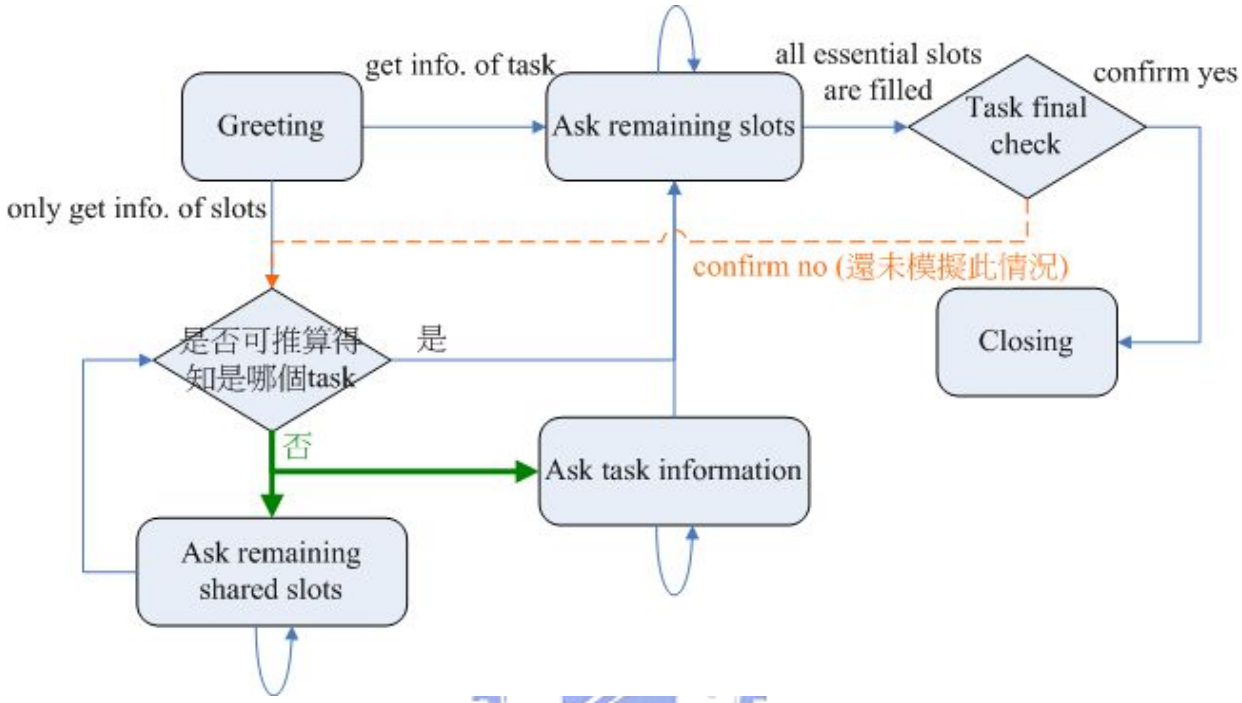

圖 3.8 最佳對話策略

雖然並非所有 states 都能學到此最佳策略，但是幾乎大部分的 states 最後都收斂到此最佳策略，所以即使並未將如何從已有哪些 slots 有值來判斷使用者可能想使用哪種功能的判斷式設計入系統中，系統最後還是可以由 Q learning 的方式學習到。另外圖 3.8 中粗線部分有兩個策略可走，至於何時會走哪條就跟定義的獎勵有關，如果此時走某一條能獲得的獎勵明顯大於另一條時，系統會選獎勵大的那方；不過要是兩者獲得的獎勵相近，系統則隨機學得其中一條。圖中的虛線是目前還未模擬的部份，因為目前將語音辨認率設定成 100%，所以在系統已得知所有跟任務有關的必要 slots 值之後，在跟使用者做最後整體任務確認時，使用者一定會回應確認，所以無法模擬虛線部分，而現在圖中虛線是預測若有模擬使用者回應否定的

情況時，系統會採取的反應。

3.3.2 對話實例

將系統學習前後的差別以對話實例來比較，下面是擷取模擬中系統和虛擬使用者之間的對話：

在《使用者想使用 task3，必要 slots 為 slot3、slot4、slot5、slot6，非必要 slot 為 task3_on》的情況下：

- ◆ 剛開始學習時

語義或行動	轉換成文字內容
S：task5_confirm	你是想使用 Task5？
U：confirm_no	不是。
S：ask_slot5&7_info	請問你要的 Slot5 與 Slot7？
U：confused	什麼？
S：ask_slot7&8_info	請問你要的 Slot7 與 Slot8？
U：give_up	系統關閉！
S：closing	謝謝，歡迎再次使用本系統。

- ◆ 學習完成之後
  - ■ 對話例句一

語義或行動	轉換成文字內容
S：greet	歡迎使用 xxx 系統，請問需要什麼幫助呢？
U：give_slot3	給予 Slot3 的值。
S：task2&3_confirm	你是想使用 Task2 還是 Task3？
U：give_task3	我想使用 Task3。
S：	請問你要的 Slot4、Slot5 與 Slot6？

ask_slot4&5&6_info	
U：give_slot4&5&6	給予 Slot4、Slot5 與 Slot6 的值。
S：task3_chk	目前功能：Task3 Slot3 為 [correct]，Slot4 為 [correct]，Slot5 為 [correct]，Slot6 為[correct]。 可以開始 task3 嗎？
U：confirm_yes	是的。
S：closing	Task3 已完成。歡迎再次使用本系統。

■ 對話例句二

語義或行動	轉換成文字內容
S：greet	歡迎使用 xxx 系統，請問需要什麼幫助呢？
U：give_slot3&4	給予 Slot3 與 Slot4 的值。
S：ask_slot5&6_info	請問你要的 Slot5 與 Slot6？
U：give_slot5&6	給予 Slot5 與 Slot6 的值。
S：task3_chk	目前功能：Task3 Slot3 為 [correct]，Slot4 為 [correct]，Slot5 為 [correct]，Slot6 為[correct]。 可以開始 task3 嗎？
U：confirm_yes	是的。
S：closing	Task3 已完成。歡迎再次使用本系統。

3.3.3 對話策略學習演化過程

接著逐步觀察系統學習對話策略的過程，可以發現隨著學習次數漸漸增加，系統學到的策略形式由簡單轉變到複雜；效率也由低提升到高。宏

觀的對話策略演化的過程就如圖 3.9 到 3.12 所示：

圖 3.9 系統學習前採取的策略

圖 3.10 約 150 個 epochs 之後

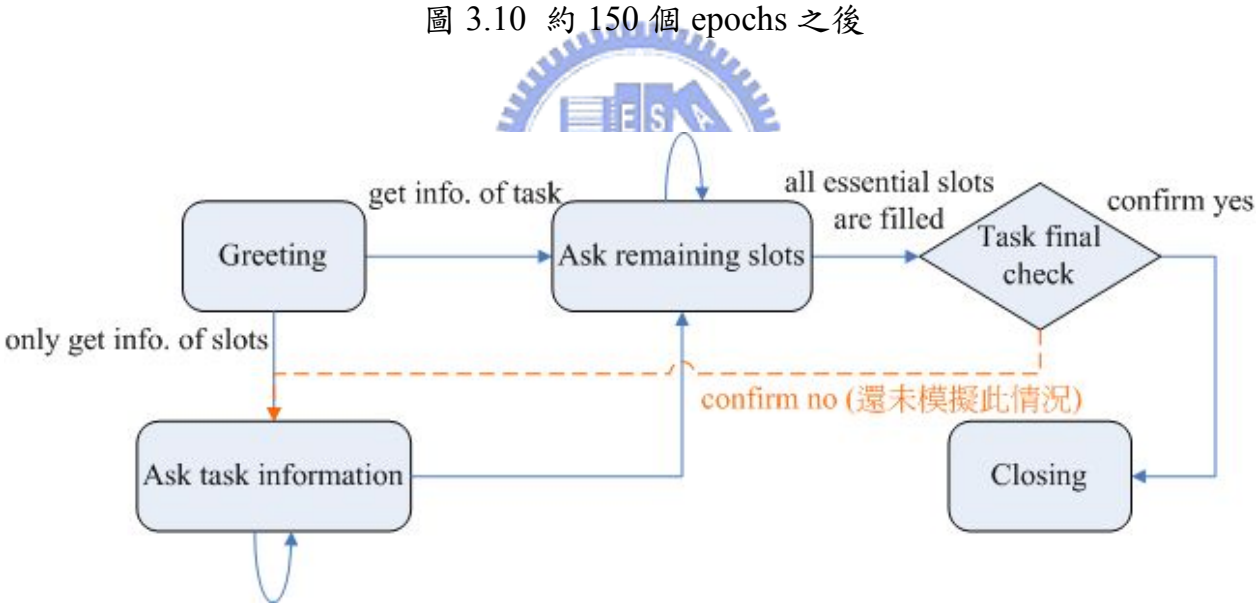

圖 3.11 約 900 個 epochs 之後

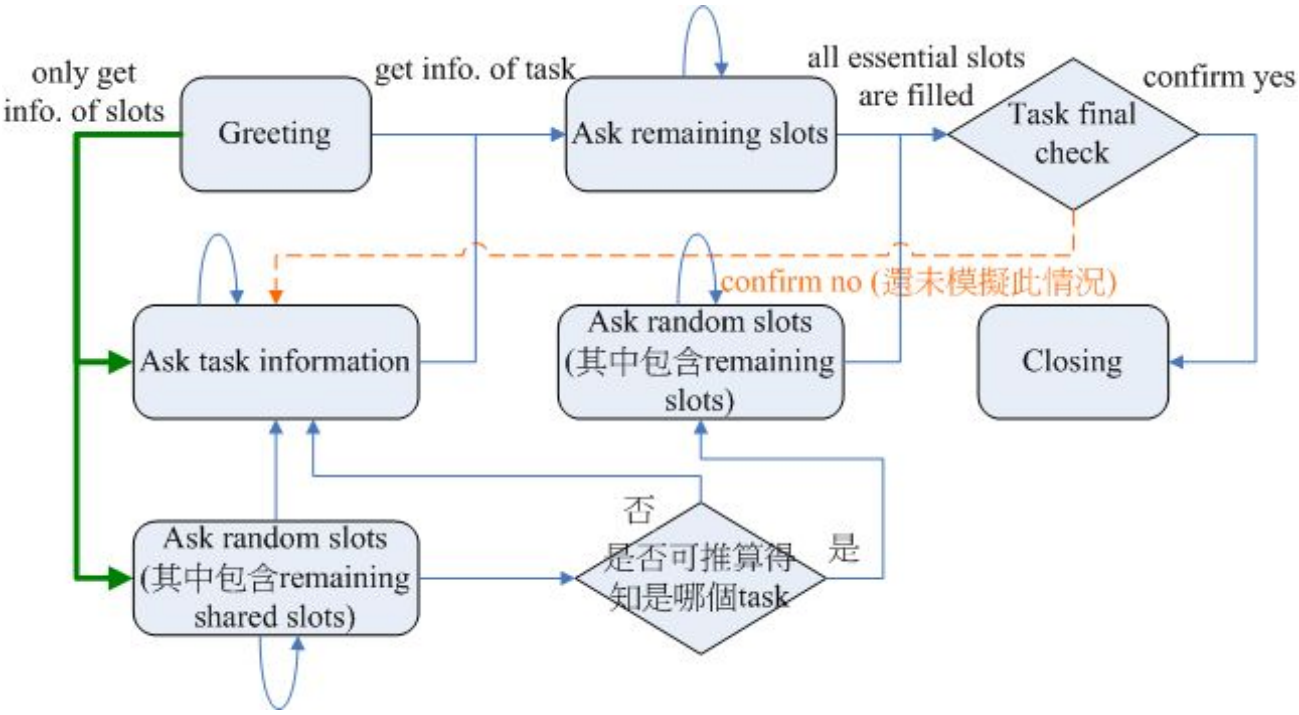

圖 3.12 約 1800 個 epochs 之後

在圖 3.9 中是系統初始時所採取的策略，之所以會有此種策略是因為在學習前，*Q* 值矩陣裡所有值都被初始化成 0，所以所有 states 經由此矩陣選取出來的行動都會是第一個 action，而系統定義的第一個 action 就是 greeting，以致系統從頭到尾都只重複系統開場白，直到使用者放棄使用或系統自動關閉為止。

3.3.4 對話策略收斂情況

大約 2300 epochs 之後，系統的策略就收斂至最佳策略（圖 3.8）。此外若深入觀察可以發現系統 states 的收斂比率也隨著學習次數增加而上升。圖 3.13 的 state 收斂率是以最後學到的對話策略為收斂基準，計算學習過程中得到的各種對話策略和收斂基準之間所有 states 採取相同 action 的比率。

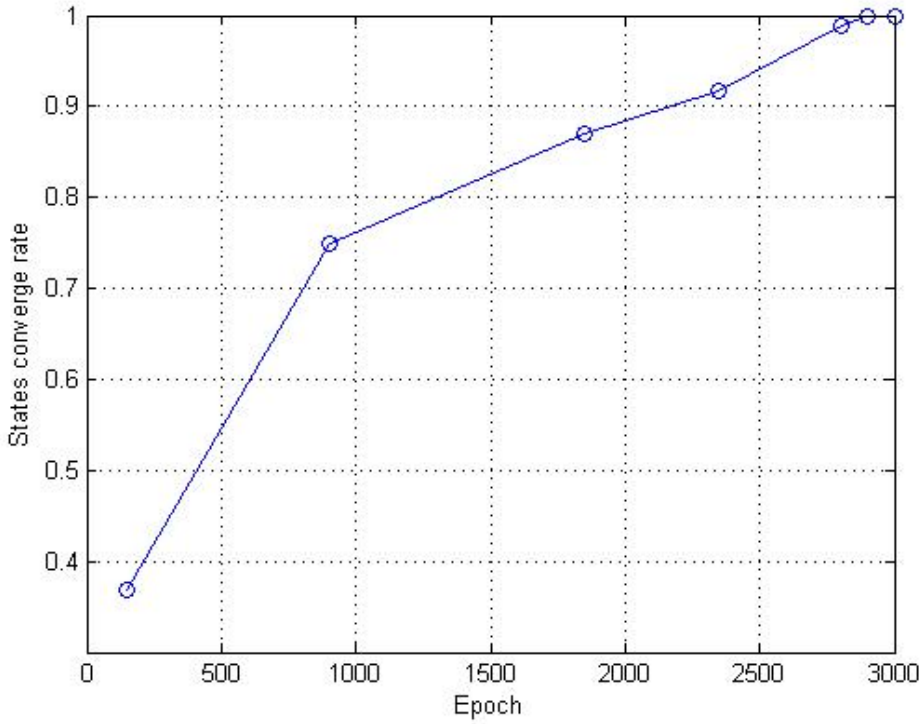

圖 3.13 States 收斂率（實驗一）

3.4 實驗二設定

實驗一的部份是簡化過的模擬，為了更進一步觀察對話策略的學習過程，於是設計了新的實驗，以期能越接近實際設計的情境越好。

由於為了接近實際狀況的對話策略學習模擬，所以考慮語音辨認產生的辨認率變化造成辨認錯誤，此外假設架設的即時語音辨認器能測量辨認結果的可信度（Confidence），而當辨認結果的可信度低於設定的標準時，則將填入此結果的 slot 狀態設成未確定，所以若也將此變化考慮到對話系統的設計中，便使 slot 擁有四種狀態：未知（Unknown）、未確定（Unconfirmed）、已確定（Grounded）、刪除（Cancelled）。其中狀態『刪除』是當使用者表明不需用到此 slot 時，便將此 slot 的狀態設為刪除。

3.4.1 對話管理員設計

仍採用實驗一的設計方式，不過因為現在每個slot有四種狀態，而要定義的states是依照目前所有slots的狀態決定，所以states數會由原本跟 2^n 有關變成跟 4^n 有關，而對話策略學習的複雜度和所要花的時間明顯跟系統的states數有關，所以為了簡化此模擬實驗的複雜度，於是重新以新的對話系統範例為實驗對象。

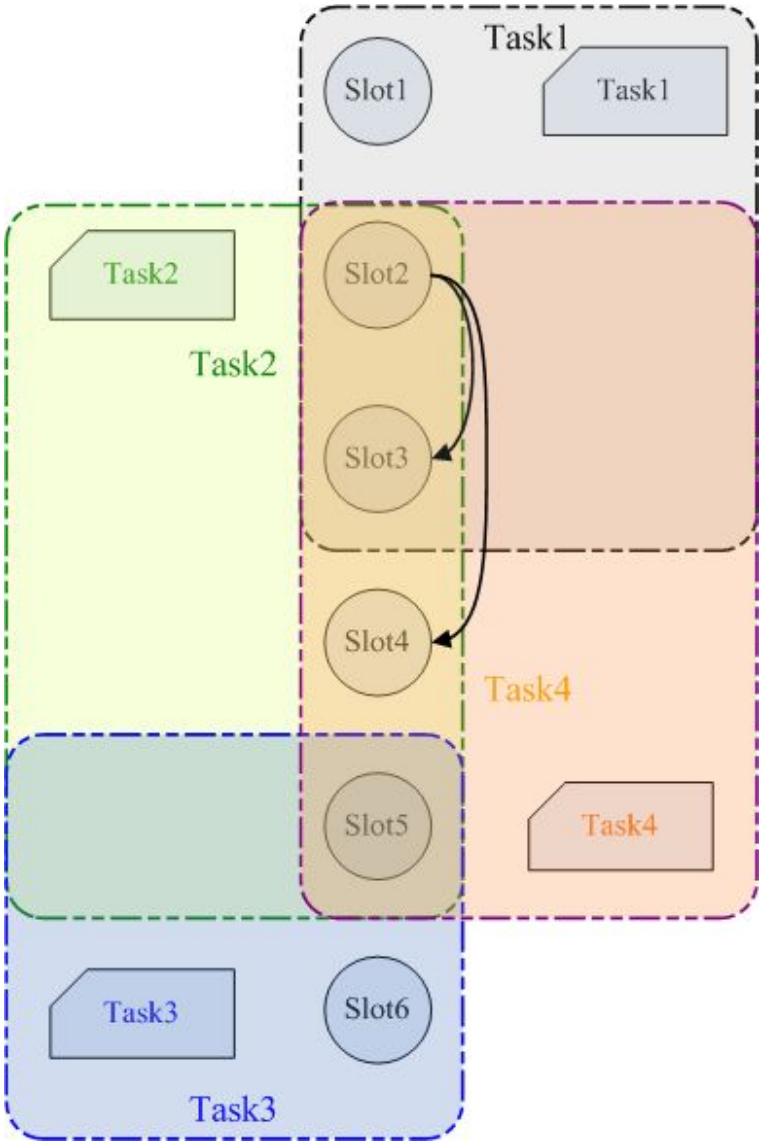

圖 3.14 各種功能使用的 slots 分布（實驗二）

新的對話系統範例如圖 3.14，提供四個功能，總共使用 10 個 slots，依

然將紀錄功能名稱的 slots 視為非必要 slots，只要必要 slots 填完就能完成功能。此外想觀察當 slots 間有相依性時，對話策略是否也能靠學習的方式處理此性質的對話系統，如圖中黑色箭頭的部份是表示 slot3 和 slot4 要當 slot2 的值已確定之後才能給值，也就是在 slot2 的值未確定的情況下，即使使用者給予 slot3 和 slot4 的值，系統仍會視而不見，不處理此訊息。接著設計出來的對話系統中，各個 state 和 action 的制定如下：

3.4.1.1 States

- 和功能相依的states（$(2^{\text{num. of slots in task}}+2)\times 2^{\text{num. of total slots}}$）：task1《$(2^3+2)\times 2^6$》、task2《$(2^4+2)\times 2^6$》、task3《$(2^2+2)\times 2^6$》、task4《$(2^4+2)\times 2^6$》
- 未知使用者想使用何種功能的state（$4^{\text{num. of total slots}}$）：$4^6$
- 其它state：初始、系統結束…等等：8×2^6

總共 7936 個 states。

3.4.1.2 Actions

- Greeting（系統開場白，只有系統問候語，並不詢問特定 slots）
  - 歡迎使用 xxx，請問需要什麼服務？
- 詢問想使用何種功能
  - $2^{\text{num. of tasks}}-1\rightarrow 2^4-1=15$
- 確認是否想使用某功能
  - 功能總數→4
- 詢問跟功能有關的 slots
  - $2^{\text{num. of total slots}}-1\rightarrow 2^6-1=63$
- 確認跟功能有關的 slots

- 同上→63

- 最後確認是否啟動功能
  - 功能數→4
- 詢問是否確定要關閉系統（使用者宣佈放棄使用系統時，系統需要跟使用者確認是否確定要關閉系統）

總 actions 數為 151 個。

3.4.1.3 對話策略

仍然和之前的一樣是由 Q 值函式來決定，所以建立了一個 7936x151 的 Q 值矩陣。

3.4.2 Q Learning 設定

大致上也是和之前一樣，不過因為 slot 的狀態變多了，所以獎勵中和 slot 有關的部份得跟著複雜化。獎勵一改變，SA-Q learning 的參數設定也得重新計算。

3.4.2.1 獎勵

使用的獎勵類型依然不變，只是第一項跟 slot 有關，所以得稍微修改。

- 目前跟各個功能最終目標的距離（尚未填值的 slots 數）

$$D = \sum_{taski} R_u \times U_{taski} + R_g \times G_{taski} + R_c \times C_{taski} - N_{taski} - R_f \times M_{taski}$$

U：未確定的必要 slots 數，G：已確定的必要 slots 數，C：刪除的必要 slots 數，N：必要 slots 的個數，M：非必要 slots 裡未填值的個數，$taski$：第 i 個功能。

定義所得的獎勵仍然是：

$$Reward = W_D D + W_R RV + W_M Mis + W_{cancel} C + W_G G + W_T T$$

目前採用的比重參數各為：

$$R_u = 0.5, R_g = 1, R_c = -1, R_f = 3$$
$$W_D = 1, W_R = -85, W_M = -6, W_{cancel} = -85, W_G = 20, W_T = -1$$

。

3.4.2.2 SA-Q Learning 參數設定

在決定何時該探索的 SA-Q learning[7]參數部份，目前設定的參數值為 $t_0 = 52.2017, \lambda = 0.98$，使系統在剛開始時大約有 0.9 的機率會探索；當此功能成功完成 10 次後，大約有 0.5 的機率會探索；當此功能成功完成 30 次後，大約只剩 0.1 的機率會探索。

3.5 實驗二結果

底下將觀察對話策略學習在各種情況下學習的成效，以及學習到的對話策略是否像預期般會趨近使用者的行為模式。

3.5.1 學習到的最佳策略

在辨認率 100%、可信度夠高的機率 65%的情況下，學到的最佳策略大致上仍跟圖 3.8 一樣，而且對話策略的確能藉由Q learning學到處理slots相依性的方式，也就是在此實驗中學到的最佳策略一定是先問使用者slot2的訊息，等到得知slot2 的訊息之後才會再跟使用者詢問slot3 或slot4 的資訊。至於策略中何時該向使用者確認已填值但辨認結果可信度過低的slots的部份，則是偏向最後所有必要slots都已有值之後才執行確認。之所以會使對話策略喜歡先詢問未知的slots，是因為獎勵裡系數定義當功能中某一

個必要slot的狀態由『未知』變成『已確定』時，系統就向完成此任務靠近一步（$R_g=1$）；如果是由『未知』變成『未確定』，系統則向完成此任務靠近半步（$R_u=0.5$），所以如果是由『未確定』變成『已確定』，系統也只是向完成此任務靠近半步而已（$R_g-R_u=0.5$）。因此跟使用者確認未確定的slots最多也不過一個slot多走半步，但是如果跟使用者詢問未知的slots不但可能每個slot向前半步，甚至有機會一個slot就向前一步，所以系統學到的策略自然就會偏向填完剩下未知的slots。

3.5.2 對話策略收斂情況

對話策略收斂情況如圖 3.15 所示：

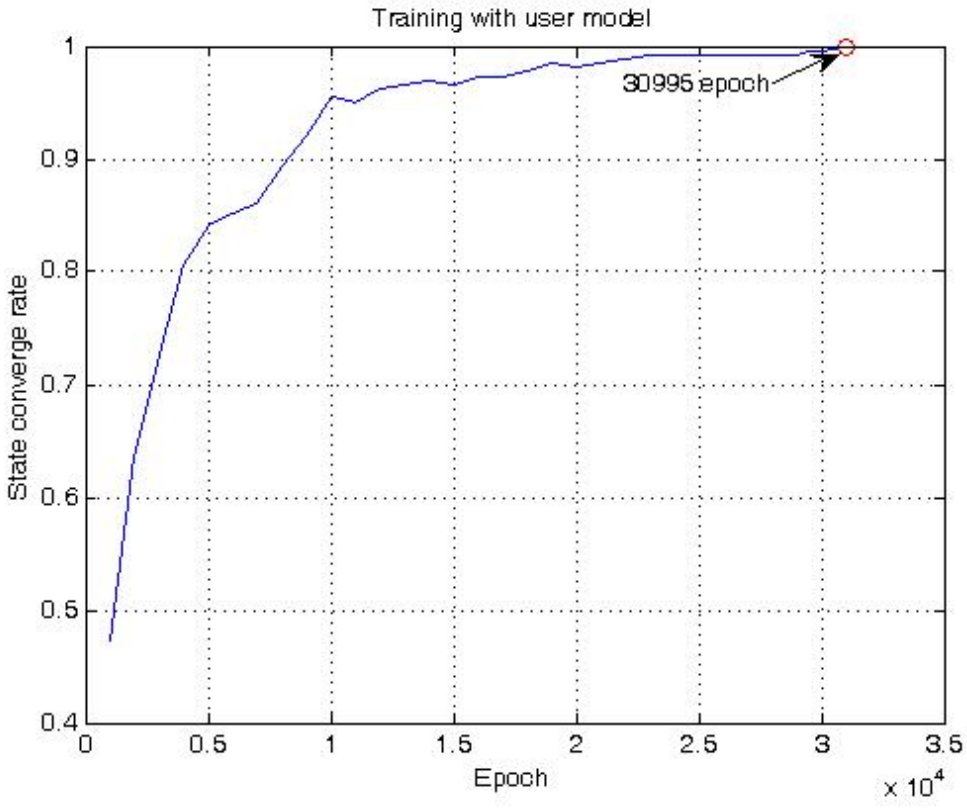

圖 3.15 States 收斂率（實驗二）

判斷對話策略是否收斂的方式是當所有 states 採用的 actions 保持不變

連續 100 個使用次數後，就判定對話策略學習已經收斂。再將最後學得的策略視為最佳策略，以其為收斂基礎求得上圖的 states 收斂率。

由於系統的 states 數明顯變大，而且總 actions 數也是之前的 2 倍多，所以比上一個模擬實驗花更多的使用次數才到達收斂。

3.5.3 進一步的實驗設計

接著為了進一步觀察策略學習學到的策略是否真的會趨近於使用者行為模式，以及辨認率和辨認結果可信度的變化對對話策略學習的影響，於是設計了各種不同的實驗情境。

3.5.3.1 不同的使用者模型

分別設計了四個不同的使用者模型，分別是 user1、user2、user3、user4，假設他們在遇到系統開場白時的回應方式一樣（給予各個 slot 的機率），主要的分別只有在面對系統詢問各個 slot 時的回應方式。大致的差別就如下表所示：

表 3.1 四種不同的使用者模型

	差別
User1	非常消極，每次只肯回答 1 個 slot。
User2	積極度中等，每次約回答 1～2 個 slots。
User3	非常積極，回答所有被問的 slots。
User4	過度積極，不只回答被詢問的，還會主動給予其它 slots。

這四個使用者模型正好代表四種完全不同的行為模式，讓他們在相同的情境下跟對話系統互動，觀察學到的對話策略和使用者之間的交互關

係。

3.5.3.2 不同的辨認環境設定

既然這次的對話策略學習模擬實驗已經將 slots 在實際即時語音辨認器下產生的狀態都考慮進去了，自然得進一步觀察辨認器對策略學習造成的影響。辨認器的影響不外乎辨認率以及辨認出來的結果可信度的高低，所以設計了以下十二種不同的辨認環境：

表 3.2 十二種不同的辨認環境

	辨認率	辨認可信度高的機率
Case1	1	0.9
Case2	0.9	0.9
Case3	0.8	0.9
Case4	0.6	0.9
Case5	1	0.65
Case6	0.9	0.65
Case7	0.8	0.65
Case8	0.6	0.65
Case9	1	0.3
Case10	0.9	0.3
Case11	0.8	0.3
Case12	0.6	0.3

這十二種辨認環境正好包含辨認率高到低，以及辨認可信度高的機率高到低的各種狀況，以便觀察即時語音辨認器對對話策略學習造成的影響。

3.5.4 結果分析

有了上述定義好的各種情況，就能對我們感興趣的部份做深入地分析。

3.5.4.1 不同的使用者分析

在辨認率 100%，辨認可信度夠高的機率是 65%的情況下，由之前設定的四種使用者模型和對話系統互動，依序學得四種對話策略（user1：strategy1、user2：strategy2、user3：strategy3、user4：strategy4）。詳細觀察這四種對話策略的確可以發現，strategy1 幾乎都是每次只問一個slot，就連跟使用者確認可信度低的slot也是每次只確認一個；而strategy2 則是在可問的slots在兩個以上時，幾乎都一次問兩個；而strategy3 也跟預料中一樣，幾乎每次都把剩下未知的slots一次問完；但是strategy4 的部分卻不全是可用的策略，因為user4 太過積極的關係，在現在所定義的獎勵下，造成常常因為對已確定的slots重新填值而扣分，而此情況又是設計時最不希望發生的狀況以至於對此扣分特別嚴重（W_R=-85），再加上回答的和系統詢問的不相符合又再扣分（W_M=-6），使得最後學到的策略不是要完成任務，反而是朝盡量不讓使用者回答slots資訊的方向發展。

再來為了觀察學到的策略的確是最符合使用者行為模式的策略，於是讓這四種使用者模型分別使用和他們互動所學到的四種對話策略，統計他們每個功能都使用 100 次所得到的使用情況。

表 3.3 User1 在各種對話策略下的使用情形

	Task1			Task2			Task3			Task4		
	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值
Strategy1	98	***6.18***	***-164.08***	***95***	***8.73***	***-241.54***	***99***	8.34	-263.77	***98***	***9.07***	***-245.62***
Strategy2	***100***	6.92	-179.83	***95***	9.54	-301.04	***99***	***8.08***	***-254.41***	97	9.03	-253.06
Strategy3	98	7.87	-217.10	76	11.8	-431.02	62	11.75	-459.21	50	14.56	-624.84
Strategy4	81	12.78	-500.67	3	19.53	-913.30	24	16.79	-770.01	2	19.64	-904.57

表 3.4 User2 在各種對話策略下的使用情形

	Task1			Task2			Task3			Task4		
	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值
Strategy1	97	6.76	-188.39	93	***8.72***	***-249.61***	72	9.44	-357.67	95	8.71	-248.19
Strategy2	***100***	7.25	-192.54	***94***	9.25	-293.80	***100***	***5.69***	***-163.74***	***99***	***8.68***	***-230.00***
Strategy3	99	***6.48***	***-168.50***	76	10.03	-374.76	95	7.17	-231.55	55	12.13	-510.11
Strategy4	92	9.02	-308.46	5	19.43	-976.16	22	16.67	-774.30	1	19.70	-902.32

表 3.5 User3 在各種對話策略下的使用情形

	Task1			Task2			Task3			Task4		
	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值
Strategy1	99	7.12	-210.9	96	8.15	-228.95	***100***	5.41	-195.18	***99***	8.06	-201.75
Strategy2	***100***	6.75	-177.12	95	8.13	-298.83	***100***	5.85	-164.15	95	8.03	-231.05
Strategy3	99	***6.51***	***-165.44***	***99***	***7.29***	***-209.25***	***100***	***5.20***	***-143.17***	98	***7.28***	***-196.06***
Strategy4	96	8.07	-263.06	26	16.78	-890.23	26	15.64	-735.85	7	18.98	-867.79

表 3.6 User4 在各種對話策略下的使用情形

	Task1			Task2			Task3			Task4		
	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值	成功次數	平均使用次數	平均客觀值
Strategy1	***100***	6.90	-262.91	99	8.21	-335.78	***100***	5.39	-228.06	***100***	8.30	-351.66
Strategy2	***100***	6.87	-248.20	98	7.94	-375.41	***100***	5.81	-178.62	***100***	7.57	-326.65
Strategy3	***100***	***6.48***	***-208.78***	***100***	***7.03***	***-245.81***	***100***	***5.30***	***-161.08***	***100***	***7.29***	***-265.55***
Strategy4	***100***	7.24	-221.05	15	17.96	-799.76	41	13.58	-612.87	14	18.08	-785.62

表 3.3 到表 3.6 中粗體斜體字的部份就是使用情況最佳的時候，也就是成功次數最多、平均客觀值（Objective value＝total reward）最大。由這些表可以看出，的確幾乎是跟自己互動所習得的對話策略最適合自己，即使有時並非最佳的情況，也離最佳情況不遠。不過 user4 的部份就並非如此，其原因還是因為 user4 太過積極，不適合當作此對話系統學習對話策略的

對象。

3.5.4.2 不同的辨認環境分析

之前的模擬都是在同一個辨認環境下（辨認率 100%，辨認可信度夠高的機率為 65%），但是不同的辨認環境對對話策略的學習一定會有不小的影響，所以模擬了各種不同的辨認環境。在模擬辨認率的部份，是以機率的方式判定辯認到的每個 slot 是否發生辨認錯誤，一但發生辨認錯誤，就視為整個 slot 的內容錯誤，不考慮將 slot A 辨認成 slot B 的情況。

3.5.4.2.1 學到的對話策略

在辨認率固定，改變辨認可信度夠高的機率，讓系統跟上述四種使用者模型互動所學到的對話策略，發現改變辨認可信度夠高的機率對學習到的對話策略幾乎是完全沒有影響，因為學到的對話策略依然還是以趨近使用者行為模式為目標。如果調整辨認率之後再觀察學習到的對話策略則會發現，當辨認率夠高時，對策略學習的影響也是一樣不大，但是如果辨認率太低，使得每次填入 slots 的內容幾乎都是錯的話，就會對策略學習造成破壞，甚至使系統無法學到正確的對話策略。

3.5.4.2.2 對話策略學習收斂速度

不同的辨認可信度變化似乎對策略學習的收斂速度影響不大，但是辨認率的高低對策略學習的收斂速度卻影響不小。表 3.7 就是在使用者模型採用 user3 時，在各種環境下收斂速度的差異。

表 3.7 辨認環境對策略學習收斂速度的影響（User3）

可信度高的機率 / 辨認率	0.9	0.65	0.3
1	30331 epochs	31314 epochs	29880 epochs
0.9	35209 epochs	39274 epochs	37986 epochs
0.8	45235 epochs	47025 epochs	45283 epochs
0.6	64362 epochs	65181 epochs	69222 epochs

3.5.4.2.3 對話策略學習效率和成果

接著觀察辨認環境對策略學習效率的影響。因為對話策略學習的目的是要使系統能在最短的對話次數中完成使用者交代的任務，而系統能完成任務就是能從中學到正確策略的先決條件，所以在學習過程中，系統完成任務的比例越高就表示學習效率越高。表 3.8 是系統和 user3 互動的學習過程中，系統完成任務的比例。

表 3.8 對話策略學習過程中，系統完成任務的比例（User3）

可信度高的機率 / 辨認率	0.9	0.65	0.3
1	0.9524	0.9	0.7376
0.9	0.9241	0.8135	0.7092
0.8	0.8332	0.7213	0.6621
0.6	0.6159	0.5028	0.1908

由表 3.8 可以發現，辨認率和辨認可信度都會影響學習效率，其中辨認率的影響明顯比較大，例如在可信度高的機率固定是 0.9，辨認率由 0.9 降至 0.6 時，系統完成任務的比例由約 0.92 降至約 0.62；而若固定辨認率為 0.9，可信度高的機率由 0.9 降至 0.65 時，系統完成任務的比例只由約 0.92 降至約 0.81，明顯小很多。

再來觀察在這些不同的辨認環境下學習到的對話策略效果如何。表 3.9 是系統在各種辨認環境下跟 user3 互動所學到的各個對話策略，然後再給 user3 在相同環境下使用這些對話系統 100 次的使用情況。

表 3.9 User3 在各種 strategy3 下的使用情況（學習和測試環境相同）

可信度高的機率 / 辨認率	0.9	0.65	0.3
1	0.9925	0.9925	0.8475
0.9	0.9475	0.89	0.7925
0.8	0.82	0.7525	0.745
0.6	0.595	0.505	0.2275

由表 3.9 也可看到，辨認率和辨認可信度都會影響學習效率，其中仍然以辨認率的影響較大，而且當辨認率降至 0.6 左右時，系統能順利完成任務的機率就幾乎只剩一半，可見如何提昇對話系統裡的即時語音辨認器的辨認率，將是一個非常重要的問題。

表 3.10 到表 3.13 是系統在幾種辨認環境下（後面括號裡的數字依序代表『辨認率』和『辨認可信度夠高的機率』）跟 user3 互動所學到的各個對話策略，然後再給 user3 在各種環境下使用這些對話系統 100 次的使用情況。

表 3.10 User3 在 strategy3（1.00，0.90）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.990	5.08	-159.96	0.940	7.44	-408.11	0.555	15.44	-1508.79
0.9	0.985	5.38	-172.66	0.938	7.83	-419.37	0.515	15.80	-1554.55
0.8	0.993	5.75	-189.25	0.903	8.51	-496.35	0.473	16.13	-1607.66
0.6	0.985	6.59	-233.10	0.913	9.38	-570.84	0.448	16.91	-1680.25

表 3.11 User3 在 strategy3（1.00，0.65）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.995	4.73	-120.33	0.993	6.70	-185.52	0.823	12.96	-426.11
0.9	0.995	5.05	-140.91	0.990	6.97	-212.72	0.835	13.19	-428.84
0.8	1.000	5.57	-160.97	0.993	7.33	-233.07	0.833	13.67	-476.26
0.6	0.990	6.34	-216.21	0.993	8.56	-325.43	0.745	14.84	-618.21

表 3.12 User3 在 strategy3（1.00，0.30）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.918	5.51	-272.14	0.968	6.95	-247.86	0.888	12.93	-381.54
0.9	0.933	5.87	-297.66	0.948	7.30	-299.02	0.830	13.28	-418.39
0.8	0.913	6.47	-363.62	0.973	7.88	-300.89	0.833	13.93	-466.20
0.6	0.935	7.11	-349.84	0.955	8.82	-406.84	0.818	14.87	-571.74

表 3.13 User3 在 strategy3（0.90，0.90）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.978	5.04	-156.23	0.853	7.84	-456.21	0.480	13.97	-1421.01
0.9	0.973	5.22	-168.52	0.855	8.20	-476.06	0.478	14.46	-1414.38
0.8	0.965	5.49	-179.09	0.858	8.47	-462.73	0.398	14.61	-1479.95
0.6	0.950	6.75	-261.54	0.828	9.72	-564.23	0.350	15.76	-1666.28

表 3.14 User3 在 strategy3（0.90，0.65）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.950	5.57	-165.63	0.895	8.05	-272.70	0.733	13.81	-475.99
0.9	0.958	5.69	-175.09	0.880	8.56	-298.15	0.713	14.32	-516.12
0.8	0.930	6.53	-222.84	0.885	9.05	-318.51	0.668	13.96	-546.60
0.6	0.915	7.25	-260.44	0.853	10.25	-410.31	0.645	15.10	-659.72

表 3.15 User3 在 strategy3（0.90，0.30）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.915	5.90	-311.95	0.963	7.16	-239.64	0.828	13.69	-382.67
0.9	0.890	6.04	-337.73	0.960	7.29	-259.38	0.840	13.73	-401.37
0.8	0.898	6.65	-398.54	0.965	7.98	-317.73	0.825	14.00	-449.55
0.6	0.925	7.20	-384.54	0.968	8.85	-366.59	0.758	14.93	-562.72

表 3.16 User3 在 strategy3（0.60，0.90）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.735	8.42	-311.53	0.338	14.41	-705.85	0.035	18.26	-1401.08
0.9	0.733	8.91	-333.26	0.310	14.90	-740.75	0.043	18.50	-1471.10
0.8	0.690	9.75	-377.74	0.225	16.04	-783.50	0.013	18.69	-1383.24
0.6	0.595	11.32	-459.12	0.170	16.99	-861.63	0.028	18.59	-1435.67

表 3.17 User3 在 strategy3（0.60，0.65）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.718	8.86	-334.01	0.273	15.85	-688.18	0.025	19.32	-896.51
0.9	0.718	9.10	-342.31	0.278	15.77	-680.01	0.045	19.24	-891.08
0.8	0.653	10.15	-394.78	0.228	16.62	-732.49	0.030	19.35	-902.84
0.6	0.583	11.78	-477.08	0.175	17.50	-777.71	0.018	19.61	-927.55

表 3.18 User3 在 strategy3（0.60，0.30）下的使用情況

可信度高的機率 / 辨認率	0.9			0.65			0.3		
	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質	成功機率	平均對話次數	平均客觀測量質
1	0.585	9.14	-491.44	0.465	13.11	-594.91	0.288	17.49	-694.77
0.9	0.585	8.63	-489.44	0.415	13.92	-650.32	0.245	17.98	-749.26
0.8	0.535	9.84	-576.09	0.440	13.72	-637.80	0.215	18.07	-765.04
0.6	0.458	11.03	-651.35	0.343	15.15	-723.48	0.180	18.85	-819.09

由表 3.10 和表 3.18 可以看出，在辨認率都很高的情況下，不論可信度分布的高低，學習到的對話策略幾乎都適用在各種辨認環境下，除了表 3.10、3.13 和 3.14 中可信度分布偏低的情況下；但是在表 3.16 到 3.18 中，不管可信度分布如何，在辨認率不高的情況下，以致學出來的策略反而不適用於許多辨認環境，尤其在可信度分布偏低的情況下更是明顯。

之所以會有這些情況是因為辨認率太低時，辨認結果錯誤的頻率太高，使得填入各個 slot 的值經常是錯的，即使系統採取到正確的行動，也會因為經常得到錯誤的 slot 值而誤以為此行動也是錯誤的，以致破壞了策略學習的過程；至於可信度分布則影響系統在 slots 狀態是未確定下的 states 的停留次數，此處停留次數越多，則此處策略的學習就越完善，所以在可

信度分布高的環境學到的策略就不適合用在可信度分布低很多的環境下。

第四章 實際對話系統之實現

經過之前的模擬實驗，已證明強化學習的確可以用來設計對話策略，而且可完全透過學習得到最適合的對話策略。接下來就是依上一章模擬的方式，實際架設一套擁有學習能力的對話系統，其中先由之前模擬的方式，透過和使用者模型互動來學習適合的對話策略，而使用者模型的部份，就由架設好的實際系統給實際使用者使用，統計使用者回應系統的方式來決定使用者模型的各參數。

4.1 系統簡介

架設的實際系統為**『智慧型口語對話汽車導航系統』**（Intelligent Transportation System，ITS），功能是讓駕駛在行車中不必以手操作傳統的人機介面，就可直接用口語撥打行動電話、詢問交通資訊以及操作汽車導航系統，系統也會用語音回應，如此便可避免駕駛在行車中因手動使用導航系統分心而發生危險。

圖 4.1 口語汽車導航系統使用情境

目前能提供的功能有**『撥打大哥大號碼』**、**『汽車導航』**、**『汽車導航資訊查詢』**，汽車導航資訊查詢部分目前能查詢的資訊有『停車場』和『加油站』。系統架構圖如圖 4.2 所示。包括即時語音辨認器，以辨認使用者的語音；對話管理模組，做對話流程控制，以及控制大哥大與汽車導航系統和擷取資訊；文字轉語音模組則將對話管理做出的回應轉換成語音輸出。

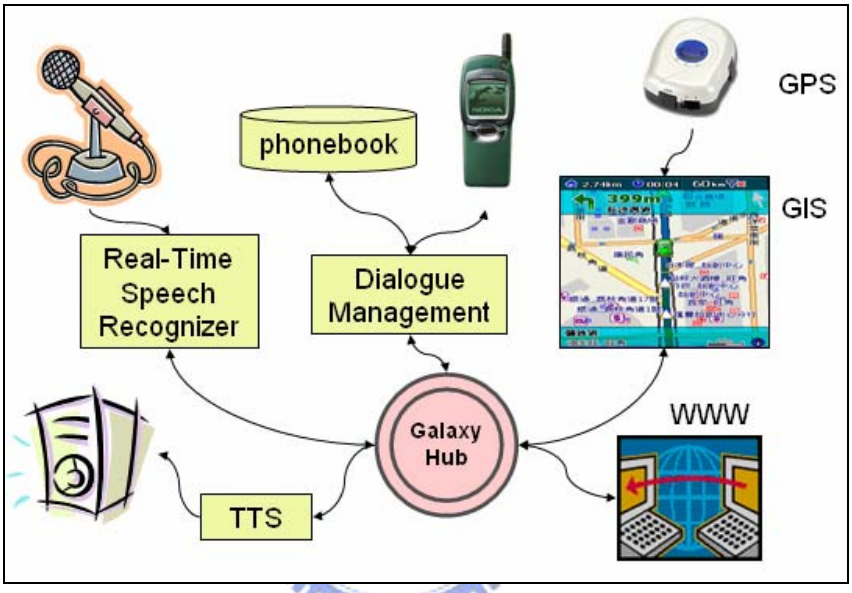

圖 4.2 口語汽車導航系統方塊圖

4.1.1 架設工具簡介

架設智慧型口語對話汽車導航系統使用的是一些好用的公開領域工具程式庫，這些工具程式庫，不但都有完整原始碼，充分的說明文件，更有教學文件（Tutorial），按部就班教導使用者如何入門。

在圖 4.2 中即時語音辨認器裡使用的語音聲學模型與語言模型，我們使用英國劍橋大學釋放出來的 Hidden Markov Model Toolkit（HTK）訓練產生。HTK 可以作大字彙語音辨認，包括使用複雜的 tri-phone 模型[17]，訓練 tri-gram 模型，也可以作 cluster-based 的語言模型[17]。架設即時語音

辨認器的部份，我們是採用同是英國劍橋大學釋放出來的 Application Toolkit for HTK（ATK） [18]，如圖 4.3，除了以便與 HTK 相容外，ATK 還可以在 windows 作業環境下執行，支援多工，所以可以很方便架設一個 windows 下的即時辨認器。此外 ATK 還提供了辨認結果可信度（Confidence）的功能，視每次辨認結果的分數高低附上相對應的可信度，可用來提升對話流程控制能力。

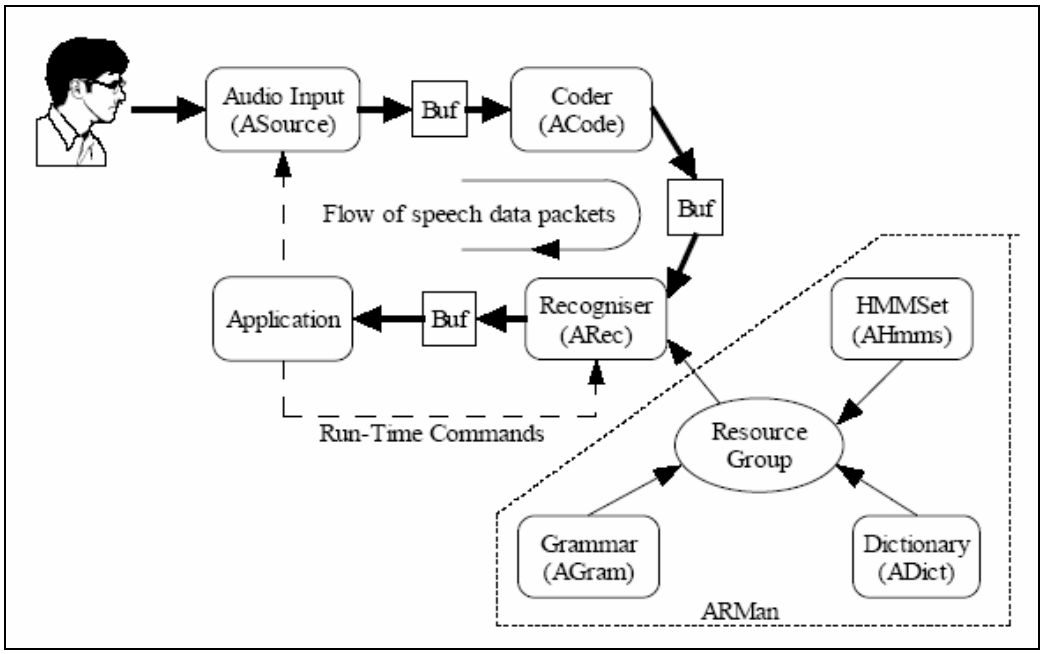

圖 4.3 由 ATK 架設的基本即時辨認器

最後使用 MIT 釋放出來的 Galaxy communicator 將各模組的輸出輸入連接起來。如圖 4.4 所示 Galaxy communicator 為一 Hub-Server 架構，各伺服器可獨立執行，再透過網路連接 Hub 來傳遞訊息。好處是其為分散式系統，各伺服器端可在網路上的不同機器執行，所以各伺服器端可獨立開發執行，方便多人開發系統各個部份，以減少後續的程式碼維護負荷；而且若單台機器計算能力不足，亦可以用多台機器來分散計算負荷。

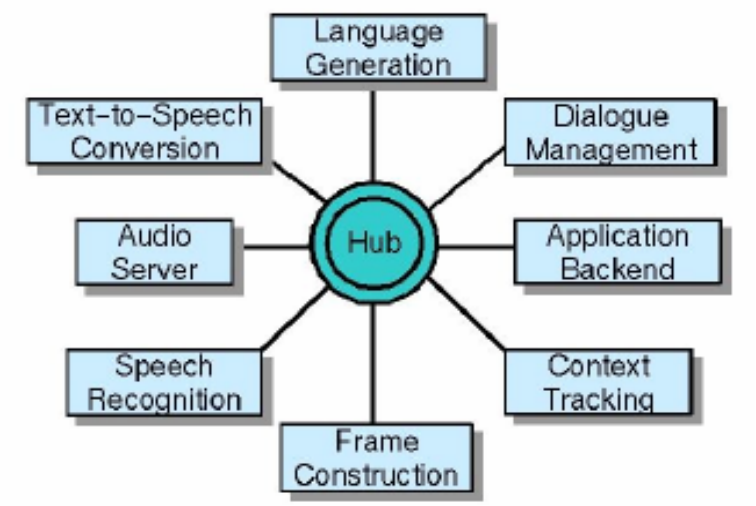

圖 4.4 Galaxy Communicator software 內部結構圖

4.2 系統設定

系統的基本構思有了，架設的工具也準備好了，接下來就是如何設計此系統。由於此系統整合了多重領域的技術，並非能由一人或單一領域單獨完成，所以底下介紹及討論的部份仍以對話管理員和對話策略為主。

4.2.1 語料收集及分析

首先得先知道使用者在使用此系統時，會發生哪些情況，以及除了原本構思中的功能外，使用者會想用哪些相關功能。為了得知此資訊，必須收集分析使用者跟系統實際互動的語料。但是在設計初期，根本沒有系統可以供使用者使用，於是先採用 Wizard of OZ 方式收集對話語料[19]，如圖 4.5 由訓練過的人充當系統來跟使用者互動，再將收集到的語料請國立交通大學語言所的同學加以分析。

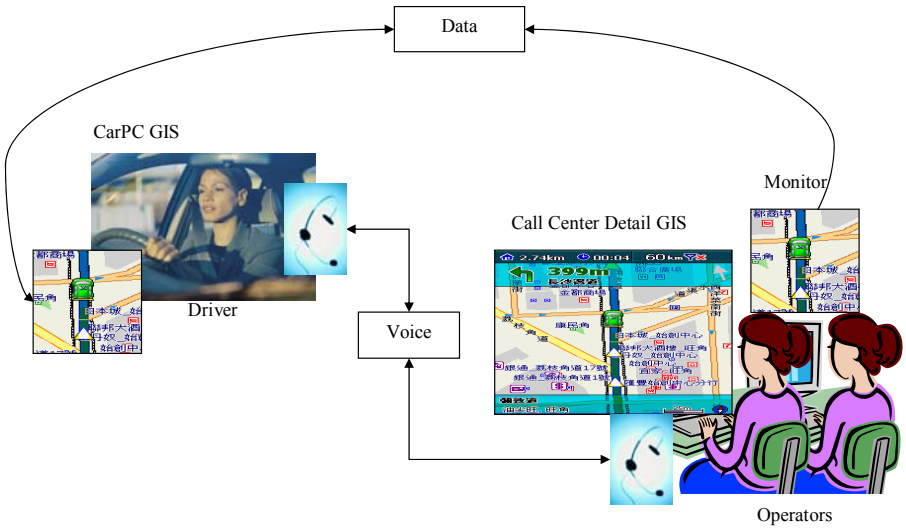

圖 4.5 使用 Wizard of OZ 來收集語料

4.2.2 對話管理員設計

參考國立交通大學語言所同學分析完成的資料可以發現，**汽車導航**的部份除了一開始跟使用者詢問目的地和喜好路徑的初始任務之外，在導航的過程中仍需應付使用者詢問的各種問題，像是在十字路口詢問行車方向；詢問經過點位置；詢問接下來的動作等等，因此也得將這些功能設計到系統中。至於**汽車導航資訊查詢**的部份則由國立清華大學電資所的同學從網路上收集各處停車場和加油站的資訊，以及收集相關對話文句，加以整理分析成各類文法句型，從整理出來的結果也能發現，**汽車導航資訊查詢**除了詢問使用者想查詢的資訊內容及搜尋資料庫外，也得做事後更進一步的服務，例如更詳細列出其中一筆使用者詢問的資訊內容，甚至能用查詢到的結果來執行導航任務。

接著參考上述整理好的資料來設計整個系統的核心『對話管理員』，設計的方式就依照第三章模擬程式的設計方法，將系統要完成任務所需要得知的各項訊息設計成填值表單，再依照各個功能和表單填值的狀態設計成

數個 states。

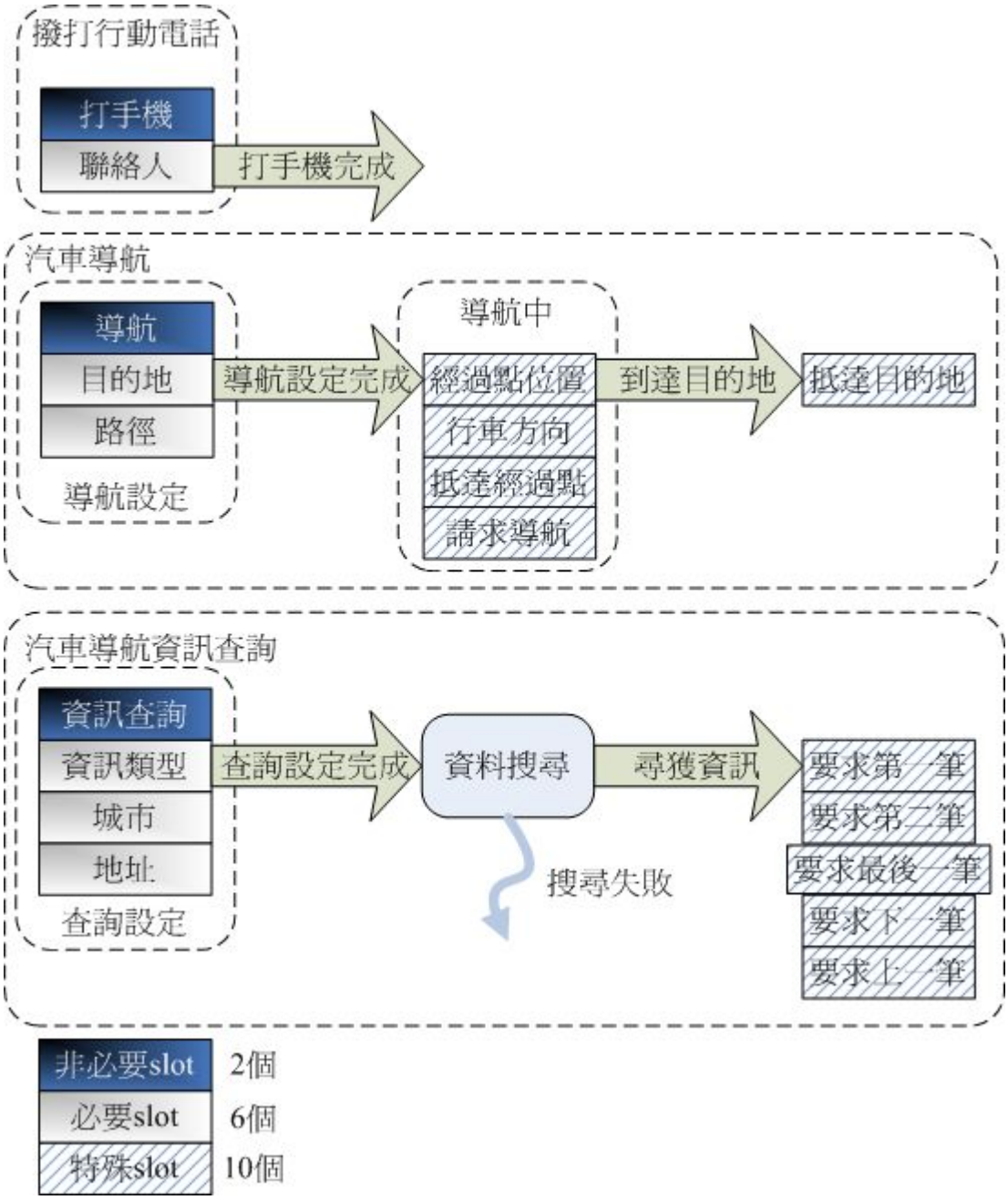

圖 4.6 ITS 系統的各個功能和 slots 的關係圖

系統的功能和表單的關係圖如圖 4.6 所示，圖中每個特殊 slot 代表使用者向系統詢問一種資訊，所以一個特殊 slot 就可以視為一個子功能。如圖所示可以分成三大功能，也就是**『撥打大哥大號碼』**、**『汽車導航』**、**『汽車導航資訊查詢』**，而**汽車導航**和**汽車導航資訊查詢**的部份又能細分成數個

小任務，例如啟動**汽車導航**前要先做『導航設定』，再來是在『導航途中對使用者的指引』，最後才能帶領使用者到達目的地；而**汽車導航資訊查詢**也是得做『查詢設定』，得知使用者想查詢的資料內容後，才能去資料庫搜尋資料，再將結果告知使用者，並等待使用者下一步的指示。

4.2.2.1 States

跟第三章後來設計的方式一樣，每個 slots 的填值狀態有四種，分別是未知（Unknown）、未確定（Unconfirmed）、已確定（Grounded）、刪除（Cancelled），所以詳細的 states 設計如下：（算式裡的 slot 指的都是必要 slots）

- 各種功能相關的states且非必要slot已填值（$2^{\text{num. of slots in task}} \times 2^{\text{num. of total slots}}$）：$2^1 \times 2^6 + 2^2 \times 2^6 + 2^3 \times 2^6 = 896$
- 各種功能相關的states但非必要slot未填值（$(2^{\text{num. of slots in task}} - 1) \times 2^{\text{num. of total slots}}$）：$(2^1 - 1) \times 2^6 + (2^2 - 1) \times 2^6 + (2^3 - 1) \times 2^6 = 768$
- 特殊 slots 相關的 states（特殊 slots 的個數）：10
- 其它例如『導航設定完成』、『資訊查詢設定完成』、『系統關閉』等等：13

所以總共有 1687 個 states。

4.2.2.2 Actions

再來就是定義系統可以採取的行動，除了啟動導航、打行動電話、搜尋資料庫等等特殊的行動外，需要學習的行動定義如下：

- 開場白（只有系統問候語，並不詢問特定 slots）：1
- 詢問想使用何種功能（$2^{\text{num. of tasks}} - 1$）：$2^3 - 1 = 7$
- 確認想使用何種功能（tasks 的個數）：3

- 詢問跟功能有關的 slots

$$\sum_{taski}\left(2^{num.\ of\ slots\ in\ taski}-1\right)-\sum_{\substack{taski\cap taskj\\ i\neq j}}\left(2^{num.\ of\ slots\ both\ in\ taski\ and\ taskj}-1\right)$$

$$+\sum_{\substack{taski\cap taskj\cap taskk\\ i,j,k互異}}\left(2^{num.\ of\ slots\ also\ in\ taski,\ taskj\ and\ taskk}-1\right)-\ldots$$

$$\rightarrow 2^1-1+2^2-1+2^3-1=11$$

- 確認跟功能有關的 slots：11
- 詢問是否確定要關閉系統（當使用者宣布放棄使用系統時，系統需要跟使用者確認是否確定要關閉系統）：12

總共有 45 個 actions。

4.2.2.3 對話策略設計

對話策略的部份也跟之前模擬的方式一樣，由建立好的 1687x45（state x action）的 Q 矩陣裡，尋找目前系統所處的 state 下，Q 最大的 action。

4.3 對話策略學習

依舊採用和虛擬使用者互動的方式，使用 SA-Q learning [7]來學習適當的對話策略。

4.3.1 策略學習相關設定

『SA-Q learning』[7]詳細的參數設定如下：$t_0=33.5598, \lambda=0.924$，可以使系統在剛開始時大約有 0.8 的機率會探索；當成功完成 10 次後，大約有 0.6 的機率會探索；當成功完成 30 次後，大約只剩 0.1 的機率會探索。

使用的『獎勵』定義也是和模擬的部份一樣，各個比重參數的設定如下：

$$W_D = 1, W_R = -30, W_M = -8, W_{cancel} = -60, W_G = 10, W_T = -1, W_{slot} = 1, W_{flag} = 0.5,$$
$$R_{unc} = 0.5, R_{gro} = 1, R_{can} = -1, R_{flag} = 0.1$$

至於『虛擬使用者』的部份，則用架設好的系統使用事先人為設定的對話策略，實際給真實的使用者使用，將使用的過程錄製下來，並整理分析以建造符合真實情況的虛擬使用者。目前使用的虛擬使用者模型，是由錄製了 10 個人的音檔統計出來的，其中每個人每個功能使用 2 次，所以每個人使用系統 6 次，因此是由系統被使用 60 次的統計資料得到的模型參數。以使用者回應系統問候語（開場白）為例，大致的使用者行為如表 4.1 所示：

表 4.1 使用者對系統問候語（開場白）的回應方式（機率）

使用者回應方式 / 使用者使用功能	表明欲使用何種功能	給予：聯絡人[I] 目的地[II] 資訊類型[III]	給予：路徑[II] 城市[III]	給予：地址[III]
撥打大哥大[I]	0.88	0.5		
汽車導航[II]	0.46	0.63	0.13	
導航資訊查詢[III]	0.33	0.79	0.13	0.46

由上表可以發現，使用者在使用第一種功能時，幾乎都會直接表明要打電話，甚至有一半的機率會直接告知想聯絡的對像；而在使用汽車導航時，則較常只告知目的地，很少會主動告知喜好路徑；要做導航資訊查詢

時，則偏好先告知系統想查詢的資訊類型，而給予地址時經常省略了所在城市。

4.3.2 學習到的最佳策略

最後學習到的最佳策略如圖 4.7 所示。

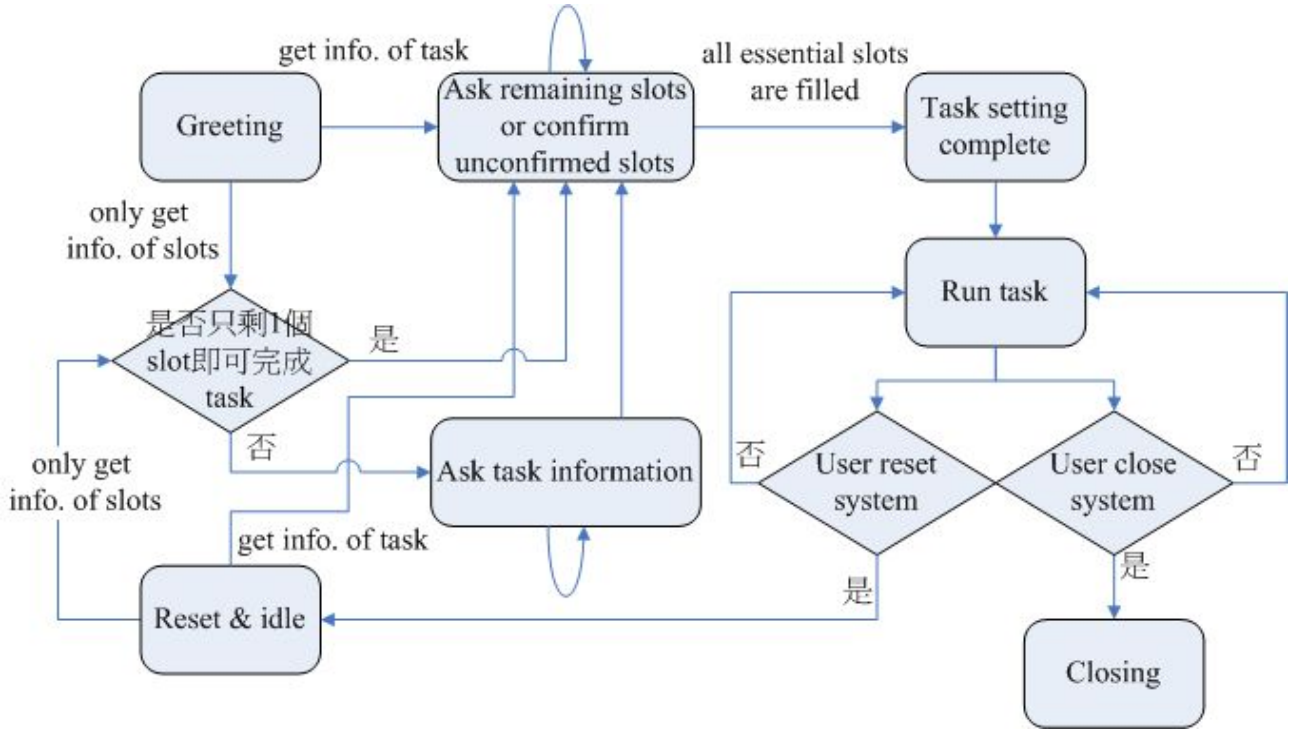

圖 4.7　ITS 最佳對話策略

以儘快完成任務設定為主，設定完成後就繼續提供使用者使用任務相關的功能。再深入觀察學到的對話策略可以發現其的確符合使用者模型的行為模式，例如導航資訊查詢方面的策略較偏向詢問使用者所要查詢的資訊類型，那是因為使用者偏好回答此訊息，而且還有少許機率會連帶給予剩下未知的訊息，甚至促使任務設定馬上完成。

4.3.3 對話策略收斂情況

對話策略的收斂情況就如圖 4.8：

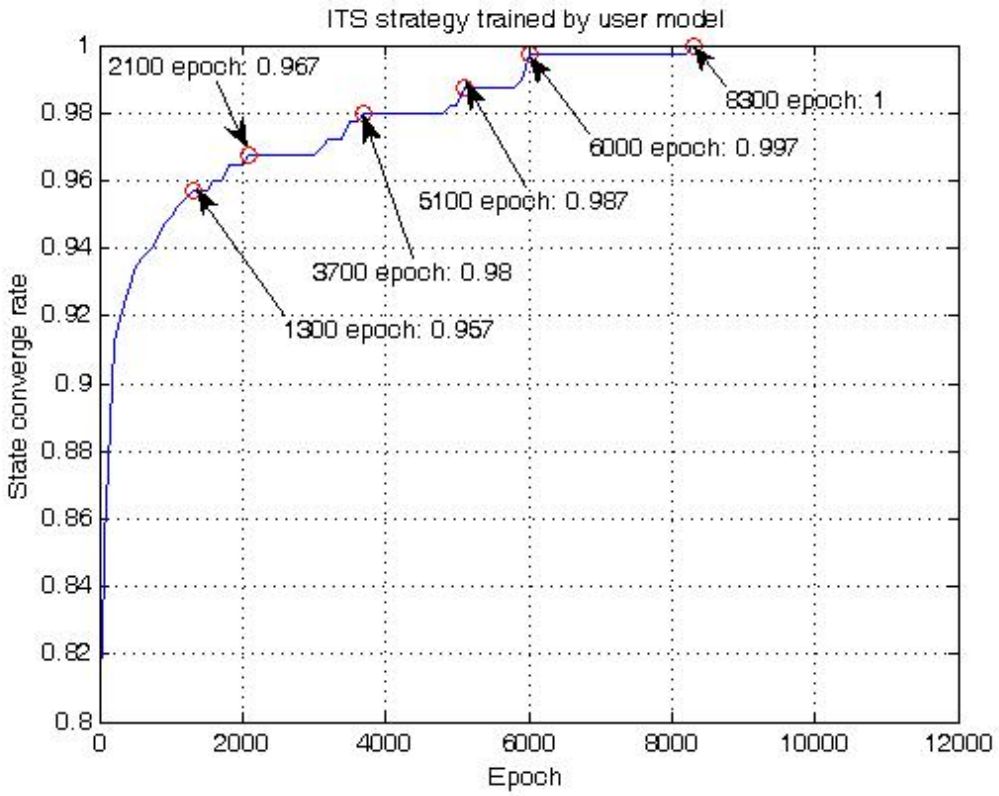

圖 4.8 ITS 對話策略收斂情況

判斷策略是否已經收斂的方式仍藉由觀察所有 states 所採取的 action 是否不再變化，當系統連續被使用 100 次中，所有 states 所採取的 action 都保持不變時，就判定對話策略學習已收斂。由於此系統對話策略的複雜度並沒有像之前模擬程式設計的範例那麼複雜，所以大約 150 個使用次數後，state 收斂率就已經超過 90%。

4.4 對話系統成果展示

整合實驗室原有的 TTS 技術，利用 ATK[18]架設即時語音辨認器，使用國立交通大學語言所和國立清華大學電資所同學整理好的文句分析和

資料庫，再利用實驗室購買的 PaPaGO[20]地圖資訊和函式庫來架設汽車導航系統，最後再將經由學習得到的最佳對話策略套到設計好的對話管理員中，以對話管理員經由 Galaxy communicator 控制整個對話系統的各個模組，於是整個對話系統就架設完畢，而且還是擁有學習能力的對話系統。整個對話系統運作的情況就如圖 4.9 所示，以下將對各個功能分別展示。

圖 4.9 智慧型口語對話汽車導航系統，使用者在導航中撥打電話

4.4.1 打手機

使用者使用此系統口語撥打行動電話的範例如下：

System	歡迎使用 ITS 口語對話系統，請問需要什麼幫助呢？
User	我想打電話給陳信宏。
System	你想要的聯絡人是否為[陳信宏]？
User	是的。
System	打電話，陳信宏。
系統撥打電話→接通→使用者掛斷電話	
System	你還要打電話嗎？
User	不必了，謝謝。

4.4.2 汽車導航

使用者使用此系統執行汽車導航：

System	歡迎使用 ITS 口語對話系統，請問需要什麼幫助呢？
User	請帶我到聯發科。
System	請問你要走的路徑？
User	求助。
System	請選擇最佳路徑，高速公路，主要道路避免或是最短路徑！
User	最佳路徑。
System	衛星導航，聯發科，最佳路徑，設定完成。
導航設定完畢→路徑規畫及簡報→啟動導航→導航中	
System	請右轉。請靠右行駛。
User	再來要怎麼走？
System	前方 500 公尺請右轉。
…到達目的地	
System	你已經抵達目的地了嗎？
User	是的，謝謝。
System	謝謝使用。

4.4.3 汽車導航資訊查詢

使用者使用此系統詢問導航資訊：

System	歡迎使用 ITS 口語對話系統，請問需要什麼幫助呢？
User	我想查光復路附近的加油站。
System	請問你要詢問的資訊是位於哪個縣市？

User	新竹市。
System	你想要查的地址是否為[高峰路]？
User	不是，是光復路。
導航資訊查詢設定完畢→資料搜尋	
System	資料搜尋中，請稍候...
→顯示搜尋結果，等待使用者指示。	
System	請問想知道哪一筆的詳細資訊？
User	請告訴我第二筆資料。
System	第二筆：光復路站　新竹市 光復路二段７１６號 035-718604　00:00-24:00　　中油
System	請問想知道哪一筆的詳細資訊？
User	謝謝再見。

第五章 結論與未來展望

5.1 結論

在本論文中，已介紹如何架設一套對話系統，也詳細說明如何利用強化學習的方式來設計對話策略，此外也證實了由此方式來設計的對話策略的確能接近使用者的行為模式，如此便能替設計者省去每次都得重新自行設計對話策略的麻煩，而且還能保證一定是最有效率對話策略。

此外更深入觀察對話策略學習的情況發現，學到的最佳策略主要都是以趨近使用者行為模式為目標，不易受到外在的即時語音辨認器所影響，但是當學習過程中辨認率過低時，就會破壞系統學習到的對話策略，使最後學到的策略已不再適用於各種辨認環境；然而如果是在辨認率高的環境下學習對話策略，即使將學到的策略用在其它不同的辨認環境下仍然擁有較佳的執行效率，所以建議在讓系統學習對話策略時，應假設處於高辨認率的環境下。

5.2 未來展望

本論文假設即時語音辨認器的辨認率只影響填入 slot 的內容正不正確，並未考慮辨認錯 slot 種類以致填錯 slot 的情況，所以在第三章後面模擬辨認環境的部份雖然已可看出大略的結果，但是仍然不算完整，所以之後可以再將模擬辨認器的部份補足。此外在建立使用者模型的部份，因為目前收集的互動語料有限，所以在設計時採用較少參數的建立方式，當收集使用者使用系統的語料夠多之後，就可以建立較多參數的使用者模型，

模擬出來的虛擬使用者也就更像實際的使用者，系統也就能從互動中學到更佳的對話策略。

此外在實際的對話系統『智慧型口語對話汽車導航系統』上，可以試著採用由語言模型（Language model）架設出來的即時語音辨認器，並設計新的文字翻譯和會話分析模組，使系統更能在實際口語的情境下使用。再來可以朝提升辨認率的方向努力，例如消除語音中的雜訊，增加辨別真正語者語音位置的精準度等等，如此不但能提升系統完成任務的效率，也可以使之後想繼續透過跟真實使用者互動來微調對話策略的動作更為順利。

參考文獻

[1] E. Levin, R. Pieraccini, and W. Eckert, " A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies ," in *IEEE Transaction on speech and audio processing, VOL. 8, NO. 1, Jan 2000.*

[2] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, no. 4, pp. 237–285, May 1996.

[3] R. S. Sutton and A. G. Barto, " *Reinforcement Learning An Introduction*." Cambridge, MA: MIT Press, 1998.

[4] Watkins, Christopher J.C.H. and Dayan, Peter (1992), "Technical Note: Q-Learning," *Machine Learning* 8:279-292.

[5] W. Eckert, E. Levin, and R. Pieraccini, "User modeling for spoken dialog systems," in *Proc. IEEE ASR Workshop*, Santa Barbara, CA, 1997.

[6] 郭建良, "口語對話系統之電腦輔助設計及分析", 台大, 電信工程研究所, 碩士論文, 2000

[7] Maozu Guo, Yang Liu, and Jacek Malec, "A new Q-learning algorithm based on the metropolis criterion," in *IEEE Transactions on systems, man, and cybernetics-part B*, VOL.34, NO.5, Oct 2004.

[8] M. A.Walker, D. J. Littman, C. A. Kamm, and A. Abella, "PARADISE: A framework for evaluation of spoken dialog agents," in *Proc. 35th Ann. Meeting Assoc. Computational Linguistics*, Madrid, Spain, 1997.

[9] E. Levin, R. Pieraccini, and W. Eckert, "Using Markov decision process for learning dialog strategies," in *Proc. ICASSP 98*, vol. 1, Seattle, WA, May 1998, pp. 201–204.

[10] Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*(5), 834–846.

[11] Kumar, P. R., & Varaiya, P. P. (1986). *Stochastic Systems:*

Estimation, Identification, and Adaptive Control. Prentice Hall, Englewood Cliffs, New Jersey.

[12] Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. "*Machine Learning*," *13*.

[13] Peng, J., & Williams, R. J. (1993). Efficient learning and planning within the Dyna framework. "*Adaptive Behavior*," *1* (4), 437–454.

[14] Sutton, R. S. (1988). Learning to predict by the method of temporal differences. "*Machine Learning*," *3* (1), 9–44.

[15] Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In "*Proceedings of the Seventh International Conference on Machine Learning*" Austin, TX. Morgan Kaufmann.

[16] Sutton, R. S. (1991). Planning by incremental dynamic programming. In "*Proceedings of the Eighth International Workshop on Machine Learning*," pp. 353–357. Morgan Kaufmann.

[17] Hidden Markov Model Toolkit (HTK)，http://htk.eng.cam.ac.uk/

[18] Application Toolkit for HTK（ATK），http://mi.eng.cam.ac.uk/~sjy/software.htm

[19] J. Glass, J. Polifroni, S. Seneff and V. Zue, "Data Collection and Performance Evaluation of Spoken Dialogue Systems: The MIT Experience," *Proc. 6th International Conference on Spoken Language Processing*, Beijing, China October 2000. http://www.sls.csail.mit.edu/sls/publications/2000/685.pdf

[20] PaPaGO!趴趴走，http://www.papago.com.tw/