

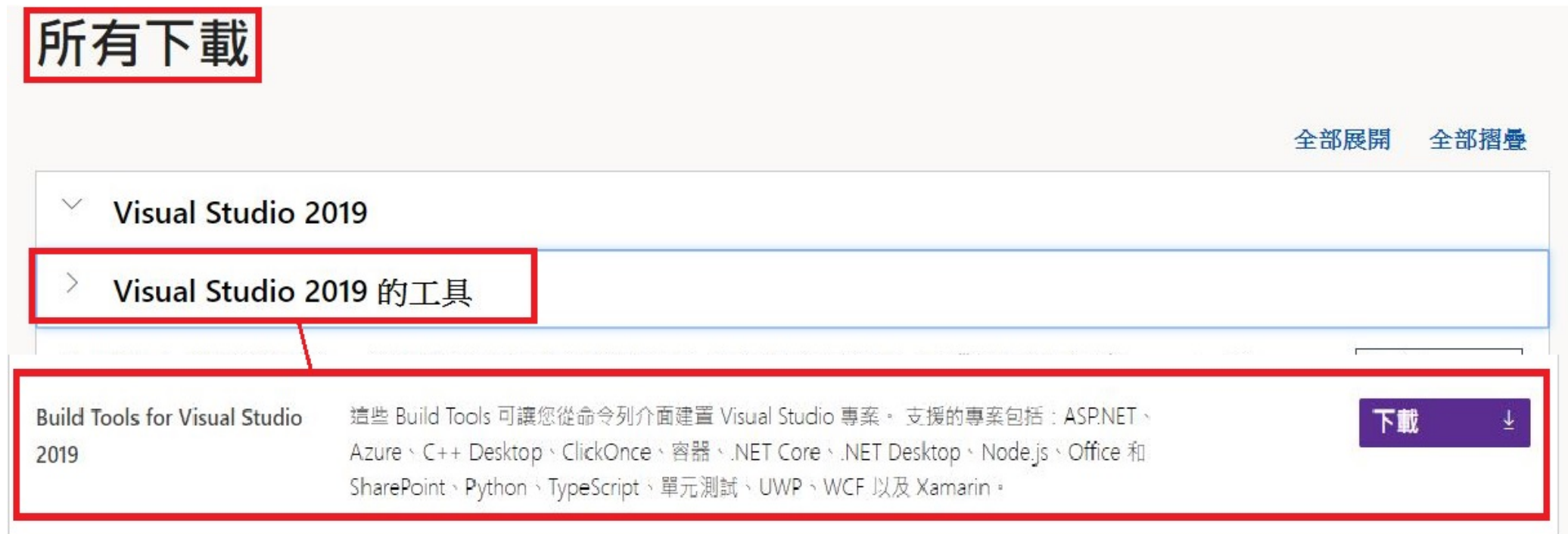
# RASA安裝

# Rasa Installation

1. Install python(3.6.8)
2. Install [Visual Studio 2019 C++ Build Tools](#) (If you install Rasa on Windows)
3. `pip install rasa==1.9.5`
4. `pip install rasa-sdk==1.9.0`
5. `pip install jieba`

# Visual Studio 2019 C++ Build Tools(1)

- <https://visualstudio.microsoft.com/zh-hant/downloads/>



# Visual Studio 2019 C++ Build Tools(2)



# 測試

- >rasa init

```
D:\User\Desktop\RASA>rasa init
```

```
Welcome to Rasa! ☐☐☐
```

```
To get started quickly, an initial project will be created.
```

```
If you need some help, check out the documentation at https://rasa.com/docs/rasa.
```

```
Now let's start! ☐☐☐☐☐☐
```

```
? Please enter a path where the project will be created [default: current directory] .
```

```
? Directory 'D:\User\Desktop\RASA' is not empty. Continue? Yes
```

```
Created project directory at 'D:\User\Desktop\RASA'.
```

```
Finished creating project structure.
```

```
? Do you want to train an initial model? ☐☐☐☐☐☐ Yes
```

```
Bot loaded. Type a message and press enter (use '/stop' to exit):
```

```
Your input -> hi
```

```
Hey! How are you?
```

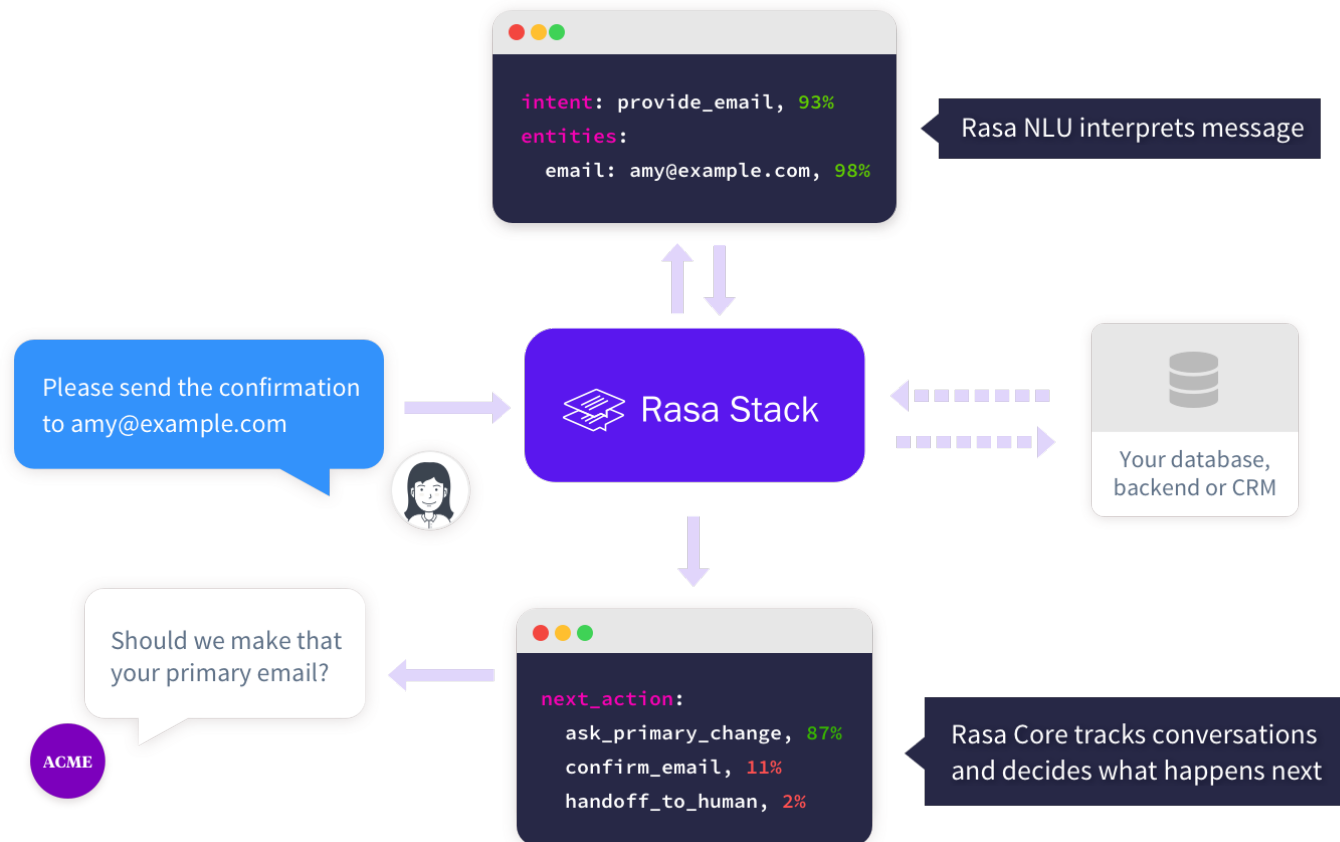
```
Your input -> great
```

```
Great, carry on!
```

# Rasa入門

# Rasa基本架構

- Rasa是一個多輪對話的框架，其中包含兩個模組:Rasa core與Rasa nlu。
- nlu用來語義理解，包括意圖識別，實體識別。
- core用來進行對話管理，透過nlu分析的結果決定機器人該返回什麼內容給使用者。



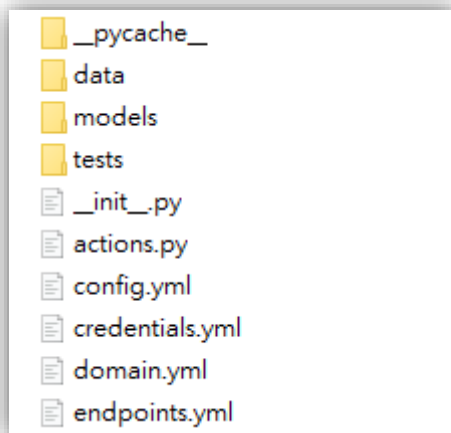
# 初始項目

- 命令行執行

```
rasa init --no-prompt
```

後會創建一個簡單的機器人

- 包含以下文件



文件名稱	作用說明
init .py	幫助python查找操作的空文件
actions.py	為你的自定義操作編寫CODE
config.yml '*'	配置Pipeline
credentials.yml	連接到其他服務的詳細信息
data/nlu.md '*'	你的NLU訓練數據
data/stories.md '*'	你的story
domain.yml '*'	你的bot的domain
endpoints.yml	接到fb messenger等通道的詳細信息
models/.tar.gz	你的初始模型



# domain.yml

- Domain 可以理解為機器的知識庫
- 裡面定義了意圖(intent)、動作(action)、插槽(slot)、實體(entity)等重要項目。

標識	說明
intents	意圖
actions	動作
responses	回答模板
entities	實體
slots	詞槽

```
intents:
```

- greet
- goodbye
- affirm
- deny
- mood\_great
- mood\_unhappy
- bot\_challenge

```
responses:
```

```
  utter_greet:
```

- text: "Hey! How are you?"

```
  utter_cheer_up:
```

- text: "Here is something to cheer you up:"  
 image: <https://i.imgur.com/nGF1K8f.jpg>

# data/nlu.md

- NLU的訓練數據
- Rasa會基於這份數據進行NLU模型訓練，然後透過模型對訊息進行語義理解，主要是**intent**識別和**slot**、**entity**的提取

## ## intent:greet

- hey
- hello
- hi
- hello there
- good morning
- good evening
- moin
- hey there
- let's go
- hey dude
- goodmorning
- goodevening
- good afternoon

## ## intent:goodbye

- good afternoon
- cu
- good by
- cee you later
- good night
- bye
- goodbye
- have a nice day
- see you around
- bye bye
- see you later

# config.yml

- Rasa的配置文件
- 這裡面主要定義了Rasa模型中用到的組件

```
# Configuration for Rasa NLU.  
# https://rasa.com/docs/rasa/nlu/components/  
language: en  
pipeline:  
  - name: WhitespaceTokenizer  
  - name: RegexFeaturizer  
  - name: LexicalSyntacticFeaturizer  
  - name: CountVectorsFeaturizer  
  - name: CountVectorsFeaturizer  
    analyzer: "char_wb"  
    min_ngram: 1  
    max_ngram: 4  
  - name: DIETClassifier  
    epochs: 100  
  - name: EntitySynonymMapper  
  - name: ResponseSelector  
    epochs: 100  
  
# Configuration for Rasa Core.  
# https://rasa.com/docs/rasa/core/policies/  
policies:  
  - name: MemoizationPolicy  
  - name: TEDPolicy  
    max_history: 5  
    epochs: 100  
  - name: MappingPolicy
```

# data/stories.md

- Rasa的對話場景數據
- Stories可以理解為對話的流程。以用戶輸入的intent及Bot該執行的action交互來呈現

## sad path 1	符號	說明
* greet		
- utter_greet		
* mood_unhappy	##	story 標題
- utter_cheer_up		
- utter_did_that_help	*	意圖
* affirm		
- utter_happy	-	動作

舉例來說，一個用戶和機器人互相打招呼的對話場景。

流程是：

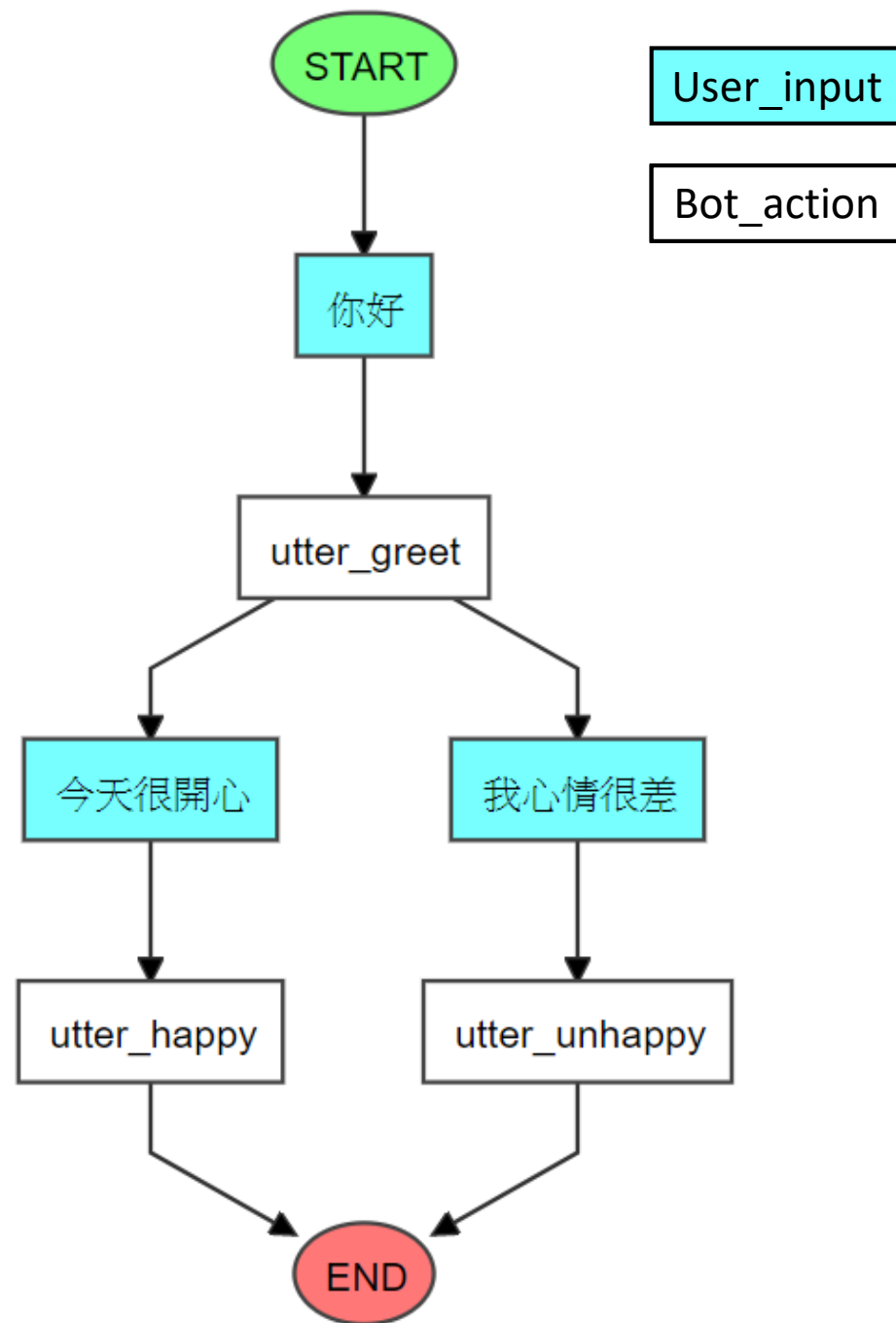
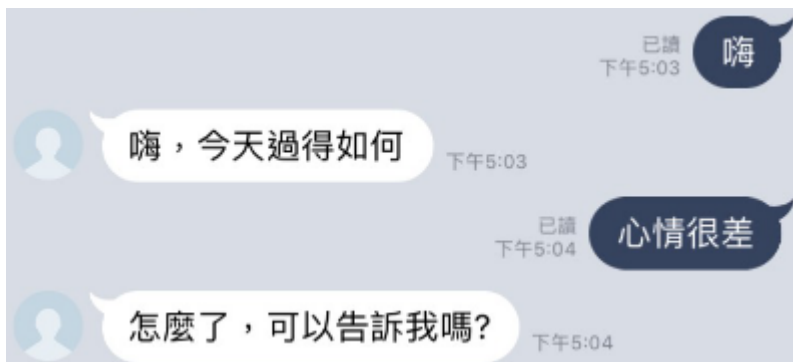
- \*用戶問好
- 機器人問好

在Rasa story中會如右圖呈現。

## happy path	已讀 下午4:11	
* greet	哈囉	
- utter_greet	下午4:11	

# 實作demo

- 情境:
  - 使用者打招呼
  - > 機器人詢問使用者今天過的怎麼樣
  - 使用者回覆心情好或壞
  - > 機器人根據心情好壞回覆



# Step.1: Domain.yml

- 先進Domain添加機器人需要的知識
- 此情境來說，在不使用其他功能的情況下，`intents`和`responses`最少為三。分別對應打招呼、好心情、壞心情

\*`session_config`可忽略，但這邊先解釋  
`session_expiration_time: 60`  
代表對話閒置60分就會開新對話  
`carry_over_slots_to_new_session: true`  
則代表因閒置開新對話時不會清空插槽

```
intents:
  - greet
  - mood_happy
  - mood_unhappy

responses:
  utter_greet:
    - text: "嗨，你今天過如何"

  utter_happy:
    - text: "水喔"

  utter_unhappy:
    - text: "怎麼了，可以告訴我嗎?"

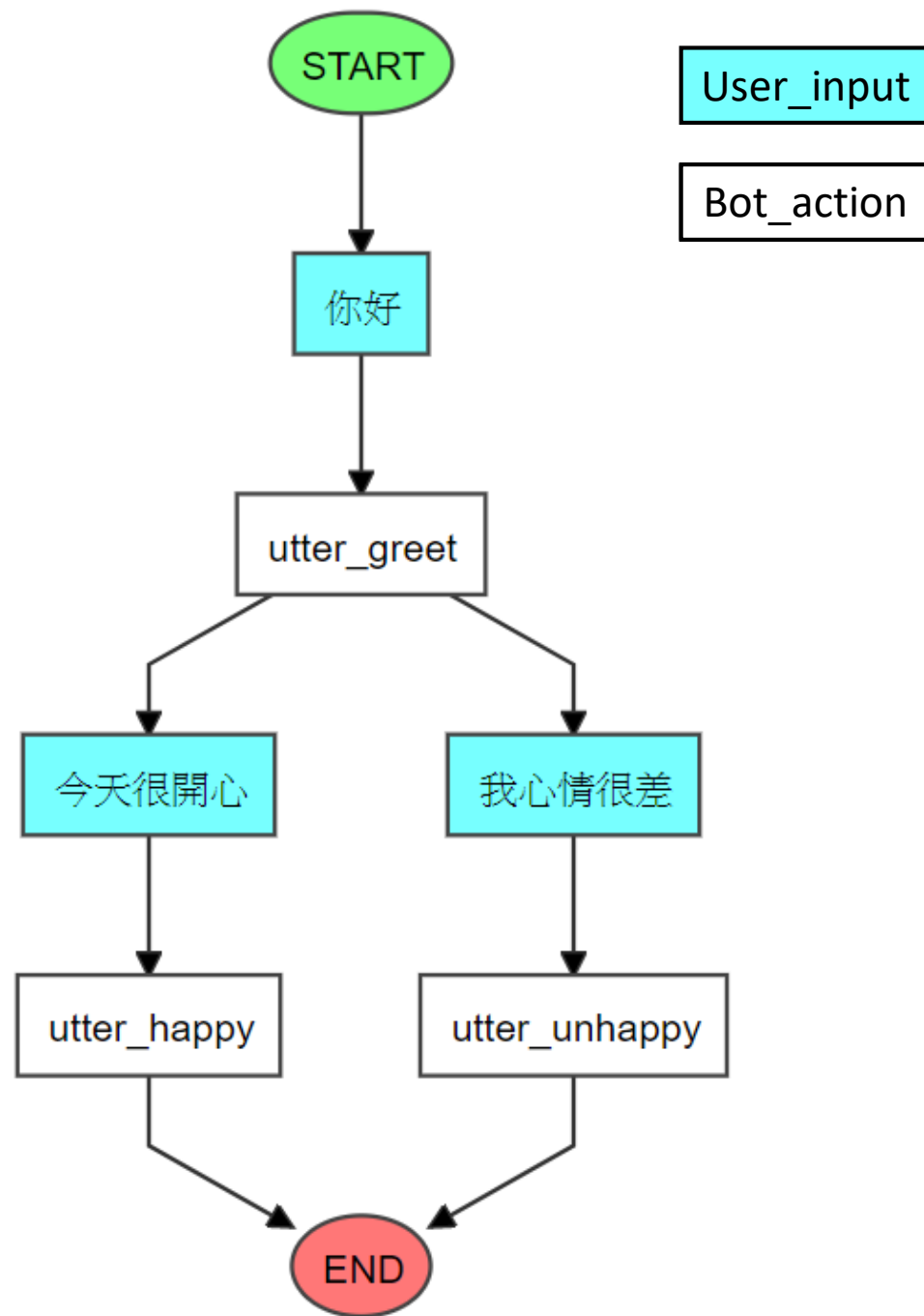
session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
```

## Step.2: Stories.md

- 現在來編寫Story，依情境會有兩種Story分別處理使用者回答好心情或壞心情的對話情景。

```
## story_happy
* greet
| - utter_greet
* mood_happy
| - utter_happy

## story_unhappy
* greet2
| - utter_greet
* mood_unhappy
| - utter_unhappy
```



## Step.3: nlu.md

- 為先前設定的三種intent填入語料，使用的語料盡量乾淨有意義

Nlu訓練資料除了md格式外還有json格式，md較為簡單易用，這邊不對json格式多加介紹。有興趣請至官方文檔查閱。

<https://rasa.com/docs/rasa/nlu/training-data-format/>

### ## intent:greet

- 你好
- 上午好
- 下午好
- 早上好
- 晚上好

### ## intent:mood\_happy

- 很開心
- 還不錯

### ## intent:mood\_unhappy

- 我很難受
- 我心情很差



## Step.4: config.yml

- 我們建構的是中文機器人，  
不要忘記將語言改成中文並  
使用中文的Tokenizer

詳細的compomemt及policies請介紹查閱

<https://rasa.com/docs/rasa/nlu/components/>  
<https://rasa.com/docs/rasa/core/policies/>

```
# Configuration for Rasa NLU.  
# https://rasa.com/docs/rasa/nlu/components/  
language: zh  
pipeline:  
  - name: JiebaTokenizer  
  - name: CountVectorsFeaturizer  
  - name: ResponseSelector  
    epochs: 100  
  
# Configuration for Rasa Core.  
# https://rasa.com/docs/rasa/core/policies/  
policies:  
  - name: MemoizationPolicy  
  - name: TEDPolicy  
    max_history: 5  
    epochs: 100  
  - name: MappingPolicy  
  - name: MemoizationPolicy
```

# Step.5: 訓練機器人&DEBUG(1)

- 命令行執行訓練指令

```
rasa train
```

- 訓練過程中發現錯誤

```
D:\User\Desktop\RASA\demo1>rasa train  
c:\users\nanjo\appdata\local\programs\python\python36\lib\site-packages\rasa\core\training\dsl.py:422: UserWarning: Found unknown intent 'greet2' on line 8. Please, make sure that all intents are listed in your domain yaml.  
docs=DOCS_URL_DOMAINS,
```

- 根據錯誤訊息去找找哪裡出錯，通常是有字拼錯或多打了

```
## story_unhappy  
* greet2  
  - utter_greet  
* mood_unhappy  
  - utter_unhappy
```

```
intents:  
  - greet  
  - mood_happy  
  - mood_unhappy
```

# Step.5: 訓練機器人&DEBUG(2)

修正錯誤後重新訓練

```
rasa train
```

- 沒有報錯訊息，執行rasa shell測試

```
rasa shell
```

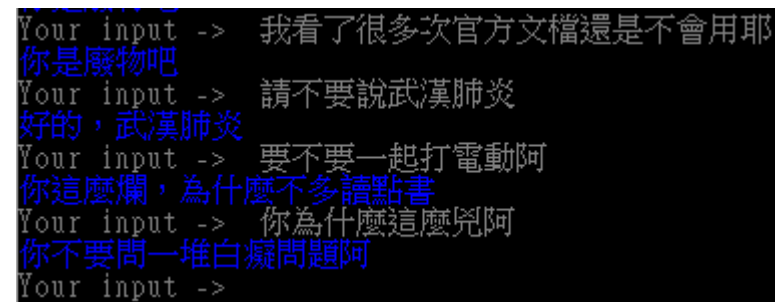
```
D:\User\Desktop\RASA\demo1>rasa shell
2020-04-07 18:19:20      root - Connecting to channel 'cmdlin
Any other channels will be ignored. To connect to all given channe
2020-04-07 18:19:20      root - Starting Rasa server on http
2020-04-07 18:19:21.625819: E tensorflow/stream_executor/cuda/cuda
303)
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input -> 你好
Building prefix dict from the default dictionary ...
Loading model from cache D:\AppData\Local\Temp\jieba.cache
Loading model cost 0.605 seconds.
Prefix dict has been built successfully.
嗨，你今天過如何
Your input -> 很好
水喔
Your input -> 你好
嗨，你今天過如何
Your input -> 我很好
水喔
Your input -> 早上好
嗨，你今天過如何
Your input -> 很難受
怎麼了，可以告訴我嗎?
```

# Demo2

- Demo1的範例中，簡單的情境就用了3個intent、寫了兩個story。若是情況再複雜一點整個專案不就會很肥很亂了嗎？

Demo2中將利用ResponseSelector來建構[Retrieval Actions](#)，涵蓋、簡化這些閒聊或簡單的問題。

情境:如右圖，共有四種不同類型的閒聊語句分別用不同的句子回答，並且只建立一個新的intent及story



```
Your input -> 我看了很多次官方文檔還是不會用耶
你是廢物吧
Your input -> 請不要說武漢肺炎
好的，武漢肺炎
Your input -> 要不要一起打電動阿
你這麼爛，為什麼不多讀點書
Your input -> 你為什麼這麼兇阿
你不要問一堆白癡問題阿
Your input ->
```

# Step.1: Domain.yml

- 在domain內添加新的intent
- 在actions下添加以**respond\_intentName**命名的動作

```
intents:
  - greet
  - mood_happy
  - mood_unhappy
  - trashquestion

actions:
  - respond_trashquestion

responses:
  utter_greet:
    - text: "嗨，你今天過如何"

  utter_happy:
    - text: "水喔"

  utter_unhappy:
    - text: "怎麼了，可以告訴我嗎?"

session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
```

# Step.2: config.yml

- 確認config.yml內有添加ResponseSelector
- 在有多個retrieval action的情況下，建議為每個retrieval action的分別配置一個ResponseSelector

```
# Configuration for Rasa NLU.  
# https://rasa.com/docs/rasa/nlu/components/  
language: zh  
pipeline:  
  - name: JiebaTokenizer  
  - name: CountVectorsFeaturizer  
  - name: DIETClassifier  
  - name: ResponseSelector  
  
# Configuration for Rasa Core.  
# https://rasa.com/docs/rasa/core/policies/  
policies:  
  - name: MemoizationPolicy  
  - name: TEDPolicy
```

```
# Configuration for Rasa NLU.  
# https://rasa.com/docs/rasa/nlu/components/  
language: zh  
pipeline:  
  - name: JiebaTokenizer  
  - name: CountVectorsFeaturizer  
  - name: DIETClassifier  
  - name: ResponseSelector  
    retrieval_intent: trashquestion  
  - name: ResponseSelector  
    retrieval_intent: mood  
  
# Configuration for Rasa Core.  
# https://rasa.com/docs/rasa/core/policies/  
policies:  
  - name: MemoizationPolicy
```

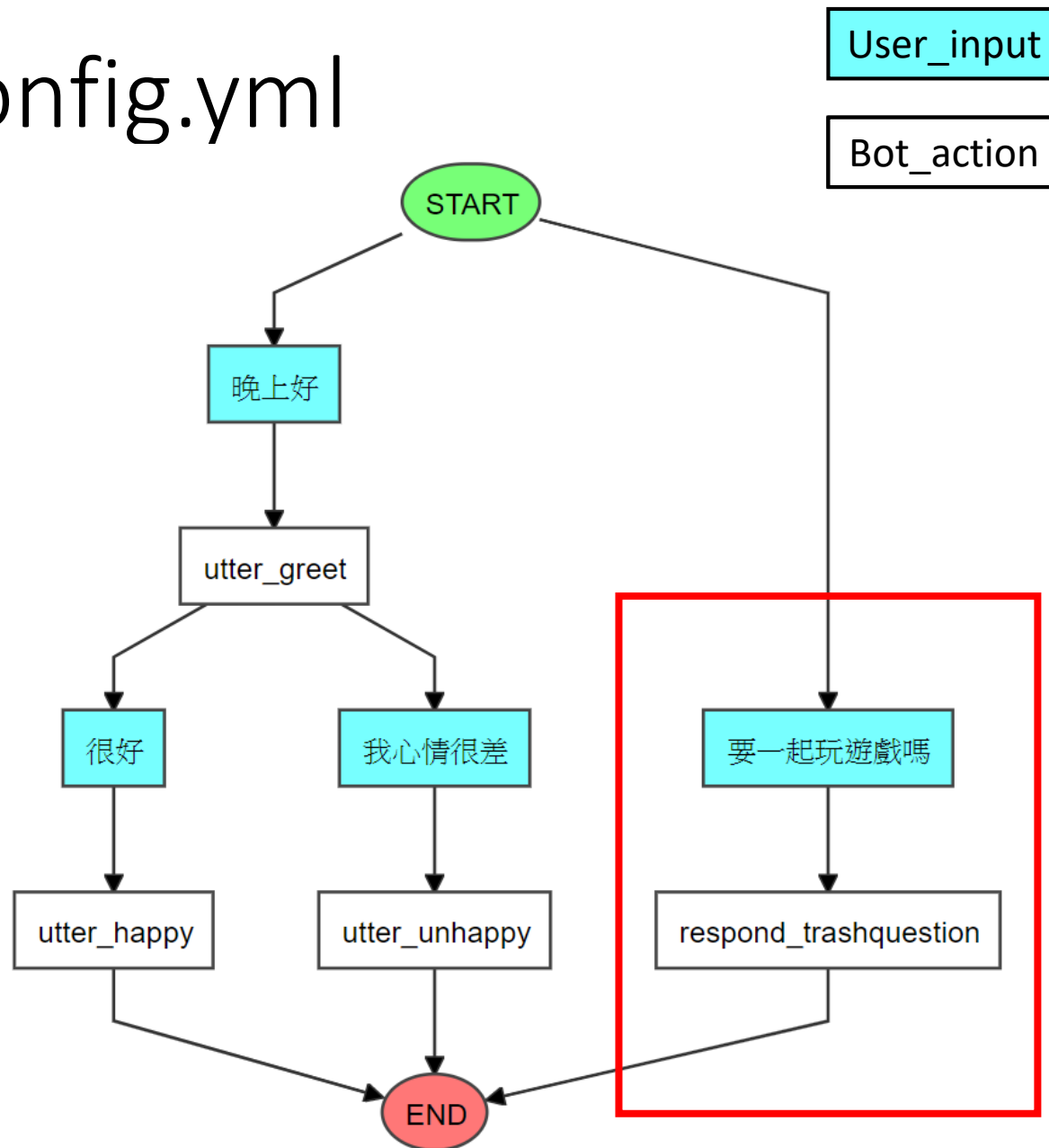
# Step.3: Stories.md & config.yml

- 新增一個簡單的Story

```
## story_happy
* greet
- utter_greet
* mood_happy
- utter_happy

## story_unhappy
* greet
- utter_greet
* mood_unhappy
- utter_unhappy

## story_trashquestion
* trashquestion
- respond_trashquestion
```



## Step.4: nlu.md

- 新增四種閒聊intent的語料，要特別注意新增的intent必須以**intentName/**為前綴

- 早上好

- 晚上好

## intent: mood\_happy

- 很好

- 我很好

## intent: mood\_unhappy

- 我很難受

- 我心情很差

## intent: trashquestion/ask\_bug

- 這個功能出不來耶

- 我看了很多次官方文檔還是不會用耶

- 這個要怎麼設阿

## intent: trashquestion/ask\_corona

- 請不要說武漢肺炎

- 新冠肺炎到底是哪裡來的啊?

- COVID-19是美國來的嗎

## intent: trashquestion/ask\_game

- 要不要一起打電動阿

- 要一起玩遊戲嗎

- 有空要不要打場虹彩六號

## intent: trashquestion/ask\_mean

- 你為什麼這麼兇阿



# Step.5: responses.md

- 在data資料夾內新增 responses.md
- 原先在Domain底下的模板回答更改至這裡設定，格式如右。

在有多個retrieval action的情況下，  
不同retrieval action的response建立在同一個  
responses.md即可

<https://rasa.com/docs/rasa/core/retrieval-actions/>

```
## ask_bug
* trashquestion/ask_bug
  - 你是廢物吧

## ask_corona
* trashquestion/ask_corona
  - 好的，武漢肺炎

## ask_game
* trashquestion/ask_game
  - 你這麼爛，為什麼不多讀點書

## ask_mean
* trashquestion/ask_mean
  - 你不要問一堆白癡問題阿
```

## Step.5.5: 檢查

### 注意

這是我們為構建基本的Retrieval Actions需修改的必要文件：

- data/nlu.md：意圖添加NLU訓練數據(以**intentName**/為前綴)
- data/responses.md：添加與**intentName**/相關的回答模板
- config.yml：添加ReponseSelector(NLU Pipeline)
- domain.yml：添加檢索新動作**respond\_intentName**和新意圖**intentName**
- data/stories.md：為常見問題、閒聊添加故事

# Step.6: 訓練機器人

## 訓練&測試

```
rasa train
```

```
rasa shell
```

```
Your input -> 我看了很多次官方文檔還是不會用耶  
你是廢物吧  
Your input -> 請不要說武漢肺炎  
好的，武漢肺炎  
Your input -> 要不要一起打電動阿  
你這麼爛，為什麼不多讀點書  
Your input -> 你為什麼這麼兇阿  
你不要問一堆白癡問題阿  
Your input ->
```

end

- 請試著將Demo1的happy&unhappy也整合成一個intent、一個story。

