1. 動機
2. 系統架構
3. Discord API
4. 斷詞系統
5. 使用者意圖偵測 Intent Detection
6. 對話處理程序 Dialog Manager
7. 文句產生模組 Natural Language Generation
8. 資料庫 Database
9. 遊戲特價查詢功能Game Deal
10. 遊戲即時新聞查詢GNN News
11. Demo

一、動機

最近這幾年，聊天機器人蔚為風潮，剛好我們組員接觸了一個聊天平台Discord，可以透過聊天室提供的API以及Java來建立我們自己的聊天機器人。

在使用過其他人設計的機器人之後，覺得這些機器人大部分只支援英文對談，或者只有一些簡單的指令式機器人。我們決定來開發一個支援中文對談而且針對電玩領域的機器人，也藉此瞭解更多關於自然語言處理的知識。

這個機器人主要是架構在Discord聊天室上，Discord聊天室聚集了許多遊戲玩家，使用者可以自行成立聊天伺服器，並建立許多文字語音頻道。其中官方也提供了一個方便的API，可以幫助開發者開發一個Bot在Discord上與使用者互動。

透過這個聊天機器人來跟聊天室內所有的使用者互動，例如提供最近的遊戲新聞、特價資訊等。雖然大致上還是個Rule Based的機器人，但還是嘗試幫機器人加入學習能力，針對使用者有興趣的遊戲類型進行推薦。

二、流程架構簡介與範例

Main

Discord API Login

MessageReceiveEvent

鬥陣特攻最近有一則新聞

《鬥陣特攻》2017 年冬境樂園限時開放 新玩法「雪怪大作戰」與超過 50 樣主題物品登場

https://gnn.gamer.com.tw/0/156490.html

最近有鬥陣特攻的新聞嗎?

Discord聊天室

User

Chatbot

最近沒有鬥陣特攻的新聞喔

Chatbot

詢問新聞對應到GameNewsProcess

意圖：詢問新聞

最近 有 鬥陣特攻 的 新聞 嗎 ?

否

是，回傳標題與網址

相關新聞是否存在?

GameNewsProcess

Dialog Manager

Intent Detection

Word Segmentation

三、Discord API

Discord官方提供的API可以讓我們安全的登入自己註冊的Bot。註冊完Bot之後會獲得一串英數大小寫混雜且字串長度約50多字的Token，使用API登入時只要提供這串Token就可以登入Bot了。Bot就像個一般使用者可以發送文字訊息、圖片訊息甚至語音訊息，名稱旁邊會有個Bot的Tag來表示這不是一般的使用者。

 ←上面是一般使用者，下面則是Bot

我們使用的Discord API版本是Discord4J，主要使用語言是Java，我們可以架設一個Server接收聊天室其他人的訊息，對這些訊息進行回應。

Send Message

Message Receive Event

Login Bot

Discord API

Main

四、斷詞系統

因為要支援中文對談，所以我們需要將中文字串斷成中文詞串。例如：「我覺得鬥陣特攻好難」，我們要將此字串斷成「我 覺得 鬥陣特攻 好 難」。這樣的前處理可以幫助我們在之後的流程中更方便的去分析獲取我們需要的資訊。

我們實作的方式是長詞優先的斷詞系統，透過建立一個字典，比對字典中符合而且長度最長的單詞，將其以空格斷開來達成。

No

WordSegmentation

Input String

MaximumMatch

Node

Dictionary

More to Match?

Yes, MaximumInDictionary

Node

Node

Output String

五、使用者意圖偵測Intent Detection

在決定要回應使用者輸入的訊息之前，我們必須偵測使用者的意圖，才知道如何回應使用者。意圖是一句話可能代表的來意，詢問、招呼、抱怨、炫耀等等。

意圖是用各種不同的Patterns去判斷的，句子中的各個詞的詞性組成的順序會有不同的Patterns。

抱怨遊戲困難的Patterns是長這樣的：  
「|我 #覺得,認為 @game &正程度副詞 &困難形容詞」

|  |  |
| --- | --- |
| 符號 | 標記意義 |
| | | 其中「|我」的「|」是代表這格是一個可選擇的部分，去掉主格也沒有關係。 例如：我覺得黑暗靈魂很難 跟 覺得黑暗靈魂很難 因為這個主格去掉也沒有關係 所以這兩個都會判斷為抱怨遊戲難度的Pattern |
| # | 「#覺得,認為」的「#」是代表固定句子的這段一定為「覺得」或者是「認為」，否則就不會被判斷為這一種Pattern。 |
| @ | 「@」代表是一個專有的辭彙，在這裡是「@game」所以是遊戲的名字，也有player代表玩家的名字、discount代表遊戲的折價金額等等。 |
| & | 「&」是一群詞的集合，例如正程度副詞有：很、非常、有點、真的等等，只要符合的都會判斷為對。 |

一個pattern會對應到一種意圖，但是一種意圖可能會有多個patterns。例如：「我覺得這遊戲很難」及「這遊戲不簡單欸」，兩個的詞性組成並不一樣，但都是在抱怨遊戲的難度。

意圖有非常多種類，因為我們專題的機器人是專門於聊遊戲方面的機器人，所以我們有列出了遊戲玩家間可能會出現的對話的句子的類型，而我們三位組員都剛好是遊戲玩家，所以利用我們自身的經驗去想出這些使用者有可能會出現的意圖。（意圖列表詳見p）

收到使用者輸入的訊息之後我們必須先斷詞，斷詞後才能逐個判斷句子內各個詞的詞性，斷詞之後用Dynamic Programming的Longest Common Subsequence分析組成詞性的Pattern，利用這些Patterns，才能知道使用者的意圖。訊息的種類有很多種，比如打招呼、抱怨、詢問等等，所以我們必須要有一個演算法去判斷使用者的來意。

舉個例子：「我覺得黑暗靈魂真的很難」

這句話會被斷成「我 覺得 黑暗 靈魂 真的 很 難」

「@game」在這個例子「黑暗靈魂」是一個四字的遊戲名字，但他一般來講會被斷開成兩個詞，我們在斷詞的詞典裡面有加入這類專有的詞彙，但不可能全部涵蓋到，所以要從演算法下手，方法後述。

接下來要做的是使用者輸入的句子與預設類型的比對。

首先我們需要一個這樣的陣列，第0格全部標為0  
再來「我」跟「|我」是有對上的，標上↖及將0+1變為1  
往下一格「我」跟「&覺得」並沒有對上，標上↑，1維持不動移下來  
往右邊看「覺得」跟「|我」並沒有對上，標上←，1 一樣維持不動的右移以此類推，將整張陣列完成後會長這樣

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 我 | 覺 | 黑暗 | 靈魂 | 很 | 難 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 我 | 0 | 1↖ | 1← | 1← | 1← | 1← | 1← |
| 覺得,認為 | 0 | 1↑ | 2↖ | 2← | 2← | 2← | 2← |
| Game | 0 | 2↑ | 2← | 2← | 2← | 2← | 2← |
| 正程度副詞 | 0 | 2↑ | 2← | 2← | 2← | 3↖ | 3← |
| 困難形容詞 | 0 | 3↑ | 3← | 3← | 3← | 3← | 4↖ |

從最後一個↖的數字與預設類型的詞數判斷是吻合的（「@」類型的詞彙會被忽略），所以這句話有符合「抱怨遊戲難度」的類型。

那接下來就要從標上去的方向回去找到「@」類型的專有名詞但被斷開的詞彙，將其彙整並回傳給中樞。

一樣從最右下的↖開始，一路找找到「靈魂」這點的方向不是↖而是←，代表這是起頭（「@」的尾巴），開始往回追蹤。

到最後收集到了「靈魂」及「黑暗」後，箭號的方向又變回↖，所以追蹤結束，將收集到的詞整理合併為「黑暗靈魂」，並將這格專有名詞的意義「@game」及「黑暗靈魂」這個詞回傳給中樞。最後將這句話標記為「抱怨遊戲難度」回傳給中樞。

六、對話處理程序 Dialog Manager

我們在獲得使用者的意圖之後，透過對話處理程序去分析我們要做出的回應。有些回應只需要照Patterns去產生就好，但有些回應需要透過額外的程序去處理，以上都可以透過DM來指派。例如使用者打招呼，DM就會對應到打招呼的回覆。或者在得知使用者「想知道星海爭霸的新聞」這件事情之後，我們透過對話處理程序將這些資訊分派到尋找相關新聞是否存在的處理程序。

More Infomation

No

Yes

ResponseManager

Dialog Manager

ResponseProcess

isProcessNeed?

User Intent

七、文句產生模組 Natural Language Generation

經過DM後進入的ResponseManager就是我們的文句產生模組，在處理完使用者的訊息之後，我們以文句產生模組來產生相對應的回應。承上例，如果處理程序發現目前沒有星海爭霸新聞，就會根據設計好的Pattern格式：

「#最近沒有 @game #的 &News」

將星海爭霸填入「@game」，從News的WordList裡面挑選一個詞（也可能是八卦、消息）填入，產生回應句「最近沒有星海爭霸的新聞」。

如果有找到相關的新聞，則Pattern格式為：

「#這裡有個 @game #的 &News \n @game\_title \n @game\_link」

就會回應「這裡有個星海爭霸的新聞」並附上新聞標題及連結。

八、資料庫 Database

聊天機器人和使用者互動的每一筆資料皆會關連到特定某一名使用者，使用讀寫檔案的方式相對較難維護，因此我們決定使用資料庫軟體來管理機器人透過互動收集到的資料。由於大三資料庫課程中使用MySQL作為教材，且本身為免費軟體，因此使用MySQL作為本聊天機器人後端的資料庫。

聊天機器人和使用者互動的每一筆資料皆會關連到特定某一名使用者，使用讀寫檔案的方式相對較難維護，因此我們決定使用資料庫軟體來管理機器人透過互動收集到的資料。由於大三資料庫課程中使用MySQL作為教材，且本身為免費軟體，因此使用MySQL作為本聊天機器人後端的資料庫，資料庫內的遊戲資料取自Steam平台。

一共有五個Table：GAME\_DEAL、GAMES、MESSAGE\_LOG、USER\_FAVORITE\_GAME、USERS。

USER是存放使用者的ID與他最後發送訊息的時間。因為有時候聊天內容可能會提及其他使用者的名稱，所以這裡存放的使用者ID可以幫助斷詞系統在斷詞的時候增加精確度。

USER\_FAVORITE\_GAME是存放玩家喜歡的遊戲，在該玩家上線時可以告訴他相關的遊戲新聞、特價資訊等。

GAME\_DEAL的表格主要存放的是遊戲特價的資訊

GAMES的表格存放的是遊戲名稱與遊戲ID的對應。

MESSAGE\_LOG是存放與使用者聊天的歷史紀錄。其中最重要的是一些玩家某個遊戲破關時提供的攻略或心得，會在其他玩家在相同遊戲遇到困難時提供給他。

具體資料庫架構如下：

users

|  |  |  |
| --- | --- | --- |
| **user\_id** | last\_message\_time | name |

user\_id為使用者的唯一識別字串，name為平常與人互動時使用的代稱。

last\_message \_time紀錄該使用者最後一次與聊天機器人互動的時間。

games

|  |  |
| --- | --- |
| **game\_id** | game\_name |

game\_id 為遊戲於 steam 平台上之唯一識別編號，game\_name 為遊戲之正式名稱。

user\_favorite\_game

|  |  |  |
| --- | --- | --- |
| **user\_id** | **game\_id** | is\_checked |

user\_id 與 game\_id為外鍵，分別來自users table 與 games table。

message\_log

|  |  |  |  |
| --- | --- | --- | --- |
| **user\_id** | **time** | message | note |

user\_id 為來自 users table 之外鍵。

game\_deal

|  |  |
| --- | --- |
| **game\_id** | deal\_title |

game\_id 為來自 games table 之外鍵。

九、遊戲特價查詢功能Game Deal

對遊戲銷售平台與開發商來說，數位版遊戲的銷售和其他形式的商品販賣一樣，很大一部分的銷售量來自特價期間。從使用者的角度來看，平常許多想要的遊戲可能會因為價格高或是數量太多而望之卻步，一到特價期間便像殺紅眼似的狂買。

無論對玩家還是廠商，特價的重要性都是相當大的，然而市面上遊戲極多，要在茫茫遊戲海中追蹤特價品項相對困難且麻煩，必須要親自連上販售的網站查詢，很容易因此而錯過特價的時機。因此我們決定在聊天機器人中加入特價查詢的功能，使用者可以在聊天中直接詢問特定遊戲是否有特價。

利用Steam官方提供取得遊戲詳細資訊的Web API，在使用者用遊戲名稱進行查詢時，從資料庫尋找是否有該遊戲的ID。有了game\_id就可以呼叫API。

呼叫方式store.steampowered.com/api/appdetails?appids={game\_id}

呼叫後可以獲得遊戲資訊的JSON檔，JSON物件中包含的資訊有 game\_id, data, price\_overview, discount\_percent等欄位，得到目前的折扣率，若折扣率為0則代表該遊戲目前沒有折扣。

若遊戲有折扣則將折扣率、遊戲連結與遊戲名稱回傳Bot，最後由Bot呼叫NLG產生回應句，告知使用者特價資訊。

以遊戲ID為203770的Crusader Kings II為例，獲得的JSON部分如下：

203770：

　success： true

data：

　　type： “game”

　　name： “Crusader Kings II”

**‧‧‧**

　price\_owerview：

　　currency： “TWD”

　　initial： 86800

　　final： 21700

discount\_percent： 75

**‧‧‧**

以上為發出store.steampowered.com/api/appdetails?appids=203770的Http Request之後獲得的JSON檔，讀取這個檔案並用JSON解析套件將其解析成一個JSON物件，沿著203770、data、price\_overview、discount\_percent的層次，取得此遊戲目前有折價75%的優惠，將折扣率75%、遊戲商品頁面的連結以及遊戲名稱回傳給回話程式，Bot最後透過NLG產生

「目前Crusader Kings II特價75% Off  
http://store.steampowered.com/app/203770/Crusader\_Kings\_II/」

告知使用者當前的優惠以及遊戲連結。

十、遊戲即時新聞查詢GNN News

這個功能可以提供使用者直接詢問Bot是否有與某個遊戲相關的新聞，也會記錄使用者詢問過的遊戲並在該遊戲有新消息時主動提供給使用者。

這個功能主要透過巴哈姆特論壇提供的即時新聞RSS來實現。讀取https://gnn.gamer.com.tw/rss.xml提供的資料，此xml檔包含了數個item，以下為其中一個部分：

|  |
| --- |
| <item> <title>【試玩】劇情內容全面進化的《人中之龍 極 2》…</title> <description>由SEGA於2017年12月7日推出的…</description> <link>https://gnn.gamer.com.tw/0/156520.html</link> <pubDate>Wed, 13 Dec 2017 20:09:05 CST</pubDate> </item> |

由<title>這個標籤標記的是整個新聞的標題，而我們也可以觀察到用《》框起來的則是這則新聞談論到的遊戲或主題。所以我們會把新聞標題與他的主題都記錄下來，最後把Link也擷取出來一起回傳給NLG。