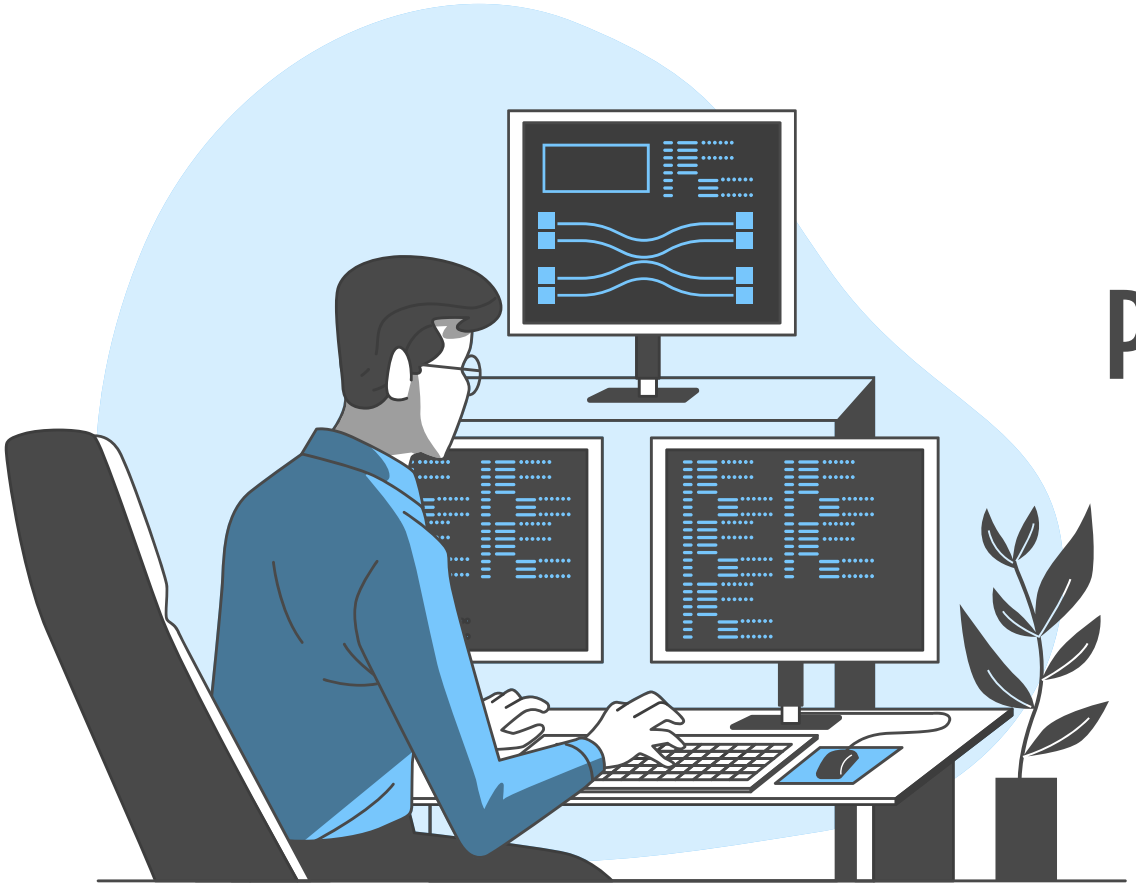

Pre-Bakeoff

2023/05/11

品而

OUTLINE

1. LangChain Crash Course
2. Bakeoff & 期末專案要幹嘛
3. 範例Tools介紹

1. LangChain Crash Course

- 影片 part 1
- 影片搭配的 colab

1. LangChain Crash Course

- Exercise colab
  (using HuggingFaceHub model)

2. Bakeoff & 期末專案

- 2~3人一組
- 設計 n 個「能讓LLM使用」的 LangChain Tool
- Based on 一堆PTT語料

2. Bakeoff & 期末專案

- 今天要完成的:
  - 分組＆初步想法 ⇒ 登記到 試算表
  - 辦一個新的 OpenAI 帳號(才有5鎂的額度)
  - 回家看影片 part 2~4

- 下週要做的:
  - 利用語料實作Tools

3. 範例 Tools

- [lopeGPT repo]

```
from lope_tools import (
    SenseTagTool,
    QuerySenseFromDefinitionTool,
    QuerySenseFromLemmaTool,
    QuerySenseFromExamplesTool,
    QueryAsbcSenseFrequencyTool,
    QueryRelationsFromSenseIdTool
    )
```

SenseTagTool(text):

```
name = "SenseTagger"
description = "輸入文章可以斷詞和標註詞性SenseID和詞義，輸出為JSON格式。"
```

```
# Example
SenseTagTool().run("他打算打我一頓。")
```

```
('([詞] 他 || [詞性] Nh || [詞意] [05238501] 代指自己和對方以外的第三人。 (0.9708))\n'
 '([詞] 打算 || [詞性] VF || [詞意] [06603401] 初步決定要做後述事件。 (1.0000))\n'
 '([詞] 打 || [詞性] VC || [詞意] [05229102] 用手或手持物打後述對象，使其感到疼痛或受到傷害。
 '([詞] 我 || [詞性] Nh || [詞意] [05238701] 代指說話者。 (0.9666))\n'
 '([詞] 一 || [詞性] Neu || [詞意] [06727301] 事物的全體。 (0.5460))\n'
 '([詞] 頓 || [詞性] Nf || [詞意] [06001902] 計算負面經驗事件次數的單位。 (1.0000))\n'
 '([詞] 。 || [詞性] PERIODCATEGORY || [詞意] )')
```

QuerySenseFromLemmaTool(text)

```
name = "QuerySensesFromLemma"
description = "搜尋CWN所有詞義，找到lemma符合目標詞的數個詞義，可以使用
Regular Expression。輸出為JSON格式。"
```

```
pprint(json.loads(QuerySenseFromLemmaTool().run("電腦")))
```

```
{'個人電腦': {'all_examples': ['一般電腦用戶最關心的系統，莫過於<個人電腦>上的多媒體產品，坦白說，<個人電腦>的多
                               '當然，網威亦想透過本案獲得新市場，網威目前在ＰＣ級網路作業系統上擁有不可動搖的地位，
                               '面對<個人電腦>走向多工作業時代，名豐資訊於日前領先推出一套能在ＤＯＳ系統下做到一機
              'definition': '提供個人使用的一種資料處理裝置，能自動接受並儲存、處理輸入的資料，然後經由一組預先存放
              'head_word': '個人電腦'},
 '電腦': {'all_examples': ['小<電腦>是三年級時，大家取的，因為我都替他們解決問題，所以大家就叫我小<電腦>。'],
          'definition': '比喻計算或記憶能力很強的人。',
          'head_word': '電腦'}}
```

QuerySenseFromDefinitionTool(text)

```
name = "QuerySensesFromDefinition"
description = "搜尋CWN所有詞義，找到definition有出現目標詞的數個詞義，
可以使用Regular Expression。輸出為JSON格式。"
```

```
list(json.loads(QuerySenseFromDefinitionTool().run("電腦")).items())[:2]
```

```
[('電通所',
  {'all_examples': ['目前電信總局已結合電信研究所及工研院<電通所>，成立VOD推動小組。',
                    '中心已於上週向<電通所>訂購一套單機版的「中華民國期刊論文索引光碟系統」，
                    '由於外界疑慮日後得標廠商，將形成壟斷市場情況，因此，<電通所>在採購規格
   'definition': '工業技術研究院，研究電腦與通訊工業的機構，在西元1990年成立，已於西元2006年
   'head_word': '電通所'}),
 ('命令',
  {'all_examples': ['如果你不確定<命令>語法，可從編輯器叫出<命令>語法參照表進行查詢。',
                    '本程式是以連續式<命令>的方式訓練神經網路上的車子訊號如何行走的\r\n程式
                    '以後只要利用telnet<命令>即可進行遠程連線，<命令>格式如英文書目方法步驟
                    '其<命令>格式為: vi. cshrc仔細查看檔中是否包含下令三行<命令>，若無則必
   'definition': '人給電腦的指令。',
   'head_word': '命令'}),
```

QueryRelationsFromSenseIdTool(sense_id)

```
name = "QueryRelationsFromSenseId"
description = (
    "輸入目標詞的SenseID（8位數字），得到目標詞的relations，取得特定的語意關係（synonym同義詞、antonym反義詞、hypernym上位詞、hyponym下位詞）。如果已經有標記過的文章，則使用文章中目標詞的SenseID，再去獲得該SenseID的relations。輸出為JSON格式。"
)
```

```
pprint(json.loads(QueryRelationsFromSenseIdTool().run("03002201")))
```

```
['synonym', '<CwnSense[04026501](不如，VJ): 沒有達到比較對象的標準。>', 'forward'],
['synonym', '<CwnFacet[0511760201](不及): 沒有達到比較對象的標準。>', 'forward'],
['synonym', '<CwnSense[05025001](還不如，VJ): 沒有達到比較對象的標準。>', 'forward'],
['synonym', '<CwnSense[05117602](不及，VJ,nom): 沒有達到比較對象的標準。>', 'forward'],
['synonym', '<CwnSense[04026501](不如，VJ): 沒有達到比較對象的標準。>', 'reversed'],
['synonym', '<CwnSense[05025001](還不如，VJ): 沒有達到比較對象的標準。>', 'reversed'],
['synonym', '<CwnFacet[0511760201](不及): 沒有達到比較對象的標準。>', 'reversed']]
```

Tools 是給LLM用的

```
tools = [
    SenseTagTool(),
    QuerySenseFromDefinitionTool(),
    QuerySenseFromLemmaTool(),
    QuerySenseFromExamplesTool(),
    QueryRelationsFromSenseIdTool(),
    QueryAsbcSenseFrequencyTool(),
]

llm = ChatOpenAI(
    model_name="gpt-3.5-turbo",
    streaming=True,
    callback_manager=CallbackManager([StreamingStdOutCallbackHandler()]),
    verbose=True,
    temperature=0,
    client=None
)

memory = ConversationBufferMemory(memory_key="chat_history")
agent_chain = initialize_agent(tools, llm, agent=AgentType.CHAT_ZERO_SHOT_REACT_DESCRIPTION, verbose=True, memory=memory)
```

Tools 是給LLM用的

```
agent_chain.run("""
步驟一: 標註原句子「她是我的朋友, 她很聰明也很美麗。」
步驟二: 利用「美麗」的SenseID, 得到詞義關係, 獲得synonym。
步驟三: 使用同義詞改寫原句子。
""")
```

> Entering new AgentExecutor chain...
Question: 如何使用同義詞改寫句子「她是我的朋友，她很聰明也很美麗。」？
Thought: 我們需要先標註原句子，然後找到「美麗」的SenseID，再查詢詞義關係，找到同義詞，最後使用同義詞改寫原句子。
Action:
```
{
  "action": "SenseTagger",
  "action_input": "她是我的朋友，她很聰明也很美麗。"
}
```
Question: 如何使用同義詞改寫句子「她是我的朋友，她很聰明也很美麗。」？
Thought: 我們需要先標註原句子，然後找到「美麗」的SenseID，再查詢詞義關係，找到同義詞，最後使用同義詞改寫原句子。
Action:
```
{
  "action": "SenseTagger",
  "action_input": "她是我的朋友，她很聰明也很美麗。"
}
```

Observation: *[[{"詞": "她", "詞性": "Nh", "SenseID": "05124901", "詞意": "代指自己和對方以外的單數女性第三者。"}, {"詞*
Thought:根據標註的結果，「美麗」的SenseID為「06527901」，我們可以使用QueryRelationsFromSenseId工具來查詢同義詞。
Action:
```
{
  "action": "QueryRelationsFromSenseId",
  "action_input": "06527901"
}
```

（接續）

```
{
  "action": "QueryRelationsFromSenseId",
  "action_input": "06527901"
}
```
根據標註的結果，「美麗」的SenseID為「06527901」，我們可以使用QueryRelationsFromSenseId工具來查詢同義詞。
Action:
```
{
  "action": "QueryRelationsFromSenseId",
  "action_input": "06527901"
}
```

Observation: *[["synonym", "<CwnSense[03032301](漂亮，VH): 形容人或物的外表好看使觀者感到愉悅而讚賞的。>", "reversed"],*
Thought:根據查詢結果，我們可以使用同義詞「漂亮」來改寫原句子。
Action:
```
{
  "action": "SenseTagger",
  "action_input": "她是我的朋友，她很聰明也很漂亮。"
}
```

（接續）

根據查詢結果，我們可以使用同義詞「漂亮」來改寫原句子。
Action:
```
{
  "action": "SenseTagger",
  "action_input": "她是我的朋友，她很聰明也很漂亮。"
}
```

Observation: *[[{"詞": "她", "詞性": "Nh", "SenseID": "05124901", "詞意": "代指自己和對方以外的單數女性第三者。"},*
Thought:使用同義詞「漂亮」改寫原句子後，新句子為「她是我的朋友，她很聰明也很漂亮。」
Final Answer: 她是我的朋友，她很聰明也很漂亮。*使用同義詞「漂亮」改寫原句子後，新句子為「她是我的朋友，她很聰明也很漂亮。」*
Final Answer: 她是我的朋友，她很聰明也很漂亮。

> Finished chain.
'她是我的朋友，她很聰明也很漂亮。'

最終output給Human的答案

偶爾會秀逗

```
agent_chain.run("""
找到「小心」的關係、同義詞, 並paraphrase原句子「我很美麗又很小心」, 輸出最後答案。
""")
```

> Entering new AgentExecutor chain...
Question: 找到「小心」的關係、同義詞，並paraphrase原句子「我很美麗又很小心」，輸出最後答案。

Thought: 我們需要使用QueryRelationsFromSenseId和SenseTagger來找到「小心」的關係和同義詞，然後使用SenseTagger來斷詞和標註詞性，最後進行paraphrase。

Action:
```
{
  "action": "QueryRelationsFromSenseId",
  "action_input": {
    "target": "小心"
  }
}
```

Question: 找到「小心」的關係、同義詞，並paraphrase原句子「我很美麗又很小心」，輸出最後答案。

Thought: 我們需要使用QueryRelationsFromSenseId和SenseTagger來找到「小心」的關係和同義詞，然後使用SenseTagger來斷詞和標註詞性，最後進行paraphrase。

Action:
```
{
  "action": "QueryRelationsFromSenseId",
  "action_input": {
    "target": "小心"
  }
}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-f8f7eb6cea86> in <cell line: 1>()
----> 1 agent_chain.run("""
      2 找到「小心」的關係、同義詞，並paraphrase原句子「我很美麗又很小心」，輸出最後答案。
      3 """)

 6 frames
/usr/local/lib/python3.10/dist-packages/langchain/tools/base.py in run(self, tool_input, verbose, start_color, color, callbacks, **kwargs)
    245                 self._run(*tool_args, run_manager=run_manager, **tool_kwargs)
    246                 if new_arg_supported
--> 247                 else self._run(*tool_args, **tool_kwargs)
    248             )
    249         except (Exception, KeyboardInterrupt) as e:

TypeError: QueryRelationsFromSenseIdTool._run() got an unexpected keyword argument 'target'
```

可以參考的

- **LangChain Doc (Custom Tool)**

  https://python.langchain.com/en/latest/modules/agents/tools/custom_tools.html#tool-dataclass

  Create Custom Tools for Chatbots in LangChain — LangChain #8

- **lopeGPT 的tools**
- **lopeGPT 的測試notebook**

Reminder

- 作業:看完剩下的3部影片、登記組別
- **設計n個tool, 不用做出一個GPT🤖**

- OpenAI的模型是以 input + output 總token數來計算5鎂的額度
- **LangChain 時不時就會更新版本😊😇🤗**

  請常常

```
!pip install -U langchain
```

不然會被開發者衝康 :)