

中原大學電機資訊學院電機工程學系  
碩士論文  
Master Thesis

程式解題系統的即時進度偵測與分析  
Instant progress detection and analysis of the program  
problem solving system

巫鈺瑩  
Yu-ying Wu

指導教授：王佳盈教授  
Advisor: Jan-Ying Wang, Ph.D.

中華民國 96 年 6 月  
June, 2019

# 中原大學碩士學位論文

## 口試委員會審定書

### 程式解題系統的即時進度偵測與分析

### Instant progress detection and analysis of the

### program problem solving system

本論文係巫鈺瑩君 (10678002) 在中原大學電機工程學系完成之碩士學位論文，於民國 96 年 6 月 28 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

<hr/>	
<hr/>	<hr/>
<hr/>	<hr/>
<hr/>	<hr/>
<hr/>	<hr/>

所 長：

<hr/>
-------

## 誌謝

在中原大學就讀研究所並且可以以電機碩士畢業是我的一大榮幸，感謝在我求學並撰寫本論文所支持、鼓勵我的人，因有他們，我才能完成這個重要的里程碑，使往後的人生更有自我的夢想與目標努力。首先感謝指導老師王佳盈老師，老師不只是在專業方面的能力非常高，並且對人生有很大的體悟，我們在此不只是對於專業領域的學習，老師更教導我們以不同眼光看世界，與他人相處技巧，或者分享每個人對於生活的發想。老師也是一個非常有同情心與愛心的人，有一門課程為服務學習，是到有身心障礙孩子的幼兒園提供技術協助，此門課就是老師所開設。以自己的專業知識幫助一些生活不便的人這一精神，讓我十分的尊敬。再者感謝 501 實驗室的學姊，同學以及一起奮鬥的學弟學妹。因為老師接了一些計畫，召集研究生們與大學部的學弟們一起進行，大家在共事中變得更加親密外，也會討論自己的將來與互相鼓勵。最後感謝偉大的父母親，在大學畢業時並不會催促的叫我出去找工作，而是叫我好好想想接下來該做什麼。於是我選擇了繼續升學唸研所，而父母也表示支持，這兩年讓我無顧慮的完成碩士學位。

# 摘要

本論文提出一線上監測網頁並分析其數據。本研究的目標，是希望能夠觀察學生在做答的狀況，不是繳交的結果而是在答題的過程。藉此研究程序分析過程的數據並且統整。我們希望提供的演算法未來能結合完整的寫作平台並且抓取實際的數據。

關鍵字：webapp,html,css,javascript,vue.js,python,colab, keywordszh



中 原 大 學

# Abstract

This paper proposes an online monitoring page and analyzes its data. The goal of this study is to be able to observe the status of the student's answer, not the result of the payment but the process of answering the question. By this, the program analyzes the data of the process and integrates it. We hope to provide an algorithm that combines a complete writing platform and captures actual data in the future.

**Keywords:**webapp,html,css,javascript,vue.js,python,colab

中 原 大 學

# Contents

口試委員會審定書	ii
誌謝	iii
摘要	iv
Abstract	v
<b>1 緒論</b>	<b>1</b>
1.1 研究背景	1
1.2 研究動機與目的	2
1.3 研究流程規劃	2
1.4 章節概要	4
<b>2 背景技術介紹</b>	<b>5</b>
2.1 web app	5
2.2 HTML 與 CSS	6
2.3 javascript	6
2.3.1 jquery	7
2.3.2 vue.js	7
2.4 Firebase	8
2.5 python	8
2.5.1 機器學習相關	8
2.5.2 Colab	9

<b>3</b>	<b>系統架構與規劃</b>	<b>10</b>
3.1	前端頁面的即時監測 . . . . .	10
3.1.1	模擬平台 . . . . .	11
3.1.2	基本統計資訊 . . . . .	12
3.1.3	輸出 . . . . .	13
3.2	基礎數據分析 . . . . .	14
3.2.1	模擬測試數據 . . . . .	14
3.2.2	數據分析 . . . . .	18
<b>4</b>	<b>實驗與執行結果</b>	<b>26</b>
4.1	網頁平台數據監測 . . . . .	26
4.2	數據的統計分析 . . . . .	32
4.3	結果討論 . . . . .	37
4.3.1	數據即時監控部份 . . . . .	37
4.3.2	數據分析部分 . . . . .	37
<b>5</b>	<b>結果討論與未來展望</b>	<b>38</b>
	<b>Bibliography</b>	<b>39</b>

中 原 大 學

# List of Figures

3.1	從 JS、HTML 以及 Vue 所打造的前端網頁中擷取資料 . . . . .	11
3.2	模擬平台介面 . . . . .	11
3.3	作答統計資訊範例 . . . . .	13
3.4	分析系統架構 . . . . .	14
3.5	執行結果漸增圖 . . . . .	15
3.6	網頁程式執行結果圖 . . . . .	15
3.7	作答字數的模擬改良 . . . . .	17
3.8	作答行數的數據模擬 . . . . .	18
3.9	整體結果 . . . . .	18
3.10	整體速度結果 . . . . .	19
3.11	示例圖片 . . . . .	19
3.12	trend 陣列輸出 . . . . .	20
3.13	trend 點狀圖 . . . . .	21
3.14	trend 圓餅圖 . . . . .	21
3.15	trend2 輸出 . . . . .	22
3.16	trend2 條碼狀方圖 . . . . .	23
3.17	trend2 分區圖 . . . . .	23
3.18	多組數據 . . . . .	25
4.1	網頁顯示頁面 . . . . .	27
4.2	簡短程式碼輸出 . . . . .	27
4.3	console 的訊息 . . . . .	28
4.4	複製貼上的輸出 . . . . .	29



4.5	模擬作答啟始畫面 . . . . .	29
4.6	模擬作答的過程畫面 . . . . .	30
4.7	螢幕錄影與 makevideo 比較 . . . . .	31
4.8	python 抓下的文字 . . . . .	31
4.9	圖形組 . . . . .	33
4.10	10 組折線圖 . . . . .	34
4.11	10 組結果表格圖 . . . . .	35



中原大學

# List of Tables

4.1 30 組數據 . . . . .	36
----------------------	----



中 原 大 學

# Listings

3.1	js 字數與行數的判讀與計算 . . . . .	12
3.2	時間欄位的模擬數據 . . . . .	16
3.3	作答字數的模擬數據 . . . . .	16
3.4	python 數據 trend2 . . . . .	22



中 原 大 學

# Chapter 1

## 緒論

### 1.1 研究背景

近年因為圍棋 AI—AlphaGo，戰勝了人類職業選手，吸引越來越多人投入到人工智慧這塊領域。人工智慧這個主題其實已經有了很多年的歷史，但是直到最近這幾年才逐漸浮出檯面並受到重視，除了硬體及各項技術的進展，主要也因為人工智慧需要大量的數據做「訓練」。對早期研究者來說，想要獲得不錯效果的最小量測試，都遠遠超過當時的計算能力以及可以取得的數據資料量，也就是要取得大量的測試數據非常不容易。但最近幾年，由於網際網路和各項技術的進展，越來越多大型的資料庫出現，並且網路上有各式各樣的數據可以取得，許多研究人員重新挖掘神經網路的價值，這其實也是透過「大數據」使得架構的模型和測試都更有效率。[1]

在另一方面，由於手機和網際網路的快速發展，一種新的網頁應用技術稱為 Progressive Web Apps (PWA) 也應運而生。目前大部份使用於手機或電腦的應用程式，都要先透過安裝才可以使用，但是 PWA 只要能進行網頁瀏覽就可以使用，完全不需要進行軟體安裝，而使用起來的感覺與一般應用程式幾乎沒有什麼區別。PWA 可以說是結合了網頁和應用程式的使用者體驗，可以直接在使用者的裝置上瀏覽和執行，不需要透過應用程式商店下載和安裝。PWA 應用不僅可以提供精彩的全螢幕體驗，甚至可以透過網頁推播通知，吸引使用者繼續參與互動 [3]。

PWA 的興起與網頁前端技術的快速發展有密切關係，前端網頁使用 HTML、XHTML、HTML5、CSS、JavaScript 等網頁標準技術製作，與後端伺服器所使用

的 PHP、ASP.NET、JSP、RoR 等程式語言有很大的不同。前端網頁的技術發展很快，可以用來製作各種 PWA 應用，目前最常見的三大架構為 AngularJS、React 和 VueJS。使用前端網頁技術，可以快速開發網頁的整體面貌，並可以利用非同步方式存取網頁所要呈現的內容，而開發過程隨時修改，隨時都可以看到更新的畫面結果，可以加速網頁應用的製作開發過程。[2]

## 1.2 研究動機與目的

近幾年我的指導教授所開授的程式課程，大多使用線上評測系統「瘋狂程設」[4]做為輔助教學平台。「瘋狂程設」主要是謝育平教授與其學生所創作的程式教學網站。[5]。這個系統可以供教師開課，讓修課學生上線練習解題。系統具有多項功能，在教學講解、作業、以及考試等各方面都提供了相當不錯的功能。另外在評測方面，可以即時提供同學編譯上的錯誤，以及執行結果與正確答案之間的差異，可以說在教學上提供了很多的幫助。雖然此評測系統十分優異，但長期使用之後，仍然會感受到一些限制和不足之處。假如這個系統可以具備一些人工智能，例如自動分析同學常犯的錯誤，評估同學學習的狀況等，那麼這個系統就可以變得更為強大，對教學應該也可以提供更多的幫助。

因此我們著手進行這個研究，希望從作答過程一直到評測結果，可以透過系統自動蒐集一些數據，並可以用這些數據做進一步的分析，未來在這個基礎之上，就有機會可以打造一個融入人工智能的程式解題平台。由於瘋狂程設是別人的作品，而且沒有提供其原始碼，因此我們先在一個雛形平台上進行研究，這個雛形平台目前由指導教師和另一位同學周身鴻進行研究開發，在此也一併表達感謝。我們希望在這個平台上，新增一個即時監控的子系統，可以即時監控同學的作答過程，並進行一些數據的統計和分析，並且希望透過這些取得的數據，可以模擬同學作答的過程。未來希望在這個基礎之上，可以增加更多的人工智慧，讓這個系統變得更加強大。

## 1.3 研究流程規劃

以下是本論文的研究構想以及流程規劃：

- 研究構想

1. 即時線上監測：測試的平台主要以網頁做為最主要和使用者互動的介面，其頁面設計採用網頁前端技術製作，我們希望利用相關的技術擷取同學作答過程的取樣資料，並能將這些資料存放到雲端的資料庫中，作為未來進一步分析的基礎。
2. 基礎數據分析：透過擷取到的抽樣資料，希望可以分析同學作答過程的一些統計資訊，也希望可以即時模擬同學作答的畫面，以及由抽樣數據產出模擬同學作答過程的影片。

- 流程規劃

1. 訂定研究主題
2. 決定研究目的與範圍
3. 背景技術討論與資料蒐集
4. 選擇開發環境
5. 實驗兩部分程式並且做出結果表格
6. 結果分析與討論
7. 結論與未來發展



中原大學

## 1.4 章節概要

- 第一章
  - 論文緒論
  - 研究背景、動機與目的及章節概述
- 第二章
  - 背景技術介紹
  - 說明研究所需之背景知識
- 第三章
  - 系統架構與規劃
  - 介紹整個系統架構與所使用之語法邏輯解說
  - 系統前半 js 與 html 之偵測
  - 系統後半分析部分
- 第四章
  - 系統細部與執行結果
  - 多加解釋細部活動
  - 分析部分之結果呈現
- 第五章
  - 結果與未來展望
  - 說明未來方向與檢討

# Chapter 2

## 背景技術介紹

本章對論文所使用到的背景技術做一些基本的介紹，以下分成幾個部份來做說明。2.1 節介紹網頁 app 定義與其對未來發展影響；2.2 節介紹所使用背景技術 html；2.3 節介紹 JavaScript、jQuery 以及開發框架 VueJS；2.4 節介紹 Firebase；2.5 節介紹 Python 語言與 Colab 執行環境。

### 2.1 web app

此小節介紹 Web App 與 Progressive Web App (PWA)。

- 何謂 Web App

就是網路應用程式 (web application)，網路應用程式風行的原因之一，是因為可以直接在各種電腦平台上執行，不需要事先安裝或定期更新等程式。此種類型的動態網頁與「網路應用程式」之間的區別一般是模糊的。最有可能接近「網路應用程式」的網站是與桌面軟體應用程式或行動應用程式具有類似功能的網站。HTML5 引入了明確的支援，使得應用程式可以作為網頁載入，可以在本地儲存資料並在離線狀態下繼續執行。[8]

- 何謂 Progressive Web App (PWA)

Progressive Web App 是希望能夠讓 Web application 盡可能的在各種環境（網路環境、手機作業系統等）下都能順暢且不減功能性的運作，並讓你的 Web App 可以：



1. 可以直接被使用者安裝至桌面執行
2. offline 使用
3. 擁有推送訊息功能
4. 開啟時看不到 URL Bar (類 Native app 的使用經驗)
5. 開啟時有 Splash Screen [7]

## 2.2 HTML 與 CSS

- HTML5

HTML5 是 HTML 最新的修訂版本，由全球資訊網協會 (W3C) 於 2014 年 10 月完成標準制定。廣義論及 HTML5 時，實際指的是包括 HTML、CSS 和 JavaScript 在內的一套技術組合。它希望能夠減少網頁瀏覽器對於需要外掛程式的豐富性網路應用服務。[9]

- CSS

層疊樣式表 (Cascading Style Sheets，縮寫：CSS；又稱串樣式列表、級聯樣式表、串接樣式表、階層式樣式表) 是一種用來為結構化文件 (例如 HTML 文件或 XML 應用) 添加樣式 (字型、間距和顏色等) 的電腦語言，由 W3C 定義和維護。CSS 不能單獨使用，必須與 HTML 或 XML 一起協同工作，為 HTML 或 XML 起裝飾作用。[10]

## 2.3 javascript

JavaScript (通常縮寫為 JS) 為一種進階的、直譯的程式語言。JavaScript 的原始碼在發往用戶端執行之前不需經過編譯，而是將文字格式的字元程式碼發送給瀏覽器由瀏覽器解釋執行。在用戶端，JavaScript 在傳統意義上被實作為一種解釋語言，但在最近，它已經可以被即時編譯執行。隨著最新的 HTML5 和 CSS3 語言標準的推行它還可用於遊戲、桌面和行動應用程式的開發和在伺服器端網路環境執行，如 Node.js。[11]

### 2.3.1 jquery

jQuery 是一套跨瀏覽器的 JavaScript 函式庫，簡化 HTML 與 JavaScript 之間的操作。jQuery 是開源軟體，使用 MIT 授權條款授權。jQuery 的語法設計使得許多操作變得容易，如操作文件（document）、選擇文件物件模型（DOM）元素、建立動畫效果、處理事件、以及開發 Ajax 程式。jQuery 也提供了給開發人員在其上建立外掛程式的能力。這使開發人員可以對底層互動與動畫、進階效果和進階主題化的元件進行抽象化。模組化的方式使 jQuery 函式庫能夠建立功能強大的動態網頁以及網路應用程式。[14][15]

### 2.3.2 vue.js

Vue (讀音 /vju:/，類似於 view) 是一套用於構建用戶界面的漸進式框架。Vue 的核心庫只關注視圖層，不僅易於上手，還便於與第三方庫或既有項目整合。另一方面，當與現代化的工具鏈以及各種支持類庫結合使用時，Vue 也完全能夠為複雜的單頁應用提供驅動。[12]

- 元件

元件是 Vue 最為強大的特性之一。為了更好地管理一個大型的應用程式，往往需要將應用切割為小而獨立、具有復用性的元件。在 Vue 中，元件是基礎 HTML 元素的拓展，可方便地自訂其資料與行為。

- 模板 Vue 使用基於 HTML 的模板語法，允許開發者將 DOM 元素與底層 Vue 實體中的資料相繫結。所有 Vue 的模板都是合法的 HTML，所以能被遵循規範的瀏覽器和 HTML 解析器解析。在底層的實現上，Vue 將模板編譯成虛擬 DOM 彩現函式。結合回應式系統，在應用狀態改變時，Vue 能夠智慧型地計算出重新彩現元件的最小代價並應用到 DOM 操作上。[13] 此外還有許多功能不一一贅述。

## 2.4 Firebase

Firebase 是一個雲端開發平台支援多種 os，可協助 app 開發快速建立後端的服務並提供即時的資料。[16] Firebase 有幾項特別強大的功能：

### 1. 事件紀錄無上限：

Firebase 有 500 種預設的事件類型，而且紀錄總量無上限。並且使用 SDK 則可以自動收集事件。

### 2. 支援原始資料自動匯出

在大量資料分析處理 Firebase 有支援自動匯出功能，可讓企業針對資料執行進行 SQL 分析查詢。

### 3. 直接行動的分析工具

Firebase 具有區隔使用者的功能，藉由案裝置、事件、或是資源 (如年齡、性別、語言) 的分類特定區隔，如此便可以更精確投遞廣告。[17]

## 2.5 python

因 python 這項語言包含太多領域，故本節介紹的部分偏向分析、人工智慧與機器學習。

### 2.5.1 機器學習相關

- 人工神經網路

人工神經網路（英語：Artificial Neural Network，ANN），簡稱神經網路（Neural Network，NN）或類神經網路，在機器學習和認知科學領域，是一種模仿生物神經網路（動物的中樞神經系統，特別是大腦）的結構和功能的數學模型或計算模型，用於對函式進行估計或近似。神經網路由大量的人工神經元聯結進行計算。大多數情況下人工神經網路能在外界資訊的基礎上改變內部結構，是一種自適應系統，通俗的講就是具備學習功能。[18]

- TensorFlow

TensorFlow 是一個用於機器學習的端到端開源平台。使初學者和專家可以

輕鬆創建機器學習模型。[19] TensorFlow 的底層核心引擎由 C 實現，通過 gRPC 實現網路互訪、分散式執行。雖然它的 Python/C/Java API 共用了大部分執行程式碼，但是有關於反向傳播梯度計算的部分需要在不同語言單獨實現。目前只有 Python API 較為豐富的實現了反向傳播部分。所以大多數人使用 Python 進行模型訓練，但是可以選擇使用其它語言進行線上推理。[20]

## 2.5.2 Colab

Colaboratory 是一個免費的 Jupyter 筆記本環境，不需要進行任何設置就可以使用，並且完全在雲端運行。借助 Colaboratory，在此筆記本上可以編寫和執行代碼、保存和共享分析結果，以及利用強大的計算資源，所有這些都可通過瀏覽器免費使用。[21] Colab 是一個可使用瀏覽器線上編輯的雲端筆記本，不需在自己的主機建構環境，並且可以使用 Google 提供的 GPU 或是 TPU。



中 原 大 學

## Chapter 3

### 系統架構與規劃

本章介紹系統架構與規劃。



#### 3.1 前端頁面的即時監測

本研究使用指導教師和另一位同學建置的解題平台雛形，在使用者針對題目作答的過程中，可以擷取其作答過程的打字內容，並定時將使用者所打的文字內容與目前時間的數據傳到 Firebase 的資料庫當中。目前網頁前端主要使用 JS 和 HTML 以及 Vue 的相關技術 (圖 3.1)

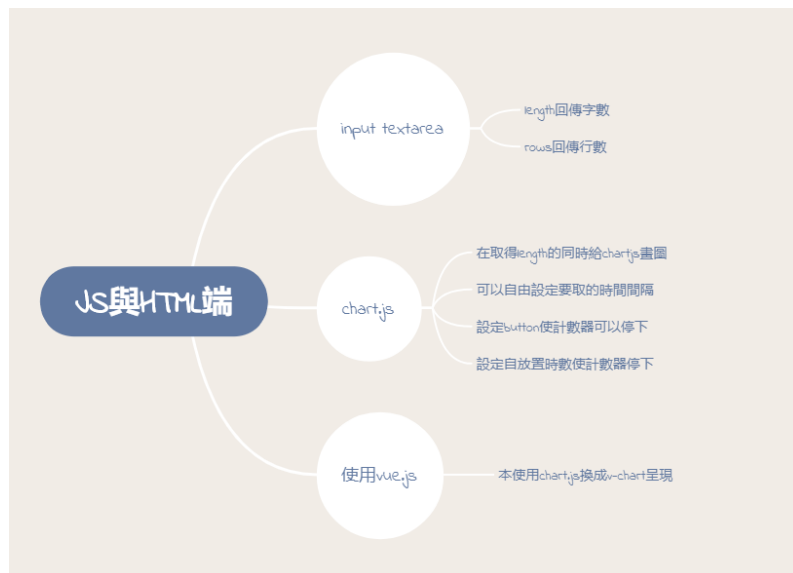


Figure 3.1: 從 JS、HTML 以及 Vue 所打造的前端網頁中擷取資料

### 3.1.1 模擬平台

- 目前所使用的解題平台雛形，其作答畫面如下圖所示。

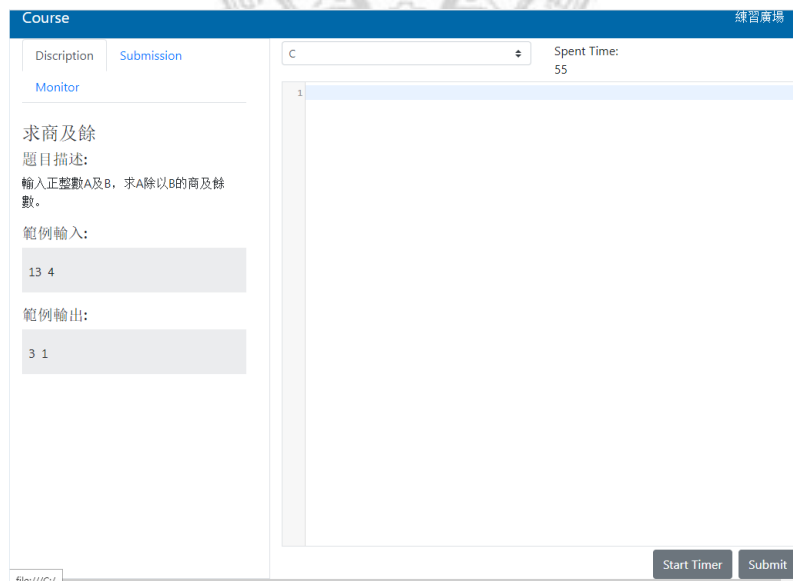


Figure 3.2: 模擬平台介面

- 功能:

1. 左方：顯示使用者正在作答的題目，包括輸入輸出的範例，供使用者

了解目前作答的題目。使用者投稿之後，系統會自動切換至 Submission 頁籤，這個頁籤是評判伺服器進行評測之後所回傳的結果，包括答題正確狀況以及程式碼的格式檢查結果。

2. 右上計算時間讀秒：右上的計時器記錄使用者所耗費的作答時間，每隔一定時間，系統會自動擷取使用者所輸入的內容，連同當時耗費的時間傳送到 Firebase 資料庫中。
3. 右下的 Stop Timer：暫停計時和傳送資料，目前做為除錯用途。

### 3.1.2 基本統計資訊

- 字數與行數的判讀與計算

取得使用者輸入的程式碼內容後，使用以下 JS 指令計算總共輸入的字數和行數：

```
1 // code = 使用者輸入的程式碼內容
2 chars = code.length;
3 lines = 1+code.match(/\r?\n|\r/g).length; // 使用正規表達式尋找換行符號
```

Listing 3.1: js 字數與行數的判讀與計算

此處用來判斷行數據的方法，主要是偵測使用者所用到的換行符號。如果使用者的程式碼文字中有其他的換行文字，可能會造成誤差。這個部份，未來可以使用鍵盤偵測來進行改良。計算出來的統計數字會使用 HTML 頁面顯示出來。

- 時間的判讀與計算

目前用來擷取資料的方法主要是透過 JS 的 setInterval 函式，setInterval 函式可以每隔一個設定時間不斷重覆執行特定的任務，在這裡主要是擷取使用者的作答內容。[22] 應該注意的一點，當作答結束之後，或者停止擷取資料之後，也要把 setInterval 所建立的程序清除掉，否則它會一直留在網頁中不斷執行，有可能會造成系統的問題。擷取到的資料被傳送到 Firebase 中，可以用來做接下來的數據分析。

### 3.1.3 輸出

- 即時監測圖：

使用者的輸入內容和時間傳送到 firebase 的資料庫之後，會自動傳送收據更新的訊息到監控程式，監控程式的目的是顯示同學作答的一些統計資訊。目前數據的監測先選定一位同學為主，同學作答的時間，與輸入的字數和行數會使用圖形顯示出來。這邊使用了 chart.js 與 vue.js 來顯示前端的數據圖形。(圖 3.3)

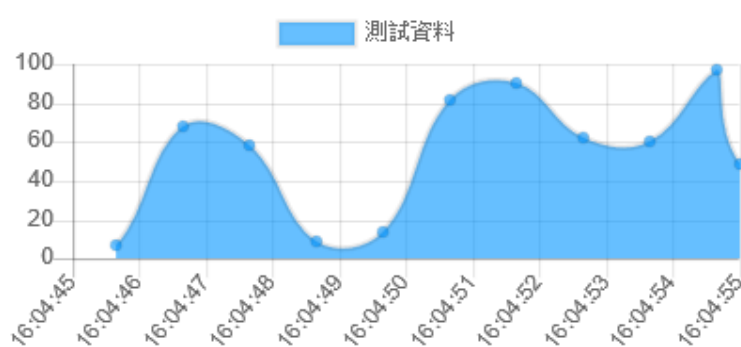


Figure 3.3: 作答統計資訊範例

呈現的統計圖每隔一段時間會隨著新的擷取資料而自動更新。資料擷取的時間間隔目前設定每秒一次，但可以隨著需要而做更改。圖表的 X 軸為時間，Y 軸為字數與行數。

- 影片輸出：

除了進行數據監測及圖形繪製，另外也可以模擬同學作答過程的畫面，這部份一樣是 Firebase 傳送的數據更新資料，以畫面顯示當時同學的畫面，進行即時的監控。另外也可以根據這些資料，自動生成模擬同學作答畫面的影片，供教師或研究人員參考。



## 3.2 基礎數據分析

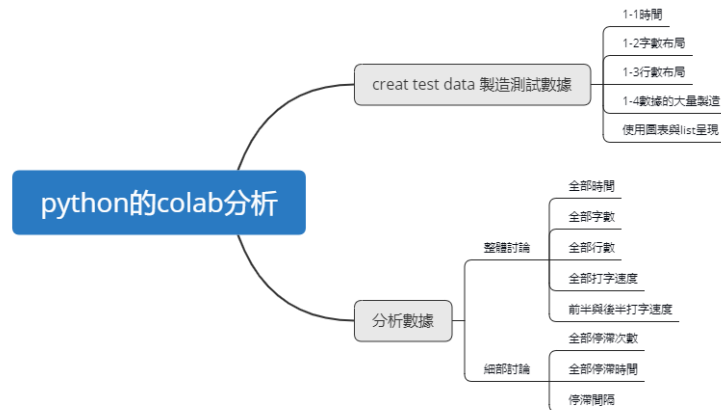


Figure 3.4: 分析系統架構

除了即時監測使用者的輸入狀況，我們也對收錄的資料做了一些基礎的統計分析，目前分析的內容包括：1. 個別同學做一題的總時間分析，以及全部同學的平均作答時間 2. 停頓分析：如果作答的內容一直沒有改變，表示這位同學可能在某處卡住了。可以先找出停頓的位置，未來可以據此做更進一步的分析處理。

### 3.2.1 模擬測試數據

討論數據的特徵

圖 (3.5) 是典型作答過程的統計數字，因為是打字數與時間的關係，所以數據自然是漸進式的。圖中佔據面積比較大的部份為字數統計，下方面積比較小的部份為行數。由於研究過程中，測試的作答平台尚未完整，因此除了採集一些個別少量的數據之外，先使用模擬產生的數據來進行分析，以觀察分析模組是否能正常運作。

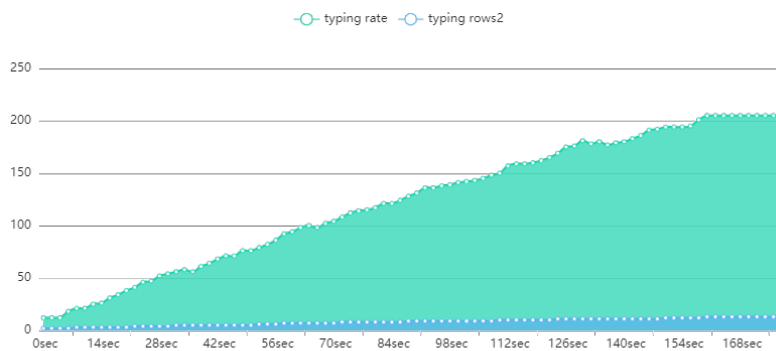


Figure 3.5: 執行結果漸增圖

在產生模擬數據之前，先討論一些作答數據的特徵和假設：1. 規定時間區間：一個班級同學做同一題目，大家所完成的時間會有差別，可以規定自做的數據時間 (ex:15 分 1 小時)，而 30 分鐘可能是大部分人完成的時間，15 分是做得快的，50 分以上是做得慢的。2. 作答字數：基本上整體分佈要像上圖一樣呈漸進增加的趨勢，但每位同學的字數增長速度可能不同，停頓的地方可能也不同，字數的增長也要設定一個合理的區間。3. 作答行數：理論上越多字的越多行，但還是要在合理的範圍，不能生成太多行數。

#### 規定格式

在統計圖 (3.5) 中包括三個欄位：時間、字數與行數。為了方便後續的分析，我們採用陣列的方式來儲存模擬數據。基本上模擬數據為三欄的陣列，其列數為模擬製造的記錄個數，以下分三部分說明各欄位數據的構成。

#### 時間欄位

由於網頁程式擷取作答內容的時間間隔是固定的，因此為時間欄位為固定間隔的時間序列。(圖 3.6)

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46,
'total rows= 173
'total time(sec)= 344
```

Figure 3.6: 網頁程式執行結果圖

每位同學在撰寫時所完成的時間也不相同，在產生數據時規定上下界，如同規定最快完成與最慢完成所需花費的時間。

```
1 data_up = 180
2 data_down = 500 #可以自行規定秒數區間
3 random_data = random.randint(data_up, data_down) #這是秒數
4 r1 = list(range(0, random_data, 2)) #r1是第一部分時間陣列的部分
5 print(r1)
6 r_row1 = len(r1)
7 r_row2 = r1[-1]
8 print("\n*total rows=", r_row1,)
9 print("*total time(sec)=", r_row2) #計算random的格數與秒數(秒數為最後一個數據)
```

Listing 3.2: 時間欄位的模擬數據

使用 random 函數進行最大秒數的選擇，後頭的第二與第三數據產生也大量地使用 random 函數。

#### 模擬字數

這個部份可以說是最重要和最複雜的部份，因為使用者作答過程可能會有非常多的變化。最初的構想的是使用者作答的字數每次取樣都會不停地增加，但每次增加的字數不同，先使用亂數來模擬增加的字數。

```
1 a = random.randint(0,10)
2 b = random.randint(a,20)
3 c = random.randint(b,30)
4 d = random.randint(c,40)
5 e = random.randint(d,50)
6 f = random.randint(e,60)
7 g = random.randint(f,70)
8 h = random.randint(g,80)
9 print(a,b,c,d,e,f,g,h)
```

Listing 3.3: 作答字數的模擬數據

如此，此程式所呈現的結果是 (8 13 21 34 38 59 64 67)，當然每執行一次都不同。但這樣一來，使用者作答的字數會無止盡的增加，但這樣的方式沒有辦法反

映使用者遇到不會的地方而卡住的情況，所以要加入”使用者在思考”的停頓狀況，也就是會要維持同樣的字數一段時間。

除了停頓之外，使用者也有可能打錯字而按下倒退 (Backspace) 鍵，另外也可能使用了選擇和刪除的功能，所以整體的數據並不會一路平滑的增加，必須在模擬數據中加上這些考量才比較合理。

於是我們新增三個模擬參數，分別用來處理增加、停頓、倒退和刪除的不同情況，先使用 random 函數決定使用者可能的三個動作，而在增加與減少的時候，數據的變化量是不均勻的，此部分一樣使用 random 函數來實現，如圖 3.7 所示。

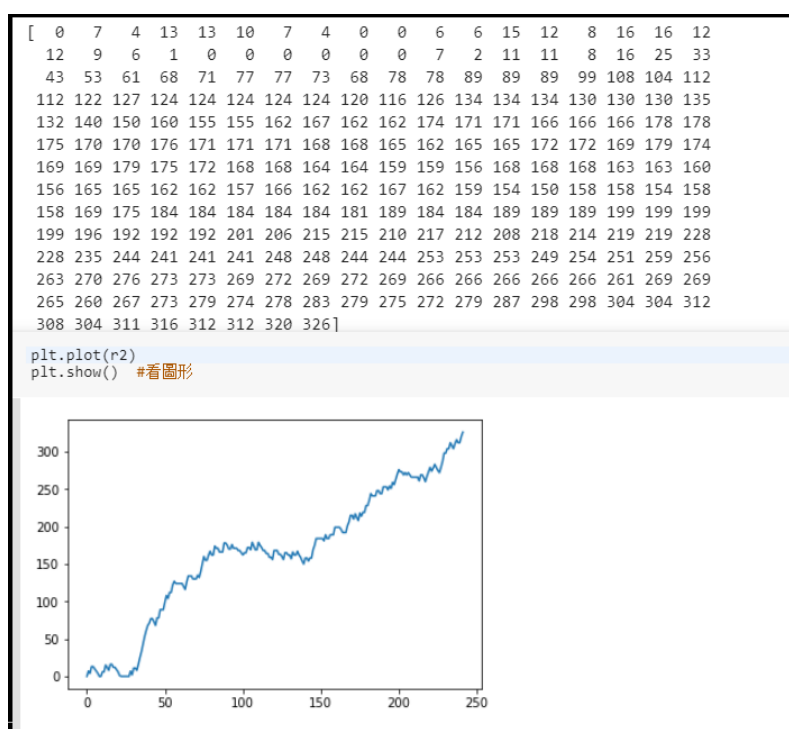


Figure 3.7: 作答字數的模擬改良

## 模擬行數

第三部分為作答行數，這部分與字數息息相關，但是換行的判斷比較簡單，一般只要觀察原始程式碼是否有換行符號即可得知。此處是製造出 0,0,0,1,1,1,2,2,2,3,3,3,3,3,3,4,4,4,..... 之類的漸增數列，但是增加的幅度遠比第二部分的字數還要緩慢許多。(圖 3.8)

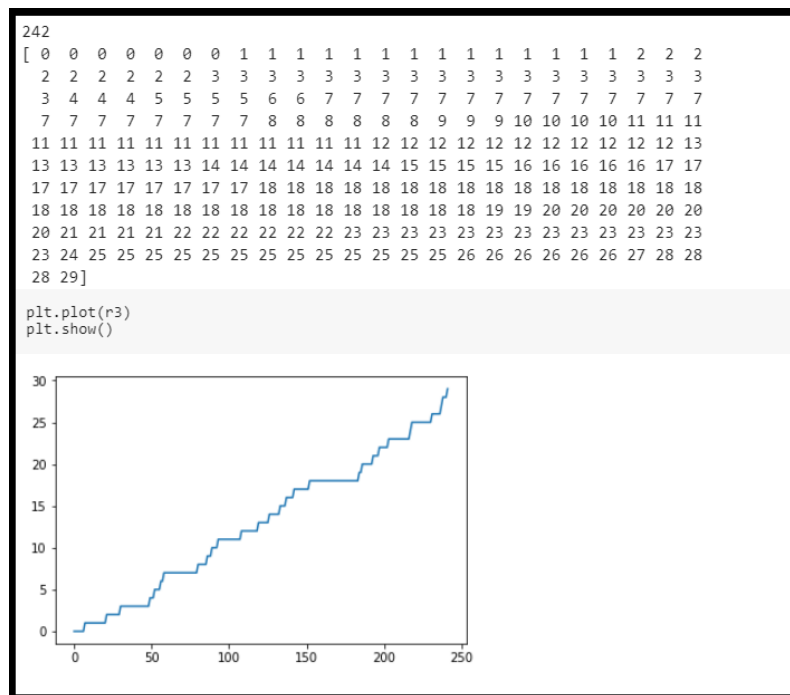


Figure 3.8: 作答行數的數據模擬

### 3.2.2 數據分析

#### 單組數據整體分析

- 可從最為直觀的計算所有數據的平均速度或是平均字數為基礎，在加上停頓點的分析。因為把數據設計成三欄的陣列，在分析步驟可以很容易抓到要的數據進行運算。
- 在陣列上最容易計算的如：作答時間、作答完畢字數、作答完畢 (圖 3.9)

```
作答時間:total sec= 288
作答完畢字數:total word count= 98
作答完畢行數:total word rows= 9
```

Figure 3.9: 整體結果

然後往下延伸計算的數據，如平均作答速度、前半作答速度、後半作答速度；在速度的計算上分兩部分是因為如果只有一個均速可能在比較上會有較大誤差，因此做多個速度的比較數據會更佳。(圖 3.10)



3. 後項減去前項為負: 打字字數減少, -1 表示

將判斷後 1,0,-1 的值存入 trend 陣列, 故此陣列為三個元素組成, 且比原本數據的陣列值少一, 因為 trend 陣列是原陣列間隔中的差分。(圖 3.12)

```
[ 0 13 8 18 15 12 18 13 13 13 16 16 13 8 14 22 17 26
26 23 18 18 15 15 15 18 18 18 13 10 10 10 10 10 7 16
16 24 20 17 17 24 24 20 29 25 20 30 30 30 27 31 42 53
50 58 68 68 63 74 84 90 85 85 95 103 107 104 104 104 109 104
104 100 96 108 105 105 113 120 127 132 132 129 126 122 122 119 116 122
118 113 109 109 109 109 119 127 133 138 138 148 145 140 136 133 138 141
141 137 146 152 147 156 163 163 163 169 175 170 170 167 167 163 160 160
160 156 156 165 165 162 168 168 163 174 170 176 171 171 166 166 166 166
173 184 180 180 189 189 184 196 193 189 186 186 183 178 178 174 170 166
162 159 159 159 169 181 181 181 181 181 185 185 185 195 195 195 190 190
201 198 195 192 187 187 182 179 179 179 187 194 190 187 182 182 191 191
201 208]
[ 1 -1 1 -1 -1 1 -1 0 0 1 0 -1 -1 1 1 -1 1 0 -1 -1 0 -1 0 0
1 0 0 -1 -1 0 0 0 0 -1 1 0 1 -1 -1 0 1 0 -1 1 -1 -1 1 0
0 -1 1 1 1 -1 1 1 0 -1 1 1 1 -1 0 1 1 1 -1 0 0 1 -1 0
-1 -1 1 -1 0 1 1 1 1 0 -1 -1 -1 0 -1 -1 1 -1 -1 -1 0 0 0 1
1 1 1 0 1 -1 -1 -1 -1 1 1 0 -1 1 1 -1 1 1 0 0 1 1 -1 0
-1 0 -1 -1 0 0 -1 0 1 0 -1 1 0 -1 1 -1 1 -1 0 -1 0 0 0 1
1 -1 0 1 0 -1 1 -1 -1 -1 0 -1 -1 0 -1 -1 -1 -1 -1 -1 0 0 1 1 0
0 0 0 1 0 0 1 0 0 -1 0 1 -1 -1 -1 -1 0 -1 -1 0 0 1 1 -1
-1 -1 0 1 0 1 1]
```

Figure 3.12: trend 陣列輸出

上半部陣列為原數據的陣列, 字數遞增; 下半部為使用 1, 0, -1 判斷產生的 trend 陣列。如此便很清楚的看出, 0 是使用者停下的部分, 之後能從 0 所停頓的時間長度, 或是停頓所發生的次數判斷分析。

- 除了把 trend 陣列顯示外, 若能畫圖更能掌握此陣列數據的特徵。(圖) 點狀圖

```
[ ] plt.plot(trend, 'b.') #點狀圖
plt.show()
```

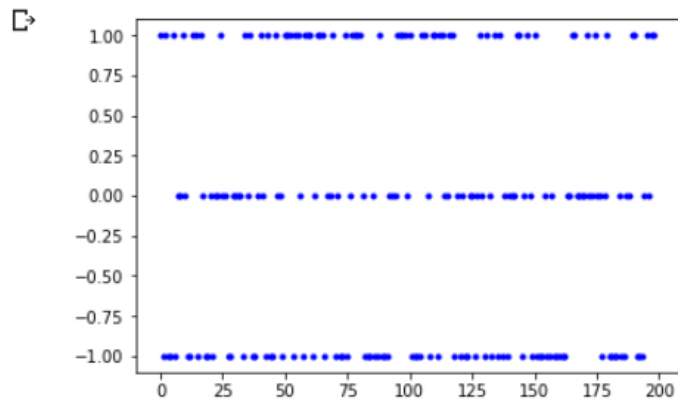


Figure 3.13: trend 點狀圖

(圖) 圓餅圖

increase: 62 decrease: 73 standstill: 64

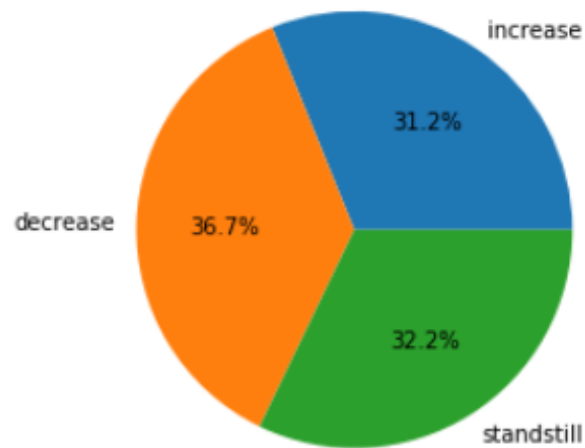


Figure 3.14: trend 圓餅圖

- 製做點狀圖或是圓餅圖可以顯示使用者停頓、打字或是刪除的密集程度。不過此處若是希望只看停頓處的地方，於是在多新增一個新的陣列，以下稱 trend2。trend2 是把 0 的元素抓出，把 0 換成 1；因此 trend2 數據便只有 0 與 1 數值。
- 部分程式：



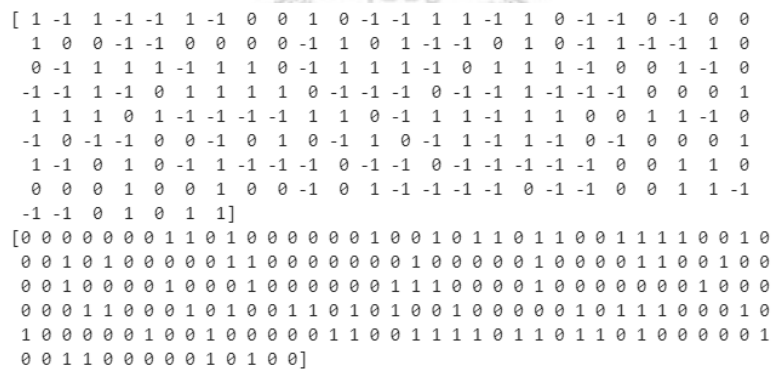
```

1 #len(trend)#trend的隔數 大約表示時間的1/2
2 trend2=np.zeros(len(trend),int)
3 for j in range (0,len(trend2)):
4     if (trend[j]!=0):
5         trend2[j]=0
6     else :
7         trend2[j]=1
8 print(trend)
9 print(trend2)#trend2是trend的停頓部分

```

Listing 3.4: python 數據 trend2

將 trend 與 trend2 顯示 (圖 3.15)



```

[ 1 -1 1 -1 -1 1 -1 0 0 1 0 -1 -1 1 1 -1 1 0 -1 -1 0 -1 0 0
 1 0 0 -1 -1 0 0 0 0 -1 1 0 1 -1 -1 0 1 0 -1 1 -1 -1 1 0
 0 -1 1 1 1 -1 1 1 0 -1 1 1 1 -1 0 1 1 1 -1 0 0 1 -1 0
-1 -1 1 -1 0 1 1 1 1 0 -1 -1 -1 0 -1 -1 1 -1 -1 -1 0 0 0 1
 1 1 1 0 1 -1 -1 -1 -1 1 1 0 -1 1 1 -1 1 1 0 0 1 1 -1 0
-1 0 -1 -1 0 0 -1 0 1 0 -1 1 0 -1 1 -1 1 -1 0 -1 0 0 0 1
 1 -1 0 1 0 -1 1 -1 -1 -1 0 -1 -1 0 -1 -1 -1 -1 -1 0 0 1 1 0
 0 0 0 1 0 0 1 0 0 -1 0 1 -1 -1 -1 -1 0 -1 -1 0 0 1 1 -1
-1 -1 0 1 0 1 1]
[0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 0
 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0
 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 1 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0
 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 0 0 1
 0 0 1 1 0 0 0 0 0 1 0 1 0 0]

```

Figure 3.15: trend2 輸出

如此只剩 0 與 1 兩個元素的 trend2 陣列畫成柱狀圖觀察，如同條碼的形狀。  
(圖 3.16) 其中線條就是使用者停頓處。

```
[ ] plt.bar(range(len(trend2)),trend2)#柱狀圖的停滯時間  
plt.show()
```

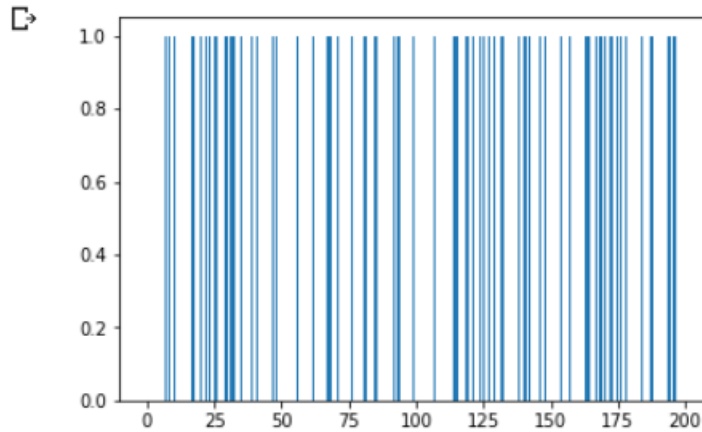


Figure 3.16: trend2 條碼狀方圖

- 最後一個圖做分區的統計，這裡計算 trend2 陣列 1 的出現次數，並且做分區的統計次數。分區的分法是察看 trend2 的全部元素個數，並且分成五等分，在一個個部分統計停頓的出現次數，例如若有 200 數據，則第一部份統計 1 出現個數為 0 50 組，第二部分統計為 50 100，以下類推。(圖 3.17)

```
14 10 9 15 16  
<BarContainer object of 5 artists>
```

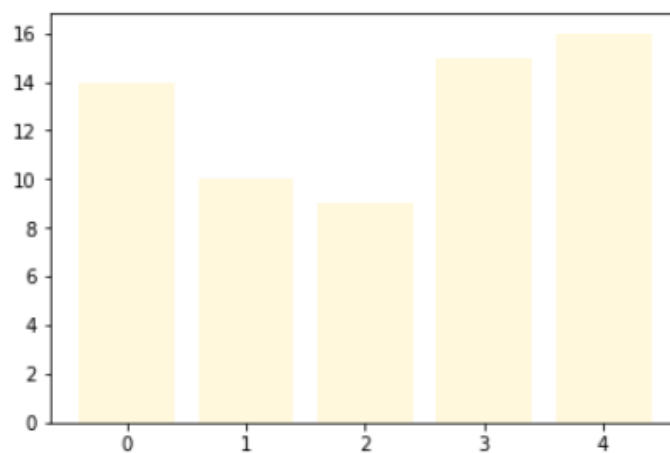


Figure 3.17: trend2 分區圖

不過如此計算的密度為強制分區，可能無法把密度情況很好的呈現出來，此部份討論於結果部份做詳細討論檢討。



#### 多組數據

以上分析都是只使用一組數據分析與統計的結果，若要在這基礎上擴增成多數數據的分析，則需要進行更多數據的製造與集合分析，所以以上的步驟需要進行多次。如此進行多次迴圈，此迴全次數可以自行修改。(圖 3.18) 為設定 10 次所呈現之全部格數與全部時間。

```
*total lattice= 109
*total time(sec)= 216

*total lattice= 214
*total time(sec)= 426

*total lattice= 108
*total time(sec)= 214

*total lattice= 93
*total time(sec)= 184

*total lattice= 163
*total time(sec)= 324

*total lattice= 125
*total time(sec)= 248

*total lattice= 142
*total time(sec)= 282

*total lattice= 187
*total time(sec)= 372

*total lattice= 183
*total time(sec)= 364

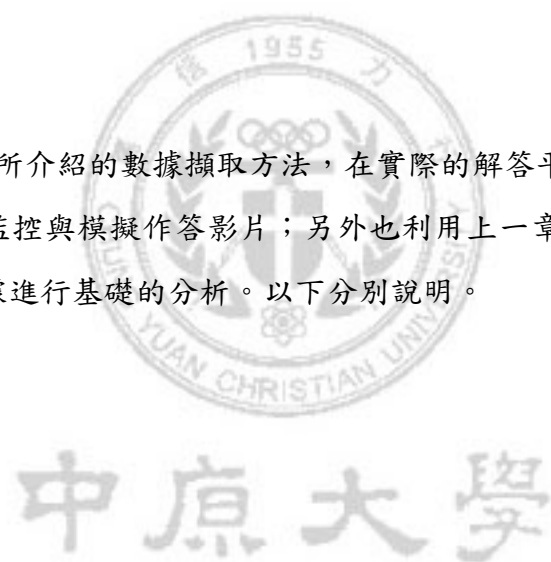
*total lattice= 243
*total time(sec)= 484
```

Figure 3.18: 多組數據

# Chapter 4

## 實驗與執行結果

本章利用上一章所介紹的數據擷取方法，在實際的解答平台雛形上，進行數據的即時採集、畫面監控與模擬作答影片；另外也利用上一章所介紹的數據分析方法，針對實際的數據進行基礎的分析。以下分別說明。



### 4.1 網頁平台數據監測

顯示頁面

在網頁的平台只有觀察使用者打字與紀錄時間並畫圖顯示。(圖 4.1) 為網頁偵測的頁面，中間的空白處就是給使用者打字的地方，並且在進行打字同時計算字數回傳，進而畫出時間字數的折線圖。

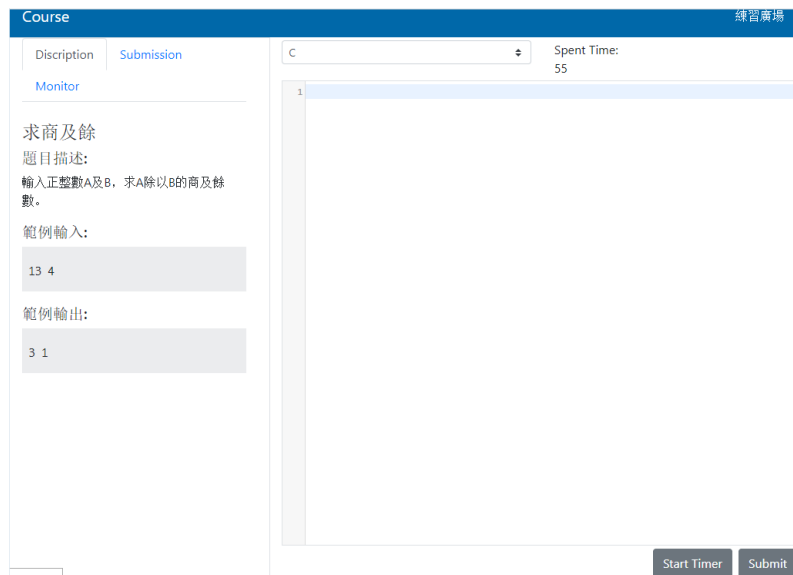


Figure 4.1: 網頁顯示頁面

## 網頁的圖形輸出

首先觀察從作答畫面所抓到的數據資料與圖形的顯示。圖 4.2 是針對一個簡單的範例題，在作答過程中所擷取的資料和圖形畫面結果：

### 1. 簡短程式碼 (圖 4.2)

左邊為實際作答的網頁畫面，右邊則是即時監測的圖形數據。在右邊的畫面中可以看到累計字數有上升或下降的情形，這表示使用者在撰寫程式時，使用了 `backspace` 倒退鍵或刪除鍵進行程式的修改。

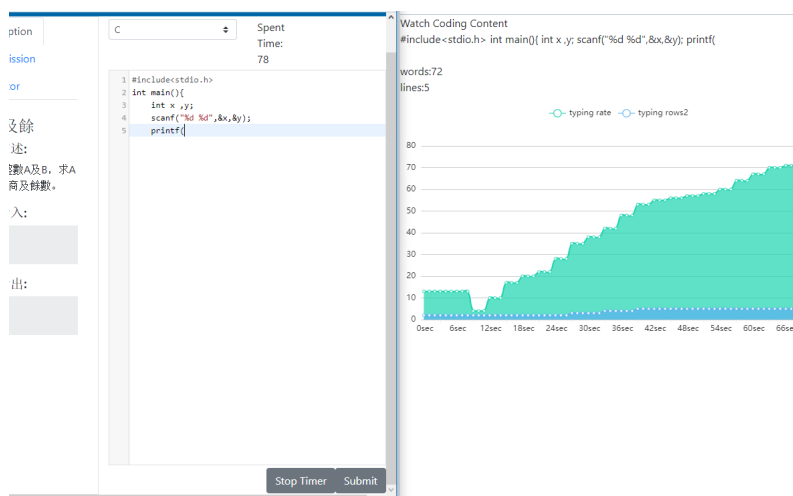


Figure 4.2: 簡短程式碼輸出

## 2. console 的訊息 (圖 4.3)

除了輸出的統計圖形，我們也可以使用 console 觀察系統擷取數據的過程，可以看到每當網頁介面的數據有所變化時，監控畫面會把數據顯示在圖形與網頁上。這裡可以看到資料每隔 3 秒鐘擷取一次，這個擷取的時間間隔必須合理的設定，如果間隔太短，固然可以得到更密集的資料，但是可能會造成太多冗餘的資料，以及資料太過龐大的問題，也會增加分析的複雜度；但如果設定太長，可能會漏失一些重要的資訊。此處設定為 3 秒鐘取一次。

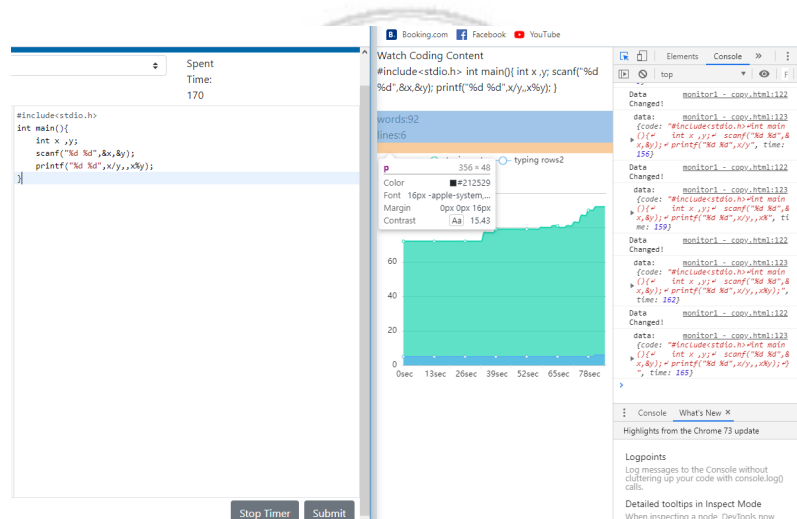


Figure 4.3: console 的訊息

## 3. 複製貼上 (圖 4.4)

若是使用者貼上大量文字的話，圖形變化會成階梯狀，主要因數據變化的幅度比自然打字大很多的緣故。透過統計畫面的顯示，可以估測使用者在此用了複製貼上的動作。

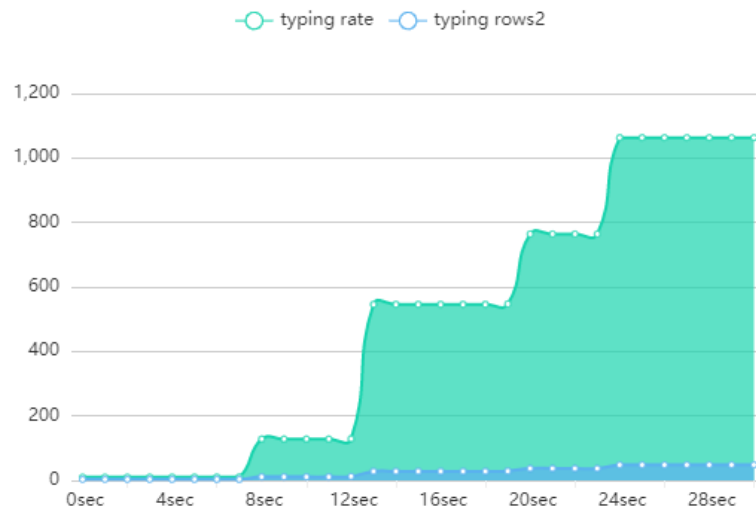


Figure 4.4: 複製貼上的輸出

#### 模擬作答畫面

除了觀察使用者作答的統計資訊之外，我們也可以利用擷取的取樣資料，模擬同學作答的畫面，也可以產出模擬作答的影片。

- 模擬作答畫面的運作:

打開模擬作答畫面，一開始會是一個空白的框 (圖 4.5)



Figure 4.5: 模擬作答啟始畫面



當使用者在作答介面進行打字的時候，模擬畫面的內容也會開始改變。前面提過 Firebase 資料庫可以自動通知程式資料有更新的事件，我們利用更新的數據來進行模擬畫面的更新動作。這樣一來，當使用者在打字的時候，模擬畫面也會在很小的延遲時間下同步更新。(圖 4.6)

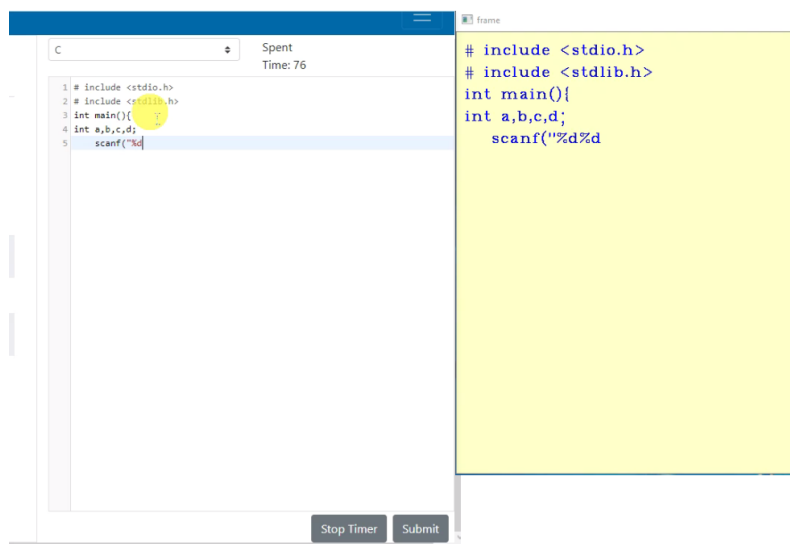


Figure 4.6: 模擬作答的過程畫面

當監控結束之後，可以把這些模擬的畫面輸出成影片，存成一個 AVI 影片檔。

- 影片動作與實際動作:

模擬作答的錄製影片，因為有少量的時間延遲，以及資料取樣的問題，在此想把輸出的 AVI 影片檔與真正打字的影片檔拿來做一比較，於是在打字的時候使用螢幕錄影軟體進行錄製，並拿來與模擬的錄製畫面相比，結果如圖 4.7 所示。

1	時間	螢幕錄影	makevideo
2	0:00:08	1 # include	# i
3	0:00:10	1 # include <	# include
4			
5	0:00:52	1 # include <stdio.h> 2 # include <stdlib.h> 3 int main(){ 4 int a,b,c,d; 5 print	# include <stdio.h> # include <stdlib.h> int main(){ int a,b,c,d; p

Figure 4.7: 螢幕錄影與 makevideo 比較

可以看出影片的動作會慢於實際的打字速度，因為在取值時有設定固定的秒數，因此若是使用者不間斷的打字必然不能完整呈現，所以模擬畫面所產生的輸出影片與實際上螢幕錄影的影片有少許延遲，但大體上已經可以接近真實的模擬使用者在打字時的種種動作。另外，我們用來製作模擬作答影片的資料，主要是取樣的文字資料，其資料量比實際在螢幕錄製的影片要少得多，這樣比較容易做到即時的處理。

- 資料的後續處理：

在第三章的基礎數據分析中，我們使用 Python 程式做了一些統計分析，那時候採用的數據是模擬產生的。那有了實際平台測試的樣本資料之後，我們也可以拿這些實際的資料來進行分析。因此我們也撰寫了一個 Python 程式，同樣經由 Firebase 把使用者的作答取樣資料抓取下來，供作後面的統計分析。抓取的取樣資料如圖 4.8 所示。

```

1 # include <stdio.h>
2 # include <stdlib.h>
3 int main(){
4 int a,b,c,d;
5 scanf("%d%d",a,b);
6 print(
(base) PS C:\Users\lab501\Desktop\codinghere\python> python analysis.py
Press Enter to continue...Received document snapshot: # include <stdio.h>
# include <stdlib.h>
int main(){
int a,b,c,d;
scanf("%d%d",a,b);
print(

```

Figure 4.8: python 抓下的文字

## 4.2 數據的統計分析

我們利用上一章所說明的 Python 程式碼在擷取的實驗數據上進行分析，此處分別使用了 10 組與 30 組的數據資料。



個別的圖形與數據

每一組數據會產生如圖 4.9 的一組圖形，以下只列每次產生的數據最基本的折線圖，共 10 組。(圖 4.10)

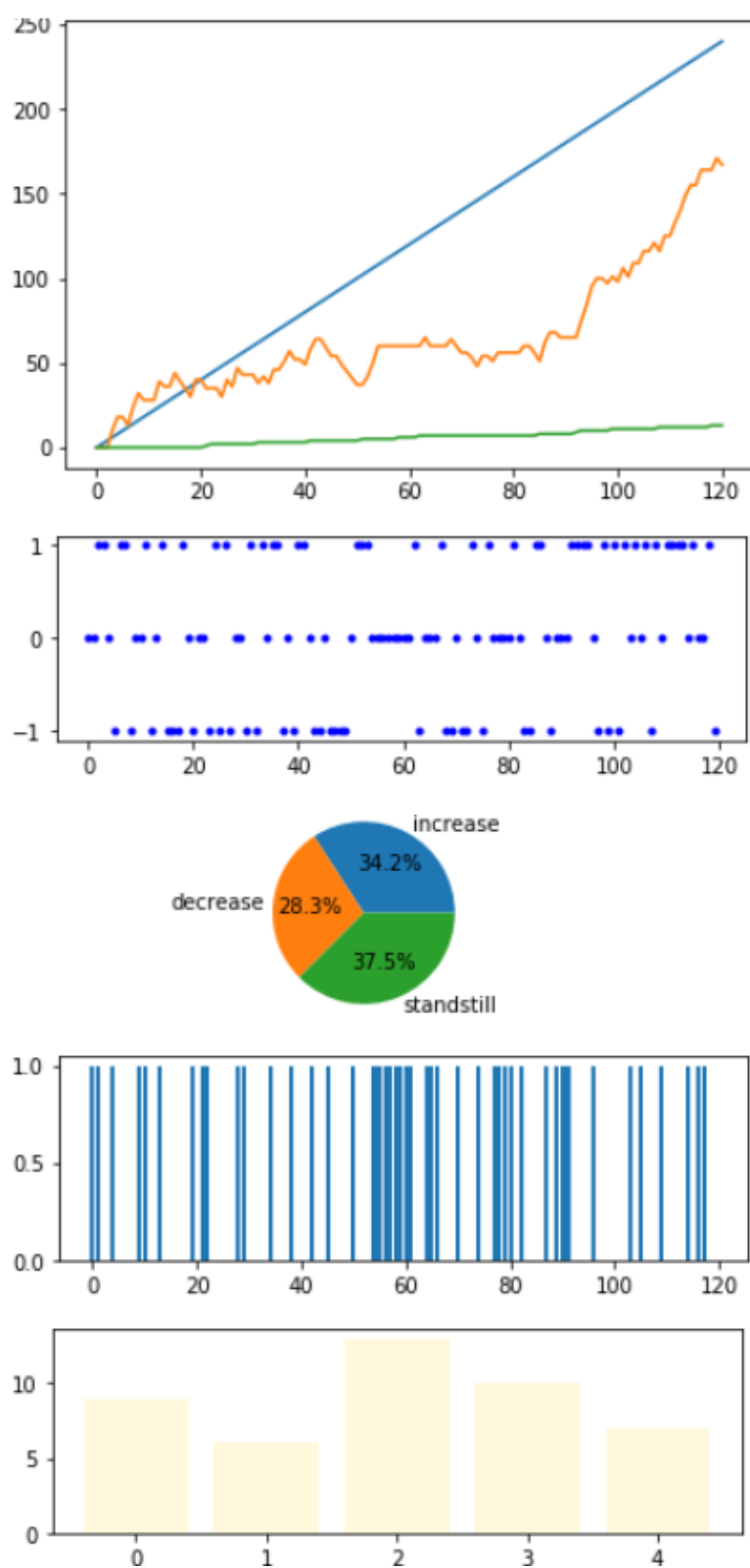


Figure 4.9: 圖形組

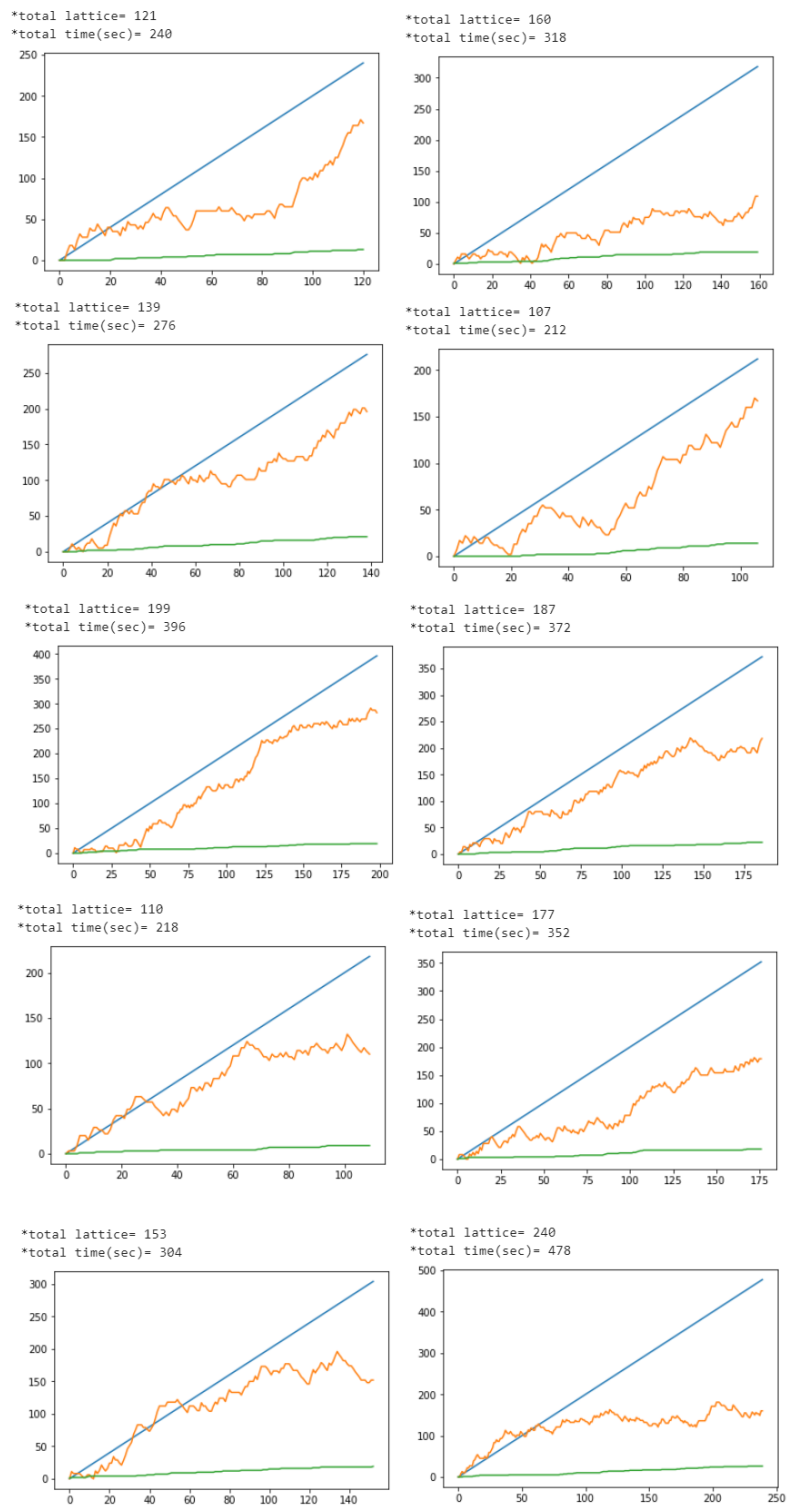


Figure 4.10: 10 組折線圖

## 統整的表格數據

把全部的數據統整起來製程表格，表格一樣使用 Python，可以直接在 Colab 上輸出顯示結果。

### 1. 10 組結果 (圖 4.11)

	隔數	秒數	作答 完畢 字數	作 答 完 畢 行 數	平均 作答 速度	前半 作答 速度	後半 作答 速度	全 停 頓 處 的 次 數	pause1	pause2	pause3	pause4	pause5
1	121.0	240.0	167.0	13.0	0.696	0.500	0.892	45.0	9.0	6.0	13.0	9.0	7.0
2	139.0	276.0	196.0	21.0	0.710	0.783	0.638	48.0	6.0	10.0	8.0	6.0	9.0
3	160.0	318.0	109.0	19.0	0.343	0.340	0.346	55.0	11.0	6.0	13.0	11.0	11.0
4	107.0	212.0	167.0	14.0	0.788	0.217	1.358	32.0	5.0	7.0	5.0	5.0	8.0
5	199.0	396.0	282.0	19.0	0.712	0.692	0.732	70.0	20.0	11.0	13.0	20.0	13.0
6	110.0	218.0	110.0	9.0	0.505	0.761	0.248	31.0	9.0	6.0	7.0	9.0	3.0
7	187.0	372.0	218.0	22.0	0.586	0.677	0.495	58.0	10.0	16.0	13.0	10.0	15.0
8	177.0	352.0	179.0	18.0	0.509	0.352	0.665	51.0	9.0	7.0	9.0	9.0	16.0
9	153.0	304.0	152.0	19.0	0.500	0.816	0.184	47.0	7.0	10.0	13.0	7.0	6.0
10	240.0	478.0	160.0	26.0	0.335	0.682	-0.013	85.0	15.0	20.0	12.0	15.0	18.0

```
print('全部平均', cloumn_avedata)
```

全部平均 [159.3 316.6 174. 18. 0.5684 0.582 0.5545]

Figure 4.11: 10 組結果表格圖

中 原 大 學

	隔數	秒數	作答完畢字數	作答完畢行數	平均作答速度	前半作答速度	後半作答速度
1	115	228	193	19	0.846	0.833	0.86
2	190	378	343	14	0.907	0.984	0.831
3	212	422	230	26	0.545	0.417	0.673
4	149	296	204	12	0.689	0.716	0.662
5	172	342	246	17	0.719	0.374	1.064
6	135	268	94	15	0.351	0.425	0.276
7	219	436	299	26	0.686	0.583	0.789
8	133	264	189	19	0.716	0.803	0.629
9	111	220	171	11	0.777	1.509	0.045
10	127	252	80	15	0.317	0.516	0.119
11	140	278	242	19	0.871	0.964	0.777
12	241	480	288	26	0.6	0.354	0.846
13	222	442	371	25	0.839	1.009	0.67
14	220	438	288	28	0.658	0.749	0.566
15	247	492	218	28	0.443	0.472	0.415
16	158	314	176	18	0.561	0.924	0.197
17	162	322	136	17	0.422	0.472	0.373
18	227	452	267	29	0.591	0.633	0.549
19	199	396	139	19	0.351	0.641	0.061
20	240	478	209	32	0.437	0.552	0.322
21	202	402	178	25	0.443	0.597	0.289
22	189	376	349	22	0.928	0.793	1.064
23	203	404	161	26	0.399	0.302	0.495
24	146	290	197	8	0.679	1.186	0.172
25	195	388	185	17	0.477	0.629	0.325
26	93	184	61	10	0.332	0.946	-0.283
27	98	194	60	2	0.309	0.33	0.289
28	93	184	132	8	0.717	0.772	0.663
29	204	406	225	18	0.554	0.951	0.158
30	120	238	91	16	0.382	0	0.765

Table 4.1: 30 組數據

2. 30 組結果，因資料較龐大所以使用表格顯示 (表 4.1))

## 4.3 結果討論

### 4.3.1 數據即時監控部份

- 本研究藉由網頁前端及時偵測使用者的打字情況，並且計算其字數與花費時間。不同於其餘答題系統需要繳交後才知曉使用者的答題狀況，此方法因為其實時偵測的特性，使用者的答題習慣也可以藉由字數與時間看出端倪。
- 目前根據擷取的資料，只做了字數、行數與時間的統計圖表，如可以增加更多的鍵盤偵測將會使系統更加完善，這是未來可以持續改善的地方。
- 在影片輸出部分，本研究比較了經由透過擷取數據所產出的模擬作答畫面與影片錄製結果，雖輸出的影片不及真正的錄影資訊來的完整與細緻，但輸出的影片能明確抓取使用者在作答時進行的改動。

### 4.3.2 數據分析部分

- 進行分析的部分最大的難處是缺少實際數據，因此在本論文在一開始研究的時候，先使用程式產生模擬數據，再以模擬數據撰寫及測試一些基本的分析方法，但模擬的數據理所當然地不及實際數據來的真實與客觀。本章則透過實際的作答平台採集實真實的數據進行分析，不過因為平台仍處於雛形階段，也沒有大量數據可以採集，因此只有基本的測試結果。
- 在討論數據停頓的密集程度的地方(圖 3.16)，目前選用的方法是把整體數據分成五個等分並寫計算每一等分的停頓次數；而此部分也是不夠嚴謹的，因可能密集的地方會被分組計算而分割，所以分區計算的方法並不能表示絕對的疏密程度。



## Chapter 5

### 結果討論與未來展望

本論文主要針對線上解題平台，製作了一個即時監控與數據分析的子系統。目前可以在作答的雛形平台上進行資料的採集，自動上傳到 Firebase 的雲端資料庫中，並且可以自動產出一些基本的作答統計資料和圖形。另外也可以透過擷取的資料，產出模擬作答的畫面和影片檔，以及進行一些基礎的資料分析。本研究所提出的方法還有很多可以改進的空間，未來可以根據這個基礎繼續擴充及改良，希望可以更加完善，並可以成為未來更具人工智能的作答平台的一個基礎。

中原大學

# Bibliography

- [1] 解讀 google 的人工智慧圍棋「大腦」. <https://www.bnext.com.tw/article/38740/BN-2016-02-22-183726-196>.
- [2] Web app 的定義. <https://phd.com.tw/knowledge/app-dev/web-app/>.
- [3] Progressive web apps 簡介. <https://support.google.com/google-ads/answer/7336531?hl=zh-Hant>.
- [4] 瘋狂程設網站. <http://coding-frenzy.arping.me/>.
- [5] 瘋狂程設論文. <https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi?o=dnclcdr&s=id=%22102MCU05392008%22.&searchmode=basic&extralimit=asc=%22%E9%8A%98%E5%82%B3%E5%A4%A7%E5%AD%B8%22&extralimitunit=%E9%8A%98%E5%82%B3%E5%A4%A7%E5>.
- [6] 網路爬蟲,wiki. <https://zh.wikipedia.org/wiki/%E7%B6%B2%E8%B7%AF%E7%88%AC%E8%9F%B2>.
- [7] Progressive web app. <https://blog.techbridge.cc/2016/07/23/progressive-web-app/>.
- [8] 網路應用程式,wiki. <https://zh.wikipedia.org/wiki/%E7%BD%91%E7%BB%9C%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F>.
- [9] Html5,wiki. <https://zh.wikipedia.org/wiki/HTML5>.
- [10] 階層式樣式表,wiki. <https://zh.wikipedia.org/wiki/%E5%B1%82%E5%8F%A0%E6%A0%B7%E5%BC%8F%E8%A1%A8>.

- [11] Javascript,wiki. <https://zh.wikipedia.org/wiki/JavaScript>.
- [12] vue.js. <https://vuejs.org/>.
- [13] vue,wiki. <https://zh.wikipedia.org/wiki/Vue.js>.
- [14] jquery. <https://jquery.com/>.
- [15] jquery,wiki. <https://zh.wikipedia.org/wiki/JQuery>.
- [16] Firebase. <https://firebase.google.com>.
- [17] Firebase 是什麼. <https://tw.alphacamp.co/blog/2016-07-22-firebase>.
- [18] Ai.wiki. <https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C>.
- [19] tensorflow. <https://www.tensorflow.org/>.
- [20] tensorflow,wiki. <https://zh.wikipedia.org/wiki/TensorFlow>.
- [21] colab. [https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=5fCEDCU\\_qrC0](https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=5fCEDCU_qrC0).
- [22] 定時器使用方式及運作機制. <https://ithelp.ithome.com.tw/articles/10209476?sc=iThelpR>.
- [23] chart. [http://jsfiddle.net/no2don/kp50Lqnh/1/?utm\\_source=website&utm\\_medium=embed&utm\\_campaign=kp50Lqnh](http://jsfiddle.net/no2don/kp50Lqnh/1/?utm_source=website&utm_medium=embed&utm_campaign=kp50Lqnh).