1. 動機
2. 系統架構
   1. 簡介Discord聊天室
   2. Discord API
   3. 斷詞系統
   4. 用者意圖偵測 Intent Detection
   5. 對話處理程序 Dialog Manager
   6. 文句產生模組 Natural Language Generation
   7. 資料庫 Database
3. Intent Detection
4. DM
5. NLG
6. Database
7. Deals
8. News
9. 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。

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、系統架構簡介

1. 簡介Discord聊天室

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。

2. 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。

3. 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。

4. 使用者意圖偵測 Intent Detection

此段程式主要是用來判斷使用者發出了什麼樣的意圖。像是「嗨」代表使用者發出Greeting的意圖。「最近有沒有星海爭霸的新聞」，我們可以得知使用者想了解的主體是「星海爭霸的新聞」。

5. 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。例如在得知使用者「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。

6. 文句產生模組 Natural Language Generation

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，就回應「最近沒有星海爭霸的新聞喔～」如果有找到相關的新聞，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。

7. 資料庫 Database

我們使用的資料庫管理系統是MySQL，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。

三、Intent Detection

在決定要回應使用者輸入的訊息之前，我們必須知道使用者的意圖，才知道如何回應使用者。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，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」

那抱怨遊戲困難的格式是長這樣的：

「|我 #覺得,認為 @game &正程度副詞 &困難形容詞」

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，那在這個例子「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。

“&”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。  
接下來要做的是➊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

“插入圖片”  
再來”我”跟”|我”是有對上的，標上↖及將0+1變為1  
“插入圖片”

往下一格”我”跟”&覺得”並沒有對上，標上↑，1維持不動移下來

“插入圖片”  
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  
“插入圖片”  
再往下”覺得”跟”&覺得”對上了，標上標上↖及將1+1變為2  
“插入圖片”  
以此類推，將整張陣列完成後會長這樣

“插入圖片”  
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。  
“插入圖片”  
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。  
最後將這句話為”抱怨遊戲難度”回傳給中樞。