# 如何製作一個 Amazon Mechanical Turk 需要的外部網頁

**黃挺豪 Ting-Hao (Kenneth) Huang**
Carnegie Mellon University
http://www.cs.cmu.edu/~tinghaoh/
tinghaoh@cs.cmu.edu

Last Update: 2016/09/11

## ● 這是什麼？

這是一份簡短的中文教學，教你如何做一個外部網頁（external web page），讓你的「熟悉 Amazon Mechanical Turk （mturk）的朋友」可以快速幫你把做好的網頁放上 mturk 、讓 workers 開始標 data。

這個教學只會觸及最低限度的 mturk 知識，我也會盡量避免使用 mturk 的術語。如果你想做一些比較複雜的功能（例如寫程式自動發 bonus 給 worker），這份文件不包含那些進階的內容。

歡迎寄信給我提問、指正或補充。我也可以把你加入共同作者。（看的人太多，我先把 comment 關掉了。）

## ● 這份文件的目標

在「不使用」database 與動態網頁（例如 PHP[1]）的情況下，製作「可以直接放上 mturk 去讓 worker 開始做」的網頁。

**在開始之前，請先確定你想標的 data 「沒辦法」用 Amazon Mechanical Turk 提供的網頁模板來標（可使用現成的 mturk command line tool[2] 來上架，不需要自己做網頁、也不太需要寫**

[1] 這份文件中講的「PHP」是個代稱，可以用任何你喜歡的動態網頁語言去替換。
[2] https://requester.mturk.com/developer/tools/clt

code ) ──唯有 mturk 提供的現成網頁模板無法滿足你的需求的情況下，你才需要自己做網頁來標 data[3]。

這份文件假設讀者會**至少一種程式語言，有基礎的網頁製作經驗，了解 HTML 和 Javascript（這份文件中的 example code 都有使用 jQuery）**，並且擁有**幫 Javaceript debug** 的基本能力。

如果你是稍有經驗的 mturk 使用者，或許可以直接跳到「Best Practice」與後續「如何 hack 你的 global constraint」這兩個章節。

## ● 你需要一個「有 https」的網頁空間

我假設你那位「熟悉 Amazon Mechanical Turk 的朋友」會負責處理所有 mturk 相關的事情，包括把 task 上架、付費、收結果，以及把 worker 寄來的 email 轉給你。所以這些我都不多做介紹。

儘管如此，請注意，mturk 要求所有 task 的外部網頁都要放在 **https** 的網址下。

所以，你必須先**找到一個「有 https 的網頁空間」**，可以讓你把做好的網頁放上去。如果你使用 mturk 提供的網頁模板，它會自動幫你 host 在 Amazon 的主機上；但現在我們談的是外部網頁（external HIT），你就必須提供自己的 URL[4]。

再來，請注意「任何」你的網頁所使用的 resource（包括任何圖片、Javascript、CSS 檔案等等），如果使用外部連結，一定要用 **https** 網址[5]。例如：

```
<script type="text/javascript" src="https://some.url.haha/crowd.js"></script>
```

[3] 如果你希望所有步驟都在網頁上操作，不想耗費時間處理比較低階的細節（例如使用 command line tool），你可以考慮使用 CrowdFlower。

[4] 很多時候「有 https 的網頁空間」不是免費的。幾乎每個 mturk 專家都有自己的一套辦法去尋找可以用的 https 網路主機。問問你朋友吧。

[5] 最容易被忽略的是 favicon。有時 debug 了半天還是有問題就是 favicon 的檔案沒放 https。

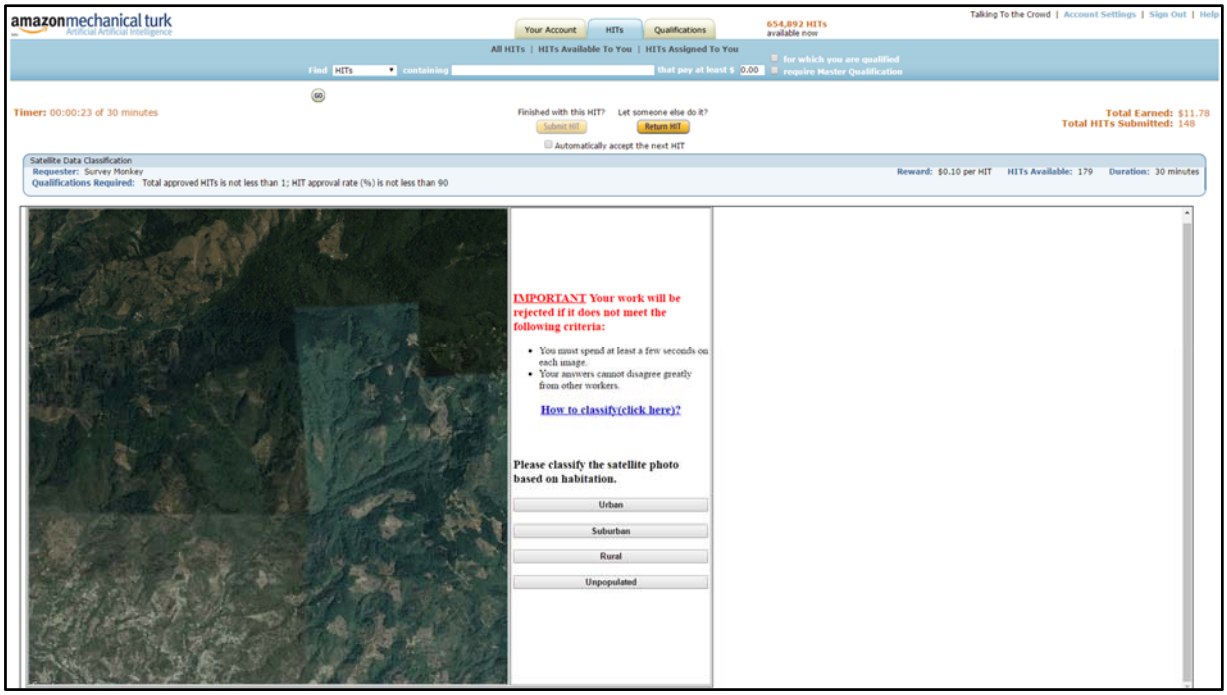
- **Mturk 是怎麼運作的？**

Amazon Mechanical Turk （mturk）是一個 crowdsourcing 平臺，案主可以在上面發佈 task、workers 也可以在上面搜尋自己想要的 task 來做。Mturk 上所有 task 都是以「網頁」的形式呈現、也都是在網頁上完成。每個 task 都有標題、描述、關鍵字，以及價格。

Workers 找到自己想要做的 task、「接下」（accept）task、做完 task、最後 submit。等案主檢查完 worker 的工作，覺得沒問題、核可以後，worker 就可以透過 mturk 從案主手上收到錢（當然 mturk 是會抽手續費的）。

Mturk 展示你的 task 的方式是：**將你的外部網頁用 iframe[6] 內嵌在 Mturk 的網頁上。原則上 worker 也是在同一個 iframe 裡做你的 task、並且 submit 給 Mturk。**

Worker 看到的介面如下圖。中間那個大框就是內嵌著你外部網頁的 iframe（高度可以在把 task 上架的時候設定）：




Worker 在你的網頁上做完你的 task 之後，網頁端 task submit 的流程如下：

[6] 大部分的網頁功能不會因此被影響，但還是有些比較罕用的功能是不能穿過 iframe 的。例如你不能動態去改網頁的 title。

1. 你的網頁上通常會做一個按鈕（submit button）讓 worker 按。

2. 按下去以後，你的網頁會 **submit（就是 HTTP 的 post）一個 form （就是 HTML 的 form）到 mturk 規定的 URL** 去。

3. Mturk 收到你的 form 以後，會自動核對 form 裡面的各個 ID，如果正確的話，它就會判定這個 task 被「完成」了。

如果你有用 database，一般來說你會讓在 worker 標的過程中直接把存進你的 database。你可以把必要的 ID 傳回給 mturk 就好（form 裡面除了 ID 之外就是空的）；

如果你沒有用 database，**你也可以把 worker 標的 data 用 Javascript 放在上面講的 form 裡，一起傳給 Mturk**。Mturk 會把收到的 data 存在它的 server 上。你那位 mturk 專家朋友會知道怎麼把 data 取下來（會是個 XML 檔或 CSV 檔）。

- ## 你需要知道的四個 ID

上面講到 mturk 會去檢查你的網頁傳給它的 form 裡面的 ID。那些 ID 是 mturk 生成的，你不能自己亂編。

把 task 上架的時候，只要你網頁本來的 URL 就好；

但上架後，**mturk 會自動在你本來的網址後加上下面這四個變數**，再嵌進 iframe 裡：

1. **workerId**：現在這個 worker 的 ID（通常是大寫 "A" 開頭），每個人都不一樣
2. **hitId**：一個 task（在 mturk 上叫作 **HIT**, Human Intelligence Tasks）會有一個 ID
3. **assignmentId**：同一個 task 可以有多個 assignments，每個 assignment 都會發給不同 worker 做。每個 assignment 都有一個 ID。
4. **turkSubmitTo**：你最後要把 form submit 去的 URL

像是下面這樣：

https://www.my.page/index.html?**workerId=XXX&hitId=XXX&assignmentId=XXX&turkSubmitTo=XXX**

如果你的 task 從頭到尾都在同個 html page，那沒問題；

如果你的 task 從頭到尾有很多個 page 跳來跳去，在過程中「每一次跳頁」都要保留這四個變數、而且只能用「接在網址後面」（get）的方法傳。不然最後會 submit 失敗。

在任何時刻你如果需要知道這四個變數的值，你都可以用簡單的 Javascript 把這些值抓下來。下面這是我最常用來抓網址後面變數的 Javascript function：

```
//function for getting URL parameters
function gup(name) {
    name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
    var regexS = "[\\?&]"+name+"=([^&#]*)";
    var regex = new RegExp(regexS);
    var results = regex.exec(window.location.href);
    if(results == null)
           return "";
    else
           return unescape(results[1]);
}
```

你可以直接像下面這樣 call 這個 function：

```
var nowWorkeId = gup("workerId");
```

- **假設你的 worker 網頁已經大致做好了，請依照下面的方法把你的網頁「mturk 化」，讓你朋友可以直接幫你在 mturk 上架**

首先，如果你的網頁裡本來沒有 form，請把下面這個 form 放進你的 html 裡：

```
<form id="mturk_form" name="mturk_form">
       <input id="mturk-assignmentId" name="assignmentId" type="hidden">
       <input id="submitted-from" name="submitted-from" type="hidden" value="crowd-page">
       <input id="form-session-id" name="form-session-id" type="hidden">
</form>
```

如果你已經有一個 form 了，請至少把下面這行加進你的 form 中[7]：

```
       <input id="mturk-assignmentId" name="assignmentId" type="hidden">
```

[7] form 中加了其他兩個不重要的 <input> 是因為有謠言表示：form 如果只有 assignmentId 一個 input 卻沒有其他 input，會被 mturk 拒絕。（但我實測的結果應該不是真的。）

然後當 worker 完成你的 task、按下你做的 submit 按鈕，你可以 call 下面這個 function 來 submit。這個 function 做的事情基本上是：把必要的資訊（主要是 assignment ID）塞進 form 裡，然後把 form submit 到 “turkSubmitTo” 這個 URL。

```
function submitToTurk(){

        if(gup("assignmentId")!="") {

         var jobkey = gup("assignmentId");
         if(gup("hitId")!="") {
                jobkey += "|" + gup("hitId");
         }

         if(gup("assignmentId") == "ASSIGNMENT_ID_NOT_AVAILABLE") {
                $('input').attr("DISABLED", "true");
                _allowSubmit = false;
         } else {
                _allowSubmit = true;
         }
         $('#mturk-assignmentId').attr('value', gup("assignmentId"));
         $("#mturk_form").attr('method', 'POST');

         if(gup("turkSubmitTo")!="") {
             $("#mturk_form").attr('action', gup("turkSubmitTo") + '/mturk/externalSubmit');
         }

    }

        $("#mturk_form").submit();

        /*For some reasons you need this*/
        return false;

}
```

如果你沒有使用 database，在 submit 前別忘了要先把 worker 標好的 data 用 Javascript 收進 form 裡。這樣 submit 之後，標好的 data 在 mturk 的 server 上就可以看得到。（<input> 必須要有 name 跟 value 這兩個 attribute，只有 id 是不行的。）

- **你的 Task 的「預覽畫面」**

在 mturk 上，worker 可以「預覽」你的 task，然後他「接受」（accept）你的 task 之後，才實際開始進行。雖然不是必要的，但你可以讓 worker 看到的「預覽」畫面和實際的工作畫面不一樣。

判斷方法是：如果 assignmentId 的值是 “ASSIGNMENT_ID_NOT_AVAILABLE”，就表示有人在預覽你的網頁。如果是其他值，表示是實際開始 work 了。

我都是用類似下面的這個方法判斷的：

```
var _allowSubmit = false;

if(gup("assignmentId") === "ASSIGNMENT_ID_NOT_AVAILABLE") {

        $("body").empty();
        $("body").css("background", "white");

        var noteDiv = $("<div class='container'><h3>Please accept the HIT.</h3></div>");
        $(noteDiv).css("margin-left", "auto");
        $(noteDiv).css("margin-right", "auto");

        $(noteDiv).css("text-align", "center");

        var img = $("<img></img>")
        $(img).attr("src",
"https://c2.staticflickr.com/4/3665/11276962563_8fc141d195.jpg");
        /* image from https://flic.kr/p/ibvo62 */
        $(img).css("width", "600px");

        $(noteDiv).append(img);

        $("body").append(noteDiv);

        $('input').attr("DISABLED", "true");

        _allowSubmit = false;

} else {
         _allowSubmit = true;
}
```

這「不是」必要的。只是如果 worker 做完你的 task 之後才發現他忘了 accept，以至於整個要重做的話，可能會不太開心。

## ● 其他做 mturk 標記網頁需要注意的細節

1. 請注意世界各地的 worker 是用各式各樣不同的瀏覽器在看你的網頁。所以請至少用兩種瀏覽器（我是用 Chrome 跟 FireFox）看一下你的網頁有沒有問題。

2. 簡單的 input 檢查一定要做。包括空白輸入、過短的輸入都應該直接 alert() 不讓他 submit。一個比較少人會注意的是 time lock，你可以設定一個合理的最短 task

completion time，例如十秒──比這個短一定是亂做的──你可以先鎖住你的 submit 鍵，倒數十秒後才 enable 它。

3. Worker 通常不會詳細閱讀過長的 instruction。很重要的事情最好寫紅字或用其他提示的方式多講幾次。或者要有好的 interface 設計讓他們比較不容易犯錯。

4. Worker 介面設計的精神是「**讓『壞 worker』搗亂或偷懶所需的 effort 和『好 worker』乖乖做 task 所需的 effort 差不多，甚至更多**」。例如你叫他寫一個句子，你可以簡單地用空白去斷 string 然後去檢查至少要有三個字、寫 code 擋他複製貼上、再加上十秒的時間鎖，讓他「趕快結束好收錢」的最輕鬆的方法就是「乖乖寫一個正常的句子給你」。這樣通常就可以擋掉大部分的壞 worker。像我老闆 Prof. Jeff Bigham（http://www.cs.cmu.edu/~jbigham/）說的：如果 worker 做的事情不如你的預期，通常是你的問題、不是他的問題。

## ● 我有蠻大一批 data 要標，最好的做法是什麼？（Best Practice）

**如果可能**的話，以下是我認為最簡單的 practice，可以避開大部份可能會發生的問題。

以下我用「你有 100 張圖片要標」[8]來當範例：

1. **將你的 data 切成幾份「邏輯上彼此獨立」、大小也差不多的小 batch，並且編號。**例如你有 100 張圖要標，最簡單的方法就是分成 100 份，一張圖一份（1-100 號）；或是 50 份，兩張圖一份（1-50 號）。以下「一份」都表示「切完之後的一份 data」。

2. **從上面切好的 data 裡，取「一份」出來，然後根據這份 data 做你的標記網頁。**這裡說的不是「一張圖」，而是「一份」。例如你每一個 task 希望 worker 標兩張圖，那「兩張圖」就是你的「一份」。然後依照前面我說的方法，去做一個「標記這兩張圖的網頁」。

3. **記得在你的最後要 submit 的 form 裡加一個 input 存「目前這份 data 的編號」，**一起傳到 mturk 上。日後可以省去非常多資料比對的麻煩。

[8] 吾友 Janet 提醒我：只是標個圖片 label 的話不用自己做網頁。答約：我想的是那種要在圖片上框 bounding box、需要特殊 interface 來做的標記。

4. 做好之後，用你最順手的程式語言，寫一個程式，根據你的 data，**把你做好的這個網頁當成模版，自動生成「全部的標記網頁」檔案**。你有 100 張圖、2 張分成一份，那就是生成 50 個幾乎一樣的網頁檔案，唯一不同的就是每個網頁裡面要標的兩張圖片不一樣（去改 <img> 裡面的 src），以及與這兩張圖片有關的資訊（例如圖片描述）也會不一樣。理論上你只要輸出 HTML 檔案就好，每個網頁的 .js 和 .css 通常都是一樣的。

5. 把你做的**所有網頁都上傳到一個 https 的網站**上。這樣你就有 50 個不同的 URL 了（例如 https://my.home/img-label-1.htm 到 https://my.home/img-label-50.htm ）。

6. **在 mturk 上，用每個 URL 發一個 task（HIT）**。如果每一份 data 你需要多個人標的話，發 task 的時候就指定多個 assignment 就可以了。例如你有一百張圖、兩張一份、每一份要五個人標，那你就是發 50 個 task（HIT）、每一個 task 有 5 個 assignment。

7. 最後，data 標完，（你的 mturk 專家朋友）用程式去 mturk 的 server 把標好的 data 收下來。

這方法看起來很蠢，但它完全避開了以下幾件事[9]：

1. 不需要使用 database。
2. 不需要找到可以放 PHP 的網頁空間。任何學校提供的靜態網頁空間（就是你放你學校個人網頁的那個地方），只要有 https （通常有）就可以了。
3. 不需要自己寫 PHP 去讀檔案。
4. 不需要自己寫 PHP 去寫檔案（以及處理可能的 conflict）。
5. 不需要用 URL 傳任何資訊[10]。

---

[9] 缺點顯然是每次你想做一些小改動，那 50 個網頁全部都要重新生成。如果你每個網頁需要讀入的 data 量不大，可以用 URL 傳；你也可以寫一個網頁模板，然後寫一個小 PHP 根據傳入的 ID 去某個檔案裡動態讀入資料；或者你可以把全部的資料都印在 Javascript 裡，直接放到 client 端去。這些都是比較靈活的方法，也比較好 scale（例如你可能每週都有新的 data 進來需要標）。但**「靈活」與「容易 scale」都是要用「robustness」去換的**。你每多加入一個動態的 componenet，就多一個會壞掉的地方。

[10] 這個優點比較不重要。但 URL 有「最大長度限制」，你要傳的資料很大時有可能會超過。

6. 最重要的是，**不需要自己寫 code 去動態指派 task 給 worker**。

如果可以的話，「**盡量避免自己動態指派 task 給 worker**」（Try to **avoid creating your own task routing system** that dynamically assigns tasks to workers）。

MTurk 背後是強韌的 AWS，而且它早從 2005 年[11]就開始指派 task 給 worker，它的 task routing system 至今已經非常成熟了。你用自己的 database 實作很難做得比它快、比它 robust。動態指派 task 會有非常多需要處理的細節[12]，你還得照顧 database 的效能[13]—強烈建議**除非必要，不然不要自己實作 task routing system**。

**你的目標是「收到好的 data」，而不是「讓網址看起來很聰明」**。（You are here to collect good data, not to make the URLs look smart.）要讓一個東西看起來很聰明通常得投入相當的時間與精力，而且標 data 又常常是一次性的工作[14]，標 data 使用的網頁在 data 標完後常常就不會再用了。在一頭栽進去前請務必想清楚，有沒有其他更簡單粗暴省時的方法（尤其當你對網頁開發不熟悉的時候）。

## ● 承上，有沒有可能我的 data 切不開？（How to Hack Your Global Constraints）

**有。**

有時 data 的標註之間彼此有「**相依性**」，例如你得先標第一層，我再根據第一層的結果標第二層。我實際做過的例子是微軟的 SIND Dataset（http://www.sind.ai/）。第一階段標完之後，標好的結果才送去第二階段標。這種情況下除非量很小或你不急，可以等第一階段全部標完、data 收下來自己整理一下再手動上架第二階段，否則很難避開自己寫 task routing。但這種情況比較少見。

---

[11] https://en.wikipedia.org/wiki/Amazon_Mechanical_Turk

[12] worker 拿了 task 卻遲遲不做完，多久之後可以開放給別人做？如果有個 assignment 被拿去做之後很慢才回來，那個 assignment 已經被我分給其他人做完了怎麼辦？一個人不能做同一份 data 兩次.....

[13] data 很大要 query 會慢、connection 一次開太多也會慢甚至會 crash、Sqlite 無法 scale up.....

[14] 2015 年 HCOMP 的 panel 上，某位來自龍頭網路公司的人士表示：在我們公司裡，大概 90% 的 crowdsourcing task 都是一次性的。

更常見的情況是你想標的 data 有**「global constraints」**。也就是說，你有一些條件是要跨「好幾份 data」去檢查的，單獨只看一份 data 沒辦法判斷。換言之，你無法做到前面所說的，每份 data 之間「邏輯上彼此獨立」。這種情況下你通常需要自己寫一些 code 去 globally 管理你的 constraint。

—其中最常見的情況就是**加在「worker」身上的 global constraint**。

例如你想限制**每個 worker 「只能最多標三份」**，再多就不行了。那這種情況下你就只能寫 code 每一份去檢查誰標了哪些，然後即時去 block 標超過三份的 worker；

又例如你的「同一張圖片」有兩種型態：彩色的和黑白的。彩圖和黑白圖會被分在不同的兩份 data 裡，但你還是**不希望同一個 worker 重複標一樣的圖片兩次**（即使一次是彩色一次是黑白的）。所以這種情況下，你就得知道誰標了哪張圖，然後寫 code 即時去阻止 worker 標「他標過的彩圖的黑白版本」（或是反過來）。

這些情況看起來無法避免要自己寫 code 去動態分派 task 給 worker，但使用一點巧思、用一些小 trick，有時只要寫一點點動態網頁（例如寫個 PHP 去讀檔）甚至只用靜態網頁，就可以滿足你的需求。

以下分別介紹如何在「靜態網頁」、「動態網頁」以及「自己寫的 task routing system 」中做一些 hack 來處理 global constraints。

第一，以**靜態網頁**來說，如果 constraint 的細節是可以事前得知的，至少有兩個地方可以讓你**偷偷 encode 你的 global constraints**：

(1) **網頁裡。**有些 constraint 是你在自動生成一大堆標記網頁的時候就知道了，你可以想辦法把那些資訊寫進你生成的網頁的 HTML code 裡，再藉由 Javascript 把他讀出來。

例如你知道這個 task 不能給特定哪個 worker 做，你可以把他的 worker ID 用 <input type=”hidden”> 藏在網頁的 html 裡，再用 Javascript 把值讀出來，跟目前頁面上的 gup(“workerId”) 做比對，如果是同一人的話，你就 call 個 empty() 把整個畫面洗掉，然後放一個「**Sorry, but this HIT is not available. Please return this HIT.**」的標示即可。

(2) **網址上。**有些 constraint 你可以在「把 task 上架」的時候知道，那你可以用「網址」把資訊 encode 進去。

例如你在 mturk 上架一個 task 時就已經知道你要 block worker A，你就把本來的 URL 後面接一個 "**?blockWorker=A**"，再用 Javascript 的 gup() 把他讀出來比對。

以下是我寫的一個小例子，在 Javascript 裡用 gup() 去取網址變數指定的一串 worker ID：https://github.com/windx0303/Compensate-MTurk-HIT

第二，有時你無法避免寫**動態網頁。**例如前面舉的那兩個例子（每個 worker 最多只能標三份、一張圖片有黑白有彩色），我就想不到「只用靜態網頁」的簡單 hack[15]。你可以寫一段 PHP 去讀寫檔案或資料庫來記錄哪個 worker 標了哪些 data，然後在 client 端 block 已經做過三份 data 的 worker。

靜態網頁上的所有資訊都必須在「事前」就知道，所以你有時還是要靠讀寫檔案或資料庫來處理「一定要知道此刻所有 task 的狀態」才能解的 global constraint。

第三，更有甚者，有時你無法避免要**自己寫 task routing system**。當你的 global constraint 比較複雜（例如「兩個 worker 的答案相同的話就不用問第三個 worker 了，問了三個 worker 答案都不一樣的話就問第四個」；又例如「如果前三份 data 都是兩票以上同意的話就不要發第四份 data」），事前無法確定 data 的份數與 assignment 數，只好等 worker 上門來了再自動去判斷現在要把哪個 task 交給他做。

即使如此，我還是鼓勵你**善用巧思**去避免「完全靠自己」動態指派 task 給 worker。你可以**利用 mturk 本來的 task 指派功能來處理比較單純的狀況**，等到情況變複雜了再用自己的 code 去指派 task。例如每個 URL 都先指向一個預設的 task，當預設的 task 的狀態發生異動（例

[15] 有複雜的 hack：你可以寫一個程式，依序每十分鐘自動上架一份 data。例如先上架第一份、十分鐘後上架第二份、二十分鐘後上架第三份。如此一來你可以賺到「時間差」，可能在上架第三份 data 的時候、第一份已經有人做完了，這時你可以把新的訊息 encode 在第三份 task 的網址裡面（例如：把已經做滿三份的 worker ID 加進網址變數裡）。如此一來你的網頁就還是靜態的，卻又可以稍微顧及一下整體的標記狀況。這個 hack 我很早就想到了，但始終沒有自己實作過（我都乖乖做動態網頁），如果你試了歡迎寄信告訴我：1) 你想標的 data 是什麼樣子、2) 你如何實作這個 hack、3) 效果如何。

如預設 task 已經不用做了），再去計算要改派哪個 task 給上門的 worker。我實作 Chorus 中的 task routing 時就是這麼做的。

標 data 總有一些很基本的**假設**，例如「每份 data 至少要有一個 worker 標」，可以利用這些假設來節省自己的計算資源。

- **網頁全部弄好了。每個 task 應該付多少錢？**

我假設你的 mturk 專家朋友會處理 mturk 的一切技術問題，你只要私下把錢給他就好。所以剩下的問題就是討論「**每個 task assignment 應該付多少錢給 worker**」。

以學術圈的 project 來說，在 mturk 上「比較常見」的薪資大概是**每小時 $6 到 $8** 左右（美金），但「合乎倫理」（ethical）的薪資是**每小時 $10**。[16]

估計每個 task 時間的簡單方法是：**「你自己」先做幾個你的 task、算算你平均要花多久，然後把那個時間乘以二或三**，就是合理的 worker 時間。你非常了解你的 task，你當然可以做得很快。許多 worker 是第一次做你的 task，會做得比你慢很多。

這估計方法只適用於時間較短（例如三、五分鐘以內）的 task。如果你的 task 是十分鐘以上的長 task，你可能要找你朋友做做看，看他們大概都花了多長時間。另外，根據我的經驗，時間很長的 task（例如二十分鐘以上）通常會稍微多付一點錢，worker 才願意留這麼久。

另外，「用完成 task 所需要的時間來估計價格」只適用於普通的 task。如果你的 task 需要 worker 有特殊經驗（例如 survey 類的 task 會要求符合某些條件的受試者）、特殊語言能力（例如西班牙文母語者）、task 本身比較複雜（例如標記一段影片中的多個 time span）、或要求 worker 分享一些較為私人的資訊，通常得付更多錢。

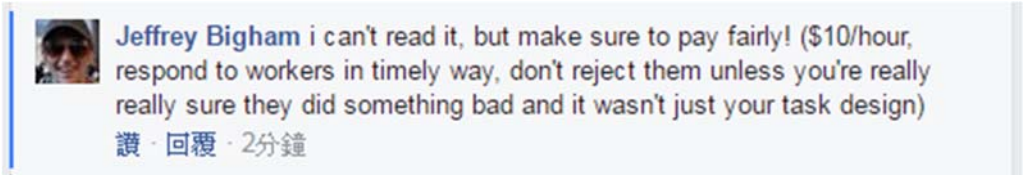
請注意以上的價格討論「不包括」 Mturk 抽的手續費。目前 Mturk 抽的手續費是 20%，如果你的一個 task 有 10 個或以上的 assignments，那 Mturk 會抽 40%。換言之，你發了一個 $1 的 task，實際上你要付給 Mturk $1.2 或 $1.4。

---

[16] MTurk 近年來逐漸排除海外 worker，所以這裡的 ethical pay 是以美國的最低工資為標準來建議的。可以參考這裡：http://wiki.wearedynamo.org/index.php/Guidelines_for_Academic_Requesters

以下是一份給學術圈的人看的「在 mturk 上發 task」的指導手冊，裡面有許多更深入的內容：http://wiki.wearedynamo.org/index.php/Guidelines_for_Academic_Requesters

來自 Prof. Jeff Bigham 的溫馨提醒：




記得**快點付錢**，盡量避免拒絕（reject） worker 做好的工作，即使品質很差—那通常是你網頁做得太爛，不是 worker 的問題。

你想做 data filtering 的話，先把 data 收下來，自己回家在後端做就好。