國立雲林科技大學資訊工程系

碩士論文

Department of Computer Science and Information Engineering
National Yunlin University of Science & Technology

Master Thesis

自然語言處理研究與應用

Research and Application of Natural Language Processing

黃俊霖

Jun-Lin Huang

指導教授：許正欣 博士

Advisor：Jeng-Shin Sheu, Ph.D.

中華民國 107 年 6 月

June 2018

國立雲林科技大學
研究所碩士班學位論文考試委員會審定書

本論文係**黃俊霖**君在本校 **資訊工程系**提論文「**自然語言處理研究與應用**」合於碩士資格水準，業經本委員會評審認可，特此證明。

口試委員：　許正欣　許正欣

黃國鼎　黃國鼎

連振凱　連振凱

指導教授：　許正欣　許正欣

所　長：郭文中

中　華　民　國　109　年　9　月　30　日

摘要

近幾年有關人工智慧與機器學習相關領域蓬勃發展，人機互動介面也變得更加智慧化，應用包含語音助理、對話情境分析、文章段落分析等，而這些應用的基礎來自於機器學習中的子領域「自然語言處理」。

本論文首先介紹近代語言學發展，之後進一步分析字詞與句法架構，接著介紹近年來受歡迎的自然語言處理工具以及演算法，並實際應用此工具及演算法，設計一個人機互動應用程式，再依照程式的執行結果，分析此程式的優缺點。

關鍵字：自然語言處理、人工智慧、機器學習

ABSTRACT

With the rapid development of information technology and networks. The need for computers to deal with natural language becomes more and more urgent. Natural language processing in addition to data searching and data mining. In recent years, a large number of human-computer interaction interfaces, have been widely used. However, in the actual research of natural language processing, the system structure used has become more and more complex. But it's still from the Chinese grammar. Like words, part of speech, and analysis of conversational situations.

In this thesis, we introduce the development of Chinese linguistics. Afterwards, analysis of Chinese words and syntactic structures. Next, we introduce natural language processing tools and algorithms that are quite popular in recent years. And use these tools, design a simple chatbot. Analyze the advantages and disadvantages of this tool and algorithm.

Keywords：Natural Language Processing, Artificial Intelligence, Machine Learning

目錄

圖目錄

第一章、研究背景與動機

近年隨著經濟發展，由於工商業需求，傳統的人工作業已經無法滿足需求，因而需要自動化的機器來取代人工作業，而資訊科技的進步，使電腦軟硬體的技術大幅進化，此時的電腦足以處理大量資料；而人工智慧與機器學習的發展，使電腦能擁有自主分析及預測的能力；語言學家透過分析語言中，字詞、句子以及語法，並將這些內容整理成語料集，使電腦能更直接的處理。

而在語言分析上，分為兩派，一派認為，必須以語言學為基礎，透過語言學家對語言現象的認識，以規則型式描述或解釋問題行為或問題特性，稱為規則派。規則派必須透過一系列的原則來描述語言，並判斷一個句子是正確或錯誤，並歸納出一系列的語法，形成一套複雜的規則集，對語言進行分析處理。

另一派則是以統計分析為基礎，稱為統計派，此方法著重於利用數學模型，從能代表自然語言規律的文字中，抽取語言現象或統計規律。統計派來自多種數學基礎與理論，例如最佳化方法、機率圖模型、神經網路、深度學習等。由此來判斷某個語言現象是常見或罕見。統計派的方法偏重於對語料庫中，人們實際使用的語言現象的統計，統計方法為語料庫語言學研究的主要內容。

由於大多數的機器學習演算法，均以統計學為基礎，因此統計派就成為現在人工智慧中不可或缺的要素，而在人工智慧之下，其中一個子領域叫做「自然語言處理」，基於人工智慧與語言學，探討機器對語言的處理以及應用，並將這些研究實作成電腦能處理的格式。

有了這些理論基礎後，再加上開放原始碼的輔助工具，使得我們能發展出一套簡易的人機互動應用程式，並透過程式的執行結果，分析此程式的優缺點，以及自然語言處理的未來發展趨勢。

第二章、現代語言學發展

2.1 現代自然語言系統簡介

一個自然語言處理系統應包含至少三個模組：語言的產生、語言的解析，以及語義的了解[1]，如圖 1 所示。

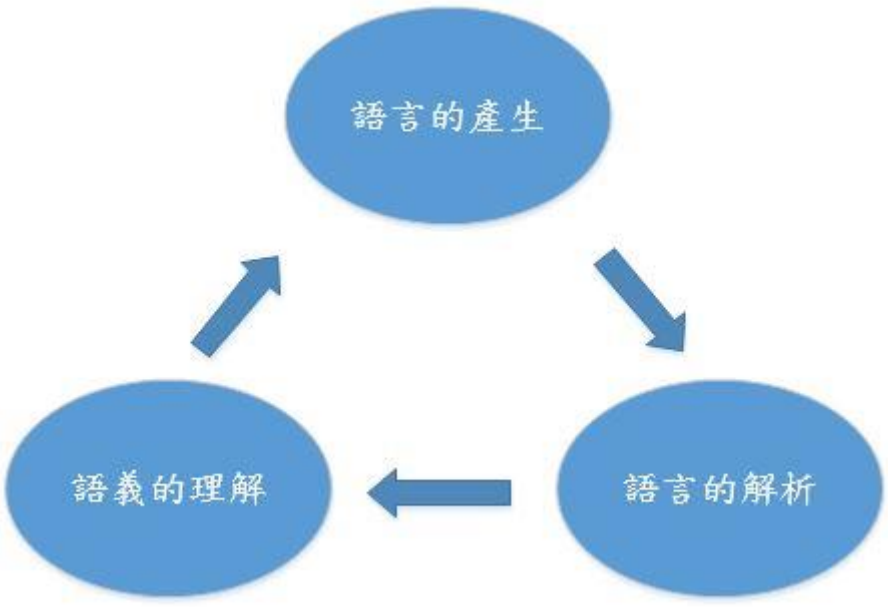

圖 1. 自然語言系統的整體邏輯

語言的作用為傳遞資訊，也是對人類文化發展相當重要的因素，而語言的產生，必須透過人類的大腦，而人類大腦負責產生語言的區塊為前腦的「布羅卡區」；語言產生後，接著就是傳遞，透過口語或是文字，傳達到接收者，而接收者收到後，必須要對此進行解析，並了解語義，而人類大腦對於語言的解析，以及語義理解，主要是透過後腦的「威爾尼克區」，此區塊除了能解析語言、理解語義，人耳接收到的語音也是透過這區塊處理。

當理解了語義後，接下來的思考、決策、資料的搜索等後續動作，則透過大腦的前額葉完成。

2.2 語句發展

人類從出生時，就不斷的學習各種事物，學說話也是學習的其中一項，兒童一開始說話時，大多只會有一個詞，在這階段所說的內容，通常為「單詞」，而這些詞彙大多是名詞或動詞，主要是表達生理需求，隨著年齡增長後，才會慢慢學習到表達看法用的形容詞。

兒童所說的單詞，實際上並不能稱做一個句子，原因在於這個單詞並沒有其他功能詞，例如時態、連接詞、冠詞等，因此又被稱作「電報式言語」（Telegraphic speech），此稱呼由 Moskowitz 所提出，意思是「由一個英文單字所組成的簡單句子」，在中文上，也就是一個詞彙所構成的句子。

而這類電報式言語，通常用來表達情緒，而成長後開始學習到的其他形容詞，若將形容詞也應用到說話上，此時的表達已經接近一個最基本的句子，若再加入語法，就可以說是一個完整的語言。

語言的傳遞，可以當作是資訊的傳遞，由人腦想出，產生字詞、透過語法組成句子，再透過媒介（聲音、文字等）傳遞至接收者，接收者收到後，透過大腦處理，解析語法、了解語意後，得知該句子所要表達的含意，這就是人類最基本的訊息傳遞方式，而此現象可透過資訊理論中常見的通訊系統模型來描述[2]，如圖 2 所示。

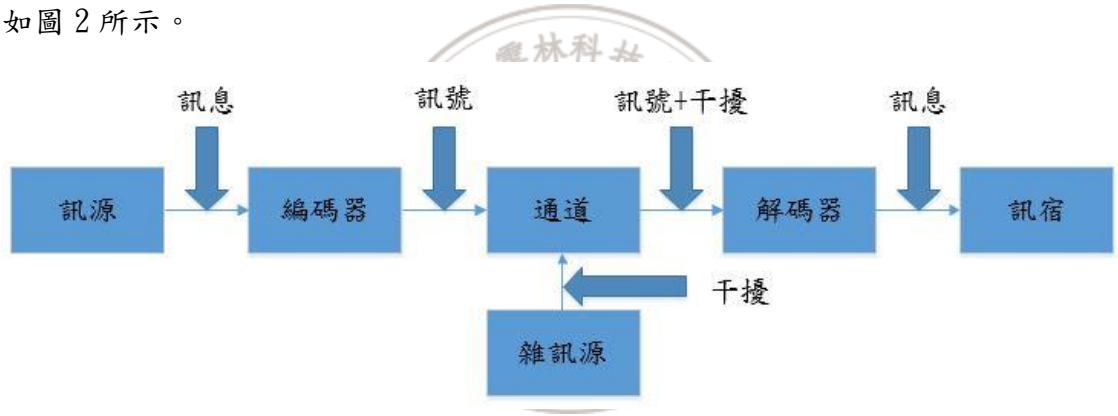

圖 2. 通訊系統模型

通訊系統主要分成五個部分。訊源用來產生訊息及訊息序列的源頭。在自然語言處理中，可以認為是人的大腦，產生想法，並轉換成詞彙及句子。編碼器將訊息變成適合在通道傳輸的物理量，此物理量可以視為一種能夠感知的存在，例如文字或是聲音。通道是指通訊系統把承載訊息的訊號，從傳送端送到接收端的媒介。可以是傳遞文字的紙張，或是傳遞聲音的空氣。解碼器用來將通道輸出，具有干擾的訊號進行轉換，變成訊宿能夠接受的訊息。大腦透過感官接收外部遞的視覺、聽覺、觸覺等訊號，透過神經網路進行處理，最後變成可以理解的內容。訊宿也就是接受訊息的人。

由通訊系統模型可得知，除了傳輸媒介以外，解編碼，也就是人類的大腦，在系統中扮演著相當重要的角色，表達一個事物，可以是一個動作，或是一個狀

態，而這兩種分別就代表動詞與形容詞，因此語言若要完整的表達一個事物，必須至少有兩個不同性質的詞彙，可以是名詞加上動詞，或是名詞加上形容詞。

前者代表了事物的「名稱」，也就是這個事物的特徵，後者則代表事物的「運動」或「狀態」，也就是動作或屬性的特徵。這種方式稱為「指稱—陳述」分化，只有這種方式，才能被稱作語言。

而這種基本形式，一般被認為是人類最初且最基本的語言，而其他語法結構都是後續發展而成型的。然而隨著時代變遷，語言發展逐漸成熟，句子的結構也變得越來越多元，去掉一些專有名詞後，仍有很多常用詞彙的詞性，必須依賴上下文來判定。

現今語法學界分為三派，一派主張從語言內部進行解釋，例如運用語法或語義，來限制句子成立的條件或是句式轉換的條件；另一派主張從語言外部進行解釋，例如使用認知心理來解釋某些語法現象、語法規則，還有一派認為，有些語法現象只能從內部進行解釋，有些只能從外部進行解釋，也有一些是與內外部均有關聯性，必須整合解釋。由此可知，在對語法現象與規則進行解釋時，需要依照實際情況進行分析。

在此基礎上，語言學界針對語法做了以下新的詮釋，不依賴於嚴格意義的形態變化，而借助於語序、虛詞等其他語法來標示語法關係和語法意義；列出語法研究和學習的最後目標，揭示語義的決定性、句法的強制性、用語的選定性，以及認知的解釋性。

2.3 語義研究

於 1938 年時，Charles W.Morris 提出符號學的三個分支：語形學、語義學和語用學[A]。而若要全面性分析一種語言，這三個方面都是不可或缺的。語法研究中的句法平面，是指對子句和句子進行句法分析。語義平面則是針對句子進行語用分析。換句話說，句法平面注重研究不同層級，語言符號之間的關係；語義平面注重於研究語言符號與所指事物之間的關係；語用平面注重於研究語言符號與使用者之間的關係。語言的三個平面[4]，如圖 4 所示。

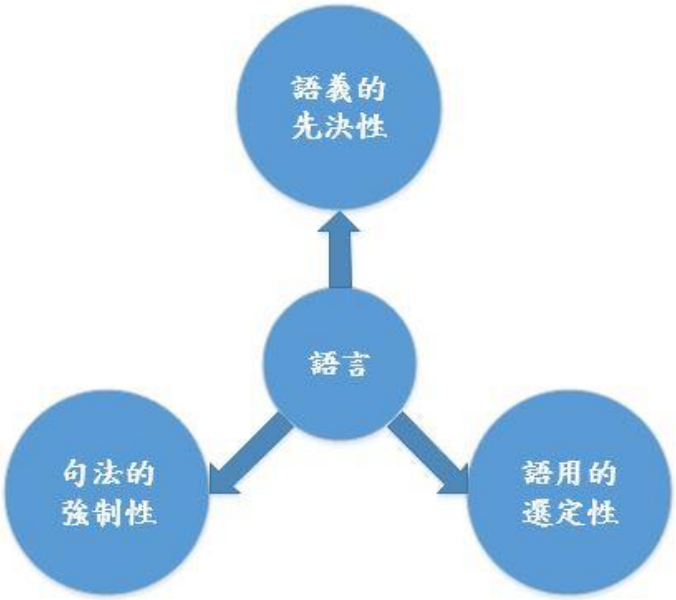

圖 3. 語言的三個平面

在語言的三個平面中，最核心的部分即為語義，語義對句子的產生和理解影響最大。語義，是指資訊的意義，在通訊領域中，可以視為編碼的意義，於自然語言處理中，就是字詞和句子的意義，字詞的含義，一般都是在長期使用下，人與人之間互相規範而形成的。

Charles J. Fillmore 於 1960 年代開始，提出了一系列的語法分析理論，此理論又被稱為「格語法」[A]。 格語法中的「格」，指的是句子中的名詞、代詞和動詞、形容詞等之間的及物性關係，格語法不僅出現在英文，幾乎所有語言中都存在此現象，而對於中文這類以語義為先的語言來說，更是重要。

格語法把句子的語義結構做為主要的研究目標，並對句子的語義平面看法為：句子是情態〈Modality〉與命題〈Proposition〉的組合，情態指的是言談者的信念、觀點、情緒、態度、立場、語氣、意圖、觀察角度等。命題則是由一個動詞，及與其相關的多個格組成。Fillmore 更認為，在字詞函數庫中，除了要標記每一個字詞在句法、語意及語音方面的特徵以外，還需標記它們「格」的特徵。

透過一系列語句、語義的研究與發展回顧，更能了解語法的規則性，以及多樣的語義關係，雖然句子中的語義關係仍相當複雜且多樣，但有了這些語言學者對語言的研究，讓機器理解人類語言的目標又更進一步。

第三章、自然語言處理工具

3.1 中文語言處理架構

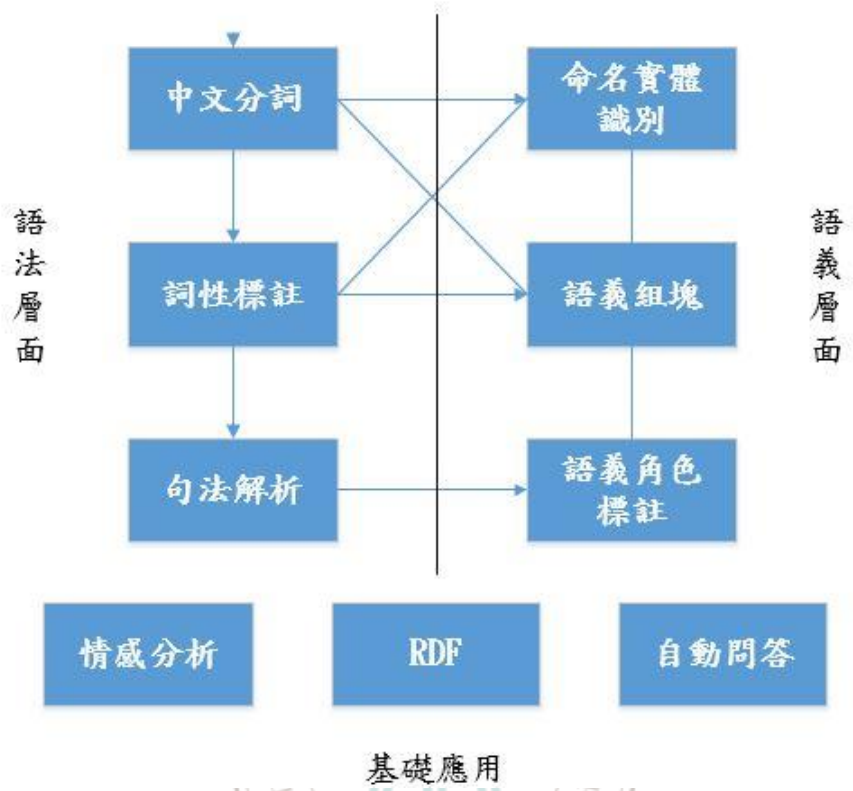

圖 4. 自然語言處理的一般架構

如圖 5 所示，為自然語言處理的一般架構[5]。右側偏重於語義層面，命名實體識別主要用來識別語料中的專有名詞，及未登錄在語料庫內的詞，例如人名、地名、公司機構名稱等。命名實體識別又受到中文分詞和詞性標註所影響。因此，若要有準確的命名實體識別，則也必須有準確的中文分詞和詞性標註技術。

語義組塊則用來確定一個以上字詞組成的子句結構，主要識別名詞性子句、動詞性子句、介詞子句等。要識別語義組塊的特徵，也必須包含基本的中文分詞及詞性標記。語義角色標記以句子中的謂語動詞為中心，預測句子中各個語法成分的語義特徵，是句子解析的最後一個環節。語義角色標記會受到句法解析及語義組塊的影響。

左側為語法層面，中文語言處理的第一步是中文分詞，因為中文不像英文是每個單字之間用空格區隔，且各單字本身就有意義，中文必須多個字才能形成一

個詞，形成詞後，這些中文字才有意義，因此在分析中文字詞之前，必須將一個句子切分成一個個單獨的詞彙，此過程稱作中文分詞〈Chinese Word Segmentation〉。若沒有執行此步驟，則無法對一個句子作進一步分析，由此可看出中文分詞是中文語言處理最重要的環節。

詞性標記〈Path-of-Speech Tagging 或 POS Tagging〉，又稱為詞類別標記。用來判斷每個字詞在一個句子中，分別扮演的語法角色。例如名詞用來表示人、事、物、地點的名稱；動詞用來表示動作或狀態變化；形容詞則用來描寫、修飾名詞，或表示性質、狀態、特徵或屬性。

要確立中文的詞性，困難點在於中文常用的詞彙，詞性大多不是固定的，換句話說，一個詞可能有多個詞性，且中文詞性不像英文有型態上的變化，例如在英文上，過去、未來、進行時的型態均會有不同的變化。因此，影響中文詞性標記精度的因素，在於要正確判斷常用詞的詞性。

句法解析，是根據既定的語法，自動推導出句子的語法結構，分析句子包含的語法單元，和這些語法單元之間的關係，並將句子轉化成一棵結構化的語法樹，句法分析為所有自然語言處理的核心模組。

3.2 常見的自然語言處理系統

哈工大語言技術平台〈Language Technology Platform，LTP〉，是由哈工大社會計算與資訊檢索研究中心研發的整合式中文語言處理系統，提供包含中文分詞、詞性標註、命名實體識別、依存句法分析、語義角色標註等豐富、高效、精準的自然語言處理技術。經過 11 年的持續研發及推廣，已成為世界最具影響力的中文處理基礎平台。目前 LTP 已被世界 500 多家研究機構和企業採用[6]。

圖 5. 哈工大語言技術平台線上示範

此語言平台除了提供全套的自然語言處理架構，如圖 6 所示，並可透過視覺

化的圖形輸出，讓使用者一目了然。

Stanford NLP [7]，為史丹佛自然語言處理團隊，是由史丹佛大學的教師及研究人員組成的團隊，此團隊致力於研究讓電腦理解人類語言，涵蓋範圍包含句子理解、機器翻譯，機率解析和標記，語法歸納、詞義消歧等。此系統提供句法解析、命名實體識別、詞性標記、分詞等。

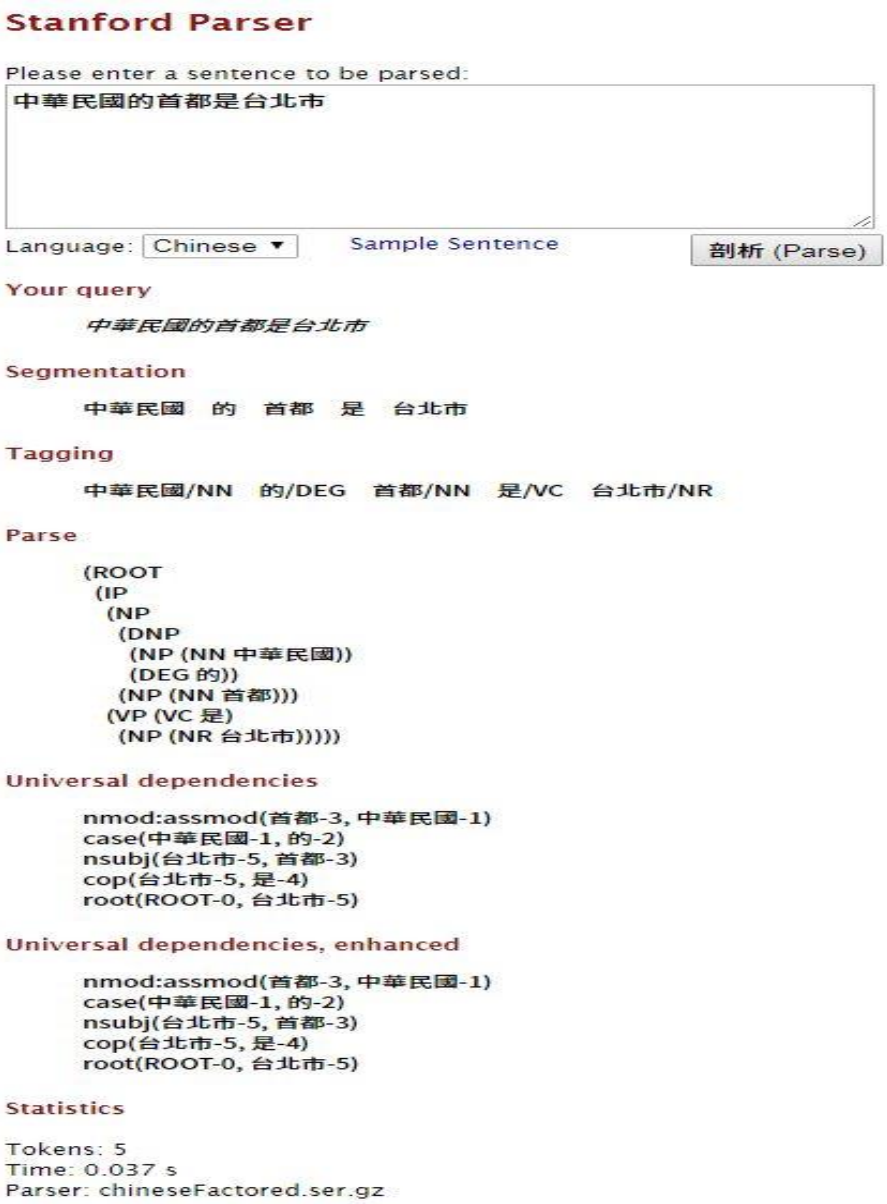

圖 6. Stanford 句法解析線上示範平台

NLTK〈Natural Language Toolkit〉[8]是基於 Python 程式語言的自然語言處理工具集，不但有方便使用的介面，更有超過五十種語料庫及詞彙資源，例如 Penn TreeBank、Penn PropBank、WordNet 等，還包含分類、分詞、標記、分析

和語義推導的基本程式架構，此工具為開放原始碼，無須支付任何費用即可使用。

結巴分詞模組 [9]基於張華平 NShort 中文分詞演算法，並以 Python 實現的中文分詞模組。NShort 為目前大規模中文分詞的主流演算法，具有演算法原理簡單、容易了解、易於訓練、高效率的大規模分詞、模型支援擴充、模型佔用資源低等優勢，並支援繁體字分詞，及以機率為基礎的使用者詞典。

結巴分詞又分為三種分詞模式，精確模式是將句子最精確切開，適用於文字分析；全模式是將句子中可以成為字詞的詞語掃描出來，但較不適合做實際應用；搜尋引擎模式則是基於精確模式上，對較長的字詞再次作切分，適用於搜尋引擎。

```
In [8]: runfile('C:/Users/user/Desktop/NLP/jieba_test.py', wdir='C:/Users/user/Desktop/NLP')
全模式
按照 | 深度 | 優 | 先 | 探索 | 的 | 策略 |  |  | 從 | 根 | 節 | 點 | 出 | 發 | 深度 | 探索 | 索解 | 解空 | 間 | 樹 |  |
精確模式
按照 | 深度 | 優先 | 探索 | 的 | 策略 | ， | 從 | 根節點 | 出發 | 深度 | 探索 | 解空 | 間樹 | 。
搜尋引擎模式
按照 | 深度 | 優先 | 探索 | 的 | 策略 | ， | 從 | 根節點 | 出發 | 深度 | 探索 | 解空 | 間樹 | 。
```

圖 7. 結巴分詞展示

NLPIR-ICTCLAS 漢語分詞系統是由張華平博士於 2002 年設計開發的一套中文分詞系統，該演算法思想來自 HMM，主要功能包含中文與英文分詞、詞性標註、命名實體識別、新詞識別、關鍵字提取，並支援使用者定義詞典，且支援多種作業系統，多種程式語言及平台[10]，如圖 8 所示，此系統也支援圖形化呈現處理結果。

圖 8. NLPIR-ICTCLAS 線上功能展示

HanLP 是以 ICTCLAS 為基礎，並以 Java 為開發語言的自然語言處理演算法函式庫，該函式庫除了提供豐富的語料資源，並實現了多個中文分詞方法，包含 N-最短路徑、CRF、感知器分詞演算法，且支援繁體中文，並可自行擴充詞典。HanLP 雖然基於 ICTCLAS，但使用更豐富的命名實體識別庫，除了人名、地名、組織機構名以外，更加入了音譯人名、日本人名等，並可進行拼音轉換、簡繁體字轉換等功能[11]。

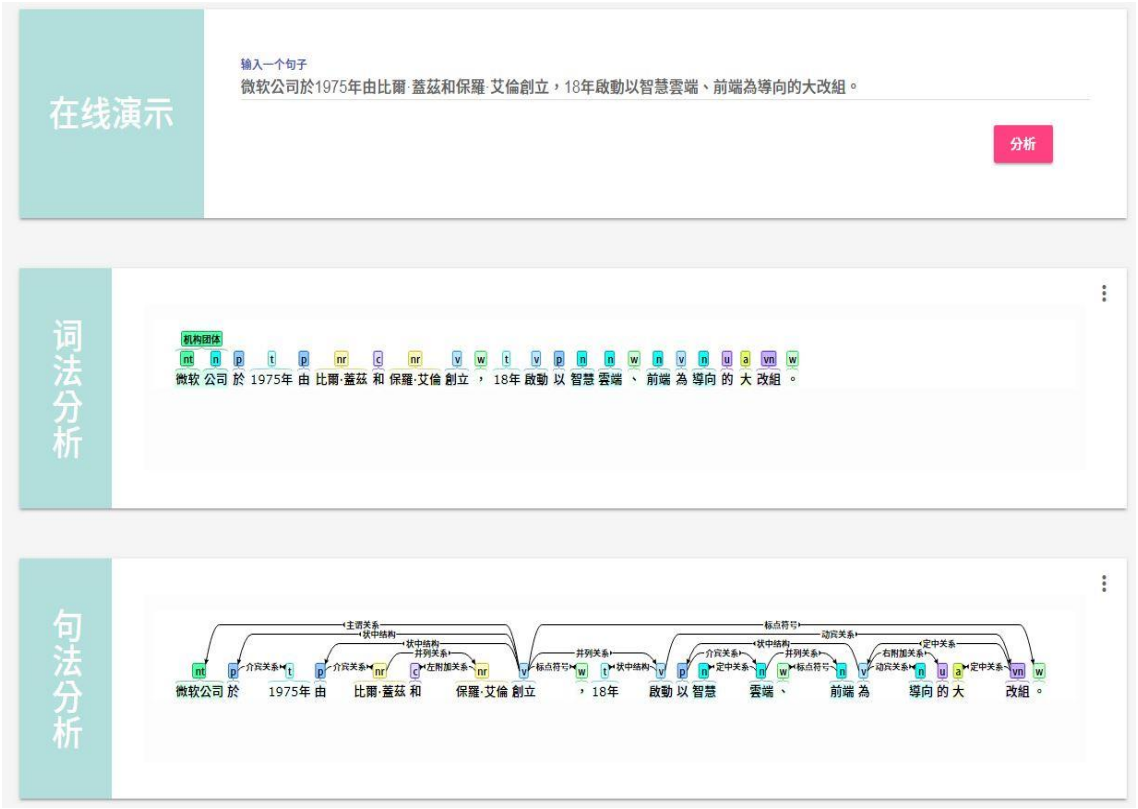

圖 9. HanLP 線上功能展示

第四章、自然語言常見演算法簡介

馬可夫鏈（Markov chain），是由俄國數學家安德烈·馬可夫所提出，馬可夫鏈又被稱作離散時間馬可夫鏈（Discrete-time Markov chain），原因在於時間和狀態均為離散的馬可夫過程，此過程為狀態空間中，從一個狀態轉換至另一個狀態的隨機過程，下一個狀態的機率分布只能由現在的狀態決定，與在它之前的所有事件無關，此性質又被稱作馬可夫性質[12]。

隱馬可夫模型（Hidden Markov Model），用來描述一個有隱藏狀態的馬可夫過程，近年來常被用於行動通訊及自然語言處理，在自然語言處理中更被用來中文分詞、詞性標記等重要的功能實現[13]，一個隱馬可夫模型中包含兩組狀態集合，以及三組機率集合，如下圖所示。

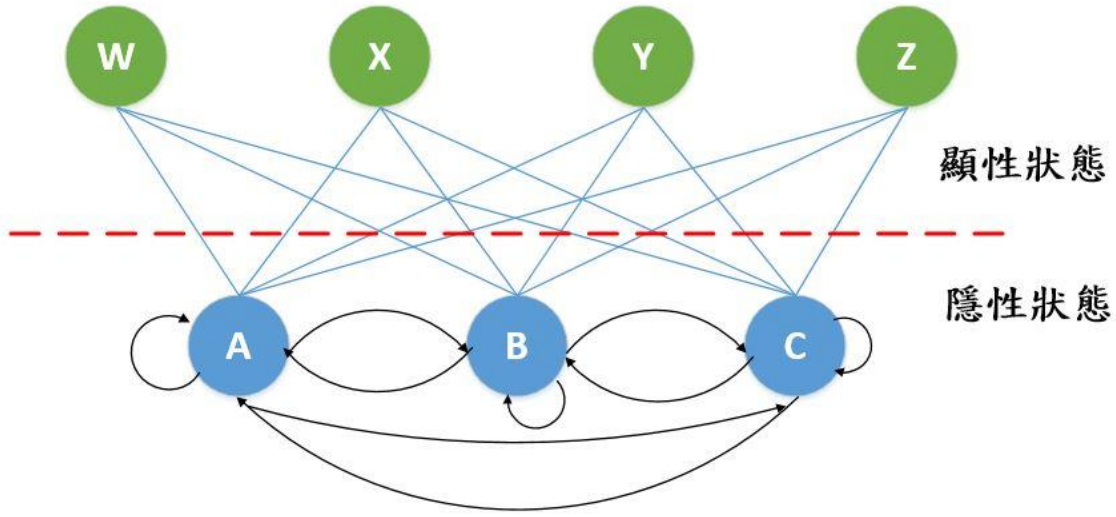

圖 10. 隱馬可夫模型架構

藍色圓圈代表隱性狀態，為系統的真實狀態，可以由一個馬可夫過程描述，而綠色圓圈代表顯性狀態，為此系統可觀察到的狀態，這兩組狀態就形成兩組集合。A、B、C 分別將連接至顯性狀態 W、X、Y、Z 之間的所有機率相加後，得到的值稱為「初始化機率向量」，各個隱狀態之間轉換的機率（黑線），可寫作一個「狀態傳輸矩陣」，而每個顯性狀態與隱性狀態之間的機率（藍線），也可寫成一個「混淆矩陣」，這三個就稱為三組機率集合。[14]

將此觀念套用於詞性標記與分析中，則可把所有常用的詞性整理成一個隱性狀態集合，若使用者自行建立語料庫，語料庫中包含詞彙、以及每個詞所對應的

詞性和詞頻，則此語料庫中的每個詞彙相當於一個顯性狀態集合，而語料庫中的詞性相當於一個混淆矩陣，詞頻則相當於此混淆矩陣內的值，而隱性狀態之間的轉換機率，則透過此語料庫進行學習，逐步完成詞性標記，此方法也是結巴分詞系統中，對於自定義詞典的處理方式。

第五章、實際應用與結果分析

此處以一個聊天室機器人應用為例，舉出此應用的執行過程，並一一實現。情境設定於使用者透過此聊天室詢問氣象資訊，接著此機器人會做對應回答，使用者的輸入通常是一個句子，電腦遇到句子時，勢必得先進行中文分詞處理，將一個句子切分成多個詞彙，再依照詞彙判斷回應主題，例如某個地點的天氣狀況，則此機器人會自動從氣象局抓取對應資料，並經過預處理後，回傳給使用者。

一般氣象資訊會依照地名做區隔，因此在字詞庫勢必得加入地區相關詞彙，以便做句子分析，在氣象資訊中，較重要的字詞庫為地名、天氣相關專有名詞等分類。

整體運作流程為，一開始由使用者以句子詢問，接著進行中文分詞處理，判斷是否有氣象相關關鍵字，若關聯度夠高，則依照詞意和語意做回答句的組成，最後回應使用者。

本應用所使用的分詞系統為結巴分詞，使用此分詞系統的原因在於，簡單易上手，分詞速度快，可支援自建字詞庫及繁體中文分詞、並設定字詞權重，適合針對特定用途做調整，系統流程圖如下。

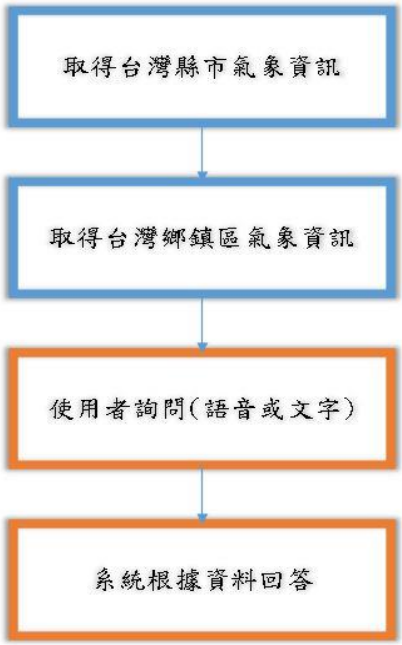

圖 11. 系統流程圖

取得台灣縣市氣象資訊的方式，是透過氣象局所提供的台灣「三十六小時天氣預報」，檔案格式為 xml，因此取得資料後，必須先做預處理，在擷取與預處理方面，本次使用了 python 的第三方套件「Beautiful Soup」，如下圖所示敘述，此套件基於知名 xml、html 解析套件 lxml 及 html5lib 上，一般用在網路爬蟲，抓取網頁資訊，並支援資料預處理，但此套件不僅能支援 HTML，也支援 XML 格式，使得開發者在取得資料時，能更方便進行預處理。

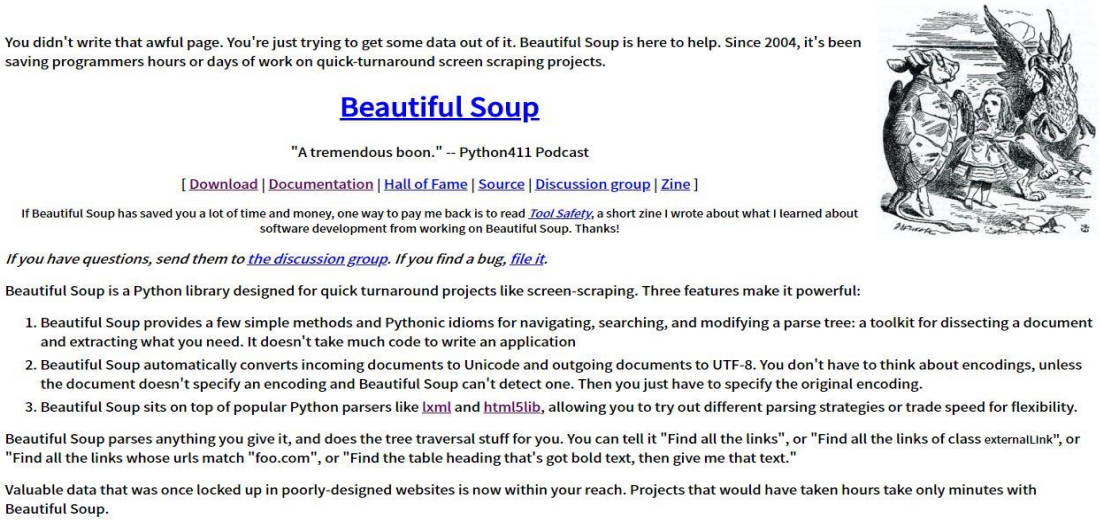

圖 12. Beautiful Soup 官方頁面

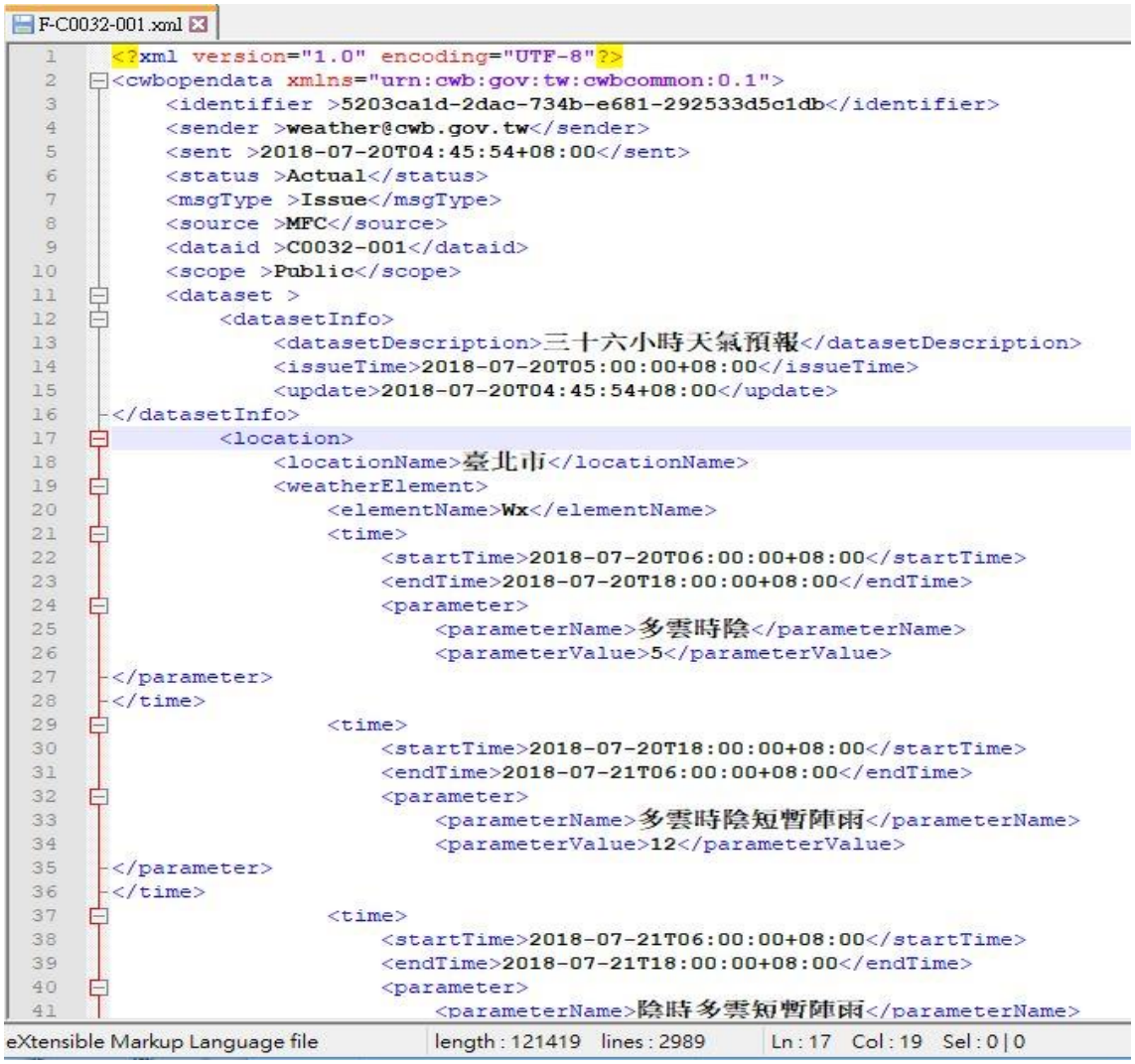

圖 13. 取得的氣象資訊(xml 格式)

由氣象局取得資料後，使用 get_text 函式將 xml 中的所有標籤去除，將各縣市資料做分類儲存，待需要時即可直接取用。

```
#取得氣象資料，並取出文字部分
f = urllib.request.urlopen("http://opendata.cwb.gov.tw/opendataapi?dataid=F-C0032-001
soup = BeautifulSoup(f, "lxml")
x = soup.get_text()
```

圖 14. 取出 xml 中純文字的方法

取得台灣鄉鎮區氣象資訊的方式為使用氣象局所提供的台灣鄉鎮區「未來一週氣象資訊」，但與台灣縣市氣象資訊不同的地方，在於向氣象局取得的資料為一個 zip 壓縮檔，使用時必須先做解壓縮，才能取得內部的 xml 資料，且鄉鎮區氣象資訊是將每個縣市分開存檔，因此必須一次處理多筆 xml 檔案。

在此處所使用的解壓縮套件為 python 內建的 zipfile 套件，並透過

extract 函式，取出壓縮檔內的檔案，即可完成解壓縮動作，再使用 get_text 函式去除 xml 標籤，做分類儲存，待需要時取用。

```
#下載解壓縮後的鄉鎮氣象資訊儲存資料夾
path = "C:/Users/user/Desktop/py_data/country_data/"
f = urllib.request.urlopen("http://opendata.cwb.gov.tw/opendataapi?dataid=F-D0047-093
zipcontent= f.read()

with open("F-D0047-093.zip", 'wb') as f:
    f.write(zipcontent)
print ("下載完成!")

print ("資料解壓縮...")

with zipfile.ZipFile(open('F-D0047-093.zip', 'rb')) as f:
    f.extract("63_Week24_CH.xml", path)
    f.extract("64_Week24_CH.xml", path)
    f.extract("65_Week24_CH.xml", path)
    f.extract("66_Week24_CH.xml", path)
    f.extract("67_Week24_CH.xml", path)
    f.extract("68_Week24_CH.xml", path)
    f.extract("09007_Week24_CH.xml", path)
    f.extract("09020_Week24_CH.xml", path)
    f.extract("10002_Week24_CH.xml", path)
    f.extract("10004_Week24_CH.xml", path)
    f.extract("10005_Week24_CH.xml", path)
    f.extract("10007_Week24_CH.xml", path)
    f.extract("10008_Week24_CH.xml", path)
    f.extract("10009_Week24_CH.xml", path)
    f.extract("10010_Week24_CH.xml", path)
    f.extract("10013_Week24_CH.xml", path)
    f.extract("10014_Week24_CH.xml", path)
    f.extract("10015_Week24_CH.xml", path)
    f.extract("10016_Week24_CH.xml", path)
    f.extract("10017_Week24_CH.xml", path)
    f.extract("10018_Week24_CH.xml", path)
    f.extract("10020_Week24_CH.xml", path)
print ("解壓縮完成!")
```

圖 15. 解壓縮下載後的 zip 檔案

由於使用者詢問時，必定會輸入地名，而結巴分詞的主要開發者均來自中國大陸，因此對於台灣的用詞涵蓋量較低，在此會使用結巴分詞的功能，匯入自建的字詞庫，而字詞庫建立方式為，透過取得氣象資料內所提供的地名建立，除了原本提供的名稱以外，還會建立一串去掉「縣、市、鄉、鎮、區」字的字詞，主要是考量使用者輸入時，不一定會輸入全名，而此狀況會影響分詞結果，導致輸出錯誤，將這些字詞加入後，可以更精確的做字詞切分。

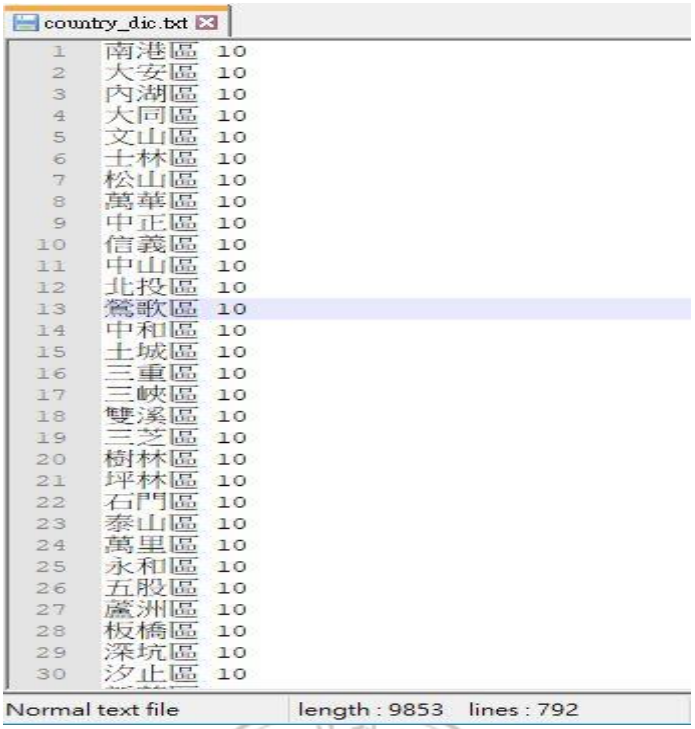

圖 16. 字詞庫格式

字詞庫的建立方式為：字詞 權重 詞性，詞性為選填項目，而權重可依照所需的分詞結果做調整，本實驗將權重都設定為比內建字詞庫還要高的 10，目的是將這些台灣地名凸顯出來，讓結巴分詞能正確的做切分。

使用者詢問時若透過語音，則必須讓系統能接收語音，並轉成文字，使分詞系統能讀取並進行分詞，因此使用 Google 所提供的語音辨識套件，也就是 Google 語音轉文字功能。

```
54 r=sr.Recognizer()
55
56 with sr.Microphone() as source:
57     print("麥克風調整中")
58     r.adjust_for_ambient_noise(source, duration=5)
59     print("想知道哪裡的天氣狀況?")
60     audio=r.listen(source)
61
62 try:
63     print("讓我來告訴你")
64     sst = r.recognize_google(audio, language="zh-TW")
65     #print(sst)
66 except sr.UnknownValueError:
67     print("我聽不懂你所說的話，請再試一次")
68 except sr.RequestError as e:
69     print("No response from Google Speech Recognition service: {0}".format(e))
```

圖 17. Google 語音轉文字使用方法

一開始先設定一變數 r，目的是用來做語音處理，接著先做麥克風調整，

再來就是接收語音訊號，接收後透過 Google 語音轉文字系統處理，若處理成功，則會輸出所說的文字，若因為網路或是 Google 伺服器問題，則會跳出錯誤訊息，要求使用者再次輸入語音訊號。當語音訊號轉成文字後，即可針對句子做切分，此處使用結巴分詞的一般模式進行分詞，並將切分後的詞存入一個陣列中，接著做鄉鎮區及縣市的比對，若有比對到資料，則輸出對應的氣象資訊。

```
seg_list = jieba.cut(sst, cut_all=False)
test = "/".join(seg_list)
sp_list = test.split('/')
```

圖 18. 結巴分詞使用方式

```
for s in city_data: #掃描city_data
    for a in sp_list: #掃描輸入的句子(已經分詞處理)
        if (str(s).find(a)) != -1: #若句子中的縣市名(全名或是片段)有出現在city_data內
            city_name = s #取得於city_data內的縣市名
            city_index = city_data.index(s) #取得縣市名於city_data的index
            print (city_name)
            print (city_data[city_index+3])
            print ('溫度 : ' + city_data[city_index+4] + '~' + city_data[city_index+5])
            print ('降雨機率 : ' + city_data[city_index+6] + '%')
```

圖 19. 字詞比對方法

在比對之前，還會有一個判斷，若使用的輸入的是「台」，則會轉換成「臺」，主要是因為氣象局提供的資料中，地名是使用「臺」字，避免判斷錯誤，才做轉換動作。

```
#若輸入的地名有'台'，則將'台'轉成'臺'
for s in sp_list:
    if (s.find('台')) != -1:
        tai_index = sp_list.index(s)
        tmp = list(s)
        tmp[0] = '臺'
        s = ''.join(tmp)
        sp_list[tai_index] = s
```

圖 20. 「台」與「臺」的轉換

由於結巴分詞系統已經經過多年的發展，已經有相當優良的分詞精確度，但對於繁體中文的支援度仍不夠精確，因此在實作時必須將系統所使用的關鍵詞彙整理成詞典，使此系統能進行學習，達到更精確的切分。在使用者輸入部分，由於是使用 Google 所提供的語音轉文字系統，因此須將語音傳送至 Google 伺服器

處理，再轉成文字傳回使用者端，若伺服器或是網路品質不穩定時，會有程式異常停止的狀況發生，因此提供第二種方式輸入，也就是純文字輸入，來避免此狀況發生，影響使用者體驗。

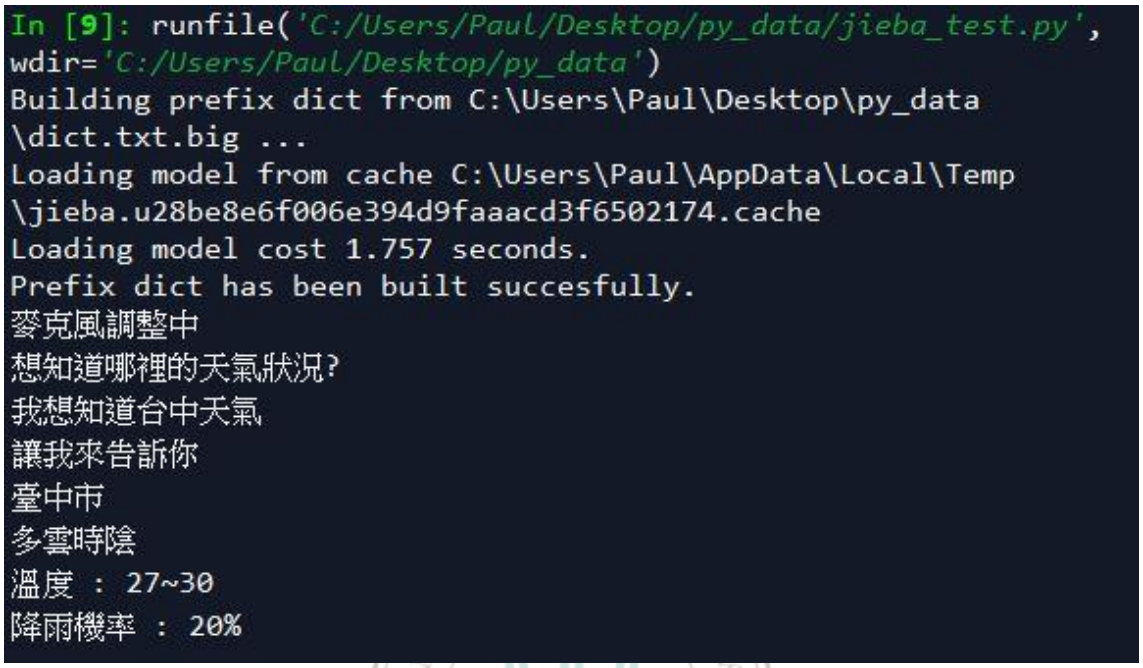

圖 21. 實際執行結果

此系統特色在於能取得使用者輸入內容的關鍵詞彙，並與目前取得資料比對，回傳所需結果，但若將自定義的字詞庫移除後，會有部分地名無法被切分，導致回傳資料出現錯誤，因此在定義字詞庫上，需特別針對應用方向設計，若應用層面擴大時，勢必要增加字詞庫內容，由此可得知，字詞庫的設計與擴充，在自然語言處理應用上相當重要。

第六章、結論

此論文從現代語言學回顧，接著介紹這些學門如何應用在自然語言處理上，並以實際應用呈現自然語言處理的核心，也就是中文分詞和語意分析，前者是用來將中文句子進行切分，方便後須取出關鍵字詞使用，後者則是理解此句子所要表達的真實意義。由於語料庫在自然語言處理中，佔有相當大的重要性，若語料庫不夠龐大，勢必會影響處理結果，因此，在此應用中所選用的結巴分詞系統，可讓開發人員加入想要的字詞，並依序分配權重，使開發時更有彈性。

雖然此系統僅在於單一句子的判讀與回答，但自然語言處理系統的未來發展展，勢必會朝向理解中長篇幅的文章，並判讀含義，現在已經有不少系統實現上下文判讀，若能善加利用，勢必在面對較長的句子或段落時，更能準確的判讀出正確語意，並做出正確的回應。

參考文獻

[1] 鄭捷, 自然語言處理：用人工智慧看懂中文,page 1-7, 2018, 佳魁資訊

[2] 馮冠, 倪寶童, 電腦網路標準教程, 清華大學出版社, 2010

[3] 同[1],page 2-32

[4] 同[1],page 2-43

[5] 同[1],page 1-9

[6] 語言雲, 語言技術平台, 哈工大社會計算與訊息檢索研究中心
https://www.ltp-cloud.com/

[7] The Stanford Natural Language Processing Group
https://nlp.stanford.edu/

[8] Natural Language Toolkit
https://www.nltk.org/

[9] 結巴中文分詞
https://github.com/fxsjy/jieba

[10] NLPIR-ICTCLAS 漢語分詞系統
http://ictclas.nlpir.org/

[11] 漢語言處理包
https://github.com/hankcs/HanLP

[12] Norris, James R, Markov chains, Cambridge University Press, 1998.

[13] Lawrence R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings of the IEEE, pp.257-286, February 1989.

[14] 演算法筆記- Hidden Markov Model
http://www.csie.ntnu.edu.tw/~u91029/HiddenMarkovModel.html

附錄

取得台灣鄉鎮區氣象資訊

```
1. import urllib
2. from bs4 import BeautifulSoup
3. import linecache
4. import zipfile
5. import glob
6. import os
7.
8. #下載解壓縮後的鄉鎮氣象資訊儲存資料夾
9. path = "C:/Users/Paul/Desktop/py_data/country_data/"
10. f =
    urllib.request.urlopen("http://opendata.cwb.gov.tw/opendataapi?data
    id=F-D0047-093&authorizationkey=CWB-68B0AD71-C5A8-4BED-B562-
    62AD58E7D598")
11. zipcontent= f.read()
12.
13. with open("F-D0047-093.zip", 'wb') as f:
14.     f.write(zipcontent)
15. print ("下載完成!")
16.
17. print ("資料解壓縮...")
18.
19. with zipfile.ZipFile(open('F-D0047-093.zip', 'rb')) as f:
20.     f.extract("63_Week24_CH.xml", path)
21.     f.extract("64_Week24_CH.xml", path)
22.     f.extract("65_Week24_CH.xml", path)
23.     f.extract("66_Week24_CH.xml", path)
24.     f.extract("67_Week24_CH.xml", path)
25.     f.extract("68_Week24_CH.xml", path)
26.     f.extract("09007_Week24_CH.xml", path)
27.     f.extract("09020_Week24_CH.xml", path)
28.     f.extract("10002_Week24_CH.xml", path)
29.     f.extract("10004_Week24_CH.xml", path)
30.     f.extract("10005_Week24_CH.xml", path)
31.     f.extract("10007_Week24_CH.xml", path)
32.     f.extract("10008_Week24_CH.xml", path)
33.     f.extract("10009_Week24_CH.xml", path)
34.     f.extract("10010_Week24_CH.xml", path)
35.     f.extract("10013_Week24_CH.xml", path)
36.     f.extract("10014_Week24_CH.xml", path)
37.     f.extract("10015_Week24_CH.xml", path)
38.     f.extract("10016_Week24_CH.xml", path)
39.     f.extract("10017_Week24_CH.xml", path)
40.     f.extract("10018_Week24_CH.xml", path)
41.     f.extract("10020_Week24_CH.xml", path)
42. print ("解壓縮完成!")
43.
44. #建立縣市資料 list
45. city_list = ['臺北市', '新北市', '桃園市', '臺中市', '臺南市', '高雄市',
46.              '基隆市', '新竹縣', '新竹市', '苗栗縣', '彰化縣', '南投縣',
47.              '雲林縣', '嘉義縣', '嘉義市', '屏東縣', '宜蘭縣', '花蓮縣',
48.              '臺東縣', '澎湖縣', '金門縣', '連江縣']
49. z = list() #將檔案讀出後，暫存用 list
50.
```

```
51. print ("檔名修改中...")
52. for fname in os.listdir(path): #讀取鄉鎮氣象資訊資料夾內所有檔案
53.     xmlfile = open(path + fname, 'r', encoding='utf-8') # fname 為檔案名稱，為.xml 格式
54.     soup = BeautifulSoup(xmlfile, "lxml")
55.     raw = soup.prettify() # raw 用來儲存未處理的完整 xml 內容 (包含所有標籤)
56.     x = soup.get_text() # x 用來儲存去除 xml 標籤後的內容
57.     fw = open('C:/Users/Paul/Desktop/py_data/rename_pre.txt', 'w', encoding="utf-8")  #建檔寫入資料，暫存用
58.     fw.write(x)
59.     fw.close()
60.
61.     fr = open('C:/Users/Paul/Desktop/py_data/rename_pre.txt', 'r', encoding="utf-8")  #讀出暫存資料，並存到 z 中
62.     for line in fr:
63.         if (line != '\n'):
64.             line = ''.join(line.split()) #清除每一行結尾的 CRLF(換行)
65.             z.append(line)  #將檔案內容一行一行讀出，並存到 z
66.     fr.close()
67.     # 縣市資訊 與 list 比對
68.      # 此處會存兩份檔案 TXT 用來計算單一縣市的鄉鎮數量 (根據<location>標籤計算)
69.      # 另一份為 XML 用來做資料預處理
70.     if (z[14] in city_list): #若縣市有在縣市資料 list 內，將檔案重新命名
71.         rename = city_list.index(z[14]) + 1 #檔名為 list 中的位置+1 (例如臺北市位於位置 0，則檔名為 1_t.txt & 1_x.xml)
72.         f_txt = open('C:/Users/Paul/Desktop/py_data/country_data_pre/' + str(rename) + "_t.txt", 'w', encoding="utf-8")
73.         f_txt.write(raw) #將 raw 存成 txt
74.         f_txt.close()
75.
76.         f_xml = open('C:/Users/Paul/Desktop/py_data/country_data_pre/' + str(rename) + "_x.xml", 'w', encoding="utf-8")
77.         f_xml.write(raw) #將 raw 存成 xml
78.         f_xml.close()
79.     z.clear() #清除暫存 list z
80. print ("檔名修改完成!")
81.
82. # 計算 <location> 數量 & 取鄉鎮、起始結束時間及天氣概述
83. z_clr = list() #新暫存陣列 儲存去除換行/空格後的字串
84. country_data = list()
85. for i in range(22):
86.     # 計算 <location> 數量
87.     count_tag = open('C:/Users/Paul/Desktop/py_data/country_data_pre/' + str(i+1) + "_t.txt", 'r', encoding="utf-8")
88.     for line in count_tag:
89.         if (line != '\n'):
90.             line = ''.join(line.split())
91.             z.append(line)
92.     count_tag.close()
93.
```

```
94.     tag = '<location>'
95.     get = 0
96.     for j in range(len(z)):
97.         if (tag == z[j]):
98.             get = get + 1
99.     z.clear() #清除暫存 list z
100.
101.        # 取鄉鎮、起始結束時間及天氣概述
102.        xml_pre =
    open('C:/Users/Paul/Desktop/py_data/country_data_pre/' + str(i+1) +
    "_x.xml", 'r', encoding="utf-8")
103.        new_soup = BeautifulSoup(xml_pre, "lxml")
104.        new_x = new_soup.get_text()
105.        fw_new = open('test_bud.txt', 'w', encoding='utf-8') #將 XML
    內容存入暫存檔 (test_bud.txt)
106.        fw_new.write(new_x)
107.        fw_new.close()
108.
109.        fr_new = open('test_bud.txt', 'r', encoding='utf-8') #讀出暫
    存檔內容，並去除 CRLF
110.        for line in fr_new:
111.            if (line != '\n'):
112.                line = ''.join(line.split()) #清除每一行結尾的 CRLF(換
    行)
113.                z.append(line) #將檔案內容一行一行讀出，並存到 z
114.        fr_new.close()
115.
116.        for k in range(len(z)): #只取原暫存陣列 z 中，非空白值，並存入新暫
    存陣列(z_clr)
117.            if (z[k] != ''):
118.                z_clr.append(z[k])
119.
120.        fin = open('test_bud.txt', 'w', encoding='utf-8') #將處理後資
    料存回暫存檔
121.        for h in range(len(z_clr)):
122.            fin.write(z_clr[h] + '\n')
123.        fin.close()
124.
125.        country = 16
126.        for L in range(get):
127.            text = linecache.getline('test_bud.txt', country) #取鄉鎮
128.            text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
129.            country_data.append(text) #將取得資料放入 list 暫存
130.            for m in range(7):
131.                text = linecache.getline('test_bud.txt',
    country+465) #START
132.                text = ''.join(text.split())
133.                country_data.append(text)
134.                text = linecache.getline('test_bud.txt',
    country+465+1) #END
135.                text = ''.join(text.split())
136.                country_data.append(text)
137.                text = linecache.getline('test_bud.txt',
    country+465+2) #DATA
138.                text = ''.join(text.split())
139.                country_data.append(text)
```

```
140.            country = country + 493
141.        the_final =
    open('C:/Users/Paul/Desktop/py_data/country_data_final/' + str(i+1)
    + "_country.txt", 'w', encoding="utf-8")
142.        for n in range(len(country_data)):
143.            the_final.write(country_data[n] + '\n')
144.        the_final.close()
145.        z.clear() #清除暫存 list z
146.        z_clr.clear() #清除暫存 list z_clr
147.        country_data.clear() #清空 country_data 暫存資料
148.        linecache.clearcache() #清除 linecache 的快取資料
```

取得台灣縣市氣象資訊

```
1. # -*- coding:utf-8 -*-
2. import urllib
3. from bs4 import BeautifulSoup
4. import linecache
5.
6. z = list()
7. city_data = list()
8. file_count = 1
9. #path = 'C:/Users/Paul/Desktop/py_data/city_data_final/' (HOME)
10. path = 'C:/Users/Paul/Desktop/py_data/city_data_final/'
11.
12. #取得氣象資料，並取出文字部分
13. f =
    urllib.request.urlopen("http://opendata.cwb.gov.tw/opendataapi?data
    id=F-C0032-001&authorizationkey=CWB-68B0AD71-C5A8-4BED-B562-
    62AD58E7D598")
14. soup = BeautifulSoup(f, "lxml")
15. x = soup.get_text()
16.
17. fw = open('data.txt', 'w', encoding="utf-8")  #建檔寫入資料，檔案編碼為
    utf-8
18. fw.write(x)
19. fw.close()
20.
21. fr = open('data.txt', 'r', encoding="utf-8") #讀檔取出資料
22. #清除空白行
23. for line in fr:
24.     if (line != '\n'):
25.         line = ''.join(line.split()) #清除每一行結尾的 CRLF(換行)
26.         z.append(line)  #將檔案內容一行一行讀出，並存到 z
27. fr.close()
28.
29. f = open('data_pre.txt', 'w', encoding="utf-8")
30. for i in range(len(z)):
31.     f.write(z[i] + '\n')
32. f.close()
33.
34. city = 12
35. for i in range(22):
36.     file_name = str(file_count) + "_city.txt"
```

```
37.     f_city = open(path + file_name, 'w', encoding="utf-8")
38.     text = linecache.getline('data_pre.txt', city)  #取縣市
39.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
40.     city_data.append(text)
41.     text = linecache.getline('data_pre.txt', city+2)  #取起始時間範圍
    1
42.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
43.     city_data.append(text)
44.     text = linecache.getline('data_pre.txt', city+3)  #取結束時間範圍
    1
45.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
46.     city_data.append(text)
47.     text = linecache.getline('data_pre.txt', city+4)    #取天氣狀況 1
48.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
49.     city_data.append(text)
50.     text = linecache.getline('data_pre.txt', city+30)   #取最低溫 1
51.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
52.     city_data.append(text)
53.     text = linecache.getline('data_pre.txt', city+17)   #取最高溫 1
54.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
55.     city_data.append(text)
56.     text = linecache.getline('data_pre.txt', city+53)   #取降雨機率 1
57.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
58.     city_data.append(text)
59.     text = linecache.getline('data_pre.txt', city+6)  #取起始時間範圍
    2
60.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
61.     city_data.append(text)
62.     text = linecache.getline('data_pre.txt', city+7)  #取結束時間範圍
    2
63.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
64.     city_data.append(text)
65.     text = linecache.getline('data_pre.txt', city+8)    #取天氣狀況 2
66.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
67.     city_data.append(text)
68.     text = linecache.getline('data_pre.txt', city+34)   #取最低溫 2
69.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
70.     city_data.append(text)
71.     text = linecache.getline('data_pre.txt', city+21)   #取最高溫 2
72.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
73.     city_data.append(text)
74.     text = linecache.getline('data_pre.txt', city+57)   #取降雨機率 2
75.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
76.     city_data.append(text)
77.     text = linecache.getline('data_pre.txt', city+10)  #取起始時間範圍
    3
78.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
79.     city_data.append(text)
80.     text = linecache.getline('data_pre.txt', city+11)  #取結束時間範圍
    3
81.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
82.     city_data.append(text)
83.     text = linecache.getline('data_pre.txt', city+12)    #取天氣狀況 3
84.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
```

```
85.     city_data.append(text)
86.     text = linecache.getline('data_pre.txt', city+38)   #取最低溫 3
87.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
88.     city_data.append(text)
89.     text = linecache.getline('data_pre.txt', city+25)   #取最高溫 3
90.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
91.     city_data.append(text)
92.     text = linecache.getline('data_pre.txt', city+61)   #取降雨機率 3
93.     text = ''.join(text.split())  #清除每一行結尾的 CRLF(換行)
94.     city_data.append(text)
95.     for a in range(len(city_data)):
96.         f_city.write(city_data[a] + '\n')
97.     f_city.close()
98.     city = city + 63 #63 為各縣市每個參數間隔行數，加上此數可變換縣市資料
99.     file_count = file_count + 1
100.        city_data.clear() #清空暫存資料
101.
```

使用者詢問與系統答覆

```
1. import speech_recognition as sr
2. import jieba
3. jieba.set_dictionary('dict.txt.big') #使用繁體字典
4. jieba.load_userdict("country_dic.txt") #匯入自建字詞庫
5.
6. city_data = list()
7. country_data = list()
8.
9. # 取得所有縣市氣象資料
10. path_ct = "C:/Users/Paul/Desktop/py_data/city_data_final/"
11. for j in range(22):
12.     y = j + 1
13.     get_city = open(path_ct + str(y) + '_city.txt', 'r',
    encoding='utf-8')
14.     for line_ct in get_city:
15.         if (line_ct != '\n'):
16.             line_ct = ''.join(line_ct.split()) #清除每一行結尾的 CRLF(換
    行)
17.             city_data.append(line_ct)
18.     get_city.close()
19.
20. # 取得所有鄉鎮區氣象資料
21. path = "C:/Users/Paul/Desktop/py_data/country_data_final/"
22. y = 0
23. country = 1
24. for i in range(22):
25.     y = i + 1
26.     get_country = open(path + str(y) + '_country.txt', 'r',
    encoding='utf-8')
27.     for line in get_country:
28.         if (line != '\n'):
29.             line = ''.join(line.split()) #清除每一行結尾的 CRLF(換行)
```

```
30.             country_data.append(line)  #將檔案內容一行一行讀出，並存到
    country_data
31.     get_country.close()
32.
33. r=sr.Recognizer()
34.
35. with sr.Microphone() as source:
36.     print("麥克風調整中")
37.     r.adjust_for_ambient_noise(source, duration=5)
38.     print("想知道哪裡的天氣狀況?")
39.     audio=r.listen(source)
40.
41. try:
42.     sst = r.recognize_google(audio, language="zh-TW")
43.     print(sst)
44.     print("讓我來告訴你")
45. except sr.UnknownValueError:
46.     print("我聽不懂你所說的話，請再試一次")
47. except sr.RequestError as e:
48.     print("No response from Google Speech Recognition service:
    {0}".format(e))
49.
50. seg_list = jieba.cut(sst, cut_all=False)
51. test = "/".join(seg_list)
52. sp_list = test.split('/')
53.
54. #若輸入的地名有'台'，則將'台'轉成'臺'
55. for s in sp_list:
56.     if (s.find('台')) != -1:
57.         tai_index = sp_list.index(s)
58.         tmp = list(s)
59.         tmp[0] = '臺'
60.         s = ''.join(tmp)
61.         sp_list[tai_index] = s
62.
63. s = ''
64. a = ''
65. # 首先判斷句子內是否有縣市名，若無輸出，代表此地名不是縣市，往下一步驟
66. for s in city_data: #掃描 city_data
67.     for a in sp_list: #掃描輸入的句子(已經分詞處理)
68.         if (str(s).find(a)) != -1: #若句子中的縣市名(全名或是片段)有出現在
    city_data 內
69.             city_name = s #取得於 city_data 內的縣市名
70.             city_index = city_data.index(s) #取得縣市名於 city_data 的
    index
71.             print (city_name)
72.             print (city_data[city_index+3])
73.             print ('溫度 : ' + city_data[city_index+4] + '~' +
    city_data[city_index+5])
74.             print ('降雨機率 : ' + city_data[city_index+6] + '%')
75. s = ''
76. a = ''
77. # 上面判斷式無輸出，則判斷句子內是否有鄉鎮區名
78. for s in country_data:
79.     for a in sp_list:
```

```
80.         if (str(s).find(a)) != -1: #若句子中的鄉鎮區名(全名或是片段)有出現
    在 country_data 內
81.             country_name = s #取得於 country_data 內的鄉鎮區名
82.             country_index = country_data.index(s) #取得鄉鎮區名於
    country_data 的 index
83.             print (country_name)
84.           print (country_data[country_index+3])
```