支援訓練語句分析與擴增之 Web API 對話機器人生成機制
Web-API-Based Chatbot Generation with Analysis and Expansion for Training Sentences

王聖凱 (Sheng-Kai Wang)
國立臺灣海洋大學 /
202 基隆市中正區北寧路 2 號
nssh94879487@gmail.com

游婉琳 (Wan-Lin You)
國立臺灣海洋大學 /
202 基隆市中正區北寧路 2 號
gn01868184@gmail.com

馬尚彬 (Shang-Pin Ma)
國立臺灣海洋大學 /
202 基隆市中正區北寧路 2 號
albert@ntou.edu.tw

摘要

對話機器人 (Chatbot) 是近年來受到廣泛歡迎的新穎技術，在 Web API 技術日趨成熟的趨勢下，如何結合 Web API 與 Chatbot 技術也開始成為備受關注的議題。本研究規劃建立一個可基於 Web API 生成 Chatbot 之半自動化方法 BOTEN 及其實作平台，透過此方法，可協助應用程式開發者快速建置出指定 Web API 的 Chatbot 介面。為了確保 Chatbot 具備足夠的自然語言理解 (Natural Language Understanding, NLU) 能力，本研究透過 TF-IDF、WordNet 與 SpaCy 技術評估開發者撰寫的訓練語句，對品質不佳的訓練語句提出警訊，以提供訓練語句修改之建議；此外，本研究亦提出一個自動擴增訓練語句數量之方法，以進一步提升 Chatbot 的意圖辨識能力。

Abstract

With Web API technology becoming increasingly mature, how to integrate Web API and Chatbot technology has become an issue of great interest. This study plans to build a semi-automatic method and tool, BOTEN. This method allows application developers to build Chatbot interfaces with specified Web APIs quickly. To ensure that the Chatbot has sufficient natural language understanding (NLU) capability, this research evaluates the training sentences written by the developer through TF-IDF, WordNet, and SpaCy techniques, and suggests the developer modify the training sentences with poor quality. This technique can also be used to automatically increase the number of training sentences to improve the capability of Intent recognition.

關鍵字：對話機器人、Web API、Rasa、WordNet、SpaCy

Keywords: Chatbot, Web API, Rasa, WordNet, SpaCy

1 緒論

對話機器人 (Chatbot) 利用電腦程式模擬真人來與使用者互動，並透過通訊平台等介面讓使用者可以用文字或語音與其進行交談，也會搭配自然語言處理來幫助對話機器人判斷使用者意圖、實體，其應用包括電子商務、客戶服務、內容宣傳、推播通知、個人助理等 (Brandtzaeg & Følstad, 2017)。目前許多企業都開始使用對話機器人於實際的營運上，典型案例包含協助處理飯店客戶問題的機器人 (Michaud, 2018)、可協助團隊管理的機器人 (Toxtli, Monroy-Hernández, & Cranshaw, 2018)、以及作為特定領域之推薦專家的機器人 (Cerezo, Kubelka, Robbes, & Bergel, 2019) 等，在應用上相當多元。

另一方面，在 Web API 技術日趨成熟的趨勢下，越來越多的公司將自身服務包裝為 Web API，以提供給第三方開發者進行應用開發及服務整合，而延續此技術趨勢，如何結合 Web API 與 Chatbot 技術也開始成為備受關注的議題。根據 Stackoverflow 的研究調查 (Abdellatif, Costa, Badran, Abdalkareem, & Shihab, 2020)，開發者對於 Integration 類型的問題最為重視，特別是 Chatbot 與 Web API 之整合。在先前相關的研究上，Vaziri 等學者開發了 SwaggerBot (Vaziri, Mandel, Shinnar, Siméon, & Hirzel, 2017)，欲試圖解決上述之議題，SwaggerBot 是一個利用 OpenAPI 規範加入擴充規範後，透過編譯器生成的 Chatbot，Web API 之開發者只要能提供具備完整規範的

Swagger (亦稱為 OAS: OpenAPI Specification ("OpenAPI Specification," 2021)) 文件，便能編譯出一個能存取 REST API 的 Chatbot。然而，SwaggerBot 以每個 API 端點的名稱作為唯一的訓練語句，故產生的 Chatbot 自然語言理解能力較不理想，需透過 Power User 為既有功能的存取創造捷徑以及捷徑的同義詞，補強其 NLU 能力。其次，各端點之間的呼叫缺乏一個完整的故事流程規劃，所以無法提供終端使用者一個友善的使用者體驗。

我們再進一步分析 Web API 與 Chatbot 之整合工作，從開發者角度有三個較困難的環節：(1) 開發者需具備 Chatbot 開發框架的詳盡知識，需要熟悉對話流程所需的文件設定與操作步驟；(2) 需能撰寫 Chatbot 與 Web API 之轉接元件(Adapter)，除了要能透過程式將 Chatbot 之組成元素 (如 Intent、Entity、Slot 等) 與 Web API 之服務功能銜接起來，還要能妥善處理服務回應結果 (Service Response) 之呈現；(3) 需要撰寫足夠品質與數量的訓練語句，以訓練出具備足夠 NLU 能力的 Chatbot，正確判斷終端使用者的意圖(Intent)，並做出適當的回覆。因此，為能降低 Chatbot 開發之複雜度，我們採取的策略是基於模型驅動工程(Model-Driven Engineering) (Schmidt, 2006) 的概念，希望能讓開發者聚焦於 Web API 與 Chatbot 之概念整合方式，而非底層框架細節與轉接元件之開發。由於 Swagger 是目前最被廣泛運用的 Web API 之描述語言，許多 Web API 之開發者均已提供 Swagger 描述文件，以供客戶端與其他開發者運用，本研究首先遵循 Swagger 之標準擴充機制，以在 Swagger 文件中加入 Chatbot 之相關元素，再建立一個可剖析與生成 Chatbot 之執行引擎： BOTEN，來輔助開發者以半自動化的方式建構指定 Web API 之 Chatbot。

在另一方面，一般常見的 Chatbot NLU 模型都會對使用者輸入的語句辨識其背後的意圖(Intent)，以此作為後續動作的執行依據。NLU 模型還可計算置信度 (Confidence)，代表對辨識結果的信心程度，Confidence 過低可能造成辨識錯誤。因此，為了提昇 Confidence、增強 Chatbot 的 NLU 能力，我們提供訓練語句的填寫機制，讓開發者可以填入多種不同句型的訓練語句，並對其進行評估，若是有不同意圖的訓練語句意思過於相近，會透過 BOTEN 介面對開發者提出警訊，建議開發者進行修改。建構完 Chatbot 後，亦可以讓開發者選擇是否要透過 WordNet 技術自動生成更多的訓練語句，以增強 Chatbot 的 NLU 能力。

預期的運作模式是由 Chatbot 開發者在 Web API 之 Swagger 文件上補充 Chabot 相關設定，填入訓練語句，接著將文件送至 BOTEN 系統後，即可自動生成可執行的 Chatbot，還可以進一步自動增加訓練語句，後續終端使用者即可使用此 Chatbot 系統，以對話的方式去使用 Web API 之功能，並能直接查看彙整過的服務回應結果，降低整合 Web API 與 Chatbot 的複雜度與開發成本。

2 背景知識與相關研究

2.1 Rasa

Rasa (Bocklisch, Faulkner, Pawlowski, & Nichol, 2017) 是開源的機器學習框架，可以用來創建有上下文功能的對話機器人。Rasa 框架具有 Rasa NLU、Rasa core sdk 兩個主要功能，Rasa NLU 負責自然語言的斷句、訓練 model，而 Rasa core sdk 負責之後的關鍵字抓取。在構建對話機器人時，意圖 (Intent)、動作 (Action) 和關鍵字 (Entity) 三者扮演了重要角色。意圖是指使用者語句的目的，由開發者所設定，在每個意圖當中都可以抓取關鍵字。

在 Rasa 的資料設定內也可以設定記憶變數 (Slot)。在多階段對話中記憶變數扮演了階段間溝通的角色，例如在第二階段時若需要第一階段的關鍵字，就可以使用記憶變數來進行儲存。本研究採用 Rasa 作為對話機器人之自然語言理解核心引擎。

2.2 凝聚力(Cohesion) 與耦合力(Coupling)

在軟體工程領域，凝聚力(Cohesion) 係指程式模組內部功能或資料的相依程度，耦合力 (Coupling) 則是程式模組之間的獨立程度 (Josikakar, 2021)。一般而言，開發者會希望提高 Cohesion 使程式較易維護並且適合再利用，降低 Coupling 以提高模組間的獨立程度，減少系統溝通錯誤的可能性，以達成 "High Cohesion, Low Coupling"之目標。

在撰寫 Swagger 時，我們同樣希望開發者能夠注意訓練語句之間的 Coupling，降低不同 Intent 之間訓練語句的相似度，避免因為不同 Intent 的訓練語句意思過於相近，導致訓練出

的 NLU 模型發生不同 Intent 間的辨識錯誤，此外還能提高模型辨識的置信度 (Confidence)，讓訓練出的模型更加準確。至於同一意圖之下訓練語句的相似度 (Cohesion)，為了避免訓練出的 Chatbot 模型發生過度擬合 (Overfitting)，開發者應盡量撰寫多樣化的訓練語句句型，因此本研究未將 Cohesion 之改善列為研究目標。

2.3 TF-IDF、WordNet 與 SpaCy

為了提高 Chatbot 模型的自然語言理解能力，在 Swagger 文件前處理的階段，我們會利用 TF-IDF、WordNet 與 SpaCy 技術計算訓練語句之間的相似度，對不同 Intent 間意思過於相近的語句提出警訊，建議開發者修改。之後可以選擇是否要進一步利用 WordNet 將使用者輸入的訓練語句進行擴增，然後又使用 SpaCy 把擴增出的語句過濾掉與原句意思相差較大的句子，進而訓練出品質更好的模型。

TF-IDF (riturajsaha, 2022) 是一種用於資訊檢索與文字探勘的常用加權技術，為一種統計方法，用來評估單詞對於文件的集合或詞庫中一份文件的重要程度。其包含兩個部分：詞頻 (term frequency，TF) 跟逆向文件頻率 (inverse document frequency，IDF)，TF 係指某一個給定的詞語在某文件中出現的次數，IDF 則是將該詞語總共出現在幾篇文件的文件數取倒數，透過將 TF 與 IDF 相乘，即可計算給定詞語在某文件中所佔的權重。

WordNet ("What is WordNet?,") 是一個由普林斯頓大學認知科學實驗室心理學教授 George A. Miller 的指導下建立和維護的英語字典。WordNet 根據 word 的意義將它們分組，每一組具有相同意義的 word 稱為一個 synset（同義詞集合），這些集合之間也由各種關係連接。透過查詢 WordNet ，系統可以找出訓練語句中，每個 word 的同義詞，然後再排列組合出原句的多個同義句。

SpaCy ("Word vectors and semantic similarity,") 是用於自然語言處理的開源程式庫，主要開發者為 Matthew Honnibal 和 Ines Montani，其透過其語料庫中的預訓練模型來生成 word 向量，便可以用向量間的 cosine 相似度直接比較兩句的相似度。

2.4 相關研究

Telang (Telang, Kalia, Vukovic, Pandita, & Singh, 2018) 等學者為對話機器人提出了一個概念框架 (Conceptual Framework)，該概念框架由五個相關職責組成： Dialog Manager (管理自然語言對話)；Inference Engine (提取使用者意圖)；Knowledge Base (進行推理和規劃)；Planner (產生執行計畫)；External Services (結合外部功能以能實際執行對話互動)，與現有的 IFTTT (If this, then that) 框架相比，複雜的對話機器人可以透過較靈活的方式開發。

Soler (Pérez-Soler, Guerra, & Lara, 2020) 等學者提出了一種用於對話機器人開發的模型驅動工程方法，設計了 Chabot 之 meta model 和領域特定語言 (DSL: Domain Specific Language): Conga， 可搭配多個 Chatbot 平台如 (Dialogflow 或 Rasa) 的程式碼解析器 (Parser) 與程式碼生成器 (Generator)，以生成 Chatbot。此方法亦提供一個平台推薦器，可根據 DSL 建議合適的目標 Chatbot 平台。

Pietro (Chittò, Baez, Daniel, & Benatallah, 2020) 等學者提出一個對話機器人生成方法，其透過網頁 HTML 之標註 (annotation) 去生成對話機器人，可設定意圖、對話語句、關鍵字、類別。此方法希望達成對話式網站瀏覽，即基於自然語言對話，讓使用者可以透過「與網站交談」來查詢所需之內容和功能，不透過鍵盤和滑鼠來操作圖形 UI，而是運用 Selenium 去自動化操作瀏覽器。

Daniel (Daniel, Cabot, Deruelle, & Derras, 2020) 等學者提出了開源的 Xatkit 平台，設計了兩種領域特定語言 (DSL)，來建構對話機器人：(1) Intent Package 使用訓練語句、上下文訊息和匹配條件來描述使用者意圖；(2) Execution Package 串聯使用者意圖與回應動作，以作為對話機器人行為定義的一部分，在執行部分通常涉及對後端服務之呼叫，則再藉由 Platform Package 之定義來實現。Xatkit 除了提供領域特定語言以定義對話機器人外，亦提供一個模組化的執行引擎，可自動部署對話機器人應用程式，並在所選平台上管理定義的對話邏輯。

3 系統設計與方法

3.1 操作概念

底下透過兩個應用情境來分析本系統之操作流程與運作模式，第一部分為開發者部署階段，第二部分為使用者操作對話機器人之情境，整體使用情境如圖 1 所示。

在此操作概念腳本中，開發者希望開發一個整合推薦景點 API 的對話機器人，為了達到這個目的，開發者需要撰寫擴充之 Swagger 文件 (BotSwagger)，並使用 BOTEN 來獲取建置 Chatbot 的設定檔。開發者輸入 BotSwagger 文件後會經過格式驗證與訓練語句品質檢查，此時可以對內容再做修改，接著便會轉換為 Rasa 的設定檔，並 Push 到 GitHub 進行版本控制。這時開發者可以決定是否要自動擴增 Rasa 的訓練語句，不論有無進行擴充，開發者都可以接著部署對話機器人。

於操作階段中，終端使用者能透過開發者建置好的對話機器人來使用 API，使用者僅需要向對話機器人下達指令，並遵循對話機器人的指示便能使用 API，獲得以表格呈現之 API 執行結果。

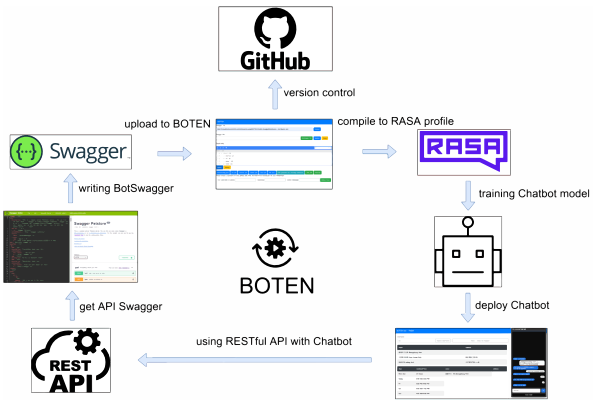

圖 1. BOTEN 使用情境圖

3.2 系統架構圖

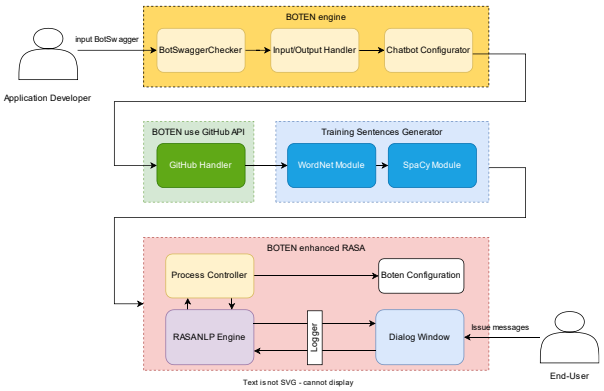

圖 2. 系統架構圖

本研究提出的 BOTEN 系統架構分為四個部分，分別為轉換階段、版本控制階段、擴增語句階段與執行階段，如圖 2 所示。(1) 轉換階段是以管線 (Pipeline) 的方式將 BotSwagger 依照順序轉換為對話機器人之設定；(2) 在版本控制階段，開發者可將 BOTEN 產生的 Rasa 設定檔推送至 GitHub Repository 進行版本控制；(3) 在擴充語句階段，BOTEN 會將生成的 Rasa 設定檔作進一步分析與處理，以提升意圖辨識能力；(4) 在執行階段中，開發者可建置與部署 Chatbot，建置好的 Chatbot 是以 Rasa 框架為底層的對話應用程式，可搭配前端 Web 介面讓終端使用者與 Chatbot 交談。

3.3 BotSwagger 訓練語句品質分析

開發者輸入 BotSwagger 時，BOTEN 會進行分析，其中訓練語句會利用 TF-IDF 技術，以所有的語句文本為基礎，先將所有文字轉成小寫，去除不具備重要意義的停用詞 (Stop Words)，對剩餘的詞語做詞形還原 (lemmatization)，接著計算所有詞語在個別語句中所佔的權重，建立語料庫索引 (corpus index)。將每個訓練語句的組成詞語用權重值來表示，即可將句子轉成一維向量，計算句子向量間的餘弦相似度 (Cosine Similarity)，即可得到兩個句子的相似度值。

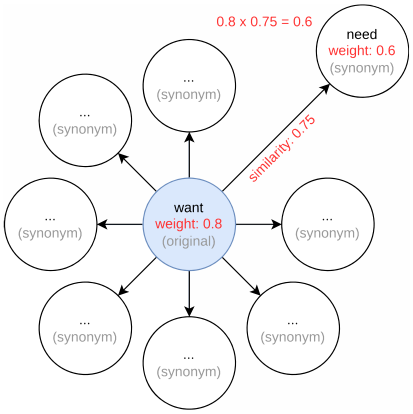

圖 3. WordNet 擴增同義詞的權重計算方式

由於 Chatbot 的訓練語句長度通常較短，有可能發生兩句雖然意思相近，但卻因為剛好沒有出現相同的詞語，導致計算出的餘弦相似度仍然偏低的情況，為此本研究提出兩種改善方法：(1) 透過 WordNet 技術以查字典的方式，找出所有訓練語句所用詞語的同義詞，並將這些同義詞也加到語料庫索引中，利用 SpaCy 技術計算同義詞與原始詞語間的相似度，以原始詞語的權重值乘上同義詞與原始詞語間的相似度，作為該同義詞在語料庫索引中的權重值，如圖 3 所示。我們將這些同義詞也視作原始詞語所屬句子的詞語，重新計算所有訓練語句間的相似度。(2) 除了透過 TF-IDF 得到的權重值計算句子間的餘弦相似度，同時也利用 SpaCy 計算句子間的相似度，由於兩種計算方式的語料庫 (Corpus) 不同，故計算出的相似度也會不同，因此便得到了兩句間的兩種不同的相似度數值，再將這兩個數值以特定比率加權計算，作為兩句間的加權相似度。

透過同時使用上述兩種相似度計算方式，目前本系統已經能準確評估訓練語句間的相似度 (Coupling)，並對相似度過高但卻屬於不同 Intent 的句子提出警訊。BOTEN 對於高相似度的判斷方式如圖 4 所示，所有的句子輪流當作基準句 (圖 4 以句子 2-1 作為基準句)，讓自己以及其他句子與基準句比較並計算相似度，基準句自身與自身的相似度則為 1.0；取得所有相似度數值後，將相同 Intent 的句子相似度相加，並比較這些相加後的數值，若有 Intent 相似度相加後的數值大於基準句所屬 Intent 的數值，則將該 Intent 相加前最大的相似度數值，所屬的語句視為相似度過高的句子。透過此方法既能保留相同 Intent 之下語句的自由度，又能找出相似度過高的句子，不論相同 Intent 之下的語句如何變化，只要其相似度的總和不要大於基準句 Intent 的相似度總和即可。關於 WordNet 尋找同義詞的標準，兩種句子相似度計算方式的權重比率，以及此方法有效性的驗證，會在後續實驗章節中說明。

	A	B	C	D
1	based on 2-1 of case6	2-1 is too similar to 1-3		
2				
3	intent - sentence number	sentence	similarity	sum
4	1-1	I want to get information about venues	0.5266195	
5	1-2	Get attractions explore	0.6880899 step.3	2.0041394
6	1-3	I want to get the venues	0.7894299	
7	**2-1**	**I want to get explore about venues**	**1.0000000**	step.2
8	2-2	Get explore of attractions	0.6942862 step.1	1.9072807
9	2-3	Search for attractions	0.2129945	
10	3-1	I want to get tips about venues	0.5754928	
11	3-2	Get tips of attractions	0.2672742	1.6865370
12	3-3	get explore about venues	0.8437699	
13	4-1	I want to get hours about venues	0.5734294	
14	4-2	Get opening time of attractions	0.2750190	1.0502737
15	4-3	Attraction opening hours	0.2018253	
16	5-1	I want to get next venues	0.6137682	
17	5-2	Get next attractions	0.2778281	1.2050529
18	5-3	see next venues	0.3134566	
19				
20				

圖 4. 語句相似度過高之判斷方式

3.4 訓練語句擴充機制

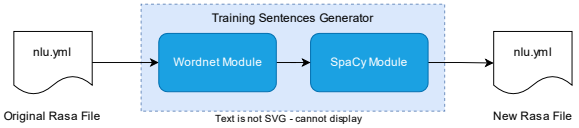

圖 5. 訓練語句擴充機制

除了 BOTEN 核心功能之外，我們亦提出了一個專門用於生成 Rasa 訓練語句的服務，透過 BOTEN 串接此服務，開發者能做到 Rasa 訓練語句的擴充，其內部機制可分為 WordNet Module 與 SpaCy Module 兩部份，如圖 5 所示。

為了將訓練語句中比較有意義的詞彙進行擴充，我們使用 WordNet 的 wup_similarity (Gupta_OMG, 2022) 相似度演算法進行擴充後新的句子與原句的比較，其全名為 Wu-Palmer Similarity，它根據同義詞在上位詞樹中相對於彼此的位置來計算相似度，LCS (Least Common Subsumer) ("Sample usage for wordnet,") 為分類樹中最深的共同祖先，depth 則是指在分類樹中的深度，其計算公式如下：

$$Wup_{similarity}(w1, w2) = 2 * \frac{depth(lcs(w1,w2))}{(depth(w1)+depth(w2))} \quad (1)$$

語句擴增的處理流程分為以下步驟：(1) 首先我們將所有的訓練語句進行斷詞，把一個句子切成一個個 Token；(2) 接著我們把每個 Token 詞型還原成 Lemma；(3) 剃除掉不重要的 Stop Word 後，完成擴充前的準備；(4) 利用

WordNet 查詢每個有意義 Token 的同義詞；(5) 透過 wup_similarity 演算法評估查詢到的同義詞與原始的 Token 的相似度，利用笛卡爾乘積交叉比較兩組同義詞的相似度，並計算平均值；(6) 只要平均值高於 0.3 則視為該 Token 的同義詞 (平均值參數取 0.3 的理由會在後續實驗章節說明) ；(7) 將這些同義詞與先前剃除掉的 Stop Word 依它們在原始句中的相對位置，將 Stop Word 以外的字詞輪流替換成新生成的同義詞，組合出多句與原句相同意思的同義句。

WordNet 擴充後新生成的眾多語句中會存在與原句意思相差過大的句子，這可能會導致 Rasa 訓練出不準確的模型，因此我們透過 SpaCy 進一步對新生成的句子進行過濾，這會使用到 SpaCy 的相似度演算法 ，其計算機制如下：(1) 將每個句子斷詞後建立多個 Token 的集合；(2) 透過 Word Embedding 的演算法，由語料庫中的預訓練模型來生成 Word 向量；(3) 計算該句所有詞向量的內積作為其句向量；(4) 比較句子向量間的 Cosine 相似度，其計算公式如下：

$$SpaCy_{similarity}(s1, s2) = \frac{s1^{-} \cdot s2^{-}}{\|s1\| \times \|s2\|} \qquad (2)$$

在使用上可分為以下步驟：(1) 把所有新生成的句子與原句逐一比較相似度，(2) 最後保留相似度大於 0.7 的句子寫回 nlu.yml 當作訓練語句，完成擴充 (相似度參數取 0.7 的理由，將在後續實驗章節說明) 。

4 案例與實驗

4.1 實驗說明

本實驗所使用的 Web API 主要來源是 APIs.guru ，此為開源的 OAS 儲存庫，目前已經發佈了大量的 Swagger 文件(Web API 描述文件)。實驗準備期間我們基於現有的 Swagger 文件將其擴增為 BotSwagger，接著我們會將撰寫好的 BotSwagger 文件上傳至 BOTEN，以進行後續實驗。

4.2 實驗目標

本實驗分為三個部份：

- **BOTEN 可行性分析：**為能評估 BOTEN 核心功能是否實際可行與有效，我們進行了 11 個案例實驗，在此我們以其中 2 個最具代表性的案例進行說明。

- **語句 Coupling 分析之參數設定實驗：**為能於 BotSwagger 訓練語句品質之 Coupling 評估時，能找出最適合的評估方法及指標，我們進行了 BotSwagger Preprocessing 相關實驗。

- **語句擴充方法參數設定與有效度實驗：**擴充 Rasa 訓練語句時，為了找出最適合的相似度參數，我們進行了一系列實驗，以評估擴充語句後 Rasa 訓練模型與真人的 Intent 分類判斷是否一致。

4.3 BOTEN 可行性分析

- **實驗案例 E1 – Foursquare:** 實驗 E1 使用 Foursquare 的景點推薦服務，使用者能透過評論定位地點來獲取獎勵。此實驗使用了所有 BOTEN 提供之擴充功能，包括設定故事性的流程，搜尋推薦景點後查詢此景點的開放時間和介紹等、提供預設參數和自動取得地理位置功能，使用者也可直接使用開發者所提供的 Client ID 來使用服務，並根據 Html5 Geolocation API 自動取得經緯度，將經緯度直接填入參數，此實驗測試了完整的 BOTEN 功能。

- **實驗案例 E2 – The Movie Database:** 實驗 E2 使用 The Movie Database 的電影推薦服務，The Movie Database 為一個電影相關資料庫，其包含電影、演員、電視節目等。此實驗設定了一個故事的流程，在搜尋熱門電影後查詢電影的詳細資訊，且使用預設參數，使用者無需申請 API key 即可使用此對話機器人。

表 1 為所有實驗運用到的 BOTEN 功能與其結果，顯示「Yes」的部分為開發者有運用到的 BOTEN 功能，顯示「N/A」的部分則是沒有運用到的 BOTEN 功能，目前分析項目不包含版本控制等通用功能。本研究的實驗案例皆成功整合了 Web API 與 Chatbot，能更便利地建置以提供 API 服務為目標之對話機器人，使用者也能方便地使用 API 服務。

驗證之功能特性	實驗 E1	實驗 E2
多個服務串接	Yes	Yes
自動取得地理位置	Yes	N/A
使用預設參數	Yes	Yes
使用 Regex	Yes	Yes
使用 JSON Path 呈現表格	Yes	Yes

表 1. 案例實驗結果分析

4.4 語句 Coupling 分析之參數設定實驗

本階段實驗以前階段之實驗案例作為參考案例，並為他們各自撰寫了一份品質良好的 BotSwagger。在 BotSwagger 轉換階段，會對開發者輸入的 BotSwagger 進行一系列驗證，評估 Chatbot 訓練語句的品質。若出現不同意圖之下的訓練語句意思過於相近，會回傳警訊訊息，一則訊息就代表一對過於相近的句子，本研究透過計算警告訊息的數量來評估實驗結果，期望品質良好的 BotSwagger 能夠完全不出現警告訊息，若出現警告訊息則視為假警報 (False Negative)。

由於 Chatbot 的訓練語句通常長度較短，單純使用 TF-IDF 效果不佳，所以本研究提出兩種改善方法，其一是透過 WordNet 對原始的語句擴充，增加彼此出現相同詞語的機會，其二是同時使用 SpaCy 計算句子相似度，而後再與先前計算的句子相似度加權計算。為使用這兩種方法，我們透過以下實驗找出以下參數：(1) WordNet 生成同義詞的判斷標準，以及 (2) TF-IDF 相似度與 SpaCy 相似度的加權比率。

- **實驗 P1 - WordNet 同義詞:** 本實驗的目的為 WordNet 生成同義詞的階段，試圖找出相似度演算法參數的最佳解。使用 WordNet 生成訓練語句詞語的同義詞時，為了生成與原始詞語意思足夠相近的同義詞，可以使用 wup_similarity 演算法或是 SpaCy 演算法計算原始詞語與同義詞的相似度，並得到介於 0 到 1 之間的相似度數值，我們以 0.1 為單位，將 0.0 至 1.0 的每個刻度作為參數，當相似度數值大於參數時，就把該同義詞視作原始句子的詞語，加入語料庫索引計算權重。

- **實驗 P2 - TF-IDF 與 SpaCy 加權比率:** 本實驗的目的為找出 TF-IDF 與 SpaCy 兩種句子相似度演算法，能得到最佳結果的加權比率。我們以 TF-IDF 比 SpaCy 比率為 1：1、2：1、1：2 三種比率分別測試，將能得到最少假警報的比率作為實驗結果。

實驗 P1 的結果發現，不論相似度參數為何，SpaCy 演算法的假警報數量普遍少於 wup_similarity 演算法，且計算時間較短，故本研究以 SpaCy 作為 BotSwagger 轉換階段，WordNet 生成同義詞的判別依據。另外還從結果觀察到，在相似度參數大於 0.6 之後，便不再出現假警報。

實驗 P2 的結果如表 2 所示，案例 E1 在相似度參數為 0.8 時，能得到最好的結果，不論 TF-IDF 與 SpaCy 的加權比率為何，皆不會產生假警報，故我們取 0.8 作為相似度參數。接著，在案例 E2 則可以發現將相似度參數設為 0.8 時，TF-IDF 比 SpaCy 比率 2：1 的實驗結果最好，只有出現一則假警報，故我們取 2：1 作為加權比率。

	案例 E1			案例 E2		
Similarity	1：1	1：2	2：1	1：1	1：2	2：1
0.0	11	9	11	5	4	4
0.1	10	11	13	5	4	4
0.2	9	9	12	2	1	2
0.3	7	5	10	4	3	4
0.4	3	3	5	1	1	1
0.5	1	1	2	2	2	2
0.6	0	0	1	3	3	2
0.7	0	0	1	3	3	3
0.8	0	0	0	3	3	1
0.9	0	0	1	2	3	1
1.0	0	0	1	2	3	1

表 2. TF-IDF 與 SpaCy 加權比率實驗假警報數

4.5 語句擴充方法參數設定與有效度實驗

本階段實驗比較擴充訓練語句後所獲得的訓練模型之Confidence，試圖找出各項變因的最佳組合(可得到最高的 Confidence)。同時我們請三位平常有 Chatbot 使用經驗的使用者協助實驗，請他們決定訓練語句對應的正確 Intent 為何，並以多數決作為正確答案，評估 Rasa 模型的判斷是否正確。接著將此案例得到的各項變因組合，套用到實驗 E2 案例中，驗證是否同樣能提昇自然語言理解能力。最後我們結合前階段 BotSwagger Preprocessing 實驗，整合所有的實驗結果，驗證是否能訓練出具備良好 NLU 能力的 Chatbot。

- **實驗 C1 - WordNet:** 本實驗目的為測試擴充語句的 WordNet 階段，試圖找出相似度演算法參數的最佳解。作法同實驗 P1，最後以能得到最佳 Confidence 的參數為實驗結果。

- **實驗 C2 - Stop Words and Lemmatisation:** 本實驗目的為測試擴充語句的前處理階段，去除原始句子的 Stop Words ，並將 Token 進行詞型還原，判斷是否能提昇模型訓練結果的 Confidence，並根據結果對最佳相似度參數進行調整。

- **實驗 C3 - SpaCy:** 本實驗目的為測試擴充語句的 SpaCy 階段，試圖找出相似度演算法參數的最佳解。將前兩次實驗的結果作為參數代入本次實驗，生成新的句子後，將原句與新生成的句子逐一比較其相似度，過濾掉部份意思相差過遠的句子。作法同實驗 P1，將能得到最佳 Confidence 的參數作為實驗結果。

- **實驗 C4 -案例 E2 擴充語句:** 將前三次實驗的結果代入本實驗，測試在不同的案例中，同樣的相似度參數是否也能提昇 Confidence，並以真人判斷當作答案，比較 Rasa 模型的判斷與真人是否相同。

本階段實驗結果如表 3、表 4 及表 5 所示，參數 0.1、0.2 及顯示為「N/A」的實驗，係因為擴充語句或著訓練模型時間過長而無法完成的實驗，其餘部份「c」(correct) 代表 Intent 判斷正確且 Confidence 高於 0.9，「a」(acceptable) 代表 Intent 雖然判斷正確，但是 Confidence 低於 0.9，至於 Intent 判斷錯誤的情況，以「i」(incorrect) 表示，最後我們將表格中最佳的結果用粗體表示。

實驗	**Original**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
實驗 C1	4c 1a 5i	N/A	**6c** **1a** **3i**	5c 1a 4i	4c 0a 6i	5c 0a 5i	4c 2a 4i	4c 1a 5i
實驗 C2	4c 1a 5i	**9c** **0a** **1i**	7c 1a 2i	7c 1a 2i	7c 1a 2i	3c 4a 3i	6c 2a 2i	5c 3a 2i
實驗 C3	4c 1a 5i	8c 0a 2i	8c 0a 2i	8c 0a 2i	7c 2a 1i	**10c** **0a** **0i**	7c 1a 2i	9c 0a 1i

表 3. Confidence 參數比較

實驗	**結果**
實驗 C4 - 擴充前	1c 5a 4i
實驗 C4 - 擴充後	**10c 0a 0i**

表 4. 案例 E2 擴充語句前後比較

根據實驗 C1 的結果，我們發現 WordNet 的 wup_similarity 演算法在相似度設為 0.4 時結果最好，所以將這個結果與實驗 C2 比較，此時卻發現在移除 Stop Word 與詞型還原的情況下，相似度若設為 0.3，Confidence 能有更好的表現，於是我們將這個結果代入下一次實驗。接著到了實驗 C3 ，若 SpaCy 相似度參數設為 0.7 ，則 Confidence 能有最好的表現，所有 Intent 皆判斷正確，由此得知 WordNet wup_similarity 參數設為 0.3，SpaCy 參數設為 0.7 ，為本階段實驗的最佳解。

透過實驗 C4，我們發現在其他案例，擴充語句同樣能夠有效提升 Confidence 表現，避免 Intent 判斷錯誤。

4.6 語句 Coupling 分析與擴充語句機制整合實驗

最後，我們對已整合語句 Coupling 分析與擴充語句機制之 Chatbot 進行整合驗證，對於三個案例 (E1, E2, 以及另一個案例 E3: Graph Hopper Direction API) 執行完整的 Chatbot 生成流程，

以驗證是否真的能生成具備良好 NLU 能力的 Chatbot。

- **實驗案例 E3 – Graph Hopper Direction API:** 實驗案例 E3 為一路線規畫服務，此外還能判斷使用者的所在地以及給定時間內能到達的地方。本研究替以上三種不同服務的呼叫端點，各自撰寫訓練語句，並透過語句 Coupling 分析的建議，改善訓練語句的內容，接著進一步透過 WordNet 擴充語句，實驗是否能增加 Chatbot 的 NLU 能力。

此實驗的結果如表 5 所示。從三個案例的實驗結果可得知，透過語句 Coupling 分析進而改善之語句 (改善語句相似度過高問題) 皆能有效提高問答正確率，而透過案例 E2 及 E3 則可發現，透過 WordNet 擴充語句，能夠更進一步提高問答準確率，建立出 NLU 能力更佳的 Chatbot。

案例	原始之語句、未擴充	原始之語句、有擴充	改善之語句、未擴充	改善之語句、有擴充
案例 E1	2c 0a 3i	2c 0a 3i	4c 1a 0i	4c 1a 0i
案例 E2	0c 3a 2i	3c 0a 2i	4c 0a 1i	**5c 0a 0i**
案例 E3	2c 0a 2i	3c 0a 1i	3c 0a 1i	**4c 0a 0i**

表 5. 整合實驗結果

5 結論與未來研究方向

本研究提出了一個基於 Web API 之半自動化對話機器人生成機制：BOTEN，可協助應用程式開發者快速建置出指定 Web API 的 Chatbot 介面。實驗展示了 BOTEN 核心系統功能均有成功實現、自然語言理解能力亦有有效的提升。

未來我們規劃進行兩個改善方向：(1) 優化 API Chatbot 之使用流程，讓使用者可以詢問先前填入了哪些參數，以提升使用便利性。此機制將由系統以自動化的方式提供，無需開發者特別撰寫此意圖的訓練語句。(2) 規劃更豐富的 API 回應資料之呈現方式，除了現有的表格形式，規劃提供包含超連結、圖片、影片等豐富資料之回覆結果。

參考文獻

Abdellatif, A., Costa, D., Badran, K., Abdalkareem, R., & Shihab, E. (2020). *Challenges in chatbot development: A study of stack overflow posts.* Paper presented at the Proceedings of the 17th international conference on mining software repositories.

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. J. a. p. a. (2017). Rasa: Open source language understanding and dialogue management.

Brandtzaeg, P. B., & Følstad, A. (2017). *Why people use chatbots.* Paper presented at the International conference on internet science.

Cerezo, J., Kubelka, J., Robbes, R., & Bergel, A. (2019). *Building an expert recommender chatbot.* Paper presented at the 2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE).

Chittò, P., Baez, M., Daniel, F., & Benatallah, B. (2020). *Automatic generation of chatbots for conversational web browsing.* Paper presented at the International Conference on Conceptual Modeling.

Daniel, G., Cabot, J., Deruelle, L., & Derras, M. J. I. A. (2020). Xatkit: a multimodal low-code chatbot development framework. *8*, 15332-15346.

Gupta_OMG, M. (2022). NLP | WuPalmer – WordNet Similarity. Retrieved from https://www.geeksforgeeks.org/nlp-wupalmer-wordnet-similarity/

Josikakar. (2021). Software Engineering | Coupling and Cohesion. Retrieved from https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/

Michaud, L. N. J. I. P. (2018). Observations of a new chatbot: drawing conclusions from early interactions with users. *20*(5), 40-47.

OpenAPI Specification. (2021). Retrieved from https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.1.0.md

Pérez-Soler, S., Guerra, E., & Lara, J. d. (2020). *Model-driven chatbot development.* Paper presented at the International Conference on Conceptual Modeling.

riturajsaha. (2022). Understanding TF-IDF (Term Frequency-Inverse Document Frequency). Retrieved from

https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

Sample usage for wordnet. Retrieved from https://www.nltk.org/howto/wordnet.html

Schmidt, D. C. J. C.-I. C. S.-. (2006). Model-driven engineering. *39*(2), 25.

Telang, P. R., Kalia, A. K., Vukovic, M., Pandita, R., & Singh, M. P. J. I. I. C. (2018). A conceptual framework for engineering chatbots. *22*(6), 54-59.

Toxtli, C., Monroy-Hernández, A., & Cranshaw, J. (2018). *Understanding chatbot-mediated task management.* Paper presented at the Proceedings of the 2018 CHI conference on human factors in computing systems.

Vaziri, M., Mandel, L., Shinnar, A., Siméon, J., & Hirzel, M. (2017). *Generating chat bots from web API specifications.* Paper presented at the Proceedings of the 2017 ACM SIGPLAN international symposium on new ideas, new paradigms, and reflections on programming and software.

What is WordNet? Retrieved from https://wordnet.princeton.edu/

Word vectors and semantic similarity. Retrieved from https://spacy.io/usage/linguistic-features#vectors-similarity