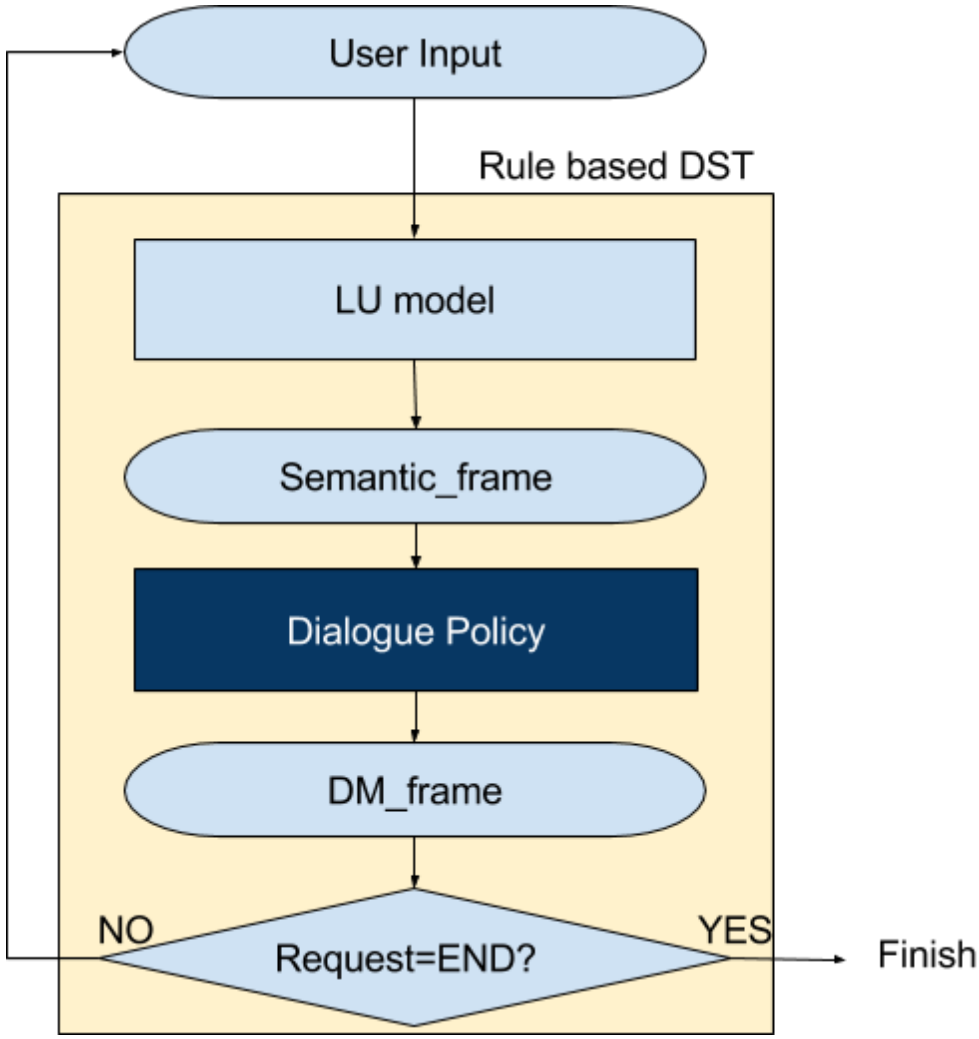
Milestone2 Report

林佳穎 林子翔 蘇柏元 徐彥旻 Hubert Lin 林鑫彤 翁福德

1. Dialogue State Tracking

我們實作一個Rule-based DST，先用LU model處理sentence，取得Semantic_frame。Dialogue Policy會將Semantic_frame與目前的State進行邏輯判斷，並取得DM_frame。最後判斷DM_frame是否結束對話，若否，則將DM_frame回傳給User Simulator要求提供下一個資訊。



Dialogue Policy

若 intent = 1查詢症狀，disease的slot沒有，則Request = info，向使用者詢問disease。若得到disease這個slot的內容，則Request = end。

若 intent = 2查詢科別，disease的slot沒有，則Request = info，向使用者詢問disease。若得到disease這個slot的內容，則Request = end。

若 intent = 3查詢醫生清單，disease和division的slot都沒有，則Request = info，向使用者詢問disease或是division。若得到disease或是division的slot內容，則Request = end。

若 intent = 4查詢醫生門診時刻表，disease,division和doctor的slot都沒有，則Request = info，向使用者詢問disease,division或是doctor。若得到disease或是division的slot內容，則用disease或是division從資料庫中查詢醫師清單，並回傳Request = choose要求使用者選擇一位醫師。若得到doctor，則Request = end。

若 intent = 5預約掛號，disease,division和doctor的slot都沒有，則Request = info，向使用者詢問disease,division或是doctor。若得到disease或是division的slot內容，則用disease或是division從資料庫中查詢醫師清單，並回傳Request = choose要求使用者選擇一位醫師。再用doctor從預約網站直接爬取時刻表，並回傳Request = choose要求使用者選擇一個時間。得到doctor & time，則Request = end。

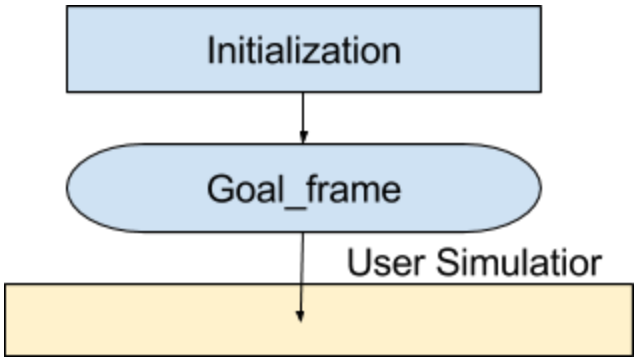
Semantic_frame 結構

{“intent”:5, (可能是1,2,3,4,5
“slot”:{
“disease”:”青光眼”
“division”:"眼科"
“doctor”:"朱筱桑"
“time”:"106.5.18"}}

DM_frame 結構範例

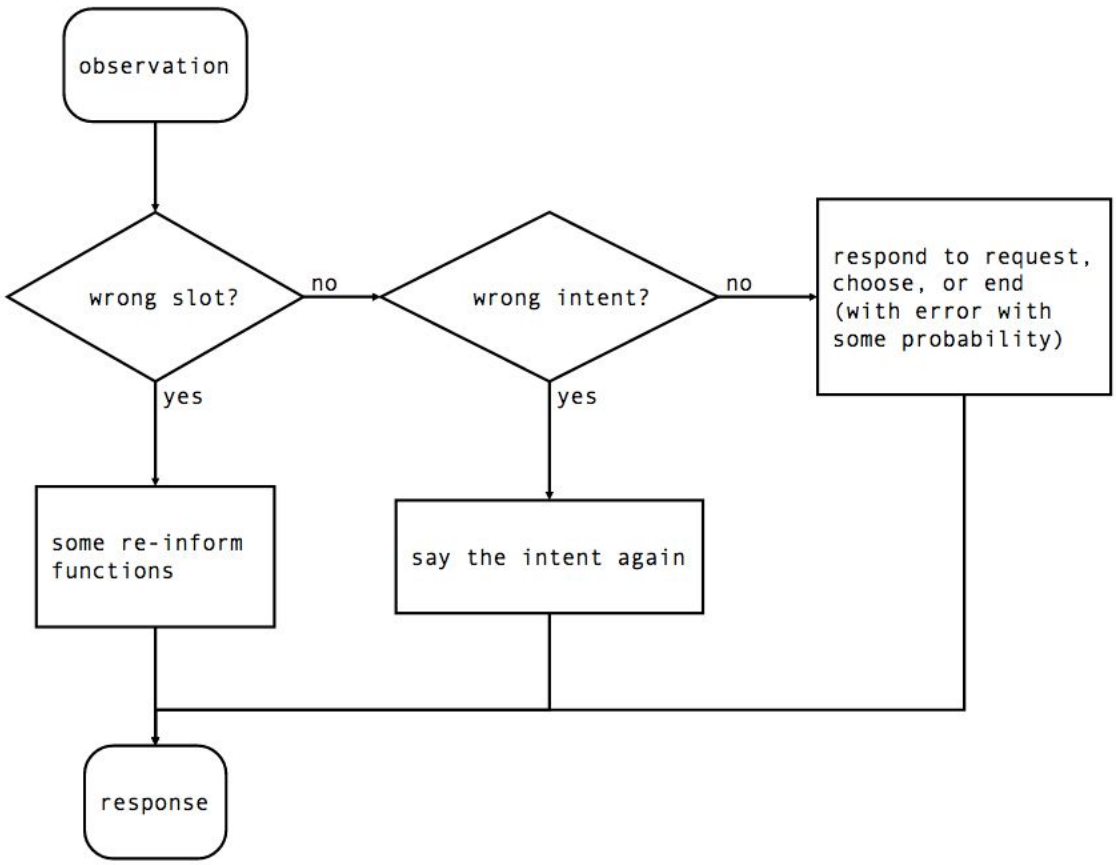
{"Request" : ”choose”, (有end,info,choose三種
"Intent" : 5, (可能是1,2,3,4,5
"Slot" : [”Doctor”], (list可能[“Intent”,”Disease”,”Division”,”Doctor”,”Time”]
"State" : {
Intent:[],
Disease:[‘青光眼’],
Division:[],
Doctor:[],
Time:[]}}

2.User Simulation



- Define the goal_frame
  goal_frame 結構:
  {“intent”:5, (Random決定:1,2,3,4,5
  “slot”:{
  “disease”:”青光眼” # 從資料庫中Random決定disease
  “division”:"眼科" # 從資料庫中查詢
  “doctor”:"朱筱桑" (從資料庫中查詢
  “time”:"106.5.18"}} (從預約掛號網站中即時爬取

Flow chart:



Components:

我們以 User.py 、UserSimulation.py、 UserSimInterface.py 三個檔案實作出 user simulator 以及讓助教與之互動的介面。在 User.py 裡頭，定義了 class User，比較重要的 data member 解釋如下：

- self.intent : 代表這位模擬出來的使用者想要問症狀(1)、問科別(2)、問醫生(3)、問門診時間(4)、或是要掛號(5)
- self.slot : 是個 dictionary，在每個 slot 對應這位模擬出來的使用者想要做的事的細節資訊，比方說如果該使用者想要掛號，其 self.slot 可能就會像是 { "disease": "過敏性鼻炎", "division": "耳鼻喉科", "doctor": "林怡岑", "time": "星期五" }，而倘若對於使用者的需求來說，有不需要或不應該出現的 slot，則將其設為 None，比如想要問症狀的使用者 self.slot 可能就會是 {"disease":"失眠", "division": None, "doctor": None,"time": None}
- self.observation: DST 傳來的 semantic frame，與前面提到的 **DM_frame** 是相同的。
- self.state: 模擬的使用者已經跟 DST 說過哪些訊息，將有說過的訊息設為 True，初始狀態為全部都是 False。舉例來說：{"disease": True, "division": True, "doctor": False, "time": False}，代表使用者已經講過症狀跟科別了，另外兩個的訊息則還沒有提供，或者是在這個使用者想達到的事情裡不需要提供。

而在 UserSimulation.py 當中，則是連接到資料庫，生成可能的 User intent 以及對應的 slot，這邊的限制是我們生成的使用者都是能在資料庫裡頭找到答案的。最後我們用 UserSimInterface.py 這份檔案整合前面兩者，做出可以跟模擬出來的使用者互動的介面。

Error model construction

在 class User 加入實作 error 的 static data member:

- ERROR_RATE: {"intent":0, "disease":0, "division":0, "doctor": 0.5, "time":0 } 設置 intent 與各種 slot 講錯的機率，比方說 "doctor": 0.5，代表使用者有 0.5 的機率在 DST 詢問 doctor 這一 slot 時講錯的機率。（本次只有實作將醫生名字講錯的狀況）

- WRONG_DOCTOR_LIST = ["李琳山", "李鴻毅", "陳縕儂", "廖世文", "楊佳玲"]

  如果機率決定的結果是會講錯的，生活忙碌以至於精神不濟的菸酒生使用者就會不小心把腦袋裡偉大教授的名子講出來。將 ["李琳山", "李宏毅", "陳縕儂", "廖世文", "楊佳玲"] 換成能夠產生跟原本要講的醫生名字相似的醫生名字的函數，是之後我們可以做的方向。

Reward Setting:

狀況	Reward
成功完成任務	20
任務失敗	-100
問使用者已經說過的資訊	-3
正常對話	-1

3. Facebook database

可以在網站上輸入seek-doctor.com或facebook搜尋seek-doctor，輸入後會產生句子。Facebook user input的database已經跟DST_joint.py的database串接起來，只是現在DST_joint.py跟Facebook 的database還無法同步，所以在網站或FB輸入端地回應會不同步。

4. Demonstration (5%)

請見：給助教的使用指南2.pdf