淡江大學
資訊創新與科技

106 學年度專題實作書面報告

吃貨 - 美食推薦

Munchy Gourmet Recommendation system

學生：　洪詠豐　403840209

陸玟臻　403840233

陳銘澤　403840308

曾博彥　403840506

黃安氏原　403845034

指導教授：　洪復一　博士

中華民國 107 年 1 月

Tamkang University

Department of Innovative Information and Technology

106 Academic Year Monograph Written Report

吃貨 - 美食推薦

Munchy Gourmet Recommendation system

Committee Members of Moral Defense

Chair: _____

_____

_____

_____

Date: January 04, 2018

授權書

茲授權淡江大學資訊創新與科技學系將本人於淡江大學發表之畢業專題，以電子出版品方式發行，例如將著作儲存於光碟，以光碟形式發行，或與電腦網路連結，提供讀者基於個人非營利性質之線上檢索、閱讀、列印等，得不限時間與地域，為學術研究目的之利用。
授權範圍：全部授權。

*立授權書人聲明並保證對上述授權之著作擁有著作權，得為此授權。唯本授權書為非專屬性之授權，立授權書人對上述授權之著作仍擁有著作權。

*上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。

*立授權書人聲明並保證上述於淡江大學發表之畢業專題所做的一切行為負責，並保證遵守中華民國相關著作權法及一切國際著作權法的規定及慣例。

學號：403840209
授權人：洪詠豐　　(簽名)

==========

學號：403840233
授權人：陸玟臻　　(簽名)

==========

學號：403840308
授權人：陳銘澤　　(簽名)

==========

學號：403840506
授權人：曾博彥　　(簽名)

==========

學號：403845034
授權人：黃安氏原　　(簽名)

==========

中華民國 107 年 1 月 4 日

摘要

在現今忙碌的社會中許多民眾為追求更高的效率，開始選擇在外用餐，而在省時與便利的外食文化逐漸形成後，外食族的人口便與日俱增；為了滿足外食族的需求與胃口外食的商機亦日益蓬勃，「去哪裡吃？」以及「吃什麼？」漸漸成了大眾每天必定需要思索的問題。由於以上問題的生成我們因此有了開發一套能子予此問題的系統，此套系統不只可以將搜尋結果顯示於地圖上亦可以在食物種類、季節、距離以及評論中為使用者選擇出更符合個人需求的選擇。

Abstract

Nowadays while the living pace in the society is gradually speeding up, to pursue higher efficiency in life, more and more people decide to eat out to save time.
To fulfill the desires of this up growing population of “fast-food junkies” restaurant industry then strikes rapidly. The question “What to eat?” and “Where to eat” may now appears more in people’s mind and becomes a day to day issue. Therefore we developed a system to not only show locations on map for the restaurant in the area but also recommend various types of restaurants sorting out by types of food, season, distance and rate. With these algorithms in use this system creates an even more personalized options for users to choose upon.

致謝

能夠完成這項專題研究我們想感謝在校園裡的資創系的每一位老師於平日在課堂或是課外的指導使我們對於此專業領域更加孰悉。當然其中，我們想特別感謝我們的指導教授洪復一老師在這半年間的指導與提點。在這半年中，老師每週撥空與我們面談，並與我們一同檢討論文以及簡報上的缺失，於我們的生涯規劃上，亦給予了許多建議，我們由衷的感謝復一老師對我們的幫助。當然專題組員們也是這個專題研究中不可或缺的一份子。雖然大家擁有的能力不盡相同，但是卻可以從中互補，可以足足的感受到團隊合作力量的強大。最後謝謝評審老師可以點出我們這次研究的各種缺失，不經一事不長一智，被點出的缺失會使我們成長，讓未來的我們可以更有能力。

Table of Contents

List of Figures

Chapter 1 Introduction

1.1 Motivation

In today's world, recommendation system has become more and more important. In this era of big data, choosing a correct data among millions of possibility has become a difficult task without using a recommendation system. Finding this topic rather interesting, and while dinning is a highly related topic to all people we decided to combine these two topics and tackle on the recommendation system by creating a restaurant recommendation system.

1.2 Purpose

The goal of this project is to improve current ecommerce system towards restaurants finding. By having a clear structure and efficiently designed algorithms, the system can be more personalized and better suits the preference of the user. A clear guidance will be made for the user which decreases the chances of confusion towards their options. With a better recommending system user can save more of their time and have a rather comfortable user journey.

1.3 Thesis Architecture

This thesis contains five chapters, and it is organized as the following text. Chapter 1 talks about the thesis' motivation and purpose. Followed by Chapter 2, introducing the background of recommendation systems in Ecommerce. In Chapter 3, which is the core of this thesis, the system is briefly introduced via architecture diagram, the designs of the system will then be thoroughly mentioned. While in Chapter 4 reader will be given a brief demonstration on how the system looks like and work, and on the last chapter, Chapter 5 a conclusion toward this thesis will be given.

Chapter 2 Background

2.1 Recommendation System in Ecommerce

Nowadays through the uprising development on technology a huge amount of internet access has been done by people all over the world. With this great amount of access towards information and data, without a doubt has gradually increased and exceed to a number which people may not be able to easily analyze by themselves. Without any help or tools in hand it seems to have become a problem to mankind's when one is finding unambiguous and useful data among all information.

In ecommerce, a paper from Shafer stated that recommendation systems are keys to perform mass customization. Which means the system will be able to suggest the appropriate product or service to suit the right user among the huge amount of information. In other words, each customer would have personalized recommendations and the website would be able to adapt itself to each user. (J. Ben Schafer, 1999)

Personalization increases the interaction between the supplier and the demander. A shopping experience system provided like so can establish a long-term relationship with customers.

Below are two main categories of recommendation systems which are: content-based system and collaborative filtering system. In the first category, the recommendation is based on the products and their properties, while the second consider the similarities between end-users.

2.1.1 Content-based System

The content-based system is a model, where the recommendations are made on the basis of the several properties of a product. In other words, the system compares the similarities or the complementarity of the elements.

For example, a search system can use this category of recommendation. Given the previous purchases, the algorithm would search in the catalog related items and suggest them to the user. If a customer has bought a great number of

western movies or has bought many books from the same author, the system would recommend other popular western movies or books from this author. (Greg Linden, 2003)

This system is also often used for directly scan the shopping cart of a customer instead of his purchase history and match the items with complementary ones. For example, the figure below shows, with a shopping cart containing a Bluetooth device, the system recommends Bluetooth dongles. (Greg Linden, 2003)

A content-based system can also use the similarity between elements to provide a recommendation. In this case, a user isn't asked to put attributes, but only a movie he liked and the system compares this item with similar ones in the base. This technique can also be performed by analyzing the purchase history of a user: it will try to match the movies the user watch and find similar ones among his store. (Lubos Democvic, 2013)

There is a main limitation to the content-based model: the scalability. Since it can work very well when the purchase history is rather small (or with the shopping basket where the collection is limited), the quality of the recommendation can drop very fast when dealing with a larger set of elements. If the user buys a lot of items, the system wouldn't be able to return pertinent recommendations. It could either be too general, like the most popular movies of a category, or too narrow like the books of the same author. (Greg Linden, 2003)

With only an "is similar to" relationship between products, the recommendation can have a rather poor. It would be more interesting to know how a product is similar to another one and the level of the similarity.

2.1.2 Collaborative Filtering System

In this category, the system builds profiles of users based on their behaviors and compares the different user profiles to provide a recommendation. For example, the system find correlation between users' purchases and recommend a product to a customer based on the items bought by similar users.
A common practice to improve this technique is to ask the user to rate the products, so it becomes much easier to find common pattern among users. By this way, the model is transformed into a segmentation model. The customer's

preferences are analyzed and the recommendations are made based on the preferences of other users of the same segment. So, the users collaborate to build their “global” profile and profit from others’ ratings. (Prassas, 2001)

To consolidate this model, not only explicit ratings can be considered, but also implicit ratings such as navigating behavior like time spent on a page or bookmarks. (Prassas, 2001)

However, defining segments of users could lead to poor quality recommendations, since all the users of a same segment are considered to have exactly the same preferences. And doing some tweaking on the segments, like using granularity, would become expensive to just determine those segments. (Greg Linden, 2003)

Also, collaborative filtering system requires time: Since it is based on past behaviors, the user has to build slowly his profile and provide data that can be analyzed. New user preferences are difficult to determine and the strategy would be rather asking to provide inputs by themselves. The participation of the user is crucial to determine

Chapter 3 System Analysis and Design

3.1 Overview

Figure 1 System Architecture

With the collection of various data from different website, data are then collected and generated through the server. After processing all information, it then integrate with the user interface to connect with the client and provide a personalized recommendations

3.2 Data Collection

3.2.1 Purpose

By using tools and techniques below, the system is able to collect large sets of data in order to recommend more options toward users.

3.2.2 Beautiful Soup

Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages. It helps pull particular content from a webpage, remove the HTML markup, and save the information. It is a tool for web scraping that helps clean up and parse the documents that should be pulled down from the web.

The Beautiful Soup documentation gives developer a sense of variety of things that the Beautiful Soup library will help with, from isolating titles and links, to extracting all of the text from the html tags, to altering the HTML within the document which they are working with. (Wieringa, 2012)

3.2.3 MySQL

MySQL is a fast, easy-to-use RDBMS being used in several small and big businesses. It is gradually becoming popular because of below reasons

- MySQL is released under an open-source license.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
  (MySQL - Introduction, 2017)

3.2.4 Python

Python is a dynamic, interpreted language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and make developer lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that occurs error as it runs. (Python Introduction, 2016)

3.2.5 Data Collection Process

Using the request (parsing URL) package to receive website's reply, Beautiful Soup can then parse web's reply into objects for itself. After parsing data can be find using library.

3.2.5.1 Data Collecting Steps

Below are the steps for how the system collects data:

1. Get sequence, restaurant name, restaurant id and restaurant status (whether it's moved, close or normal) then store them in the database
2. Use restaurant id to fetch furthermore detailed data.

Figure 2 Fetching Detailed Data

3. Next, use restaurant id to catch comment and reply, comment id, comment user, reply establish time.

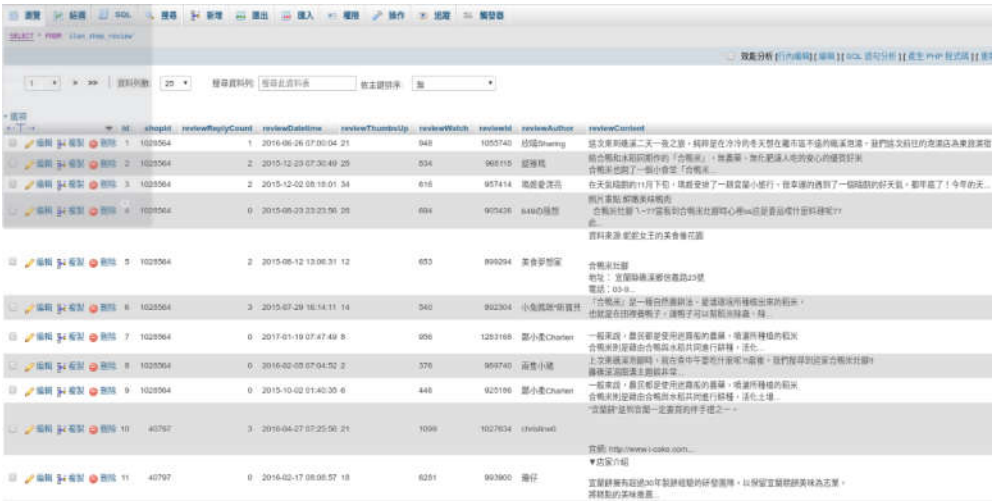

Figure 3 Fetching Detailed Data 2

4. Last comment id to catch responding user, comment and responding time.

3.2.5.2 Data Collection Rules

- Duplication: If duplication happens and function cannot be found, then the system splits or finds build in function in python to compare id and eliminate

```
for i in result:
    if i not in no_duplicate:
        no_duplicate.append(i)
return no_duplicate
```

Figure 4 Duplication Elimination

- Time Transferring: the system will transfer the data into usable form in order to make the DBS receive successfully

```
def extract_date(string):
    dtime = datetime.datetime.strptime(str(string), "%Y-%m-%dT%H:%M:%SZ").strftime('%Y-%m-%d %H:%M:%S')
    return dtime
```

Figure 5 Time Transferring

- Updating: If the restaurant id was changed, the system might keep oriented towards the unreadable page, thus when such scenario happens, the system will get the new shop id and replace the old one.

- Page Validation: If the id that's search on does not exist and keep oriented towards same page with not usable data, the system will then stop fetching the id's relating data

```
def page_is_validated(parsed):
    # count page item
    item_count = parsed.find('h2', class_="num").find('b').text
    if int(item_count) is 0:
        return False
    elif int(item_count) is not 0:
        return True
    else:
        print(">>Unknown error check validator")
        return False
```

Figure 6 Page Validation

3.2.5.3 Data Collection Result

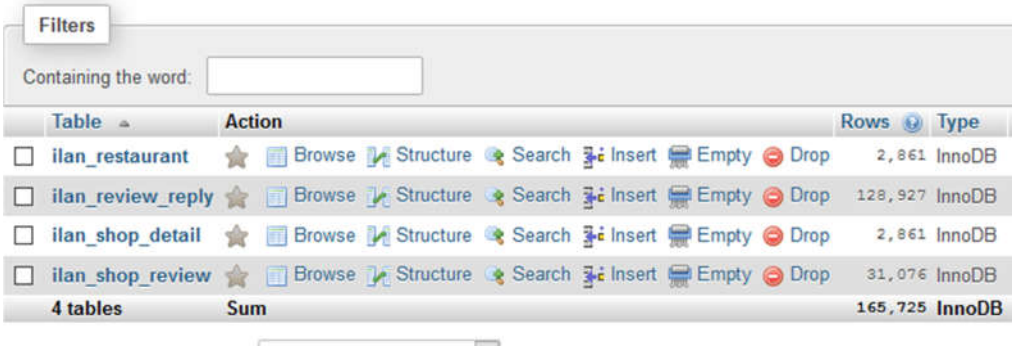

Figure 7 Data Collection Result

3.3 Data Processing

3.3.1 Graph Theory

The first theorem of graph theory was first presented by Leonhard Euler to find a solution known as the Seven Bridges. Since then, many other graph theorems appeared and graphs have proved that they can model a lot of different-looking. They have also been used to understand problems in many fields such as biology, transportation or social networks. (Hürlimann, 2014)

This has led to the development of graph algorithms, a class of algorithms to compute operations on graphs.

3.3.2 Graph Algorithm

There are plenty of existing graph algorithms. Some widely used algorithms that the system have included will be explained below

3.3.2.1 Depth First Search

It is an algorithm for searching a graph or tree data structure. The algorithm starts at the root (top) node of a tree and goes as far as it can down a given branch (path), and then backtracks until it finds an unexplored path, and then explores it. The algorithm does this until the entire graph has been explored. Many problems in computer science can be thought of in terms of graphs. For example, analyzing networks, mapping routes, scheduling, and finding spanning trees are graph problems. To analyze these problems, graph search algorithms like depth-first search are useful. (Depth-First Search (DFS), 2017)

3.3.2.2 Breadth First Search

It is an algorithm for traversing or searching tree or graph data structures which starts at the tree root and explores the neighbor nodes first, before moving to the next level neighbors.

3.3.2.3 Shortest Path System

The algorithm creates a tree of shortest paths from the starting vertex, the source, to all other points in the graph. The graph can either be directed or undirected. One stipulation to using the algorithm is that the graph needs to have a nonnegative weight on every edge. Dijkstra's algorithm finds a shortest path tree from a single source node, by building a set of nodes that have minimum distance from the source. (Dijkstra's Shortest Path Algorithm, 2018)

3.3.3 Graph Database and Neo4j

In the last chapter, the benefits of using correlation search between data were shown. To compute those operations, it is important to find and use the right tools and technology. Since, graph models and graph algorithms are used in this paper, this chapter will explain how to use graph databases to store our data and how to query it by algorithms computations.

3.3.3.1 Graph Database

Graph database is a NoSQL data model which focuses on entities and the relation between them. There are several available graph databases, but, in this paper Neo4j will be presented and used.

Neo4j is an open-source graph database written by Java programming language and is said to be fully ACID, in contrary of most of the database models.

3.3.3.2 Query Performance

The architecture of Neo4j is built in such a way to be scalable with Big Data. Hongchang Huang and Ziyu Dong tested the query performance of Neo4j and came with some interesting results. First, the architecture is defined in such a

way to be traversed very rapidly. The data is not only physically stored in a way to be read rapidly by allowing fixed size spaces for tuples, but also by using multiple cache layers. (Hongcheng Huang, 2013)

Their result are quite interesting: as the complexity rises, the stability of a query remains rather at the same level. Thus, it isn't fool to think that graph databases could bring some solutions to today's internet overload of information.

3.3.4 Preprocess Data

- Export csv file from mySQL database
- Process csv file with python and panda
  - Making timestamp into proper format of date, month, year

```
In [6]: #extract year, month, day
        from datetime import datetime
        for i in range(len(rdf)):
            date=datetime.strptime(rdf['reviewDatetime'][i],'%Y-%m-%d %H:%M:%S')
            rdf.loc[i,'year']=date.year
            rdf.loc[i,'month']=date.month
            rdf.loc[i,'day']=date.day

In [61]: rdf.head(5)
```

Out[61]:

p	reviewWatch	reviewId	reviewAuthor	reviewTimestamp	year	month	day	SDRate	SDDelicious	SDServiceRate	SDE
	2131	279023	amy的秘密花園	1.33366e+09	2012	4	6	1	2	1	1
	353	1015828	卡夫卡愛旅行	1.45974e+09	2016	4	4	3	4	5	1
	1078	1027634	christine0	1.46171e+09	2016	4	27	4	4	3	4
	6177	993900	珊仔	1.45567e+09	2016	2	17	5	4	5	5
	6115	987196	糖糖♥	1.45421e+09	2016	1	31	3	5	1	4

Figure 8 Preprocess Data wth Python Panda

  - Extract the division from address for each restaurant
  - List each category of each restaurant into a two column table
- Import csv file into Neo4j with Neo4j load csv function

3.3.5 Further data modeling and analyze in Neo4j

3.3.5.1 Data Modeling

Like most of NoSQL database systems, there is no specific rules to design a database. From the developer point of view, the flexibility is one of the main advantages, of those advanced database systems. There is no need to normalize

the database like in a rational environment. However, the flexibility does not mean that it is right to put whatever wherever. The model should be depends on both the data and the queries [max-flight]. As the crawling data are restaurant reviews, the initial data modeling is proposed as below. Further graph manipulation is explained in later sub chapter

3.3.5.2 Data Model

- **Nodes:**

The nodes in the graph will represent entities:

- Restaurant
- Review
- Author
- Division
- Category

- **Relationships:**

There will be 4 initial relationships:

- ABOUT(from Review to Restaurant)
- WROTE(from Author to Review)
- IN_DIVISION (from Restaurant to Division)
- IN_CATEGORY(from Restaurant to Category)

- **Properties:**


```
{
  "SDCategory": "其他速食"
}
```


Figure 9 Properties of Category


```
{
  "authorId": "amy的秘密花園",
  "nodeId": 2558
}
```


Figure 10 Properties of Reviews' Author

```
{
  "SDEnvRate": 1,
  "month": 4,
  "year": 2013,
  "SDRate": 2,
  "SDServiceRate": 2,
  "SDDeliciousRate": 2,
  "day": 4,
  "reviewId": 443922,
  "reviewTimestamp": 1365054592
}
```


Figure 11 Properties of Reviews


```
{
  "SDAddress": "宜蘭縣礁溪鄉礁溪路六段220
號",
  "SDEnvRate": 13,
  "price": 21,
  "SDServiceRate": 13,
  "SDRate": 40,
  "latitude": 24.83440001763,
  "shopName": "老如意手工水餃",
  "shopId": 125531,
  "SDDeliciousRate": 13,
  "nodeId": 4182,
  "longitude": 121.78123701771
}
```


Figure 12 Properties of Restaurant

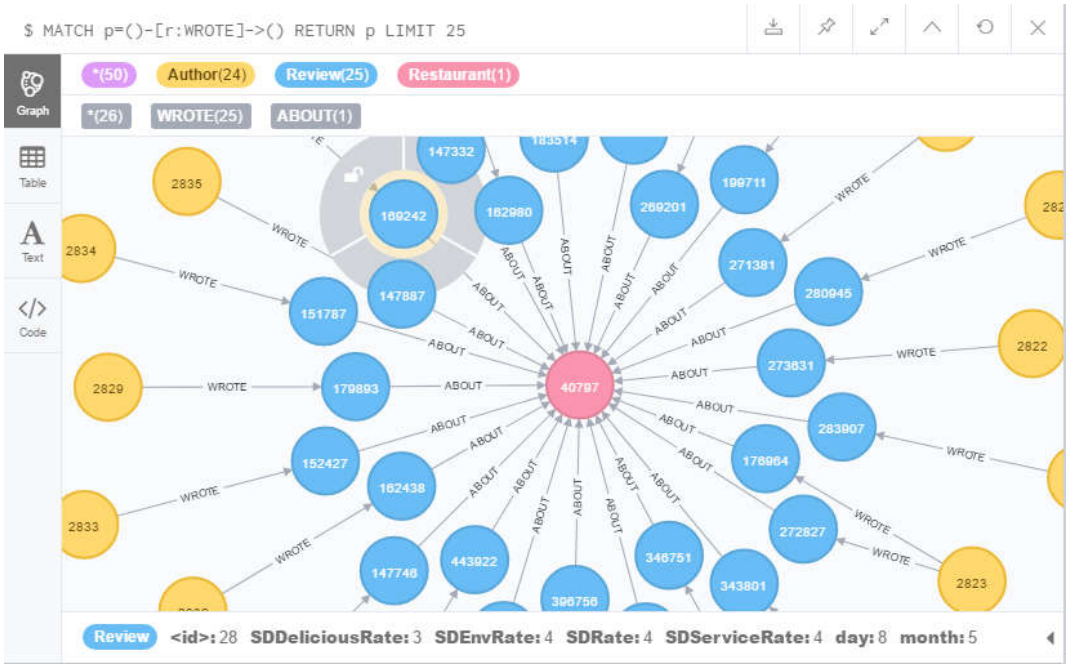

Figure 13 Initial Database View

3.3.5.3 Use case

3.3.5.3.1 Restaurant near you

```
WITH point({latitude:24.8229533748,longitude:121.771853579}) AS mapcenter

MATCH (reco:Restaurant)-[:IN_DIVISION]->(b:Division{e_name:"Jiaoxi Township"})

WITH reco, distance (point({latitude: reco.latitude, longitude: reco.longitude}), mapcenter) AS distance
RETURN reco, distance  ORDER BY distance LIMIT 25
```

Figure 14 Restaurant Near you Query

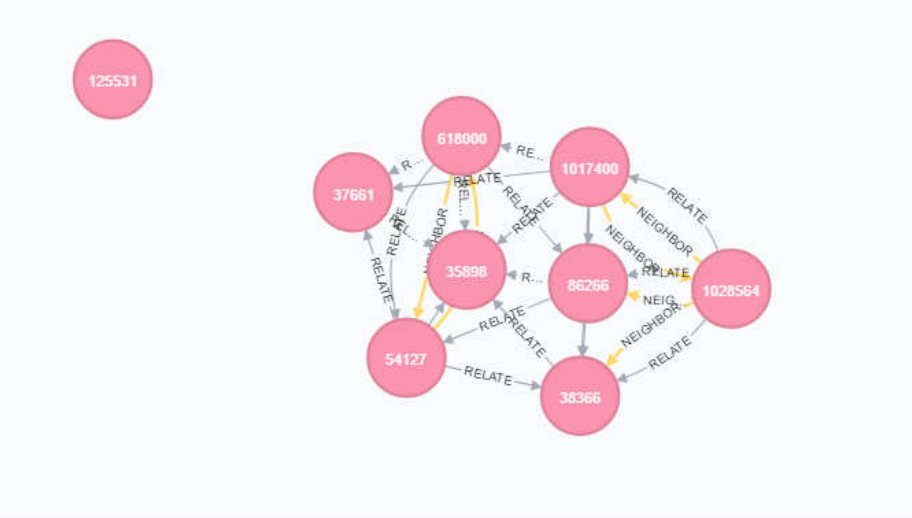

Figure 15 Near you Result

3.3.5.3.2 Top Restaurants for each Month and Season

- Here we need to provide to create the Month and Season nodes for quicker query
- For each division, create twelve month node, connect the reviews of a restaurant with the corresponding month in the division

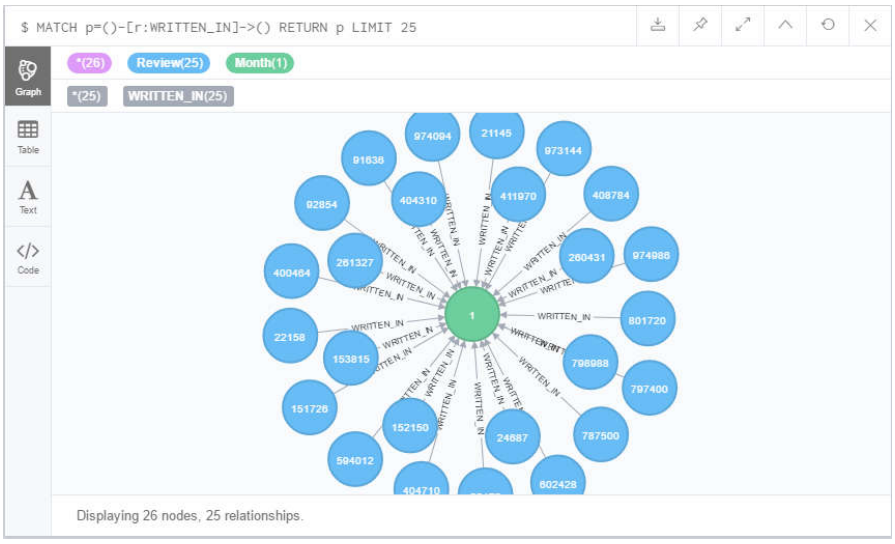

Figure 16 Creating months result

```
match (res:Restaurant)-[:IN_DIVISION]-(area:Division), (area)-
[:IN_MONTH]->(m:Month),(res)<-[r:ABOUT]-(re:Review)-[:WRITTEN_IN]->(m) where
area.e_name="Jiaoxi Township" and m.month=4

with res,m, count(r) as rs

with  res,m order by rs desc limit 3

merge (m)-[:MONTH_TOP]->(res)
```

Figure 17 Creating MONTH_TOP relationship

- For each division, choose the restaurants with the number highest number of reviews limit to 3 and connect to the corresponding month for better inquiry

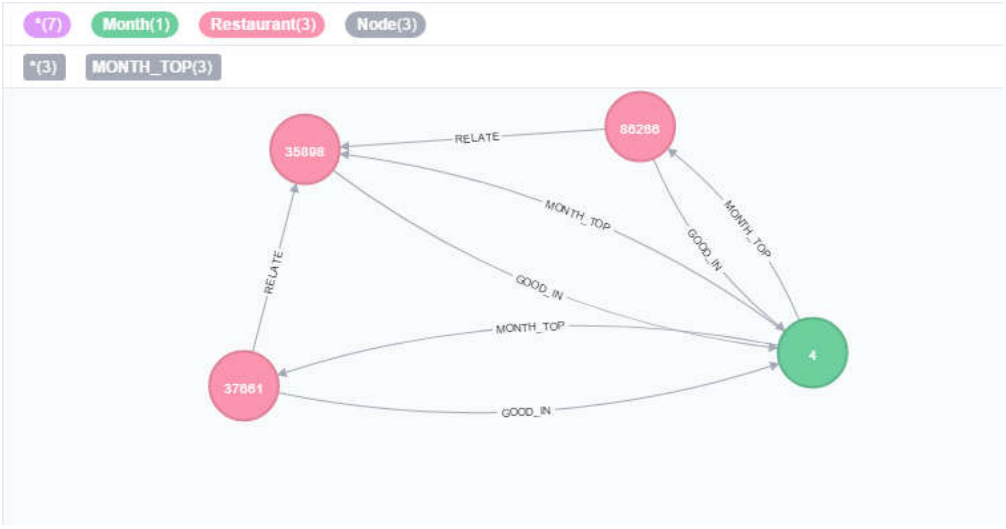

Figure 18 MONTH_TOP query result

3.3.5.3.3 Top Places in each Division

```
match (res:Restaurant)-[:IN_DIVISION]->(area:Division), (res:Restaurant)<-[:ABOUT]-(r:Review)
where 2013<=r.year and 2017>=r.year  and area.e_name="Jiaoxi Township"
 with area,res, count(r) as reviewCount
with  area,res ,reviewCount order by reviewCount desc limit 5
match (res)-[:IN_DIVISION]->(area),(res)<-[:ABOUT]-(r:Review)
where 2013<=r.year and 2017>=r.year
with area,res,avg(r.SDRate) as avg
merge (res)<-[:TOP_PLACE{top_place:avg}]-(area)
```

Figure 19 Top Places creating query

- Collect the review of restaurant in recent years
- Limit them to top 5
- In top 5, calculate the avg of review rating during recent years
- Create a [:TOP_PLACE] relationship from res to the division for later quick query
- Example of Recommendation:

```
match (reco:Restaurant)<-[rel:TOP_PLACE]-(d:Division) where d.e_name="Jiaoxi Township"
with reco,rel.top_place as top_place
return reco order by top_place desc
```

Figure 20 Top places recommendation

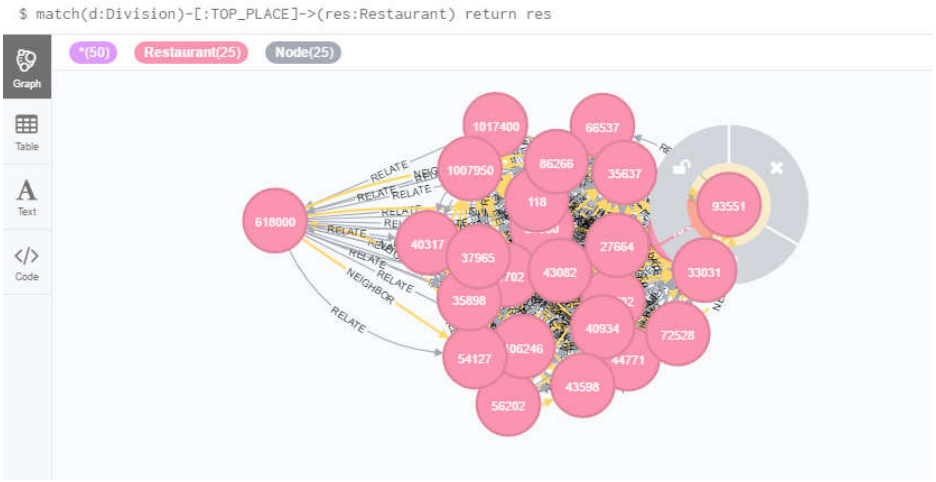

Figure 21 Result of top places query

3.3.5.3.4 Restaurant Tendency Toward A Certain Month

For one restaurant, its popularity depends on a lots on seasons, for example in winter lots of would go for hot pot, or in summer lots of people will choose for ice cream (pics)

- Base on this factor, the restaurant tendency toward a certain month is created as a relationship between one restaurant and month and this tendency is leveled by the average rating of reviews for that restaurant in the month

```
Match (res:Restaurant)<-[:ABOUT]-(r:Review),(r)-[rel:WRITTEN_IN]->(m:Month)
with res,m, count(rel) as mm, avg(r.SDRate) as m_avg
with res,m,mm,m_avg
merge(res)-[:BY_MONTH{m_avg:m_avg,no_reviews:mm}]->(m)
```

Figure 22. Create BY_MONTH relationship

3.3.5.3.5 Create RATED RELATIONSHIP to easier computer similarity between restaurant

```
//create latest rate relationship
match (n:Restaurant)-[]-(r:Review)-[]-(au:Author)
with n,au, collect(r) as rs
unwind rs as r
with n,au,max(r.reviewTimestamp) as max
match (n)-[]-(new_r:Review)-[]-(au) where new_r.reviewTimestamp=max
merge (au)-[:RATED{rating:new_r.SDRate}]->(n)
```

Figure 23. Create RATED relationship query

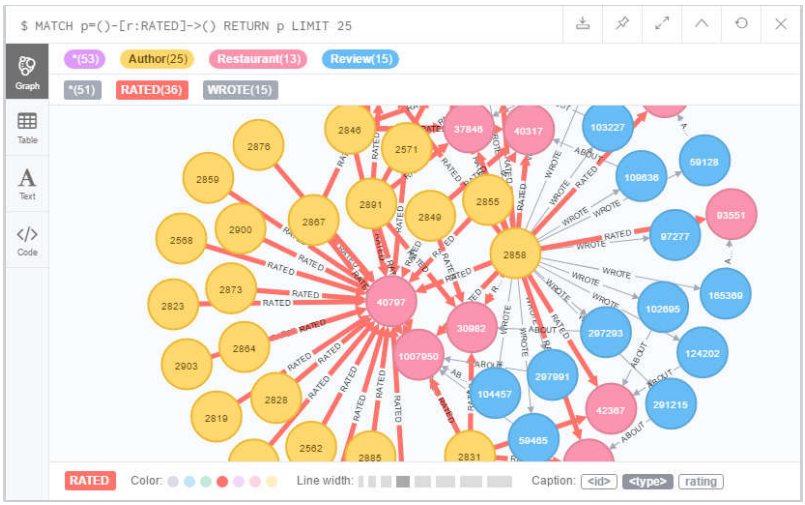

Figure 24 Create RATED relationship result

3.3.5.3.6 Relating Restaurant:

- Relating Restaurant is defined as the restaurant which usually are together. User is a between, an user who go restaurant I also goes to restaurant II
- The restaurants are connected through users, the connection [:RELATE] relationship is weighted with number of common users between them.

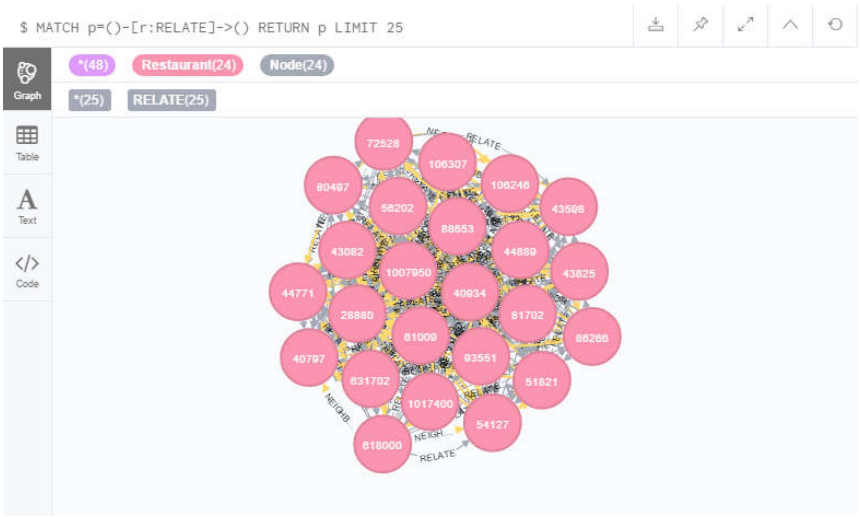

Figure 25 Create RELATE relationship result

- Gather the top 10 relating restaurants of each restaurants and create the NEIGHBOR relationship for the top 10 relating restaurants

```
match (n1:Restaurant) where n1.shopId=149
with (n1)
match (n1)-[rel:RELATE]->(n2:Restaurant)
where not ((n2)-[:NEIGHBOR]->(n1))
with n1,n2, rel order by rel.common_au desc limit 10
merge (n1)-[:NEIGHBOR{common_au:rel.common_au}]->(n2)
```

Figure 26 create NEIGHBOR relationship query

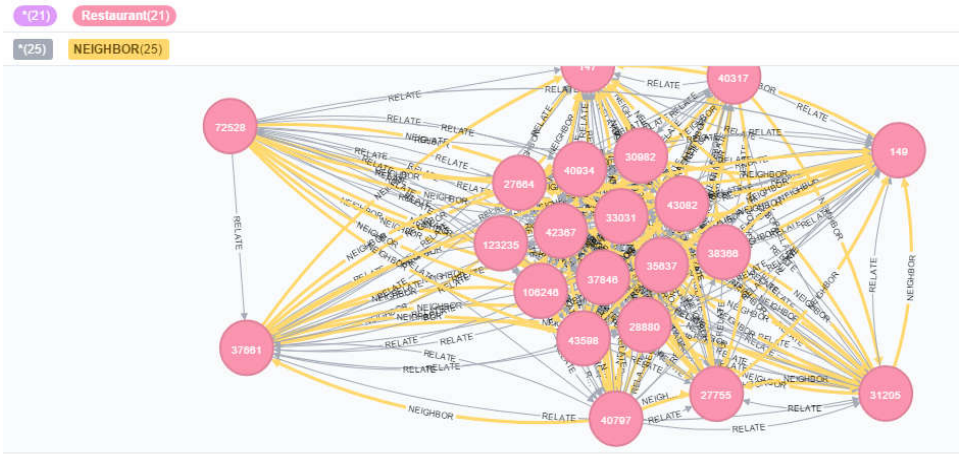

Figure 27 Result of create NEIGHBOR relationship

- Since the property common_au of [:NEIGHBOR]and cost for shortest weighted path are inversely related. The cost of each relationship is set as
  - Neighbor.cost= max([:NEIGHBOR.common_au])+1-Neighbor.common_au
- From the user's interest, relating restaurant is suggested by starting from user's interesting, with the shortest weighted path limit 20 result

```
match (n:Restaurant{shopId:149}),
path=(n)-[r:NEIGHBOR*1..3]-(a)
with reduce(cost=0, r IN rels(path) | cost+r.cost) AS totalWeight,path,(n),(a)
return distinct a.shopId,totalWeight order by totalWeight limit 20
```

Figure 28. Shortest weight path query

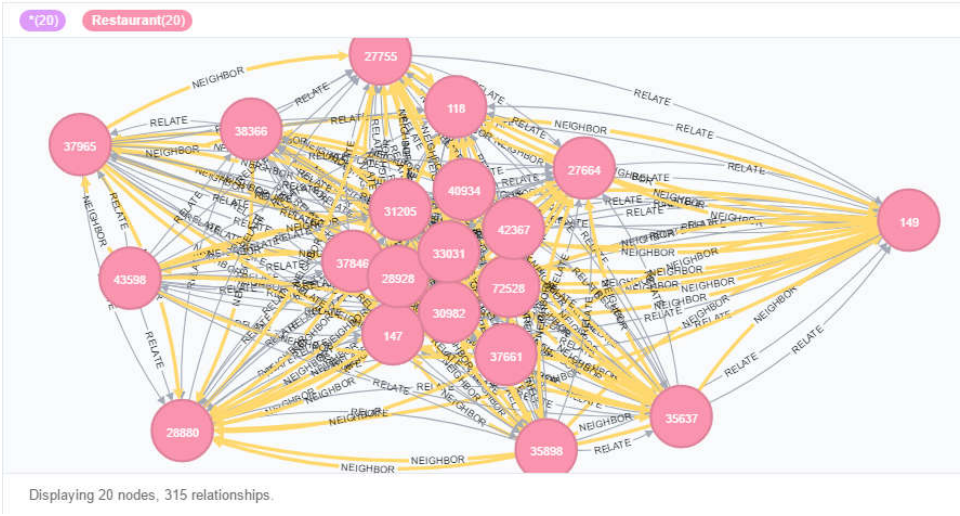

Figure 29. Shortest weighted path result

Chapter 4 Implementation and Discussion

4.1 User Interface Implementation

This User Interface/ website is implemented with Flask framework, along with HTML, CSS, and JavaScript.

4.1.1 HTML

HTML (Hypertext Markup Language) is the standard markup language for creating Web pages. The HYML elements are represented by tags written using angle brackets and they are also the building blocks of HTML pages. HTML tags label pieces of content such as “heading”, “paragraph”, and “table”. Instead of displaying the HTML tags, browsers uses them to render the content of the page. (HTML Introduction, 1999-2017)

4.1.2 CSS

CSS stands for Cascading Style Sheets, is a language that describes the style of an HTML document and how HTML elements should be displayed on screen, paper, or in other media. It saves a lot of work, and is able to control the layout of multiple web pages all at once. (CSS Introduction, 1999-2017)

4.1.3 JavaScript

JavaScript is the programming language of HTML and the Web.

4.1.4 Flask

Flask is a BSD licensed micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. (Flask 0.12.2, 1990-2018)

Example applications which uses Flask framework includes Pinterest, LinkedIn, and the community web page for Flask itself.

Flask can also be called as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program. (Ronacher, 2013)

4.2 Demonstration

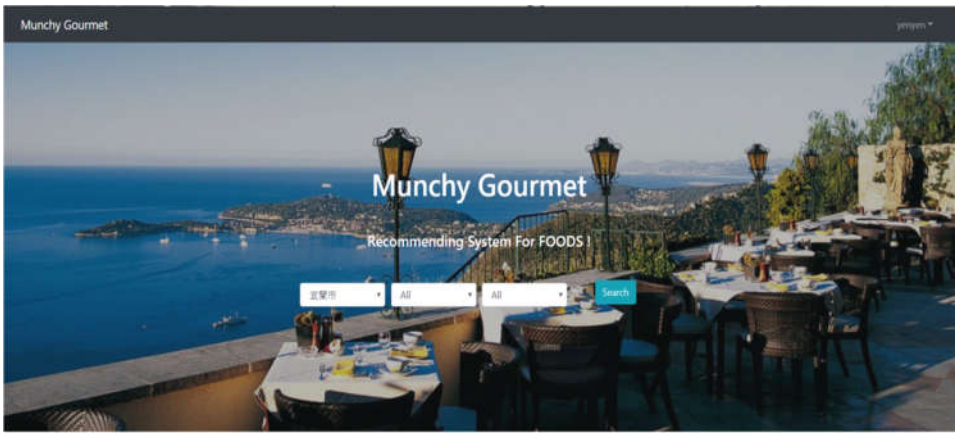

Figure 30 User Interface

Chapter 5 Conclusion

The purpose of developing this system is to solve the hesitant which people nowadays often meet when making decisions among a huge variety of food choices. Through the data collection and processing mechanisms demonstrated in above thesis, we have created a system which suits customers better than just randomly searching through mass data. We believe that our recommendation system can better generate all restaurants in the area and provide users a higher efficiently designed personalized options in order for our users to solve their daily worries

In the future, we are interested in working on machine learning which may predict users' taste and lead the system to give an even more accurate user preference guidance. .

References

1. *CSS Introduction*. (1999-2017). Retrieved from w3schools.com: https://www.w3schools.com/css/css_intro.asp
2. *Depth-First Search (DFS).* (2017). Retrieved from BRILLIANT: https://brilliant.org/wiki/depth-first-search-dfs/
3. *Dijkstra's Shortest Path Algorithm.* (2018). Retrieved from BRILLIANT: https://brilliant.org/wiki/dijkstras-short-path-finder/#dijkstras-algorithm
4. *Flask 0.12.2*. (1990-2018). Retrieved from Python Software Foundation: https://pypi.python.org/pypi/Flask
5. Greg Linden, B. S. (2003). *Amazon.com Recommendations.*
6. Hongcheng Huang, Z. D. (2013). *Research on architecture and query performance based on distributed graph database Neo4j.*
7. *HTML Introduction*. (1999-2017). Retrieved from w3schools.com: https://www.w3schools.com/html/html_intro.asp
8. Hürlimann, T. (2014). *Transportation Logistics.* University of Fribourg.
9. J. Ben Schafer, J. K. (1999). *Recommender Systems in E-Commerce.* Minneapolis: University of Minnesota.
10. Lubos Demovic, E. F. (2013). *Movie Recommendation Basedon Graph Traversal Algorithms.*
11. *MySQL - Introduction*. (2017). Retrieved from tutorialspoint: https://www.tutorialspoint.com/mysql/mysql-introduction.htm
12. Prassas, G. (2001). *A Recommender System for Online Shopping Based on Past Customer Behaviour.*
13. *Python Introduction*. (2016, Aug 5). Retrieved from Google Developer: https://developers.google.com/edu/python/introduction
14. Ronacher, A. (2013). *Flask Extension*. Retrieved from Flask: http://flask.pocoo.org/extensions/
15. Wieringa, J. (2012, Dem 30). *Intro to Beautiful Soup.* Retrieved from The Programming Historian: https://programminghistorian.org/lessons/intro-to-beautiful-soup