

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
1 緒論	1
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
2 背景技術介紹	5
2.1 web app	5
2.2 HTML 與 CSS	6
2.3 javascript	6
2.3.1 vue.js	7
2.3.2 jquery	7
2.4 Firebase	8
2.5 python	8
2.5.1 機器學習相關	8
2.5.2 Colab	9

3	系統架構與規劃	10
3.1	系統前半 js 與 html 之偵測	10
3.1.1	模擬平台	11
3.1.2	計算	12
3.1.3	輸出	12
3.2	系統後半 colab 分析部分	14
3.2.1	製造測試數據	14
3.2.2	分析數據部分	18
4	實驗與執行結果	26
4.1	網頁平台數據結果	26
4.2	python 的討論分析	31
4.3	結果討論	36
4.3.1	html 網頁偵測部份	36
4.3.2	python 分析部分討論	36
5	結果討論與未來展望	37
	Bibliography	38

中 原 大 學

List of Figures

3.1	系統前半 js 與 html 架構	10
3.2	模擬平台介面	11
3.3	範例圖片	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	16
3.7	第二部分數據呈現與折線圖	17
3.8	第三部分數據呈現與折線圖	18
3.9	整體結果	19
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	24
4.1	網頁顯示頁面	26
4.2	簡短程式碼輸出	27
4.3	console 的訊息	28
4.4	複製貼上輸出	28

4.5	makevideo 檔初始	29
4.6	makevideo 檔與網頁介面	30
4.7	螢幕錄影與 makevideo 比較	30
4.8	python 抓下的文字	31
4.9	圖形組	32
4.10	10 組折線圖	33
4.11	10 組結果表格圖	34



中 原 大 學

List of Tables

4.1 30 組數據	35
----------------------	----



中 原 大 學

Listings

3.1	js 字數與行數的判讀與計算	12
3.2	python 數據第一部分	16
3.3	python 數據第二部分	16
3.4	python 數據 trend2	22



中 原 大 學

Chapter 1

緒論

1.1 研究背景

近年因為神經網絡圍棋 AI——AlphaGo，戰勝了人類職業選手，越來越多人投入到人工智慧這塊領域。神經網絡其實已經有了多年的歷史，但是直到最近才浮出檯面與重視。是因為人工智慧需要大量的「訓練」。對早期研究者來說，想要獲得不錯效果的最小量測試，都遠遠超過計算能力和能提供的數據的大小，也就是取得大量的測試數據非常不容易。但最近幾年，一些能獲取大量資源的團隊重現挖掘神經網絡，其實就是透過「大數據」來使測試更有效率。[1]

安裝於手機或是電腦亦或其他行動裝置的應用 app，如在瀏覽器上達成同樣效果，而不需進行安裝動作，可謂網頁 app，以下稱 web app。前端網頁使用 HTML、XHTML、HTML5、CSS、Java Script 等網頁標準技術製作，後端使用 PHP、ASP.NET、JSP、RoR 等程式語言開發，並連結資料庫或其它資料來源，且透過瀏覽器輸入網址後執行。Web App 只要使用裝置的瀏覽器輸入網址即可執行測試。若有任何問題，程式修改後，可以快速的進行測試，甚至有時只需要簡單的重新整理網頁即可。[2]

Progressive Web Apps(PWA) 就是結合網頁和應用程式的使用者體驗。PWA 可以直接在使用者的裝置主螢幕上安裝與執行，不需要透過應用程式商店取得。因為有網路應用程式資訊清單檔，所以 PWA 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]

1.2 研究動機與目的

本校電機系開設的計算機概論課程所使用的線上評測系統是瘋狂程設 [4] 此系統為謝育平老師與其學生之原創網站。[5]

鑒於雖此評測系統十分優異，但使用了長時間之後些許感受到此系統有些許地方不足夠，或是希望此系統多一些應用等等。而且若要使用此系統的程式評測，必須下載桌面版方能使用此評測系統，而網頁版只供使用者查詢或是紀錄查看。在紀錄部分，正是我想所討論與擴展處，系統僅僅只是把每位同學的作答狀況記錄下；比如說只包含每人的作答時間與全部字數等，並且計算評分。

而我們想知道的是，同學在作答中的行為，想分析個人的作答狀況；在本系統可能無法看到太過細部的東西。因此設計一新的系統，可以在線使用，透過瀏覽器傳遞，時間間隔更小的分析並記錄使用者的狀況。類似一種簡單的爬蟲程式。[6]

1.3 研究流程規劃

因為想從瘋狂程設此系統發想，首先要做的是為觀察原本的系統: 已經做到哪些，以及想要加上哪些功能等等。

- 本研究認為最主要核心為兩處

1. 線上式觀察: 因認為桌面板操作的普及與便利性不及使用瀏覽器的網頁版本，新的系統架設在網頁前端才會更加方便。
2. 分析數據: 原系統只是記錄完成後的最終數據，我們若想知道更加細微的地方可能無從所知。因此結合網頁與分析，希望同學們一邊作答時，後面網頁的程式一邊運作，如此便可以知道不同於繳交的數據，還知道作答中的細部數據。

- 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

本章節對本文所使用到之背景技術介紹與討論。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 節介紹所使用背景技術 javascript 與其擴展外掛與建構使用者界面的不同框架 vue 與 jquery 2.4 節介紹所使用的背景技術 Firebase 2.5 節介紹 python 與大數據的關係，與使用的 colab，他是 Google 公開的一個 Python Notebook 工具。

2.1 web app

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

- 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]

- 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]

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 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。

2.3.2 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]

2.4 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，並且可以使用 google 提供的 GPU 或是 TPU，不需在自己的主機建構環境。



中原大學

Chapter 3

系統架構與規劃

本章介紹系統架構與佈局規劃。

3.1 系統前半 js 與 html 之偵測

本研究欲建設一模擬的平台，在此使用者可以打字於一視窗內，此視窗能傳送使用者所打出的文字與計算時間的數據至往後的分析系統裡。(圖 3.1)



Figure 3.1: 系統前半 js 與 html 架構

3.1.1 模擬平台

- 在進行觀測同學們的打字程式前，首先要建立一個平台，才能讓使用者在上面進行打字與實驗。在此使用了一網頁介面進行對使用者打字的前端數據收集，此介面借用了一程式解題平台的網頁介面；在此介面基礎下除了能使用原本的解題，並且能使用本研究的計算字數與時間的功能，並且將種種數據由取間隔相同時間依序存入 firebase。(圖 3.2)

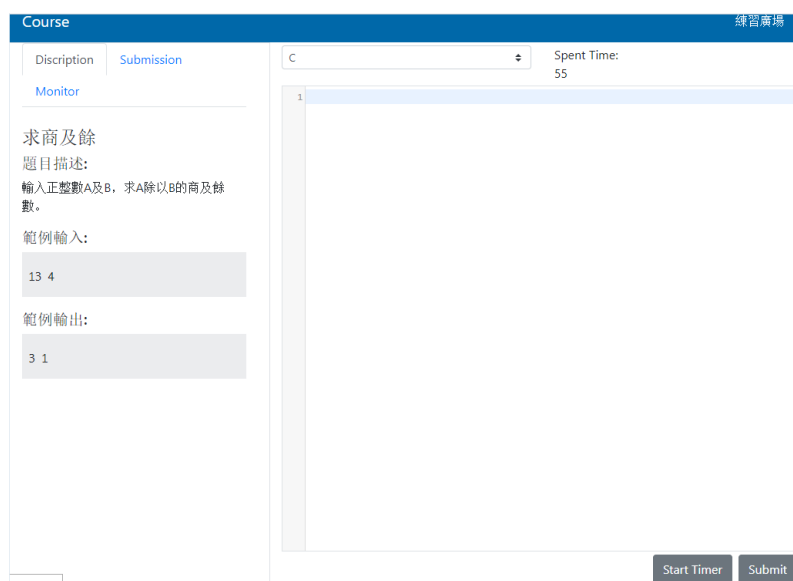


Figure 3.2: 模擬平台介面

- 功能:
 - 左方頁籤: 左方會顯示使用者正在作答的題目，與其範例輸，就是個供使用者查看當前題目的地方。而 Submission 頁籤則是當使用者已經完成他的作業按下 submit 鍵(右下角)時，然後這筆資料就會傳送到一評判伺服器進行評測並傳回結果，因此介面與與評測系統的屬於另一範疇，在此便不再贅述。
 - 右上計算時間讀秒: 右上的計時器便是可以記錄使用者的作答時間，在使用者打字進行的同時便會一起計時與記錄其數據。而當按下右下的 stop timer 便會停止計時。

3.1.2 計算

- 字數與行數的判讀與計算

擷取 js 程式：

```
1 _this.code = doc.data().code;  
2 _this.words = doc.data().code.length;  
3 _this.lines = 1+doc.data().code.match(/\r?\n|\r/g).length;
```

Listing 3.1: js 字數與行數的判讀與計算

在使用者打字在作答區時，可以使用 javascript 的 vue 語法回傳在文字框的字行數據。行數據的判斷方法有些許差異，在此我用的是只要使用者打自使用到換行，如 enter 字符，便會判斷為增加一行。獲取這些數據後會回傳至 html 網頁顯示。

- 時間的判讀與計算

我所使用的方法是 setTimeout 與 setInterval 函數，但是 setTimeout 函數只能循環一次，而 setInterval() 可以不斷循環，故使用 setInterval 更好。[22] 也就是說，我加上了固定秒數、時間的基礎上回傳字數資料，相當於固定的時間定時的收集使用者打在網頁介面的數據；如此一來，把時間與數據集中便可以知道時間與使用者打字的關係，進而能夠做到接下來分析的動作。

3.1.3 輸出

- 圖表：

除了能夠獲取實質上的數據數字外，我使用了 chart.js 與 vue.js 進行數據繪圖與整理。[23](圖 3.3)

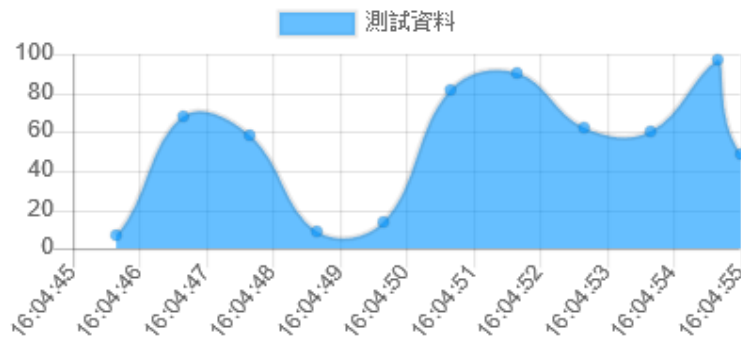


Figure 3.3: 範例圖片

此範例圖片是每隔一段時間隨機取一數字並畫成圖，推送至 html 網頁，他是即時更新的圖表。而在前頭所提及的 `setInterval()` 函數，可與圖表結合，可以更改 `setInterval()` 函數所選取的秒數擷取不同時間。所以圖表的輸出為：X 軸時間，Y 軸字數與行數。

- 遠端輸出:

圖表與其字數行數的觀察會傳送到 firebase 的數據庫並且把資訊做一個暫時性的儲存，而輸出的圖表與數據是由 firebase 下抓進來的，並且顯示至另一 html 網頁檔，以下稱為 monitor 檔。加入 monitor 檔此步驟的目的是因為若要觀看同學們的作答狀況便可以從此網頁進行遠端的觀看而不用動到原本的介面。

- 影片輸出:

在進行圖表與字數的計算時，有另一項動作也在同步的進行中；便是錄影，在傳送字數與時間的數據時，此錄影部分會 firebase 抓取使用者時間與變動的數據，並且存取這些變動部分到一暫存地，然後此程式便會抓取暫存地的數據製成影片並且輸出。也就是此部分影片可以模擬使用者在打字行動。

- 其他:

除了 monitor 檔之外的輸出外，也有另一項輸出為 python 檔的輸出，此檔也是經由 firebase 的數據庫下把程式載下，並且此功能與之後的分析數據部分息息相關。

3.2 系統後半 colab 分析部分

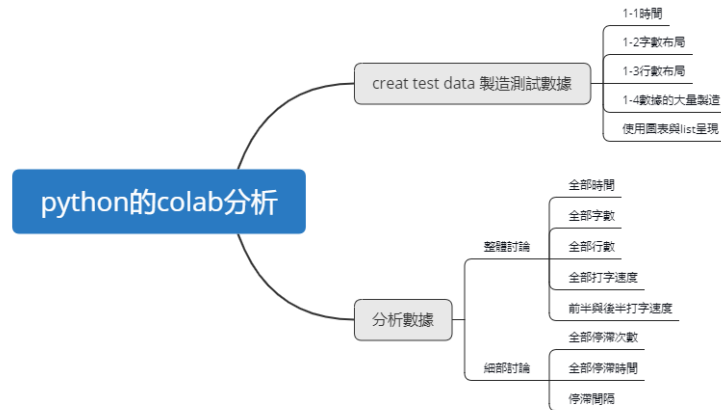


Figure 3.4: 分析系統架構

在後半部分本研究換了一種程式語言撰寫，為 python。因為拿到的實際數據十分有限，於是在觀察實際數據的基礎下使用 python 進行模擬數據與分析的方向大致為: 1. 某同學做一題的總時間分析，統計大家平均的作答時間 2. 停頓處: 若在某處的數據發現停頓，那麼表示這位同學可能在此處卡住

3.2.1 製造測試數據

討論數據的特徵

圖 (3.5) 為在 html 的程式中執行後的結果，因為是打字數與時間的關係，所以數據自然是漸進式的。佔比面積較大為字數，下部分小的為行數。也就是時間與字行數的關係圖。但如果數據要一個個採集變不是如此容易，於是面臨測試數據不夠於給之後的分析程式使用，於是便發想也是使用程式製造出類似於測驗結果的數據測試。

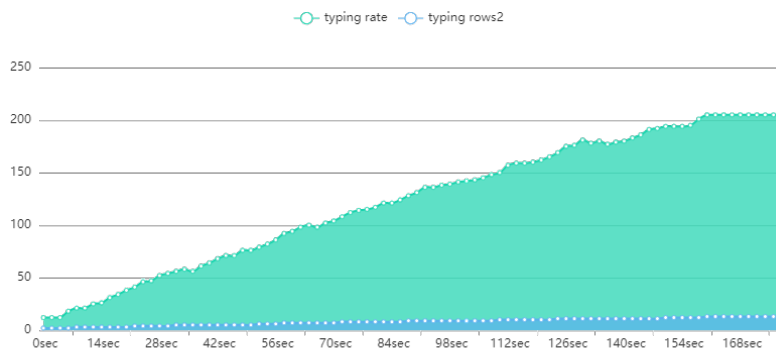


Figure 3.5: 執行結果漸增圖

如下討論數據的特徵: 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. 第二欄數據的製造: 第二欄是字數的統計, 要像是圖 (某) 中一樣漸進的增加, 但是每位同學的字數可能不同, 或是停頓的地方可能也不同。它們的字數可能也要設定一個區間。3. 第三欄為行數, 理論上越多字的越多行, 但還是必須規定不能生成太多行數。

規定格式

觀察圖 (3.5) 可以知道有好幾組數據在學生打字的時候同步進行計算, 而必須規定一數據格式進行傳輸與分析時會較為方便。這裡為了方便撰寫程式製造類似打字數據的程式, 便規定數據必須有三行陣列, 也就是 $3 \times (\text{時間格數})$ 陣列 (圖某), 如此陣列的表示數據在作往後的分析或是呈現給使用者看更能靈活地使用。以下分三部分討論其中數據的構成。

數據第一部分

第一部分討論的是時間, 網頁程式所抓取的時間是固定的, 因此時間的間隔相同。(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: python 數據第一部分

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。

數據第二部分

第二處可謂最為重要、關鍵與複雜的部份，畢竟是數據最有變化性的部分。一開始構想的是使用者被網頁觀測到的數據是不停增加，但不會每一段時間增加幅度相同。

```
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: python 數據第二部分

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。如此撰寫的話，字數會無止盡的增加，在短短時間不可能寫如此快速。只有增加的數據無法反映使用者若是遇到不會的地方的情況，我想加入”若是使用者在思考”的停頓，就是會維持同一個字數許久，而數據不停增加則模擬不出。打字字數增加，有些地方增加特別快，但也有停頓或是寫錯的地方。如此在打錯的時候使用剪下 (ctrl+x) 或是倒退 (Backspace)，所以整體的數據並不會一路平滑的增加，藉由此情狀模擬在作答一個題目數據呈現的變化。

於是在使用者不停增加第二部分字數的基礎上修改至可能停駐思考或是寫錯重來的時候。新增三個參數分別代表增加、停留、倒退，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亮是不等的，此部分也使用 random 函數實現。(圖 3.7)

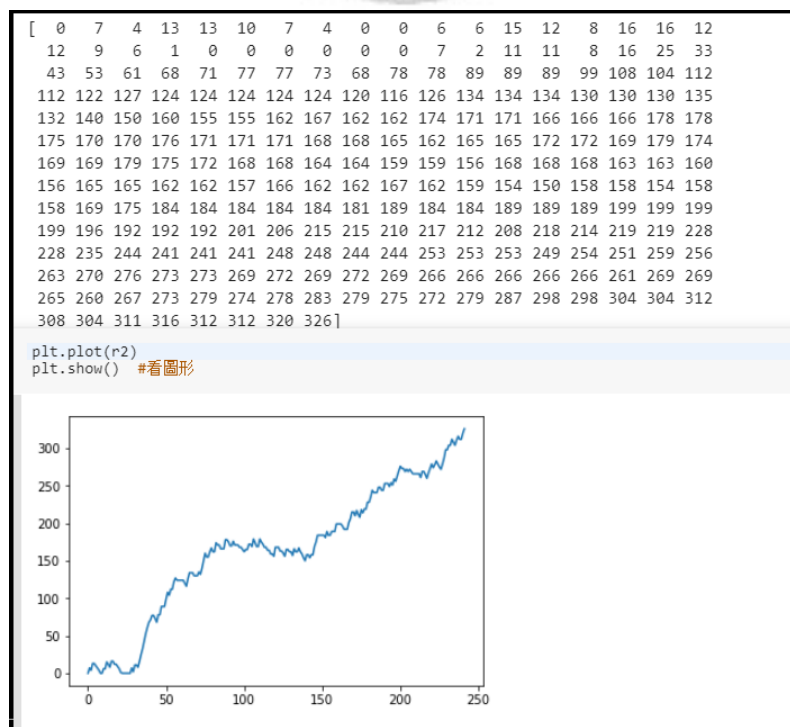


Figure 3.7: 第二部分數據呈現與折線圖

數據第三部分

第三部分為行數此部分不用著墨太多，雖然第二部分字數與第三部分行數息息相關，但是換行並不用太過關注，或是只要使用原程式碼 (js) 的監測便會知道使用者換行的時候第三部分想製造出 0,0,0,1,1,1,2,2,2,3,3,3,3,3,3,4,4,4,..... 如此的陣列漸增關係，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)

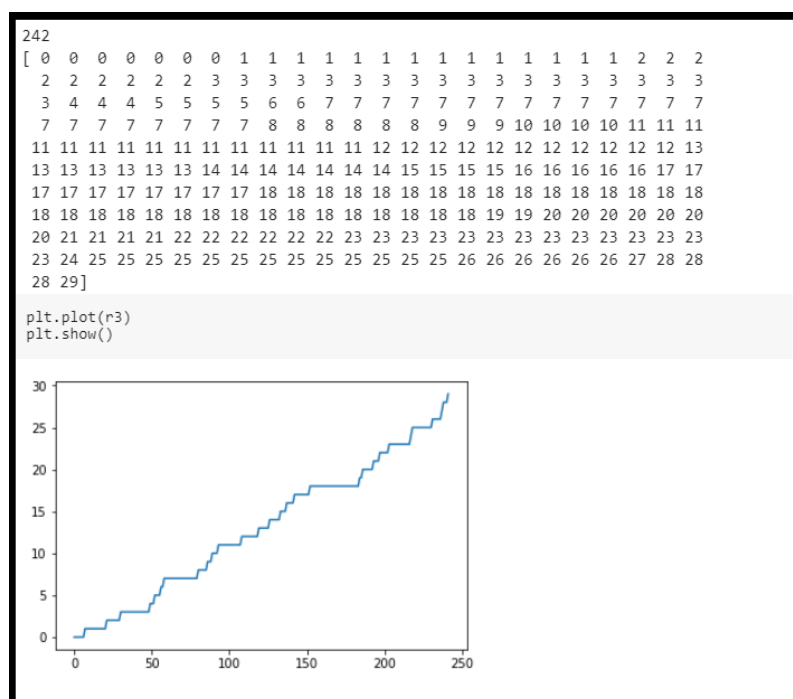


Figure 3.8: 第三部分數據呈現與折線圖

3.2.2 分析數據部分

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個陣列的原因，在分析步驟可以很容易抓到要的数据進行運算。
- 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

然後在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集大誤差，因此做多個速度的比較數據會更佳。(圖 3.10)

平均作答速度: 0.340277777777778 字/sec
前半作答速度: 0.270833333333333
後半作答速度: 0.409722222222222

Figure 3.10: 整體速度結果

- 收集這些速度的數據在之後比較多組，也就是多個同學更能知道每個人的打字速度，除了分析打字的字數外速度也可以加入進行細部分的分析。

單組數據細部分析

- 除了單組數據的整體分析，最總體的數據與平均時間外，打字的停滯點也是至關重要。
- 要找出數組中的斷點若是直接觀察是非常容易的，如 (圖 3.10)

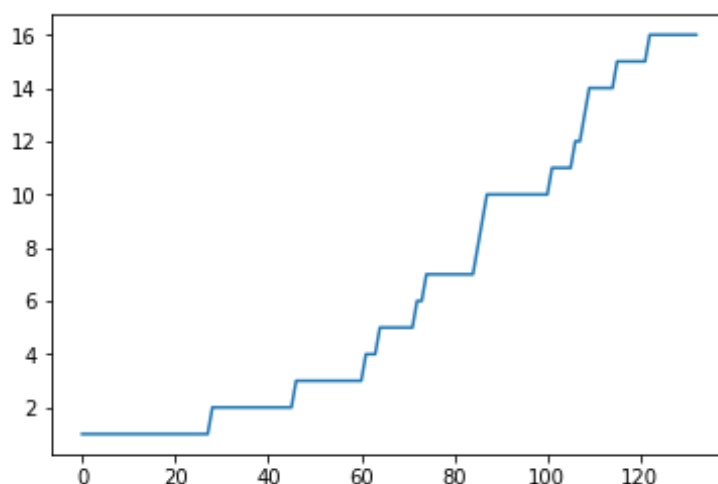


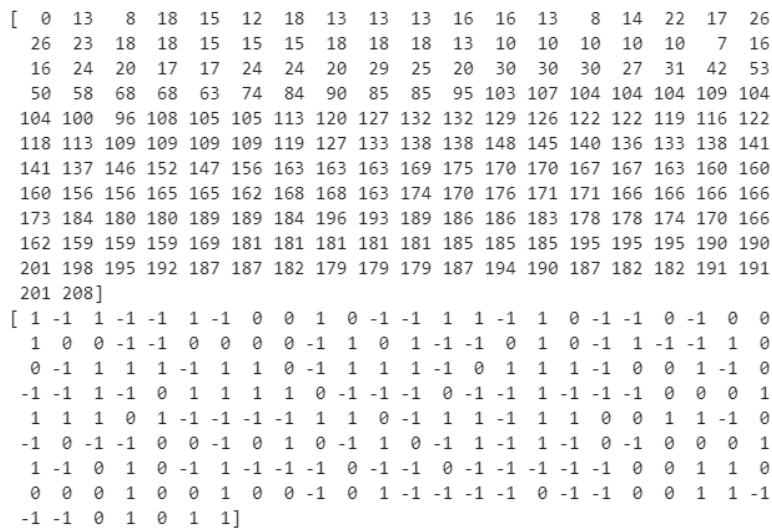
Figure 3.11: 示例圖片

若是看到打字沒有增加，在圖中是平行線段，就代表動作停滯、維持同一字數。

- 找出停滯部分本研究所使用方法是，使陣列的後項數據剪去前項數據，並且判斷此數據值若是正數，則判斷增加；若是 0，判斷停滯；若是負數，判斷刪除倒退。可將此判斷數據化存成另一陣列，在此以下稱 trend 陣列。

1. 後項減去前項為正: 打字字數增加，1 表示
2. 後項減去前項為零: 打字字數不變，0 表示
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 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 95 103 107 104 104 109 104
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 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()
```

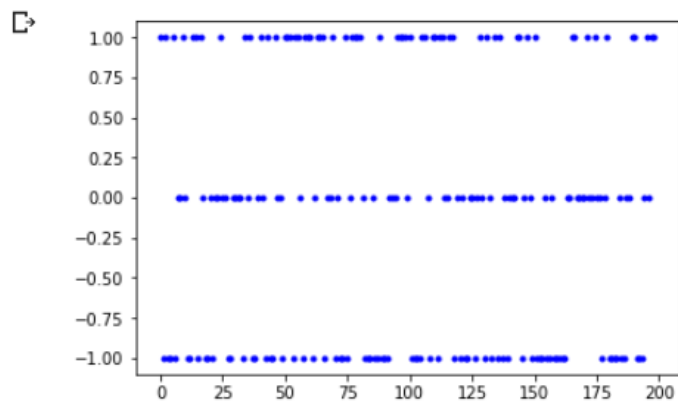


Figure 3.13: trend 點狀圖

(圖) 圓餅圖

increase: 62 decrease: 73 standstill: 64

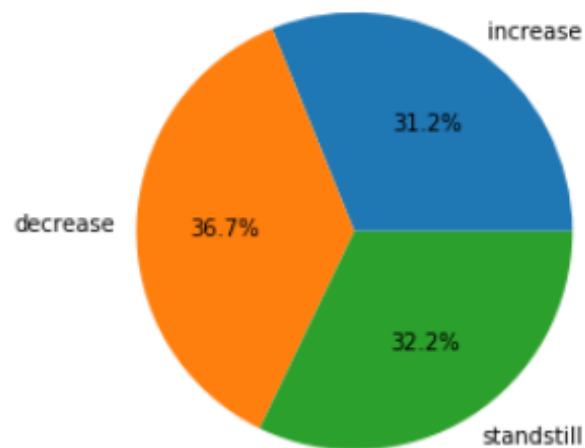


Figure 3.14: trend 圓餅圖

- 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：

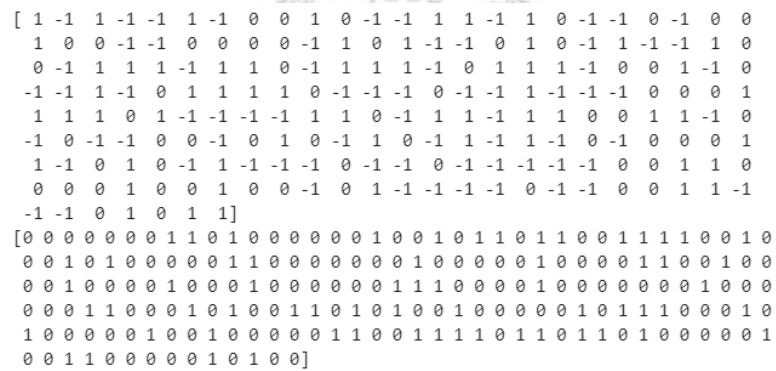

```

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()
```

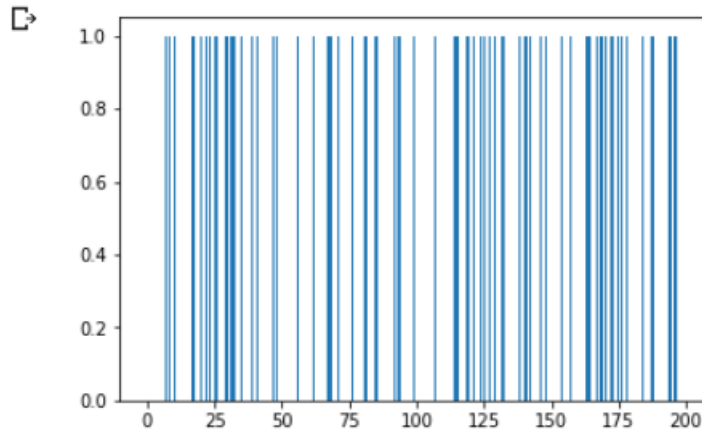


Figure 3.16: trend2 條碼狀方圖

- 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>
```

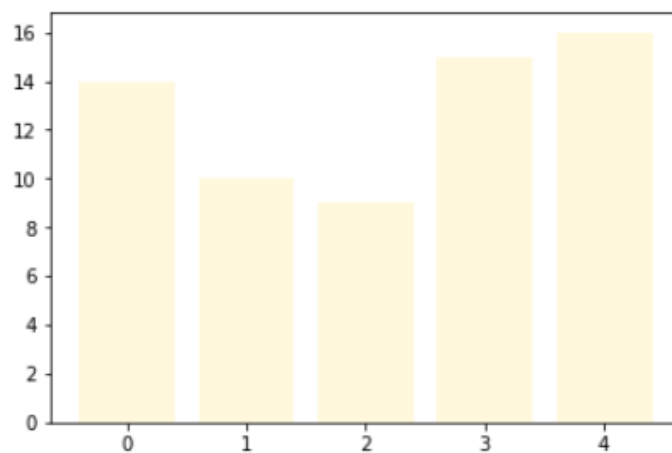


Figure 3.17: trend2 分區圖

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

```
*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

以上分析都是只使用一組數據分析與統計的結果，若要在此基礎上擴增成多數據的分析，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。



Chapter 4

實驗與執行結果

本章節討論網頁與分析的結果與輸出數據

4.1 網頁平台數據結果

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。

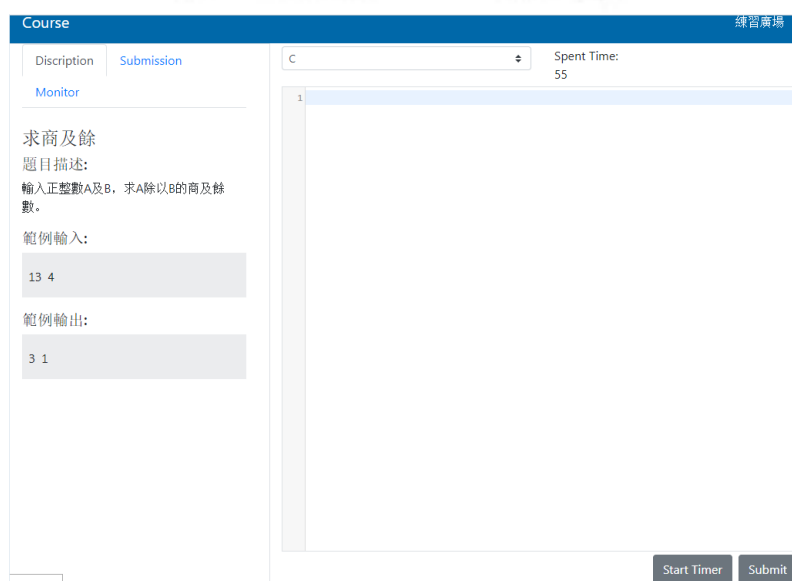


Figure 4.1: 網頁顯示頁面

網頁的圖形輸出

首先先觀察 monitor 檔所抓到的數據與圖形的顯示。因本研究是基於討論學生在做題目的分析，於是我找了一些文字試打並擷取 monitor 檔的圖形畫面結果：

1. 簡短程式碼 (圖 4.2)

左邊部分為 html 的網頁使用者介面，右邊為 monitor 檔所呈現的的圖形數據，並且圖上可以看出在打字時若遇到打錯的地方，所使用 backspace 倒退鍵的情況也變多了，所以圖上呈現才會有更多的彎曲；上升一點下降一點就是使用者在改錯的地方。

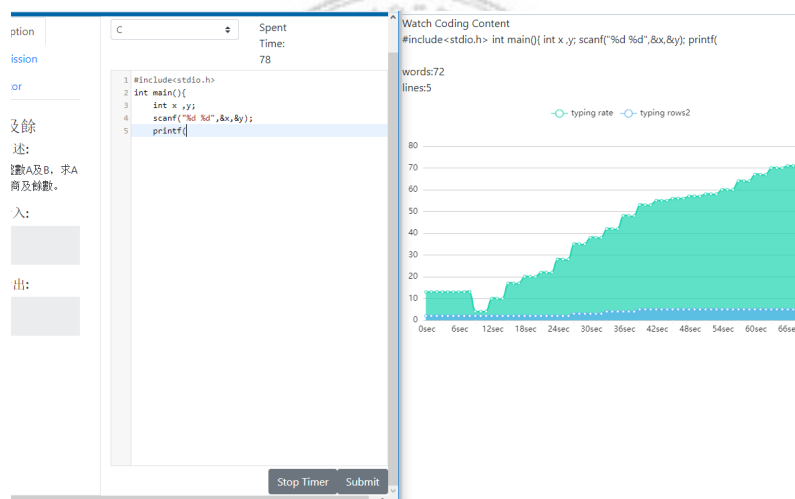


Figure 4.2: 簡短程式碼輸出

2. console 的訊息 (圖 4.3)

把上面打到一半的程式繼續往下打，並且把 console 打開觀察，可以看到每當網頁介面的數據有所變化時，monitor 檔便會知曉每次增加或減少的變動，進而把數據顯示在圖形與網頁上。右邊 console 部分更可以看出什麼時間的時候程式碼進行到哪一部分。在此設定為 3 秒鐘取一次，能在 console 清楚的看到，之所以不需要太密集取值是因為適量的調控數據使數據不要太多或過於繁雜，若是一次取太多值，則需要更多的空間存取，如此之後的存取與分析會變得更複雜。因此在此處只設定 3 秒鐘取一次。

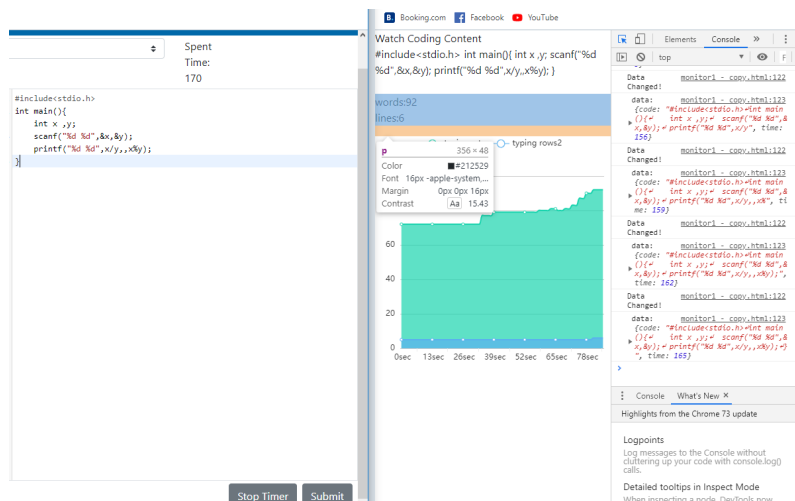


Figure 4.3: console 的訊息

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因數據變化的幅度比自行打字大很多，此為模擬若是使用者頻繁的貼上。因此如使用者使用了貼上鍵，此頁面是能夠偵測到的。

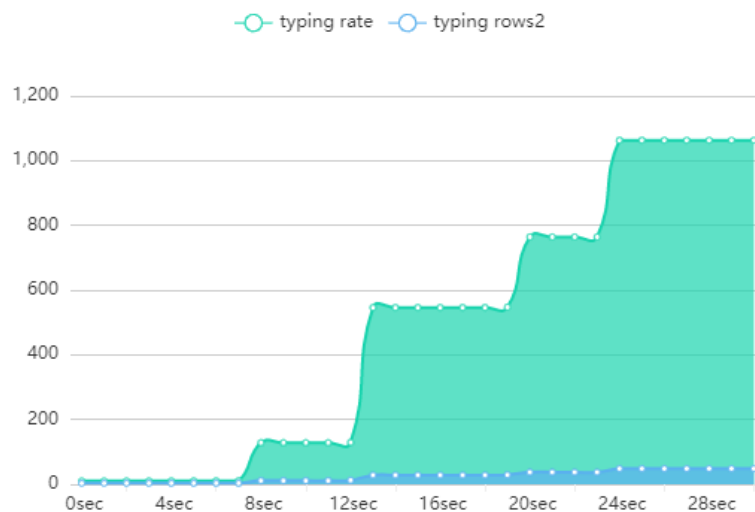


Figure 4.4: 複製貼上輸出

網頁的影片輸出

除了在 monitor 檔或是 monitor 檔裡的 console 中可以觀察使用者在打字的時候的動作外，把這些數據綜合起來則可以統整成一個影片並且輸出。以下稱此功能

的檔案為 makevideo 檔。

- makevideo 檔的運作:

打開 makevideo 檔會是一個空白的框 (圖 4.5)

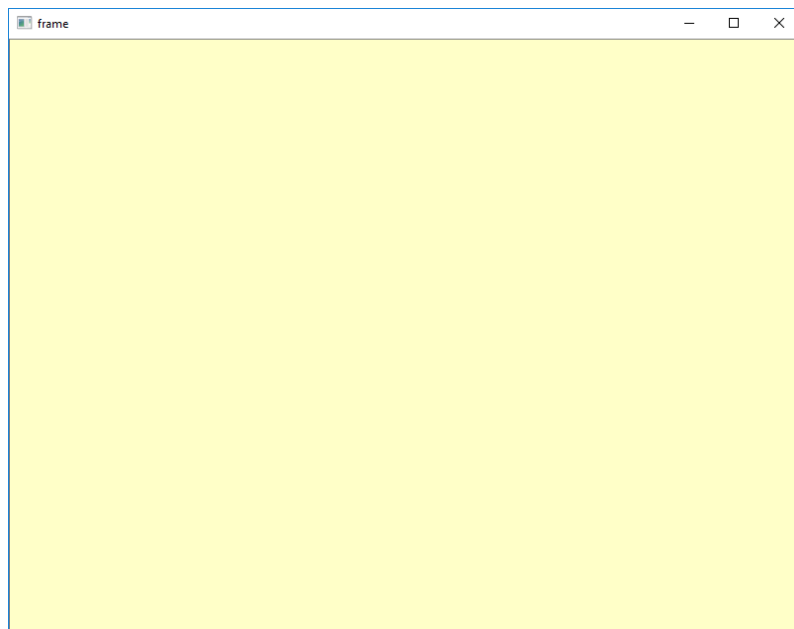


Figure 4.5: makevideo 檔初始

並且當使用者在打字的介面開始打字時，此框便也會開始動作。前面也提到此數據也是經由 firebase 傳送的數據進行更新；也就是當使用者在一邊打字時，此 makevideo 檔也會跟著觀察更新使用者所打的字在框框上。

(圖 4.6)

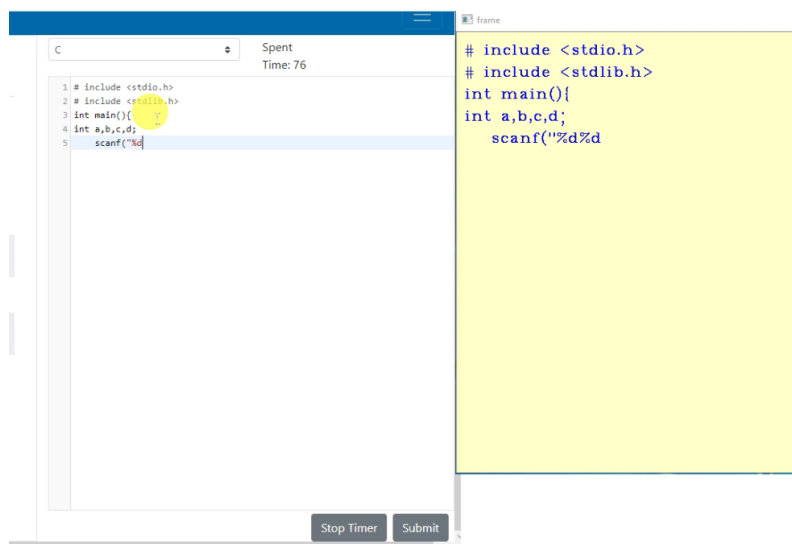


Figure 4.6: makevideo 檔與網頁介面

在結束擷取之後，便會把這些被觀察的使用者動作輸出成影片並且存取為一個 AVI 影片檔。

- 影片動作與實際動作:

makevideo 檔輸出後的 AVI 所發生的動作就跟執行 makevideo 檔所呈現的框裡的動作一樣。在此想把輸出的 AVI 影片檔與真正打字的影片檔做比較，於是在打字的時候一邊錄了螢幕錄影。

(圖 4.7) 為螢幕錄影的影片與 makevideo 所產生的影片比較。

1	時間	螢幕錄影	makevideo
1	0:00:08	1 # include	# i
2	0:00:10	1 # include <	# include
3			
4	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
5			

Figure 4.7: 螢幕錄影與 makevideo 比較

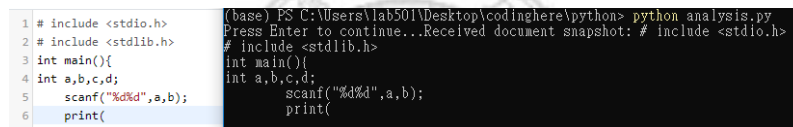
可以看出影片的動作會慢於實際的打字速度，因為在取值時有設定固定的秒

數，因此若是使用者不間斷的打字必然不能完整呈現，所以 makevideo 檔所產生的輸出影片與實際上螢幕錄影影片是延遲的。

但是此種方法確實可以很好的模擬使用者在打字時後的種種動作。

- python 檔:

在前面架構與規劃有提到，不只是可以呈現網頁上的數據圖形與影片，也多撰寫一個 python 程式同樣經由 firebase 把使用者的打字狀況抓下來。抓下來的目的是因為之所使用的分析程式大量使用 python。(圖 4.8)



```
1 # include <stdio.h>
2 # include <stdlib.h>
3 int main(){
4 int a,b,c,d;
5 scanf("%d%d",a,b);
6 printf(
(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);
printf(
```

Figure 4.8: python 抓下的文字

4.2 python 的討論分析

因本研究所寫的 python 程式碼可以指定實驗數據的數量進進行製作與分析，在此便指定 10 組與 30 組結果。

個別的圖形與數據

每一組數據便會產生 (圖 4.9) 的一組圖形，在此便不把全部圖形列出，以下只列每次產生的數據最基本的折線圖，共 10 組。(圖 4.10)

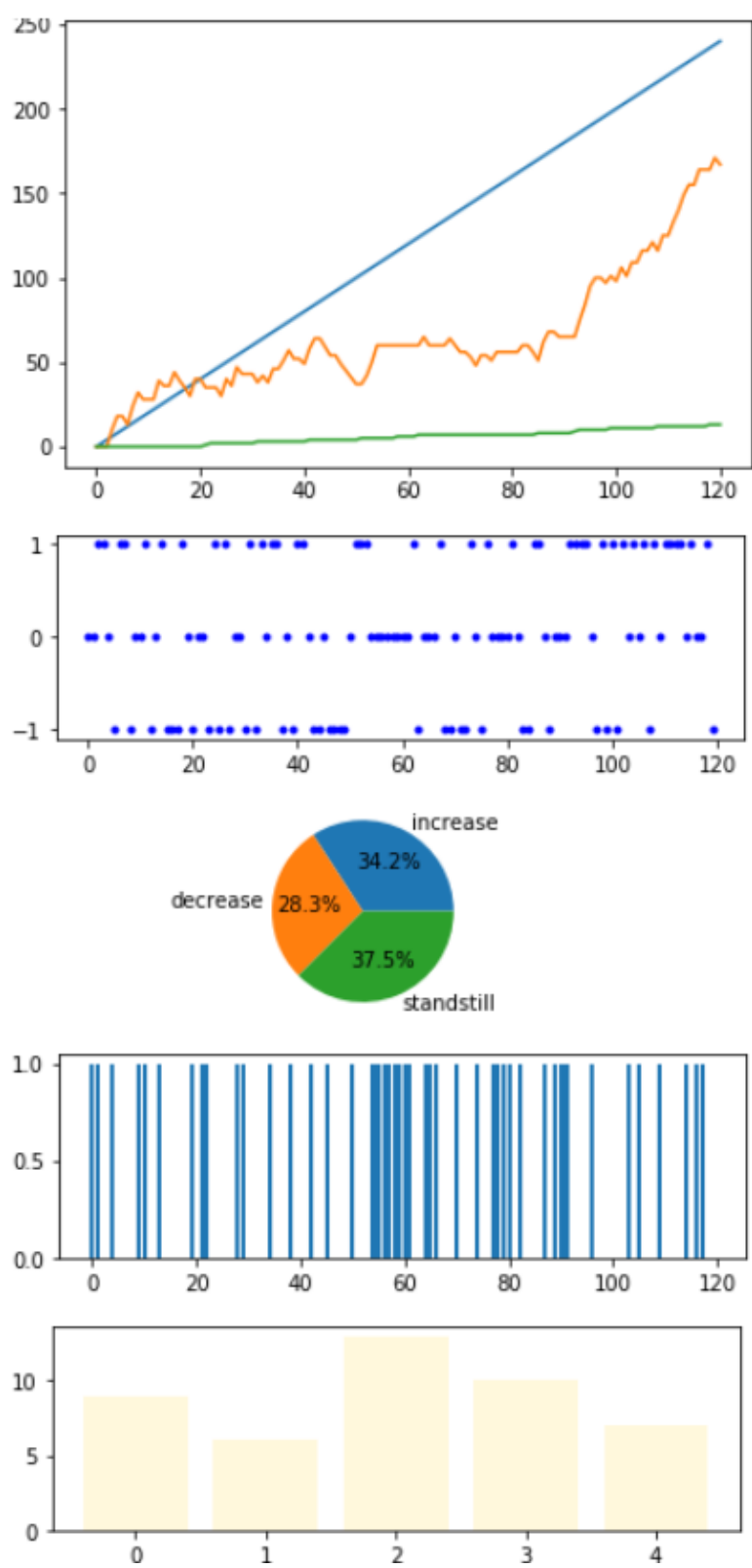


Figure 4.9: 圖形組

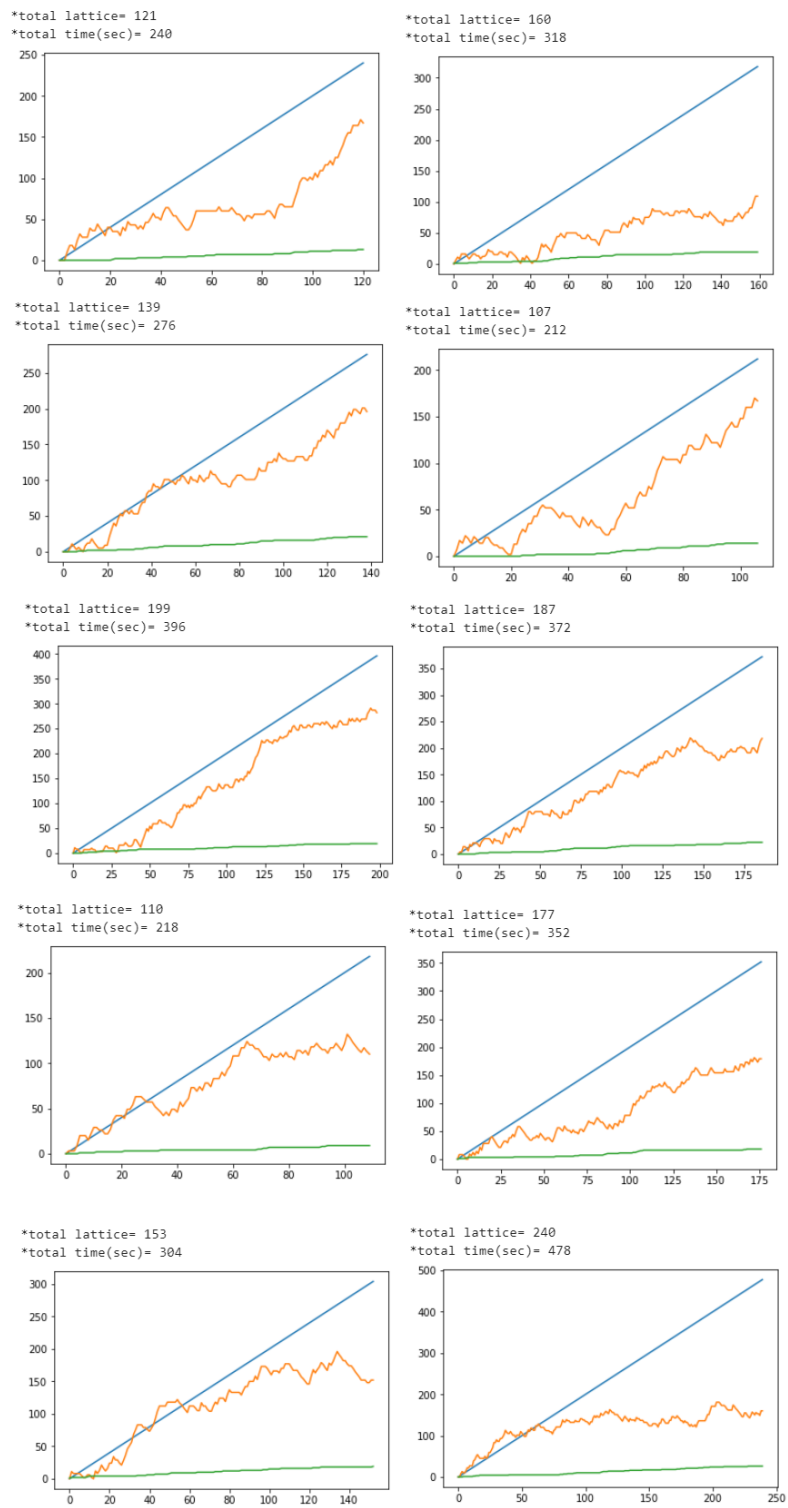


Figure 4.10: 10 組折線圖

統整的表格數據

把全部的數據統整起來製程表格，表格也是使用 python 在 colab 可以輸出顯示結果。

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 html 網頁偵測部份

- 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。
- 在網頁上只擷取了字數、行數與時間，如可以增加更多的鍵盤偵測將會使系統更加完善，此處為能多加以改良完善的地方。
- 在影片輸出部分，本研究比較了經由網頁數據形成的影片與實際使用螢幕錄影的效果，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網頁進行的改動。

4.3.2 python 分析部分討論

- 進行分析的部分最大的難處就是缺少實際數據，因此在本文中程式去模擬實際的數據。但是因在此使用大量的 random 函數與諸多限制，如為了使程式撰寫方便設的上下界線使數據密集化等。模擬的數據理所當然地不及實際數據來的真實與客觀。
- 再討論數據停頓的密集程度的地方(圖 3.16)，我所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

本研究提出了對前端網頁文字及時偵測與分析的方法，屆時有非常多地方需要更多的功能亦或完整化，如加入更多鍵盤與滑鼠上的偵測、爬蟲概念的使用；或是收集真實的數據做完整的系統分析而不是使用模擬數據。未來或許可以把分析部分直接放入網頁中同步分析而不需要把數據截取出來討論，也就是完整地整合全部的流程系統至同一平台，如此讓偵測與分析最佳化。

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.
- [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.