COMP 4801 Final Year Project
Detailed Intermediate Report

Project Title:
An online food catalogue based on open crowd sourcing

Supervisor:
Dr. T.W. Chim

Group Members:
Lai Cheuk Hin (3035186152)
Leung Lok Ming (3035125031)

Date of Submission:
21 January 2018
Abstract

Online food catalogues are widely used by the citizens in Hong Kong. These catalogues provide search engine for the users to search information about the restaurants. However, the processes of gathering information usually rely on the cooperation between the platform and restaurants. Meanwhile, for those platforms that offer functions for users to upload photos, the information in photos are not utilized as there are no analysis about the images. These practices are not effective enough for handling large amount of data.

To resolve the problems above, this the project aims to provide an online platform for people to upload information and photos of the restaurants in Hong Kong. The system provides increasingly accurate searching results by applying the technologies of photo recognition and machine learning based on the data collected.

This paper studies the strengths and weaknesses of the existing online food catalogues on the market, and introduces the design, implementation and technologies used for the application to be developed in the project. At the end of this paper, it reports the current status of working process and the future planning of this project.

Acknowledgment

We would like to express our most sincere gratitude to our supervisor Dr. Chim T W, for his guidance and insightful comments during the process of the final year project.
Table of Contents

Abstract .............................................................................................................................................. 2
Acknowledgment ................................................................................................................................. 2
List of Figures ....................................................................................................................................... 5
List of Tables ....................................................................................................................................... 6
Abbreviations ....................................................................................................................................... 7

1. Introduction ..................................................................................................................................... 8
2. Previous Works ............................................................................................................................... 10
3. Deliverable ....................................................................................................................................... 11
   3.1. Function List ............................................................................................................................... 11
       3.1.1. Membership System ............................................................................................................. 11
       3.1.2. Restaurant Searching .......................................................................................................... 11
       3.1.3. Menu Viewing/ Editing ....................................................................................................... 12
       3.1.4. Food Photo Uploading ....................................................................................................... 12
       3.1.5. Restaurants/ Dishes recommendation ............................................................................... 12
4. Methodology ...................................................................................................................................... 13
   4.1. Application Design Pattern ....................................................................................................... 13
   4.2. Programming Language ........................................................................................................... 14
   4.3. System Architecture .................................................................................................................. 15
   4.4. Server Hosting ........................................................................................................................... 17
   4.5. Client-Server Communication .................................................................................................. 18
   4.6. Optical Character Recognition ............................................................................................... 19
   4.7. Deep Learning .......................................................................................................................... 20
5. Current Status ................................................................................................................................... 22
   5.1. Back-end Development ............................................................................................................. 22
   5.2. Computer Vision ....................................................................................................................... 22
       5.2.1 Optical Character Recognition (OCR) .................................................................................. 22
       5.2.2 Photo Recognition ............................................................................................................... 24
   5.3. Machine Learning ...................................................................................................................... 25
6. Limitations and Recommendations ................................................................................................. 26
   6.1. Inaccurate Data Entry ............................................................................................................... 26
   6.2. Improper User Behavior ........................................................................................................... 26
   6.3. Implementing photo recognition function ................................................................................. 27
   6.4. Implementing optical character recognition function ............................................................ 27
7. **Working Plan** ......................................................................................................................... 28
   7.1. Work Accomplished to Date ................................................................................................. 29
   7.2. Future Planning...................................................................................................................... 29
8. **Conclusion** .............................................................................................................................. 30
References ........................................................................................................................................ 31
List of Figures

Figure 1: A screenshot of the photo upload page in Openrice..........................10
Figure 2. The UI of the restaurant searching function.................................12
Figure 3: Structure of MVP pattern................................................................13
Figure 4: System architecture .......................................................................15
Figure 5: Comparison on the price of two cloud platforms..........................17
Figure 6: The accuracy rate of using Traditional CV and Deep Learning.........20
Figure 7. A part of the result of the OCR function.......................................23
Figure 8: Timeline of 3 phases in the project..............................................28
List of Tables

Table 1: CombSort Strict CPU test on 3 different server-side languages ..................15
Table 2: Performance test on 4 different APIs .........................................................19
Abbreviations

API: Application Programming Interface
OCR: Optical Character Recognition
UI: User Interface
MVC (design pattern): Model-View-Controller
MVP (design pattern): Model-View-Presenter
CPU: Central processing unit
GPU: Graphics processing unit
AWS: Amazon Web Services
GCP: Google Cloud Platform
REST: Representational State Transfer
SOAP: Simple Object Access Protocol
1. Introduction

Dining is one of the most important concerns of the citizens nowadays. There are many online food catalogues on the market that allow people to search for foods and restaurants. However, most of these food catalogues relying on the cooperation between platform and the restaurants, all the information need to be updated manually. This kind of business strategy requires lots of manpower, which is not effective to handle a large amount of restaurant data in Hong Kong. Even though some of the catalogues include functions for users to upload contents like photos of food and menu, these data will only be uploaded to the server without any analysis on the images. As a result, the information contained in the photo are being ignored and wasted.

This project aims to solve the problems mentioned above, human required data processing and waste of information in photos, by using machine learning and photo recognition. Applying machine learning is a possible approach to solve the first problem. Through machine learning, system can learn to guess the needs of users and give more accurate recommendations every time, as a result less human is needed for decision making. For the second problem, OCR, one of the technologies of photo recognition, is useful in this situation. It can extract the characters from photos and convert them into a text form for the system to read. Therefore, the information in the photo can also be utilized.

This project will be delivered in form of a mobile application on iOS platform written in Swift. It will focus on the OCR system for the menus and the image analysis for
foods. To reduce the complexity of machine learning, only deep learning will be discussed.

This paper makes three contributions. First, it reviews the existing online food catalogues on the market. Second, it introduces the system design pattern of the mobile application. It also compares different design patterns in terms of their functionally and testability. Third, it illustrates how the photos of food and menu can be used in deep learning and the values of the result.

The remainder of this paper proceeds as follows. First, it offers a discussion on the choices of technologies. Second, it reviews the limitation of the application and give recommendations. It closes with the plan for future development of the application.
2. Previous Works

Openrice is a famous food and restaurant guide in Hong Kong and some other Asian areas. Openrice’s restaurant search engine allows users to search from its database. Users can also rate and write reviews on restaurants to provide first-hand information for the others. There is photo upload function for users to share the photos of food and menu (see figure 1), as shown in the upload page, users need to input information to the form manually and most of the fields are optional, which results many uploaded photos do not contain description as it is time consuming to input the fields. At the same time, the system does not analyze the information contained in the photos, the food in photos cannot be recognized and classified by the system automatically.

Apart from Openrice, Yelp is also a typical crowd-sourced review online platform for restaurants around the world. With a large user base, Yelp can provide the dataset for developers to do their analysis. Researches about using reviews from Yelp dataset to improve the restaurant has been done, showing the potential of the data of reviews. By downscaling the precision of the review from restaurants to meals, the data collected is able to reveal more information from the customers. Yelp is recently developing an AI to recognize the user’s photo. The technology is planned to use at photo beautification and auto-chopping. With the current APIs such as Google Vision.

Both the advantages and disadvantages of these two online food catalogues can be taken as references for the project. From the experience of Openrice, the system of...
this project is recommended to be able to analyze photos and fill in part of the information automatically. Yelp gives an idea of using AI to help on recognizing photo, which can be used in this project to increase the accuracy of the result from the computer vision APIs.

3. Deliverable

An iOS application *Footo* will be delivered in this project. It is a platform for users to access our database that contains information of restaurants, and provide data by sending them to the server. *Footo* is in the frontend side of the entire system. The UI and functions design are completed at the current stage. The analysis of photos and data in backend server side is yet to be done.

3.1. Function List

This section contains the brief introduction of each function.

3.1.1. Membership System

Users will be able to register as a member of the application before posting new picture or menu into the system. However, all the photo records of dining will be available to users for browsing. Users’ names will be on the list of contributors of each menu. This feature perhaps can create the scene of belonging to the users.

3.1.2. Restaurant Searching

Users can search the restaurants nearby. Recommendations will be shown on the mapscreen of the application (See Figure 1). In the UI, the pins represent the location of the restaurants on the map.
3.1.3. Menu Viewing/Editing

Users can view the digitalized menu in the application. Each dish is presented with the photo taken by other users using this app. In addition, registered user can upload or revise the information on the menu to keep the menu up-to-date.

3.1.4. Food Photo Uploading

Users can take photos for their food and upload it into the system. The system automatically analyses the photo and suggest the corresponding food item from the menu. After uploading the photo, other users can see it in the menu page.

3.1.5. Restaurants/ Dishes recommendation

The system will suggest restaurants that fit the user based on the dining habit of the user.
4. Methodology

4.1. Application Design Pattern

Model-View-Presenter (MVP) will be applied as the design pattern of this application (see Figure 2). The structure of Model-View-Presenter can be considered as the operation of a computer. View is like the input and output devices (i.e. monitor and keyboard), Presenter is similar to the CPU and Model is the hard disk. Keyboard receives the user actions to CPU and CPU updates the image shown on the monitor. CPU also update the data in hard disk if necessary, and the hard disk can change data contained in the register of the CPU. Considering the whole operation, hard disk is not required to know the state of the monitor and keyboard as CPU manages their jobs. Similarly, View and Model are completely separated by the Presenter in MVP.

In the testing phase of software development, Mocking and Unit tests are usually applied for testing the functionality of the application. Mocking means to replace the functions that are unrelated to the part of testing by an empty function, such that the test can focus on the target function. Unit test is the test for a small individual part of source code. In the traditional Cocoa MVC design pattern suggested for iOS applications, it is hard to separate the View and Controller because the View Controllers are closely involved in View’s life cycle, so it is difficult to mock the Views in unit tests. Therefore, the testability of MVC is bad due to the tightly coupled View and Controller. On the other hand, in the MVP pattern, there is a Presenter that
contains the UI business logic for the View, and the View here is generally passive. There is no layout code in the Presenter and it is only responsible for updating the View with data and state. The interaction between Presenter and View is through an interface, hence it is easier mock the View and perform unit tests than in MVC. Thence MVP is more appropriate for this project, as there will be numerous tests needed for handling the data.

4.2. Programming Language
Swift is chosen as the language for developing the frontend iOS application in this project. Comparing to Objective C, Swift have higher performance in terms of the program speed according to Apple's research [1]. Also, Swift has a safe programming pattern due to the strict rules of optional types. This pattern makes clear nil optional values, giving Swift a more stable property in terms of application development and reduce the time for debugging when comparing to Objective-C. While Objective C allows programmers to call a method containing a nil pointer variable, this may lead to the app working improperly in some situations. Therefore, the time needed for bug fixing in Swift would be less when comparing to Objective C.
4.3. System Architecture

The system of this project will be in form of a client-server architecture (see Figure 3). The three-major component are the frontend application, server and database. The users of the application can only see the content on the client side. If they want to access the database, they have to call an API to communicate with the server.

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>System time</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP 5.6.4</td>
<td>102.69s</td>
<td>104.20s</td>
<td>2497508 KB</td>
</tr>
<tr>
<td>HHVM 3.5.0</td>
<td>12.56s</td>
<td>14.83s</td>
<td>362488 KB</td>
</tr>
<tr>
<td>Node.js v0.10.35</td>
<td>2.64s</td>
<td>2.64s</td>
<td>92240 KB</td>
</tr>
</tbody>
</table>

Table 1: CombSort Strict CPU test on 3 different server-side languages

Node.js will be used as server-side language. According to the test performed by Sanchez [2], Node.js has a faster CPU time and more efficient RAM usage while comparing to the other popular server-side languages (see Table 1). The CPU time of Node.js is only 2.64 seconds, which is about five times faster than HHVM. A server with faster CPU time means it can handle more jobs in a period, it is an important property for the server of an open crowd sourcing application because it usually has to
handle multiple upload and download requests at a short period of time.

MongoDB is chosen to be the database at system backend because it is a NoSQL database, which is also known as non-relational database. Although the functionalities of NoSQL database are not as many as the traditional SQL database, NoSQL is more scalable to handle large amount of user inputs and outputs [3]. NoSQL is scaling horizontally, it is easy to add servers into the system for raising the database capacity. On the other hand, SQL is scaling vertically. To deal with increased demand, the server is required to be upgraded, which costs much more than adding another server. Hence MongoDB is preferred for this project as it can be easier to upscale the database for the sake of coping with the increasing number of users in the future.
4.4. Server Hosting

Amazon Web Services (AWS) and Google Cloud Platform (GCP) are two typical cloud platforms being used for server hosting. As the development duration of this project is less than one year, the plans of subscribing over one year are not considered. A high CPU server is more appropriate than a high memory server as its performance is better and there are not much samples that require memory to store. To compare two platforms in terms of their prices (see Figure 4), hosting a high CPU server with on-demand plan in AWS is more affordable than in GCP. It is only about $0.0258 per GB-hour on AWS, but it is $0.04 per GB-hour in GCP. Therefore, AWS is chosen for hosting the server in this project.
4.5. Client-Server Communication

The frontend application is completely separated with the backend server and database. To establish communication between both sides, RESTful API is used as the interface for requests and responses. In the case of the other typical Web service SOAP, it relies exclusively on XML to give communication services [4]. Although SOAP has built-in error handling functions, it requires large amount of codes to perform simple tasks because the XML structure is must be created every time, which is time consuming and rigid. REST is more flexible than SOAP because it only relies on a simple URL. There are diverse types of HTTP request can be used in RESTful API, which are GET, POST, PUT, and DELETE. These requests are easy to be created in the form of URL and do not required to much any complex format. Therefore, it is faster and easier to establish communication by using REST than SOAP.
4.6. Optical Character Recognition

The Google Cloud Vision API will be used for performing OCR. When our system sends the photo of menu taken by user to Google Cloud Vision API, the API will extract the words from the photo and return a list of text contained in the menu.

Table 2: Performance test on 4 different APIs

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Minimum</th>
<th>Maximum</th>
<th>90th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>1.1s</td>
<td>0.302s</td>
<td>3.64s</td>
<td>1.97s</td>
</tr>
<tr>
<td>Google</td>
<td>0.98s</td>
<td>0.4s</td>
<td>1.79s</td>
<td>1.12s</td>
</tr>
<tr>
<td>Clarifai</td>
<td>2.17s</td>
<td>0.81s</td>
<td>7.35s</td>
<td>3.34s</td>
</tr>
<tr>
<td>Microsoft</td>
<td>1.38s</td>
<td>0.81s</td>
<td>4.22s</td>
<td>2.14s</td>
</tr>
</tbody>
</table>

Walls [5] has executed a performance test on 4 different popular APIs that contains OCR service. In the test, 3000 files were sent to each of those APIs and the time taken were recorded. According to the test result (see Table 2), the average performance of Google Cloud Vision API is only 0.98s, which is the best among those 4 APIs.

Although the average time taken of Amazon’s API is 1.1s, which is only 0.12s more than Google’s API, it still affects a lot on the performance as our system is expected to accept a large amount of user uploads. There is a large different in total when every upload takes 0.12s more. The large number of uploads is also the reason of average time being considered as the main concern but not the individual minimum or maximum, because a single maximum or minimum does not give much impact on the entire performance. Therefore, Google Cloud Vision API is chosen in the project.
4.7. Deep Learning

Google Cloud Vision API also provide service on photo recognition, it returns some possible guesses about the items in the photos. This function is used for filling the information of food photos automatically. The system will choose the most related guess and fill it as the name of food. However, the result returned by the API is not guaranteed to be always accurate, then it requires user to correct the wrong guess manually. To reduce the chance of giving amiss guesses, deep learning can be applied.

![Deep Learning for Visual Perception](image)

Figure 6: The accuracy rate of using Traditional CV and Deep Learning

According to Barker [6], there was a GPU system developed for the use of military target recognition. The accuracy rate raised after replacing traditional computer vision by deep learning (see Figure 5), when the system was using traditional computer vision, accuracy of the system varied widely, and the maximum was only about 75%. At 2012, deep learning is applied to this system. After few years of neural network training by the images from Imagenet, the maximum accuracy rate of the system raised to 97%. This is a convincing evidence to prove that deep learning can effectively enhance the accuracy of a visual perception system. Therefore, instead of
only relying on the result returned from the API, deep learning is also applied into our system to increase the accuracy of guessing. Every time a photo is uploaded, the system will record both the guesses from API and the final input submitted by the user. All the records will be added together and form a neutral network. Such that the system can find an answer by searching in that neutral network, and the answer can be more precise because it is based on all the past experience but not only the API.
5. Current Status

5.1. Back-end Development

Currently, the node.js server has been deployed on the Amazon Web Service and the database has been deployed on MongoDB Atlas. There are the popular cloud computational platform providing a stable service with reasonable price. For example, most affordable virtual machine provided by the Amazon EC2 service costs only $0.0058 per hour, which means that it costs less than US$0.2 per day.

5.2. Computer Vision

In this section, different APIs are evaluated based on the needs of this application.

5.2.1 Optical Character Recognition (OCR)

Computer vision technology will heavily involve in this application. The major uses of computer vision in this application are photo recognition and text recognition. Although there exists computer vision library like OpenCV for developing the program for handling image, considering the time and resource needed to construct and train the program from scratch, computer vision API with optical character recognition (OCR) function and object recognition like Google Cloud Vision API and Microsoft Azure Computer Vision API will be considered in this project.

Considering the requirement of accuracy in recognizing text on the menu in order to reduce the further edition time, Google Cloud Vision API was selected as the API for
the OCR function, which coverts photo into computer readable text, based on the result of the following test.

In the test, the ability of recognizing the text on the menu is tested for the Google Cloud Vision API and Microsoft Azure Computer Vision API. A typical image of a menu is used as the input of the OCR (See Figure 5). This test is to simulate the normal situation when a user is trying to take the photo of the menu and upload it into

![Figure 7. A part of the result of the OCR function.](image)

As the detailed method of handling the returning text data is not detailedy decided yet, the method to rate each API is to count the number of correct the name of dish and the price.

In the result, Microsoft Azure Computer Vision API could not recognize most of the small text below the name of the burger and incorrectly recognize most of the text in the “Shakes” and “Drinks” section on the menu. On the other hand, Google Cloud Vision API was able to recognize most of the text on the menu including the small text under
the burger’s name. For example, under “CLASSIC CHEESEBURGER” in the “BURGERS” section, there is a sentence - “5oz beef patty, Wisconsin cheddar, lettuce, tomato, pickle, Circus sauce”. Surprisingly, Google Cloud Vision API returned the result including those text with only some minor mistakes. The misinterpreted words include “oz” as “az” and “5” as “S”. However, the frequency of the misinterpreted words is narrow which users can manually correct them without many operations. Hence, this result shows the capability of the Google Cloud Vision API.

5.2.2 Photo Recognition

Photo recognition is an important part for the photo uploading process in Fooo as the application suggests the name of the dish based on the photo uploaded by the user. Since the suggestion process involving artificial neural network with input nodes, the API must provide a number of tag after processing. In the application, the choice of the API is still Google Cloud Vision API based on the result of a test [7].

The test is to examine the accuracy of image labelling of different APIs, including Microsoft Azure Computer Vision, Google Cloud Vision, Cloud Sight etc. The result of the test showed that Google Cloud Vision API has the capability of recognizing specific category of object which is essential to the system. Besides, the Cloud Sight API has an extraordinary capability of forming accurate caption of the photo. However, it cannot give the system individual tags. Therefore, the Google Cloud Vision API should be the most suitable one among those APIs.
5.3. Machine Learning

Machine learning technique is used in the photo recognition and prediction function. In the current design, tags returned by Google Vision API are used as the features vectors of the algorithms, and the algorithm will fit itself so that it could predict and classify the ID of the food on the menu. This function is for assisting the user to upload the photo for a certain item on the menu.

The selection of machine learning algorithm and the approach to implement the machine learning algorithm affect the accuracy of the prediction. Currently, in the architecture of the machine learning algorithm, the system performs repetitive fittings by using the multilayer perceptron network, which is a feedforward artificial neural network.

The training data is obtained from each restaurant record, where a list of items on the menu are stored with an identifier. At the beginning, since there will be no data in the training data set, the system will not be able to predict any result from the incoming image. However, once there is a data inputted by a user, the system can predict the food based on the existing neuro network.

The idea is retrieved from the popular open crowd sourcing machine learning platform such as the online chat bots learning from the user input. With the growth of the training data set, the accuracy of the prediction will be higher.

It is now decided to implement the machine learning part independently using Python, apart from the original node.js server.
6. Limitations and Recommendations

6.1. Inaccurate Data Entry
This open crowdsourcing application is going to accept data entries from a large population of participants. There might be inaccurate data entries inputted into the system. To mitigate the risk brought by this situation, a voting system can be implemented. As it is resource consuming to validate all obtained information, the system is trust-based, assuming all data entries are correct. If a data entry is flawed, the users can vote down to it. As a result, that entry’s ranking will be lower and there will be less chance for the other users to see it.

6.2. Improper User Behavior
A reporting system can also be implemented to handle a more serious situation. If a user uploads any information that contain improper content, the other users can report this to the system, the system will check if the photo contains any adult content by the vision API. When the system detects improper content from the photo or there is a number of reports towards a data entry, that data entry is being suspended and hid until the verification by system administrator.
6.3. Implementing photo recognition function

There exists a limitation that the computer vision API provided by Google could not identify the object in the photo with high accuracy. Since the categories available in both the APIs is not designed for food only, further analysis is required for the categories data returned by the API. The system should process them with a library of food categories made by our own. This is the reason of involving a further deep learning process in the system. After applying deep learning logic, the system can compare the analysis result of the API with previous results, then provide several accurate suggestions based on the menu in that restaurant.

6.4. Implementing optical character recognition function

In this application, OCR is used as a tool for user to input data of the menu. The main challenge of implementing this function is that the OCR API is not guaranteed to be accurate. It would easily be affected by the lighting and angle when the user takes the picture. Therefore, to reduce the impact, the system cannot be fully relied on the OCR. Users are allowed to edit and adjust the contents after they have taken the photo, as a result the data uploaded to the server can be more accurate.
7. Working Plan

Figure 8: Timeline of 3 phases in the project

This project is separated into three major phases (see Figure 6), the first phase is the design phase for designing the system and researching on the choices of technology. Then it is followed by the implementation phase, both the frontend application and backend system are being implemented in this phase. Testing phase is the final phase of the whole project, the data being collected by the application will be used to analyze the accuracy of the result. After fine tuning the system, it will be presented and displayed in the final year project exhibition.
7.1. Work Accomplished to Date

Currently, the design phase is already completed. The UI of frontend application and system architecture are designed. These designs are being used as the blueprint for implementation in phase 2.

The project is now in the implementation phase. A working prototype is developed for demonstrating the basic function of the application, for example the upload function and the search function. On the other hand, the server and database are also set up with simple request and response function. However, the logic of deep learning in the server is yet to be done.

7.2. Future Planning

In the next few months, the functions of application and server will still in development. However, the main focus will move to the logic of deep learning in the server. Since the faster the deep learning logic is completed, the more time can be used on collecting data and testing in phase 3. The testing phase is planned to be executed at the last 3 months, where testing and fine tuning of the application will be done as the preparation before the final presentation.
8. Conclusion

This paper has introduced the design of the entire system. With the communication between frontend application and backend server, the users are able to upload or search for information about the restaurants and foods. This paper discussed the reason of choosing Node.js, MongoDB and Google Cloud Vision API for implementing the system. This paper also described a case of applying machine learning into the computer visual perception system. The result suggests that deep learning approach can effectively raise the accuracy of guessing. The design on frontend application is completed at this stage, it would not be time consuming on frontend implementation. In the future, the focus could be put on designing and implementing backend server and database, especially for the complicated logic of deep learning.
References


