基於Google網站服務之及時氣象估測系統
A Just-In-Time Meteorological Estimating System based on Google Web Services

何以宣*、張惟婷†、曹佑嘉、林軒至、黃連進
淡江大學資訊工程學系
E-mail: *dodo8484@gmail.com, †tt90089@gmail.com

摘要 -目前常見的媒體平台提供的氣象資訊都有及時性不足及區域劃分不精細的問題。因此本研究利用中央氣象局所屬的氣象觀測站，根據使用者指定的地點，找出三座最近的觀測站氣象資訊，配合數學公式、演算法以及氣象學等理論，估測出較及時的氣象資訊。並整合Google Map API 提供友善的介面和精確的定位。背後使用雲端運算作爲整套系統的伺服器，負責處理氣象資料的更新及支援系統的運作。而以網頁的呈現方式，更能跨越各種備有瀏覽器的電子平台。

關鍵詞：GAE，Google Map，及時氣象，雲端運算

一、 緒論

1-1 研究動機

自古以來氣象資訊就和民眾的生活息息相關，尤其近幾年來，氣候變化異常，導致氣象難以預測；在災害頻傳的同時，此議題更被重視。然而目前各媒體平台所提供的氣象資訊，不論是報紙、新聞或是網頁，在時間的區分上面大多以"上午"或"下午"爲單位，其時效性可能長達數小時，無法提供及時的氣象資訊。

另外，氣象資訊的取得須仰賴氣象觀測站中各個儀器所測得的資料，而一座完整的氣象站維護不易，需耗費一定的資源和人力。也因此較難建立密集的氣象觀測網，也造成民眾獲得的氣象資訊在區域的劃分上大都以"縣"或"市"爲單位。但對於大範圍區域而言，其天氣變化在各個較小範圍區域的城鎮還是有相當程度上的落差，無法提供更精確的氣象資訊。

因此如果能將分散的氣象資訊整合，並依據推估公式加以推估，提供較及時且精確的氣象資訊，將會是非常實用的一項研究。我們選擇Google雲端運算作爲服務平台，對氣象資訊進行儲存和運算；它以服務導向架構爲主，能更有效減少系統開發與維護的成本與風險[2]。

1.1 實作概述

本研究主要針對三個層面進行探討與實作：

第一個層面以氣象資訊的估測爲主軸，必須結合大氣學、數學和統計學來對氣象資料進行運算推測，而對於不同的氣象型態如風速、溫度和氣壓等，皆有個別的估測方法。每個氣象因子間也是息息相關。整體而言，最重要的是估測的準確性。

第二層面則是雲端運算部分，雲端運算的基礎設備成本非常高，通常需龐大的資金才能架設。因此使用者通常使用中階的雲端運算平台[4]，根據CPU、網路與資料庫等使用量付費，這正爲本研究所著重的部分，故使用雲端運算平台作伺服器，來處理及推估氣象資訊[3]。

第三層面則是根據研究[1]的實作方法，以Web base爲系統的主要架構，讓任何具備有瀏覽器的電子產品皆可使用本系統，並嵌入Google Maps，將全臺灣的氣象觀測站標於地圖，詳列氣象資訊，讓使用者能透過Google Maps輕易的指定確切地點進而獲得估測出的氣象資訊。

二、 系統實作

2-1 系統運作方式

我們將伺服器建置於Google所提供的免費應用服務平台，作爲雲端運算伺服器，如此可免除網站的建置與維護成本[10]。於該平台開發應用程式，主動定時至中央氣象局與環保署取得各觀測站之量測資料，存入資料庫中，作爲估測依據。本系統幾乎支援目前所有瀏覽器，甚至智慧型手機上面專用的瀏覽器也可以操作本系統。

2-2 雲端運算伺服器

以Google應用服務引擎(Google App Engine，GAE)進行實作，建立一個專案，並使用GAE的SDK模擬雲端環境架設本系統。在確定程式穩定度和完成度後，才進一步的將整個專案上傳至GAE伺服器，測試本系統在雲端運算環境下真正的執行速度以及網路流量的控制等。

2-3 建置資料庫

依照不同的氣象因子-溫度、濕度、風向、風速和紫外線，建立各氣象觀測站的資料庫，儲存各項氣象數據並記錄氣象觀測站的站名與座標。GAE的資料庫-Datastore，爲非關聯式資料庫，以BigTable格式儲存，具高效能和高壓縮率的特質，搜尋語法爲類似SQL的GQL語法，較特別的GAE是以呼叫API進行資料庫的存取。

2-4 網頁解析

撰寫兩支Web spider程式，一支前往中央氣象局所公開提供的氣象資訊網頁，針對全臺灣46座氣象站[5]蒐集各個溫度、濕度、風向、風速、氣壓，接著呼叫API 將資料存進Datastore。另一支前往行政院環保署的紫外線觀測網頁，同樣解析HTML格式後，存進資料庫內。

2-5 客戶端

本系統使用AJAX作爲客戶端互動網頁的技術，其中包含了Django的Web框架[12]、Google Maps和JavaScript提供即時的互動模式，讓使用介面多樣化且友善。然而在氣象觀測站的搜尋、觀測點的判定及氣象數據的推測上，以JavaScript爲主，此架構讓系統能具有較及時的反應。

2-5 嵌入Google Maps

而GAE所提供的是轉址過後的網址，無法得知實際網址，故在嵌入Google Map的實作上，有一定的限制，

我們選擇使用Google Maps API第3版。因爲這個版本不需要API Key，能夠完全地與GAE相容，且載入速度更快，效能更高，還多了地形、街景和路線規劃等服務。

而地形可說是影響氣象數據的關鍵因素之一[9]，Google Maps對每一個位置皆提供地形高度，本系統也將高度數值放入氣象數據估測的推導中，提升氣象數值的精準度，不論在理論或實作上，可以更具說服力。

三、 氣象資料估測

3-1 估測區域

目前實作的估測範圍以中華民國領土爲主。估測區域的判定上大致分爲兩步驟。

第一步是參考經緯度，根據觀測點的經緯度來判斷位於臺灣本島、外島抑或是臺灣以外的非觀測地點。外島包含澎湖、金門、馬祖、彭佳嶼、東吉島、蘭嶼、綠島、琉球嶼和釣魚臺，皆列於本系統的估測範圍內。實作方法是先求出島嶼東西南北的經緯度極限值，形成一四方形區塊，觀測點只有在區塊內才能進行第二步的判讀，否則視爲不在於估測範圍。

第二步是判斷觀測點是否位於陸地上。根據Google Maps所提供的高度數據，以海平面爲基準，高於海平面爲則爲陸地；低於海平面表示估測點位於海上，不適用於本系統的估測演算法，因此不列入估測範圍內。

3-2 觀測站搜尋

全臺灣共有46座氣象觀測站，其中臺灣本島占了38座，外島包含：澎湖、金門、馬祖、彭佳嶼、東吉島、蘭嶼、綠島、琉球嶼各有一座氣象觀測站。觀測點如果位於外島，則直接使用島上的觀測站資訊，而臺灣本島則是必須搜尋最近的氣象觀測站，作爲估測的依據。

搜尋的實作上，先依各氣象站與觀測點的距離進行排序。由近到遠，排序好之後，根據最近的三座氣象站，使用Heron's formula[13]判斷觀測點是否位於此三座氣象站所形成的三角形內。

假使排序好的氣象站由近而遠分別爲 St_0 、 St_1 、 St_2 、…、 St_{37} ，若最近的前三座氣象站 St_0 、 St_1 、 St_2 形成之三角形無法將觀測點圍住，則固定 St_0 與 St_1 ，將 St_2 更換成 St_3 ，進行Heron's formula判斷，直到更換至 St_{10} 仍無法包圍住觀測點，則改成固定 St_0 與 St_2 ，將 St_1 更換爲 St_3 ，再進行判定，同樣最多只更換至 St_{10} ，仍未將觀測點圍住表示此觀測點位於整體氣象站的外圍。

3-3 推估數據

針對這觀測點所在的位置，內圍或外圍這兩種不同的情況，我們使用兩種不同的演算法進行數據的推估。在內圍的觀測點必須找到三座氣象站將其包圍後，使用線性內差計算數據；而在外圍的觀測點則是搜尋一座最近的氣象站，直接使用該點的氣象資料作爲估測值。

Case 1: 氣象站內圍

使用線性內插法估測氣象值。假設三座氣象站分別爲A、B、C三點，觀測點爲P，則 $\overline{AP}$ 交 $\overline{BC}$ 於D 點。欲求P點的氣象資訊 P_{data} ，首先利用B、C、D三點的座標分別求出其距離 $\overline{BD}$ ，、 $\overline{DC}$ 與 $\overline{BC}$ ，而D點的氣象數據 D_{data} 可由B、C點的氣象資訊 B_{data} 、 C_{data} ,使用線性內插求出：

$$D_{data} = \frac{B_{data} * \overline{DC} + C_{data} * \overline{BD}}{\overline{BC}} \quad (1)$$

同樣地，可利用 A、P、D三點的座標求出其距離 $\overline{AP}$ 、 $\overline{PD}$ 與 $\overline{AD}$ ，再使用一次線性內插，以A點的氣象資訊 A_{data} 與 D_{data} 求出P 點的氣象數據 P_{data} ：

$$P_{data} = \frac{D_{data} * \overline{AP} + A_{data} * \overline{DP}}{\overline{AD}} \quad (2)$$

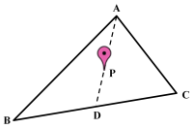

圖1 線性內差法估測p點之氣象資料

Case 2: 氣象站外圍

位於外圍的觀測點，無法找到三座將其圍住的氣象站，故不適用於線性內插估測。爲簡化系統設計，直接利用最靠近觀測點之氣象站資料進行高度校準推估計算。

3-4 溫度與氣壓

在大氣學的理論上，平均高度每上升100公尺，溫度會下降0.65℃[19]，在估測的實作上，依高度校準溫度值，進行完數學推估後，同樣依高度校準估測溫度值。第一步：從三座氣象觀測站中，找出高度最高的氣象站。第二步：將另外兩氣象站的溫度，調整爲與最高氣象站同一高度下的溫度值，進行估測。

第三步：將估測後的數據也進行高度校準，將觀測點與最高氣象站的高度差調整溫度資料。

氣壓方面，大氣密度的變動很大，密的地方高，疏的地方低。因爲熱脹冷縮的定理，空氣是流體，若是受熱，空氣就膨脹，密度就減小，氣壓降低；反之，遇冷空氣收縮，密度就增大，氣壓升高。因此通常高氣壓地區，氣溫較低。空氣也會隨高度升高而變稀，大氣密度銳減，因此高山的氣壓比地面上低。雖然氣壓隨高度而遞減的遞減率並非固定不變，但平均可歸納爲每上升8.5公尺，氣壓便下降一個百帕(hPa)。

因爲氣壓與溫度類似，隨地形變動，進行數學推導前，須經過高度校準，推估後的數據，在實際採納前，同樣地經過校準。實作的第一步的演算法與溫度相同，從三座氣象觀測站中，找出高度最高的氣象站。第二步是將另外兩氣象站的氣壓，調整爲與最高氣象站同一高度下的氣壓值，再進行估測，估測的數據也必須進行校準，根據觀測點與最高氣象站的高度差，調整氣壓值。

3-5 濕度與紫外線強度

濕度與紫外線強度之估測過程與「溫度與氣壓」相同，也利用內插法，只是沒有用高度進行校準。

3-6 風向與風速之估測

向量是指一個具有大小與方向的幾何現象。風的特性與向量類似，若將風速視爲向量大小；風向則視爲方向，就能轉變爲向量，並以數學方法進行計算與估測。

假設A、B、C三座氣象站的風速與風向轉換成向量後分別爲： $A(v_a, w_a)$ 、 $B(v_b, w_b)$ 、 $C(v_c, w_c)$ 。利用這3個觀測站估測P點的風向與風速 $\vec{U}(v,w)$ ，如圖2所示。

P點風強度與方向的估測方式與溫度的估測方式相似；皆使用線性內插法估算，兩者差別只在於溫度估算的單位爲純量，而風向估算的單位則爲向量。在計算出觀測點P的風向後，便可依據此向量 $\vec{U}(v,w)$ ，計算出風的強度 $\overline{U}$ 。

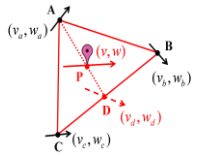

圖 2 風向與風速之估測

所先利用(3)與(8)求出風向 $\vec{U}(v,w)$ ：

$$D(v_d, w_d) = \frac{\overline{CD} \times B(v_b, w_b) + \overline{BD} \times C(v_c, w_c)}{\overline{CB}} \qquad (3)$$

$$\vec{U}(v,w) = \frac{\overline{DP} \times A(v_a, w_a) + \overline{AP} \times D(v_d, w_d)}{\overline{AD}} \qquad (4)$$

接者，利用(5)求出風速 $\overline{U}$ ：

$$\overline{U} = \sqrt{v^2 + w^2} \qquad (5)$$

四， 研究結果

我們在Google應用服務引擎申請免費帳號，建立應用應用程式，驗證本系統的理論與方法，使用者只要使用Browser皆可開啓本系統。以下是已架設爲氣象估測系統網址：http://meteorological-estimating.appspot.com/map/

本系統能夠定時的更新全台氣象站資訊，並根據使用的所指定的地點，找尋其最近的三座氣象站，進行氣象資訊的推估，再將推估出的氣象資訊分類成使用者能輕易了解的圖像資訊。

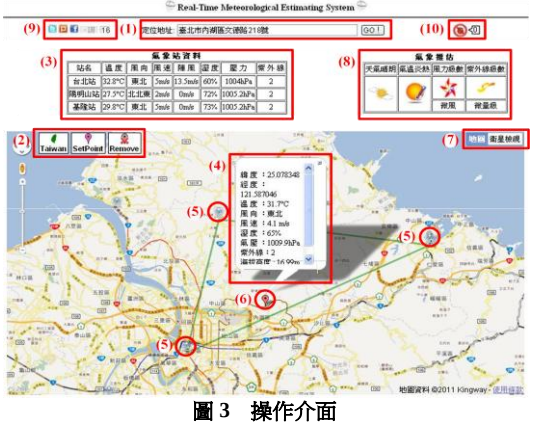

圖 3 操作介面

(1) 定位地址輸入欄
(2) Google Map Control
Taiwan:將視野調回到整個臺灣
SetPoint : 放置定位標誌(可拖曳)
Remove : 清空Map上所有標記
(3) 最近的三座氣象站資訊
(4) 推估出的氣象資訊
(5) 氣象站標記
(6) 觀測點標記
(7) 視圖選擇鈕
(8) 氣象資訊分類圖
(9) 交友社群分享鈕
(10) Bug回報系統

本系統主要的功能是進行氣象資訊推估，因此推估出數據的精確度非常重要，所以研究最後進行了一連串的測試，針對各個氣象因子的推測結果，分析系統的準確度。由於建立一個氣象觀測點的成本很高，且維護非常地不容易，所以在實測上很難自行測量氣象資訊並與系統估測出的氣象數據做比對，因此將以氣象站估測氣象站的方式，測試系統的推估精準度。

整個臺北縣市共有六座氣象觀測站，占臺灣的百分之十六，且每個氣象站的資訊充足、故障率低，因此，本研究以臺北地區作爲系統測試的環境。在此條件下，影響氣象數據的不定因素少，同質性的地形與區域條件多，能夠測出本系統所使用演算法的準確性。

測試方法是利用板橋、新店與陽明山等3個氣象觀測站，推估臺北氣象站的氣象數據，再與該氣象觀測站測得的資料做比對，判斷系統估測出的數據之準確性，如圖5～10所示。我們估測6個不同氣象因子：溫度、風向、風速、濕度、氣壓與紫外線強度，並做爲本系統所提供的估測數據，是否具有參考價值之標準，測試日期爲2010年10月21日到10月27日，估測結果如圖4所示。

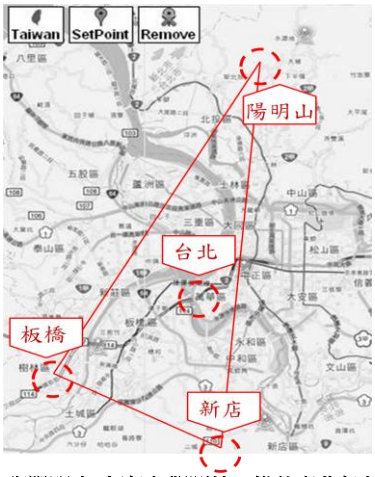

圖4 板橋、新店與陽明山3個氣象觀測站，推估臺北氣象站的氣象數據

溫度估測測試結果，如圖5所示，準確度相當高，平均誤差約0.5℃，而兩次的測試數據有一個共同的特徵，就是推估溫度比實際溫度略低，利用此特性，若求出一個誤差參數，套用於同類型地區，就能提高準確度。

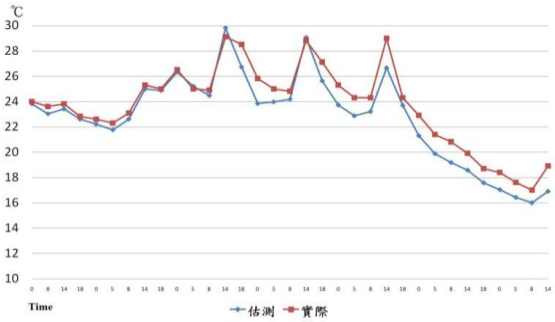

圖 5 2010 年 10.21~10.27 溫度觀測值

風向與風速的誤差則較高，如圖6與7所示，這是由於影響風的因素較多，難以全部考慮，但大體上波型浮動都很類似，往後在風的估測上還能更深入研究。

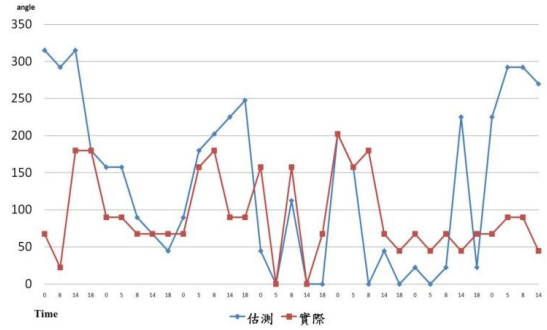

圖 6 2010 年 10.21~10.27 風向觀測值

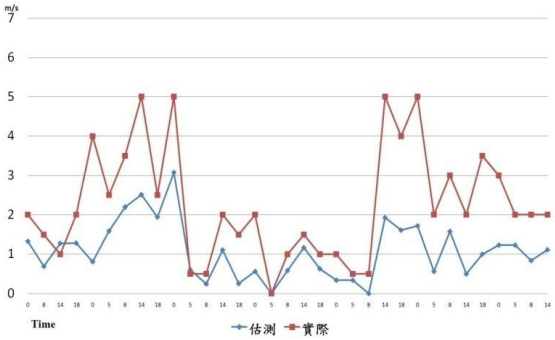

圖 7 2010 年 10.21~10.27 風速觀測值

濕度的準確率也很高，如圖8所示。從結果來看，平均誤差不到百分之十。

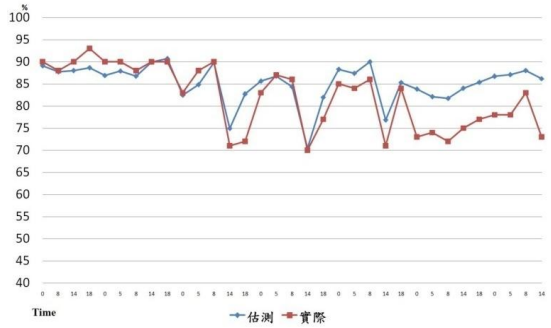

圖 8 2010 年 10.21~10.27 濕度觀測值

氣壓的平均誤差值更是低於2hPa，如圖9所示，可以是準確度最高的一項結果，但是估測值很明顯地都比實際值要高，也是有待改進的一部分。

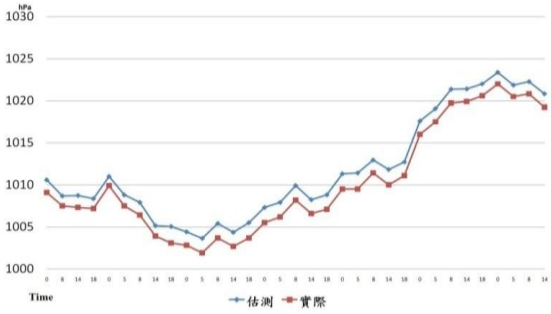

圖 9 2010 年 10.21~10.27 氣壓觀測值

最後的紫外線測試誤差指數也不大，如圖10所示。

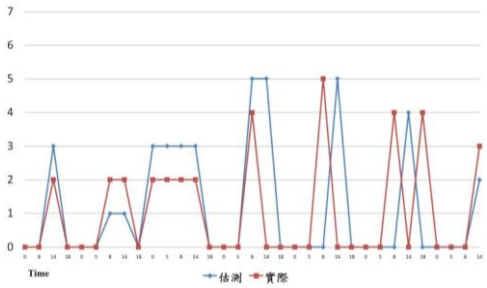

圖 10 2010 年 10.21~10.27 紫外線觀測值

伍、 結論

整體而言，本系統改變了傳統的氣象呈現模式，以使用者爲觀點，提供使用者希望的氣象資訊，更能實際應用於日常生活中。例如：出門前可以先參考本系統估測目的地的氣象資訊，做好準備，確保不會使用不夠及時的氣象資訊而措手不及。

在目前傳統的氣象資訊模式下，本研究提供了另一種模式的氣象資訊，能提供及時的氣象資訊，也能指定精確的地點。實作上使用雲端運算技術處理氣象數據，並結合Google Maps與互動網頁作爲介面，可快速地回應使用者選定的地點，提供具參考價值的及時氣象數據。

從上述的估測結果而言，本系統仍有很大的進步空間，尤其對風速與風向氣象因子，仍有深入的研究探討空間。目前本系統僅提供中華民國領土內的氣象估測服務，往後如能將此系統，根據某地區的地形與氣象特性進行修改，也能成爲某國或某特定地區所專屬的氣象估測系統。除了氣象估測，在使用介面的互動上突破了傳統單向傳遞氣象資訊的模式，如能與現今常見的媒體平台結合，此模式未來或許有機會成爲新型的氣象資訊展示模式。

六、 參考資料

[1] 黃耀德，”整合Google Map 之即時氣象量測系統”，私立淡江大學資訊工程系碩士班論文，未出版，2010。

[2] 蔡鈞華，”雲端運算SOA 服務平台架構與運作機制之研究”，國立政治大學資訊管理研究所碩士論文，未出版，2009。

[3] 中央氣象局，URL: "www.cwb.gov.tw/"

[4] George Lawton ,” Developing Software Online with Platform-as-a-Service Technology” ,IEEE computer Society ,june 2008.

[5] Weather Stations URL:“www.cwb.gov.tw/V6/eservice/docs/overview/organ/stations/”

[6] Steven C. Chapra, “Applied Numerical Methods Second Edition”, Mc Graw Hill 2008

[7] Stephen H. Friedberg, Arnold J. Insel, Lawrence E. Spence, “Linear Algebra(Fourth Edition)”, Pearson Education International 2003

[8] Byers, Horace Robert, “General meteorology”, McGraw-Hill 1959

[9] Cole, Franklyn W., “Introduction to meteorology”, Wiley 1975

[10] Gartner Say's Cloud Computing Will Be As Influential As E-business, URL: “gartner.com/it/page.jsp?id=707508”

[11] Google Map API V3 ， URL:“code.google.com/intl/zh-W/apis/maps/documentation/javascript/”

[12] AJAX Wiki URL: “zh.wikipedia.org/zh-tw/Ajax”

[13] Heron's formula Wiki URL: “en.wikipedia.org/wiki/Heron's_formula”

[14] Alexander Power , “The Django Form Validation Framework on Google App Engine”, Google App Engine ,April, 2008