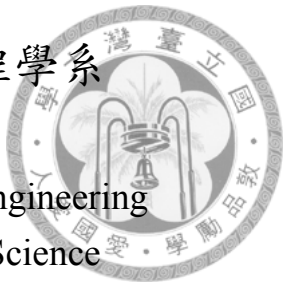

國立臺灣大學電機資訊學院資訊工程學系
碩士論文
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

內容串流中實體特性偵測之研究
Detection of Entity Properties in Content Stream

李卿澄
Qing-Cheng Li

指導教授：陳信希 博士
Advisor: Hsin-Hsi Chen, Ph.D.

中華民國 103 年 7 月
July, 2014

國立臺灣大學碩士學位論文
口試委員會審定書

內容串流中實體特性偵測之研究

Detection of Entity Properties in Content Stream

本論文係李卿澄君（學號 R01922024）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 103 年 7 月 31 日承下列考試委員審查通過及口試及格，特此證明

口試委員：陳信希

（指導教授）

蔣銘偉　郭俊麟

系主任　許永真

誌謝

在這兩年的求學過程中，首先我要感謝指導教授陳信希老師在這兩年間的指導與提點，使我領略了做研究的樂趣。還要感謝自然語言處理實驗室一同努力的夥伴宗倫、長昇、婉珊，給予建議的基鑫學長、界人學長、瀚萱學長、宇錚學長、綵思學姊，以及不時與我討論的志杰、俊杰、詠翔、彥賓、聖倫。

而出了實驗室，我要感謝不時會在系館各處遇到並提供建議，以及提供課業上各種支援的家偉、俊宇、挺宇、偉哲、瑋宗、姿君、均達、文傑、瀚生、承恩。

除此之外也要感謝一同創造點擊率的世恩、宇新，一起組織讀書會的偉寧、祐棠，一起參加黑客松的騰保，一起在雅虎實習的快樂夥伴郝德、偉翰，當然還有老同學宗翰、冠宇。

最重要的還要感謝我的父母、雙胞胎弟弟給予我全力的支持，讓我無後顧之憂的專注於課業上，順利完成這兩年的學業。

謝謝你們，謝謝大家。

李卿澄 民國 103 年 7 月

摘要

世界上的知識日新月異，透過志願編輯者更新的知識庫無法跟上知識產生與改變的速度，如何縮短知識產生與知識庫更新間的差距，也就是知識庫加速，便成為了重要的議題。

知識庫中記載的實體與其特性也是相當重要的知識，本研究提出了基於樣式，自資訊匯集而成之內容串流中快速地偵測文件是否包含特定實體特性的方法。偵測流程包含了樣式比對、樣式篩選與特性消歧義等步驟。透過樣式比對與實體特性與樣式的關聯偵測實體特性，存在樣式的品質、可信賴度、對映特性的歧義等問題，本研究於樣式比對前進行樣式篩選，比對後進行特性消歧義以降低上述問題的影響。

實驗結果分析了樣式信心值、可信賴度、特性歧義度對效能造的影響，發現特性消歧義的步驟中，引入實體類型資訊與使用簡單貝氏分類器後，偵測效能有顯著的提升。

透過實體特性的偵測，有助於自內容串流中篩選對知識庫更新有幫助的文章，以供志願編輯者作為更新與維護知識庫的依據。

關鍵字：知識庫加速、樣式比對、實體特性偵測

Abstract

World knowledge varies with time, but the change of knowledge about an entity often waits for a long time before a human editor update it in knowledge base (KB). How to accelerate the update of KB is an important problem, it's also called knowledge base acceleration (KBA).

In this paper, we propose a method that detects entity's properties in content stream efficiently and effectively base on patterns. The detection process has three phases including pattern selection phase, pattern matching phase and property disambiguation phase. pattern quality, reliability and ambiguity are three major issues in the process.

The experimental results show the impact of patterns' confidence value, reliability and ambiguity degree. We found that using the entity type information and naive bayes classifier improve the performance of the detection system.

Detection of entity's properties filters documents from content stream. It's helpful for human editors to use the information in those documents to update the KB.

Keywords: Knowledge Base Acceleration, Pattern Matching, Entity's Property Detection

目錄

圖目錄

表目錄

第一章　緒論

近年來，隨著網際網路的快速發展，網際網路上開始出現越來越多彙整人類知識的網站與資源，例如 Wikipedia[1]，目前已經有超過 280 種語言，其中光是英語的條目就超過 4,500,000 條，是透過來自世界各地的志願編輯者一字一句的貢獻建立而成。

除了讓世界各地的人們可以在網際網路上共享知識之外，電腦也得以利用人類的知識，輔助、改善、甚至自動化進行多面向的人工智慧應用，例如資料探勘、知識擷取、自動問答系統等。

為了達成這些目標，使電腦可以看懂人類的知識，於是出現了各式各樣透過擷取網路上的知識資源，產生的結構化知識資料庫，甚至進一步彼此相互鏈結。如圖 1.1 所示，目前已有不少的結構化知識資料庫公開。

1.1　背景介紹

在現今這個資訊社會，網際網路是重要的知識來源。當有新的事件發生時，例如某人的出生或死亡、某政治人物贏得了選舉、某地發生了天災等等，這些資訊便會在網路上透過網路新聞、網誌、論壇、社群網站、微網誌等管道流傳。這代表著可能有新的知識出現了，或者舊有的知識發生了變動。而一些知識資料庫如 Wikipedia 的志願編輯者注意到這些資訊之後，便會據此更新 Wikipedia 上的條目，將這些新的知識加入資料庫之中。

這種用來儲存實體（Entity）與實體的特性（Properties）、實體與實體間的關係（Relationships）的資料庫被稱為知識庫（Knowledge Base），Wikipedia 就是一個知識庫。

[1] http://www.wikipedia.org/

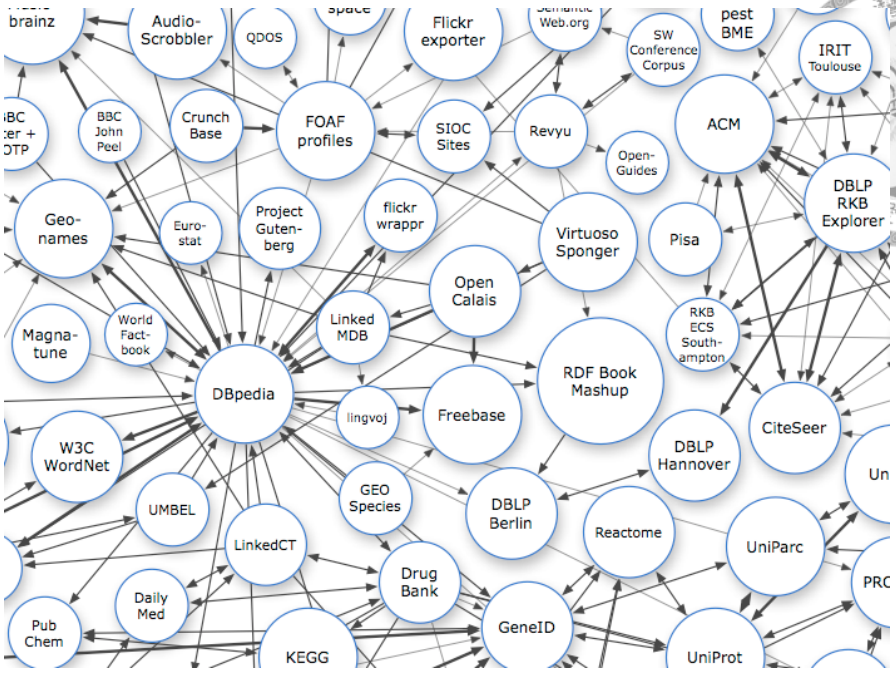

圖 1.1: 結構化知識資料庫之間的鏈結
資料來源：linkeddata.org（2011）

Wikipedia 用文章（條目）的形式儲存了人物、組織、公司、城市、事件等實體，文章透過超連結（Hyperlinks）互相鏈結，包含超連結的句子描述了實體與實體間的關係。除此之外，還有的條目會包含資訊框（Infoboxes），以半結構化的形式描述了實體的特性，圖 1.2 就是一個 Wikipedia 條目的例子。圖中的句子「... who was the co-founder, chairman, and CEO of *Apple Inc.*」，說明了 Steven Jobs 與 Apple 公司之間的關係：Jobs 是 Apple 公司的 CEO。而這個關係對 Jobs 這個實體來說，代表其具有「was CEO of」的特性。也就是說，與其他實體間的關係帶有實體特性的資訊。

除了 Wikipedia 之外，還有 DBpedia[2] (Lehmann et al., 2014)、YAGO[3] (Suchanek et al., 2007)、Freebase[4] (Bollacker et al., 2008) 等規模大小不一、不同應用的知識庫。其中 YAGO、DBpedia 是透過自動化的方法擷取 Wikipedia 的內容，Freebase 則是仰賴社群更新，這些知識庫都直接或間接仰賴志願者將新知識加入。

[2]http://dbpedia.org/
[3]http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago//
[4]https://www.freebase.com/

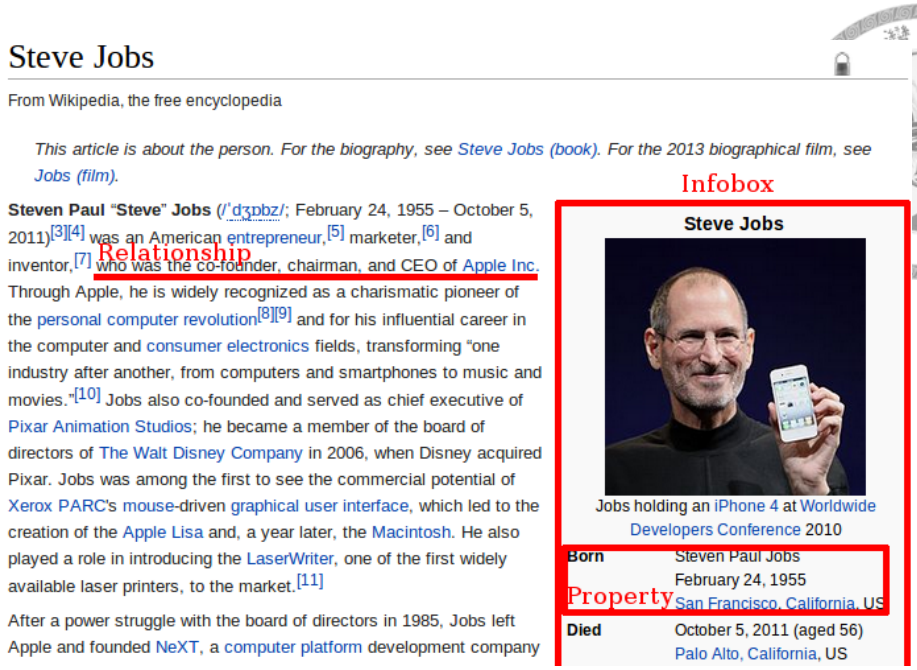

圖 1.2: Wikipedia 條目與資訊框

1.2 研究動機

知識庫內所記載的知識直接或間接仰賴志願編輯者的維護與更新，但這些人數比起資料庫中所需要維護的實體顯然少非常多。這意味著知識庫的更新總是落後於新知識——無論是過去從來不知道的知識或是舊有知識內容的更動——的產生，當新知識產生一段時日之後，知識庫的維護者才會更新知識庫。Frank et al. (2012) 追蹤了約 60,000 個被 Wikipedia 引用的來源網頁，計算從該網頁產生的時間與被 Wikipedia 引用的時間差，如圖 1.3 所示，中位數約為一年，呈現了這種落後。因此，如何讓人工編輯的速度可以跟上新知識出現的速度便成為一個重要的課題。

為了縮短這個差距，自 2012 年起美國 NIST[5]的 TREC[6]開始舉辦知識庫加速（Knowledge Base Acceleration, KBA）競賽，提供了一個從由新聞、部落格、論壇等內容所組成、依時間所排列的龐大內容串流[7]（Content Stream），模擬真實世界中資訊的產生。希望參賽者可以建立一個系統，由給定的內容串流，對有興趣的實體[8]，包含了人物、組織或建築，進行過濾，推薦文件給知識庫的編輯者，提示

[5]National Institue of Standards and Technology
[6]Text Retrieval Conference
[7]2012 年包含了 4,973 個小時，2013 年包含了 11,948 小時。(Frank et al., 2013) 每小時包含約 100,000 份文件。
[8]2012 年有 27 個人物、2 個組織，2013 年有 98 個人物，12 個組織與 24 個設施。

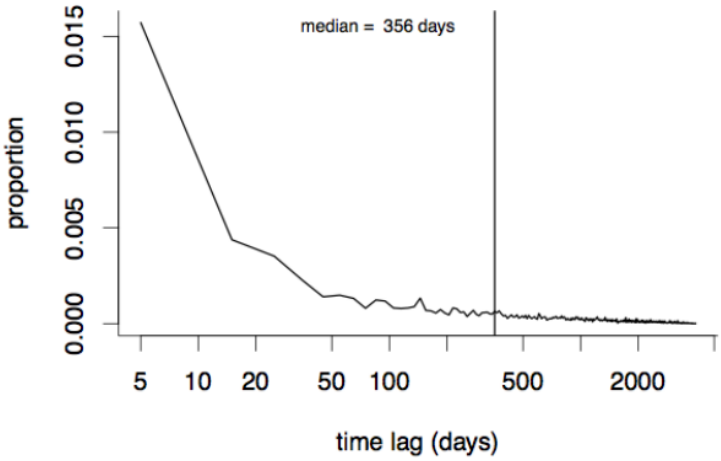

圖 1.3: Wikipedia 引用延遲
資料來源：Frank et al. (2012)

這份文件可能含有新的知識可供更新知識庫。

要從可能包含新知識的內容串流中，過濾出可能可以協助更新知識庫的文件，則需要知道文章中是否包含了我們關注的實體以及知識。例如文章中出現了句子「Jobs was born in San Francisco, California on February 24, 1955」，這句子中就包含了「Jobs」、「San Francisco」、「California」等實體，透過「was born in」這個樣式（Pattern），可以知道「Jobs」與「San Francisco」、「California」間具有「出生於」的關係，包含「出生地」這項實體特性。若可以快速地辨識文章中是否包含這些資訊，推薦給知識庫的編輯者參考，便能夠進一步地加速知識庫的更新與維護。

1.3 研究目標

本研究的目標是建立一個過濾系統，可以快速地處理內容串流。對於內容串流內的每一份文件，偵測該文件中是否存在我們感興趣的實體特性。

本研究將利用樣式來判斷是否有實體間關係這樣的特性存在於文件之中，也就是說，若有一個樣式與關係的資料庫，透過樣式比對，當文句比對成功時，透過資料庫尋找對應的關係。

整個偵測的過程要兼顧效率和效能兩項要素。這個系統要有效率，如果樣式比對或對句子進行更深入的剖析的效率不高，就沒有辦法處理內容串流大量的文

件。這個系統要有效能，透過樣式判斷關係的效能可能會受到樣式本身的品質（Quality）、樣式的覆蓋度（Coverage）、樣式值得信賴的程度（Reliability）、樣式－關係對映的歧義問題（Ambiguity）等因素影響。

透過這個系統，我們可以知道經過處理的文件到底是完全沒有提及我們感興趣的實體特性，還是提到一個，甚至多個實體特性。有這樣的資訊就可以進行更進一步的分析，例如這個特性是屬於哪個實體的？這個特性的內容是什麼？來判斷是不是應該要對實體的特性內容進行更新，甚至是自動更新，以幫助知識庫的更新與維護。

1.4 論文架構

本論文共分成五個章節。第一章是緒論，簡介本研究的背景、動機與目標。第二章是相關研究與文獻，列舉與知識庫相關的研究與資源，以及關係與樣式相關的研究。第三章是研究方法，提出如何由內容串流中偵測實體特性的方法與步驟。第四章是實驗結果與分析，說明實驗資料集、評估方式與結果分析。第五章是結論與未來展望，總結研究成果。

第二章　相關研究與文獻

本章將分別介紹知識庫相關的資源與研究，涵蓋了結構化知識庫（Structural Knowledge Base）、知識庫加速（Knowledge Base Accerleration）、與知識庫的應用（Applications of Knowledge Base），以及樣式與實體間關係的相關研究。

2.1　知識庫

2.1.1　結構化知識庫

相對於 Wikipedia 以文章來儲存知識，Lehmann et al. (2014) 的 DBpedia、Suchanek et al. (2007) 的 YAGO、以及 Bollacker et al. (2008) 的 Freebase 等都以結構化的方式儲存知識。DBpedia、YAGO、與 Freebase 都提供以 TTL（Terse RDF[1] Triple Language）格式儲存的資料，以供處理與利用。

DBpedia 是一個大型、多語言的知識庫，自 Wikipedia 擷取知識。透過 Wikipedia 條目內的資訊框（Infobox），擷取實體的特性，並連結實體。同時進一步整理了資訊框內的特性，以人工將語意相同的實體特性合併。

YAGO 除了自 Wikipedia 擷取資訊外，還搭配了 WordNet[2]，建立一個輕量、高品質與覆蓋度的實體與關係資料庫。其以 < 實體, 關係, 實體 > 的形式儲存事實（Facts），即實體與實體間的關係，並定義關係的領域（Domain）與範圍（Range）、階層式架構的實體類型（Types）與其他資源如 DBpedia、WordNet 間的連結。

Freebase 是一個人類知識社群協作的結構化可擴展元組（Scalable tuple）資料庫。與 Wikipedia 一樣，是由志願者創建與維護其中的資料，並與其它知識庫互有連結。

[1] Resource Description Framework
[2] http://wordnet.princeton.edu/

表 2.1: TREC 知識庫加速競賽資料量
資料來源：Frank et al. (2013)

	2012	2013
Corpus	7 months (4,973 hours) >400M documents 40% English Oct 2011 to Apr 2012.	17 months (11,948 hours) >1B documents 60% English or unknown Oct 2011 to Feb 2013
Queries	27 people 2 organizations all from Wikipedia	98 people, 19 organizations, and 24 facilities. Fourteen inter-related communities of entities, such small towns like Danville, KY, and Fargo, ND, and academic communities like Turing award winners.
Assessing	70% agreement on "central"	3198 hours have >0 vitals 76% agreement on "Vital" (replaced "central")
Submissions	11 teams, 40 runs	13 teams, 140 runs
Metrics	F_1, Scaled Utility	F_1, Scaled Utility

2.1.2 知識庫加速

Frank et al. (2013, 2012) 縱覽了 2012、2013 年的 TREC 知識庫加速競賽，連續兩年都有的 CCR（Cumulative Citation Recommendation）任務是過濾內容串流，推薦對更新目標實體有幫助的文章，以縮短知識產生與知識庫更新之間的時間差。表 2.1 是這兩年 CCR 任務中內容串流的資料量與目標實體的數量，內容串流中每小時約有 100,000 份文件。CCR 任務主要專注在尋找文章中是否有出現目標實體（Mentions/Zero Mentions），再進一步依照有幫助的程度分成四個等級（Garbage/Neutral/Useful/Vital）。

Kjersten and McNamee (2012) 是 2012 年效能最好的團隊，以具名實體、單字詞（Unigram）作為特徵訓練 SVM 分類器，對每個目標實體訓練一個二元分類器。把尋找實體的問題作為主題分類（Topic Classification）問題來處理。

Wang et al. (2013) 則是 2013 年效能最好的團隊。以查詢擴展（Query Expansion）、分類、與學習式排序法（Learning to rank）研究這個問題。其中分類是使用以文件特徵、實體特徵、文件與實體特徵、時間特徵、與引用特徵訓練隨機森林（Random Forest）分類器。

此外，Bonnefoy et al. (2013) 也專注於尋找串流中的實體，提出了當出現新的實體時不需要新的訓練資料集的方法。使用了文件中心特徵、實體資料特徵、與時間特徵訓練隨機森林分類器，以尋找串流中與實體高度相關的文件。並且以 TREC KBA 2012 的資料做測試，效能比 Kjersten and McNamee (2012) 還要好。

本研究與 TREC 知識庫加速競賽的目標相同，都是為了縮短從知識的產生或變動到知識庫更新之間的時間差距。不同的地方是，知識庫加速競賽專注於目標實體的尋找，再判斷是否有幫助更新知識庫。本研究的目標是尋找實體特性，也就是知識庫加速中最終希望更新的資訊。若知道哪些實體特性是會隨著時間變化的，當偵測到這種實體特性時便能知道有可能有新的知識產生。

2.1.3 知識庫的應用

知識庫的應用常見於問答系統或實體消歧義上。

在問答系統的應用上，知識庫提供了一個知識的集合，也就是答案的來源，Adolphs et al. (2011) 將問句轉換為查詢語言，對 YAGO 進行查詢並回答問題。Yao and Van Durme (2014) 利用 Freebase 的資料，以邏輯形式（Latent Logical Forms）對映問題與答案。Berant et al. (2013) 將知識庫視為一個實體為點，關聯為邊的圖，假設答案只會存在於周遭的節點上，成為一個主題圖（Topic Graph），將問句剖析後在圖上走訪尋找答案。

而在具名實體消歧義（Named Entity Disambiguation）上的應用，Mendes et al. (2011) 提供了一個系統將文本內的實體連結至 DBpedia。他們將 DBpedia 的實體建立向量空間模型（Vector Space Model），但與傳統 VSM 不同的是，IDF（Inverse Document Frequency）反應一個字在整個資料集內的重要程度，卻沒辦法反應一個字在歧義候選選項內的重要程度。為了要能夠找出一個字對消歧義的能力，他們提出 ICF（Inverse Candidate Frequency）來取代 IDF 計算字在向量空間中的權重，定義 $ICF(word) = log\frac{|R|}{n(word)}$，其中 $|R|$ 是候選實體的數量，$n(word)$ 是字在候選實體中出現的次數。將 DBpedia 的實體以向量空間模型建模，以 $TF * ICF$ 計算字的權重，將文句與候選的實體向量空間模型計算相似度，取最相似的作為結果。

2.2 樣式與實體間關係

實體間的關係是可能隨著時間而產生變化，知道哪些關係會變化對於協助知識庫的更新是有幫助的，可以更注意那些會變動的關係。Takaku et al. (2012) 提到

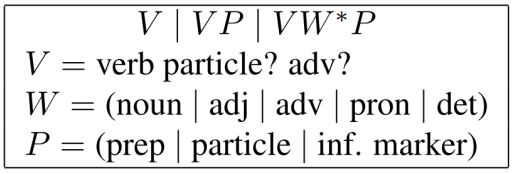

圖 2.1: 詞性標記組成的正則表示式
資料來源：Fader et al. (2011)

有些實體間的關係是相對恆常的，例如《1Q84》的作者是村上春樹是不會改變的事實；而有些關係則是隨著時間而變動的，例如美國的總統是布希，只有在某一段時間內是正確關係。此研究將實體間的關係分為是否為恆常（Constant）或是否唯一（Unique）。恆常是指是否這組關係不會隨著時間改變，唯一是指抽換實體後關係是否仍然正確。此研究人工挑選 1,000 組關係並利用時間、實體出現在特定時段內的頻率、文法等特徵對關係進行分類。

在擷取關係的部分，Fader et al. (2011) 建立了一套開放資訊擷取系統（Open Information Extraction System），可以擷取開方式的關係 <arg_1, $relation$, arg_2>，而不限於人工標定的關係。過去的開放式資訊擷取系統會有非相干擷取（Incoherent Extraction）或無資訊擷取（Uninformative Extraction）的問題，透過句法限制（Syntactic Constraint），制定由詞性標記所組成的正則表示式，例如圖 2.1 ，定義關係可以出現的形式，避免非相干擷取的問題。詞彙限制（Lexical Constraint）認為一個合法的關係應該在大型的語料庫之中會有許多不重複的參數（Argument）。由大型語料庫中擷取出的關係中，不重複參數配對的數量超過一個定值才會被認為是合法的關係，以解決無資訊擷取的問題。

而 Wikipedia 中，每一篇文章，或稱條目，是描寫一個特定的實體，Moro and Navigli (2012) 自 Wikipedia 文章中提及其他實體的句子擷取關係，由於同一種關係可能由不同的字樣來表達，為了建立同義字樣集，定義了連結實體間關係的字樣間的相似度是字樣連結的實體重複的比例，比例越高越相似，相似的字樣就可以聚集成同義字樣集，並且應用 Wikipedia 的分類與上下文給予同義字樣集語意上的分類。

Nakashole et al. (2012a,b) 建立了一個名為 PATTY 的分類集（Taxonomy），自 Wikipedia、New York Times 中擷取出樣式，建立同義樣式集（Pattern Synset），並將同義樣式集合併，連結知識庫定義的實體關係。

PATTY 的關係是以「<Domain> Pattern <Range>」表示，Domain 與 Range 是實體的類型（Types），來自知識庫 YAGO、DBpedia。而 PATTY 的樣式是以單字（Words）、詞性標記（Part-Of-Speech Tags）、萬用字元（Wildcards）所組成，例如「*<person>*'s [[adj]] voice * *<song>*」中，Domain 是 *person*，即第一個實體的類型是 *person*；Range 是 *song*，即第二個實體的類型是 *song* 。「[[adj]]」是詞性標記，代表這裡可以填入任意形容詞，「voice」是單字，必須完全匹配，「*」是萬用字元，可以填入任何字。例如「Amy Winehouse's soft voice in 'Rehab'」這個句子就符合這個樣式。

PATTY 擷取樣式的方式是自 Wikipedia、New York Times 抽取句子，以史丹佛剖析器（Stanford Parser[3]）對句子建立相依性剖析樹，以 YAGO、Freebase 作為實體的字典，判斷若句子中有兩個實體，則把兩實體間在樹上的最短路徑上的字句擷取出來作為樣式，並進一步合併樣式為同義樣式集。比起 Fader et al. (2011)，PATTY 可以抽取任意的關係，不被詞彙或句法所限制。

抽取出樣式後，為了評估一個樣式的品質，PATTY 定義了一個信心值（Confidence）。對每一個樣式，都存在一組支持集（Support Set），由符合樣式的實體對（Pairs of entites）所組成。透過支持集的大小，對每個樣式計算信心值，將樣式中的實體類別改為類別繼承架構中的更廣義的父節點類別，以可以填入的實體對數作為分母，支持集作為分子，得到的分數就是心信值，代表一個樣式的品質。

除了同義樣式集之外，PATTY 還做了關係釋義（Relation Paraphrasing）：給定一個來自知識庫的關係，判斷一個樣式是否可以描述此一關係。圖 2.2 為其成果，如 YAGO 的「wasBornIn」關係可以用「[[con]] grew up」、「was bron [[con]] lives in」等樣式來描述。PATTY 內有 127,811 個樣式可以描述 225 種 DBpedia 關係，43,124 個樣式可以描述 25 種 YAGO 關係。其透過隨機選取 1,000 組釋義來評估，平均的精確度（Precision）是 0.76±0.03。本研究的實驗採用了 YAGO 關係釋義中的 24 種 YAGO 關係，於本論文第 4 章有詳細說明。

本研究擬使用樣式來尋找關係、實體特性，PATTY 正好提供了所需要的樣式，以及樣式可以描述哪些關係等資料。

[3]http://nlp.stanford.edu/software/lex-parser.shtml

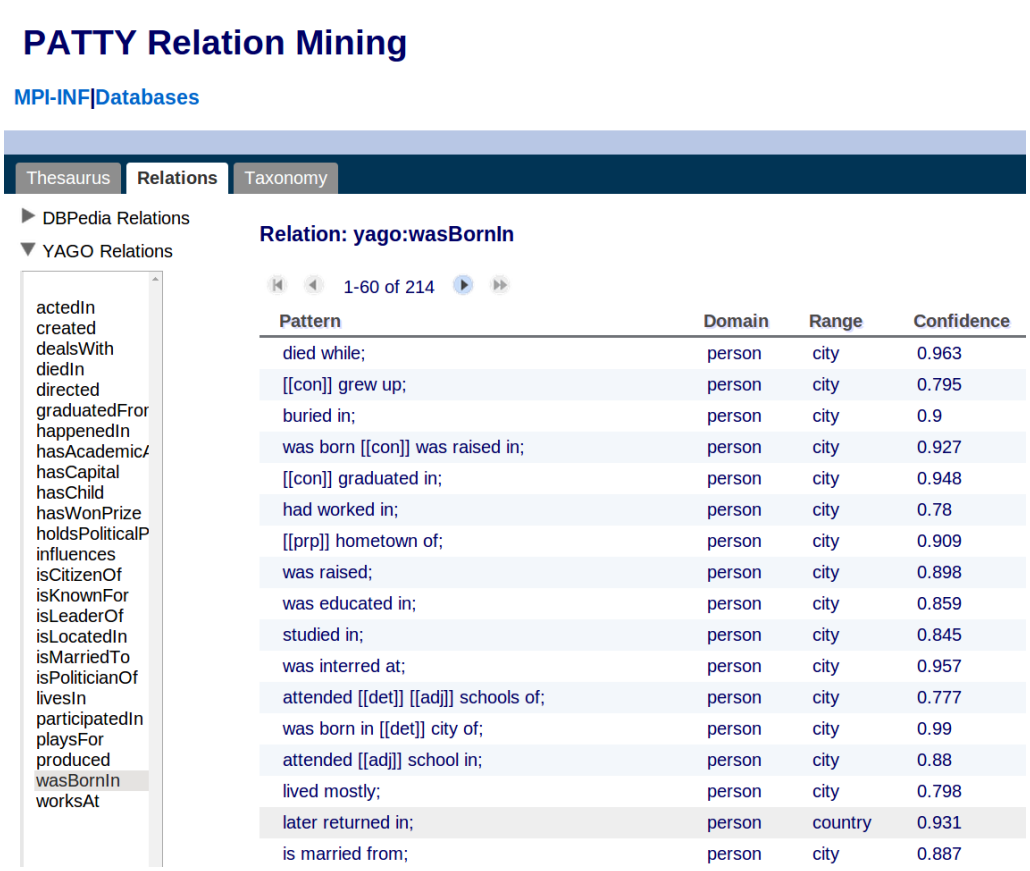

圖 2.2: PATTY Online Relations

第三章　研究方法

本研究擬自內容串流中偵測實體的特性，而實體的特性之一：實體間關係，則可以語句樣式表現，因此本研究主要利用語句樣式來偵測這樣的特性是否存在。本章將介紹所使用的資源、樣式比對、樣式篩選、與特性歧義的問題與解決方法。

3.1　以樣式偵測特性

當我們在字裡行間要描述一個實體的特性時，應該會有某些特定的樣式。例如我們知道 Jobs 的出生地這個特性是 San Francisco，我們可能會以「Jobs was born in San Francisco」這樣的句子來表達這個概念。而在上述句子之中「was born in」就是一個可以用來表示出生地此一實體特性的樣式，當我們在句子中看到「was born in」時，就可以推測這個句子存在某人的出生地是某地這樣的實體特性。

基於這樣的想法，本研究將利用樣式的出現有無來決定是否存在實體特性，設計了如圖 3.1 的偵測流程。

圖 3.1 中，首先要先有一份實體特性與樣式的關聯表，知道每個實體特徵可以由哪些樣式來表達。以上述例子來說，就會有一張表如表 3.1 列有「出生地」這項實體特性，並列出所有可以用來描述這項實體特性的樣式如「was born in 」、「[[adj]] childhood in」、「lived in」等；表上也會有其他的實體特性如「生活地」，並列出樣式如「lived in」、「been working in」等可以描述這項實體特性。

利用這份關係表，對文章中的句子進行比對，檢查關係表中的樣式是否出現。若有出現，則這篇文章可能包含樣式對映的實體特性。例如句子「Jobs was born in San Francisco」包含了「was bron in」，但不包含「lived in」等其他樣式。透過樣式「was born in」可以從關聯表中查到此樣式屬於實體特性「出生地」，因此

表 3.1: 實體特性與樣式範例關聯表

Property	Patterns
出生地	was bron in, [[adj]] childhood in, lived in
生活地	lived in, been working in

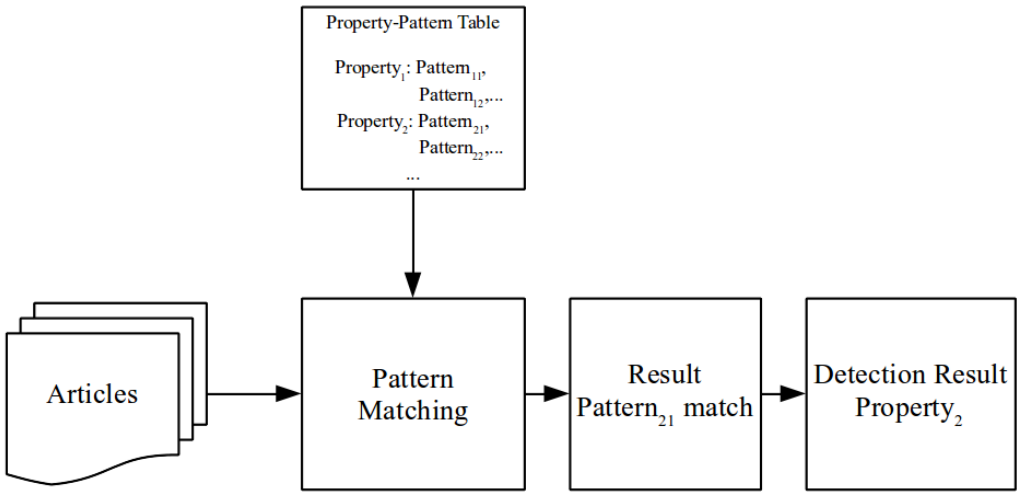

圖 3.1: 以樣式偵測特性流程概念

推論這個句子包含了「出生地」這項實體特性資訊。樣式比對的細節會在第 3.2 節中詳述。

本研究擬利用 PATTY 已經提供的關係釋義表，作為偵測流程中所需要的實體特性與樣式關聯表。表中包含了知識庫 YAGO 內定義的實體間關係、YAGO 關係的領域與範圍（表 3.2），以及可以用來描述這個關係的樣式。

光是這樣的流程並不足以偵測實體特性。使用樣式來偵測實體特性還會有樣式覆蓋度（Coverage）、品質（Quality）、可信賴度（Reliability）、與歧義性（Ambiguity）的問題。

樣式覆蓋度問題是指對於某個實體的特性，在關係表內屬於這個實體特性的樣式佔所有可以表達該特性的樣式之比例。以表 3.1 為例，其中「生活地」這項實體特性表中列了 2 個樣式，但實際上有更多的樣式可以表達這個概念，例如「been lived at」就沒有列在表中，這樣有此樣式的句子就沒辦法被偵測為含有實體特性的句子。

樣式的品質則是一個字串到底能不能夠作為一個樣式，一個樣式的品質不高，

表 3.2: YAGO Relations in PATTY

YAGO Relation	Domain	Range	Number of patterns
actedIn	wordnet_actor	wordnet_movie	2023
created	yagoLegalActor	Thing	3215
dealsWith	wordnet_location	wordnet_location	366
diedIn	wordnet_person	wordnet_city	1352
directed	wordnet_person	wordnet_movie	1228
graduatedFrom	wordnet_person	wordnet_university	2129
happenedIn	wordnet_event	yagoGeoEntity	47
hasAcademicAdvisor	wordnet_person	wordnet_person	632
hasCapital	wordnet_location	wordnet_location	24
hasChild	wordnet_person	wordnet_person	3620
hasWonPrize	yagoLegalActorGeo	wordnet_award	78
holdsPoliticalPosition	wordnet_person	wordnet_person	1173
influences	wordnet_person	wordnet_person	2461
isCitizenOf	wordnet_person	wordnet_country	813
isKnownFor	yagoLegalActor	Thing	1574
isLeaderOf	wordnet_person	yagoLegalActorGeo	465
isLocatedIn	yagoPermanentlyLocatedEntity	yagoGeoEntity	1300
isMarriedTo	wordnet_person	wordnet_person	4276
isPoliticianOf	wordnet_person	wiki_states_of_US	465
livesIn	wordnet_person	wordnet_location	718
participatedIn	yagoLegalActorGeo	Thing	89
playsFor	wordnet_person	wordnet_organization	1491
wasBornIn	wordnet_person	wordnet_city	1226
worksAt	wordnet_person	wordnet_organization	1602

那麼這樣的樣式有可能太廣泛的出現在句子之中，致使這樣的樣式沒有辦法區辨出實體屬性。例如 PATTY 關係釋義中的 YAGO:actedIn 內的樣式「links」就只有這一個單字，用以描述實體特性顯得太廣泛了些，以此擷取回的文章有包含實體特性的比例就較低。

樣式可信賴度則是一個樣式到底能不能用來表達實體特性。例如，對於特性 YAGO:diedIn，以「since worked in」來描述完全文不對題，又如 YAGO:wasBornIn 中的樣式「also opened in」也無法描述特性。

而為了處理樣式的品質、可信賴度問題，應該要在圖 3.1 流程中的關聯表與樣式比對中增加一個樣式篩選的步驟，篩選出可信賴、有一定品質的樣式來進行樣式比對，此部份將於第 3.3 小節說明詳細內容。

經過樣式比對後，每篇文章可能存在數個樣式，如果在關聯表中每個樣式只對映到一個關係，那便沒有歧義。但一個樣式可能存在於多個實體特性的關聯表內，例如樣式「first met with」就被 PATTY 認為可以表示 5 種特

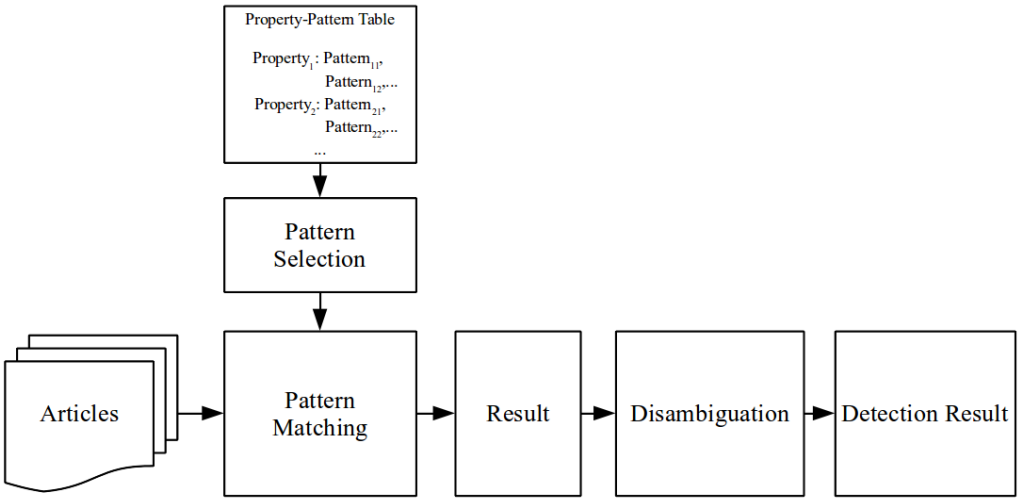

圖 3.2: 以樣式偵測特性流程

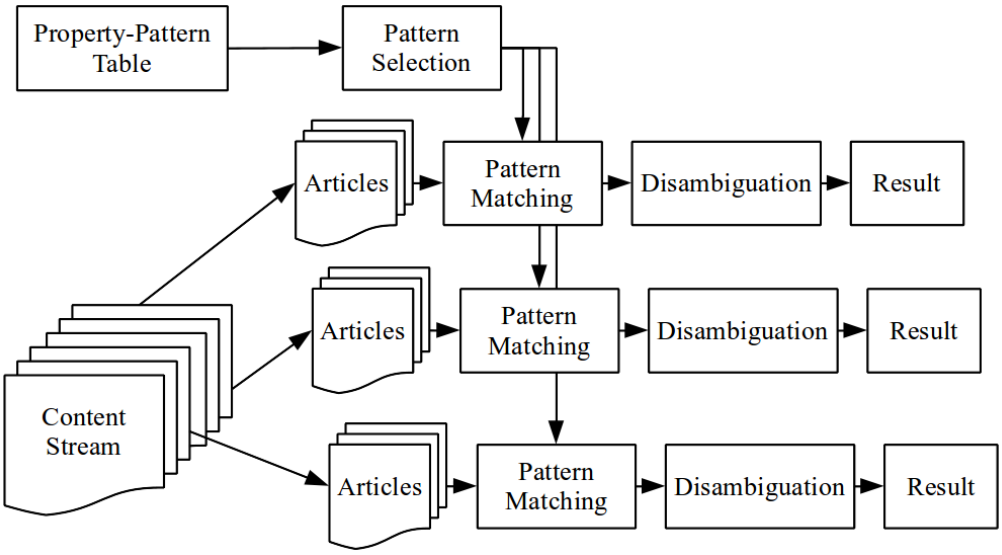

圖 3.3: 平行處理架構

性（hasAcademicAdvisor、isKnownFor、isMarriedTo、influences、hasChild）。因此在樣式被比對出來之後，整個偵測流程中還需要一個消歧義的步驟，決定當此樣式出現時，到底是 5 種特性中的哪一種？哪幾種？或者都沒有？這個部份將在第 3.4 節中說明。

補上了樣式篩選與消歧義後的流程圖應修正為圖 3.2。樣式比對使用經過篩選的樣式，比對出來後進行消歧義才完成偵測實體特性之流程。由於偵測實體特性的流程對於每一分文件都是相同，且彼此互相獨立，因此可以平行處理文件，如圖 3.3 所示，處理內容串流的大量文件。

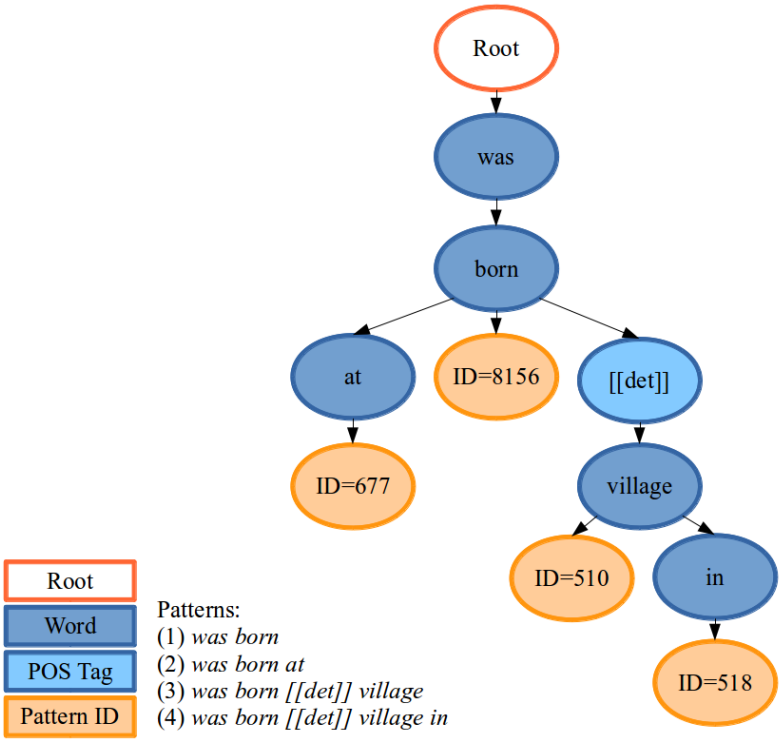

圖 3.4: 樣式前綴樹

3.2 樣式比對

樣式比對是要從句子之中確定某一個樣式是否存在，PATTY 中提供的 25 個 YAGO 關係有 43,124 個樣式，但其中有不少是重複出現的，實際上總共有 19,031 個獨特的樣式，如果對每個句子都做 19,031 次比較顯然不是一個好方法。

為了讓樣式比對可以有效率些，我們將所有的樣式建立成一顆前綴樹（Prefix Tree）。先將所有首字相同的樣式合併成同一個節點，再看第二個字是否相同，如果不同就分支出去，依此類推，最後再連結到樣式編號，即可透過編號查詢該樣式可能表示的實體特性。如圖 3.4 所示，四個樣式「was born」、「was born at」、「was born [[det]] village」和「was bron [[det]] willage in」的首字皆為 was，所以併成一個節點。第二個字也都相同，所以也併成一個節點。而此時「was born」已經完成了，所以 born 節點下新增一個樣式編號的節點。剩下三個樣式的第三個字有詞性標記「[[det]]」和單字「at」，便在 born 節點下新增 [[det]] 和 at 節點。at 節點完成了「was bron at」樣式，而 [[det]] 節點則繼續將後續的字加入這棵前綴樹上，最後這棵樹上所有葉子都是一個樣式編號。

有了樣式前綴樹後，再來就是利用這棵樹來比對句子中是否有出現其中的樣

式。演算法 1 描述了比對的過程。

演算法 1 的輸入是擬比對的文章 A 以及已經預先建立完成的樣式前綴樹 T，輸出是文章 A 所出現過的樣式及其出現的位置。對於文章 A 中的每個句子 S，由於樣式包含詞性標記，因此需要將 S 標記，實驗中使用的是 Textblob 的 APTagger[1] 來進行快速的詞性標記。接下來開始從句子的第一個字開始，如果這個字或其詞性標記有出現在樣式前綴樹 Root 下的第一層節點中出現，便將這個節點和此字於句子中的位置記錄下來。再看句子中的下一個字，此時先檢查暫存紀錄中的字是不是能夠繼續往下走這棵樣式前綴樹，如果可以往下走代表還沒有發現樣式；如果子節點已經是帶有樣式編號的樹葉的話，就是發現了一個樣式，然後紀錄下來。如果發現沒有辦法繼續往下走，則代表在未來的步驟中不可能走到任何一個紀錄樣式編號的樹葉，因此將這組紀錄從暫存紀錄中移除。最後便可以知道有哪些樣式出現在文章中以及出現的位置。

演算法 **1** 樣式比對演算法

輸入： A: 文章; T: 樣式前綴樹;
輸出： P: 文章 A 中出現的樣式;

```
1: P ←[]                                                    ▷ 初始化 P
2: for all Sentence S in A do
3:     對 S 進行詞性標記
4:     tmpP ←[]                                             ▷ 初始化暫存陣列
5:     for i from 0 to length of S do
6:         (word,POStag) ← S[i]
7:         for all possiblePattern in tmpP do
8:             if word or POStag in T's possiblePattern.depthInTree+1 level nodes
   then
9:                 continue
10:            else
11:                if T's possiblePattern.depthInTree+1 level node is PatternID then
12:                    add (PatternID,startPoint,i as endPoint) into P
13:                end if
14:                remove possiblePattern from tmpP
15:            end if
16:        end for
17:        if word or POStag in T's first level nodes then
18:            add (depthInTree,i as startPoint) into tmpP
19:        end if
20:    end for
21: end for
22: return P
```

[1] https://github.com/sloria/textblob-aptagger

演算法 1 中同時於暫存紀錄中的可能樣式最多只會跟樣式前綴樹的深度相同，若文章的長度是 n 個字、每個字對多可能檢查暫存同前綴樹深度 m 次，則演算法 1 的時間複雜度是 O(mn)。

3.3 樣式篩選

樣式篩選是希望在建立樣式前綴樹提供樣式比對使用前，先對樣式進行一些篩選，以解決樣式品質、可信賴度甚至是歧義問題。針對這三個問題，主要有三種篩選的方法。

首先是關於樣式的品質，由於 PATTY 對每個樣式有計算一個信心值，例如 YAGO:actedIn 中的樣式「links」的信心值是 0.143，而「starred in [[det]] film」的信心值是 0.937，後者相較於前者更適合作為樣式來進行特性偵測。這裡主要是針對信心值進行篩選，設定信心值門檻，不使用信心值低於門檻值的樣式進行樣式比對。

再來是可信賴度的部份，由於我們認為可信賴度是用來是樣式能不能表達特性的程度，因此定義了計算可性度為一個樣式出現的次數中，其出現的文件確實具有實體特性的比例。例如，「was born」曾出現在 22,543 份文件之中，其中有 12,038 份文件確實有「出生地」這項特性，因此樣式「was bron」對「出生地」這項特性的可信賴度是 0.534。而「school in」曾出現在 1,486 份文件中，其中有 387 份文件有「出生地」這項特性，其可信賴度為 0.26，相較於「was bron」來的低。可信賴度越高代表這個樣式出現時有越高的可能可以代表存在實體特性，反之可信賴度較低的樣式則可能根本無法掉表實體特性。藉由可信賴度，我們可以篩選出較值得信賴的樣式進行比對。

最後是歧義的問題，有部份的樣式可能同時表示多種實體特性，以表 3.1 的例子來說，「lived in」同時可能表示「出生地」與「生活地」兩種實體特性，這樣的樣式我們就稱其歧義度為 2，代表此樣式可以表達兩種實體特性。表 3.3 即顯示了各信心值下各種歧義度的樣式統計，約有 40% 的樣式具有歧義，其中樣式「worked from」的歧義度甚至高達 17（livesIn、hasAcademicAdvisor、diedIn、isLeaderOf、isKnownFor、isMarriedTo、created、influences、isLocatedIn、wasBornIn、worksAt、graduatedFrom、actedIn、produced、di-

表 3.3: 樣式數量與信心值、歧義度統計

歧義度	信心值 > 0 樣式數量	信心值 > 0.7 樣式數量	信心值 > 0.8 樣式數量	信心值 > 0.9 樣式數量
1	11381	8913	5951	1879
2	4778	4175	2328	316
3	1255	1048	683	187
4	666	600	473	67
5	635	605	278	18
6	77	68	50	10
7	132	131	52	12
8	49	49	13	2
9	26	26	12	8
10	13	13	0	0
11	12	12	9	5
12	2	2	0	0
13	0	0	0	0
14	2	2	0	0
15	0	0	0	0
16	0	0	0	0
17	3	3	0	0
總計	19031	15647	9849	2504

rected、isCitizenOf、isPoliticianOf)。歧義度大的樣式也可以視為沒辦法精確的表達特性，在樣式篩選時，會將歧義度太大的樣式捨去，避免過度混淆而影響了系統的效能。

3.4 特性消歧義

經過樣式篩選、比對之後，文章中出現的樣式可以代表的特性仍可能具有歧義，表 3.4 就呈現了各種特性之間所屬的樣式重疊的狀況。表中的數字代表樣式重疊的比例，沒有數字代表兩組樣式無重疊或者重疊的比例甚低。第 i 排第 j 欄的數字是 $\frac{|property\ i's\ patterns \cap property\ j's\ patterns|}{|property\ i's\ patterns|}$。例如表中第一排第二欄的值是 0.33，代表可以用來表達 actedIn 的樣式之中有 33% 也可以用來表達 created 這個特性。表中可以看到像是 diedIn、livesIn、wasBornIn 互有重疊，或是像 happendIn、hasCapital、participatedIn 幾乎完全不與其他特性混淆。針對歧義問題，本研究使用了以下幾種方法嘗試消歧義。

首先是使用實體的類型資訊。由於 PATTY 的實體特性如表 3.2 有每個特性的

領域（Domain）與範圍（Range），如果可以知道被描述的實體的類型為何，就可以依據特性的領域來進行判斷。例如特性 actedIn 的領域是演員（wordnet_actor），經過樣式比對之後有一個 actedIn 的樣式存在文章之中，如樣式「appeared in」存在 Wikipedia 的 Jahsha Bluntt 條目中，但 Jahsha 是籃球員（wordnet_basketball_player）而非演員，如果沒有 Jahsha 的類型資訊，系統就會判定這篇文章存在特性 actedIn，但透過類型資訊，知道 Jahsha 不是演員，就可以判斷特性 actedIn 不應該存在這篇文章之中。

再來對於有歧義的樣式，需要有挑選特性的策略。當一個樣式自文章中被比對出來時，要判斷實體特性的結果有存在一個到數個實體特性或其實沒有實體特性等可能性，要選擇一個，多個或不選便是一個問題。對於候選的樣式，有以下幾種選擇策略：(1) 只選一個、(2) 設定一個挑選門檻、(3) 全部都選。

這裡可以利用第 3.3 小節中提過的樣式對特性可信賴度作為選擇的依據。在 (1) 只選一個的策略下，挑選候選特性中可信賴度最高的特性。例如樣式「have lived in」的候選特性有 isLocatedIn、wasBornIn、isCitizenOf、livesIn、isLeaderOf，在「have lived in」總共 95 次出現中，代表 isLocatedIn 有 52 次、wasBornIn 有 21 次、isCitizenOf 有 8 次、livesIn 有 6 次、isLeaderOf 有 2 次，換算可信賴度分別為 0.547、0.221、0.084、0.063、0.021。在策略 (1) 下，選擇可信賴度最高的 isLocatedIn 作為該樣式所代表的特性。但這樣只會固定都選一個特性，沒有辦法因應可能有多個特性的情形，因此有了策略 (2)，對可信賴度進行正規化，設定一個門檻以篩選之。正規化是將所有候選特性可信賴度除以可信賴度中最大的值，經過正規化的可信賴度分別是 1、0.404、0.154、0.115、0.038。如果門檻為 0.5，那只有 isLocatedIn 能通過門檻。如果門檻值設定為 0.2，那麼 isLocatedIn、wasBornIn 都會被當作此樣式代表的特性。透過特性選擇的策略，可以嘗試降低歧義的影響。

最後，除了使用挑選策略之外，我們認為樣式出現的前後文對於該判定哪些實體特性存在或不存在是有影響的。當某個描述特定的實體特性的樣式出現時可能會伴隨出現一些固定的字詞。如果沒有那些字詞伴隨出現，有可能是與實體特性無關的句子。

例如，dealsWith 是指國家與國家間締約的特性，其中樣式「divided into」若

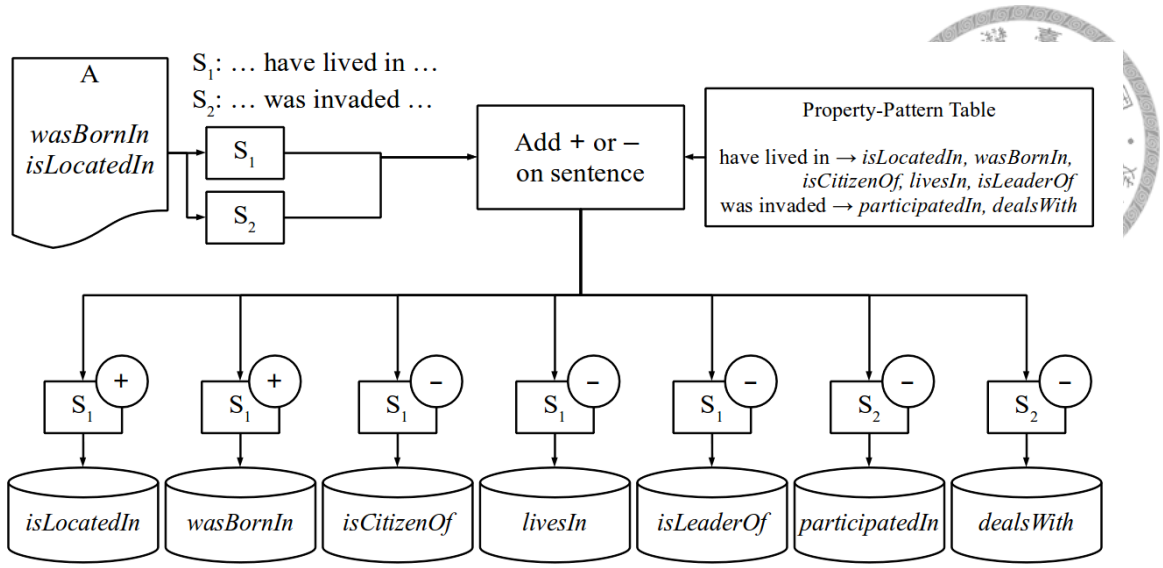

圖 3.5: 標記句子流程

用來描述 dealsWith，前後文就常會出現一些國家名、行政區之類的詞彙等。又例如 holdsPoliticalPosition 的樣式出現時，常常會接上首相、大使、市長等詞彙。若沒有這些詞彙伴隨的出現，常常是無關的句子。

我們希望可以利用樣式出現時，句子的前後文內的字來幫助選擇特性。對此，我們先從過去的文件之中把所有含有樣式的句子取出，僅將句子中所有的字轉為小寫，並沒有其他任何處理，將確實含有文件所屬特性的句子對該特性標上正向標記後放入該特性分類，對於文件沒有出現的特性標上負向標記後放入特性分類。例如文章 A 具有 wasBornIn、isLocatedIn 兩個實體特性，並有 S_1 與 S_2 兩個句子。其中的句子 S_1 包含樣式「have lived in」，可以代表 isLocatedIn、wasBornIn、isCitizenOf、livesIn、isLeaderOf 等 5 種特性；句子 S_2 包含樣式「was invaded」，可以代表 participatedIn、dealsWith 等 2 種特性。由於文章 A 並沒有「was invaded」可以代表的特性，因此我們將 S_2 加上負向標記，並放入 participatedIn 與 dealsWith 的分類中。而對 S_1，則加上正向標記後放入 wasBornIn 與 isLocatedIn 的分類之中，加上負向標記後放入 isCitizenOf、livesIn、isLeaderOf 的分類之中。整個加上標記的流程如圖 3.5 所示。

每個實體特性分類中包含了許多句子，有的句子具有正向標記，代表這個句子具有該實體特性；有的句子具有負向標記，代表這個句子不包含該實體分類。對於每個實體特性，我們以分類中的句子所出現過的詞作為特徵，正、負向標記

的詞頻作為特徵值，訓練了一個二元的簡單貝氏分類器[2]（Naïve Bayes Classifier），用來區分是正向，也就是有包含實體特性的句子，是負向，也就是不包含實體特性的句子。

這些分類器的使用時機是當樣式被比對出來之後，我們知道這個樣式出現的句子以及透過樣式與特性關聯表可以查詢到候選的實體特性。對於每個實體特性都已經有一個分類器了，這讓每一個分類器對該句進行分類，分類結果為正向代表句子含有該分類器所屬的特性，反之則代表不存在該分類器所屬的實體特性。

[2]與 http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html 內的演算法相同。

表 3.4: 特性中樣式重疊狀態

數字代表各特性間的樣式重疊比例，也就是有多少比例的樣式同時出現在其他的特性的樣式中。無數字者代表無重疊或比例趨近於 0。

Property	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
(1) actedIn	1.00	0.33		0.01	0.31	0.01		0.02		0.09		
(2) created	0.21	1.00		0.01	0.17	0.11		0.01		0.10	0.02	0.03
(3) dealsWith			1.00						0.01	0.01		
(4) diedIn	0.01	0.03		1.00	0.01	0.04		0.01		0.03		0.01
(5) directed	0.51	0.44		0.01	1.00	0.07		0.03		0.06		
(6) graduatedFrom	0.01	0.16		0.02	0.04	1.00		0.01		0.02		0.04
(7) happenedIn			0.02				1.00					
(8) hasAcademicAdvisor	0.06	0.07		0.02	0.06	0.05		1.00		0.18		
(9) hasCapital			0.21	0.08					1.00			
(10) hasChild	0.05	0.09		0.01	0.02	0.01		0.03		1.00		0.13
(11) hasWonPrize	0.08	0.69			0.01			0.04		0.21	1.00	0.21
(12) holdsPoliticalPosition	0.01	0.08		0.01		0.07				0.40	0.01	1.00
(13) influences	0.07	0.13		0.01	0.02	0.03		0.13		0.35		0.13
(14) isCitizenOf	0.02	0.07		0.27	0.02	0.10		0.01		0.03		0.03
(15) isKnownFor	0.09	0.48		0.04	0.15	0.24		0.10		0.18	0.03	0.05
(16) isLeaderOf	0.01	0.58		0.18	0.01	0.41		0.01		0.04	0.11	0.11
(17) isLocatedIn		0.01	0.02	0.23		0.03		0.01		0.02		
(18) isMarriedTo	0.06	0.10		0.01	0.02	0.02		0.09		0.43		0.06
(19) isPoliticianOf	0.01	0.07		0.50	0.02	0.11		0.01		0.04		0.02
(20) livesIn	0.01	0.10		0.54	0.02	0.09		0.02		0.03		0.02
(21) participatedIn	0.02	0.01		0.02	0.01					0.07		0.01
(22) playsFor		0.04		0.02		0.13				0.02		
(23) wasBornIn	0.01	0.03		0.55	0.01	0.06		0.02		0.02		
(24) worksAt	0.02	0.35		0.04	0.06	0.63		0.02		0.03	0.03	0.06

Property	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)
(1) actedIn	0.08	0.01	0.07			0.14					0.01	0.02
(2) created	0.10	0.02	0.23	0.08		0.14	0.01	0.02		0.02	0.01	0.17
(3) dealsWith	0.01				0.07	0.01					0.01	
(4) diedIn	0.02	0.16	0.05	0.06	0.22	0.04	0.17	0.29		0.02	0.50	0.05
(5) directed	0.05	0.01	0.19			0.08	0.01	0.01		0.01	0.01	0.07
(6) graduatedFrom	0.03	0.04	0.18	0.09	0.02	0.05	0.02	0.03		0.09	0.03	0.47
(7) happenedIn												
(8) hasAcademicAdvisor	0.51	0.01	0.25		0.02	0.59	0.01	0.02			0.04	0.05
(9) hasCapital												
(10) hasChild	0.24	0.01	0.08	0.01	0.01	0.51	0.01	0.01		0.01	0.01	0.01
(11) hasWonPrize		0.04	0.68	0.64				0.04			0.03	0.64
(12) holdsPoliticalPosition	0.27	0.02	0.07	0.04		0.23	0.01	0.01				0.09
(13) influences	1.00	0.01	0.09		0.01	0.51		0.01			0.01	0.03
(14) isCitizenOf	0.04	1.00	0.11	0.22	0.20	0.07	0.36	0.39		0.06	0.33	0.13
(15) isKnownFor	0.14	0.06	1.00	0.18	0.03	0.23	0.04	0.06		0.05	0.05	0.35
(16) isLeaderOf	0.02	0.38	0.62	1.00	0.22	0.11	0.20	0.26		0.04	0.26	0.64
(17) isLocatedIn	0.01	0.12	0.04	0.08	1.00	0.03	0.11	0.17		0.01	0.28	0.03
(18) isMarriedTo	0.30	0.01	0.09	0.01	0.01	1.00	0.01	0.01		0.01	0.02	0.03
(19) isPoliticianOf	0.02	0.63	0.14	0.20	0.31	0.09	1.00	0.61		0.06	0.44	0.15
(20) livesIn	0.04	0.44	0.13	0.17	0.30	0.06	0.40	1.00		0.03	0.56	0.10
(21) participatedIn						0.07			1.00		0.02	0.02
(22) playsFor		0.03	0.05	0.01	0.01	0.01	0.02	0.02		1.00	0.02	0.11
(23) wasBornIn	0.03	0.22	0.07	0.10	0.30	0.06	0.17	0.33		0.03	1.00	0.07
(24) worksAt	0.04	0.06	0.34	0.19	0.03	0.07	0.04	0.05		0.11	0.05	1.00

第四章　實驗結果與分析

本章進行了以樣式偵測特性的實驗，以 Wikipedia 的文章作為模擬內容串流文件、以 YAGO 的資料作為答案、以 PATTY 提供的樣式對 Wikipedia 的文章進行偵測，並介紹評估的標準以及對實驗結果的分析。

4.1　測試資料集

為了模擬內容串流文件，本研究以 Wikipedia 的條目文章作為內容串流中的文件，自 2013 年 3 月的 Wikipedia Dumps[1]擷取文件。

由於一開始並無文件中包含哪些實體特性的正確資料，而 YAGO 的 YAGO Facts 提供了 <subject, property, object> 的資訊，因此我們使用 YAGO Facts 的 propety 作為參考答案，並將 subject 連結至 Wikipedia 的文章，這樣就有每篇文件具有哪些特性的參考答案。

樣式則是使用 PATTY 提供的樣式釋義，其包含了知識庫定義的關係，即本研究擬偵測之實體特性，以及可用以表達該特性之樣式集。實驗採用的是其中的 YAGO 關係（YAGO Relations），一共有 25 組關係，但其中一組關係（YAGO:Produced）在 YAGO 中沒有找到對映的 Wikipedia 條目，因此僅對餘下 24 組關係進行實驗。

接下來只留下存在這 24 種特性的 Wikipedia 條目，為了防止 YAGO Facts 中存在某特性但文章中卻根本沒有提及的狀況發生，我們只留下 <subject,property,object> 中 subject 條目內有出現 object 字詞片段的特性留下，若 object 內的字根本沒有出現在文章中，則捨去這個特性。經過處理後，最後共有 334,469 篇條目作為測試資料集，表 4.1 顯示了各個特性在測試資料集中的數量。

[1]http://dumps.wikimedia.org/

表 4.1: 各個特性於測試資料集中出現的文章數

Property	Number of documents
actedIn	3971
created	8665
dealsWith	111
diedIn	11729
directed	2036
graduatedFrom	10474
happenedIn	1792
hasAcademicAdvisor	628
hasCapital	363
hasChild	2113
hasWonPrize	8916
holdsPoliticalPosition	1190
influences	869
isCitizenOf	10974
isKnownFor	73
isLeaderOf	1397
isLocatedIn	216772
isMarriedTo	3555
isPoliticianOf	170
livesIn	7661
participatedIn	346
playsFor	42751
wasBornIn	47141
worksAt	1981

表 4.2 統計了在不同的信心值下與不同的樣式歧義下，樣式的數量以及含有這些樣式的文件累計總數。由表中可以看出其實當樣式歧義度超過 5 之後所涵蓋的累計文件增長速度已經趨於平緩，因此本實驗最多將只採用歧義度不大於 5 的樣式進行。

由於資料集內的 Wikipeida 條目皆是描寫實體，在假設文章的內容都是以描寫該實體的前提下，可以利用 YAGO Simple Types 作為該文描述實體之類型，搭配樣式的領域（Domain）資訊輔助偵測。

而第 3 章中所提到需要利用過去的資料進行統計、訓練分類器，則是將測試資料集分為五等份，以其中四等份作為過去的資料，並對其進行計算可信賴度，以及利用字詞作為特徵，對每個特性訓練分類器。餘下的一份進行測試，最後將五份結果平均為實驗結果。

表 4.2: 樣式總數量、出現數量、累計涵蓋文章與比例於不同信心值之統計

歧義度	信心值 > 0				信心值 > 0.7			
	樣式數	出現	累計涵蓋文章	比例	樣式數	出現	累計涵蓋文章	比例
1	11381	7267	250737	74.97	8913	5854	244888	73.22
2	4778	2976	275736	82.44	4175	2697	272395	81.44
3	1255	886	278820	83.36	1048	745	274265	82.00
4	666	503	281256	84.09	600	458	276782	82.75
5	635	465	283830	84.86	605	442	279603	83.60
6	77	64	283858	84.87	68	57	279648	83.61
7	132	100	284128	84.95	131	100	280096	83.74
8	49	43	284275	84.99	49	43	280342	83.82
9	26	25	284295	85.00	26	25	280385	83.83
10	13	11	284297	85.00	13	11	280387	83.83
11	12	9	284299	85.00	12	9	280391	83.83
12	2	2	284299	85.00	2	2	280391	83.83
13	0	0	284299	85.00	0	0	280391	83.83
14	2	2	284299	85.00	2	2	280391	83.83
15	0	0	284299	85.00	0	0	280391	83.83
16	0	0	284299	85.00	0	0	280391	83.83
17	3	3	284299	85.00	3	3	280391	83.83
總計	19031	12356	–	–	15647	10448	–	–

歧義度	信心值 > 0.8				信心值 > 0.9			
	樣式數	出現	累計涵蓋文章	比例	樣式數	出現	累計涵蓋文章	比例
1	5951	4029	222371	66.48	1879	1121	155402	46.46
2	2328	1697	251552	75.21	316	206	163265	48.81
3	683	487	258715	77.35	187	110	170820	51.07
4	473	355	261633	78.22	67	47	171978	51.42
5	278	243	262321	78.43	18	16	173397	51.84
6	50	43	262364	78.44	10	8	173457	51.86
7	52	41	262452	78.47	12	8	174396	52.14
8	13	15	263419	78.76	2	2	174476	52.17
9	12	12	263514	78.79	8	8	175287	52.41
總計	9840	6922	–	–	2499	1526	–	–

4.2 評估標準

在偵測系統的效率方面，我們以每顆核心在每分鐘內可以處理多少文件來評估，以及要達到 TREC 知識庫加速競賽的每小時處理約 100,000 篇的規模需要多少運算核心或機器。

在偵測系統的效能方面，我們希望了解對每一個實體特性，利用樣式自文章中偵測該特性的效能。因此，以精確率（Precision）、召回率（Recall）、F_1 分數（F_1 Score）進行評估。精確率公式如式 4.1，召回率公式如式 4.2。

$$Precision = \frac{|\{relevant\ documents\} \cap \{retrived\ documents\}|}{|\{retrived\ documents\}|} \tag{4.1}$$

$$Recall = \frac{|\{relevant\ documents\} \cap \{retrived\ documents\}|}{|\{relevant\ documents\}|} \quad (4.2)$$

實驗流程中樣式比對、特性消歧義等步驟皆以句子為單位進行實驗，本因針對每個句子的偵測結果進行評估，但由於缺乏實例層級的答案資訊，僅有透過 YAGO 實體引申至 Wikipedia 條目文章的特性資訊，因此將條目文章中所有的句子所產生的實體特性皆視為屬於該條目所指之實體，並以條目為單位進行精確率與召回率的計算。

以文章為單位進行計算，遇到文章中句子所產生的實體特性不屬於文章所指之實體，又或者在別的文章中的句子產生了屬於文章所指之實體的特性，遇到這兩種狀況皆無法處理與評估。

對於一個實體特性，經過偵測之後會有被標為有此特性的文件與無此特性的文件。其中，相關的文件（Relevant documents）即真正存在該特性之文件；尋回的文件（Retrived documents）即偵測系統標記為擁有此特性之文件。精確率評估在尋回的文件之中有多少文件真正存在該特性；召回率評估在真正擁有該特性的文件中有多少被系統偵測到。

F_1 分數則是綜合評估精確率與召回率，計算方式如式 4.3，為精確率與召回率的調和平均。

$$F_1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4.3)$$

除了評估個別特性的效能之外，以宏觀平均（Macro Average）及微觀平均（Mirco Average）分別計算精確率與召回率及 F_1 分數來評估整體效能。以精確率為例，宏觀平均的計算如式 4.4，而微觀平均的計算則如式 4.5。

$$Macro\ Avg\ Precision = \frac{\sum_i^n precision_i}{n} \quad (4.4)$$

$$Mirco\ Avg\ Precision = \frac{\sum_i^n |\{relevant\ documents\}_i \cap \{retrived\ documents\}_i|}{\sum_i^n |\{retrived\ documents\}_i|} \quad (4.5)$$

4.3 實驗結果

4.3.1 效率

在圖 3.2的偵測流程之中，本實驗將其拆解為兩個部份來評估效能。前半部份為樣式比對的效率，後半部份為特性消歧義的效率。

樣式比對的效率在時脈為 2.5GHz 的中央處理器為每顆核心每分鐘 592 份文件，特性消歧義的效率在同樣條件下為每顆核心每分鐘 2750 份文件。要在一小時內處理 100,000 文件僅需要 3 顆核心即可完成。而本實驗使用一台配備時脈 2.5GHz、24 核心中央處理器的工作站僅需半小時即可處理完全部約 330,000 份文件。

4.3.2 原始效能

依照圖 3.2 的流程，但在樣式篩選的部分僅篩選了無歧義度以及歧義度不大於 5 的樣式，而無特性消歧義的結果如表 4.3。表中左半部是每一個特性使用無歧義度，也就是歧義度為 1 的樣式，進行特性偵測的結果，由於沒有歧義問題，所以無需進行特性消歧義。

右半部則使用了歧義度不大於 5 的樣式，也就是包含了無歧義、歧義度 2 至 5 的樣式，而不進行特性消歧義，可以看到的結果是，由於沒有進行特性消歧義而的對每篇文章回答盡可能多的特性，因此召回率提升，但精確率也因此下降，F_1 分數則因為精確率降太多而變低。

由此也可以看出太寬鬆地依據樣式回報特性會降低效能，後續的實驗將陸續加入樣式篩選與特性消歧義的方法，嘗試改善效能。

4.3.3 樣式篩選

信心值

我們利用 PATTY 提供的信心值進行樣式的篩選，選定了門檻值分別為 0（沒有門檻）、0.7、0.8 與 0.9，只使用樣式信心值大於門檻值的樣式進行偵測實驗，其結果的宏觀平均與微觀平均 F_1 分數如圖 4.1 所示。

表 4.3: 使用無歧義之樣式與歧義度 ≤5 之樣式實驗結果

Property	無歧義			歧義度 ≤5		
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score
actedIn	0.0435	0.5163	0.0802	0.0281	0.7976	0.0543
created	0.0823	0.8171	0.1495	0.0558	0.9500	0.1055
dealsWith	0.0034	1.0000	0.0067	0.0022	1.0000	0.0043
diedIn	0.0622	0.3688	0.1065	0.0551	0.6468	0.1015
directed	0.0393	0.2867	0.0691	0.0290	0.9403	0.0562
graduatedFrom	0.1472	0.4752	0.2248	0.1130	0.7857	0.1976
happenedIn	0.1323	0.6676	0.2209	0.1315	0.6731	0.2201
hasAcademicAdvisor	0.0281	0.4329	0.0528	0.0129	0.6892	0.0254
hasCapital	0.0530	0.3430	0.0917	0.0059	0.4582	0.0116
hasChild	0.0284	0.6302	0.0544	0.0182	0.9724	0.0358
hasWonPrize	0.1212	0.0110	0.0202	0.0969	0.0738	0.0838
holdsPoliticalPosition	0.0302	0.7598	0.0580	0.0151	0.8945	0.0298
influences	0.0153	0.8271	0.0301	0.0082	0.9559	0.0162
isCitizenOf	0.1199	0.1295	0.1245	0.0879	0.3901	0.1434
isKnownFor	0.0021	0.4464	0.0042	0.0008	0.9380	0.0016
isLeaderOf	0.0468	0.0233	0.0311	0.0193	0.4696	0.0371
isLocatedIn	0.8452	0.5208	0.6445	0.7709	0.5618	0.6499
isMarriedTo	0.0383	0.6623	0.0724	0.0274	0.9665	0.0534
isPoliticianOf	0.0163	0.2355	0.0305	0.0037	0.7025	0.0073
livesIn	0.0747	0.0553	0.0635	0.0673	0.4438	0.1169
participatedIn	0.0223	0.6549	0.0431	0.0202	0.7151	0.0393
playsFor	0.5239	0.5181	0.5210	0.3829	0.6096	0.4703
wasBornIn	0.2862	0.0825	0.1280	0.2523	0.3962	0.3083
worksAt	0.0439	0.3563	0.0781	0.0210	0.8650	0.0409
Macro Average	0.1169	0.4509	0.1857	0.0927	0.7040	0.1639
Micro Average	0.2257	0.4356	0.2973	0.1255	0.5603	0.2051

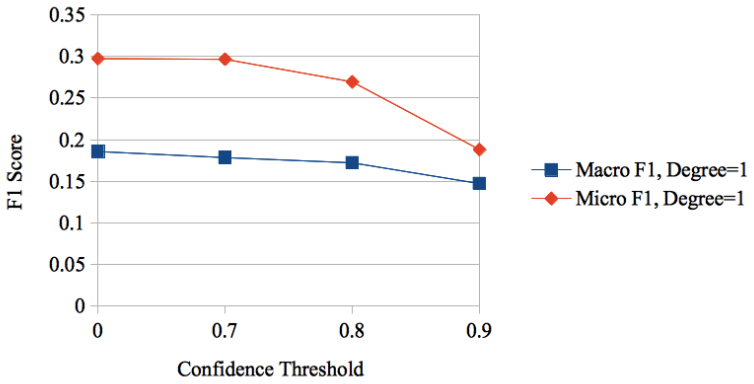

圖 4.1: 無歧義度下 F1 分數平均隨信心值變化圖

就整體效能而言，設定信心值門檻值並沒有辦法提升系統效能，反而使效能變差。這是由於設定門檻值後，通過門檻值的樣式數量減少，導致召回率下降太多，到門檻值為 0.9 時，有部分的特性（hasCapital、isKnownFor）已經沒有可用的樣式。但部分的特性 F_1 分數因精確率比較大幅度的提升而略微上升，表 4.4 中，信心值門檻由 0.8 昇至 0.9，特性 actedIn、created、diedIn、hasAcademicAdvisor、holdsPoliticalPosition、influences、isMarriedTo 等的精確率皆有較大幅度的提升。

表 4.4: 信心值門檻為 0.8 與 0.9 各特性效能

Property	Confidence > 0.8			Confidence > 0.9		
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score
actedIn	0.0490	0.3196	0.0850	0.0833	0.1032	0.0922
created	0.1005	0.6682	0.1747	0.1564	0.2208	0.1831
dealsWith	0.0128	0.8935	0.0252	0.0114	0.5805	0.0224
diedIn	0.0603	0.3479	0.1028	0.1191	0.2179	0.1540
directed	0.0258	0.1403	0.0436	0.1693	0.0145	0.0267
graduatedFrom	0.1022	0.2326	0.1420	0.1129	0.0736	0.0891
happenedIn	0.0638	0.1374	0.0872	0.3010	0.0139	0.0266
hasAcademicAdvisor	0.0288	0.3254	0.0530	0.0468	0.2426	0.0785
hasCapital	0.1117	0.0113	0.0206	0.0000	0.0000	–
hasChild	0.0297	0.5289	0.0563	0.0337	0.1872	0.0572
hasWonPrize	0.1212	0.0110	0.0202	0.1193	0.0008	0.0015
holdsPoliticalPosition	0.0350	0.7283	0.0667	0.0437	0.4819	0.0801
influences	0.0195	0.7098	0.0379	0.0616	0.3715	0.1057
isCitizenOf	0.1131	0.0790	0.0930	0.1368	0.0293	0.0483
isKnownFor	0.0024	0.4068	0.0047	0.0000	0.0000	–
isLeaderOf	0.0419	0.0120	0.0187	0.0264	0.0056	0.0092
isLocatedIn	0.8806	0.4001	0.5502	0.9066	0.1853	0.3078
isMarriedTo	0.0405	0.5531	0.0755	0.0935	0.2915	0.1416
isPoliticianOf	0.0142	0.1527	0.0260	0.0118	0.0720	0.0202
livesIn	0.0709	0.0466	0.0563	0.0643	0.0080	0.0143
participatedIn	0.0220	0.5917	0.0423	0.0266	0.4552	0.0503
playsFor	0.4966	0.2762	0.3550	0.5432	0.0121	0.0236
wasBornIn	0.3194	0.0736	0.1196	0.3073	0.0313	0.0569
worksAt	0.0435	0.1779	0.0699	0.0404	0.0524	0.0456
Macro Average	0.1169	0.3260	0.1721	0.1423	0.1521	0.1471
Micro Average	0.2331	0.3190	0.2694	0.3213	0.1328	0.1880

可信賴度

透過了過去的統計資料，也就是過去樣式出現時具有特性的比例，對可信賴度進行實驗。設定了比例為 0、0.3 與 0.5 三個門檻值。

實驗的結果呈現如圖 4.2，可以看出當門檻值調整至 0.3 時，整體系統效能有提升，但再向上調整時就沒有太多上升。原因是因為當門檻值提升至 0.5 時，有些特性（hasCapital、holdsPoliticalPosition、isKnownFor、isLeaderOf、isPoliticianOf、participatedIn）已經沒有可用的樣式。可信賴度的門檻設定留下了在過去文件中對精確率來說比較好的樣式，由表 4.5 也可以看到精確率隨著門檻值提高而升高，

當然因為使用的樣式越來越少，所以召回率也隨之下降。

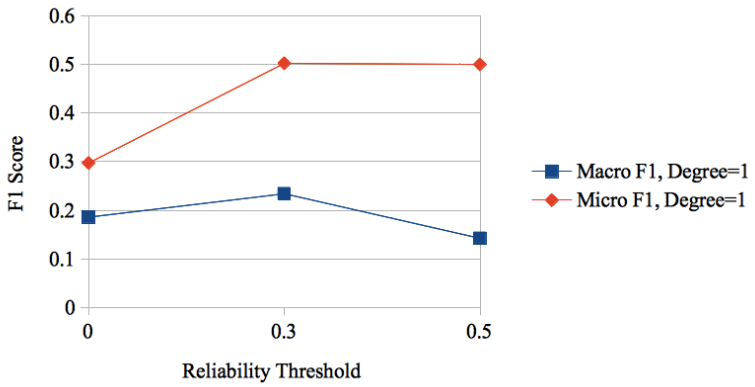

圖 4.2: 無歧義度下 F1 分數平均隨可信賴度變化圖

表 4.5: 可信賴度門檻與各特性效能

Property	可信賴度 > 0			可信賴度 > 0.3			可信賴度 > 0.5		
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score
actedIn	0.0435	0.5163	0.0802	0.4038	0.3380	0.3680	0.5109	0.1773	0.2632
created	0.0823	0.8171	0.1495	0.3122	0.5858	0.4073	0.4790	0.4485	0.4632
dealsWith	0.0034	1.0000	0.0067	0.2499	0.7309	0.3725	0.0250	0.0037	0.0065
diedIn	0.0622	0.3688	0.1065	0.3737	0.1130	0.1736	0.4004	0.0145	0.0280
directed	0.0393	0.2867	0.0691	0.5506	0.1305	0.2110	0.6140	0.1133	0.1913
graduatedFrom	0.1472	0.4752	0.2248	0.3869	0.2307	0.2891	0.4965	0.0521	0.0943
happenedIn	0.1323	0.6676	0.2209	0.5055	0.3098	0.3842	0.7744	0.1833	0.2964
hasAcademicAdvisor	0.0281	0.4329	0.0528	0.3272	0.0513	0.0887	0.0333	0.0014	0.0027
hasCapital	0.0530	0.3430	0.0917	0.0000	0.0000	–	0.0000	0.0000	–
hasChild	0.0284	0.6302	0.0544	0.2886	0.2432	0.2640	0.3607	0.0267	0.0497
hasWonPrize	0.1212	0.0110	0.0202	0.3697	0.0027	0.0054	0.5167	0.0008	0.0015
holdsPoliticalPosition	0.0302	0.7598	0.0580	0.0000	0.0000	–	0.0000	0.0000	–
influences	0.0153	0.8271	0.0301	0.2137	0.2900	0.2461	0.1952	0.0103	0.0195
isCitizenOf	0.1199	0.1295	0.1245	0.3640	0.0118	0.0229	0.5043	0.0039	0.0077
isKnownFor	0.0021	0.4464	0.0042	0.0000	0.0000	–	0.0000	0.0000	–
isLeaderOf	0.0468	0.0233	0.0311	0.0000	0.0000	–	0.0000	0.0000	–
isLocatedIn	0.8452	0.5208	0.6445	0.8527	0.5206	0.6465	0.8665	0.5188	0.6490
isMarriedTo	0.0383	0.6623	0.0724	0.3307	0.2113	0.2579	0.5033	0.0523	0.0947
isPoliticianOf	0.0163	0.2355	0.0305	0.1082	0.0191	0.0324	0.0000	0.0000	–
livesIn	0.0747	0.0553	0.0635	0.3983	0.0110	0.0214	0.4500	0.0013	0.0026
participatedIn	0.0223	0.6549	0.0431	0.1000	0.0114	0.0204	0.0000	0.0000	–
playsFor	0.5239	0.5181	0.5210	0.6013	0.5157	0.5552	0.6683	0.4971	0.5701
wasBornIn	0.2862	0.0825	0.1280	0.3951	0.0588	0.1024	0.4841	0.0083	0.0162
worksAt	0.0439	0.3563	0.0781	0.3516	0.1095	0.1670	0.4951	0.0271	0.0514
Macro Average	0.1169	0.4509	0.1857	0.3118	0.1873	0.2340	0.3491	0.0892	0.1421
Micro Average	0.2257	0.4356	0.2973	0.7046	0.3908	0.5027	0.8014	0.3638	0.5004

歧義度

由表 4.2 觀察，當歧義度不超過 5 時所能涵蓋的文件數量就已經非常多了，因此這裡測試了歧義度累計由 1 到 5，也就是使用歧義度 1 的樣式、使用歧義度 1 至 2 的樣式、使用歧義度 1 至 3 的樣式、使用歧義度 1 至 4 的樣式、使用歧義度

1 至 5 的樣式一共 5 組實驗觀察系統效能的變化結果如圖 4.3，由於使用越來越多的樣式，因此召回率快速提高，但也因此有了歧義問題，使得精確率下降，也讓 F_1 分數下降。由於有歧義問題，後續的實驗將嘗試消歧義的方法與分析結果。

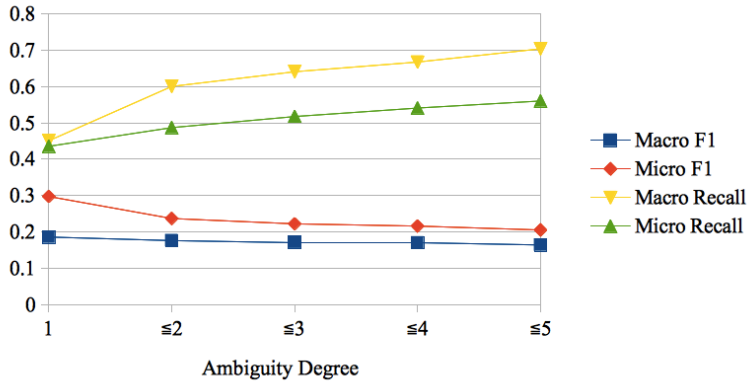

圖 4.3: F1 分數與召回率隨歧義度變化圖

綜合討論

經過了對信心值、可信賴度與歧義度的實驗，可以知道透過篩選信心值與可信賴度可以提升精確率，但同時會對召回率造成影響，而使用歧義度高的樣式時可以提升召回率，但也同時因為歧義問題造成精確率的下降。

綜合了信心值、可信賴度、歧義度三項因素，整體效能表現排序如表 4.6，前 5 名中的可信賴度門檻 0.3，前 4 名皆為信心值門檻 0，歧義度依次遞減。雖然單獨觀察歧義度時，使用歧義度越大的樣式不利於整體效能表現，但有利於召回率。配合可信賴度篩選，可以減緩隨著歧義度增加而下降的精確率的下降幅度，使得整體效能的表現有所提升。

雖然可信賴度門檻在 0.3 時的整體表現比起門檻值為 0 來的好，但如同表 4.5 所示，門檻值為 0.3 時，hasCapital、holdsPoliticalPosition、isKnownFor、isLeaderOf 等特性已經無樣式可用，因此無法偵測這些特性。由於會造成部份特性無法偵測，後續的實驗中將可信賴度門檻固定於 0。

表 4.6: 樣式篩選整體效能表現（宏觀平均）

排名	信心值門檻	可信賴度門檻	歧義度	Precision	Recall	F_1 Score
1	0	0.3	≤5	0.2916	0.2754	0.2833
2	0	0.3	≤4	0.2962	0.2690	0.2819
3	0	0.3	≤3	0.2851	0.2593	0.2716
4	0	0.3	≤2	0.2911	0.2353	0.2602
5	0.7	0.3	≤5	0.2823	0.2015	0.2351

4.3.4 特性消歧義

實體類型資訊

由於特性具有領域與範圍，這裡假設文章中的語句所描述的對象都是條目名稱所指之實體，便可以利用實體的類型資訊與特性領域進行比對，將不合類型的特性消除掉。例如 Wikipedia 條目 Jahsha Bluntt 是描述實體 Jahsha 這位籃球員，文章中出現了 actedIn 的樣式，但 actedIn 的領域是演員，因此可將 actedIn 消除掉。加入實體類型資訊後，將不可能屬於實體的特性消除掉，在評估方面，便不會使召回率下降，還有機會可以提升精確率。

圖 4.4 便是加入實體類型資訊前後對不同歧義度的效能比較結果。加入實體類型資訊後，使得精確率有大幅提升，例如以使用歧義度不大於 2 的樣式時，如表 4.7 數據所示，actedIn 的 False Postive 從 13,560 降至 489、happendIn 的 False Postive 從 1,567 降至 89，都有大幅度的精確率提升，甚至是本來精確率就已經很高的 isLocatedIn 都有相當提升（False Postive 從 4,771 下降至 455）。

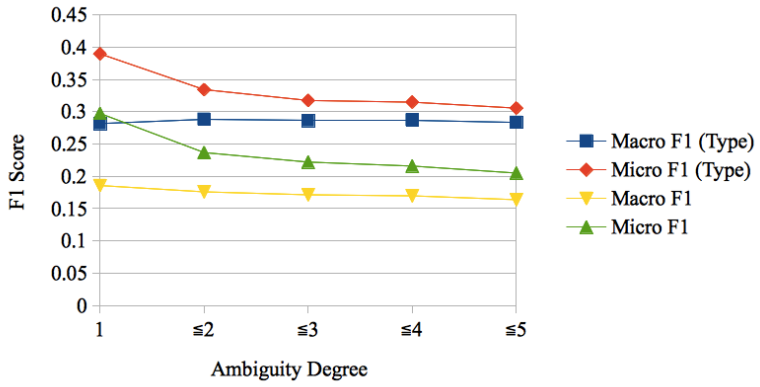

圖 4.4: 加入實體類型資訊前後效能比較

表 4.7: 樣式歧義度 ≤2 時有無加入實體類型資訊各特性效能

Property	無實體類型資訊			加入實體類型資訊		
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score
actedIn	0.0373	0.6691	0.0707	0.5300	0.6691	0.5915
created	0.0729	0.8853	0.1348	0.1072	0.8853	0.1912
dealsWith	0.0022	1.0000	0.0043	0.0049	1.0000	0.0098
diedIn	0.0490	0.4720	0.0888	0.0995	0.4720	0.1643
directed	0.0482	0.9067	0.0915	0.0826	0.9067	0.1514
graduatedFrom	0.1336	0.6335	0.2206	0.1881	0.6335	0.2900
happenedIn	0.1315	0.6731	0.2201	0.7641	0.6731	0.7157
hasAcademicAdvisor	0.0158	0.5892	0.0308	0.0258	0.5892	0.0494
hasCapital	0.0059	0.4582	0.0116	0.0064	0.4582	0.0126
hasChild	0.0219	0.9453	0.0429	0.0357	0.9453	0.0689
hasWonPrize	0.1160	0.0111	0.0203	0.1782	0.0111	0.0209
holdsPoliticalPosition	0.0180	0.8733	0.0352	0.0284	0.8733	0.0549
influences	0.0108	0.9184	0.0214	0.0195	0.9184	0.0383
isCitizenOf	0.0871	0.2361	0.1273	0.1667	0.2361	0.1954
isKnownFor	0.0013	0.7799	0.0027	0.0016	0.7799	0.0033
isLeaderOf	0.0252	0.0900	0.0393	0.0386	0.0900	0.0541
isLocatedIn	0.8287	0.5401	0.6540	0.9816	0.5401	0.6968
isMarriedTo	0.0349	0.9561	0.0674	0.0558	0.9561	0.1055
isPoliticianOf	0.0052	0.4960	0.0102	0.0125	0.4960	0.0243
livesIn	0.0729	0.1141	0.0890	0.1196	0.1141	0.1168
participatedIn	0.0203	0.7012	0.0395	0.0233	0.7012	0.0451
playsFor	0.4526	0.5311	0.4887	0.5489	0.5311	0.5398
wasBornIn	0.2459	0.1916	0.2154	0.4806	0.1916	0.2740
worksAt	0.0355	0.7368	0.0678	0.0520	0.7368	0.0971
Macro Average	0.1030	0.6003	0.1759	0.1896	0.6003	0.2882
Micro Average	0.1564	0.4870	0.2367	0.2544	0.4870	0.3342

特性篩選

當使用具有特性歧義的樣式時，必須設法消除歧義。而這裡的特性選擇是多選多的問題，以下實驗了幾種挑選特性的策略：(1) 只選擇一個特性、(2) 篩選門檻 0.5、(3) 篩選門檻 0.75、(4) 全選。

其中 (1)、(2)、(3) 利用過去文件所計算之可信賴度，(1) 選擇正確率最高的特性，(2) 與 (3) 將可信賴度正規化後，選擇門檻值以上之特性。

圖 4.5、4.6 呈現實驗結果，(1) 只選擇了一個忽略了可能有複數個特性存在，(4) 全選則太寬鬆，以至於精確率降太多。

由巨觀平均的角度來看，在使用任何歧義度之樣式的情況下，設定 0.5 的正確篩選門檻的表現都來得比其他策略還要好，基本上都是精確率小幅提升，而召回率沒有太大改變。是一個介於 (1) 和 (4) 之間的折衷策略。

從微觀平均的角度來看，則是篩選門檻 0.75 的表現比較好，越高的篩選門檻使得能夠被選擇的特性數量減少，False Postive 也隨之減少，特性 hasChild 的 False Postive 較篩選門檻 0.5 少了約一萬份的文件，佔整體降幅約一半，使得門檻 0.75 的微觀平均表現較好。

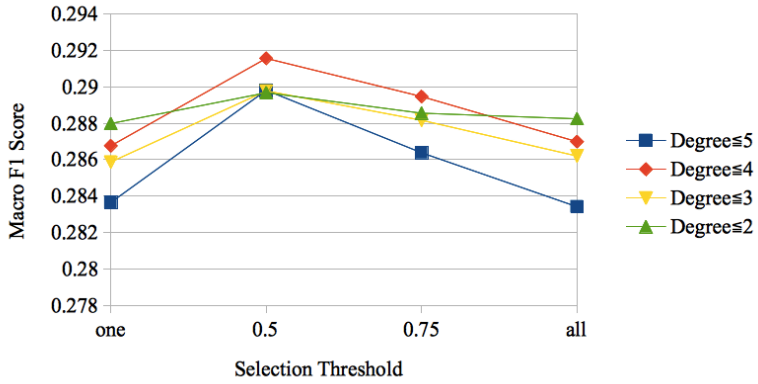

圖 4.5: 不同歧義度下篩選策略效能表現（宏觀平均）

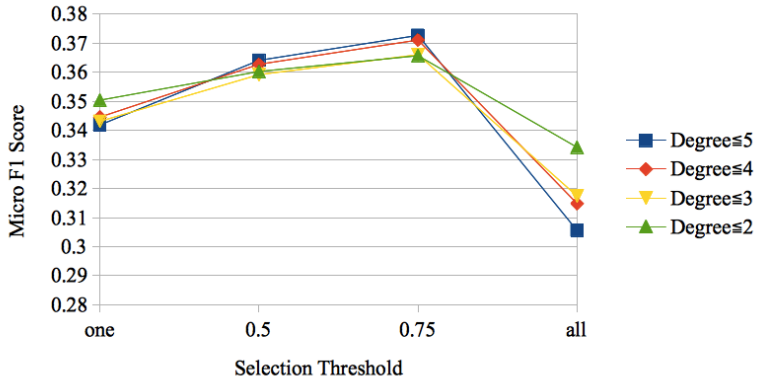

圖 4.6: 不同歧義度下篩選策略效能表現（微觀平均）

簡單貝氏分類器

由於樣式有出現仍不一定具有特性，我們認為這與樣式出現的前後文可能有關聯，因此針對每個特性，以過去的文件訓練了 24 個簡單貝氏分類器。當樣式出現時，以分類器將樣式出現的句子分為存在或不存在該特性。

圖 4.7 顯示了使用簡單貝氏分類器對不同歧義度的效能比較。而表 4.8 則是歧義度不小於 5、使用實體類型資訊、樣式篩選門檻 0.5 時有無使用分類器的效能比較。

圖 4.7 與表 4.8 中可以看到使用了簡單貝氏分類器後整體的效能有了大幅度的提升。從個別特性來看，大多數的特性如 created、dealsWith、diedIn、directed、graduatedFrom、happendIn、hasAcademicAdvisor、hasCapital、hasChild、hasWonPrize、holdsPoliticalPosition、influences、isCitizenOf、isMarriedTo、livesIn、participatedIn、

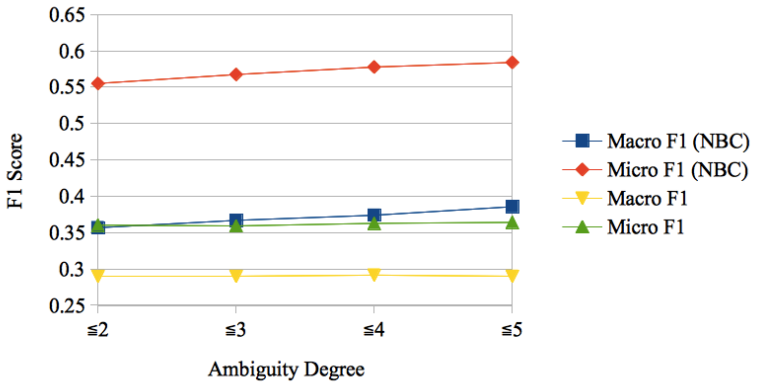

圖 4.7: 使用簡單貝氏分類器前後效能比較

playsFor、worksAt 等的精確度都有了大幅的提升。但這些精確度的增加主要並非來自 True Postive 的提升，而是 False Postive 的大幅下降，事實上 True Postive 的數量也減少了，而使得 False Negative 上升，因此精確率大幅提升但召回率下降。例如 diedIn 的 False Postive 由 10,299 降低至 1,344，True Postive 由 1,276 降至 850，而 False Negative 則是由 1.032 升至 1,458。由 False Postive 的大幅下降可以知道，使用了前後文資訊使得判斷變得嚴格，大大降低了誤報為 Postive 的數量。

而 isKnownFor、isPoliticianOf 則是因為這些特性訓練時有正向標記的的句子太少，以至於分類時完全沒有正向的結果，所以精確率與召回率皆為 0。表 4.1 也顯示這兩個特性的文件數與其他特性比起來算是比較少了些。

綜合討論

在使用簡單貝氏分類器之前，基於樣式篩選，再搭配上實體類型資訊與特性篩選，整體效能表現排序如表 4.9 所示，可以看出加入實體類型資訊再搭配篩選門檻為 0.5 的開選策略後的整體效能表現較佳，篩選門檻為 0.5 時，使用歧義度不大於 4 的樣式會有最好的效果。

使用簡單貝氏分類器後的表現排序則如表 4.10 所示，排名前 4 都是使用高歧義度的樣式，使召回率相對較高，簡單貝氏分類器主要大幅提升了精確率。而篩選策略 (4) 是全選，相較於精確率更偏好召回率，在表中的排名 1、2 之間也反映了這種傾向。

表 4.8: 樣式歧義度 ≤5 時有無使用簡單貝氏分類器各特性效能

Property	不使用分類器			使用分類器		
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score
actedIn	0.5332	0.7420	0.6205	0.5985	0.5495	0.5730
created	0.0904	0.9429	0.1651	0.3036	0.7617	0.4342
dealsWith	0.0107	1.0000	0.0211	0.3724	0.6352	0.4696
diedIn	0.1145	0.5688	0.1906	0.3868	0.3807	0.3837
directed	0.0784	0.9186	0.1445	0.4229	0.7072	0.5293
graduatedFrom	0.1706	0.7783	0.2798	0.4446	0.6204	0.5180
happenedIn	0.7641	0.6731	0.7157	0.9916	0.5238	0.6855
hasAcademicAdvisor	0.0393	0.5086	0.0730	0.4667	0.0130	0.0253
hasCapital	0.0064	0.4582	0.0126	0.2336	0.3891	0.2919
hasChild	0.0351	0.9603	0.0678	0.3155	0.6858	0.4322
hasWonPrize	0.1165	0.0727	0.0895	0.7733	0.0065	0.0129
holdsPoliticalPosition	0.0348	0.8548	0.0669	0.4324	0.0271	0.0511
influences	0.0199	0.9224	0.0390	0.1878	0.6360	0.2900
isCitizenOf	0.1631	0.2843	0.2073	0.5582	0.0241	0.0461
isKnownFor	0.0027	0.4934	0.0054	0.0000	0.0000	–
isLeaderOf	0.1055	0.1680	0.1296	0.2000	0.0008	0.0015
isLocatedIn	0.9804	0.5614	0.7139	0.9917	0.5499	0.7075
isMarriedTo	0.0529	0.9626	0.1003	0.2380	0.5994	0.3407
isPoliticianOf	0.0281	0.4393	0.0528	0.0000	0.0000	–
livesIn	0.1453	0.1740	0.1584	0.6474	0.0097	0.0191
participatedIn	0.0232	0.6970	0.0450	0.3000	0.0074	0.0144
playsFor	0.5055	0.6091	0.5525	0.8742	0.6043	0.7146
wasBornIn	0.4782	0.3779	0.4221	0.5231	0.2856	0.3695
worksAt	0.0538	0.5618	0.0981	0.4770	0.1137	0.1836
Macro Average	0.1897	0.6137	0.2898	0.4475	0.3388	0.3856
Micro Average	0.2740	0.5426	0.3641	0.7428	0.4813	0.5841

表 4.9: 樣式篩選與特性消歧義整體效能表現（無分類器，宏觀平均）

排名	信心值門檻	歧義度	類型資訊	篩選策略	Precision	Recall	F_1 Score
1	0	≤4	有	(2)	0.1927	0.5986	0.2915
2	0	≤5	有	(2)	0.1896	0.6137	0.2898
3	0	≤3	有	(2)	0.1923	0.5869	0.2897
4	0	≤2	有	(2)	0.1954	0.5590	0.2896
5	0	≤4	有	(3)	0.1934	0.5750	0.2894

表 4.10: 樣式篩選與特性消歧義整體效能表現（宏觀平均）

排名	信心值門檻	歧義度	類型資訊	篩選策略	分類器	Precision	Recall	F_1 Score
1	0	≤5	有	(4)	有	0.4266	0.3531	0.3864
2	0	≤5	有	(2)	有	0.4474	0.3387	0.3856
3	0	≤5	有	(1)	有	0.4321	0.3457	0.3841
4	0	≤5	有	(3)	有	0.4447	0.3312	0.3796
5	0	≤4	有	(4)	有	0.4230	0.3423	0.3784

4.3.5 錯誤分析

在整個過程我們分析了錯誤可能的來源，由於整體或個別特性偵測的效能以精確率、召回率作為評估標準，能夠影響這兩個指標的是 True Postive、False Postive 以及 False Negative 三者。True Postive 是經過偵測流程後，認為存在特性且實際上存在；False Postive 是經過偵測流程後，認為存在特性但實際上不存在；False Negative 是經過偵測流程後，認為不存在特性但實際上存在。

False Postive 的成因基本上可以分成兩類：(1) 由歧義性造成、(2) 樣式出現但不具有特性。(1) 是當有候選樣式時，有機會回答正確但沒有成功選對特性造成的錯誤。這是由於假設有樣式出現就一定有特性資訊，一定要進行選擇，但樣式雖然出現，但有可能根本就不存在特性資訊，也就是 (2)。

而 False Negative 的來源除了偵測流程中的誤判，例如在特性篩選策略中低於門檻值或被分類器判為不存在特性之外，還有一個固定的部份是來自於樣式的覆蓋率不足所致。表 4.2 中，當信心值門檻為 0，使用歧義度不大於 5 的樣式時，最多可以涵蓋 283,830 篇文章，佔測試資料集約 85%，這同時意味著另外約 15% 的文章中的特性是目前無法偵測的。

由於大部分實驗中的精確率相較於召回率偏低，因此我們又分析了影響精確率的 False Postive 的成因所佔的比例。圖 4.8 分析了使用分類器前後個特性的 False Postive 中的成分，發現大部分 False Postive 的來源皆屬於第二類樣式出現但不具有特性。圖中的黑、白線條就是使用分類器前後第二類錯誤佔所有 False Postive 中的比例。使用了分類器後第二類除了部份特性如 hasAcademicAdvisor 之外，雖然所佔比例有所下降，但仍然是最主要的錯誤來源，要如何有效地克服這些錯誤是未來的研究課題之一。

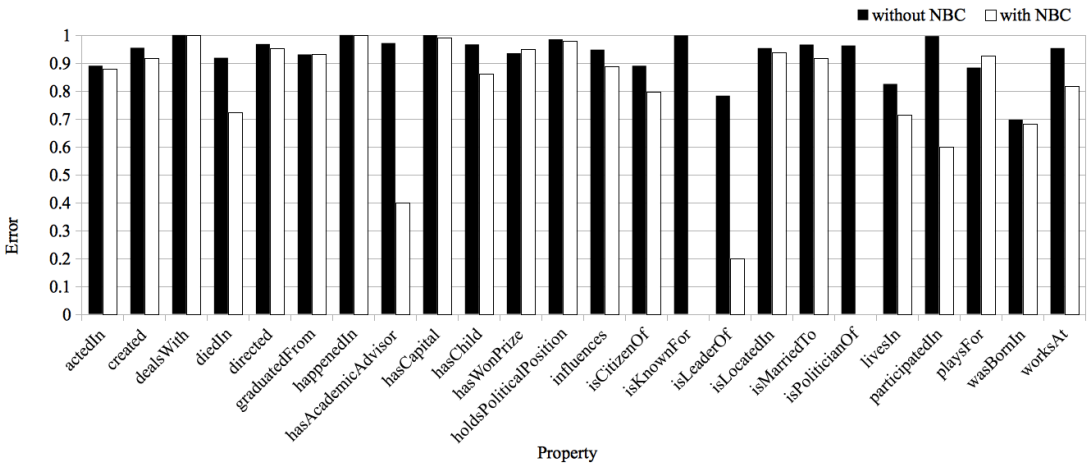

圖 4.8: 使用簡單貝氏分類器前後錯誤分析

第五章　結論與未來展望

5.1　結論

有別於過去的研究多半著重於於內容串流中找尋實體相關的文件，本研究專注於於內容串流中尋找實體的特性，提出了於基於樣式於內容串流中偵測實體特性的方式。自內容串流中比對樣式，透過實體特性與樣式間的關聯，快速地偵測文件是否包含特定的實體樣式。由於偵測流程文件與文件是互相獨立的，整個偵測的流程可平行化，具備可擴展性，能夠處理模擬真實世界的內容串流。

本研究也為偵測特性的過程加入了樣式篩選與特性消歧義等步驟以提升偵測效能。樣式篩選針對了樣式信心值、可信賴程度以及樣式歧義度三個面向進行篩選。對樣式篩選越嚴格，則可以使用的樣式越少，因此召回率會下降。例如對信心值進行篩選，精確率有提升，但整體效能仍下降；對可信賴度進行篩選，精確率隨有提升但隨著門檻提高，太高的門檻反而使整體效能下滑。而採用歧義度越高的樣式，則可採用的樣式總數增加，召回率有提升，而整體效能緩步下滑。

特性消歧義的步驟則是進行於樣式比對之後，用以區辨樣式出現的地方是否無實體特性或有實體特性，是一個或多個實體特性。透過引入實體類型資訊捨去不符合條件的特性，可在不改變召回率的情形下使系統的效能有顯著的提升。針對特性的篩選實驗了四種篩選策略，由宏觀平均的角度來看，選擇正規化後正確率大於 0.5 的特性是最有幫助的。最後再透過簡單貝氏分類器對特性進行存在或不存在的二元分類，可以顯著提升系統效能。總體而論，引入實體類型資訊、以簡單貝氏分類器進行分類可以大幅提升系統效能。

5.2 未來展望

就偵測流程本身，分析錯誤主要來自三類。第一類是來自歧義性，第二類是無法分辨有無特性存在，以及第三類覆蓋度不足。其中以第二類為影響精確率的主要錯誤來源，如何消彌這幾種錯誤以繼續改善偵測效能是未來的研究課題之一。

在以樣式出現的前後文訓練簡單貝氏分類器的部份，則可以嘗試其他的分類方法，或嘗試漸進式學習（Incremental Learning）以增強系統效能。

而本研究所使用之樣式，以及樣式與實體特性之關聯表來自於 PATTY，在樣式的覆蓋度以及與實體特性間的關聯度強弱受到一定程度的限制，在未來希望可以自網路資源擷取樣式降低覆蓋度的影響，並與知識庫特性建立更細緻的關聯，以降低歧義性造成的影響。

TREC 知識庫加速競賽專注於串流文件中尋找目標實體相關的文件，並判斷是否有助於更新知識庫，而本研究專注於文件中尋找實體特性，若本研究能搭配知識庫加速競賽之實體尋找，又能知道何種實體特性是會隨時間而有所變化，就能更容易的判斷是否有新知識的產生或變動。

關於測試資料集，因為沒有正確的答案標記，只能透過 YAGO 來進行推測，並假設所有的語句都是描寫該條目，希望未來可以針對文件，甚至到句子等級的答案標記，可以進行更詳細的研究。本研究目前是針對文件等級進行偵測，若有更細緻的答案標記，或許可以進行句子等級的偵測。更進一步，希望能夠知道句子中樣式描述的對象實體與其在句子中的位置，才能更精確地偵測實體特性。而 Wikipedia 的條目屬於較長的文章，也希望未來可以針對微網誌、社群留言等短訊息進行偵測。使得偵測的面向更為廣泛以及更為細緻，甚至是自動化地對知識庫進行更新。

參考文獻

Adolphs, P., Theobald, M., Schafer, U., Uszkoreit, H., and Weikum, G. (2011). Yago-qa: Answering questions by structured knowledge queries. In *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, pages 158–161.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *Proceedings of ACL*, pages 1533–1544.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Bonnefoy, L., Bouvier, V., and Bellot, P. (2013). A weakly-supervised detection of entity central documents in a stream. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 769–772, New York, NY, USA. ACM.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Frank, J. R., Bauer, S. J., KleimanAWeine, M., Roberts, D. A., Tripuraneni, N., Zhang, C., Re, C., Voorhees, E. M., and Soboroff, I. (2013). Evaluating stream filtering for entity profile updates for trec 2013. In *Proceedings of The 22th Text REtrieval Conference (TREC 2013)*.

Frank, J. R., Kleiman-Weiner, M., Roberts, D. A., Niu, F., Zhang, C., Ré, C., and Soboroff, I. (2012). Building an entity-centric stream filtering test collection for trec 2012. In *Proceedings of The 21th Text REtrieval Conference (TREC 2012)*.

Kjersten, B. and McNamee, P. (2012). The hltcoe approach to the trec 2012 kba track. In *Proceedings of The 21th Text REtrieval Conference (TREC 2012)*.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2014). DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA. ACM.

Moro, A. and Navigli, R. (2012). Wisenet: Building a wikipedia-based semantic network with ontologized relations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1672–1676, New York, NY, USA. ACM.

Nakashole, N., Weikum, G., and Suchanek, F. (2012a). Discovering and exploring relations on the web. *Proc. VLDB Endow.*, 5(12):1982–1985.

Nakashole, N., Weikum, G., and Suchanek, F. (2012b). Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1135–1145, Stroudsburg, PA, USA. Association for Computational Linguistics.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA. ACM Press.

Takaku, Y., Kaji, N., Yoshinaga, N., and Toyoda, M. (2012). Identifying constant and unique relations by using time-series text. In *Proceedings of the 2012 Joint Conference*

on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, pages 883–892, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wang, J., Song, D., Lin, C.-Y., and Liao, L. (2013). Bit and msra at trec kba ccr track 2013. In *Proceedings of The 22th Text REtrieval Conference (TREC 2013)*.

Yao, X. and Van Durme, B. (2014). Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.