ICB Milestone-3 Report

- Group Member:
  - R04922029 薛祐婷
  - R05922062 楊宗翰
  - R05944004 巫承威
  - R05922136 黃博政
  - R05922131 祝子軒

- Demo site:
  - Rule-Based: http://ntu-course-chatbot.ml (http://ntu-course-chatbot.ml/)
  - RL-Based: http://ntu-course-chatbot.ml/rl (http://ntu-course-chatbot.ml/rl)

• Speech/multimodal API

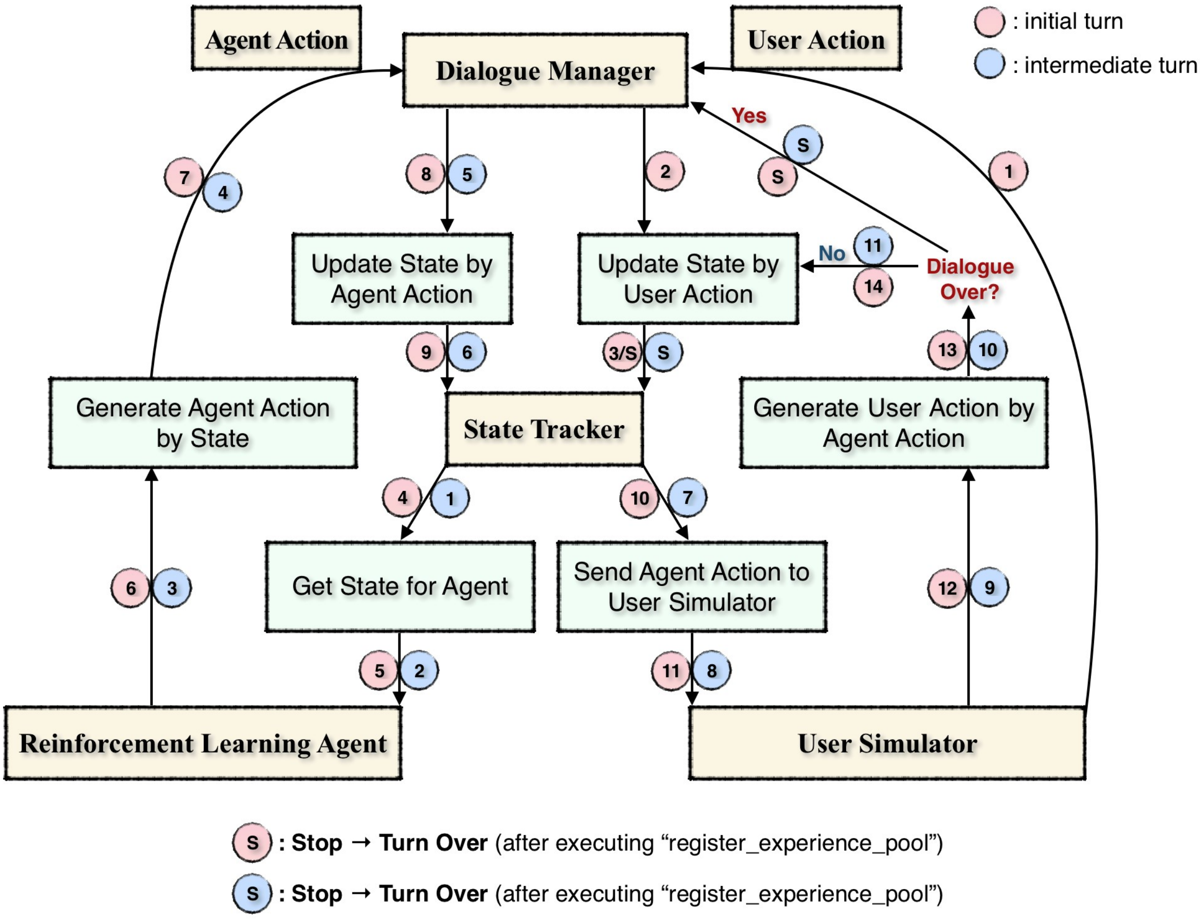
Google Chrome Speech API
We choose Chrome speech API for its simplicity and robustness. With a few lines of code, we could integrate it with our web interface quickly. To our surprise, the recognition performs quite well that most of the entities(陳縕儂， 林軒田， 智慧對話機器人) are correctly spelled. We also synthesize server response to speech. In such manner, we could create a more interactive chatbot that acts like real human.

• Reinforcement Learning based Dialogue Policy

1. Dialogue Policy Optimization

- **Reinforcement Learning Agent (based on TC-Bot (https://github.com/MiuLab/TC-Bot))**
  After referencing to the TC-Bot[1], we built our RL-Agent based on its system structure with some modifications to fit our original system.

  - **System Overview**
    The following diagram shows the overview of our system:



We will briefly introduce each step in **initial turn** of a dialogue as follows:

1. User Simulator (**Usersim**) will **sample an initial action randomly**
2. Dialogue Manager (**DM**) passes the user action to its state tracker (**ST**) (Note that the **state** tracked in the ST equals to the **observation**, and it will be **transformed by agent** into the **real state**)
3. ST updates its current state (with **user action**)
4. ST prepares the state for reinforcement learning agent (**RL-Agent**)
5. RL-Agent gets the state from ST
6. RL-Agent **transforms** the state into the **real satate** (which will be the **input** to **Deep-Q-Network**)
7. RL-Agent uses either **rule-based** or **RL-based policy** (ϵ**-greedy**) to determine its action
8. DM pass the agent action to its ST
9. ST updates its current state (with **agent action**)
10. ST sends the last agent action to Usersim
11. Usersim checks whether the agent action reaches its goal or not
12. Usersim generates user action according to the current state sent from ST and also the last agent action
13. Usersim returns both the **user action** and a **signal** which points out whether the dialogue is over or not to DM
14. If the dialogue is not over, DM will send the user action to its ST and then the ST

will update its current state

Since the intermediate processes of each turn are similar with each other, here we only list the initial turn of a dialogue. The other processes will start from the step-4 and so on.

In our system, RL-Agent will predict the **action index** first, and then **query the database** under the constraints of infrom slots which are tracked in the ST's current slot. Finally, it will **fill in the corresponding values** for each **slot** and then send it (**semantic frame**) to the Natural Language Generartion (**NLG**) model to generate the natural language to the Usersim/real user.

- **State/Observation Transformation**

  In our implementation, we adopted several features such like **user action**, **agent action**, etc. For each action, we use **One-Hot Encoding** method to record the slots that appeared in either **inform_slots** or **request_slots** or both of them since we have defined the **action set** and **slot set** in the beginning. In addition, we also addressed the current state stored in the ST, number of turns of the current dialogue state, number of turns in the format of one-hot encoding and the length of query results under the constraints of inform slots.

  Let's take the following observation as an example:

  - **Pre-defined Dialogue Action Set**

```
0  request
1  confirm
2  multiple_choice
3  inform
4  closing
5  thanks
6  request_title
7  request_instructor
8  request_schedule_str
9  request_classroom
10 request_designated_for
11 request_required_elective
12 request_sel_method
13 inform_unknown
14 deny
15 other
```

  - **Pre-defined Slot Set**

```
0 serial_no
1 title
2 instructor
3 classroom
4 schedule_str
5 designated_for
6 required_elective
7 sel_method
8 when
```

- `user_action`： `'inform'`

```
user_act_rep: [0, 0, 0, 1, 0, ..., 0] (len: 16)
```

- `user_inform_slots`： `['title', 'instructor']`

```
user_inform_slots_rep: [0, 1, 1, 0, ..., 0] (len: 9)
```

- `user_request_slots`： `[]`

```
user_request_slots_rep: [0, 0, ..., 0] (len: 9)
```

- `current_inform_slots`： `['classroom', 'designated_for']`

```
current_slots_rep: [0, 0, 0, 1, 0, 1, 0, 0, 0] (len: 9)
```

- `agent_action`： `'multiple_choice'`

```
agent_act_rep: [0, 0, 1, 0, 0, ..., 0] (len: 16)
```

- `agent_inform_slots`： `['title']`

```
agent_inform_slots_rep: [0, 1, 0, ..., 0] (len: 9)
```

- `agent_request_slots`： `['required_elective']`

```
agent_request_slots_rep: [0, ..., 0, 1, 0, 0] (len: 9)
```

- `turn`： `6` (assume `Max Turn`：20)

```
turn_rep: [1] (len: 1)
turn_onehot_rep: [0, 0, 0, 0, 0, 0, 1, ..., 0] (len: 20)
```

- `query_results`： `[course_1, course_2, course_3]`

```
query_results_rep: [3] (len: 1)
```

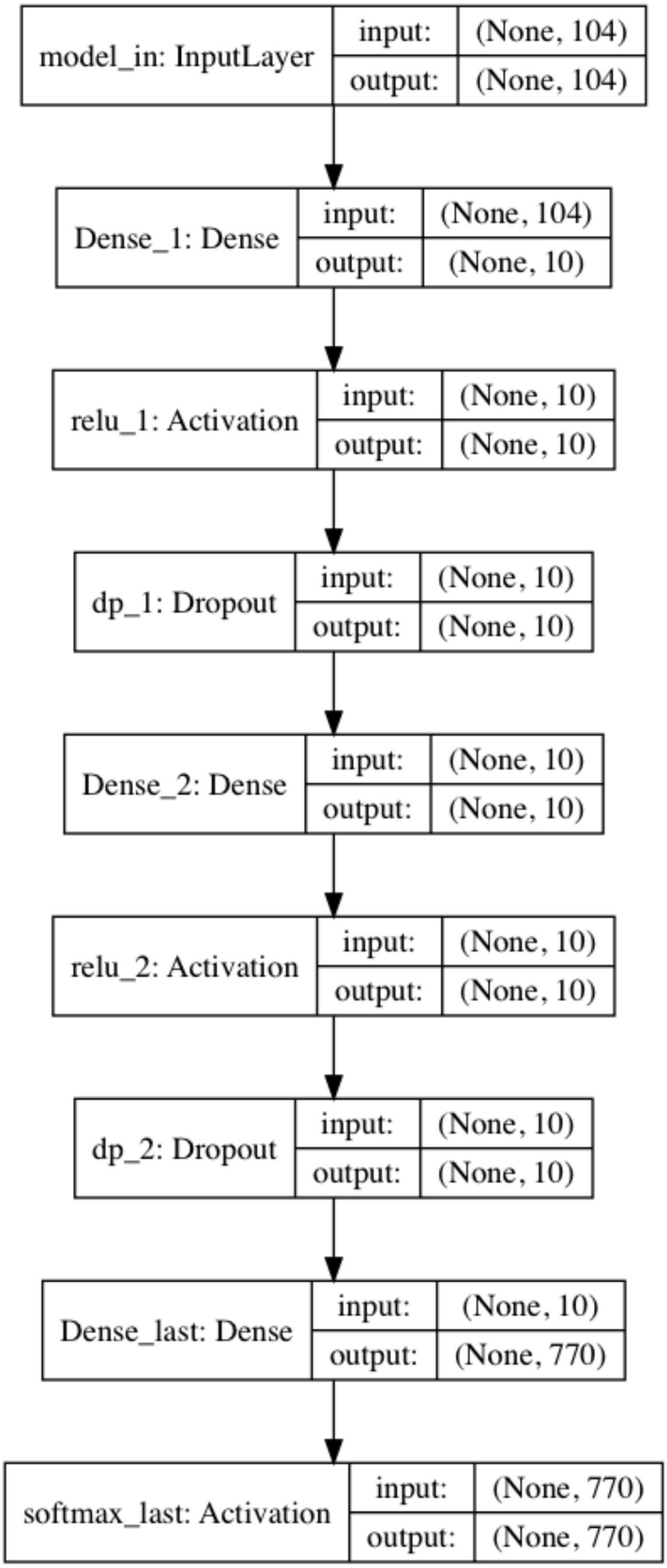
So the dimension of our state equals to:

```
16 + 9 + 9 + 9 + 16 + 9 + 9 + 20 + 1 = 99
```

But since the actual turns may exceed the limitation of `Max Turn`, so we plus additional `4` turns to it. Thus, the final dimension of the state will be `99 + 4 = 103`.

- **Deep-Q-Network (DQN) Model Structure**

We used **Keras** to build our **DQN** model. The structure of our model is shown as follows:
(the input dimension in this figure is the old version of our model, we have tried other different structures and features continuously):



Follow the structure of TC-Bot (https://github.com/MiuLab/TC-Bot), we also use **Experience-Replay Pool** and **ϵ-Greedy** to enhance the performance of our **DQN RL Model**.

- **Experience Replay Pool**
  Since the approximation of Q-value using non-linear functions like the above neural network is not very stable.The most important trick to solve this problem

is called **Experience Replay**. During the dialogue process, all the episode (s, a, r, s') are stored in replay pool D (we use a python list to store it). When training the network, random mini-batches sampled from the replay pool are used instead of most the recent transition, which will greatly improve the stability.

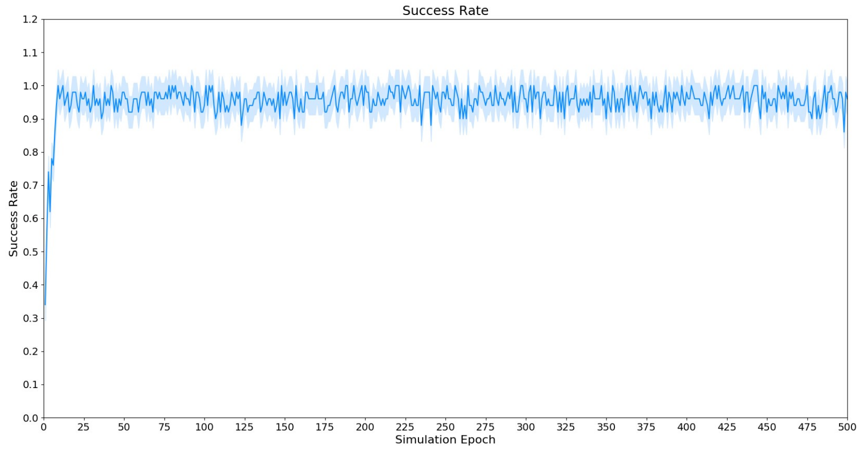
- **ϵ-Greedy (Exploration vs. Exploitation)**

  There is another issue in the reinforcement learning algorithm which called Exploration vs. Exploitation. How much of an agent's time should be spent on exploiting its existing known-good policy, and how much time should be focused on exploring new, possibility better, actions? These scenarios often happen in real life too. For example, we face on which restaurant to go to on Saturday night. We all have a set of restaurants that we prefer, based on our policy/strategy book $Q(s, a)$. If we stick to our normal preference, there is a strong probability that we'll pick a good restaurant. However, sometimes, we occasionally like to try new restaurants to see if they are better. Thus, RL agents will face the same problem. In order to maximize future reward, they need to balance the amount of time that they follow their current policy (this is called being "greedy"), and the time they spend exploring new possibilities that might be better. We adopt a popular approach is called ϵ greedy approach. Under this approach, the policy tells the agent to try a random action some percentage of the time, as defined by the variable ϵ, which is a real number between 0 and 1. The strategy will help the RL agent to occasionally try something new and see if we can achieve ultimate strategy.

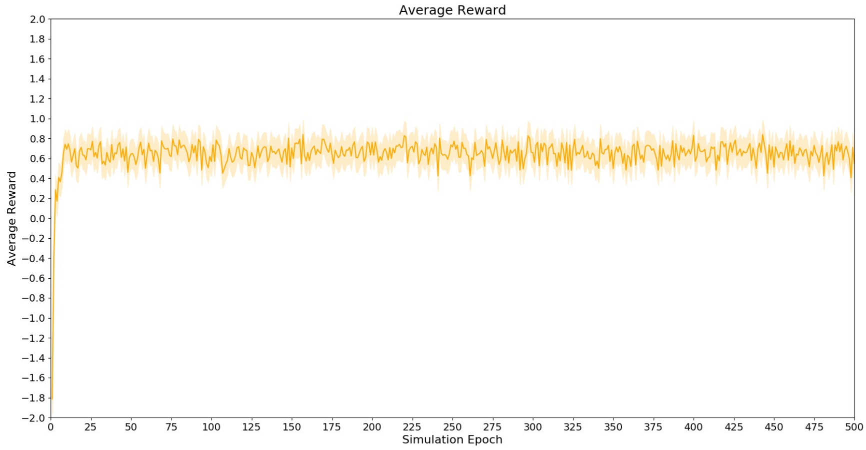
- **Evaluation**

  We will show 3 different results in this part.
  (Note that the #turns now will plus 1 after each action, which means that the #turns calculated below needed to be divided by 2)

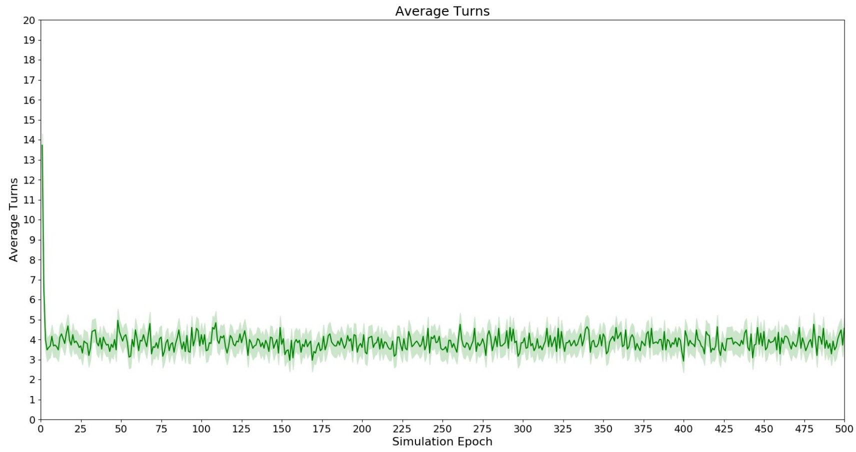
  - **Reward Calculation**
    - Failed: Max Turn * 2.5 / 100
    - Success: Max Turn * 10 / 100
    - Penalty: -30 / 100 (e.g., request for existed slot)
    - Each Turn: -20 / 100
  - Current **Online** RL-Model
    - Learning Curve
      - **Success Rate**



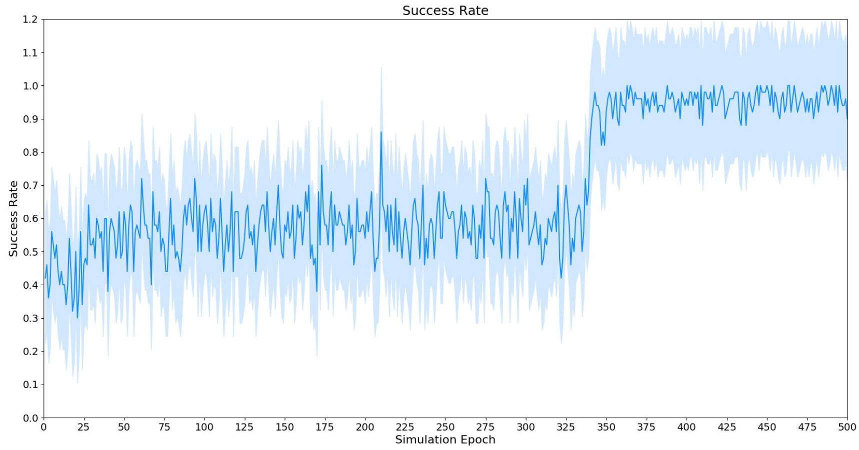
- **Average Reward**



- **Average Turns**

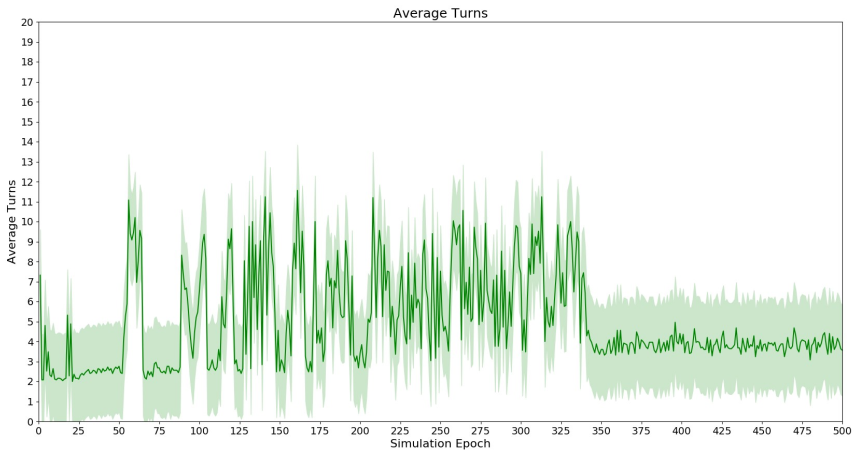


- Latest **Semantic-Level** RL-Model (trained by semantic frame)
  - Learning Curve
    - **Success Rate**

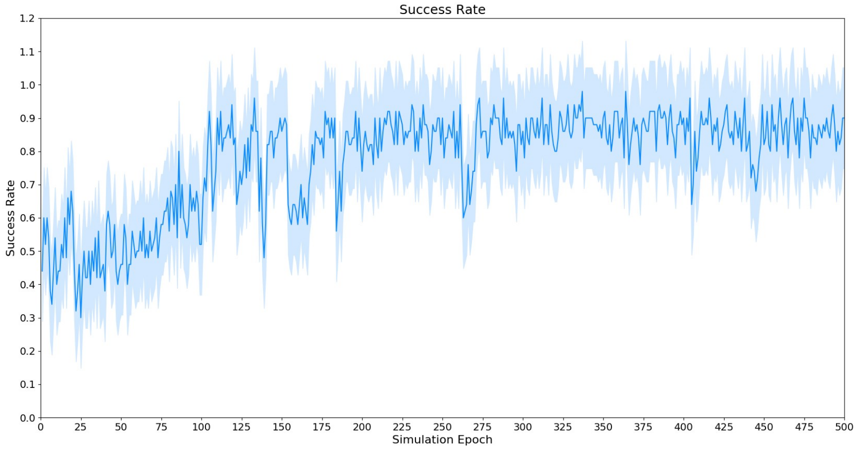


- **Average Reward**

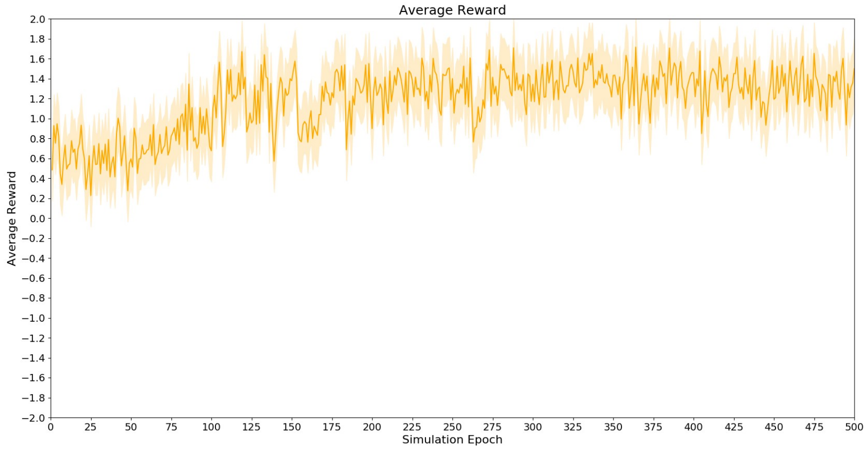
- **Average Turns**



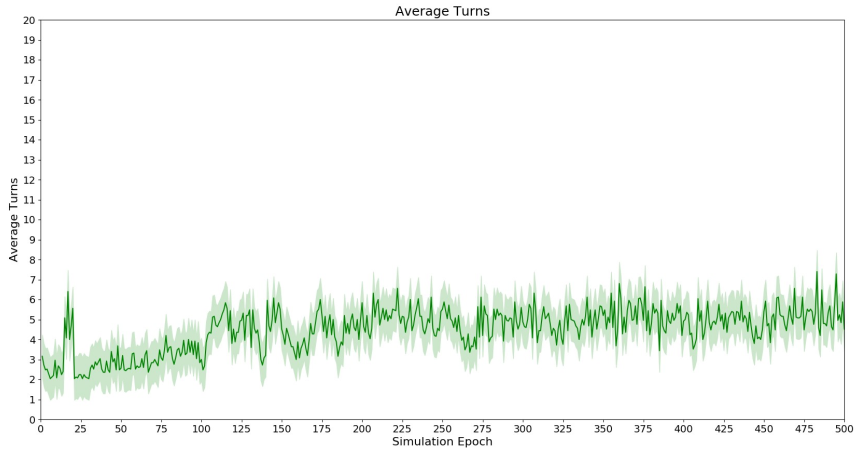
- Latest **NLU-Level** RL-Model (the semantic frame input contains errors from NLU)
  - Learning Curve
    - **Success Rate**



- **Average Reward**

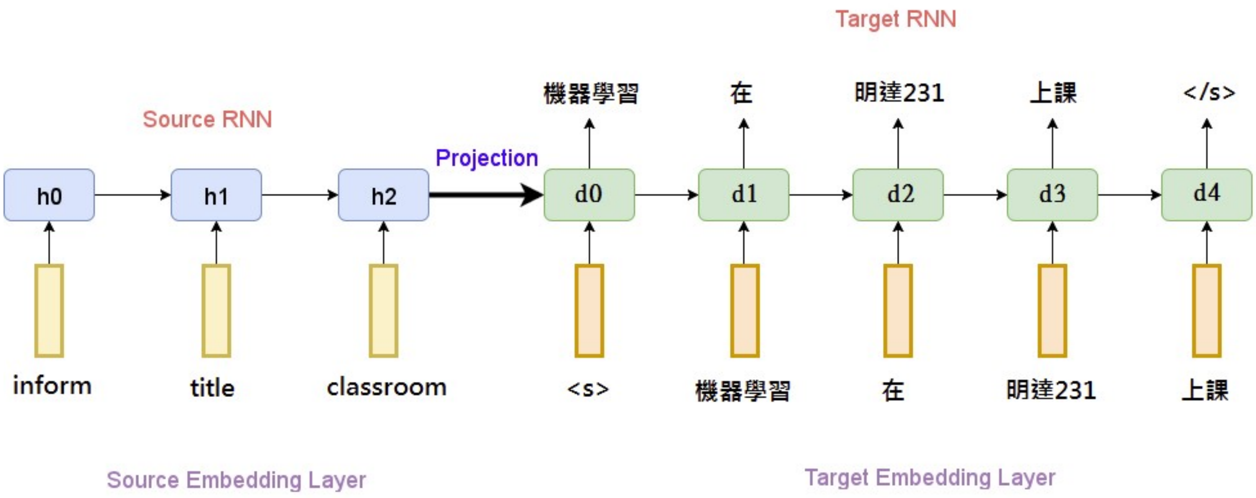


- **Average Turns**



- ## RNN-based NLG (Natural Language Generation)

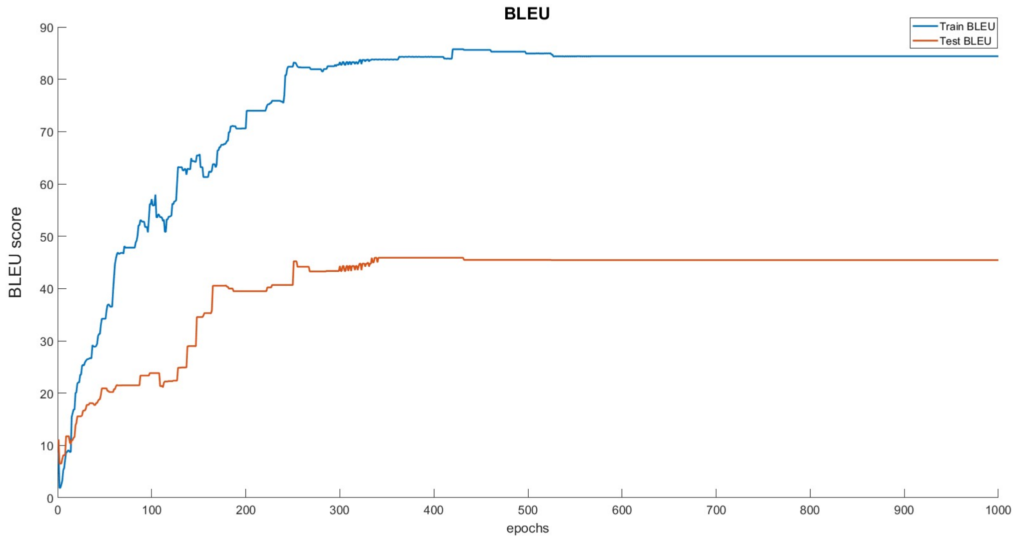
We use pytorch to train a Seq2Seq (https://github.com/MaximumEntropy/Seq2Seq-PyTorch) model transforming semantic frame into natural language. The seq2seq model has two part, encoder and decoder, encoder will encode semantic frame and project it to decoder part, while the decoder part will learn the natural language sentences. Since the order of the slots should not matter, we shuffle the slots of each training instance in each epoch. We build the training set and testing set by different people, so the bleu score on training set is about 85 and testing set is about 45.



- Model
  - Seq2Seq
    - (src_embedding): Embedding(25, 16, padding_idx=1)
    - (trg_embedding): Embedding(123, 80, padding_idx=1)
    - (encoder): LSTM(16, 128, batch_first=True)
    - (decoder): LSTM(80, 128, batch_first=True)
    - (encoder2decoder): Linear (128 -> 128)
    - (decoder2vocab): Linear (128 -> 123)
- Human handcraft template examples
  - Training Data (#sentences=52)
    - request_template
      → Template('{{ask}}課程名稱是什麼')
      → Template('{{ask}}是哪位老師的課呢')
      → ...
    - inform_template
      → Template('在{{when}}上課，要準時喔')
      → Template('上課地點是{{classroom}}，不要走錯囉')
      → Template('{{designated_for}}的{{required_elective}} {{title}}課 授課老師是{{instructor}}在{{classroom}}上課 ')
      → ...
    - end_template
      → Template('感恩')
      → ...
  - Testing Data (#sentences=22)
    - request_template
      → Template('{{ask}}是哪一門課?'),
      → Template('{{ask}}哪位教授開的?'),
      → ...

- inform_template
  → Template('是在{{when}}上課。'),
  → Template('教室在{{classroom}}。'),
  → ...
- end_template
  → Template('感謝你'),
  → ...

- Some results
  - {'diaact':'inform', 'inform_slots':{'title':'自然語言處理', 'classroom':'資105', 'when':'星期四'}}
    → 星期四的自然語言處理課的上課地點是資105
  - {'diaact':'inform', 'inform_slots':{'title':'智慧對話機器人', 'instructor':'陳縕儂'}}
    → 陳縕儂老師開的智慧對話機器人課
  - {'diaact':'inform', 'inform_slots':{'title':'離散數學', 'designated_for':'資訊系','classroom':'資102'}}
    → 資訊系開的離散數學課上課地點是資102
  - {'diaact':'inform', 'inform_slots':{'serial_no':'95046', 'title':'機器學習','instructor':'李宏毅'}}
    → 流水號95046李宏毅老師開的機器學習課
  - {'diaact':'request_title', 'request_slots':{}}
    → 請告訴我課程名稱是什麼
  - {'diaact':'request_instructor', 'request_slots':{}}
    → 請告訴我老師是哪位
  - {'diaact':'thanks', 'inform_slots':{}}
    → 感恩
- Bad result
  - {'diaact':'inform', 'inform_slots':{'title':'智慧對話機器人', 'required_elective':'選修'}}
    → 智慧對話機器人課是在classroom上課
- BLEU score on Training and Testing
  (We compute BLEU based on character ngrams, since a slight inconsistency in word segmentation could cause large loss of word-based BLEU score.)



- # Performance for Simulated Dialogues

  - Example-1

```
================Episode  28 Start================
User:     星期一加簽方式是2的課有哪些
Agent:    請問是哪位老師開的?
User:     簡潔
Agent:    請從以下選擇一個：<br><a href='#' class='selection'>德文三下</a><br><a href='#' class='selection'>德文二下</a><br>
User:     德文二下喔
Agent:    流水號<a href='#' class='serial_no'>28131</a>。課名是德文二下。授課教師是簡潔。
Simulation Episode  28: Success (Reward: +1.60000, #Turns:  6)
================Episode  28 Over!================
```

  - Example-2

```
================Episode  19 Start================
User:     是必修還選修
Agent:    請問課程名稱是?
User:     叫做政治人類學當代議題
Agent:    流水號<a href='#' class='serial_no'>63840</a>。課名是政治人類學當代議題。授課教師是林開世。必選修：選修。
Simulation Episode  19: Success (Reward: +1.80000, #Turns:  4)
================Episode  19 Over!================
```

  - Example-3

```
================Episode  38 Start================
User:     有哪些課
Agent:    請問是誰開的?
User:     李存智老師
Agent:    請從以下選擇一個：<br><a href='#' class='selection'>國文下</a><br><a href='#' class='selection'>聲韻學甲下</a><br>
User:     聲韻學甲下喔
Agent:    流水號<a href='#' class='serial_no'>52172</a>。課名是聲韻學甲下。授課教師是李存智。
Simulation Episode  38: Success (Reward: +1.60000, #Turns:  6)
================Episode  38 Over!================
```

  - Example-4

```
================Episode   9 Start================
User:     幾何學二的教室在哪
Agent:    流水號<a href='#' class='serial_no'>52415</a>。課名是幾何學二。授課教師是王金龍。在天數102上課。
Simulation Episode   9: Success (Reward: +2.00000, #Turns:  2)
================Episode   9 Over!================
```

  - Example-5

```
================Episode   3 Start================
User:     動物解剖生理學下是必修嗎
Agent:    請從以下選擇一個：<br><a href='#' class='selection'>星期五</a><br><a href='#' class='selection'>星期四</a><br>
User:     星期五的課
Agent:    流水號<a href='#' class='serial_no'>43560</a>。課名是動物解剖生理學下。授課教師是邱智賢。必選修：必帶。
Simulation Episode   3: Success (Reward: +1.80000, #Turns:  4)
================Episode   3 Over!================
```

  - Example-6

```
================Episode  48 Start================
User:    日文二下的教室在哪
Agent:   請問是哪個時間上課的?
User:    星期四上課
Agent:   請從以下選擇一個：<br><a href='#' class='selection'>星野風紗子</a><br><a href='#' class='selection'>李欣倫</a><br><a href='#' class='selection'>江佩璇</a><br><a
href='#' class='selection'>洪淑娟</a><br><a href='#' class='selection'>黃意婷</a><br>
User:    黃意婷的課
Agent:   流水號<a href='#' class='serial_no'>62377</a>。課名是日文二下。授課教師是黃意婷。在共310上課。
Simulation Episode  48: Success (Reward: +1.60000, #Turns:  6)
================Episode  48 Over!================
```

- ## References

  - Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. “Sequence to sequence learning with neural networks. (http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf)” Advances in neural information processing systems. 2014.

  - Bordes, Antoine, and Jason Weston. “Learning end-to-end goal-oriented dialog. (https://arxiv.org/pdf/1605.07683.pdf)” Proceedings of The 5th International Conference on Learning Representations. 2017.
    (last updated: 2017/4/17)

  - Serban, Iulian Vlad, et al. “Generative Deep Neural Networks for Dialogue: A Short Review. (https://arxiv.org/pdf/1611.06216.pdf)” arXiv preprint arXiv:1611.06216 (2016).
    (last updated: 2017/4/17)

  - Shah, Pararth, Dilek Hakkani-Tür, and Larry Heck. “Interactive reinforcement learning for task-oriented dialogue management. (https://static.googleusercontent.com/media/research.google.com/zh-TW//pubs/archive/45734.pdf)” NIPS 2016 Deep Learning for Action and Interaction Workshop. 2016.
    (last updated: 2017/4/17)

  - Li, Xiujun, et al. “A User Simulator for Task-Completion Dialogues. (https://arxiv.org/abs/1612.05688)” arXiv preprint arXiv:1612.05688 (2016).
    (github-repo here (https://github.com/MiuLab/TC-Bot))
    (last updated: 2017/4/23)

  - Li, Xuijun, et al. “End-to-end task-completion neural dialogue systems. (https://arxiv.org/abs/1703.01008)” arXiv preprint arXiv:1703.01008 (2017).
    (github-repo here (https://github.com/MiuLab/KB-InfoBot))
    (last updated: 2017/4/23)

  - Yun-Nung Chen, et al. "End-to-End Memory Networks with Knowledge Carryover for Multi-Turn Spoken Language Understanding (https://www.microsoft.com/en-us/research/wp-content/uploads/2016/06/IS16_ContextualSLU.pdf)
    (last updated: 2017/5/6)