國立臺中科技大學資訊工程系碩士班

碩 士 論 文

巨量 XML 真實程度模型之建構與介面設計

Veracity Model of Big XML Data and API Design

教導教授：陳世穎 Shin-Ying Chen
研 究 生：陳宇威 Yu-Wei Chen

中 華 民 國 106 年 7 月

巨量 XML 真實程度模型之建構與介面設計

Veracity Model of Big XML Data and API Design

教導教授：陳世穎　**SHIN-YING CHEN Ph.D**

研 究 生：陳宇威　**Yu-Wei Chen**

國立臺中科技大學

資訊工程系碩士班

碩士論文

A Thesis

Submitted to

Department of Computer Science and Information Engineering

National Taichung University of Science and Technology

in Partial Fulfillment of the Requirements

for the Degree of

Master of Engineering

July 2017

Taichung, Taiwan, Republic of China

中華民國 **106** 年 **7** 月

國 立 臺 中 科 技 大 學 研 究 所

論文口試委員會審定書

本校 資訊工程系 陳宇威 君

所提論文

巨量 XML 真實程度模型之建構與介面設計

合於碩士資格水準，業經本委員會評審認可。

口試委員：張峯豪 黃健哲

陳世穎

指導教授：陳世穎

研究所所長：

中 華 民 國 106 年 7 月 20 日

巨量 XML 真實程度模型之建構與介面設計

學生：陳宇威　　指導教授：陳世穎

國立臺中科技大學資訊工程系碩士班

中文摘要

XML（eXtensible Markup Language）作為網路資訊交換之資料格式，隨著數位資訊指數成長趨勢，同樣具有巨量資料（Big Data）的特徵，眾多文檔及串流資訊匯聚成為巨量 XML 資料。近年來，巨量資料處理成為學界與產業界極重要的議題，眾多人力、物力投入資料分析的領域，如何處理巨量資料成為學界與產業界極重要的議題。

巨量資料具有 5V 的特性（Volume、Velocity、Variety、 Veracity 與 Value），其中真實性（Veracity）表示巨量資料的真實程度，與資料分析之結果是否可靠和可信有關，然而巨量資料的大、快、疑、雜特性，資料真實性並非能在有限時間內清晰分辨，使用者也缺乏對理解資料真實性的處理方式。

本論文針對 XML 資料是否有參考價值的程度，建立巨量 XML 真實程度模型（Veracity Model of Big XML Data），適用於巨量資料情境之下，協助資料使用者依照本論文模型架構定義 XML 真實模型及程度計算方法，計算真實程度及資料視覺化協助判讀 XML 文件真實性。本研究在建立巨量 XML 真實程度模型過程中，首先逐步建構具有階層性的真實維度、真實屬性及量化因子；接續設計模型介面方法，透過物件導向設計、統一塑模語言 UML 表達設計需求，以 Python 程式語言進行模型介面設計，並實作 XML 真實模型的計分方法，量化此理解的真實程度（veracity degree），最後以模型應用範例闡述應用結果，該模型可適用在巨量資料環境，提供使用者參考。

*關鍵字：*巨量資料、XML、資料真實性、真實程度、介面設計

Veracity Model of Big XML Data and API Design

Student：Yu-Wei Chen Advisor： Shin-Ying Chen

Department of Computer Science and Information Engineering
National Taichung University of Science and Technology

Abstract

XML (eXtensible Markup Language), as a data exchange format in network system, has the same features as Big Data. Along with the growth trend of digital information index, many documents and streaming information gather as big XML data. Big data processing has been of considerable importance with massive resources and effort invested in data analyzing throughout academic and industrial fields in recent years. How to deal with Big data becomes an issue of great significance for the academics and industries.

Big data has the features of 5Vs, volume, velocity, variety, veracity, and value. Veracity represents how authentic the Big data is, which is related to the reliability and credibility of data analysis results. However, with the other features of massive volume, fast processing, doubtful and diverse sources, the veracity of Big data is hardly achieved in a limited time; neither have users the proper tools for distinguishing data veracity.

In this thesis, a veracity model of big XML data is built to provide references for analyzing XML data. Based on the model structure in this thesis, users are able to define the veracity model and the calculation method of veracity degree for big XML data. The authenticity of XML files is hence interpreted through the calculation results of veracity degree and data visualization. In the process of establishing this veracity model of big XML data, we first construct the hierarchical real dimension, real attribute, and quantization factor, and then design the model interface method. Design requirements are presented in class diagram through the object-oriented design and UML (Unified Modeling Language). The model interface is designed in Python programming language, and the scoring method of veracity model of big XML data is implemented to quantify the veracity degree as interpreted. Lastly, application examples of this model are taken to elaborate the application results. This model is available as references for users in the big data environment.

Keywords: Big Data, XML, Data Veracity, Veracity Degree, API Design

目次

表目錄

圖目錄

第一章 緒論

1.1 研究背景

近年來，隨著資訊的發達與普及，大量、多樣來源的數位資訊呈現指數成長趨勢，這些數據無處不在，IBM[3]於 2015 年推估，每天有超過 2.5 quintillion bytes 資料被創造，且超過 90%的資料是近 2 年內產生，資料來源舉凡感測器、交通與航班訊息、金融交易資訊、社群媒體網站等，資料類型包含網站、文字、圖片或影音、GPS 訊號等，這些龐大、變動迅速且多樣的數據匯流成所謂巨量資料（或稱大數據）。巨量資料處理已成為學界與產業界極重要的議題，眾多人力物力投入巨量資料分析的領域，隨著資訊科技及數位生活的發展趨勢，越來越多應用以 XML 格式表示資料，而這些網路上的 XML 資料來源彙整起來將成為巨量 XML 資料。

XML 資料係指可擴展標記語言（eXtensible Markup Language，簡稱 XML 標記語言），是用來建立描述結構化資料標示語言的語言，通過標記，電腦之間可以交換資料並處理資料內容。XML 標記語言由 SGML 標記語言精簡並延續 SGML 的優點，於 1995 年開始發展，並於 1998 年由 W3C (World Wide Web Consortium, http://www.w3.org/)發佈為標準（XML 1.0）。XML 融合 SGML 的資料相容性與 HTML 的簡單標記法，提供一套簡單、標準並可擴展的語法，將各種資訊如文字、表格或圖形等以原始的方式儲存；並於儲存的過程中加入可供辨識的元素，讓伺服器或客戶端應用程式可藉此供辨識的元素，將資訊內容做進一步處理得到所需的資訊。

巨量資料擁有的大量、快速、多樣、真偽難辨與潛在價值等稱為「5V」的特性，5V 分別為 Volume、Velocity、Variety、Veracity、與 Value，其中真實性（Veracity）表示巨量資料的真實程度，與資料分析的結果是否可靠和可信有關，所謂「垃圾進，垃圾出」，不可靠或不可信的資料將對資料分析、儲存與運用造成偏誤及無意義的結果。因此，對於網路上傳輸的各類文件其來源的真實性對於資料的分析就有重要的影響。有鑑於此，巨量資料真實性之特性應有協助判斷資料信賴與否的參考指標。

在本研究中，所謂真實性（Veracity）的意義並非避免使用有人為刻意改造的假資料，而是使用的 XML 文件資料的特性是否具有參考價值。此 XML 文件資料的特性取決於應用程式（或服務）對於 XML 文件資料的理解，亦即應用程式（或服務）認為具備那些特性的 XML 文件才是真實有效的，例如有相同的資料來源、有相同的文件結構（DTD 或 XML Schema）、...等可能的特性。本研究將研究如何把使上述對於 XML 文件資料的理解以模型描述，並建立估計函數，以量化此理解的真實程度（veracity degree）。

1.2 XML 文件結構

由於 XML 技術的發展與進步，XML 廣泛地用來作為跨平台之間交換資料的形式，越來越多商業應用採用 XML 表示資料。例如，MathML [32] 與 EbXML [9]分別用來描述數學和電子商務之 XML 格式。

XML 文件必須是結構完整的（well-formed）。一份結構完整的 XML 文件必須遵守僅有一個最上層元素、至少有一個元素、每一個元素都有起始元素（start tag）與結束元素（end tag）且呈巢狀結構、大小寫有別、...等規則。另一方面，所謂的 valid XML 文件是指該 XML 文件不但是 well-Formed，並且在 XML 文件中的元素、屬性以及其他單元（如 Entity、Notation ... 等），皆符合其引用的 DTD 或 XML Schema 之規範。

XML 採用 DTD（Document Type Definition）或 XML Schema（XSD）來描述文件的邏輯結構。DTD 文件型別定義（Document Type Definition，DTD）用以描述 XML 文件的結構規則，可規範所有元素、屬性與實體間的相互關係，做為標準化文件格式之目的。例如，制定一致的上下游廠商之訂單格式或文件出版之統一格式。而 XML Schema 目的與 DTD 相同，它可定義出現在文檔中的元素及其子元素、子元素的次序、屬性，並可定義子元素的數目、元素和屬性的資料類型。一個帶有 DTD 的 XML 文件依 W3C 展示範例如下：

```
1  <?xml version="1.0"?>
2  <!DOCTYPE note [
3    <!ELEMENT note (to,from,heading,body)>
4    <!ELEMENT to       (#PCDATA)>
5    <!ELEMENT from     (#PCDATA)>
6    <!ELEMENT heading (#PCDATA)>
7    <!ELEMENT body     (#PCDATA)>
8  ]>
9  <note>
10   <to>Alice</to>
11   <from>Bob</from>
12   <heading>Reminder</heading>
13   <body>Don't forget the meeting!</body>
14 </note>
```

!DOCTYPE note 第 2 行定義此文檔是 note 類型的文檔。

!ELEMENT note 第 3 行定義 note 有元素："to、from、heading,、body"。

!ELEMENT to 第 4 行定義 to 元素為"#PCDATA"類型。

!ELEMENT from 第 5 行定義 from 元素為"#PCDATA"類型。

!ELEMENT heading 第 6 行定義 heading 元素為"#PCDATA"類型。

!ELEMENT body 第 7 行定義 body 元素為"#PCDATA"類型。

假如 DTD 位於 XML 源文件的外部，那麼它應通過下面的語法被封裝在一個 DOCTYPE 定義中：

```
1  <!DOCTYPE 根元素 SYSTEM "文件名">
```

這個 XML 文檔和上面的 XML 文檔相同，但是擁有一個外部的 DTD:

```
1  <?xml version="1.0"?>
2  <!DOCTYPE note SYSTEM "note.dtd">
3  <note>
4      <to>Alice</to>
5      <from>Bob</from>
6      <heading>Reminder</heading>
7      <body>Don't forget the meeting!</body>
8  </note>
```

以下為包含 DTD 的"note.dtd" 文件：

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

通過 DTD 可使 XML 文件達到以下目標：每一個 XML 文件均可攜帶一個有關其自身格式的描述；獨立的團體可一致地使用某個標準的 DTD 來交換數據；可使用某個標準的 DTD 來驗證從外部接收到的數據；可以使用 DTD 來驗證自身的數據。

至於 XML Schema 係基於 XML 的 DTD 替代者，在 2001 年 5 月 2 日成為 W3C 標準[33]，可描述 XML 的文檔結構，XML Schema 語言也可作為 XSD（XML Schema Definition）來引用。XML Schema 可定義出現在文件中的元素、屬性、哪個元素是子元素、子元素的次序、子元素的數目、元素和屬性的數據類型等，下面這個例子是一個名為"note.xsd" 的 XML Schema 文件，它定義了上面那個 XML 文檔的元素：

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.w3school.com.cn"
 xmlns="http://www.w3school.com.cn"
 elementFormDefault="qualified">
<xs:element name="note">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="to" type="xs:string"/>
          <xs:element name="from" type="xs:string"/>
          <xs:element name="heading" type="xs:string"/>
          <xs:element name="body" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

對 XML Schema 的引用：

```
1  <?xml version="1.0"?>
2  <note
3    xmlns="http://www.w3school.com.cn"
4    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5    xsi:schemaLocation="http://www.w3school.com.cn note.xsd">
6  <to>George</to>
7  <from>John</from>
8  <heading>Reminder</heading>
9  <body>Don't forget the meeting!</body>
10 </note>
```

XML 文件可依據 DataGuides 模型[21]建立其摘要結構。在 DataGuide 模型中，此摘要結構為一樹狀結構，稱之為 DataGuides。DataGuides 僅表示 XML 文件所有路徑的摘要結構，而非完整的 XML 文件。亦即，相同元素名稱序列構成的所有路徑在該 XML 文件的 DataGuides 中僅以該序列表示路徑摘要。許多 XML 應用以 DataGuides 描述 XML 文件的結構，並加速查詢。

1.3 巨量資料的 5V

描述巨量資料的性質時通常會以五個 V 的特性描述，分別為規模性（Volume）、時效性（Velocity）、多樣性（Variety）、 精確性（Veracity）、與價值（Value）。其意義可理解為「巨量資料規模大且格式多樣，必須及時處理完成，而這些資料必須是具有可信度的，處理的結果能提升巨量資料的價值。」

在五個 V 中，Volume 表示巨量資料規模量大，傳統的資料庫系統已無法負荷；Velocity 表示處理的時效，巨量資料產生迅速，系統必須及時處理完成；Variety：指的是資料的形態多樣，包含資料庫、XML 文件、文字、影音、串流、...等等結構性、半結構性、非結構性的資料；而由於巨量資料來源多元，所以 Veracity 指的是這些資料是否可靠、品質是否足夠、或是結構是否完整，亦即可信度是否充足。Value 是巨量資料處理流程開始與結束都會牽涉到的特性，透過巨量資料處理產生更多有價值的資料。

在設計巨量資料應用時，以巨量資料的 5V 特性進行屬性分析有助於觀察該巨量資料之特徵。例如，Facebook 社群資料具有資料量極大（volume）、訊息產生速度極快（velocity）、訊息內容無結構化等特性（variety），而針對不同的議題，社群成員發布訊息內容的真實性有程度差別（veracity），因此產生價值的方式可以依據這些特性加以分析與設計（value）。反之，對於擷取傳統資料庫中內容的應用，具有資料量大（volume）、資料產生速度快（velocity）、資料具結構化（variety）、且可保證資料的真實性（veracity），而產生價值的方式則依據這些特性加以分析並設計（value）。因此，5V 特性的分析，是設計巨量資料處理應用一個重要的依據。

在資料分析時，缺乏真實的資料將增加分析誤判的機會，並增加資訊處理的負擔，因此，判斷資料是否真實至關重要。雖然資料必須是可靠且可信的，但是基於兩個理由，巨量資料並無須或無法保證是完全真實，因而必須建立量化的真實程度。

第一個原因是巨量資料格式多樣且來源多元。由於資料來源多元，就無法保證取得的資料完全真實（亦即資料正確性、資料完整性、...等要求並無法保證達成）。

第二個原因是因為若要求資料完全正確，如此一來在進行分析前必須進行資料淨化的程序。但是因為巨量資料規模大，資料淨化將花費許多時間。若有考量真實情形進行資料淨化，則資料得以乾淨的，真實性的特徵得以保留。但是在求取處理速度而捨棄資料淨化步驟的情況下，資料分析的結果必須能呈現資料是否正確可信，因此將真實性程度化有其必須性。

基於前述原因，Veracity 的特性是可有不同程度變化且可據以區別的。例如，對於 XML 文件而言，若以是否符合 DTD 規範區分，結構完整（well-formed）與完整且正確（valid）的 XML 文件在 veracity 特性上就有不同程度的差別。對於兩份 well-formed XML 文件而言，由於 XML 文件元素可以由使用者自訂，因此可能出現同名異義（或異名同義）的元素，意即也許元素名稱不同，但都是意指相同的事物，此時就應具備較高的真實程度。若更進一步比較兩份文件皆相同的 DTD 來規範 XML 文件結構，則在真實程度上就可能提高。

巨量資料的真實程度（veracity degree）的意義可以從兩個觀點進行觀察，第一個觀點是資料理解性（data understandability）的角度。第二個觀點是巨量資料評效基準（big data benchmark）之巨量資料產生器（big data generator）產生巨量資料的觀點。

真實程度的第一個觀點是從資料理解性（data understandability）的角度詮釋 Veracity 的意義。假設有兩份 XML 文件，一份為基準文件 B，一份為被量測文件 M；使用者可以從多個角度去描述「以基準文件觀之，這份被量測文件多少可能是真實的」。例如文件 M 與文件 B 來源相同，因此文件 M 是應該是真實的。或是，文件 M 與文件 B 有相同的 DTD，因此文件 M 應該是真實的。或是，文件 M 與文件 B 的 DataGuides 相同，因此文件 M 應該是真實的。這些對於來源相同、DTD 相同、DataGuides 相同的「想法」，就是用所謂的資料理解性來描述被量測文件 M 是否真實。而由於這樣的想法可以多元且其重要性可能有差異，因此真實性就可以有程度上的區別。

從巨量資料評效基準的研究觀察到巨量資料之 veracity 特性的另一層意義。在相關研究中，當建立巨量資料產生器（big data generator）時，其對於 veracity 特性的分析是「此產生器產生的資料必須保留原始資料（raw data）的特性」。其意義為原始資料是一個基準，表示百分之百的完全真實；因此，原始資料具有什麼特性，產生器產生的資料也必須依據該原始資料的特性產生，才能作為評效基準（benchmark）使用的巨量資料。

1.4 研究動機與目的

巨量資料分析與處理已成為目前重要的研究議題。而且，巨量資料的處理在蒐、存、取、析、用等步驟上，已經建立一套流程與機制。另一方面，由於 XML 是一個網路資料交換的標準，建立巨量 XML 資料的相關應用服務是必然的趨勢。若能以巨量資料的 5V 來描述與分析巨量 XML 資料的特性，更能設計出符合這些特性的應用服務。因此，本研究先分析巨量 XML 資料的 5V 特性如下。

1. Volume：資料量大，因此 XML 文件的節點、路徑、結構數量多且龐大。
2. Velocity：關聯式資料庫、NoSQL 資料庫、網路交換文件、開放資料（open data）、...等資料，都可能採用 XML 格式建立，因此資料來源多元，且文件產生速度迅速。
3. Variety：XML 文件結構為半結構化文件；相關的半結構化文件例如 JSON 文件，在語意上有相關聯。若能與 XML 文件相互轉換，可建立多樣的應用服務。
4. Veracity：由於資料來源多元、文件結構變化多、無 DTD 或 XML schema 規範的文件可能有同名異義或異名同義的元素、...等原因，因此 XML 文件資料上 veracity 的特性可以加以程度的區別，以鑑別文件的真實程度，提供使用者參考。
5. Value：XML 文件的價值為原始內容與結構；經過處理分析後，產生的價值則由應用上之處理邏輯而異。

依據上述分析，若把焦點聚焦於 Veracity 的維度上思考巨量 XML 資料的應用服務，可以引導出一些創新的 XML 應用方向。例如在建立 XML 文件的真實程度（veracity degree）後，當進行資料分析時，此分析過程可依據資料的真實程度鑑別分析結果的可信度，提供使用者參考。此方式有別於傳統資料分析的方式，於資料淨化階段將不真實的資料完全過濾，意即參與資料分析的資料是完全正確的。然而，在從事巨量資料分析的實務上，資料淨化階段將耗費相當多的處理時間。因此，為了加速資料處理分

析速度，若省略資料淨化階段，但註記資料真實程度的差異，再於處理結果呈現可信度的區別，則使用者便能依據處理結果可信度進行決策參考。

在上述的想法下， 巨量 XML 資料處理將可由「完全正確的 XML 資料處理分析」發展成「提供可信程度的 XML 資料處理分析」。可能產生的新的研究或應用類別包括 XML query 查詢結果的可信度、XML data mining 的分群技術、XML similarity 的新的定義、以最有可能的 XML 文件結構修補因為網路傳輸遺失缺損的結構、...等。

因此，本研究的動機在於由於利用巨量 XML 資料其 veracity 的特性，區別 XML 文件的真實程度，並依據此真實程度區別資料分析處理結果的可信度。

基於上述說明，本研究目的旨在建立 XML 文件之真實程度估算機制，以區別 XML 文件的真實程度，使用者可透過本研究歸納及舉例的真實維度、真實屬性及關鍵指標所呈現的真實值判斷該 XML 文件真實性，配合資料視覺化進行觀察，作為因應 XML 文件在資料量大、迅速、多樣的巨量資料環境中，定義所需的真實屬性供資料真實程度之參考。

1.5 研究流程

下圖為本研究的流程圖，內容安排如圖 1-1 論文架構

圖 1-1 論文架構圖

第一章：包含研究背景、動機、目的與流程。

第二章：依據本研究範圍，探討巨量資料真實性與 XML 文件比對等相關文獻理論與技術。再加以分析整理後，作為後續研究參考依據。

第三章：設計 XML 真實模型。

第四章：設計實現真實模型之介面。

第五章：說明資料 XML 真實模型建立與實作。

第六章：模型與系統演算法實作成果。

第七章：針對本研究提出真實模型與介面設計，以及根據本研究建立的 XML 真實性模型實作結果提出結語與未來展望，期望能作為後續 XML 真實性評估相關後續研究之參考。

第二章 文獻探討

XML 文件具有樹狀結構特性，鑑於 XML 文件有關 Veracity 研究不多，本章以資料真實性有關的 XML 研究，包含 XML similarity、data veracity 進行探討。

2.1 XML similarity

XML 文件相似度（XML similarity）係基於 XML 文件的結構資訊比較兩者間之相似性，比對 XML 文件相似度的方法，約可分為兩類，第一類方法對 XML 樹狀結構的編輯距離（edit distance）作為比較基礎，需採用多少新增、刪除及修改節點的動作使兩份文件相同[25]，後續研較包含改進其效能，以求更快速的計算兩棵 XML 有序樹之間最小樹編輯距離的方法[1,16,26]。

第二類方法係透過二棵樹彼此之間的最小樹編輯距離能在多項式時間內得到解答[6,22,27]，將樹以有效率的編碼方式加以重新呈現。[6,27]提出一種以樹編輯距離為基礎的 XDOCSim 方法，運用此相似度比對方法，用來比對 XML 文件對應的 XML grammar(例如，DTD 或者 XSD)，其作法是先將 DTD 或者 XSD 先依其提出的規則轉換為對映的 DTD 樹或者 XSD 樹，接著再計算每一份 XML 文件形成的 XML 樹與 DTD 樹或者 XSD 樹之樹編輯距離為何，為了因應 DTD 及 XSD 本身的意涵，則在其採用的 XDOCSim 方法中，加入了計算節點之最小及最大出現的次數，以利進行以 DTD 樹或者 XSD 樹進行 XML 文件的分類。除此之外，仍有學者以圖形理論為基礎，評估兩棵 XML 樹之間的相似程度。[22]則是以圖形理論為基礎建立起兩棵 XML 樹之關連圖（association graph），透過尋找關連圖中的最大 cliques 的方式，找出兩棵 XML 樹之間最相似的子樹（maximal subtree isomorphisms）。因此，這類方法並不如樹編輯距離方法的相似度比對方式來的精確，但相對的擁有較快速度的相似度比對效能。

在巨量資料面臨的大、快、疑、雜的情境，巨量資料真實性計算 XML 樹編輯距離將導致效能不佳，XML similarity 僅討論文件結構，如用以解釋 XML 巨量資料真實性尚有不足之處。

2.2 Data veracity

[18]本文認為，巨量資料具有不同的特性，從而影響其品質（quality）。巨量資料會因為資料收集方法、資料清理或分析方式而造成資訊偏差（biased）、模糊、與不準確。[18]亦說明目前少有討論資料真實性的研究。[18]提出三個維度來探討資料真實性，各維度以正負向量關係表示，包括資料的客觀或主觀（objectivity or subjectivity）、真實或假造（truthfulness or deception ）與可信或不可信（credibility or implausibility）三個維度，並目前可以針對此三個維度運作的工具。此三個維度主要是應用於分析文章內容的資料為主。此外，[18]將此三個維度整合為「巨量資料真實指數（big data veracity index）」，此指數能有效運用於分析文章內容的資料品質（data quality）。而[4]以對於資料的信賴度（confidence）描述資料真實性，[4]指出資料真實性指的是資料準確度以及他如何相關於商業價值（business value）。

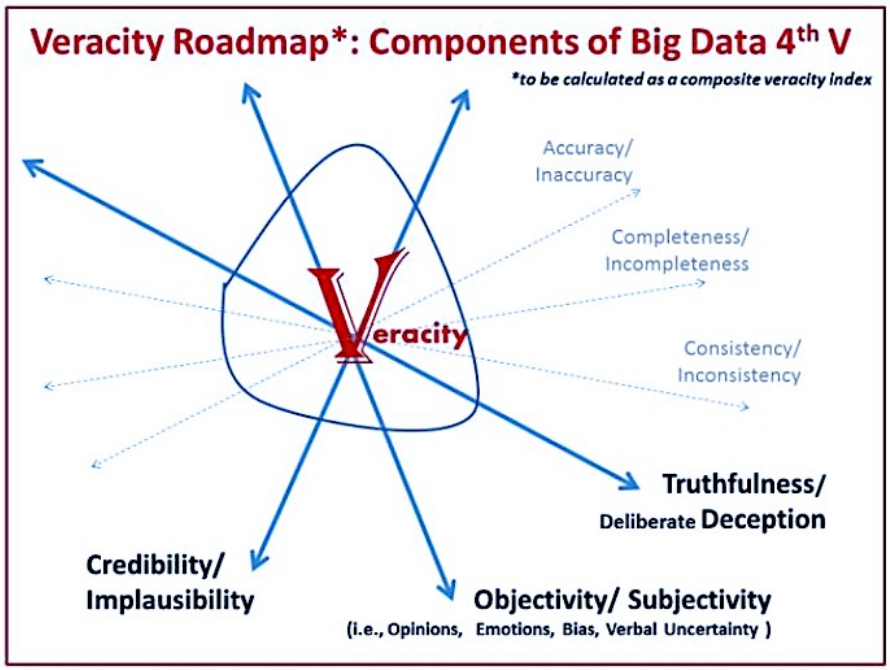

圖2-1 Conceptualization of the Components of Big Data Veracity

[18]與本論文的差異有下列三點。第一，[18]以文章內容的資料為主，而本論文以 XML 資料為應用對象。第二，[18]以資料品質(data quality)的觀點描述資料真實性，而本論文以資料理解性(data understandability)的觀點描述 XML 資料的真實程度。第三、[18]建立的三維模型無法反映以此資料理解性的觀點描述 XML 資料的真實程度，本研究建構真實模型(veracity model)，可彈性的依據不同應用服務上的 XML 資料建立真實維度(veracity dimensions)，估算資料的真實程度。而且此模型可以彈性的擴展至其他應用，非僅限於巨量 XML 資料。因此，[25]與[11]分別以資料品質與資料準確度定義資料真實性；而事實上這些定義是可以包含在本論文建立的真實維度中。

第三章 XML 真實模型設計

有鑒於巨量資料的資料量大（volume）、資料產生速度快（velocity）、資料結構多樣（variety）、且衡量資料的真實（veracity）與否關乎資料產生的價值（value）。對於現在興起的資料科學領域，資料不真實意味著接續的資料分析與產出有偏誤之虞，在缺乏具體量化資料真實性的情況下，設計可廣泛適用及描述資料真實性模型以幫助使用者理解資料真實性有其重要性。

本研究首先以資料理解性（data understandability）描述文件的真實程度從定義「真實模型」（Veracity Model），其次描述「真實維度」（veracity dimensions）的意涵，進而探討建立各維度的評估方式，用以估計算真實程度（veracity degree），建構巨量資料真實模型描述如下：

3.1 真實模型

巨量資料來源多元、文件結構變化多端、有無符合文件結構規範，以及可能有同名異義或異名同義的元素等原因納入考量，使資料上 veracity 的特性可以加以程度的區別，本研究建構「真實模型」（Veracity Model）以描述資料文件的特性是否具有參考價值的真實程度模型，用以鑑別文件的真實程度，提供使用者參考。

在建構及定義真實模型部分，真實模型可以從多種面向評價與判斷其真實屬性及真實值，不同面向的評價方式本研究稱為「真實維度」（veracity dimensions），每個真實維度各自有其不同「真實屬性」（veracity attributes），每個真實屬性應具備可計算程度的「量化因子」（quantization factor）。真實程度模型可對不同的 XML 文件特性進行評價，且評價的方法應保留使用者自定義的彈性，以及適用於各種 XML 文件真實程度的泛用性為目標。在此 XML 真實模型中，本研究假設有一份基準 XML 文件 B 與被量測 XML 文件 M；而文件 M 將被量測與文件 B 的真實程度。茲將真實模型說明如下。

3.1.1 真實維度

在 XML 真實模型模型(veracity model）V 有 n 個真實維度 D_i，真實維度（veracity dimensions）用以描述可依據使用者對資料邏輯的理解進行定義，將衡量 XML 文件真實程度的各種面向以 D_i 進行描述，可以包含文件的來源、文件結構、文件內容及文件時間等維度。真實模型 V 將有 n 個真實維度，D_i 表示真實模型第 n 個維度，表示式為：

$$V = \{D_i\}，i=1..n。$$

3.1.2 真實屬性

在 XML 真實模型的真實維度 D_i 中，包含與此維度相關的「真實屬性（veracity attributes）」。真實屬性可依據使用者對資料邏輯的理解進行定義。例如，從資料來源作為觀察維度 D_1，來源的真實屬性可以包括來源之來源網址 $P_{1,1}$、編碼 $P_{1,2}$；文件結構 D_2 的屬性可包括各種文件結構規範 $P_{2,1}$、資料交換規範 $P_{2,2}$ 等。文件的內容、文件紀錄時間亦可作為屬性。真實維度 D_i 將會有 d_i 個真實屬性，$P_{i,j}$ 表示真實維度 D_i 的第 j 個屬性，其表示式為：

$$D_i = \{P_{i,j}\}，j=1..d_i。$$

3.1.3 量化因子

真實維度、真實屬性提供了觀察資料真實性的面向，而為了具體量化做為可供評估真實程度的數值，模型 V 中建立各真實屬性的「量化因子」（quantization factors）具體量化為可供評估真實程度的數值。例如，現有資料文件以來源網址作為真實屬性 $P_{1,1}$，量化來源網址方式眾多，使用者可分別從網址名稱的字串比對是否相符進行判斷、透過來源網址的網域層級樹狀結構階層相似性分階層加權計算，透過頂級網域域名(com, edu, gov 等)及頂級國家域名(tw, us, eu 等)比對計算等，各種數值判斷方法可取 1 個或多個做為量化評估的量化因子，$Q_{i,j,k}$ 表示第 i 個真實維度、第 j 個真實屬性第 $P_{i,j}$ 個量化因子，表示式如下：

$$P_{i,J} = \{K_{i,j,p}\}，k=1..\ p_{i,j}$$

據此，本論文提出的真實模型具有真實維度、真實屬性、量化因子，以計算真實程度的階層特性，如下圖 3-1。

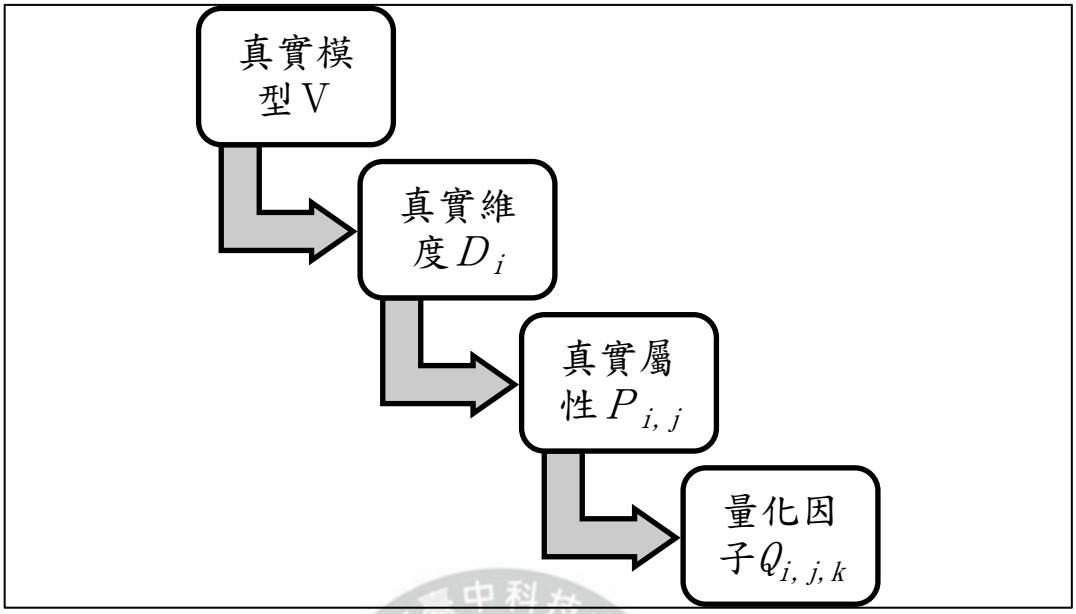

圖 3-1 真實模型之階層性表示

3.2 真實程度

依據真實維度 D_i 中每個 $P_{i,j}$ 的性質或重要性，計算被量測文件的「真實程度」（veracity degree）。真實維度 D_i 之真實值 V_i 的計算方式可依據其真實屬性 $P_{i,j}$ 來建構估計函數；亦即，$V_i = F_i (P_{i,1}, P_{i,2}, ..., P_{i,m})$。

有了評估真實模型、真實維度、真實屬性及量化因子，所計算出的真實程度分數可用資料視覺化呈現，在 n 個真實維度建立的雷達圖上標示 V_i 的座標，i= 1 to n。因此，文件 M 對於文件 B 的「真實程度」定義為所有真實值 V_i 在其座標 D_i 圍出的區域面積，表示為 $V_{M|B} (D_1, D_2, ..., D_n)$。

真實屬性 $P_{i,j}$ 可藉由各該量化因子 $Q_{i,j,k}$ 附屬之量化函數 $f(Q_{i,j,k})$ 計算真實屬性 $P_{i,j}$ 真實程度$|P_{i,j}|$。而各真實屬性的真實程度因為度量的差異，而有可能相對過大或過小。因此，計算 D_i 的值時，以 $w_{i,j}$ 做為正規化各真實程度$|P_{i,j}|$之係數。

$$|P_{i,j}| = \sum_{k} f(Q_{i,j,k})$$

$$|D_i| = \sum_{j} \left(w_{i,j} \times |P_{i,j}|\right)$$

$$= \sum_{j} \left(w_{i,j} \times \sum_{k} f(Q_{i,j,k}) \right)$$

如圖 3-2 所示,假設有六個真實維度 D_1、D_2、...、D_6，目前量測文件 M 對於基準文件 B 在此六個維度的值分別為 66、70、50、76、80、95，則雷達圖上這六個值包圍覆蓋的面積即文件 M 對文件 B 之真實程度 $V_{M|B}$。

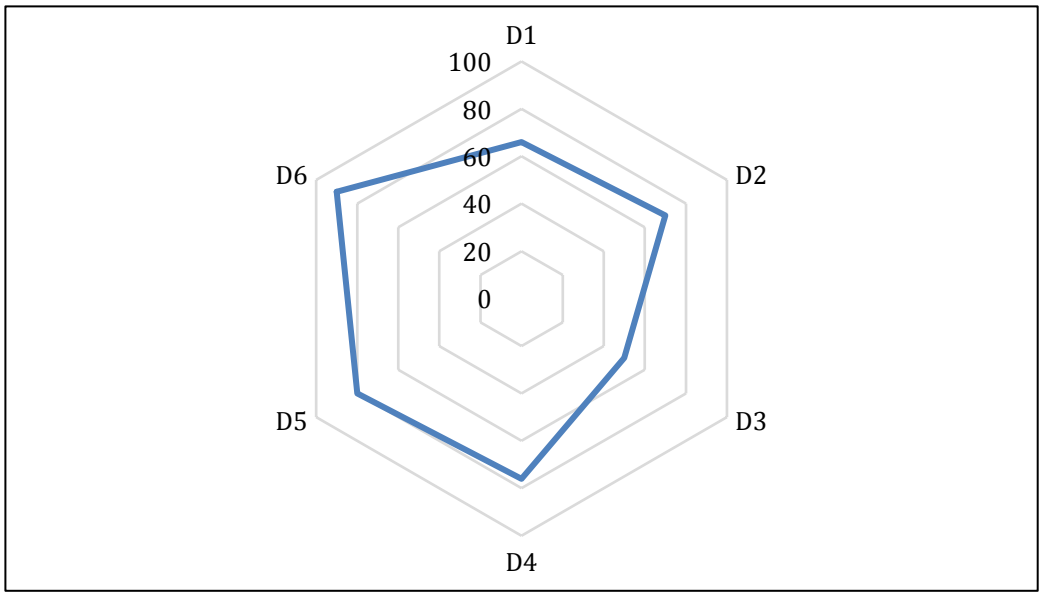

圖 3-2 真實模型、真實維度、真實值與真實程度

真實模型各維度或屬性亦可透過資料視覺化方式，用圖像提供使用者觀察資料的面向。例如：資料建立時間、文件次序、資料來源地理位置、資料時區分佈、資料文件的樹狀結構、資料欄位出現次數...等。

第四章 模型介面設計

為實際運用本研究建立之真實模型，本章提出「真實模型介面（API of the Veracity Model of Big Data）」之設計。運用物件導向設計（Object-Oriented Design, OOD）[19]方法，其中包含可定義跟評估真實屬性的物件（Object），物件可各自獨立、組合，並且以統一塑模語言（Unified Modeling Language, UML）描述模型介面，達到「彈性」及「適應性」目的，以供巨量資料分析所需。有關介面設計目的及設計方式說明如下。

4.1 模型介面設計目的

4.1.1 彈性（Flexibility）

本研究提出的真實模型介面目的之一為「彈性」，意即使用者運用本介面可有充足的彈性使用能力。在真實模型中，使用者從各種不同的真實維度、真實屬性及量化因子進行真實程度計算，需要對於使用者足可彈性因應需求的模型介面，以便自行定義屬性及計分方法。透過物件導向設計，使用者可從中運用或繼承其全部或部分屬性，設計符合觀察資料文件及自定義的程式。

4.1.2 適應性（Suitability）

本研究提出的真實模型介面另一目的為「適應性」，適應性可分為二種特徵，其一為可適用於各類 XML 文件，如 XML 檔案、XML 串流文件，是為本模型介面「應用的適應性」。另外意即此模型介面可以適用於多種媒體環境中，依據使用者需求適應於資料處理情境，是為本模型介面「系統平台的適應性」，例如：運用於巨量資料處理情境中，與 Apache Hadoop[34]、Apache Spark[2]等都能融入於處理情境，供使用者分析巨量資料應用。再者，在資料處理的蒐、存、析、取、用流程中，加入該介面作為資料搜集過濾器，不符合真實程度的資訊加以警示或剔除，抑或運用在資料分析時真實程度的參考。

4.2 模型介面設計方法

4.1.1 物件導向設計 OOD

XML 真實模型設計可用物件導向設計進一步將介面設計具體化，真實模型、真實維度、真實屬性及量化因子以 OOD 物件（Object）進行介面程式開發，物件組成包括屬性 （Attributes）及方法（Methods），用以依據訊息及該物件內部狀態作出反應，可透過「封裝」為類別（class）供他人運用，或者使用他人封裝的類別，增加介面的易用性，並實現模型介面彈性與適應性的目標。

物件導向設計可用諸多物件導向程式語言，如：Java、Python、C++、Ruby、JavaScript、Scala、Swift、R、GO、PHP 等進行程式設計，而前述程式語言亦有為數眾多的套件（Package）或稱模組（Module）可供運用，以程式語言 Java 為例，Java 平台[15]提供了數千個已經封裝好的類別，這些類別形成的集合即為 API；另以 Python 程式語言為例，截至 2017 年 5 月底止已有 109,682 個套件登載於 PyPI 可供運用[20]。再者，OOD 可以透過「繼承」的方式將父類別（parent class）、超類別（super class），或基底類別（base class）繼承運用，其中方法也會被自動繼承過來，達到程式碼的再用性（reusability）。本模型將採用 Python 物件導向語言進行模型介面開發。

4.1.2 統一塑模語言 UML

XML 真實模型建立透過介面設計實現，在描述模型方面，具有階層特性的物件類別可以使用統一塑模語言[45]（Unified Modeling Language, UML）進行描述，UML 是一種將開發中、物件導向、軟體密集系統的製品用於闡述、視覺化、構建和編寫一個的開放方法。UML 的物件圖（Object diagram）可說明各物件之間的彼此關係，便於在開發流程中使程式設計者、專案轉理者與客戶之間溝通。在本 XML 真實模型介面中，將借由 UML 物件圖表示各類別包含的屬性與方法，以及類別之間的相互關係。

在透過 UML 表達各類別物件而言，本模型的真實模型、真實維度、真實屬性及量化因子各有其應具備的物件屬性與方法，分別以模型的功能需求建立模型類別與類別方法設計。

在真實模型類別（class Veracity）中具有真實維度列表屬性、計算真實程度方法及添加真實維度的方法；在真實維度類別（class Dimensions ）中具有定義維度名稱、真實屬性的列表，以及設定名稱方法及增加真實屬性的方法；在真實屬性類別（class Property）中具有定義量化因子屬性、真實屬性之屬性，以及增加設定量化因子的方法；在量化因子（class Quantization)中，具有定義量化因子屬性及計算真實屬性真實程度的方法；以模型類別（class Model）呼叫上述類別進行模型實例化。

另一方面， UML 可表達各類別物件彼此之間關係，在本模型介面設計中，一個完整的真實模型具有數個不同定義的真實維度，每個真實維度各自有數個不同的「真實屬性」描述該維度特徵，而每個真實屬性應具備 1 項以上可供量化程度的「量化因子」。UML 物件圖以線條與箭頭型式表達物件之間關聯性，如：標註「1..* 」：表示一到多個、標註 m..n 表示 m 個到 n 個。而以箭頭起始點為實心菱形、終點為實心箭頭表示兩者之間的組合（Composition）關係，可進一步針對物件關係描述如下：

真實模型類別（class Veracity）是真實維度（class Dimensions ）的組合(Composition)，一個真實模型可對應多個真實維度；真實維度類別(class Dimensions）是真實屬性（class Property）的組合，一個真實維度可對應多個真實屬性；真實屬性（class Property）是量化因子（class Quantization）的組合，一個真實屬性可對應多個量化因子；真實模型物件以 Model 物件負責建立（以<< create >>標記）。

基於上述模型介面需求，為求模型與程式設計界面無落差，本研究採用具有程式介面轉換為 UML 類別圖式功能的整合開發環境（Integrated Development Environment，IDE）進行 UML 設計，透過由捷克軟體公司 JetBrains 開發的「Intellij IDEA」Java 整合式開發環境[14]，由 JAVA 物件導向程式設計之程式碼轉譯，UML 相對表示如圖 4-1。

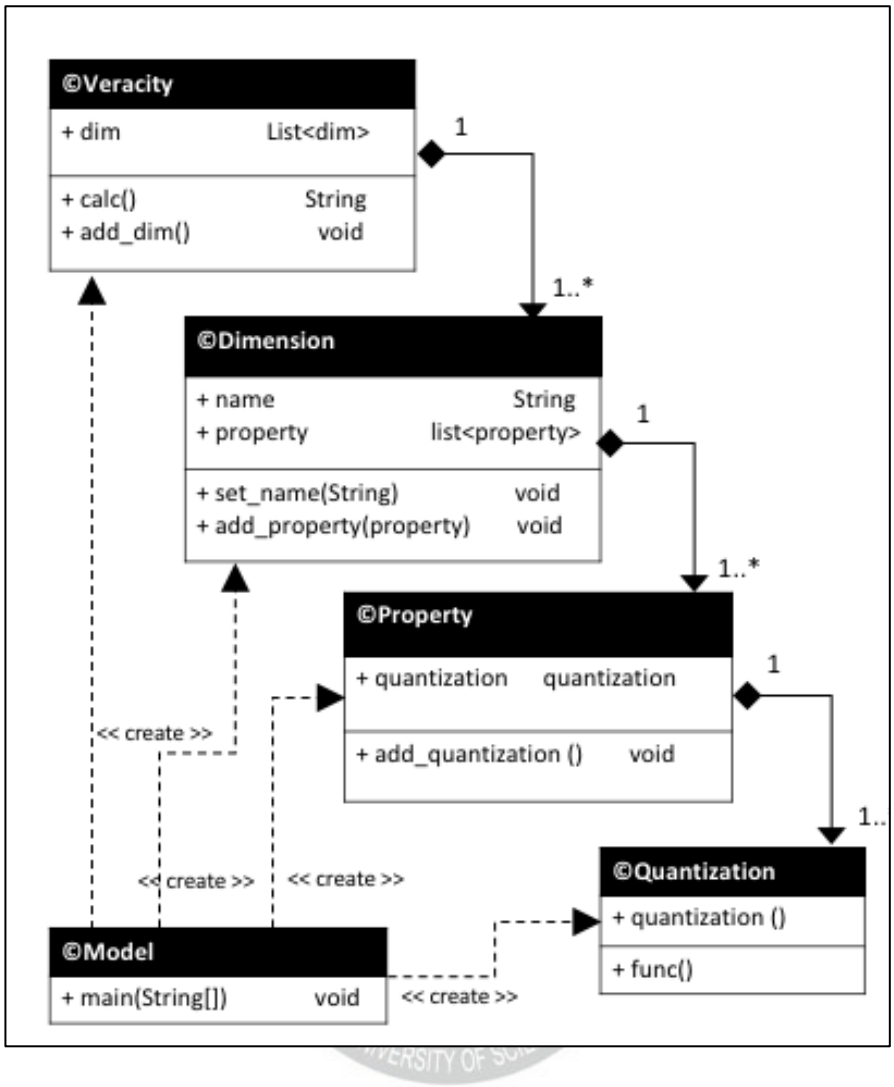

圖 4-1 XML 模型介面設計 UML 類別圖

依據圖 4-1 類別圖，描述 XML 模型介面內容及彼此關係如下：

模型包含 Veracity、Dimension、Property、Quantization 及 Model 計 5 個類別。Veracity 物件為真實模型為真實維度之集合，並以真實維度展開計算真實程度，具有 dim 維度 list 屬性，及計算程度分數的 calc 方法、增加真實維度的 add_dim 方法，由 Model 物件建立，一個 Veracity 可對應 1 個以上的 Dimension 物件。

Dimension 真實維度為真實屬性之集合。具有維度名稱 name 的字串型別屬性、property 屬性 list 屬性，及維度將命名的 set_name 方法、增加真實屬性的 add_property 方法。由 Model 物件建立，一個 Dimension 可對應 1 個以上的 Property 物件。

Property 真實屬性為量化因子之集合。具有 quartization 屬性，及添加量化因子的 add_quantization 方法。由 Model 物件建立，一個 Property 對應 1 個 Quantization 物件。Quantization 量化因子用以計算真實程度分數，具有計算分數 func 方法供使用者實作。

基於圖 4-1 表示出真實模型、真實維度、真實屬性及量化因子彼此之間的關聯性，實心菱形箭頭起始表示兩者之間的組合（Composite aggregation）關係，以數字表達個物件 1 對多或 1 對 1 的組合特性，並且透過 Model 建立實例，成為 XML 真實模型完整的模型介面示意。

4.1.3 Python 模型介面設計

在進行資料分析流程中，使用者常用的 Python 程式語言作為模型設計媒介。Python 為直譯式語言，亦屬物件導向程式，可在未經修改下於 Windows、Linux、Mac OS 等作業系統跨平台使用，其簡潔的語法哲學、透過有著自動判定型別的，加上強大的模組擴充性，無論是運用在資料存取、機械學習、深度學習、網站架設、資料視覺化等領域其代表性與時俱增，依據專門追蹤軟體程式碼品質的 TIOBE[30]調查顯示，截至 106 年 6 月 Python 為第四大熱門的程式語言，而 IEEE Spectrum 調查顯示，Python 成為 2017 年最受歡迎的程式語言第一名[28]，依據上節 UML 模型，我們以 Python 實作此模型介面，達到彈性及適應性之目的。

具體模型介面設計而言，本研究透過 Python 動態語言的特性，專注於設計物件產生的行為，舉例如以物件導向設計鴨子，「當他走路像鴨子、游泳像鴨子、叫聲像鴨子，那他就是隻鴨子」[8]，而非物件的種類[40]。以類別的組合代替相互繼承，減少使用者在程式實作時因為模型的階層特性，使類別相互呼叫可能造成的繼承混淆問題。我們以 Python 進行 XML 真實模型介面設計如下：

設計真實程度類別（clsss Veracity），如圖 4-2 所示，所有的 Python 物件皆繼承於 Object 物件，可將”class Veracity(Object):”表示為”class Veracity:”以保持程式碼簡潔性（第 2 行）；連續 3 個雙引號所包含的文字為 Veracity 物件的說明內容，可跨行紀錄文字，當使用者以 help()查詢物件說明時，將會回傳該字串（第 3 至 4 行）及物件架構；使用__init__()方

法（Method）定義在類別之中表示為初始化流程，方法前後各有兩個底線，在 Python 中代表除了在類別以外的其他位置不要直接呼叫（第 5 行），在呼叫__init__()方法時，建立的類別實例會傳入作為方法的第一個參數，第一個參數必須明確作為接受實例之用[39]，慣例上取名為 self（第 5 行）。

在建立類別實例時，可從 self 之後依序定義參數，若要設定實例屬性，透過 self 與點運算子來設置。建 Veracity 類別實例時，使用者需輸入 dim 會在建立的實例上增加 dim 參數。其中 dim 為表示真實維度的 Veracity 類別參數，*dim 表示該參數接受不定長度的引數，如：”D1”,”D2”。Python 預設該特性為 Tuple 內建形態，與串列（List）相似，在本模型介面設計使用者輸入的 dim 值皆會透過 list(dim)轉換為串列，轉換結果如[”D1”,”D2”]所示，如果使用者未輸入任何參數，則本模型預設 dim 將建立空串列[]，便於後續對 dim 的賦值操作。（第 7 行）

本研究設計 Veracity 類別包含計算程度分數的 calc()方法、增加真實維度的 add_dim()方法，calc()方法由使用者輸入分數 score 及權重 weight，計算結果回傳至 self.score（第 12 至 14 行）；呼叫 add_dim()方法需提供參數 d，參數 d 將增加於 Veracity.dim 串列之後，達成模性介面設計所需增加真實維度之功能。（第 16 至 17 行）

```
1   #設計 Veracity 物件
2   class Veracity:
3       """真實模型為真實維度之集合，並以真實維度展開
4       計算真實程度"""
5       def __init__ (self, *dim):
6       # *dim 參數傳入建立 List，如無責建立[]
7           self.dim = list(dim) if dim else []
8       #  建立計算 score 的方法
9       def calc(self, score, weight=1):
10          self.score = score*weight
11          return self.score
12      #建立增加 dim 在 dim list 的方法
13      def add_dim(self, d):
14          self.dim.append(d)
```

圖 4-2 XML 真實模型介面-設計 Veracity 物件

設計真實維度類別（clsss Dimension），目的係達成真實維度為真實屬性之集合之功能，如圖 4-3 所示，

使用者建立 Dimension 類別實例時需輸入表示真實屬性名稱的 properties 參數，在__init__()方法中，第二個參數*properties 為表示真實維度的 Dimension 類別參數，該參數接受不定長度的引數，如未輸入則建立 properties 空串列以便後續操作（第 5 至 7 行），使用者輸入 properties 參數無論是否為複數，將依序建立為 properties 串列（第 8 行）。

Dimension 類別提供設定 Dimension 名稱的 set_name()方法，便於使用者建立實例時建立可供識別之名稱（第 9 至 12 行）；亦有 add_property()方法，呼叫 add_property ()方法需提供參數 p，參數 p 將增加於 Dimension.properties 串列之後，達成模性介面設計所需增加真實屬性之功能。（第 14 至 15 行）

```
1   #設計 Dimension 物件
2   class Dimension:
3       """真實維度為真實屬性之集合"""
4       def __init__(self, *properties):
5           # 如果沒有 properties，則建立 list
6           if properties is None:
7               properties = []
8           self.properties = list(properties)
9       # 建立設定 name 的方法
10      def set_name(self, name):
11          self.name = name
12          print('set Dimension: {}'.format(name))
13      #建立增加 property 在 property list 的方法
14      def add_property(self, p):
15          self.properties.append(p)
```

圖 4-3 XML 真實模型介面-設計 Dimension 物件

設計真實屬性類別（clsss Property），目的係達成真實屬性為量化因子之集合之功能，如圖 4-4 所示，使用者建立 Property 實例時需輸入表示量化因子的 quantization 參數，如未輸入則建立 quantization 空串列以便後續操作（第 5 至 7 行），使用者輸入 quantization 參數無論是否為複數，將依序建立為 quantilization 串列（第 8 行）。

Property 類別提供設定 add_quantization ()方法，呼叫 add_quantization ()方法需提供參數 q，參數 p 將增加於 Property. quantization 串列之後，達成模性介面設計所需增加量化因子之功能。（第 14 至 15 行）

```
1   #設計 Property 物件
2   class Property:
3       """真實屬性為量化因子之集合"""
4       def __init__(self, *quantization):
5           # 如果沒有 quantization，則建立 list
6           if quantization is None:
7               quantization = []
8           self.quantization = list(quantization)
9       # 建立設定 name 的方法
10      def add_quantization(self, q):
11          self.quantization.append(q)
```

圖 4-4 XML 真實模型介面-設計 Property 物件

設計量化因子類別（class Quantization），目的係達成透過量化因子用以計算真實程度分數之功能，如圖 4-5 所示，量化因子依本模型介面 UML 在建立 Quantization 實例時初始化以 pass 表示不作任何動作（第 4 至 5 行）。

Quantization 類別提供計算量化因子的計算方法 calc_method()，該方法具備定義計算公式之計算方法 calc_method 參數，參數本身亦可為物件，使用者透過設計之計分方法與計分結果，舉例如使用後續 XML 真實模型實作所定義的 xml_version()方法，回傳計分結果，達成模性介面設計所需增加量化因子之功能。

```
#設計 Quantization 物件
class Quantization:
    """量化因子用以計算真實程度分數"""
    def __init__(self):
        pass
    #設定量化因子計算方法
    def calc_method (self, calc_method):
        self.calc_method = calc_method
        return self.calc_method
```

圖 4-5 XML 真實模型介面-設計 Quantilization 物件

另一方面，XML 模型介面需要具備處理 XML 檔案、XML 串流、甚至 XML 大檔案之能力。Python 可處理 XML 的模組之中，例如基於 W3C DOM（Document Object Model）規範實現的 xml.dom 模組，DOM 處理時必須先輸入整個檔案進行剖析，以便能夠對文件的各部分進行存取，對於較大的 XML 文檔及串流資訊處理較難，如果超過記憶體負荷則將導致錯誤發生。另有 xml.sax 模組為使用 SAX（Simple API for XML），係基於事件的 API，一邊讀取 XML 文件一邊進行剖析，可針對剖析過程特定事件進行處理，即可適用於大型及串流 XML 文件情境。

本研究 Python 模型介面設計採用官方推薦之 xml.etree.ElementTree 模組（以下簡稱 ET），該套件為輕量級的 Python API，係由 C 語言實現，比原本以 Python 實作底層的版本快上 15 至 20 倍[29]，在 Python 3.3 版本之前，要在介面使用 C 實作的 ET 必須讀入 xml.etree.cElementTree 套件而與原本使用 Python 實作的 ET 不同，自 Python 3.3 版之後 xml.etree.ElementTree 套件統一為 C 語言實作。

ET 模組可在解析和創建 XML 情境中使用，針對快速解析和記憶體使用進行了優化，ET 與 xml.dom 相比速度更快[29]；ET.iterparse 也提供如同 xml.sax 般的巨量、即時解析的解決方案，無需將整個文檔加載到內存中，皆可處理 XML 檔案及 XML 串流資訊。

預設 ET.iterparse 方法之參數包含 source、events 及 parser[35]，其中資料來源 source 參數是指 XML 檔案名稱或 XML 串流，XML 串流為一系列包含 XML 文件的事件，在 ET.iterparse 方法的 events 參數('start', 'end', 'start-ns', 'end-ns')可設計每個步驟解析 XML 的方式，start 將首次遇到標籤時觸發；end 將在遇到結束標記時觸發，並且已經讀取了其中的所有內容；'start-ns'及 'end-ns'分別表示遇到命名空間時的處理方式。當使用者要處理 XML 串流事件或巨量 XML 文件時，可透過建立以 ET.iterparse 處理 XML 文件的物件，將該物件作為量化因子 Quantization 類別之 calc_method()方法之參考物件，即可將計算結果回傳與呈現。

量化因子 calc_method ()方法的 calc_method 參數本身亦為物件，可直接引述使用 ET 所設計的計分方法，使 XML 模型介面之資料來源可以是 XML 檔案及 XML 串流事件，達到模型介面設計彈性及適應性目的。

第五章 XML 真實模型建立與實作

5.1 XML 真實模型建立

本節建構「XML 真實程度模型」（Veracity Model of XML）以具體描述 XML 文件資料的特性是否具有參考價值的真實程度模型，用以鑑別文件的真實程度，該模型可適用在巨量資料環境，提供使用者參考。其方式包括資料來源多元、文件結構變化多端、有無符合 DTD 或 XML schema 規範，以及可能有同名異義或異名同義的元素等原因納入考量，使 XML 文件資料上 veracity 的特性可以加以程度的區別。

XML 真實程度模型係循第三章建立的「真實程度模型」思維進行建構，首先依照文件的假設有一份基準 XML 文件 B 與被量測 XML 文件 M；而文件 M 將被量測與文件 B 的真實程度。使用者可依照鑑別真實程度的動機與目的，在觀察基準文件 B 後，自行定義出描繪真實程度輪廓的數個真實維度（veracity dimensions），依照描述維度的不同特徵及特徵量化計算方法，設計各真實維度的真實屬性（veracity attributes）及量化因子（quantization factors），並決定量化真實程度的尺度，使用者可權衡真實屬性的重要性調整計分權重，以符合使用者鑑別真實程度的動機與目的。為了使用者能建立 XML 真實模型建立，進一步提供使用者真實模型建立方法與本研究所提供的真實模型架構與實作。

5.1.1 真實模型建立方法

真實程度的決定是一個「開放的問題」（open issue），會因不同應用而有不同的真實維度，也涉及使用者對資料理解的程度。本研究參考 2014 年 Daniela Fogli 等提出針對企業網站的品質評定方法「EQ-EVAL assessment methodology」[11]，運用於使用者評量真實程度的方法，透過「準備（Preparation）」、「分析（Analysis）」、「測量（Measure）」及「綜合處理（Synthesis）」等 4 階段程序評量 XML 真實程度。

首先「準備」階段，使用者應搜集相關資訊，諸如：識別真實程度的動機與目標、基準 XML 本身的目標，以及建立可接受的真實程度範圍的定性尺度與評估基準，將有助後續真實程度的判別及釐清。

其次「分析」階段，分析基準 XML 文件的組成要素、選擇需要評估的真實維度，以及選擇每個維度的重要真實屬性，在不失去評估準確性的情況下可以丟棄的那些的識別對效率有重要的影響的特性。

第三「測量(Measure)」階段，定義採取真實程度的程度衡量尺度(scale)，進而設計每個組成成分的量化因子，建立評量方法。必須注意的是，對所有量化因子設定相同的尺度標準將有助於後續的綜合處理，而設計評量方法時須考量模型及文件的目標及動機、量化因子的期望真實程度，並可依據專家判斷[13]或使用者對資料的理解建立評量方法。

最後「綜合處理（Synthesis）」階段，整合所有量化因子的真實程度評分結果，提出表格或圖形的真實程度模型結果報告。

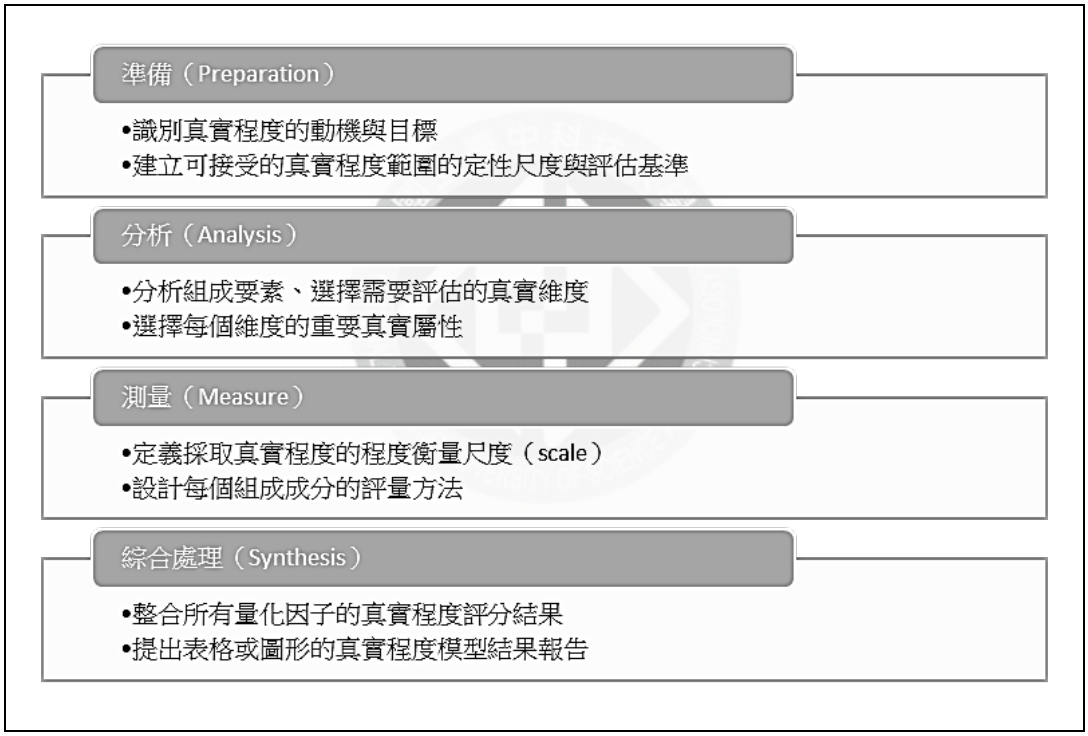

圖 5-1 EQ-EVAL assessment methodology

本研究整理使用者應具備的建立巨量 XML 真實模型方向，包含(1) 識別真實程度的動機與目標；(2)決定真實維度；(3)計算分數的方法；(4)程度計分採用的尺度。首先識別真實程度動機與目標，有助於釐清使用者判斷立場及判斷依具，牽涉真實程度模型架構偏向 XML 內容或結構，俾憑真實模型中重要或不重要之真實屬性，調整計分權重以反映識別真實程度之動機與目標；第二為「真實維度」的整體架構將會決定 XML 真實模型的輪

廓，使用者對資料理解越清晰，將有助於真實模型的效果；第三為決定真實模型的計分方法，透過真實屬性之量化因子設計程度判別方式，運用程式設計進行實作以減少人工判別負擔；第四為設計採用的計分尺度，即如何用量化方式表達程度高低，本研究以 0 至 100 的計分採用尺度，以便後續綜合整理模型的評分結果。

5.1.2 真實模型架構建立

在建立真實模型、設計真實模型介面之後，為了針對鑑別 XML 文件的真實程度，將巨量資料來源多元、文件結構變化多端、有無符合 DTD 或 XML schema 規範，以及可能有同名異義或異名同義的元素等原因納入考量，使 XML 文件資料上 veracity 的特性可以加以程度的區別，本研究建構「XML 真實程度模型」（Veracity Model of XML）以具體描述 XML 文件資料的特性是否具有參考價值的真實程度模型，用以鑑別文件的真實程度，該模型可適用在巨量資料環境，提供使用者參考。

其次，真實模型須決定真實維度 D_i 相關的真實屬性 $P_{i,j}$。再者，真實屬性 $P_{i,j}$ 之量化因子 $Q_{i,j,k}$ 用以計算各項量化分數，本研究將各項量化因子真實程度分數由最小至最大設計為 0 至 100，真實屬性 $P_{i,j}$ 分數為所屬各量化因子 $Q_{i,j,k}$ 分數加總、真實維度 D_i 之真實程度為所屬各真實屬性 $P_{i,j}$ 加總，XML 真實模型之真實程度為各真實維度 D_i 分數加總。

本研究歸納出一些對於基準 XML 文件而言具有資料理解性的真實維度，例如：文件來源 D_1、文件結構 D_2、文件內容 D_3、時間先後 D_4 等維度。文件來源 D_1 可以取得 XML 文件宣告的重要資訊、文件結構 D_2 可以依照 XML 文件整體樹狀結構進行分析評估、文件內容 D_3 涉及 XML 文件的元素及路徑特性、時間先後 D_4 可以用時間次序進行評估，在上述列舉的真實維度中，分別找出每個維度相關的真實屬性。例如，文件資料來源的真實屬性可以包括來源之 URI（universal resource indicator）、編碼、URL 網域層級碼、...等； 結構的屬性可包括 DTD、XML schema、DataGuides 等；文件的內容屬性可包括路徑、路徑與其節點值、文件節點數量、文件規模等； 而時間的屬性可包括文件建立、修改時間的先後次序、...。

另外，由於 XML 廣泛運用於各領域，不同 XML 文件對於各項真實維度及真實屬性重要性不同，在實際計算真實程度時，使用者可透過對資料理解性進行各項計分權重調整，計算 D_i 的值時，以 $w_{i,j}$ 做為正規化各真實程度 $|P_{i,j}|$ 之係數，以產生具有資料理解及代表性的真實程度。

不同的 XML 真實維度 D_i 及其真實屬性 $P_{i,j}$ 分別有不同特性，為方便使用者從資料理解性（data understandability）層面了解真實維度模型，本章建立 XML 真實模型，其中各真實維度 D_i 及其真實屬性 $P_{i,j}$ 、$Q_{i,j,k}$，舉例說明如下：

在真實維度-文件來源 D_1 中，依真實模型設計真實屬性為 $P_{1,1}$「文檔宣告」及 $P_{1,2}$「URI」。

表 5-1 真實維度 D_1、真實屬性及量化因子

真實維度	真實屬性	量化因子
D_1 = 文件來源	$P_{1,1}$ = 文檔宣告	$Q_{1,1,1}$ = XML 版本、 $Q_{1,1,2}$ = 編碼
	$P_{1,2}$ = URI	$Q_{1,2,1}$ = URI 網域名稱 $Q_{1,2,2}$ = URN $Q_{1,2,3}$ = URL

真實屬性 $P_{1,1}$「文檔宣告」係指 XML 文件第一行所標示的內容，如：<?xml version="1.0" encoding="UTF-8" standalone="yes"?>，其中版本宣告（version）為必要項，編碼（encoding）及獨立宣告（standalone）為選擇項，比對的文件是否有宣告，抑或是是否有相同宣告，即可做為評判真實程度參考。鑑於現有 XML 版本有 1.0、1.1，及尚在討論階段未正式公告的 2.0 版等，其中 XML 1.0 於 1998 年由 W3C 定義發布，至 2008 年更新至第五版，為現今較常被採用的版本；XML 1.1 於 2004 年發布，於 2006 年更新至第二版，主要係配合 XML 採用的 Unicode 編碼系統版本持續演進而發展[10]，惟 XML 1.1 並非完全向下相容，W3C 建議如非特殊環境應採用 1.0 版本為主；XML 2.0 僅在討論中尚未發布，其設計有向下兼容 1.0 版本，即 XML 2.0 版文件可以採用 1.0 版的特徵。基於前述版本編號及是否向下

兼容的特性，可以做為計算真實程度的依據。本研究設計 XML 版本是否批配及向下兼容性為 $Q_{1,1,1}$「XML 版本」，編碼是否相符為 $Q_{1,1,2}$「編碼」。

真實屬性 $P_{1,2}$「URI（Uniform Resource Identifier）」，URI 是 XML 文件提供一種避免資料元素名互相稱衝突的方法，使用其獨特性用來識別某項資源，將 URI 的參引加到元素或屬性名稱的前面，能使名稱具有單一性。通常，URI 以網域名稱（domain name）及具有路徑識別性的文件系統組成，而網域名稱有階層性[44]，頂級域名（Top-level domains，TLDs）及國家代碼頂級域名（country code top-level domain，ccTLDs），前者如：com、gov、org、edu；後者如：tw（台灣）、de（德國）、eu（歐盟）、jp（日本）、tw（台灣）等，除了頂級域名，還有二級域名（SLD，second-level domain），即最靠近頂級域名左側的欄位。如：zh.wikipedia.org 中，wikipedia 就是二級域名（如在頂級域名後面，還存在一級域名，則 zh 為二級域名），URI 在階層性描述（hierarchical part）之後亦可能存在有搜尋條件（query）及分段（fragment）等特徵。本研究設計量化因子 $Q_{1,2,1}$URI 名稱依照 URI 網域及搜尋條件、分段描述的階層性進行比對，層級由高至低比對結果作為評估方法。

另外，XML 文件亦有採用 URN[45]用以識別，URN 用以定義某件事物的身份，例如：ISBN 0-486-27557-4(urn:isbn:0-486-27557-4)無二義性地標識出莎士比亞的戲劇《羅密歐與朱麗葉》的某一特定版本[45]，圖 5-2 為[45]舉例 URI 及 URN 的組成方式；URL 位址（Uniform Resource Locator）為網路識別來文之方式，XML 文件之下載位置以「協定類型:[//伺服器位址[:埠號]][/路徑]檔案名[?查詢][#片段]」[31]組成，鑑於 URN 及 URL 其獨特性，本研究設計量化因子 $Q_{1,2,2}$ URN 路徑、$Q_{1,2,3}$ URL 路徑的階層特性計算真實程度評估方法。

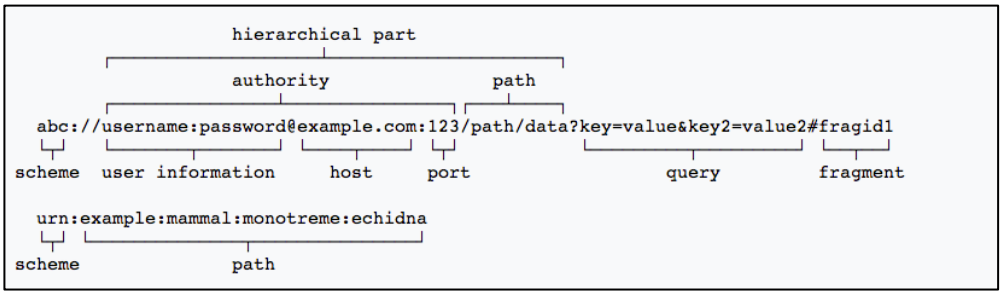

圖 5-2 URI 及 URN 的組成方式

在真實維度-文件結構 D_2 中，依真實模型設計真實屬性為 $P_{2,1}$「結構規範」及 $P_{2,2}$「結構比對」。

表 5-2 真實維度 D_2、真實屬性及量化因子

真實維度	真實屬性	量化因子
D_2 = 文件結構	$P_{2,1}$ = 結構規範	$Q_{2,1,1}$ = valid XML $Q_{2,1,2}$ = DTD 或 XSD
	$P_{2,2}$ = 結構比對	$Q_{2,2,1}$ = 最大結構深度 $Q_{2,2,2}$ = 最大樹寬、 $Q_{2,2,3}$ = 元素數量

真實屬性 $P_{2,1}$「結構規範」係指 XML 文件必須是結構完整的（well-formed），如果 XML 文件中的元素、屬性以及其他單元（如 Entity、Notation ... 等），皆符合其引用的 DTD 或 XML Schema 之規範，則稱為 valid XML 文件，該文件結構是否遵守規範、以及該文件遵守規範為何，可作為真實維度參考。使用者可將基準與比對文件將是否為 valid XML 文件作為量化因子 $Q_{2,1,1}$，以 XML 是否引述相同的 DTD 或 XSD 作為量化因子 $Q_{2,1,2}$，判斷其相似程度。

真實屬性 $P_{2,2}$「結構比對」可借重 DataGuides [21]方式協助 2 份 XML 文件的真實程度。DataGuides 表示樹狀資料結構文件有路徑的摘要，依照結構樹搜尋方式，運用在 XML 文件的 DataGuides 中僅以該序列表示路徑摘要，可描述 XML 資料結構、提升查詢效率，能因應巨量資料有著量大、快速及多變之情況不明特性，隨著 XML 資料讀取組成之樹狀結構進行描述。本研究透過 DataGuides 處理的樹狀結構，將 XML 中相同元素、屬性組合的結構特徵設計 $Q_{2,2,1}$「最大結構深度」、$Q_{2,2,2}$「最大樹寬」及 $Q_{2,2,3}$「元素數量」，計算樹狀結構相似程度。

在真實維度-文件內容 D_3 中，依真實模型設計真實屬性為 $P_{3,1}$「元素」、$P_{3,2}$「屬性」及 $P_{3,2}$「路徑」。

表 5-3 真實維度 D_3、真實屬性及量化因子

真實維度	真實屬性	量化因子
D_3 = 文件內容	$P_{3,1}$ = 元素	$Q_{3,1,1}$ = 元素名稱、 $Q_{3,1,2}$ = 元素數量 $Q_{3.1.3}$ = 同義元素
	$P_{3,2}$ = 屬性	$Q_{3,2,1}$ = 屬性名稱、 $Q_{3,2,2}$ = 屬性數量
	$P_{3,3}$ = 路徑	$Q_{3,3,1}$ = 指定路徑 $Q_{3,3,2}$ = 路徑數量

在真實維度-文件內容 D_3 中，可針對 XML 文件的元素、路徑、元素相依性等內容進行真實程度估算，舉例如關鍵元素名稱存在與否、關鍵元素數量、總元素數量；關鍵元素之間之路徑、相關路徑數量；關鍵父子元素、兄弟元素相依關係等，使用者可透過對資料理解的程度自行定義文件內容的真實程度計算方式。

真實屬性 $P_{3,1}$「元素」係指 XML 文件中的元素（tag），在一份 XML 文件由諸多元素組成的樹狀結構，不同應用領域的元素命名規則不同，元素的數量及重要程度亦有不同，舉例如：交換氣象資訊的 XML 文件中以 <temp>元素紀錄溫度，交換金融資訊的 XML 文件未必需要具有該元素。使用者可透過對資料的理解性，針對基準文件關鍵的元素名稱及元素數量，以及針對真實意義相近之元素，分別作為量化因子 $Q_{3,1,1}$、$Q_{3,1,2}$ 及 $Q_{3.1.3}$。

真實屬性 $P_{3,2}$「元素屬性」係指 XML 文件中元素內所描述的屬性，舉例如：<元素 屬性=value>內容</元素>，由於真實世界中採用資料紀錄方式不同，有些 XML 文件將交換的數值紀錄於內容中，有些則直接紀錄於屬性中，故對於元素屬性的運用方式亦屬於 XML 文件特徵之一，使用者即可依其對資料理解性，設計量化因子 $Q_{3,2,1}$「屬性名稱」及 $Q_{3,2,2}$「屬性數量」。

真實屬性 $P_{3,3}$「路徑」係指元素與元素之間路徑的相似性，由於路徑牽涉到 XML 文件資料描述邏輯，不同 XML 文件描述邏輯有別，元素間走訪路徑亦有所差異，指定走訪路徑的存在與否可作為量化因子 $Q_{3,3,1}$，相同走訪路徑在數量上的差異可作為量化因子 $Q_{3,3,2}$，以估算基準文件與量測文件之間真實程度。使用者對於文件內容的理解，分別透過元素、屬性、路徑及相依性分析及估算真實程度，可對於基準文件與量測文件之間真實程度有具體的衡量基準。

在真實維度-文件時間 D_4 中，依真實模型設計真實屬性為 $P_{4,1}$「檔案時間」及 $P_{4,2}$「更新頻率」。

表 5-4 真實維度 D_4、真實屬性及量化因子

真實維度	真實屬性	量化因子
D_4 = 時間	$P_{4,1}$ = 檔案時間	$Q_{4,1,1}$ = 建立時間
	$P_{4,2}$ = 更新頻率	$Q_{4,2,1}$ = 文檔更新頻率

文件時間 D_4 可依建立檔案的「檔案時間」、「順序性」來判斷其文件特徵。使用者可以設計符合檔案建立時間的邏輯，例如擷取即時性資訊時，資料建立時間與擷取資料時間相差 1 月，則缺乏真實性；另外如果以固定週期發佈的資料，是否間隔週期相同也可以作為真實評估標準。

真實屬性 $P_{4,1}$「檔案時間」係指在應用 XML 資料時，所應具備的時間特性，如果建立檔案的時間超過容許範圍，舉例如基準文件為今日建立之檔案，量測文件係於 5 日前所建立，則該檔案真實程度應有差異，使用者可以針對該基準文件對於時間特性的要求設計合理建立時間區間，依據合理建立區間進行真實程度估算，即以建立時間設計為量化因子 $Q_{4,1,1}$。

真實屬性 $P_{4,2}$「文檔更新頻率」係指超過更新頻率所產生的 XML 文件應有真實程度差異，舉例如基準文件更新頻率為 15 分鐘，而量測文件的更新頻率為 1 小時。使用者可依據 XML 更新頻率定義量化因子 $Q_{4,2,1}$。

綜合上述所舉例設計的各真實維度、真實屬性及量化因子對應關係，使用者可參考下表 5-5，綜合 XML 設計真實程度模型：

表 5-5 建立真實模型、真實維度、真實屬性及量化因子

真實維度	真實屬性	量化因子
D_1 = 文件來源	$P_{1,1}$ = 文檔宣告	$Q_{1,1,1}$ = XML 版本 $Q_{1,1,2}$ = 編碼
	$P_{1,2}$ = URI	$Q_{1,2,1}$ = URI 網域名稱 $Q_{1,2,2}$ = URN $Q_{1,2,3}$ = URL
D_2 = 文件結構	$P_{2,1}$ = 結構規範	$Q_{2,1,1}$ = valid XML $Q_{2,1,2}$ = DTD 或 XSD
	$P_{2,2}$ = 結構比對	$Q_{2,2,1}$ = 最大結構深度 $Q_{2,2,2}$ = 最大樹寬 $Q_{2,2,3}$ = 元素數量
D_3 = 文件內容	$P_{3,1}$ = 元素	$Q_{3,1,1}$ = 元素名稱 $Q_{3,1,2}$ = 元素數量 $Q_{3.1.3}$ = 同義元素
	$P_{3,2}$ = 元素屬性	$Q_{3,2,1}$ = 屬性名稱 $Q_{3,2,2}$ = 屬性數量
	$P_{3,3}$ = 路徑	$Q_{3,3,1}$ = 指定路徑 $Q_{3,3,2}$ = 路徑數量
D_4 = 文件時間	$P_{4,1}$ = 檔案時間	$Q_{4,1,1}$ = 建立時間
	$P_{4,2}$ = 更新頻率	$Q_{4,2,1}$ = 文檔更新頻率

5.2 XML 真實模型實作

在建立 XML 真實模型之後，使用者得依據模型計算真實程度，透過各維度相關的「真實屬性（veracity attributes）」將資料邏輯的理解進行定義，再透過「量化因子」（quantization factors）具體量化做為可供評估真實程度的數值。使用者可透過對資料的理解設定量化因子計算方式，鑒於提供使用者可選擇的實作方式，分別依可計算 XML 真實模型真實程度之演算法及實作方式說明如下所述。

5.2.1 真實維度-文件來源 D1

在真實維度-文件來源 D1 中，依真實模型設計真實屬性為 $P_{1,1}$「文檔宣告」及 $P_{1,2}$「URI」。

$P_{1,1}$「文檔宣告」之真實屬性建立，本節以「XML 版本是否相同」為量化因子 $Q_{1,1,1}$，「文件編碼（語言）」是否相同為 $Q_{1,1,2}$。以 XML 宣告出現於文件第一行的資訊，如：<?xml version="1.0" encoding="UTF-8" standalone="yes"?>萃取相關資訊進行真實程度計算。

$Q_{1,1,1}$「XML 版本」分數設計：如果基準文件與量測文件相符者，分數為 100；如果不相符者，基準文件如為 2.0 版，量測文件如為 1.0 版，分數為 70 分；如非前述情形則分數為 0 分，並回傳為數值。

演算法為：透過正規表達式方法基準文件 b_XML 及量測文件 m_XML 將取出宣告的版本編號（第 3 至 4 行），正規表達式設計以’ r'\d\.\d''篩選文件，取出的結果即為 1.0、1.1 及 2.0 等字串，取值結果分別回傳於 b_version 及 m_version 變數，在 Python 程式語言會在運算子進行運算時，自動將 1.0、1.12 等字串轉換為浮點數；進行 XML 版本真實程度條件判斷式進行真實程度分數評估（第 5 至 11 行），如果 b_version 等於 m_version，則分數為 100，另因 XML 版本可以向下相容，如果 b_version 為 2.0、 m_version 為 1.0，則分數為 70，其他比對情形分數為 0。

```
1  #比對 XML 版本
2  def xml_version(b_XML, m_XML):
3      b_version = b_xml 文件版本編號
4      m_version = m_xml 文件版本編號
5      if b_version == m_version:
6          return score = 100
7      elif b_version == 2.0 and m_version == 1.0:
8          return score = 70
9      else:
10         score = 0
11     return score
```

圖 5-3 比對 XML 版本演算法

量化因子 $Q_{1,1,2}$「文件編碼（語言）」鑑於 XML 文件有宣告編碼的特性，以及考量 XML 文件宣告編碼與實際 XML 文件資料內容編碼不符的可能性，文件編碼可採取兩種方式判別真實程度，其一是依據該 XML 文件宣告編碼進行判別，如：encoding="UTF-8"該編碼即為 UTF-8，可以透過正規表達式尋找該編碼字串；第二種方法式透過讀取整份文檔直接判讀，直接判讀文件的編碼可以借助 Python 的 chardet 模組[12]。chardet 源自於 Mozilla 瀏覽器判別編碼的編碼識別模組[17]，可如同瀏覽器般自動偵測並判別編碼，惟自動偵測非完全正確，chardet 模組將回傳自動偵測結果及可信程度，依其官方文件說明，如果遇到 XML 文檔宣告與 chardet 偵測結果不同時，應以文檔宣告之編碼為主。使用者亦可採用近似編碼者給予不同程度的分數，本研究設計以：如果基準文件與量測文件相符者，分數為 100；如果不相符者則分數為 0 分，並回傳為數值。

第一種量化因子 $Q_{1,1,2}$「文件編碼（語言）」演算法為：透過正規表達式 r'encoding[\s=[(\'|\")\w\d(\-|\.)]*'字串（第 3 行）取出基準文件 b_XML 及量測文件 m_XML 中宣告的版本編碼（第 4 至 5 行），取值結果回傳於 b_encoding 及 m_encoding；比對 b_encoding 及 m_encoding 是否相符，如果相同則分數為 100，如果不同則分數為 0，回傳相似程度（第 5 至 10 行）。

```
1  #比對宣告編碼(方法一)
2  def xml_encoding(b_XML, m_XML):
3     regex = r'encoding[\s=[(\'|\")\w\d(\-|\.)]*'
4     b_encoding = b_xml 編碼
5     m_encoding = m_xml 編碼
6        if b_ encoding == m_ encoding:
7           score = 100
8        else:
9           score = 0
10    return score
```

圖 5-4 比對宣告編碼(方法一)演算法

舉例第二種量化因子 $Q_{1,1,2}$「文件編碼（語言）」以 chardet 模組進行自動偵測之算法為：透過 glob 模組讀入基準文件及量測文件所在的資料夾，以二進位制讀取 XML 文件檔案（第 6 行）；運用 chardet 模組之 UniversalDetector 方法偵測基本 b_XML 文件及量測文件 m_XML 之編碼，將偵測結果的編碼分別回傳至 b_encoding 及 m_encoding（第 7 至 8 行）；比對 b_encoding 及 m_encoding 是否相符，如果相同則分數為 100，如果不同則分數為 0，回傳相似程度。（第 9 至 13 行）。

```
1  #比對 chardet 自動偵測文檔編碼(方法二)
2  import glob
3  from chardet.universaldetector import UniversalDetector
4
5  def xml_chardet_encoding(b_XML, m_XML):
6     以二進位制讀取 basic_XML、measure_XML
7     b_encoding = basic_XML 檔案偵測編碼結果
8     m_encoding = measure_XML 檔案偵測編碼結果
9     if b_ encoding equal to m_ encoding:
10       score = 100
11    else:
12       score = 0
13    return score
```

圖 5-5 比對宣告編碼(方法二)演算法

真實屬性 $P_{1,2}$ 「URI」以量化因子 $Q_{1,2,1}$ 網域層級依照 URI 網域及搜尋條件、分段描述的階層性進行比對，層級由高至低比對結果作為評估方法；URN 相同與否比對結果作為量化因子 $Q_{1,2,2}$ 評估方法。

$Q_{1,2,1}$「URI 名稱」分數設計：如果基準文件及量測文件皆未引用，則分數為 30；如基準文件未引用而量測文件有引用，則分數為 50；如基準文件引用 URI 而量測文件未引用，則分數為 0；如果有引用 URI，真實程度分數= $(n_{(M|B)}/n_B)*100$，其中 n_B 為基準文件之 URI 總階層數，$n_{(M|B)}$ 為 M 文件與 B 文件之 URI 相似至第 n 階之階數，如果各階層比對結果皆相符則分數為 100，皆不符者分數為 0。

$Q_{1,2,1}$「URI 名稱」演算法為：讀入基準文件 b_xml 及量測文件 m_xml，利用正規表達式篩選 uri，正規表達式設計以 r'xmlns[\:(\=\'|\"\/)\w\d(\-|\.)]*'（第 3 行）將 b_xml、m_xml 擷取命名空間字串，分別將字串存入 b_re 及 m_re 變數（第 4 至 5 行）；接著判斷 b_re 如為空值，m_re 是否為空值分數分別為 30、50（第 6 至 10 行），如 b_re 存在而 m_re 為空值則分數為 0（第 11 至 12 行）。

接續將 b_re 及 m_re 透過分隔符號切割字串為串列（List），舉例如：['www' ,'example' ,'com']形式，其結果分別回傳至 b_uri_list 及 m_uri_list 串列變數（第 14 至 15 行）；將 b_uri_list 及 m_uri_list 進行比對，以迴圈觀察 m_uri_list 的每項成分是否同樣為 b_uri_list 成分，如果有的話則加入新的串列 c_list（第 16 行）。計算分數以 c_list 串列的長度除以 b_uri_list 長度乘以 100，即如果相似的階層越高，分數越趨近於 100，並回傳相似程度。（第 17 至 18 行）

```
1  #URI 計分方法
2  def xml_uri(b_XML, m_XML):
3      regex = r'xmlns[\:(\=\'|\"\/)\w\d(\-|\.)]*'
4      b_re = b_xml 利用 regex 篩選 uri 字串
5      m_re = m_xml 利用 regex 篩選 uri 字串
6      if b_re is None:
7          if m_re is None:
8              score = 30
9          else:
10             score = 50
11     elif m_re is None :
12         score = 0
13     else:
14         b_uri_list = b_re 依據分隔符號切割組成的串列
15         m_uri_list = m_re 依據分隔符號切割組成的串列
16         c_list = b_uri_list 及 m_uri_list 相同成份重組串列
17         score = (len(c_list) / len(b_uri_list))*100
18     return score
```

圖 5-6 URI 計分方法

$Q_{1,2,2}$「URN 名稱」分數設計：如果基準文件及量測文件皆未引用，則分數為 30；如基準文件未引用而量測文件有引用，則分數為 50；如基準文件引用 URN 而量測文件未引用，則分數為 0；如果有引用 URN，真實程度分數= $(n_{(M|B)}/n_B)*100$，其中 n_B 為基準文件之 URI 總階層數，$n_{(M|B)}$ 為 M 文件與 B 文件之 URI 相似至第 n 階之階數，如果各階層比對結果皆相符則分數為 100，皆不符者分數為 0。

$Q_{1,2,2}$「URN 名稱」演算法為：讀入基準文件 b_xml 及量測文件 m_xml，利用正規表達式篩選 uri，正規表達式設計以 r'urn[\:(\=\'|\"\/)\w\d(\-|\.)]*'（第 3 行）將 b_xml、m_xml 擷取包含'urn'內容的字串，分別將字串存入 b_re 及 m_re 變數（4 至 5 行）；接著判斷'urn'如不在 b_re 之中，m_re 是否有'urn'字串分數分別為 30、50（第 6 至 10 行），如 b_re 存在'urn'而 m_re 則否分數為 0（第 11 至 12 行）。

接續將 b_re 及 m_re 透過分隔符號切割字串為串列（List），舉例如：['www' ,'example' ,'com']形式，其結果分別回傳至 b_urn_list 及 m_urn_list 串列變數（第 14 至 15 行）；將 b_urn_list 及 m_urn_list 進行比對，將 b_urn_list 及 m_urn_list 的相同成分，加入並建立新的串列 c_list（第 16 行）。計算分數以 c_list 串列的長度除以 b_urn_list 長度乘以 100，即如果相似的階層越高，分數越趨近於 100，並回傳相似程度。（第 17 至 18 行）

```
1   #urn 計分方法
2   def xml_urn(b_XML, m_XML):
3       regex = r'urn[\:(\=\'|\"\/)\w\d(\-|\.)]*'
4       b_re =  依據 regex 篩選 b_xml 字串
5       m_re =  依據 regex 篩選 m_xml 字串
6       if 'urn' not in b_re:
7           if 'urn' not in m_re:
8               score = 30
9           else:
10              score = 50
11      elif 'urn' not in m_re:
12          score = 20
13      else:
14          b_urn_list = b_re 依據分隔符號切割組成的串列
15          m_urn_list = m_re 依據分隔符號切割組成的串列
16          c_list = b_urn_list 及 m_uri_list 相同成份重組串列
17          score = len(c_list) / len(b_urn_list))*100
18      return score
```

圖 5-7 URN 計分方法

$Q_{1,2,3}$「URL」分數設計：比對 XML 來源 URL 路徑，真實程度分數=$(n_{(M|B)}/n_B)*100$，其中 n_B 為基準文件之 URL 總階層數，$n_{(M|B)}$ 為 M 文件與 B 文件之 URI 相似至第 n 階之階數，如果各階層比對結果皆相符則分數為 100，皆不符者分數為 0，如使用者未提供明確 URL 者分數為 0。

$Q_{1,2,3}$「URL」演算法為：讀入基準文件 b_xml 及量測文件 m_xml 之 URL，利用正規表達式設計以 r'\W'將 URL 依據分隔符號切格組成串列，分別將串列存入 b_re 及 m_re 變數（3 至 4 行）；接著分別判斷 b_re 及 m_re 是否為空值 None，如為空值則表示 b_xml 或 m_xml 未引用 URN，則分數為 0（第 5 至 8 行）。

接續進行 b_url_list 及 m_url_list 進行比對，將 b_url_list 及 m_url_list 的相同成分，加入並建立新的串列 c_list（第 10 行）。計算分數以 c_list 串列的長度除以 b_url_list 長度乘以 100，即如果相似的階層越高，分數越趨近於 100，並回傳相似程度。（第 14 至 15 行）

```
1   #urn 計分方法
2   def xml_url(b_XML_URL, m_XML_URL):
3       b_re = b_XML_URL 依據分隔符號切割組成的串列
4       m_re = m_XML_URL 依據分隔符號切割組成的串列
5       if b_XML_URL is None:
6           score = 0
7       elif m_XML_URL is None:
8           score = 0
9       else:
10          c_list = b_url_list 及 m_url_list 相同成份重組串列
11          score = len(c_list) / len(b_urn_list))*100
12      return score
```

圖 5-8 URN 計分方法

5.2.2 真實維度-文件結構 D2

在真實維度-文件結構 D2 中，依真實模型設計真實屬性為 $P_{2,1}$「結構規範」及 $P_{2,2}$「結構比對」。

$P_{2,1}$「結構規範」之真實屬性建立，XML 文件必須結構完整（well-formed），而 XML 文件中的元素、屬性以及其他單元皆符合引用的 DTD 或 XML Schema 之規範將稱為合法的（valid XML），本研究設計量化因子 $Q_{2,1,1}$ 為是否屬於 valid XML，$Q_{2,1,2}$ 為採用是否同樣為 DTD 或 XSD 之規範。

$Q_{2,1,1}$「valid XML」之分數設計係偵測基準文件及量測文件是否為 valid XML，依據對資料的理解性設計分數，本研究假設如果同樣為非 valid XML 文件，分數為 20；如果基準文件為 valid XML 而量測文件則否，分數為 0，如果基準文件非 valid XML 而量測文件為 vaild 則分數為 50，如果皆為 vaild 則分數為 100。

演算法為：判別基準文件 b_XML 與量測文件 m_XML 是否為 vaild XML，記錄布林值於指定變數，首先需篩選文件是否有引用 DTD 或 XSD，透過 Python 第三方模組 lxml.etree，將 b_xml 及 m_xml 與各該引述的.dtd 或.xsd 檔進行比對，將驗證與否結果(Ture or False)分別回傳於 b_xml_valid 及 m_xml_valid 變數（第 3 至 4 行）。依 b_xml 及 m_xml 為驗證與否情形，回傳相似程度。（第 5 至 12 行）

```
1   #valid XM
2   def vaild_xml(b_XML, m_XML):
3       b_xml_valid = b_xml 驗證與否
4       m_xml_valid = m_xml 為驗證與否
5       if b_xml_valid is False and m_xml_valid is False:
6           score = 20
7       elif b_xml_valid is Ture and m_xml_valid is False:
8           score = 0
9       elif b_xml_valid is False and m_xml_valid is Ture:
10          score = 50
11      else: score = 100
12      return score
```

圖 5-9 valid XM 計分方法

$Q_{2,1,2}$「DTD 或 XSD」分數設計：如果基準文件或量測文件有引用相同的 DTD 或 XSD，為 100；如果分別採用不同的 DTD 格式或 XSD 格式，分數為 20；如因基準文件沒有引用 DTD 或 XSD，量測文件引用 DTD 或 XSD 則分數設計為 50；如基準文件與量測文件皆未引用，分數為 30。

演算法為：判別基準文件 b_XML 與量測文件引用類型，以正規表達式 regex= [xsd]+[dtd]+ '搜尋字串"dtd"及"xsd"（第 3 行），將 b_XML 及 m_XML 引用 DTD、XSD 或為引用的判別結果輸入至 b_type 及 m_type 變數（第 4 至 5 行）；依 b_xml 及 m_xml 引用型別類型，（第 6 至 15 行）回傳相似程度。

```
1  # DTD 或 XSD
2  def vaild_type(b_XML, m_XML):
3     regex = [xsd]+[dtd]+ '
4     b_type= b_xml 為引用類型
5     m_type= m_xml 為引用類型
6     if b_type is None
7       if m_type is None:
8          score = 30
9       else:
10         score = 50
11    elif b_type == m_type:
12       score = 100
13    else:
14       score = 20
15    return score
```

圖 5-10 DTD 或 XSD 計分方法

$P_{2,2}$ 「結構比對」之真實屬性建立，以 XML 文件為樹狀結構文件，具有根元素、文檔必須有關閉元素、元素對大小寫敏感、元素必須被正確的嵌套、屬性前必須加引號的基本特性，故在進行 XML 真實模型實作時，可將其文件節構特性分別作為量化因子，以處理 XML 模組進行解析。舉例將「文件結構深度」作為量化因子 $Q_{2,2,1}$，結構樹「最大樹寬」作為 $Q_{2,2,2}$，「元素數量」為 $Q_{2,2,3}$。

為方便後續的結構比對應用，模型實作採用 XML 檔案一次讀入、結構重複判別進行設計，Python 語言進行 XML 真實模型實作可採用 xml.etree.ElementTree.XMLParser 模組，經由本研究設計的處理方法，將 XML 文件轉為巢狀列表（nested list）。

其算法為：從 xml.etree.ElementTree 輸入 XMLParser 模組（第 2 行）；建立 XMLParser 處理方法 XML2List，透過 XML 文件讀取過程中，每次遇到元素時，如果尚未達最大深度則深度加 1，並建立該階層的「深度、元素及屬性」串列，將前述每個串列資訊添加於 xml_list 末段(第 8 至 13 行)；走訪每一個葉節點元素處理方式為深度-1(第 15 至 16 行)，如果使用者呼架呼叫處理後資料所回傳資訊(第 17 至 19 行)。而第 21 至 26 行係說明如何運用 XMLParser 將 XML2List 方法運用於處理 XML 文件時機。

```
1  # XML 處理器 XML2List() 方法
2  from xml.etree.ElementTree import XMLParser
3
4  建立初始化列表 xml_list =[]
5  class XML2List:
6     maxDepth = 0
7     depth = 0
8     #走訪每一個元素的起始處理方式
9     def start(self, tag, attrib):
10       self.depth += 1
11       if self.depth > self.maxDepth:
12             self.maxDepth = self.depth
13       xml_list.append([self.depth, tag, attrib.keys()])
14    #走訪每一個葉節點元素處理方式
15    def end(self, tag ):
16        self.depth -= 1
17    #呼叫處理後資料所回傳資訊
18    def close(self):
19       print ('最大深度:{},'.format(self.maxDepth))
20
21 #建立 XMLParser 處理器實例
22 >>parser = XMLParser(target= XML2List() )
23 #讀入 XML_file 進行處理
24 >>parser.feed(open(XML_file).read())
25 #顯示 XML_filet 處理結果
26 >>print(xml_list)
```

圖 5-11 XML 處理器 XML2List()方法

另外本研究實作以 Python 語言為基礎之 DataGuides，萃取 XML 結構深度、元素、屬性等關鍵資訊，使用者可透過操作 List 截取所需資訊。DataGuides 演算法為：讀入經由前述 XML2List 方法所建立的 xml_list 串列（第 2 行），先建立新的串列 dataGuidesList（第 3 行），以 for 迴圈及條件判斷將未重複的「深度-元素-屬性」串列組加入 dataGuidesList，將 dataGuidesList 處理結果回傳（第 5 至 8 行）。

```
1  #建立基於 Pythom XML2List 的 dataGuides 方法
2  def dataGuides(xml_list):
3      '''以 XML2List 萃取出 DataGuides'''
4      dataGuidesList = []
5      for x in xml_list:
6          if x not in dataGuidesList :
7              dataGuidesList.append(x)
8      return dataGuidesList
```

圖 5-12 基於 Pythom XML2List 的 dataGuides 方法

在後續資料處理方面，除了解析前述的 xml_list 及 DataGuidesList 進行結構比對的方法之外，透過 Python 的 Pandas 模組以 DataFrame 資料格式進行資料處理可提供使用者便捷的資料操作。DataFrame 是 Python 受到 R 語言啟發實作的資料處理方式，係 column-oriented 二維表格結構，可以儲存字串、布林值、時間、數值等不同資訊，具有如同關連式資料庫數據處理功能，也便於進行篩選、切片等操作，圖 5-12 為 Pandas 的 DataFrame 建立方法，讀入資料並且設定索引欄位名稱為深度、元素與屬性。（第 4 行）

```
1  #以 DataFrame 進行資料處理方法
2  import pandas as pd
3  data = dataGuides(xml_ist)
4  df = pd.DataFrame(data, columns=['Depth','Tag','Attirb'])
```

圖 5-13 以 DataFrame 進行資料處理方法

$Q_{2,2,1}$「文件結構深度」為最大結構深度，計算方法以比對基準文件與量測文件最大深度，以深度大者為分母、深度小者為分子，分數以 0 至 100 計算，深度完全相同為 100，演算法為：偵測基準文件 b_XML 與量測文件 m_XML 之最大結構深度，偵測方式以前述 XML2List 方法解析的而成的 xml_list，或是使用 DataGuides 方法所萃取的 dataGuidesList 判別文件深度，b_XML 及 m_XML 文件最大深度分別回傳至 b_max_depth 及 m_max_depth 變數（第 3 至 4 行）。進行最大結構深度條件式判斷，如果 b_max_depth 等於 m_max_depth 則分數為 100，如果 b_max_depth 大於 m_max_depth，則分數為 m_max_depth 除以 b_max_depth 乘以 100；如果 b_max_depth < m_max_depth，分數為 b_max_depth / m_max_depth * 100，將分數結果回傳。（第 5 至 12）

```
1   # 最大結構深度
2   def xml_max_depth(b_XML, m_XML):
3       b_max_depth = b_XML 最大深度
4       m_max_depth = m_XML 最大深度
5           if b_max_depth == m_max_depth:
6               score = 100
7           elif b_max_depth > m_max_depth:
8               score = m_max_depth / b_max_depth * 100
9           else:
10              b_max_depth < m_max_depth:
11              score = b_max_depth / m_max_depth * 100
12      return score
```

圖 5-14 最大結構深度方法

$Q_{2,2,2}$「最大樹寬」為 XML 文件最大之同階層元素數，故需先計算各階層元素總數，再取出最大元素數進行程度計算，計算方法以比對基準文件與量測文件最大樹寬，以樹寬大者為分母、樹寬小者為分子，分數 0 至 100 計算，完全相同為 100。

演算法為：偵測基準文件 b_XML 與量測文件 m_XML 之各階層的元素數量，將 b_wc 及 m_wc 字典格式中所顯示的同階層最大元素數量，建立 Python 的字典資料格式，內容為{ 'Depth':深度數值, 'Count':元素數量}，分別將計算結果回傳至 b_wc 及 m_wc 變數（第 5 至 8 行）；將 b_wc 及 m_wc 取出最大樹寬 b_max_width 及 m_max_width（第 9 至 10 行）進行樹寬比對，真實程度依前述計分方式，回傳相似程度。（第 11 至 18 行）

```
1   # 最大結構寬度方法
2   def xml_max_width(b_XML, m_XML):
3       b_max_width = b_xml 最大寬度
4       m_max_width = m_xml 最大寬度
5       b_wc = { 'Depth' : bdsg.argmax() +1,
6                'Count' : bdsg.max() }
7       m_wc = { 'Depth' : mdsg.argmax() +1,
8                'Count' : mdsg.max() }
9       b_max_width = b_wc['Count']
10      m_max_width = m_wc['Count']
11      if b_max_width == m_max_width:
12          score = 100
13      elif b_max_width > m_max_width:
14          score = m_max_width / b_max_width * 100
15      else:
16          b_max_width < m_max_width:
17          score = b_max_width / m_max_width * 100
18      return score
```

圖 5-15 最大結構寬度方法

$Q_{2,2,3}$「元素數量」為 XML 文件元素總數，計算方法以比對基準文件與量測文件元素總數，以元素總數大者為分母、元素總數小者為分子，分數以 0 至 100 計算，完全相同為 100。

演算法為：計算基準文件 b_XML 與量測文件 m_XML 之元素總數，將計算數值存於變數 b_total_tags 及 m_total_tags（第 3 至 4 行）；依前述計算方法比對文件最大深度，回傳相似程度。（第 5 至 12 行）

```
1   # 元素總數量方法
2   def xml_total_tags(b_XML, m_XML):
3       b_total_tags = b_xml 元素總數
4       m_total_tags = m_xml 元素總數
5       if b_total_tags == m_total_tags:
6           score = 100
7       elif b_total_tags > m_ total_tags:
8           score = m_total_tags / b_total_tags * 100
9       else:
10          b_ total_tags < m_ total_tags:
11          score = b_total_tags / m_total_tags * 100
12      return score
```

圖 5-16 元素總數量方法

值得一提的是，鑑於真實世界中許多 XML 文件會將不同地理位置、時區、歷史資訊附加於同份文件中，以致即使是相同的 XML，紀錄的資料筆數、元素總數、樹寬等結構資訊不一致，故本研究建議使用者在資料結構判別真實情形時，運用 DataGuides 方法可刪除重複的資訊，將有助於釐清及定義真實程度。

5.2.3 真實維度-文件內容 D3

在真實維度-文件內容 D3 中，依真實模型設計真實屬性為 $P_{3,1}$「元素」及 $P_{3,2}$「屬性」及 $P_{3,3}$「路徑」。

$P_{3,1}$「元素」之真實屬性建立，係針對 XML 文件中指定的元素存在與否及存在數量進行程度判別，舉例「元素名稱」之存在與否為 $Q_{3,1,1}$，指定元素的「元素數量」是否相似為 $Q_{3,1,2}$ 。

$Q_{3,1,1}$「元素名稱」之演算法係於量測文件之中，搜尋指定基準文件中的元素名稱存在與否，有則分數為 100，無則為 0。

演算法為：比對指定元素 key_tag 是否在於基準文件 b_XML 及量測文件 m_XML（第 3 行），如果 key_tag 沒有在 b_XML 中，表示使用者所輸入的指定元素名稱不存在，則由 b_XML 之所有元素與 m_XML 所有元素進行比對，如有重覆元素存於變數 c_tag，並顯示 b_XML 沒有您所輸入的 key_tag，採用兩份文件相同元素進行比對，分數為 c_tag 數量/b_XML 元素數量*100（第 4 至 6 行）；如 b_XML 有 key_tag，則透過 key_tag 是否存在於 m_XML 進行比對，如無則分數為 0，如有則分數為 100，並回傳相似程度。（第 8 至 12 行）

```
1   #指定元素名稱
2   def xml_key_tag(b_XML, m_XML, key_tag):
3       if b_xml.find(key_tag) is None:
4           c_tag =b_XML 及 m_XML 重覆之元素
5           print ('b_XML 沒有您所輸入的 key_tag，
                採用兩份文件相同元素進行比對)
6           score = c_tag 數量/b_XML 元素數量  *100
7       else:
8           if m_xml.find(key_tag) is None:
9               score = 0
10          else:
11              score = 100
12      return score
```

圖 5-17 指定元素名稱方法

$Q_{3,1,2}$「元素數量」之分數設計：比對基準及量測文件中的指定元素數量，有則分數為 100，無則為 0，計算方法以比對基準文件與量測文件指定元素數量，以指定元素數量多者為分母、指定元素數量少者為分子，分數以 0 至 100 計算，完全相同為 100。

演算法為：比對指定元素 key_tag 是否在於基準文件 b_XML 之中，透過 b_xml.find(key_tag)可以取得文件中 key_tag 所屬的元素、屬性及內容，如無則參照圖 5-17 指定元素名稱方法，由 b_XML 之所有元素與 m_XML 所有元素進行比對，如有重覆元素存於變數 c_tag，並顯示 b_XML 沒有您所輸入的 key_tag，採用兩份文件相同元素進行比對，分數為 c_tag 數量/b_XML 元素數量*100（第 3 至 7 行）；如果 b_XML 存在 key_tag 元素，則偵測基準文件 b_XML 與量測文件 m_XML 之指定元素數量，該總數為大於等於零之整數，將值存於 b_key_tag_count 及 m_key_tag_count 變數中（第 8 至 9 行）；依照前述分數設計，比對 b_key_tag_count 及 m_key_tag_count 元素總數，元素總數多者則為分母、元素總數較少者為分子，再兩者相除承以 100，回傳相似程度。（第 10 至 17 行）。

```
1   # 指定元素數量方法
2   def xml_key_tag_count(b_XML, m_XML, key_tag):
3       if b_xml.find(key_tag) is None:
4           c_tag =b_XML 及 m_XML 重覆之元素
5           print ('b_XML 沒有您所輸入的 key_tag，
6                   採用兩份文件相同元素進行比對)
7           score = c_tag 數量/b_XML 元素數量  *100
        else:
8           b_key_tag_count = b_xml 的 key_tag 數量
9           m_key_tag_count = m_xml 的 key_tag 數量
10          if b_key_tag_count == m_key_tag_count:
11              score = 100
12          elif b_key_tag_count > m_key_tag_count:
13              score = m_key_tag_count / b_key_tag_count * 100
14          else:
15              b_key_tag_count < m_key_tag_count:
16              score = b_key_tag_count / m_key_tag_count * 100
17      return score
```

圖 5-18 指定元素數量方法

$Q_{3.1.3}$「同義元素」之分數設計：建立基準文件之元素作為基本詞庫，基本詞庫容許使用者增加字詞，透過基準詞庫與量測文件之元素進行模糊比對，相似字詞數量佔量測文件元素數量比例，分數以 0 至 100 計算，完全相同為 100。

演算法為：先將基準文件 b_XML 及 m_XML 以圖 5-12 描述之 DataGuides 方法產生元素列表 b_tag_list 及 m_tag_list（第 3 至 4 行）；建立基本詞庫 basic_dict，該詞庫係由 b_tag_list 加上使用者呼叫本方法時傳入的自訂字詞串列 words 參數，舉例如：自訂增列具有詞義參考價值之串列[‘Alice’,’Bob’,’Tom’,]等，以增加使用者運用詞庫進行同義元素模糊比對之彈性（第 5 行）；接續建立模糊比對篩選器 regex，以正規表達式將字詞各字元之間插入'.*?'，舉例如字詞 tmp 變為 t.*?m.*?p.*?，表示具有 tmp 等 3 字組成的字串可以透過模糊比對搜尋符合結果（第 6 行）；將 m_tag_list 內符合 regex 篩選不重複元素串列存入變數 result（第 7 行）；依前述分數設計，result 數量及 m_XML 以 DataGuidesz 方法產生之元素數量兩者相除乘以 100，回傳相似程度。（第 8 至 9 行）。

```
1  # 指定元素數量方法
2  def xml_fuzzyfinder(b_XML, m_XML, words, *args):
3      b_tag_list = b_XML 以 dataguides 方法產生之元素串列
4      m_tag_list = m_XML 以 dataguides 方法產生之元素串列
5      basic_dict = b_tag_list + words
6      regex =  建立將字元插入'.*?之模糊比對篩選器
7      result = m_tag_list 內符合 regex 篩選不重複元素串列
8      score = len(result) / len(m_tag_list) * 100
9      return score
```

圖 5-19 同義元素模糊比對方法

$P_{3,2}$ 「屬性」係針對 XML 文件中指定的屬性存在與否及存在數量進行程度判別，舉例「元素屬性」之存在與否為 $Q_{3,2,1}$，指定元素的「屬性數量」是否相似為 $Q_{3,2,2}$ 。

$Q_{3,2,1}$「屬性名稱」之分數設計：於量測文件中，搜尋指定基準文件中的屬性名稱存在與否，如基準文件沒有屬性，則量測文件有無屬性分別為 20 及 30；如基準文件有屬性，而未有輸入的指定屬性時，改採用基準文件與量測文件相同之屬性數量計算分數，完全相同分數為 100。

演算法為：建立 B_XML 及 m_XML 屬性之串列 b_attr 及 m_attr（第 3 至 4 行），如果 b_attr 為空值，則 m_attr 是否為空值分別為 20 及 30（第 6 至 9 行）；如 b_attr 非空值，指定屬性名稱 key_attr 是否在於基準文件 b_xml 中，如果 key_attr 沒有在 b_XML 中，表示使用者所輸入的指定元素名稱不存在，由 b_XML 之所有屬性與 m_XML 所有屬性進行比對，如有重覆署性存於變數 c_attr，並顯示 b_XML 沒有您所輸入的 key_attr，採用兩份文件相同屬性進行比對，分數為 c_attr 數量/b_XML 屬性數量*100（第 10 至 15 行）；若 key_attr 存在於 b_XML 文件中，而且亦存在於 m_XML 中， 有則為 100，無則分數為 0，並回傳相似程度。（第 16 至 21 行）

```
1   #指定屬性名稱方法
2   def xml_key_attr(b_XML, m_XML, key_attr):
3       b_attr = b_XML 之屬性串列
4       m_attr = m_XML 之屬性串列
5       if b_attr is None:
6           if m_attr is None:
7               score = 30
8           else:
9               score = 20
10      else:
11          if b_xml.find(key_attr) is None:
12              c_attr =b_XML 及 m_XML 重覆之屬性
13              print ('b_XML 沒有您所輸入的 key_attr，
14                      採用兩份文件相同屬性進行比對)
15              score = c_attr 數量/b_XML 屬性數量  *100
16          else:
17              if m_xml.find(key_attr) is None:
18                  score = 0
19              else:
20                  score = 100
21      return score
```

圖 5-20 指定屬性名稱方法

$Q_{3,1,2}$「屬性數量」之演算法為：於量測文件中，搜尋指定基準文件中的屬性名稱之數量相似程度，計算方法以比對基準文件與量測文件關鍵屬性總數，以屬性總數大者為分母、屬性總數小者為分子，分數以 0 至 100 計算，完全相同為 100。

演算法為：建立 B_XML 及 m_XML 屬性之串列 b_attr 及 m_attr，如果 b_attr 為空值，則 m_attr 是否為空值分別為 20 及 30（第 3 至 9 行）；如果 b_attr 非空值，比對指定屬性 key_attr 是否在於基準文件 b_XML 之中，如果 key_attr 沒有在 b_XML 中，由 b_attr 與 m_attr 所有屬性進行比對，如有重覆屬性存於變數 c_attr，並顯示 b_XML 沒有您所輸入的 key_attr，分數為 c_attr 數量/b_XML 屬性數量*100（第 10 至 15 行）；如果 b_XML 存在 key_attr 屬性，則偵測基準文件 b_XML 與量測文件 m_XML 之指定屬性數量，將值存於 b_key_attr_count 及 m_key_attr_count 變數中比對屬性總數，屬性總數多者則為分母、屬性總數較少者為分子，再兩者相除乘以 100，回傳相似程度。（第 16 至 24 行）。

```
1   # 指定屬性數量
2   def xml_total_attrs(b_XML, m_XML, key_attr):
3       b_attr = b_XML 之屬性串列
4       m_attr = m_XML 之屬性串列
5       if b_attr is None:
6           if m_attr is None:
7               score = 30
8           else:
9               score = 20
10      else:
11          if b_xml.find(key_attr) is None:
12              c_attr =b_XML 及 m_XML 重覆之屬性
13              print ('b_XML 沒有您所輸入的 key_attr，
14                      採用兩份文件相同屬性進行比對)
15              score = c_attr 數量/b_XML 屬性數量  *100
16      else:
17          b_key_attr_count = b_xml 的 key_attr 數量
18          m_key_attr_count = m_xml 的 key_attr 數量
19          if b_key_attr_count >= m_key_attr_count:
20              score = m_key_attr_count / b_key_attr_count * 100
21          else:
22              b_key_attr_count < m_key_attr_count:
23              score = b_key_attr_count / m_key_attr _count *100
24      return score
```

圖 5-21 指定屬性數量方法

$P_{3,3}$ 「路徑」係指 XML 文件可用 XPath[36]搜尋引用文檔中的節點，搜尋 XML 文件中元素、屬性、文本、命名空間、處理指令及註釋，透過 Xpath 路徑表達式可以搜尋指定路徑之元素。在計算 XML 真實模型程度時，可運用 XPath 特性進行指定路徑的判斷。舉例「指定路徑」之存在與否為 $Q_{3,3,1}$，指定路徑的「路徑數量」相似程度為 $Q_{3,3,2}$ 。

$Q_{3,3,1}$「指定路徑」之分數設計：指定基準文件具有路徑關聯的 2 個元素，搜尋量測文件中指定的路徑存在與否，有則分數為 100，無則為 0。

其演算法為：指定元素 tag_perent, tag_children 可形成指定路徑，tag_perent 係 tag_children 的父節點，首先判別 tag_perent、tag_childre 等 2 變數是否為基本文件 b_XML 之元素，如無則由 b_XML 之所有路徑與 m_XML 所有路徑進行比對，如有重覆路徑存於變數 c_path，並顯示 b_XML 沒有您所輸入的路徑，採用兩份文件相同路徑進行比對，分數為 c_path 數量/b_XML 路徑數量*100（第 3 至 7 行）；將 b_xml 透過 tag_perent 與 tag_children 形成的路徑，以 b_xml. find(tag_perent).find(tag_childre)取出解析結果存入於變數 b_path，另將量測文件 m_XML 解析路徑結果存入 m_path 變數（第 9 至 10 行）；判別 b_path 如果其值為 None，則分數為 0、m_path 如果其值為 None，則分數為 0，如路徑存在於 b_XML 及 m_XML，分數為 100，回傳相似程度。（第 11 至 17 行）

```
1   # 指定路徑方法
2   def xml_path(b_XML, m_XML , tag_perent, tag_childre):
3     if tag_perent not in b_xml or tag_childre not in b_xml:
4       c_path =b_XML 及 m_XML 重覆之路徑
5       print ('b_XML 沒有您所輸入的路徑，
6               採用兩份文件相同路徑進行比對)
7       score = c_path 數量/b_XML 路徑數量  *100
8     else:
9       b_path = tag_perent 及 tag_children 於 b_XML 路徑
10      m_path = tag_perent 及 tag_children 於 m_XML 路徑
11      if b_path is None:
12        score = 0
13      elif m_path is None:
14         score = 0
15      else:
16        score = 100
17    return score
```

圖 5-22 指定路徑方法

$Q_{3,3,2}$「路徑數量」計分方式為：XPath 指定路徑，分別在基準文件與量測文件之中所具有的路徑數量相似程度，計算方法以比對基準文件與量測文件指定路徑總數，以路徑總數大者為分母、路徑總數小者為分子，分數以 0 至 100 計算，完全相同為 100。

演算法為：指定元素 tag_perent, tag_children 可形成指定路徑，tag_perent 係 tag_children 的父節點，首先判別 tag_perent、tag_childre 等 2 變數是否為基本文件 b_XML 之元素，如無則由 b_XML 之所有路徑與 m_XML 所有路徑進行比對，如有重覆路徑存於變數 c_path，並顯示 b_XML 沒有您所輸入的路徑，採用兩份文件相同路徑進行比對，分數為 c_path 數量/b_XML 路徑數量*100（第 3 至 7 行）；將 b_xml 透過 tag_perent 與 tag_children 形成的路徑，以 b_xml.find(tag_perent).find(tag_childre)取出解析結果存入於變數 b_path，另將量測文件 m_XML 解析路徑結果存入 m_path 變數（第 9 至 10 行）；分別計算 b_path 及 m_path 數量存入變數 b_path_count、m_path_count（第 11 至 12 行）。依據前述計分方式，比較 b_path_count 及 m_path_count，分數計算以數值大者為分母、數值較小者為分子，兩者相除乘以 100，回傳真實程度。（第 13 至 20 行）

```
1   # 指定路徑數量
2   def xml_path_count(b_XML, m_XML ,
                       tag_perent, tag_childre):
3     if tag_perent not in b_xml or tag_childre not in b_xml:
4       c_path =b_XML 及 m_XML 重覆之路徑
5       print ('b_XML 沒有您所輸入的路徑，
6             採用兩份文件相同路徑進行比對)
7       score = c_path 數量/b_XML 路徑數量  *100
8     else:
9       b_path = tag_perent 及 tag_children 於 b_XML 路徑
10      m_path = tag_perent 及 tag_children 於 m_XML 路徑
11      b_path_count = b_path 數量
12      m_path_count = m_path 數量
13      if b_path_count == m_path_count:
14        score = 100
15      elif b_path_count > m_path_count:
16        score = m_total_tags / b_path_count * 100
17      else:
18        b_path_count < m_path_count:
19        score = b_path_count / m_path_count * 100
20    return score
```

圖 5-23 指定數量方法

5.2.4 真實維度-文件時間 D4

在真實維度-文件時間 D4 中，依真實模型設計真實屬性為 $P_{4,1}$「檔案時間」及 $P_{4,2}$「更新頻率」。

$P_{4,1}$「檔案時間」真實屬性中，$Q_{4,1,1}$「建立時間」可作為評估量化因子。

$Q_{4,1,1}$「建立時間」計算方式為：使用者可以針對基準文件對於時間特性的要求設計合理建立時間區間，如：多少分、秒、時、日、年、月等，依據合理建立時間區間進行真實程度估算，舉例如量測文件與基準文件差異超過 500 日則不真實，或差異 500 秒為不真實。量測文件與真實程度建立區間之單位相同者為 100，等於及超過合理區間則分數為 0。

演算法為：timedelta 為使用者所設定的時間區間（第 3 行），將基準文件 b_xml 建立時間存入變數 b_time、量測文件 m_xml 建立時間存入變數 m_time（第 4 至 5 行）；將 b_time - m_time 取絕對值的計算結果存於變數 c_time，c_time 之差異天數存入 c_day，即基準文件與量測文件的建立時間區間與天數（第 6 至 7 行）；如果使用者未輸入 timedelta，則以(365 - c_dat) / 365 * 100 表示基準文件與量測文件在差異日期計算真實程度，文件日期越相近則分數越高（第 8 至 9 行）；如有輸入 timedelta，比對 c_time 與 timedelta 區間相似情形計算真實程度，如果 c_time - timedelta > 0 即表示實際建立時間區間大於使用者設定的合理區間，真實分數為 0（第 7 至 8 行），否則分數依據前述計算方式，回傳真實程度。（第 10 至 11 行）

```
1   # 建立時間方法
2   def xml_time_delta (b_XML, m_XML , timedelta ):
3       timedelta=  合理時間區間
4       b_time = b_xml 建立時間
5       m_time = m_xml 建立時間
6       c_time = abs(b_time - m_time)
7       c_day = c_time 之差異天數)
8       if timedelta is None:
9           score = (365 - c_dat) / 365 * 100
10      if c_time – timedelta > 0:
11          socre = 0
12      else:
13          score = (timedelta - c_time) / timedelta * 100
14      return score
```

圖 5-24 建立時間方法

$P_{4,2}$ 更新頻率真實屬性以 $Q_{4,2,1}$「更新頻率」進行真實程度估算。

$Q_{4,2,1}$「更新頻率」係指依據基準文件的來源更新頻率，以及比對量測文件來源的更新頻率兩者進行真實程度判別，舉例如基準文件更新頻率為 15 分鐘，而量測文件的更新頻率為 12 小時，則應有真實程度差異。

計算方式以基準文件更新頻率與容許範圍（權重值）之乘積，比較文件更新頻率，差異越小越相似，完全相同分數為 100，否則趨近於 0。

演算法為：鑒於實際更新頻率應以複數以上的建立時間平均值計算，即基準文件及量測文件應為同來源、不同時間所產生的檔案，使用者需先下載於本機或伺服器資料庫，本研究採用文件分別儲存於不同資料夾，由使用者提供基準文件及量測文件所在的資料夾 b_folder 及 m_folder，分別產生同資料來源資料夾檔案列表 b_folder_list 及 m_folder_list（第 3 至 4 行）；分別計算 b_folder_list 及 m_folder_list 更新頻率，以資料夾內檔案建立平均間隔時間計算,計算結果分別存入 b_frequency 及 m_frequency(第 5 至 6 行）；將基準文件更新頻率 b_frequency 乘以權重 weight，以表示基於基準文件可接受合理頻率範圍，計算結果存入變數 b_frequency_w（第 7 行）；將 b_frequency_w 與量測文件的更新頻率 m_frequency 進行比較，依前述計算方式，回傳真實程度。（第 8 至 14 行）

```
1   #更新頻率方法
2   def xml_time_ frequency(b_folder, m_folder , weight ):
3       b_folder_list =  同資料來源資料夾檔案列表
4       m_folder_list =  同資料來源資料夾檔案列表
5       b_frequency = b_XML_folder 內資料平均間隔時間
6       m_frequency = m_XML_folder 內資料平均間隔時間
7       b_frequency_w = b_frequency * weight
8       if b_frequency = m_frequency:
9           score = 100
10      elif b_frequency_w > m_frequency:
11          score = m_frequency / b_frequency_w * 100
12      else:
13          socre = b_frequency_w / m_frequency * 100
14      return score
```

圖 5-25 更新頻率方法

5.2.5 綜合計算 XML 真實模型分數

綜合前述各項真實維度、真實屬性及量化因子計算真實分數，使用者可透過對資料的理解性分別設定不同的權重，最終取得基於基準文件的量測文件 XML 模型真實程度分數，並可透過資料視覺化進行觀察。本研究採用 Python 可使用的 Jupyter Notebook 直譯器，以 iplotter.ChartJSPlotter 模組視覺化實作，圖 5-26 為該模組繪製之 XML 真實程度模型雷達圖的程式碼示意，圖 5-27 為繪圖結果。

```
# 建立雷達示意圖方法
from iplotter import ChartJSPlotter
plotter = ChartJSPlotter()
data = {
        "labels": [各維度名稱],
        "datasets": [
          {"label": "量測文件 M1",
            "borderColor": "rgba(179,181,198,1)",
            "data": [M1 各維度分數] },
          {"label": "量測文件 M2",
            "borderColor": "rgba( 0,0,255,1)",
            "data": [M2 各維度分數] },
          {"label": "量測文件 M3",
            "borderColor": "rgba(255,99,132,1)",
            "data": [M3 各維度分數]}]}
plotter.plot_and_save(data, 'radar', options=None)
```

圖 5-26 建立雷達示意圖方法

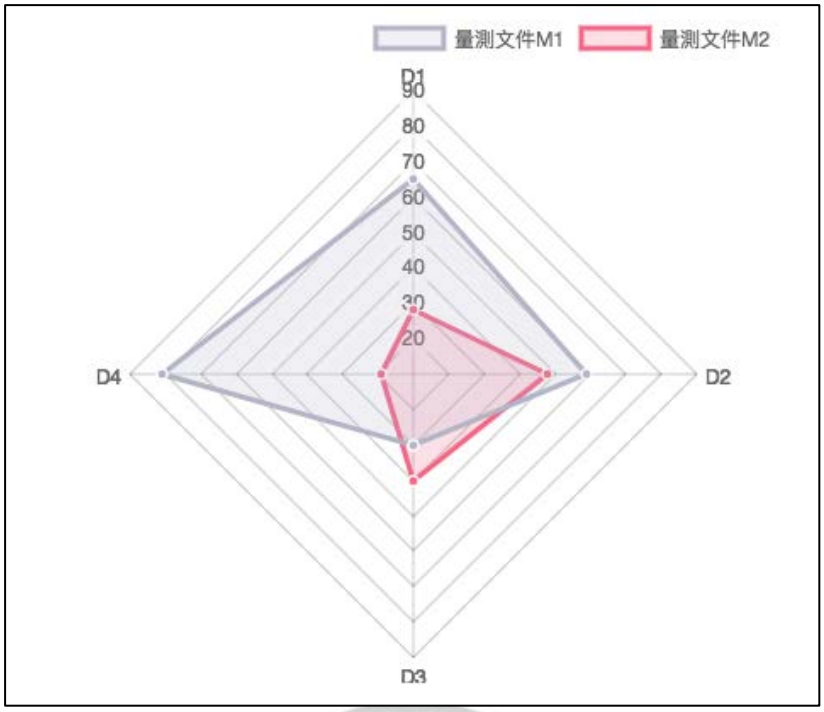

圖 5-27 真實程度模型雷達示意圖

第六章 模型應用

應用「巨量 XML 真實程度模型」以取得量測文件 XML 程度，使用者需依據需求建立真實模型，建立方向包含(1) 識別真實程度的動機與目標；(2)決定真實維度；(3)計算分數的方法；(4)程度計分採用的尺度。

本章模型應用以臺灣自動氣象站-氣象觀測資料[41]做為基準資料，依據已知基準文件特性，評價已知與基準文件不同的複數量測文件真實程度；評價目的係研究真實程度對已知不同文件的真實程度。

透過本論文提出的真實模型，計算 XML 文件的真實程度，提供未來使用該模型判別未知量測文件的鑑別能力參考。使用之範例文件及應用結果說明如下：

6.1 範例文件

本範例以臺灣氣象資訊[41]作為基準文件 B，另一方面，以美國國家海洋暨大氣總署（NOAA）氣象資料[23]、OpenWeatherMap.org 各地目前氣候資訊[7]、台灣政府開放交通資料[37]分別做為量測文件 M1、M2 與 M3，進行真實程度計算，以說明本論文建構之以資料理解性觀點建立的真實模型與其真實程度。

基準文件 B 為「臺灣自動氣象站-氣象觀測資料」，該資料為臺灣資料開放平台[42]熱門引用數據。文件 B 每 10 分鐘更新一次，元素包含測站 ID、觀測時間、溫度、濕度、風向及風速、觀測時間前推 15 分鐘內發生最大風速等資訊，該文件結構計有 9,471 行、40 個欄位，文件最深 5 層，係一般常見的 XML 資料設計方式，表 6-1 為自動氣象站資料元素名稱說明文件[43]、圖 6-1 為該 XML 資料文件內容摘要、圖 6-2 為該 XML 樹狀內容。

表 6-1 自動氣象站 XML 元素說明表

元素名稱	中文說明
stationId	觀測站 ID
STNM	觀測站編號
OBS_TIME	觀測資料時間
TIME	未使用

元素名稱	中文說明
LAT	緯度 (座標系統採 TWD67)
LON	經度 (座標系統採 TWD67)
ELEV	高度，單位：公尺
WDIR	風向，單位：度，風向 0 表示無風
WDSD	風速，單位：公尺/秒位 攝氏
TEMP	溫度，單位：攝氏
HUMD	相對濕度，單位 百分比率，此處以實數 0-1.0 記錄
PRES	觀測站氣壓，單位：百帕
H_24R	日累積雨量，單位：毫米
WS15M	觀測時間前推 15 分鐘內發生最大風速，單位：公尺/
WD15M	觀測時間前推 15 分鐘內發生最大風風速，單位：度
WS15T	觀測時間前推 15 分鐘內發生最大風的發生時間，hhmm (小時分鐘)
CITY	縣市
CITY_SN	縣市編號
TOWN	鄉鎮
TOWN_SN	鄉鎮編號

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <cwbopendata xmlns="urn:cwb:gov:tw:cwbcommon:0.1">
3      <identifier>ca5efb48-b33a-4ea9-ac83-48c3b5524179</identifier>
4        <sender>weather@cwb.gov.tw</sender>
5        <sent>2016-11-12T09:35:52+08:00</sent>
6        <status>Actual</status>
7        <msgType>Issue</msgType>
8        <dataid>CWB_A0001</dataid>
9        <scope>Public</scope>
10       ....
      </cwbopendata>
31133
```

圖 6-1 氣象站 XML 內容摘要

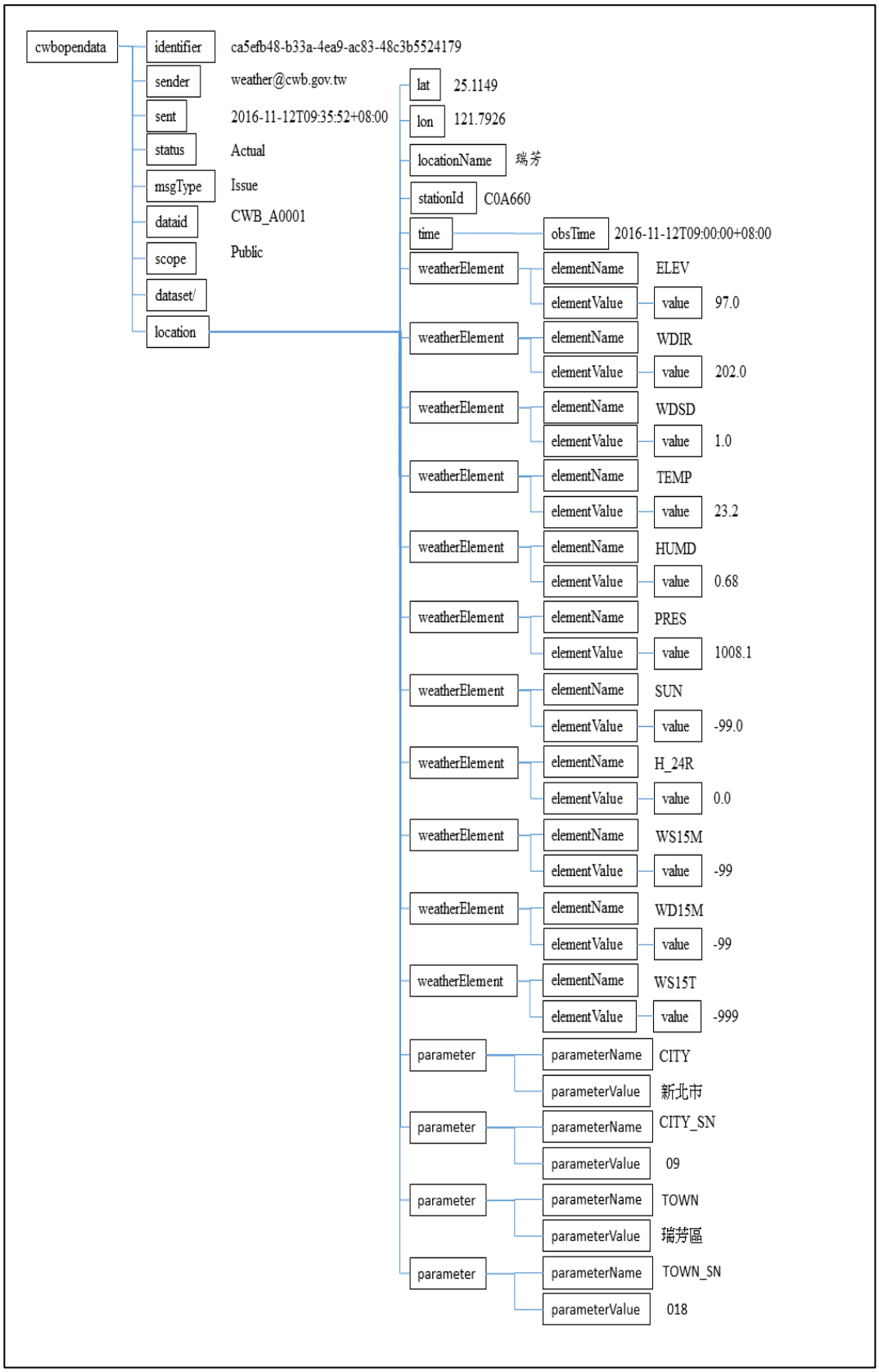

圖 6-2、自動氣象站 XML 樹狀內容

測量文件 M1 作採用美國國家海洋暨大氣總署（NOAA）氣後觀測資訊[23]。NOAA 為了為預測環境變化、保護與管理海洋資源，具有專業氣象預測資訊，提供氣候監測、預先警示數據。在目前天氣狀況 XML 資料中，提供美國 1,800 個地點的氣候資訊，並提供 XML 及 RSS 以幫助訊息的發布，在同一時間點下載 NOAA 分享的 XML 文件壓縮檔，解壓縮後 XML 文件計有 4,672 份，由於每份文件結構相同，將以其中一份 XML 文件資訊進行真實程度估算。表 6-2 為 NOAA 氣後觀測資料元素名稱說明文件[5]，圖 6-3 為該 XML 資料文件內容摘要、圖 6-4 為其 XML 樹狀內容。

表 6-2 NOAA 氣候觀測 XML 元素說明表

元素名稱	說明
Credit	固定宣告為 NOAA's National Weather Service
credit_URL	固定宣告為 http://weather.gov/
url	固定宣告為 http://weather.gov/images/xml_logo.gif
Title	固定宣告為 NOAA's National Weather Service
Link	固定宣告為 http://weather.gov
suggested_pickup	固定宣告為 15 minutes after the hour
suggested_pickup_period	固定宣告為 60
Location	觀測站地點
station_id	觀測站 ID
Latitude	緯度
longitude	經度
observation_time	最近觀測時間，如：Last Updated on Dec 26 2016, 12:00 am AST
observation_time_rfc822	最近觀測時間，如：Mon, 26 Dec 2016 00:00:00 -0400
temperature_string	溫度
temp_f	華氏溫度，如：21.0
temp_c	攝氏溫度，如：-5.9
wind_string	風速及風向資訊字串，如：from the North at 18.4 gusting to 27.6 MPH (16 gusting to 24 KT)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="latest_ob.xsl" type="text/xsl"?>
<current_observation version="1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.weather.gov/view/c
  urrent_observation.xsd">
     <credit>NOAA's National Weather Service</credit>
     <credit_URL>http://weather.gov/</credit_URL>
     <image>
        <url>http://weather.gov/images/xml_logo.gif</url>
        <title>NOAA's National Weather Service</title>
        <link>http://weather.gov</link>
     </image>
     <suggested_pickup>15 minutes after the hour
     </suggested_pickup>
     <suggested_pickup_period>60</suggested_pickup_period>
     <location>Unknown Station</location>
     <station_id>42S39</station_id>
     <observation_time>Last Updated on Dec 22 2016, 6:30 am AST
     </observation_time>
        <observation_time_rfc822>Thu, 22 Dec 2016 06:30:00 -0400
        </observation_time_rfc822>
     <wind_string>East at 2.2 MPH (1.94 KT)</wind_string>
     <wind_dir>East</wind_dir>
     <wind_degrees>110</wind_degrees>
     <wind_mph>2.2</wind_mph>
           …
  </current_observation>
```

圖 6-3 NOAA 氣候觀測 XML 內容摘要

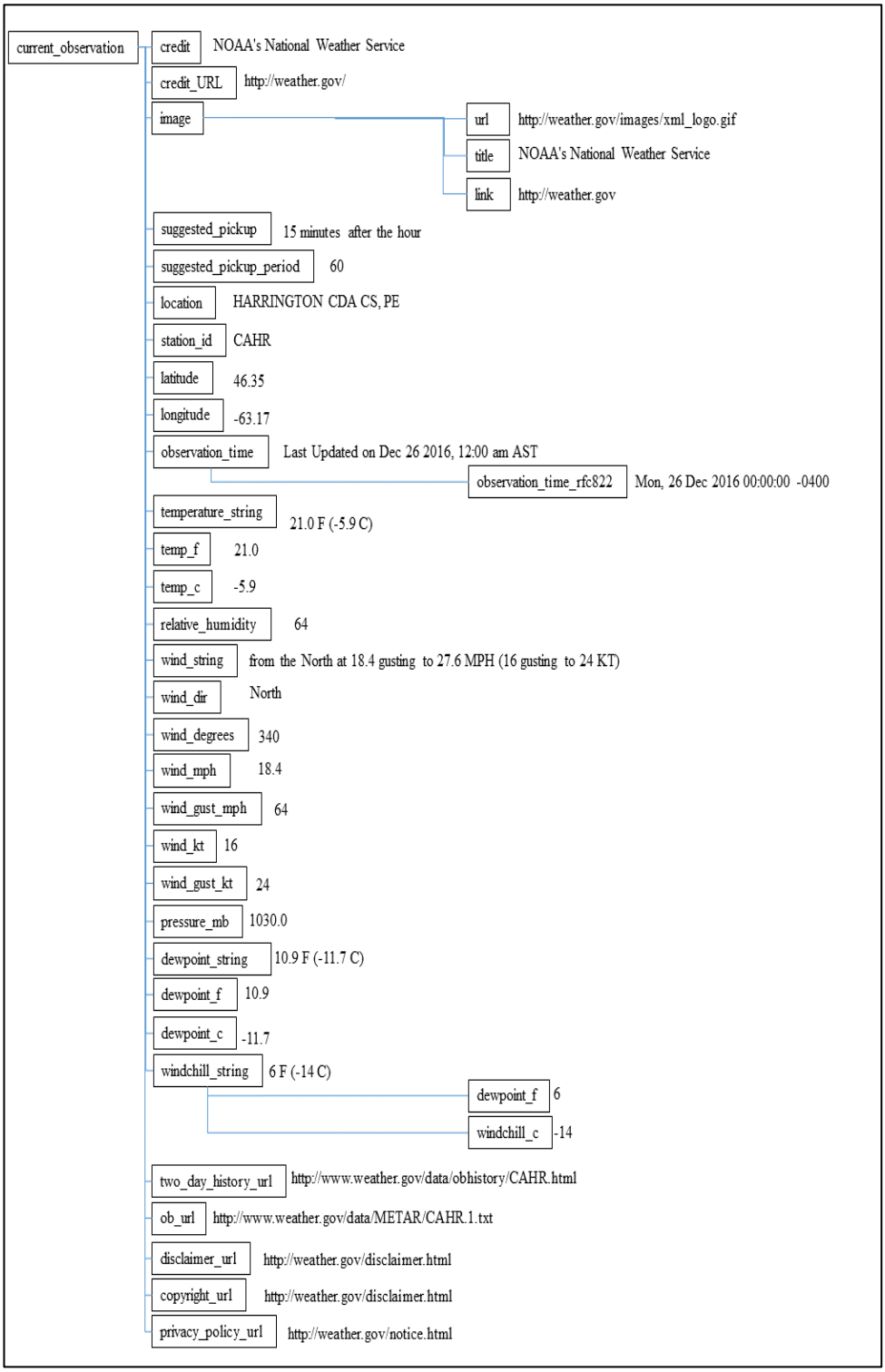

圖 6-4 NOAA 氣候觀測 XML 樹狀內容

測量文件 M2 為位於美國的 OpenWeatherMap 組織所提供的各地「目前氣候資訊 API」[7]，該資料可透過各地城市名稱、城市 ID、經緯度、ZIP CODE 等方式進行呼叫，回傳的結果格式包含 XML、JSON 或 HTML，帳號分為免費及各種付費方式，以免費帳號而言，每地區之資訊可於 10 分鐘呼叫 1 次、一個帳號至多 1 分鐘內呼叫 60 次各地 API 資訊，本研究以呼叫程式名稱台中（TaiChung）所回傳的 XML 資訊進行分析。表 6-3 為資料元素名稱說明、圖 6-5 為內容摘要、圖 6-6 為 XML 樹狀內容。

表 6-3 OpenWeatherMap 即時氣象資訊 XML 元素說明表

元素名稱	說明
city.id	城市 ID
city.name	城市名稱
city.coord.lon	城市經度
city.coord.lat	城市緯度
city.country	國家編碼如(GB, JP etc.)
city.sun.rise	日出時間
city.sun.set	日落時間
temperature.value	溫度
temperature.min	該時間點的區域最低溫度
temperature.max	該時間點的區域最低溫度
temperature.unit	測量單位，可能是攝氏、華氏, Kelvin
humidity.value	濕度值
humidity.unit	單位%
pressure.value	大氣壓力值
pressure.unit	大氣壓力單位 hPa
wind.speed.value	風速值, mps
wind.speed.name	風速類型
wind.direction.value	風向，度（氣象）
wind.direction.code	風向代碼。可能值為 WSW，N，S 等
wind.direction.name	風向全名
clouds.value	雲量
clouds.name	雲霧的名字
visibility.value	可見度，米
recipitation.value	降雨量, mm
precipitation.mode	可能的值是 不，天氣現象名稱為雨，雪

weather.number	天氣狀況 id
weather.value	天氣條件名稱
Weather icon	天氣圖標 ID
lastupdate.value	上次更新數據時

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <current>
3     <city id="1668399" name="Taichung">
4        <coord lon="120.68" lat="24.15"></coord>
5        <country>TW</country>
6        <sun rise="2017-07-19T21:20:58" set="2017-07-
   20T10:46:10"></sun>
7     </city>
8     <temperature value="300.15" min="300.15" max="300.15"
   unit="kelvin"></temperature>
9     <humidity value="83" unit="%"></humidity>
10    <pressure value="1011" unit="hPa"></pressure>
11    <wind>
12       <speed value="1.24" name="Calm"></speed>
13       <gusts></gusts>
14       <direction value="12.5026" code="NNE" name="North-
   northeast"></direction>
15    </wind>
16    <clouds value="75" name="broken clouds"></clouds>
17    <visibility value="10000"></visibility>
18    <precipitation mode="no"></precipitation>
19    <weather number="803" value="broken clouds"
   icon="04n"></weather>
20    <lastupdate value="2017-07-20T13:30:00"></lastupdate>
21 </current>
```

圖 6-5 OpenWeatherMap 即時氣象資訊 XML 內容摘要

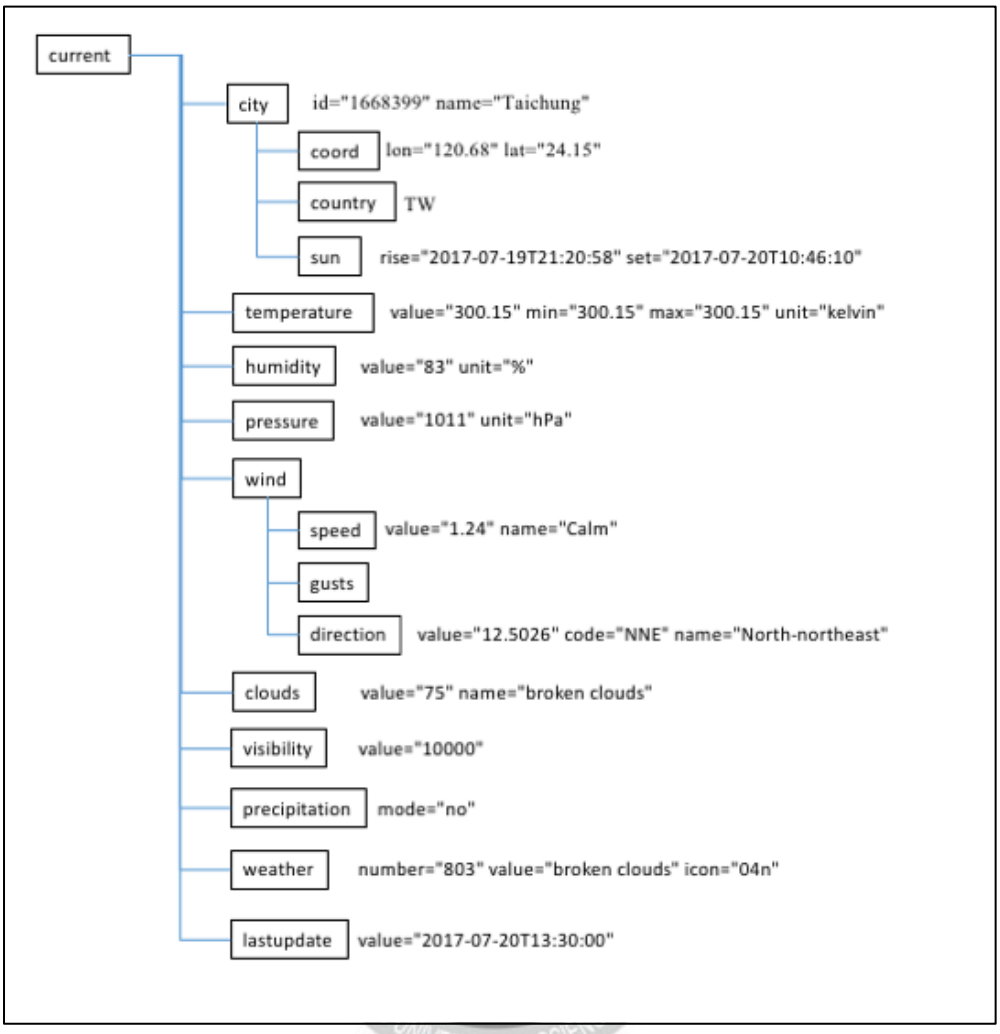

圖 6-6 OpenWeatherMap 即時氣象資訊 XML 樹範例

測量文件 M3 為交通部開放資料「VD 一分鐘動態資訊」[37]，該資料庫資料來源為「高快速公路整體路網交通管理系統」與「高速公路電子收費系統」，資料格式係依據交通部發布之「路側設施即時交通資訊發布標準格式」及「國道高速公路電子收費交通資料蒐集支援系統（Traffic Data Collection System，TDCS）使用手冊[38]。提供 VD 狀態及偵測值的標準格式，每分鐘提供各車道不同車種的速率、流量及佔有率，並標示 VD 運作狀況。表 6-4 為資料元素名稱說明、圖 6-7 為內容摘要、圖 6-8 為 XML 樹狀內容。

表 6-4 交通部 VD 動態資訊 XML 元素說明表

元素名稱	說明
vdid	設備代碼
status	狀態
vsrdir	車道方向，以 0 和 1 表示
vsrid	車道代碼
carid	車種代碼
speed	依車道逐一詳列 1 分鐘平均速率偵測值(單位:kph)，如：24。
volumn	依車道/車種逐一詳列 1 分鐘流量偵測值，如：66。
Lane-occupy	依車道逐一詳列 1 分鐘佔有率偵測值(單位:%)，如:50。
Data-collecttime	資料蒐集時間

```
1     <?xml version="1.0" encoding="utf-8"?>
2     <XML_Head version="1.1" listname="VD 一分鐘動態資訊"
3       updatetime="2009/10/06 11:31:23" interval="60">
4        <Infos>
5           <Info vdid="63000VD-1" status="0" datacollecttime
6             ="2009/10/06 11:30:00" >
7              <lane vsrdir = "0" vsrid="1" speed="24" laneoccupy="30">
8                 <cars carid="S" volumn="66" />
9                 <cars carid="T" volumn="74" />
10                <cars carid="L" volumn="60" />
11             </lane >
12          <lane vsrdir ="0" vsrid="2" speed="26" laneoccupy="30">
13                <cars carid="S" volumn="66" />
14                <cars carid="T" volumn="74" />
15                <cars carid="L" volumn="60" />
16          </lane >
            …
49489    </XML_Head>
```

圖 6-7 交通部 VD 動態資訊 XML 內容摘要

圖 6-8 VD 一分鐘動態資訊 XML 樹範例

6.2 應用結果

取得需要計算真實程度的基準文件及量測文件之後,使用者可透過「準備（Preparation）」、「分析（Analysis）」、「測量（Measure）」及「綜合處理（Synthesis）」等 4 階段程序評量 XML 真實程度，建立方向包含(1)識別真實程度的動機與目標；(2)決定真實維度；(3)計算分數的方法；(4)程度計分採用的尺度。

鑒於真實程度的決定是一個「開放的問題」，模型應用首先定義進行真實程度評價動機及目的：依據已知基準文件特性，評價已知 3 個不同的 XML 文件真實程度，目的係研究真實程度對已知不同文件的真時程度，提供未來使用該模型判別未知量測文件的鑑別能力參考；第二，本研究採用真時真實 XML 模型維度分別為文件來源 D1、文件結構 D2、文件內容 D3、文件時間 D4 等維度進行真實程度描述；第三，計算分數的方法設計於各量化因子，透過 Python 語言進行程式設計程式設計實作；第四，計算分數方法依[11]統一尺度由最小至最大為 0 至 100，以便綜整 XML 真實模型之真實程度計算結果。

原始真實程度為能符合本模型應用動機與目的，設計能表示使用者評價真實程度之權重，權重採用文檔宣告、結構比對、指定元素之原始真實程度加計 50%，即原始程度乘以權重值 150%，以表示編碼對及指定元素對於判別 XML 真實程度之重要性。

依據前述臺灣自動氣象站-氣象觀測資料為基準文件 B、以美國國家海洋暨大氣總署（NOAA）氣象資料[24]、OpenWeatherMap 即時氣象資訊[7]、台灣政府開放交通資料[37]分別做為被量測文件 M1、M2 與 M3 進行真實程度計算， 建立 XML 真實模型，模型各維度對應的真實屬性及量化因子整理如表 6-5。

表 6-5 真實維度、真實屬性及關鍵指標及其權重

真實維度	真實屬性	量化因子	權重
D_1 = 文件來源	$P_{1,1}$ =文檔宣告	$Q_{1,1,1}$ = XML 版本 $Q_{1,1,2}$ = 編碼	150%
	$P_{1,2}$ =URI	$Q_{1,2,1}$ = URI 網域名稱 $Q_{1,2,2}$ = URN	100%
D_2 = 文件結構	$P_{2,1}$ =結構規範	$Q_{2,1,1}$ = valid XML $Q_{2,1,2}$ = DTD 或 XSD	100%
	$P_{2,2}$ =結構比對	$Q_{2,2,1}$ = 最大結構深度 $Q_{2,2,2}$ = 最大樹寬 $Q_{2,2,3}$ = 元素數量	150%
D_3 = 文件內容	$P_{3,1}$ =元素	$Q_{3,1,1}$ = 指定元素名稱 $Q_{3,1,2}$ = 指定元素數量 $Q_{3.1.3}$ = 同義元素	150%
	$P_{3,2}$ =屬性	$Q_{3,1,1}$ = 屬性名稱 $Q_{3,1,2}$ = 屬性數量	100%
	$P_{3,3}$ =路徑	$Q_{3,3,1}$ = 指定路徑 $Q_{3,3,2}$ = 路徑數量	100%
D_4 = 文件時間	$P_{4,1}$ =檔案時間	$Q_{4,1,1}$= 建立時間	100%
	$P_{4,2}$ =順序性	$Q_{4,2,1}$ = 文檔更新頻率	100%

依據本研究 XML 真實模型的真實維度、真實屬性及量化因子，使用者即可依據資料特性自行設計不同量化函數， 最後設定出真實程度數值。

在真實維度 D_1「文件來源」計分方面，設定真實屬性為「文檔宣告」$P_{1,1}$及「URI」$P_{1,2}$，文檔宣告屬性分別有 $Q_{1,1,1}$「XML 版本」及 $Q_{1,1,2}$「編碼」以評估真實程度；URI 屬性分別有 $Q_{1,2,1}$ 「URI 網域名稱」及 $Q_{1,2,2}$「URN」評估真實程度。

其中量化因子 $Q_{1,1,1}$「XML 版本」擷取文件宣告的 XML version 進行比較，基準文件 B 之 XML 版本為 1.0、量測文件 M1 版本為 1.0、量測文件 M3 版本亦為 1.0，透過 XML 真實模型實作的計算方式 xml_version()，並考慮設計權重計 150%，得知 M1、M2 及 M3 分數皆為 150。

量化因子 $Q_{1,1,2}$「編碼」可擷取文件宣告的編碼（encoding）或透過程式自動判讀。判讀編碼如以文件宣告為主，其中基準文件 B 宣告編碼為 UTF-9、量測文件 M1 宣告編碼為 ISO-8859-1、量測文件 M2、M3 宣告編碼為 UTF-8；如果採用 Python 語言 chardet 模組之 UniversalDetector 方法偵測編碼，取得結果 B、M1、M2 及 M3 文件編碼分別為 UTF-8、ASCII、UTF-8 及 UTF-8，M1 宣告編碼與偵測編碼有差異。本研究採用宣告編碼，透過 XML 真實模型實作的計算方式 xml_encoding()，並考慮設計權重計 150%，以 B 文件為基準得知 M1、M2 及 M3 分數分別為 0、150 及 150。

量化因子 $Q_{1,2,1}$「URI 網域名稱」採用文檔宣告引述的 URI，其中基準文件未引述、量測文件 M1 引述 xsd=http://www.w3.org/2001/XMLSchema，M2、M3 未引述，透過 XML 真實模型實作的計算方式 xml_uri()，如果基準文件未引用 URI，量測文件有、無引用 URI 分別為 50、30，得知 M1、M2 及 M3 分別為 50、30 及 30。

量化因子 $Q_{1,2,2}$「URN」採用文檔宣告引述的 URN，其中基準文件引述 urn:cwb:gov:tw:cwbcommon:0.1、量測文件 M1、M2 及 M3 皆未引述 URN，透過 XML 真實模型實作的計算方式 xml_urn()，如果基準文件有引述 URN、量測文件沒有引用則分數為 20，得知 M1、M2 及 M3 皆為 20。

量化因子 $Q_{1,2,3}$「URL」採用文檔來源之網址，基準文件為 http://data.gov.tw/node/9176，量測文件 M1 之 URL 為 http://w1.weather.gov/xml/current_obs/、量測文件 M2 之 URL 為 http://api.openweathermap.org/data/2.5/weather?q=Taichung&mode=xml&appid=[apiid]及 M3 之 URL 為 http://tisvcloud.freeway.gov.tw/vd_value.xml.gz。分別透過網址的階層相似性計算真實程度，M1 有相同階層有 http 及 gov、M2 相同階層為 http、dara，M3 相同階層為 http、gov、tw，量測文件 M1、M2 及 M3 真實程度分別為 33.33、33.33 及 50。

在真實維度 D_2「文件結構」計分方面，設定真實屬性為「結構規範」$P_{2,1}$及「結構比對」$P_{2,2}$，結構規範屬性分別有 $Q_{2,1,1}$「valid XML」及 $Q_{2,1,2}$「DTD 或 XSD」以評估真實程度；結構比對屬性分別有 $Q_{2,2,1}$「最大結構深度」、$Q_{2,2,2}$「最大樹寬」及 $Q_{2,2,3}$「元素數量」評估真實程度。

量化因子 $Q_{2,1,1}$「valid XML」以基準文件及量測文件是否為 valid XML 進行真實程度計算，其中基準文件 B、量測文件 M1、M2 及 M3 僅有 M1 為 valid XML，依本研究 XML 真實模型所設計的計分方法 vaild_xml()進行真實程度計算，如果基準文件非 valid XML 而量測文件屬於 valid XML 則分數為 50 分、如果基準文件及量測文件皆非 valid XML 則分數為 20 分，計算真實程度結果 M1、M2 及 M3 分數分別為 50、20 及 20。

量化因子 $Q_{2,1,2}$「DTD 或 XSD」依照基準文件及量測文件引用 DTD 或 XSD 進行真實程度計算，基準文件 B、量測文件 M1、M2 及 M3 僅 M1 引述 XSD，文件 B、M2 及 M3 皆未引述 DTD 或 XSD，依本研究 XML 真實模型所設計的計分方法 vaild_type()進行真實程度計算，因基準文件 B 沒有引用 DTD 或 XSD，M1 引用 XSD 分數為 50，M2 及 M3 未引用則分數皆為 30。

在進行 $P_{2,2}$「結構比對」時，為取得 XML 文件的相關結構資訊，分別將基準文件 B、量測文件 M1、M2 及 M3 以本研究設計 XML 處理器之 XML2List() 方法解析文件結構，經解析為巢狀串列（nested list），XML 文件每行內容經解析後組合為子串列，子串列為深度、元素及屬性之集合，取得結構資訊如圖 6-9、6-10 及 6-11。

```
XML_file = 'O-A0001-001.xml'                # 基準文件B
parser = ET.XMLParser(target=XML2List())
parser.feed(open(XML_file).read())
pprint(depthList[:10])                      # 回傳前10個[深度,標籤,屬性.key()]
```

```
[[1, '{urn:cwb:gov:tw:cwbcommon:0.1}cwbopendata', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}identifier', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}sender', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}sent', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}status', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}msgType', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}dataid', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}scope', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}dataset', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}location', dict_keys([])]]
```

圖 6-9 基準文件 B 結構資訊摘述

```
1 XML_file = '42S39.xml'                # 量測文件M1
2 parser = ET.XMLParser(target=XML2List())
3 parser.feed(open(XML_file).read())
4 pprint(depthList[:10])                # 回傳前10個[深度,標籤,屬性.key()]
```

```
[[1,
  'current_observation',
  dict_keys(['version', '{http://www.w3.org/2001/XMLSchema-instance}no
NamespaceSchemaLocation'])],
 [2, 'credit', dict_keys([])],
 [2, 'credit_URL', dict_keys([])],
 [2, 'image', dict_keys([])],
 [3, 'url', dict_keys([])],
 [3, 'title', dict_keys([])],
 [3, 'link', dict_keys([])],
 [2, 'suggested_pickup', dict_keys([])],
 [2, 'suggested_pickup_period', dict_keys([])],
 [2, 'location', dict_keys([])]]
```

圖 6-10 量測文件 M1 結構資訊摘述

```
1 XML_depth_list('weather-openweathermap-taichung.xml') #量測文件M2
```

```
[[1, 'current', 'dict_keys([])'],
 [2, 'city', "dict_keys(['id', 'name'])"],
 [3, 'coord', "dict_keys(['lat', 'lon'])"],
 [3, 'country', 'dict_keys([])'],
 [3, 'sun', "dict_keys(['rise', 'set'])"],
 [2, 'temperature', "dict_keys(['value', 'unit', 'max', 'min'])"],
 [2, 'humidity', "dict_keys(['value', 'unit'])"],
 [2, 'pressure', "dict_keys(['value', 'unit'])"],
 [2, 'wind', 'dict_keys([])'],
 [3, 'speed', "dict_keys(['value', 'name'])"],
 [3, 'gusts', 'dict_keys([])'],
 [3, 'direction', "dict_keys(['value', 'name', 'code'])"],
 [2, 'clouds', "dict_keys(['value', 'name'])"],
 [2, 'visibility', "dict_keys(['value'])"],
 [2, 'precipitation', "dict_keys(['mode'])"],
 [2, 'weather', "dict_keys(['value', 'icon', 'number'])"],
 [2, 'lastupdate', "dict_keys(['value'])"],
```

圖 6-11 量測文件 M2 結構資訊摘述

```
1 XML_file = 'vd_value.xml'              # 量測文件M2
2 parser = ET.XMLParser(target=XML2List())
3 parser.feed(open(XML_file).read())
4 pprint(depthList[:10])                # 回傳前10個[深度,標籤,屬性.key()]
```

```
[[1, 'XML_Head', dict_keys(['interval', 'listname', 'version', 'update
time'])],
 [2, 'Infos', dict_keys([])],
 [3, 'Info', dict_keys(['datacollecttime', 'vdid', 'status'])],
 [4, 'lane', dict_keys(['laneoccupy', 'speed', 'vsrid', 'vsrdir'])],
 [5, 'cars', dict_keys(['volume', 'carid'])],
 [5, 'cars', dict_keys(['volume', 'carid'])],
 [5, 'cars', dict_keys(['volume', 'carid'])],
 [4, 'lane', dict_keys(['laneoccupy', 'speed', 'vsrid', 'vsrdir'])],
 [5, 'cars', dict_keys(['volume', 'carid'])],
 [5, 'cars', dict_keys(['volume', 'carid'])]]
```

圖 6-12 量測文件 M3 結構資訊摘述

本研究以 Python 語言實作之 DataGuides 方法，萃取 XML 文件結構中不重複的深度、元素、屬性等關鍵資訊，有助於對 XML 文件結構的理解，圖 6-10、6-11 及 6-12 分別為基準文件 B、量測文件 M1、M2 及 M3 之 DataGuides，各 XML 文件行數從 31,133、33、21 及 49,489 行，簡化為 25、22、16 及 5 行，有助於提升文件結構分析效率。

```
#基準文件B
dataGuides(xml_list)
```

```
[[1, '{urn:cwb:gov:tw:cwbcommon:0.1}cwbopendata', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}identifier', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}sender', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}sent', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}status', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}msgType', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}dataid', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}scope', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}dataset', dict_keys([])],
 [2, '{urn:cwb:gov:tw:cwbcommon:0.1}location', dict_keys([])],
 [3, '{urn:cwb:gov:tw:cwbcommon:0.1}lat', dict_keys([])],
 [3, '{urn:cwb:gov:tw:cwbcommon:0.1}lon', dict_keys([])],
 [3, '{urn:cwb:gov:tw:cwbcommon:0.1}locationName', dict_keys([])],
 [3, '{urn:cwb:gov:tw:cwbcommon:0.1}stationId', dict_keys([])],
 [3, '{urn:cwb:gov:tw:cwbcommon:0.1}time', dict_keys([])],
 [4, '{urn:cwb:gov:tw:cwbcommon:0.1}obsTime', dict_keys([])],
 [3, '{urn:cwb:gov:tw:cwbcommon:0.1}weatherElement', dict_keys([])],
 [4, '{urn:cwb:gov:tw:cwbcommon:0.1}elementName', dict_keys([])],
 [4, '{urn:cwb:gov:tw:cwbcommon:0.1}elementValue', dict_keys([])],
 [5, '{urn:cwb:gov:tw:cwbcommon:0.1}value', dict_keys([])],
 [3, '{urn:cwb:gov:tw:cwbcommon:0.1}parameter', dict_keys([])],
 [4, '{urn:cwb:gov:tw:cwbcommon:0.1}parameterName', dict_keys([])],
 [4, '{urn:cwb:gov:tw:cwbcommon:0.1}parameterValue', dict_keys([])]]
```

圖 6-13 基準文件 B 之 DataGuides

```
#量測文件M1
dataGuides(xml_list)
```

```
[[1,
  'current_observation',
  dict_keys(['{http://www.w3.org/2001/XMLSchema-instance}noNamespaceSchemaLocation', 'versio
n'])],
 [2, 'credit', dict_keys([])],
 [2, 'credit_URL', dict_keys([])],
 [2, 'image', dict_keys([])],
 [3, 'url', dict_keys([])],
 [3, 'title', dict_keys([])],
 [3, 'link', dict_keys([])],
 [2, 'suggested_pickup', dict_keys([])],
 [2, 'suggested_pickup_period', dict_keys([])],
 [2, 'location', dict_keys([])],
 [2, 'station_id', dict_keys([])],
 [2, 'observation_time', dict_keys([])],
 [2, 'observation_time_rfc822', dict_keys([])],
 [2, 'wind_string', dict_keys([])],
 [2, 'wind_dir', dict_keys([])],
 [2, 'wind_degrees', dict_keys([])],
 [2, 'wind_mph', dict_keys([])],
 [2, 'wind_gust_mph', dict_keys([])],
 [2, 'wind_kt', dict_keys([])],
 [2, 'pressure_string', dict_keys([])],
 [2, 'pressure_mb', dict_keys([])],
 [2, 'mean_wave_dir', dict_keys([])],
 [2, 'mean_wave_degrees', dict_keys([])],
 [2, 'disclaimer_url', dict_keys([])],
 [2, 'copyright_url', dict_keys([])],
 [2, 'privacy_policy_url', dict_keys([])]]
```

圖 6-14 量測文件 M1 之 DataGuides

```
#量測文件M2
dataGuides(m2)
```

```
[[1, 'current', 'dict_keys([])'],
 [2, 'city', "dict_keys(['id', 'name'])"],
 [3, 'coord', "dict_keys(['lat', 'lon'])"],
 [3, 'country', 'dict_keys([])'],
 [3, 'sun', "dict_keys(['rise', 'set'])"],
 [2, 'temperature', "dict_keys(['value', 'unit', 'max', 'min'])"],
 [2, 'humidity', "dict_keys(['value', 'unit'])"],
 [2, 'pressure', "dict_keys(['value', 'unit'])"],
 [2, 'wind', 'dict_keys([])'],
 [3, 'speed', "dict_keys(['value', 'name'])"],
 [3, 'gusts', 'dict_keys([])'],
 [3, 'direction', "dict_keys(['value', 'name', 'code'])"],
 [2, 'clouds', "dict_keys(['value', 'name'])"],
 [2, 'visibility', "dict_keys(['value'])"],
 [2, 'precipitation', "dict_keys(['mode'])"],
 [2, 'weather', "dict_keys(['value', 'icon', 'number'])"],
 [2, 'lastupdate', "dict_keys(['value'])"],
```

圖 6-15 量測文件 M2 之 DataGuides

```
#量測文件M3
dataGuides(m3)
```

```
[[1,
  'XML_Head',
  "dict_keys(['listname', 'updatetime', 'interval', 'version'])"],
 [2, 'Infos', 'dict_keys([])'],
 [3, 'Info', "dict_keys(['status', 'datacollecttime', 'vdid'])"],
 [4, 'lane', "dict_keys(['vsrdir', 'speed', 'laneoccupy', 'vsrid'])"],
 [5, 'cars', "dict_keys(['carid', 'volume'])"]]
```

圖 6-16 量測文件 M3 之 DataGuides

另外本研究在後續資料處理方面，進一步透過 Python 的 Pandas 模組以 DataFrame 資料格式進行資料處理，可提供使用者便捷的資料操作。圖 6-13、6-14 及 6-15 分別為基準文件 B、量測文件 M1、M2 及 M3 DataGuides 之 DataFrame，其中第一欄為 0 開始的 index，第二欄為深度 Depth、第三欄為元素 Tag、第四欄為屬性 Attirb。

	Depth	Tag	Attrib
0	1	cwbopendata	
1	2	identifier	
2	2	sender	
3	2	sent	
4	2	status	
5	2	msgType	
6	2	dataid	
7	2	scope	
8	2	dataset	
9	2	location	
10	3	lat	
11	3	lon	
12	3	locationName	
13	3	stationId	
14	3	time	
15	4	obsTime	
16	3	weatherElement	
17	4	elementName	
18	4	elementValue	
19	5	value	
20	3	parameter	
21	4	parameterName	
22	4	parameterValue	

圖 6-17 基準文件 B DataGuides 之 DataFrame

	Depth	Tag	Attrib
0	1	current_observation	'{http://www.w3.org/2001/XMLSchema-instance}no...
1	2	credit	
2	2	credit_URL	
3	2	image	
4	3	url	
5	3	title	
6	3	link	
7	2	suggested_pickup	
8	2	suggested_pickup_period	
9	2	location	
10	2	station_id	
11	2	observation_time	
12	2	observation_time_rfc822	
13	2	wind_string	
14	2	wind_dir	
15	2	wind_degrees	
16	2	wind_mph	
17	2	wind_gust_mph	
18	2	wind_kt	
19	2	pressure_string	
20	2	pressure_mb	
21	2	mean_wave_dir	
22	2	mean_wave_degrees	
23	2	disclaimer_url	
24	2	copyright_url	
25	2	privacy_policy_url	

圖 6-18 量測文件 M1 DataGuides 之 DataFrame

	Depth	Tag	Attrib
0	1	current	
1	2	city	'id', 'name'
2	3	coord	'lon', 'lat'
3	3	country	
4	3	sun	'set', 'rise'
5	2	temperature	'value', 'min', 'unit', 'max'
6	2	humidity	'value', 'unit'
7	2	pressure	'value', 'unit'
8	2	wind	
9	3	speed	'value', 'name'
10	3	gusts	
11	3	direction	'value', 'name', 'code'
12	2	clouds	'value', 'name'
13	2	visibility	'value'
14	2	precipitation	'mode'
15	2	weather	'value', 'number', 'icon'
16	2	lastupdate	'value'

圖 6-19 量測文件 M2 DataGuides 之 DataFrame

	Depth	Tag	Attrib
0	1	XML_Head	'version', 'updatetime', 'interval', 'listname'
1	2	Infos	
2	3	Info	'status', 'datacollecttime', 'vdid'
3	4	lane	'laneoccupy', 'vsrdir', 'speed', 'vsrid'
4	5	cars	'volume', 'carid'

圖 6-20 量測文件 M3 DataGuides 之 DataFrame

量化因子 $Q_{2,2,1}$「文件結構深度」以基準文件與量測文件最大深度進行真實程度計算，基準文件 B、量測文件 M1、M2 及 M3 最大深度分別為 5、3、3 及 5 層，以深度大者為分母、深度小者為分子，分數以 0 至 100 計算，使用 xml_max_depth 方法計算後，結果需採計權重 150%，M1、M2 及 M3 分數分別為 90、90 及 150。

量化因子 $Q_{2,2,2}$「最大樹寬」以基準文件與量測文件各階層元素總數，再取出最大元素數進行真實程度計算，本應用採用 DataGuides 進行分析，基準文件 B、量測文件 M1、M2 及 M3 最大樹寬分別為 9、22、10 及 1，以樹寬大者為分母、樹寬小者為分子，分數以 0 至 100 計算，並考慮設計權重計 150%，dg_eachtag_sum()方法計算結果 M1、M2 及 M3 分數為 61.37、135 及 16.67。

另外各階層數寬可透過資料視覺化進行觀察，圖 6-21 為文件 B、M1、M2 及 M3 各階層數寬分布圖，橫軸為深度，縱軸為元素數量，基準文件 B 各階層數寬分布為紅線、M1 為藍線、M2 為綠線、M3 為黑線。

```
1 xml_depth_plot(bxml)                          #基準文件B各階層數寬分布 -紅線
2 xml_depth_plot(m1 , color= "blue")   #量測文件M1各階層數寬分布-藍線
3 xml_depth_plot(m2 , color= "green") #量測文件M2各階層數寬分布-綠線
4 xml_depth_plot(m3 , color= "black") #量測文件M3各階層數寬分布-黑線
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10a0b3e48>
```

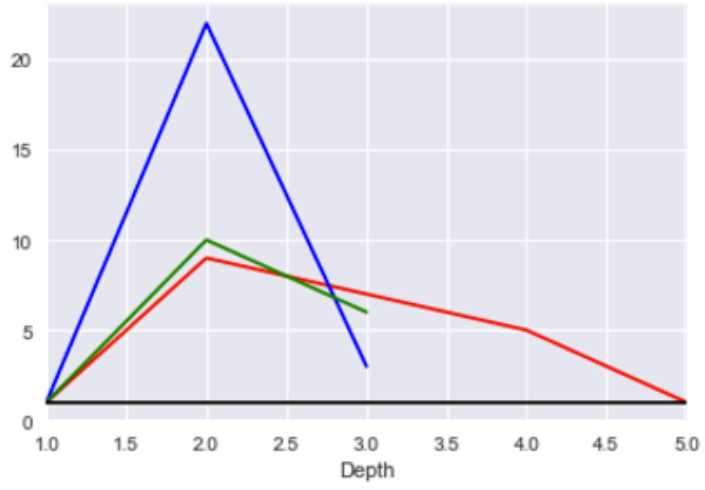

圖 6-21 文件 B、M1、M2 及 M3 各階層數寬分布圖

量化因子 $Q_{2,2,3}$「元素數量」透過基準文件與量測文件元素總數進行真實程度計算，本應用採用 DataGuides 進行分析，基準文件 B、量測文件 M1、M2 及 M3 元素總數分別為 23、26、17 及 5，XML 真實模型 xml_total_tags()方法計算，考慮設計權重計 150%後，M1 及 M3 分數分別為 132.69、110.87 及 32.61。

在真實維度 D_3「文件內容」計分方面，設定真實屬性為「元素」$P_{3,1}$ 及「路徑」$P_{3,2}$，元素分別有 $Q_{3,1,1}$「指定元素名稱」及 $Q_{3,1,2}$「指定元素數量」以評估真實程度；屬性分別有 $Q_{3,2,1}$「屬性名稱」及 $Q_{3,2,2}$「屬性數量」以評估真實程度；路徑分別有 $Q_{3,3,1}$「指定路徑」及 $Q_{3,3,2}$「路徑數量」評估真實程度。

量化因子 $Q_{3,1,1}$「指定元素名稱」以指定基準文件中的元素名稱存在與否進行真實程度計算，鑑於 B 文件應用範圍為氣象，本範例指定元素名稱為「location」，然而量測文件僅有 M1 有該指定元素，M2 及 M3 則無，依本研究 XML 真實模型所設計的計分方法 xml_key_tag ()進行真實程度計算，並考量設計權重加計 50%，由於皆沒有指定元素，計算真實程度結果 M1、M2 及 M3 分數為 150、0 及 0。

量化因子 $Q_{3,1,2}$「指定元素數量」以指定基準文件中的元素數量與量測文件之指定元素數量進行真實程度計算，本研究以圖 5-18 指定元素計分方法，指定元素為 temp，如不指定元素名稱則該方法進行元素相似程度進行計算，量測文件 M1、M2、M3 則無該指定元素，依本研究 XML 真實模型設計的計分方法 xml_key_tag_count()進行計算，計算真實程度結果 M1、M2 及 M3 分數皆為 0。

量化因子 $Q_{3,1,2}$「同義元素」以基準文件元素及自行增加之字詞之詞庫，模糊存在於量測文件之元素數量進行真實程度計算，本研究以圖 5-19 同義元素模糊比對方法進行計算，並且參考基準文件為氣象領域之 XML，自行增加與氣象相關之字詞列表如 ：['wind','temp','pressure','time','weather'] 建立基本詞庫，模糊比對量測文件 M1、M2 及 M3，結果基本字詞詞庫計有 28 個字詞，M1 模糊比對出相關字詞計有'wind_string'等 11 個、M2 模糊比對出相關字詞計有 wind'等 5 個、M3 則無相關字詞，字詞模糊比對結果如表 6-6。依本研究設計之計分方法 xml_fuzzyfinder()進行計算，分數為相

關字詞數量佔量測文件本身元素數量乘以 100，計算真實程度結果 M1、M2 及 M3 分數分別為 42.31、29.41 及 0，並考量設計權重加計 50%，M1、M2 及 M3 分數分別為 63.47、44.12 及 0。

表 6-6 字詞模糊比對結果

詞庫字詞	源自基準文件元素計 23 個： 'cwbopendata', 'identifier', 'sender', 'sent', 'status', 'msgType', 'dataid', 'scope', 'dataset', 'location', 'lat', 'lon', 'locationName', 'stationId', 'time', 'obsTime', 'weatherElement', 'elementName', 'elementValue', 'value', 'parameter', 'parameterName', 'parameterValue' 使用者自行添加字詞計 5 個： 'wind', 'temp', 'pressure', 'time', 'weather'
M1 相關字詞	'wind_string', 'wind_degrees', 'wind_gust_mph', 'wind_mph', 'observation_time', 'pressure_mb', 'wind_kt', 'pressure_string', 'location', 'observation_time_rfc822', 'wind_dir'等 11 個
M2 相關字詞	wind', 'temperature', 'weather', 'lastupdate', 'pressure'等 5 個
M3 相關字詞	無相關字詞

量化因子 $Q_{3,2,1}$「屬性名稱」及 $Q_{3,2,2}$「屬性數量」計算真實程度部分，本範例不指定屬性，依本研究 xml_key_attr()方法則將採用基準文件與量測文件共同屬性比例進行真實程度計算，鑑於解析基準文件 B 得知該文件未使用屬性標記資訊，且量測文件 M1 未採用屬性、M2 及 M3 有採用，故採用本研究 XML 真實模型所設計的計分方法 xml_key_tag() 及 xml_total_attrs()計算真實程度，M1、M2 及 M3 分數分別為 30、20、20。

量化因子 $Q_{3,3,1}$「指定路徑」及 $Q_{3,3,2}$「路徑數量」以基準文件指定具有路徑關聯的 2 個元素，搜尋量測文件中指定的路徑是否存在、路徑數量作為判別真實程度計算，本研究指定在基準文件 B 大量出現的「weatherElement」及其子元素「elementValue」作為指定路徑，經比對量測文件 M1、M2 及 M3 皆未具有該指定路徑，故採用本研究 XML 真實模

型所設計的計分方法 xml_path ()及 xml_total_attrs()計算真實程度，M1、M2 及 M3 分數皆為 0。

在真實維度 D_4「文件時間」計分方面，設定真實屬性為「檔案時間」$P_{4,1}$及「順序性」$P_{4,2}$，檔案時間屬性有 $Q_{4,1,1}$「建立時間」、順序性有 $Q_{4,2,1}$「文檔更新頻率」評估真實程度。

量化因子 $Q_{4,1,1}$「建立時間」以基準文件及量測文件合理的建立時間計算真實程度，本範例以使用者未採用建立合理時間區間進行判別，透過本研究圖 5-23 之 xml_time_delta()方法，改採基準文件與量測文件在差異日期計算真實程度，M1、M2 及 M3 建立時間與基準文件差異分別為 147 日、63 日及 143 日， M1、M2 及 M3 真實程度分別為 59.73、82.74 及 60.82。

```
import time, os
from pprint import pprint

b_time  = time.localtime(os.stat('O-A0001-001.xml').st_atime)
m1_time = time.localtime(os.stat('42S39.xml').st_ctime)
m2_time = time.localtime(os.stat('weather-openweathermap-taichung.xml').st_ctime)
m3_time = time.localtime(os.stat('vd_value.xml').st_ctime)

pprint(b_time)
pprint(m1_time)
pprint(m2_time)
pprint(m3_time)
```

```
time.struct_time(tm_year=2017, tm_mon=8, tm_mday=6, tm_hour=22, tm_min=29, tm_sec=28, tm_wday=6, tm_yday=218, tm_isdst=0)
time.struct_time(tm_year=2016, tm_mon=12, tm_mday=22, tm_hour=20, tm_min=38, tm_sec=44, tm_wday=3, tm_yday=357, tm_isdst=0)
time.struct_time(tm_year=2017, tm_mon=7, tm_mday=20, tm_hour=22, tm_min=27, tm_sec=42, tm_wday=3, tm_yday=201, tm_isdst=0)
time.struct_time(tm_year=2016, tm_mon=12, tm_mday=26, tm_hour=16, tm_min=15, tm_sec=16, tm_wday=0, tm_yday=361, tm_isdst=0)
```

圖 6-22 偵測文件建立時間

量化因子 $Q_{4,2,1}$「文檔更新頻率」以基準文件的來源更新頻率，以及比對量測文件來源的更新頻率兩者進行真實程度計算，其中基準文件 B、量測文件 M1、M2 及 M3 更新頻率分別為 60 分鐘、20 分鐘、10 分鐘及 1 分鐘，依本研究 XML 真實模型所設計的計分方法 xml_time_frequency()進行真實程度計算， M1、M2 及 M3 分數分別為 33.33、16.7 及 1.67。

綜合前述各真實維度之真實屬性具有的量化因子，判別真實程度比對情形整理如表 6-7，依據使用者自定義的量化函數計算出各個量化因子，並統一尺度範圍為 0 至 100，部分真實屬性設計有權重 150%。

表 6-7 真實程度比對情形

量化因子	文件 B	文件 M_1	文件 M_2	文件 M3
$Q_{1,1,1}$ XML 版本	1.0	1.0	1.0	1.0
$Q_{1,1,2}$ 編碼	UTF8	ISO-8859-1	UTF-8	UTF-8
$Q_{1,2,1}$ URI 網域層級	無	xmlns:xsd="http://www.w3.org/2001/XMLSchema"	無	無
$Q_{1,2,2}$ URN	urn:cwb:gov:tw:cwbcommon:0.1	無	無	無
$Q_{1,2,3}$ URL	http://data.gov.tw/node/9176	http://w1.weather.gov/xml/current_obs/	http://api.openweathermap.org/data/2.5/weather?q=Taichung&mode=xml&appid=[apiid]	http://tisvcloud.freeway.gov.tw/vd_value.xml.gz
$Q_{2,1,1}$ valid XML	否	是	否	否
$Q_{2,1,2}$ DTD 或 XSD	無	XSD	無	無·
$Q_{2,2,1}$ 最大結構深度	5	3	3	5
$Q_{2,2,2}$ 最大樹寬	9	22	10	1
$Q_{2,2,3}$ 元素數量	23	26	17	5
$Q_{3,1,1}$ 元素名稱	location	有	無	無
$Q_{3,1,2}$ 同義元素	28 組字詞	11 組相似	5 組相似	0 組相似
$Q_{3,1,2}$ 元素數量	temp	0	0	0
$Q_{3,1,1}$ 屬性名稱	無屬性	無	有	有
$Q_{3,1,2}$ 屬性數量	無屬性	無	有	有
$Q_{3,3,1}$ 指定路徑	weatherElement-elementValue	無	無	無
$Q_{3,3,2}$ 路徑數量	1	無	無	無
$Q_{4,1,1}$ 建立時間	2017/5/18	2016/12/22 (差 147 日)	2017/7/20 (差 63 日)	2016/12/26 (差 143 日)
$Q_{4,2,1}$ 文檔更新頻率	60 分鐘	20 分鐘	10 分鐘	1 分鐘

接著使用者以 XML 真實模型的階層性計算各維度權重，各量化因子計算真實程度分數整理如表 6-8。

表 6-8 真實程度

真實維度	真實屬性	量化因子 5	M1 真實程度	M2 真實程度	M3 真實程度	已包含權重
D_1 文件來源	$P_{1,1}$ 文檔宣告	$Q_{1,1,1}$ XML 版本	150	150	150	150%
		$Q_{1,1,2}$ 編碼	0	150	150	
	$P_{1,2}$ URI	$Q_{1,2,1}$ URI	50	30	30	100%
		$Q_{1,2,2}$ URN	20	20	20	
		$Q_{1,2,3}$ URL	33.33	33.33	50	
D_2 文件結構	$P_{2,1}$ 結構規範	$Q_{2,1,1}$ valid XML	50	20	20	100%
		$Q_{2,1,2}$ DTD 或 XSD	50	30	30	
	$P_{2,2}$ 結構比對	$Q_{2,2,1}$ 最大結構深度	90	90	150	150%
		$Q_{2,2,2}$ 最大樹寬	61.37	135	16.67	
		$Q_{2,2,3}$ 元素數量	132.69	110.87	32.61	
D_3 文件內容	$P_{3,1}$ 元素	$Q_{3,1,1}$ 指定元素名稱	150	0	0	150%
		$Q_{3,1,2}$ 指定元素數量	0	0	0	
		$Q_{3,1,2}$ 同義元素	63.47	44.12	0	
	$P_{3,2}$ 屬性	$Q_{3,2,1}$ 屬性名稱	30	20	20	100%
		$Q_{3,2,2}$ 屬性數量	30	20	20	
	$P_{3,3}$ 路徑	$Q_{3,3,1}$ 指定路徑	0	0	0	100%
		$Q_{3,3,2}$ 路徑數量	0	0	0	
D_4 文件時間	$P_{4,1}$ 檔案時間	$Q_{4,1,1}$ 建立時間	59.73	82.74	60.82	100%
	$P_{4,2}$ 順序性	$Q_{4,2,1}$ 時間間隔	33.33	16.7	1.67	

本研究計算各真實維度分數採用量化因子加總計算，並考量評量真實程度之動機與目的，於 $P_{1,1}$ 文檔宣告、$P_{2,2}$ 結構比對、$P_{3,1}$ 指定元素等 3 項真實屬性之原始真實程度加計 50%，即於 $Q_{1,1,1}$ XML 版本、$Q_{1,1,2}$ 編碼、$Q_{2,2,1}$ 最大結構深度、$Q_{2,2,2}$ 最大樹寬、$Q_{2,2,3}$ 元素數量、$Q_{3,1,1}$ 指定元素名稱、$Q_{3,1,2}$ 指定元素數量之權重為 150%，以表示該真實屬性特徵對於判別 XML 真實程度之重要性。

鑒於本模型經過加權分數設計，M1 經加權之各真實維度分數分別為 253.33、384.06、240 及 93.06；M2 分別為 383.33、385.87、40 及 99.44；M3 分別為 400、249.28、40 及 62.49。為使 XML 真實程度模型對使用者有更好的理解能力，本研究應用案例將真實維度分數透過所屬之真實屬性及其權重，以算術平均數計算，得知 M1 各真實維度分數為 50.67、76.81、43.35、46.53；M2 各真實維度分數為 76.67、77.17、12.02、49.72；M3 各真實維度分數分別為 80、49.86、6.67、31.25。然而各真實屬性因有權重不同導致真實維度計算時非 0 至 100 之尺度關係，接續透過正規化將各真實維度最小值及最大值限制於 0 至 100 之尺度範圍內，正規化後真實維度 M1 分數分別為 42.22、59.09、35.70、46.53；M2 各真實維度分數為 63.89、59.36、9.90、49.72；M2 各真實維度分數為 66.67、38.35、5.49、31.25，如表 6-7 所示，經正規化雷達圖測繪真實程度如圖 6-23。

表 6-7 真實程度-正規化前、後比較表

	不含權重			含權重(未正規化)			真實程度(正規化後)		
維度	M1	M2	M3	M1	M2	M3	M1	M2	M3
D1	40.67	56.67	60.00	50.67	76.67	80.00	42.22	63.89	66.67
D2	57.87	54.78	36.57	76.81	77.17	49.86	59.09	59.36	38.35
D3	36.04	9.92	5.71	43.35	12.02	5.71	35.70	9.90	5.49
D4	46.53	49.72	31.25	46.53	49.72	31.25	46.53	49.72	31.25

除透過資料視覺化方式以雷達圖觀察文件真實程度情形，本模型應用之真實程度值採用各維度於雷達圖標示之面積做為真實程度值，$V_{M1|B}$、$V_{M2|B}$ 與 $V_{M3|B}$ 分別為 4115.02、4024.39 及 2510.91，而 V_B 真實程度最大

為 27216.67，使用者可依據計算出的真實值對照 V_B 最大值，做為 XML 文件真實程度參考。

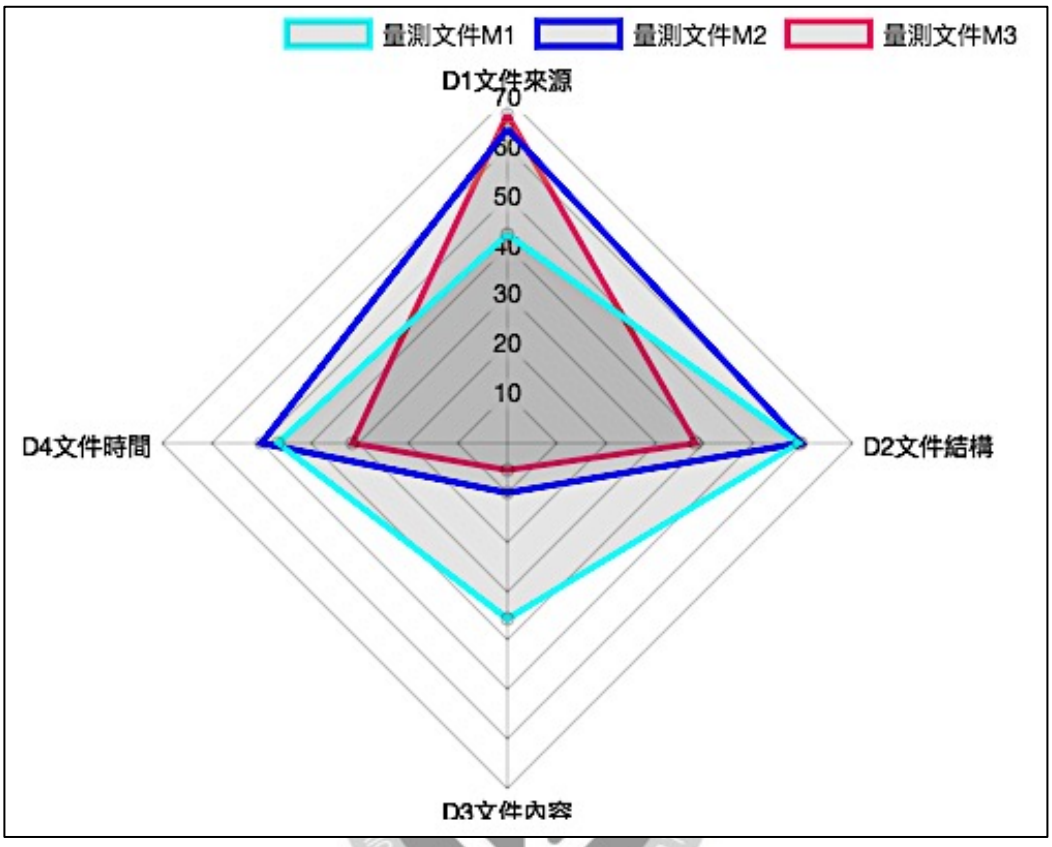

圖 6-23 量測文件 M1、M2、M3 真實程度視覺化呈現

面對巨量資料的大、快、疑、雜特性，資料真實性並非能在有限時間內清晰分辨，如能在有限時間內提供判斷來源 XML 的真實程度有其參考價值。然而為了示範模型應用成果，本章模型應用採已知彼次不相同之 XML 文件 4 份，目的係研究真實程度對已知不同文件的真時程度，提供未來使用該模型判別未知量測文件的鑑別能力參考。

採用不同 XML 文件及眾多量化因子進行真實程度評價結果，使用者可依評價真實程度之動機及目的解讀真實程度值代表之意義。在本章模型應用中，如僅採用單一指標觀察，部分量化因子分數為最高尺度值 100 或幾近最高尺度值，然而透過 XML 真實程度模型綜合判斷，各 XML 文件真實維度分數反映基準文件與量測文件之間有諸多差異性，舉例如量測文件 M1、M2 及 M3 各真實維度僅有 M1 之 D_2 文件結構、M2 之 D_1 文件來源及 M3 之 D_1 文件來源計分超過 50，真實維度 D_3 文件內容及 D_4 文件時間維

度計分皆不超過 50，依本應用之動機可解讀為 D_3 文件內容、D_4 文件時間具有鑑別真實程度之代表性，惟仍須綜合觀察各真實維度之間關係俾利真實程度計分結果反映量測文件之真實性。

另外本章模型應用考量動機與目的，表示編碼對及指定元素對於判別 XML 真實程度之重要性，設計權重 $P_{1,1}$ 文檔宣告、$P_{2,2}$ 結構比對、$P_{3,1}$ 指定元素之原始真實程度加計 50%，結果顯示源自臺灣之開放資訊之文件 B 及 M3 皆採用 UTF-8，源自美國之 NOAA 氣象資訊編碼採用 ISO-8859-1、OpenWeatherMap.org 採用 UTF-8，顯示以編碼判別使用國家或區域有其侷限；最大結構深度及指定元素相同為 XML 文件真實性具備之特徵之一，權重設計突顯該真實屬性的重要性，增加真實程度計算結果之參考價值。

最後以本應用所使用文件觀察，基準文件 B、量測文件 M1 及 M2 為氣象資訊領域之 XML，M3 為交通領域之 XML，在設計模型計分方式時，如有考量 XML 不同領域之應用特性，在本範例之應用結果有真實程度高低之分，交通領域之 M3 真實程度為 2510.91，相較於與基準文件相同領域之量測文件 M1、M2 真實程度 4115.02 及 4024.39 有程度高低之別。

在本應用中，巨量 XML 真實程度模型之真實程度反應，使用者仍須考量巨量資料環境特性，即便彼此之間不同的文件仍有可能在某項重要之真實屬性相似，使用者於設計模型時亦應考量不同文件遇到相似真實程度之情形，透過使用者對資料理解性設計真實模型計分方式有助於增加真實程度參考價值。

第七章 結論與未來工作方向

7.1 結論

XML 作為網路應用普及的資料格式，隨著資訊普及與數據量指數成長趨勢，XML 資料同樣面臨巨量資料的大量、快速、多樣、可信及價值等 5V 特性，由於資料來源、資料搜集技術等因素而造成資料偏差，如能將 XML 文件資料特性以多種角度描述，協助資料使用者對資料有真實程度的判斷，將有助於加速資料處理及對資料的理解。

本論文建立巨量 XML 真實程度模型，適用於巨量資料情境之下，協助資料使用者依照本論文模型架構定義 XML 真實模型及程度計算方法，計算真實程度及透過資料視覺化協助判讀 XML 文件真實性。在建立巨量 XML 真實程度模型過程中，首先逐步建構具有階層性的真實維度、真實屬性及量化因子；接續設計模型介面方法，透過物件導向設計、統一塑模語言 UML 以類別圖表達設計需求，以 Python 程式語言進行模型介面設計，並實作 XML 真實模型的計分方法，最後以模型應用範例闡述應用結果，使 XML 得以具體量化資料真實性。

本研究在學術方面具有量化資料特性的 veracity 真實程度估算架構與方法，對於探討巨量資料 veracity 特性的研究可以基於此模型發展合適的真實維度、真實屬性、量化因子及真實程度的計算方式；在應用的重要性方面，本研究可以產生新的應用服務，幫助產業界開發創新的應用服務，提升產業巨量資料分析的能量。

7.2 未來工作方向

本研究提出能適應巨量資料環境之巨量 XML 真實程度模型，透過 Python 物件導向程式設計，提出具有彈性、適應性，能具體量化程度的 XML 真實程度計分方法，後續研究者可就模型介面設計、巨量資料情境運用分析、資料視覺化應用、模型應用效能等面向繼續發展。

參考文獻

[1] Algergawy, A., Nayak, R., & Saake, G. (2010). Element similarity measures in XML schema matching. *Information Sciences*, *180*(24), 4975-4998. http://dx.doi.org/10.1016/j.ins.2010.08.022

[2] *Apache Spark™ - Lightning-Fast Cluster Computing*. (2017). *Spark.apache.org*. Retrieved 7 April 2017, from http://spark.apache.org/

[3] *Big Data and Analytics*, *2015*. (2017). Retrieved 30 May 2017, from http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[4] Big data means big opportunities for chemical companies. https://home.kpmg.com/xx/en/home/insights/2016/07/big-data-means-big-opportunities-chemical-companies.html. Accessed 20 March, 2017.

[5] Chery P. Product document description: NWS current observations using RSS and XML based formats 5/7/2004 part I -mission connection part II - technical description, (2008). http://products.weather.gov/PDD/NWS_Current_Observations_RSS_XML.pdf. Accessed January 21, 2017.

[6] Costa, G., Manco, G., Ortale, R., & Ritacco, E. (2013). Hierarchical clustering of XML documents focused on structural components. *Data & Knowledge Engineering*, *84*, 26-46. http://dx.doi.org/10.1016/j.datak.2012.12.002

[7] *Current weather and forecast - OpenWeatherMap*. (2017). *Openweathermap.org*. Retrieved 28 July 2017, from http://openweathermap.org/

[8] *Duck typing*. (2017). *En.wikipedia.org*. Retrieved 11 March 2017, from https://en.wikipedia.org/wiki/Duck_typing

[9] *EbXML*. (2017). *En.wikipedia.org*. Retrieved 2 July 2017, from https://en.wikipedia.org/wiki/EbXML

[10] *Extensible Markup Language (XML) 1.1 (Second Edition)*. (2017). *W3.org*. Retrieved 10 July 2017, from https://www.w3.org/TR/xml11/#sec-xml11

[11] Fogli, D., & Guida, G. (2015). A practical approach to the assessment of quality in use of corporate web sites. *Journal Of Systems And Software*, *99*, 52-65. http://dx.doi.org/10.1016/j.jss.2014.09.006

[12] *Frequently asked questions — chardet 3.0.3 documentation*. (2017). *Chardet.readthedocs.io*. Retrieved 7 March 2017, from http://chardet.readthedocs.io/en/latest/faq.html?highlight=xml

[13] Hughes, R. (1996). Expert judgement as an estimating method. *Information And Software Technology*, *38*(2), 67-75. http://dx.doi.org/10.1016/0950-5849(95)01045-9

[14] *IntelliJ IDEA: the Java IDE for Professional Developers | JetBrains*. (2017). *JetBrains*. Retrieved 3 March 2017, from https://www.jetbrains.com/idea/

[15] *Java Platform SE 7*. (2017). *Docs.oracle.com*. Retrieved 25 March 2017, from https://docs.oracle.com/javase/7/docs/api/

[16] Knuth, D. E. (2011). The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1. Pearson Education India.

[17] Li, S. (2017). *A composite approach to language/encoding detection*. *Www-archive.mozilla.org*. Retrieved 17 MArch 2017, from http://www-archive.mozilla.org/projects/intl/UniversalCharsetDetection.html

[18] Lukoianova, T., & Rubin, V. (2014). Veracity Roadmap: Is Big Data Objective, Truthful and Credible?. *Advances In Classification Research Online*, *24*(1), 4. http://dx.doi.org/10.7152/acro.v24i1.14671

[19] *Object-oriented design*. (2017). *En.wikipedia.org*. Retrieved 24 March 2017, from https://en.wikipedia.org/wiki/Object-oriented_design

[20] *PyPI - the Python Package Index : Python Package Index*. (2017). *Pypi.python.org*. Retrieved 10 July 2017, from https://pypi.python.org/pypi

[21] Roy Goldman, J. (2017). *DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases*. *Citeseerx.ist.psu.edu*. Retrieved 11 July 2017, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.8594

[22] Saha, B., & Srivastava, D. (2014, March). Data quality: The other face of big data. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*(pp. 1294-1297). IEEE.

[23] Service NoNW, Team NIS. XML data feeds - current conditions - NOAA's national weather service. http://w1.weather.gov/xml/current_obs/. Accessed January 17, 2017.

[24] Service NoNW, Team NIS. XML data feeds - current conditions - NOAA's national weather service. http://w1.weather.gov/xml/current_obs/all_rss.zip . Accessed January 3, 2017.

[25] Tai, K. (1979). The Tree-to-Tree Correction Problem. *Journal Of The ACM*, *26*(3), 422-433. http://dx.doi.org/10.1145/322139.322143

[26] Tekli, J., & Chbeir, R. (2012). A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics. *Web Semantics: Science, Services And Agents On The World Wide Web*, *11*, 14-40. http://dx.doi.org/10.1016/j.websem.2011.10.002

[27] Tekli, J., Chbeir, R., Traina, A., Traina, C., & Fileto, R. (2015). Approximate XML structure validation based on document–grammar tree similarity. *Information Sciences*, *295*, 258-302. http://dx.doi.org/10.1016/j.ins.2014.09.04

[28] *The 2017 Top Programming Languages*. (2017). *IEEE Spectrum: Technology, Engineering, and Science News*. Retrieved 28 May 2017, from http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

[29] *The cElementTree Module*. (2017). *Effbot.org*. Retrieved 10 July 2017, from http://effbot.org/zone/celementtree.htm

[30] *TIOBE Index | TIOBE - The Software Quality Company*. (2017). *Tiobe.com*. Retrieved 10 May 2017, from https://www.tiobe.com/tiobe-index

[31] *URL*. (2017). *En.wikipedia.org*. Retrieved 15 May 2017, from https://en.wikipedia.org/wiki/URL

[32] *W3C Math Home*. (2017). *W3.org*. Retrieved 1 July 2017, from http://www.w3.org/Math/

[33] *W3C XML Schema*. (2017). *W3.org*. Retrieved 4 May 2017, from https://www.w3.org/XML/Schema

[34] *Welcome to Apache™ Hadoop®!*. (2017). *Hadoop.apache.org*. Retrieved 4 July 2017, from http://hadoop.apache.org/

[35] *xml.etree.ElementTree — The ElementTree XML API — Python 3.6.2rc2 documentation*. (2017). *Docs.python.org*. Retrieved 20 February 2017, https://docs.python.org/3.6/library/xml.etree.elementtree.html#xml.etree.ElementTree.iterparse

[36] *XPath Tutorial*. (2017). *W3schools.com*. Retrieved 10 July 2017, from https://www.w3schools.com/xml/xpath_intro.asp

[37] 交通部. 路側設施即時交通資訊 http://tisvcloud.freeway.gov.tw/vd_value.xml.gz . Accessed January 10, 2017.

[38] 交通部. 路側設施即時交通資訊.發布標準格式 v1.1. http://tisvcloud.freeway.gov.tw/路側設施即時交通資訊發布標準格式(含補充).pdf Accessed 11 January 2017.

[39] *定義類別*. (2017). *OPENHOME.CC*. Retrieved 3 July 2017, from https://openhome.cc/Gossip/Python/Class.html

[40] 林信良(2016). *Python3.5 技術手冊*. 碁峰資訊股份有限公司

[41] 國家發展委員會.自動氣象站-氣象觀測資料. http://data.gov.tw/node/9176. Accessed January 10, 2017.

[42] 國家發展委員會.自動氣象站資料集說明檔. http://data.gov.tw/wise_search?nodetype=metadataset&sort=BrowseCount_1 . Accessed January 22, 2017.

[43]國家發展委員會.自動氣象站資料集說明檔. http://opendata.cwb.gov.tw/opendatadoc/DIV2/A0001-001.pdf. Accessed January 7, 2017.

[44]*域名*. (2017). *Zh.wikipedia.org*. Retrieved 5 May 2017, from https://zh.wikipedia.org/wiki/%E5%9F%9F%E5%90%8D

[45]*統一識別模型*. (2017). *Zh.wikipedia.org*. Retrieved 10 July 2017, from https://zh.wikipedia.org/wiki/%E7%BB%9F%E4%B8%80%E8%B5%84%E6%BA%90%E6%A0%87%E5%BF%97%E7%AC%A6