COEN 281 Data mining - 2018 Spring

I read, I see, I order

Chen Zheng

Yue Wang

Shaobo Wang

Jing Xu

Anusha Suresh

Contents

2. Introduction

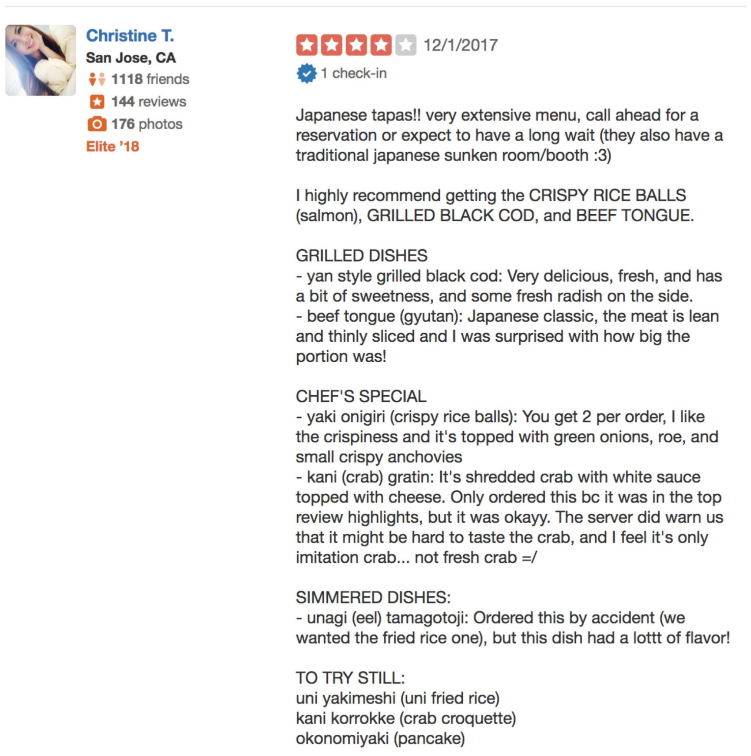
2.1 Objective

As recent years, YELP has become the dominated reviewing website to help the customers to choose the right restaurant. Customers like taking other people's opinions if they decide to go eat at this restaurant or not. Helping the user efficiently to identify the photo is critical part to improve the user's experience.

We will build the application to recognize the food name in the sentence and return back the food image corresponding to the same name with food ingredients, cooking method, and nutrition.

2.2 What is the problem?

When users are reading the comment on yelp, a lot of times they are curious



Christine T.
San Jose, CA
1118 friends
144 reviews
176 photos
Elite '18

12/1/2017
1 check-in

Japanese tapas!! very extensive menu, call ahead for a reservation or expect to have a long wait (they also have a traditional japanese sunken room/booth :3)

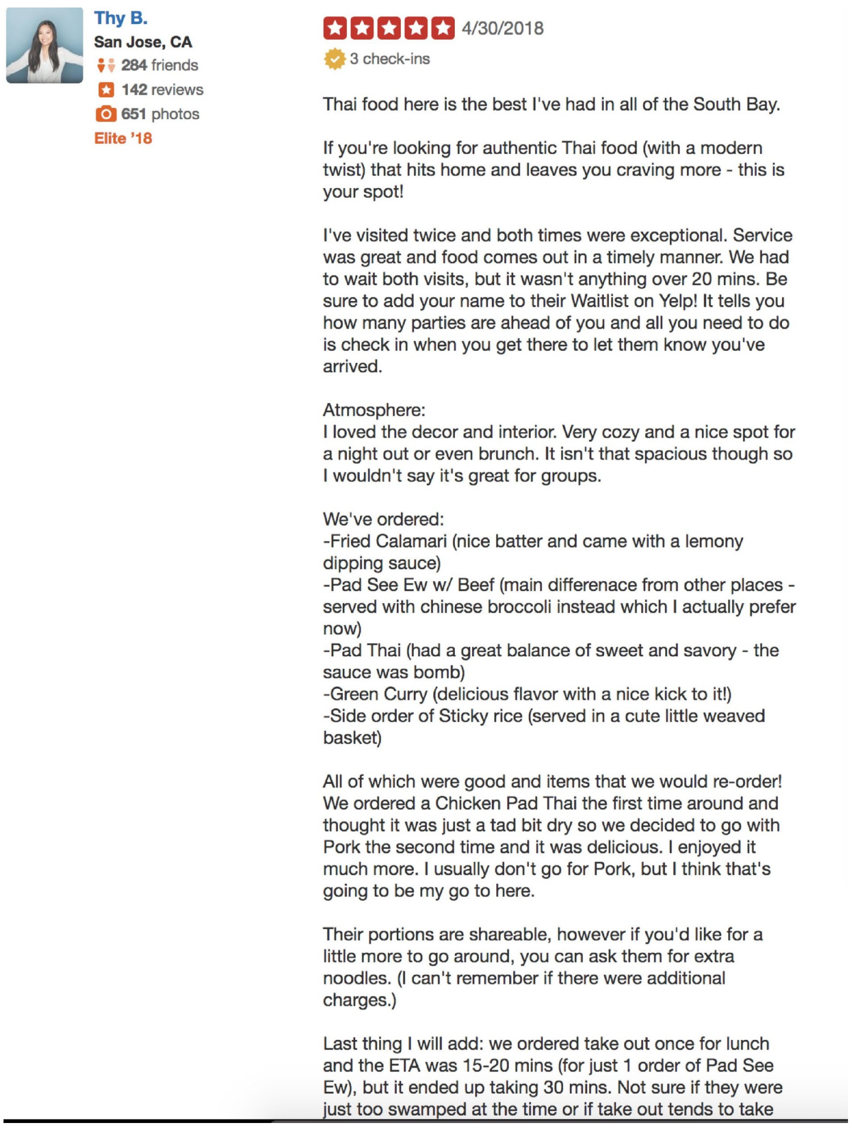
I highly recommend getting the CRISPY RICE BALLS (salmon), GRILLED BLACK COD, and BEEF TONGUE.

GRILLED DISHES
- yan style grilled black cod: Very delicious, fresh, and has a bit of sweetness, and some fresh radish on the side.
- beef tongue (gyutan): Japanese classic, the meat is lean and thinly sliced and I was surprised with how big the portion was!

CHEF'S SPECIAL
- yaki onigiri (crispy rice balls): You get 2 per order, I like the crispiness and it's topped with green onions, roe, and small crispy anchovies
- kani (crab) gratin: It's shredded crab with white sauce topped with cheese. Only ordered this bc it was in the top review highlights, but it was okayy. The server did warn us that it might be hard to taste the crab, and I feel it's only imitation crab... not fresh crab =/

SIMMERED DISHES:
- unagi (eel) tamagotoji: Ordered this by accident (we wanted the fried rice one), but this dish had a lottt of flavor!

TO TRY STILL:
uni yakimeshi (uni fried rice)
kani korrokke (crab croquette)
okonomiyaki (pancake)



Thy B.
San Jose, CA
284 friends
142 reviews
651 photos
Elite '18

4/30/2018
3 check-ins

Thai food here is the best I've had in all of the South Bay.

If you're looking for authentic Thai food (with a modern twist) that hits home and leaves you craving more - this is your spot!

I've visited twice and both times were exceptional. Service was great and food comes out in a timely manner. We had to wait both visits, but it wasn't anything over 20 mins. Be sure to add your name to their Waitlist on Yelp! It tells you how many parties are ahead of you and all you need to do is check in when you get there to let them know you've arrived.

Atmosphere:
I loved the decor and interior. Very cozy and a nice spot for a night out or even brunch. It isn't that spacious though so I wouldn't say it's great for groups.

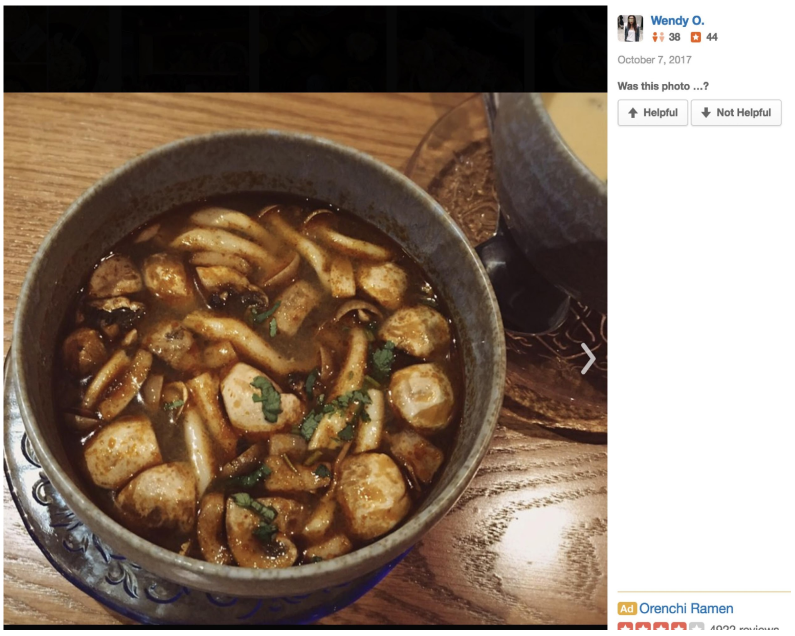
We've ordered:
-Fried Calamari (nice batter and came with a lemony dipping sauce)
-Pad See Ew w/ Beef (main differenace from other places - served with chinese broccoli instead which I actually prefer now)
-Pad Thai (had a great balance of sweet and savory - the sauce was bomb)
-Green Curry (delicious flavor with a nice kick to it!)
-Side order of Sticky rice (served in a cute little weaved basket)

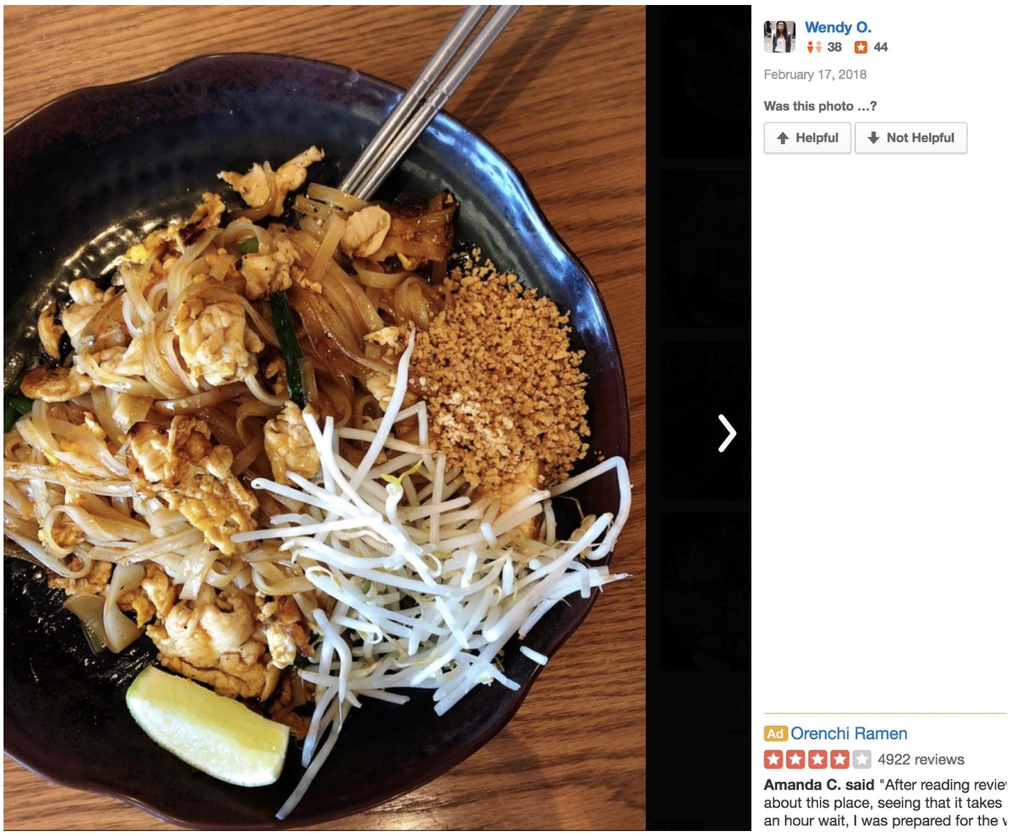
All of which were good and items that we would re-order! We ordered a Chicken Pad Thai the first time around and thought it was just a tad bit dry so we decided to go with Pork the second time and it was delicious. I enjoyed it much more. I usually don't go for Pork, but I think that's going to be my go to here.

Their portions are shareable, however if you'd like for a little more to go around, you can ask them for extra noodles. (I can't remember if there were additional charges.)

Last thing I will add: we ordered take out once for lunch and the ETA was 15-20 mins (for just 1 order of Pad See Ew), but it ended up taking 30 mins. Not sure if they were just too swamped at the time or if take out tends to take

what type of the food looks like in this restaurant. I clipped two comments which reviews the local Japanese food restaurant and makes some recommendations from YELP. As we can see, a lot of recommended food names are Japanese, ultimately this would take long time to do your own research to find out the what the food looks like. Since they are directly translated from Japanese, you have no clues what are them or maybe food have different names and you cannot remember each of them. Nowadays, more and more people care about the fitness, they also would like to know the specific nutritional parts of the selected food. We can provide the ingredient parts and create the nutrition label for this chosen food. Such as protein, carbs, sugar, fat and so on. We understand that the most appealing part is picture which you can actually see corresponding to the menu. However, we need carefully to think of how to choose the most qualified photo for the user, instead of dragging the random picture from the internet, user actually prefer to see the real one directly from this restaurant that they are interested in. In another scenario, when foreigners come to visit another country, it is very difficult for them to order right food because of lacking the local culture and language barrier. If there is a tool which could accurately help users to identify the food immediately, when users are reading the comments online, this would definitely improve the user experience a lot. Users can simply click the food name in the sentence and there will be small window indicating the related food, ingredients, cooking direction and calories popped out. Finding the correct picture sounds straightforward. If there is existed corresponding name tag for this kind of food on the yelp, we only need to do one to one function matching.





Wendy O.
38
44
February 17, 2018
Was this photo ...?
Helpful
Not Helpful
Ad Orenchi Ramen
4922 reviews
Amanda C. said "After reading revie
about this place, seeing that it takes
an hour wait, I was prepared for the

However, some people forget to describe their food pictures when they uploaded to the YELP and there are a lot of photos don’t come with name tag. As we can see these two photos come from YELP which not include name tag on them. So, even the original restaurant does have the existed photos, there is still high probability that user could not accurately find food pictures on the YELP. At this time, we could not use the one to one function to simply match the food name. At this scenario, developing the application to recognize these pictures without name tag is very necessary for helping the user. Our training model will identify these pictures under the different food categories. Lately, all these unnamed photos will be assigned name tag by our application and it will prioritize the original photos will be chosen from the same restaurant which user queries for. Moreover, there may be another case that new restaurant has just opened, there are not many customer pictures on YELP database yet and it is frequently to see that user write the comment, rather than uploading images to the YELP. We can only recognize the food name first and find similar picture through our database.

2.3 Why this is a project related to this class?

According to the Oracle webpage. The definition of Data mining is the practice of automatically searching large stores of data to discover patterns and trends that go beyond simple analysis. Data mining uses sophisticated mathematical algorithms to segment the data and evaluate the probability of future events. Data mining is also known as Knowledge Discovery in Data(KDD)

They key properties of data mining are:

- Automatic discovery of patterns
- Prediction of likely outcomes
- Creation of actionable information
- Focus on large data sets and database

we collect a lot of food photos as the training data to predict the food category. Also, we use the CNNs model to train these sample pictures and get the new pattern to recognize the photo by certain criterial. Let the machine to identify the photo quality by itself. This also help us deeply understand what the relationship between our data and model is. How could we extract the data from the image and build our own model to identify the new picture. Our project initially needs large data sets to train the model to improve the accuracy.

2.4 Why other approach is no good?

There are certain simple techniques we can think of now: First of all, using the one to one function to directly match the same name tag which already was already uploaded in YELP database and return to the user when they are clicking on the word in the sentence. This approach is quite simple. Since previous users are lazy or forget to put name tag for every picture, it cannot guarantee that there is a corresponding name tag in the database. On the other hand, some new restaurant doesn’t even have that many pictures yet. Second technique is that using the online searching tool, like google, searches the food photo. This may sound like most efficient way to do it but there are still many disadvantages. Firstly, As I mentioned before that costumers are more interested in the pictures of restaurant where they may eat at, original restaurant pictures should be always considered at the first to display. Secondly, online searching tool may return many unrelated pictures, which may confuse the user to recognize the food.

2.5 Why you think your approach is better?

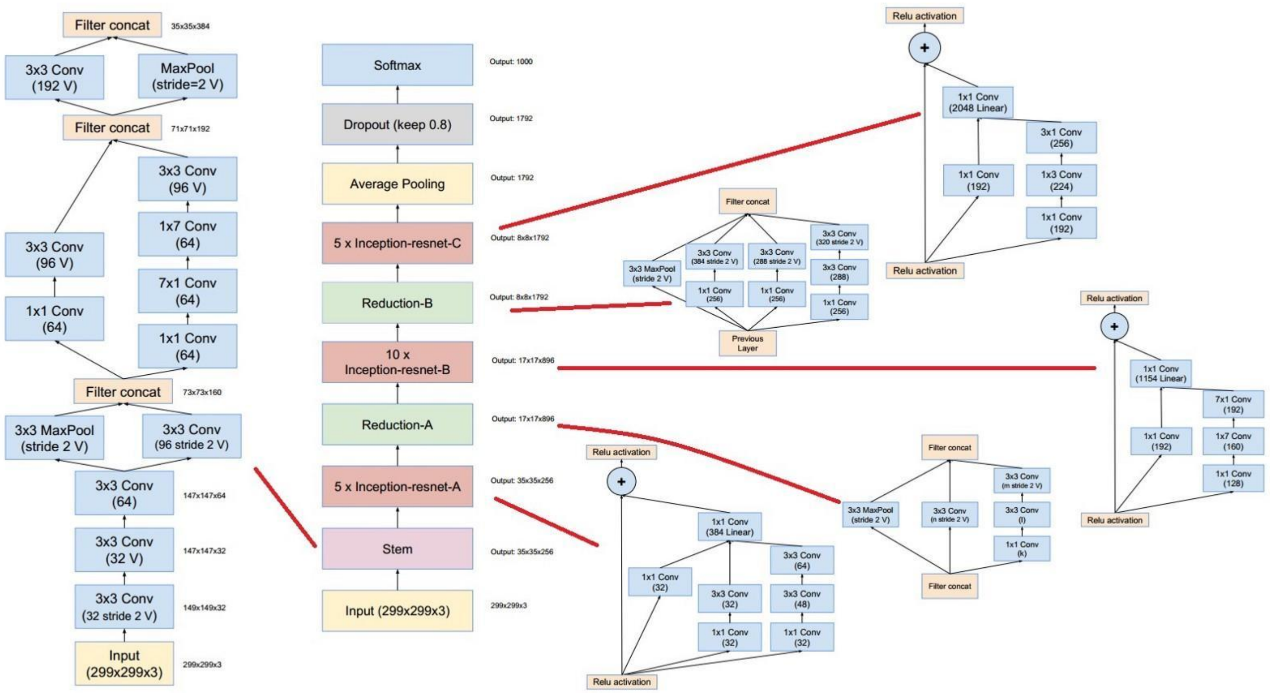
We combine these two simple ways together and add our own approach to better solve this problem. Our simple ideal is that if there is name tag on the Yelp photo dataset, we will automatically give the certain pictures by looking for the previous picture with the same or similar name tag. This could be simply achieved by linear searching. We add CNNs training model to generate the name tag for these pictures that don’t have name tag on them. If there are no photos matching the query in the restaurant of yelp datasets, we will look up our own database and retrieve the most similar photo back to the users. At the end, instead of giving random qualified food picture back, the query may pick up multiply candidates at the waiting pool first, then our CNNs model will select the most delicate photo by comparing the other quality of pictures.

3. Theoretical bases and literature review

3.1 Our Solution:

3.1.1 Fine-tuned DCNN model trained on food image datasets

Google Inception DCNN is an architecture for image recognition, specifically, classification. A model called GoogLeNet won the Imagenet Large Scale Visual Recognition Challenge in 2014, in which models were asked to identify about 1000 different classes of objects after have been trained on 1.2 million images across 1000 categories. It was able to and reach an unprecedentedly high level of performance in general object recognition, thanks to the introduction of Inception. The release of Inception by the end of 2015 has made it greatly convenient to get a high performance as well as a high accuracy on object recognition, because instead of building our own convolutional neural network (CNN) and training it through many epochs, we can take advantage of this suitable pre-trained model and slightly modify it to make it meet our goals.



Inception-ResNet-V3 Architecture

Our solution to food image recognition problem is to use a pre-trained Inception V3 network as the starting point, playing a role of priori knowledge, and then fine-tune it by adjusting the architecture in the output layer (classification layer) of the network, to adapt it to the food image datasets to be able to classify different food properly.

Our food image datasets include three parts: ETH Food-101, UEC FOOD 100 and UEC FOOD 256. These three datasets have 101 food categories with 101,000 images, 100 food categories with 14,000 images and 256 categories with 32,000 images respectively.

A few examples of the test datasets images (a) UEC FOOD 100 and UEC FOOD 256 and (b) ETH FOOD-101

3.1.2 Intelligent & Interactive matching method of food information-displaying to enhance user experience on YELP

The objective of deciding to implement a model to classify food images is to enhance the experience when a user is going through the reviews of a specific restaurant on YELP. Instead of knowing only names of some dishes mentioned in the reviews, users would be able to find a much more useful set of information, which includes not only the name, but also corresponding images, ingredients and the cooking method of selected dishes.

The way we implementing it is to detect if users' are selecting key words (the name of a dish) in a review, and then bind to selecting event a handler that displays the information about the dish. The data needed in part 2 is based on part 1, in which most of the food has been classified into different categories and got at least one tag. We match the key words selected by users with tags already existed in our database to decide which row of information about the food to display.

3.2 Differences from others' solution:

3.2.1 Differences between DCNN and other image recognition methods

Several studies have been focusing on food image recognition.

Bag of words (BoW) treats image features as words. In document classification, a bag of words is a sparse vector of occurrence counts of words; that is, a sparse histogram over the vocabulary. In computer vision, a bag of visual words is a vector of occurrence counts of a vocabulary of local image features. SIFT is a popular model with BoW approaches.

Fisher kernel, named after Ronald Fisher, is a function that measures the similarity of two objects on the basis of sets of measurements for each object and a statistical model. In a classification procedure, the class for a new object (whose real class is unknown) can be estimated by minimizing, across classes, an average of the Fisher kernel distance from the new object to each known member of the given class.

KNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

One major difference between DCNN and above mentioned approaches is that good features are learned automatically using a general-purpose procedure in DCNN, while not in other approaches. It’s the key advantage of DCNN.

Another difference is whether a model has a priori knowledge or not. Since a model is designed to recognize images and it cannot learn everything from scratch and need some knowledge to be in its structure directly. Google Inception DCNN is such a model because as mentioned above, it has been already trained on a datasets which contain 1.2 million images across 1000 categories. While many other models need to be trained from scratch.

3.2.2 Differences on information displaying about food key words in users’ reviews on website YELP

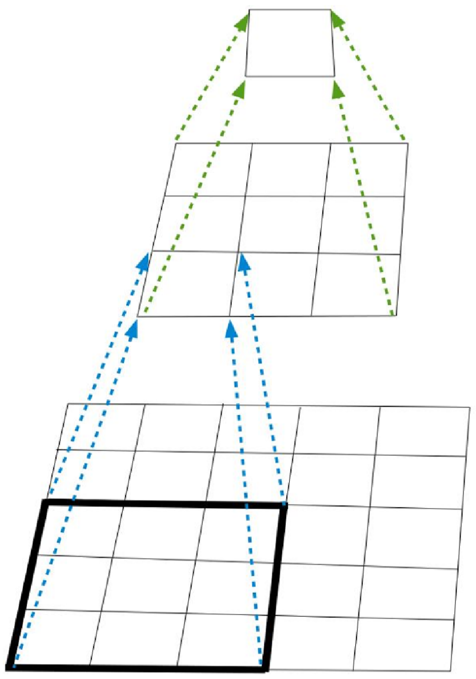
The conventional way for a user of YELP who is gathering information from other users is to read the reviews posted by them. If lucky enough, the text goes with some pictures, but not always in this way. If not, the user would probably get confused by the name of some dishes, which lead to a bad experience. Even though restaurants on YELP always display some pictures of food, they can’t do it in a customizing way. Specifically, all the pictures are displayed in a certain region and have no connections with user reviews. The way our team is going to implement is to build a connection between user reviews and food images. In this way, when user is reading a review which contains a key word of food name, he or she can easily get more information about it by select the key word, which leads to a much better user experience.

3.3 Why our solution is better

3.3.1 Food image recognition solution

1. Less overfitting

   The straightforward way of improving the performance of networks is to increasing the size of them, which always leads to overfitting and time consuming problems. However, Google Inception uses a method called network-in-network, which was first proposed by Lin et al, to address overfitting problem. In conventional convolutional layers, the input is collected and processed using different kernels which represent different features. Instead, network-in-network builds smaller neural networks with more complex structures to abstract the data falling within local receptive field. In this way, overfitting problem could be avoided to the utmost extent.



2. Less training time

   Since fully-connected layers will lead to a dense computation, Google Inception replace them with sparsely-connected ones, not only implement in this way in the main modules, but even within micro-networks. Another main idea of Inception is to think about approximating the optimal, sparse structure of a convolutional vision network by readily available dense components. In this way, Google Inception is faster than many other models when getting trained.

3. More accuracy

   By fine-tuning the classification layers of Google Inception, our model will get a more suitable categories on food. Along with training on datasets ETH Food-101, UEC FOOD 100 and UEC FOOD 256, it'll provide a more accurate recognition on food images.

3.3.2 User experience enhancing solution

1. Saving time and enhancing experience

   Without recognizing and searching pictures by themselves, users who are reading reviews can get all the relevant information of a food with ease. All the information will be recognized and matched by our model automatically. It will save much time and enhance experience greatly for users.

2. Showing detailed information

   Not only knowing the name of some food from other users reviews, user can get more information about the food. Our implementation would extract tags of a specific food, along with ingredients and cooking method, in which users could be very interested.

4. Goals

While textual reviews have become prominent element for potential customers to learn relevant information about the restaurant in Yelp, the inclusion of images can significantly increase the effectiveness of a review. The main goal of our project is to implement a food-recognition-based information displaying system to enhance the review.

The system consisting of three main components: 1) a food recognition or food classifier to predict a tag for each image, 2) a mechanism to map a food name in a review to the most relevant image, 3) a chrome extension, by which user can view what one food looks like while a food name in customers' review list is selected.

4.1 Image recognition or image classifier

We use an image recognition or image classifier to classify all the food images shown in a Yelp restaurant into different categories and then predict a tag for each of them as shown in Figure 1. Once a person selects a dish

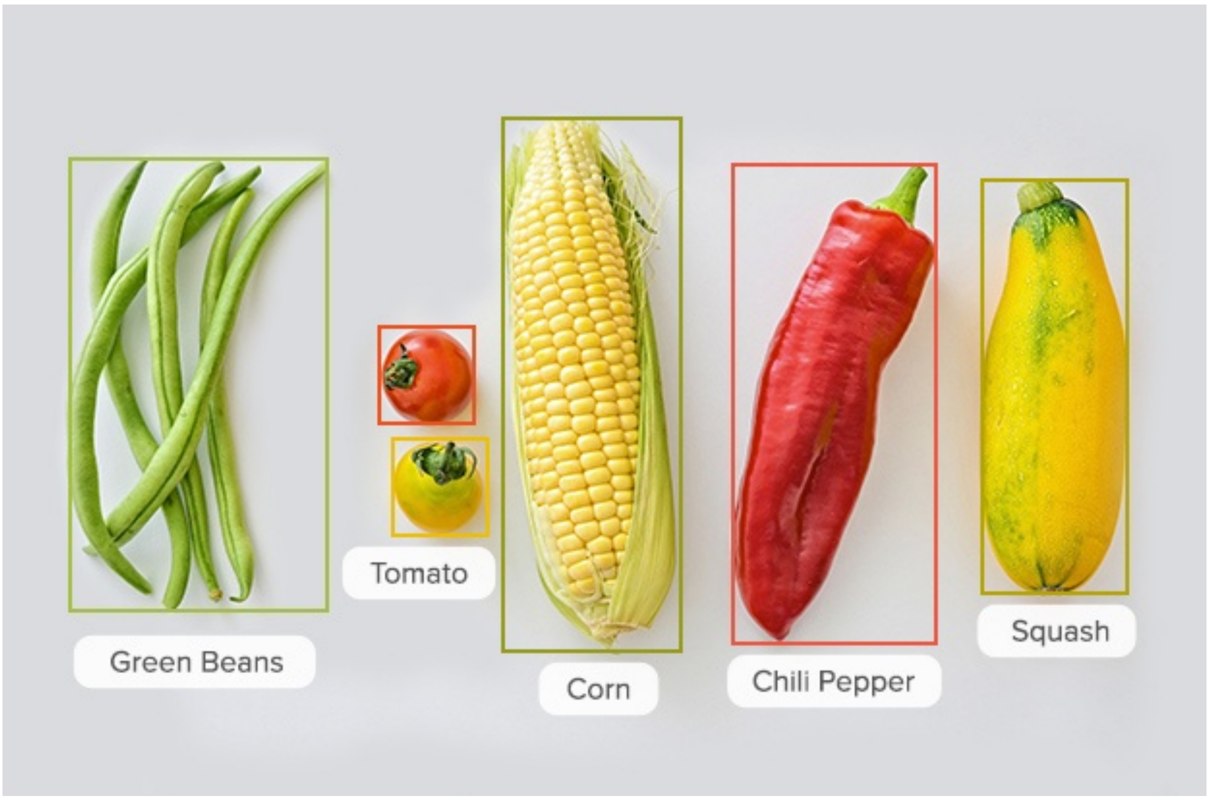


Figure 4-1 Image classifier

name in this restaurant, we can match this dish name with a certain image.

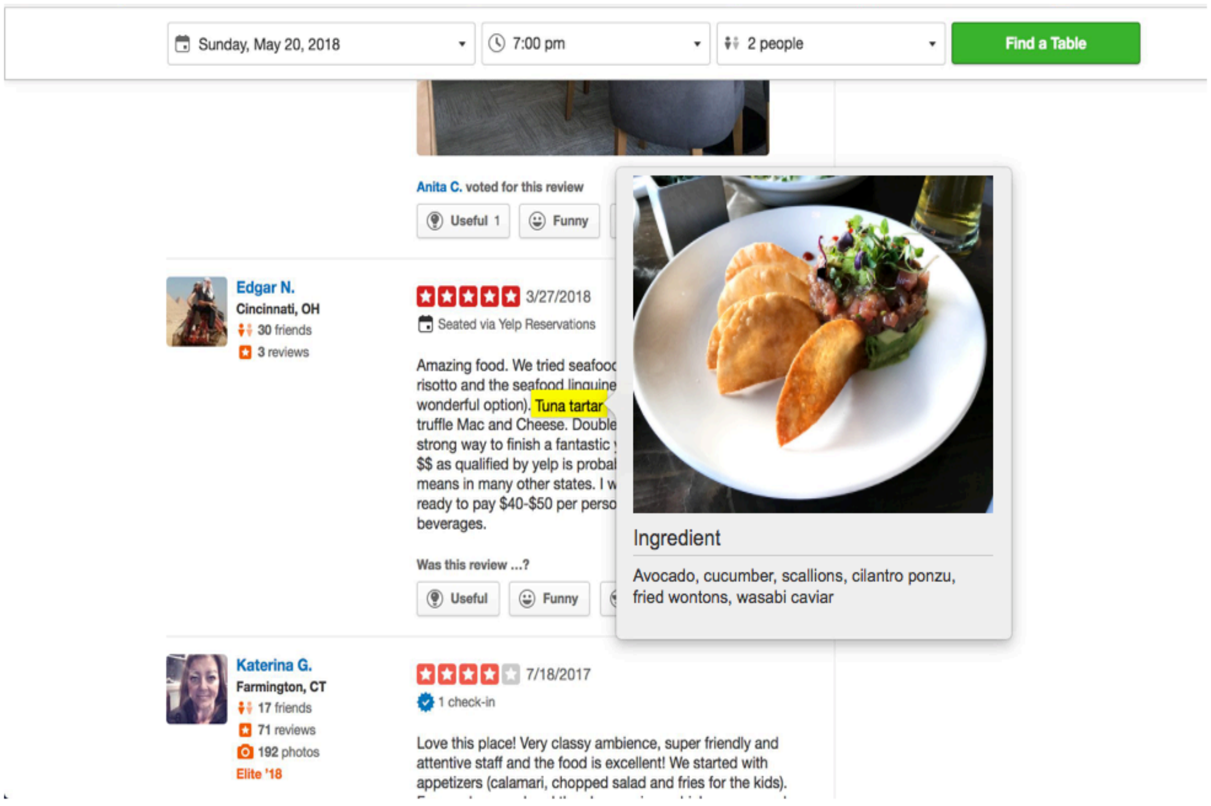
4.2 A mechanism to map review to image

When user is looking through a webpage, he might select many words, our mechanism should detect whether current selected words are components of a food name. For example, if the word “Tiramisu” has been selected, we should say “Okay, it is a desert”. When it occurs to “Tirana”, we should know “it is not a food name”.

After a valid food name is selected, we will use it as keyword to find related image.

4.3 Chrome extension for Yelp

Our prototype is shown in Figure 4-2, when user is looking review list in Yelp, a food name "Tuna Tartar" is highlighted, then the image of it will



Anita C. voted for this review

Edgar N.
Cincinnati, OH
30 friends
3 reviews

3/27/2018
Seated via Yelp Reservations

Amazing food. We tried seafooc
risotto and the seafood linguine
wonderful option). Tuna tartar
truffle Mac and Cheese. Double
strong way to finish a fantastic
$$ as qualified by yelp is probal
means in many other states. I w
ready to pay $40-$50 per perso
beverages.

Was this review ...?

Ingredient

Avocado, cucumber, scallions, cilantro ponzu, fried wontons, wasabi caviar

Katerina G.
Farmington, CT
17 friends
71 reviews
192 photos
Elite '18

7/18/2017
1 check-in

Love this place! Very classy ambience, super friendly and attentive staff and the food is excellent! We started with appetizers (calamari, chopped salad and fries for the kids).

Figure 4-2 Preview Image for selected dish name "Tuna tartar" in a review

pop up.

4.4 Two other optional goals

There are two other goals we want to achieve if we have enough time: 1) a food ingredient recognition to retrieve recipes 2) and image quality scoring algorithm. We can get food ingredient from image by CNN, and when user selects the dish name, an image and its ingredient will be displayed on the screen. Since there are a lot of pictures containing same tag in some restaurants, the results of a food name may contain more than one picture. Image quality scoring algorithm is designed to figure out which picture is best among these candidates as shown in Figure 4-3. We will prefer to choosing the first picture as final result.

5. Methodology

5.1 Methodology for building Chrome extension for Yelp

- how to collect input data?

Input data contains two parts, the first one is current restaurant id and the image urls in current restaurant. Web app can access it through Yelp API and send the restaurant id and image urls to our backend.

The second one is the words selected by the user. They can be collect through JavaScript.

- how to solve the problem?
- algorithm design

  Extensions are made of different, but cohesive, components. Components can include background scripts, content scripts, an options page, UI elements and various logic files. The processing flow of typical usage is as follows:

  1. User opens a page in Yelp.
  2. Web app collect current restaurant information through Yelp API and send it to backend.
  3. When user highlight some words, send these words as a query to backend.
  4. Displaying the results from backend in a popup view.
- language used

  JavaScript

  HTML 5
- Tool

  Atom
- How to generate output

Once the words are selected, they will become highlighted and the result image will be shown in a popup view.

● How to test against hypotheses

Test Case for Web Extension:

Test Case ID	Test Scenario	Test Steps	Expected Results
TU01	Check result with invalid food name.	1.Select some words that can not form a food name by mouse.	Nothing will happen
TU02	Check result with valid food name and the category of food is NOT in database.	1. Select some words that can form a food name by mouse.	A popup view will show but does not display any image.
TU03	Check result with valid food name and the category of food is IN our database.	1. Find a food which doesn't exit in our database. 2. Select this food name in a review by mouse.	A popup view will show and display the image of this food.
TU04	Check result with a food name which matches no image in current restaurant.	1. Find a food which image is not IN current restaurant. 2. Select this food name in a review by mouse.	A popup view will show and display the image which is exiting in our database.
TU05	Check result with a food name which matches only one image in current restaurant.	1. Find a food and make sure there is only one image in current restaurant 2. Select this food name in a review by mouse	A popup view will show and display the only image.

TU06	Check result with a food name which matches more than one image in current restaurant.	1. Find a food and make sure there are more than one image in current restaurant 2. Select this food name in a review by mouse	A popup view will show and display the best one among all the images
TU07	Network error handling	1. Disconnect Wi-Fi 2. Find a valid food name and select it	A popup view will show and display the network error message

5.2 Methodology for mapping review to image

5.2.1 how to generate/collect input data

5.2.1.1 Food dataset

There are some public food datasets for food-related applications such as dietary assessment, computational cooking, food recipe retrieval and so on. We have listed below the famous food datasets available and their characteristics.

Table 1 Public food datasets

Dataset	Image/Category	Comments
PFID dataset (Pittsburgh Food Image Dataset)	4,556 fast food images/	PFID contains only standardized fast food images taken under laboratory conditions.
ChineseFoodNet dataset	185,628 /208 food categories	ChineseFoodNet covers most of popular Chinese food, and these images include web images and photos taken in real world under unconstrained conditions
VIREO-172	a total of 353 ingredient labels and 110,241 images	Chinese Food dataset

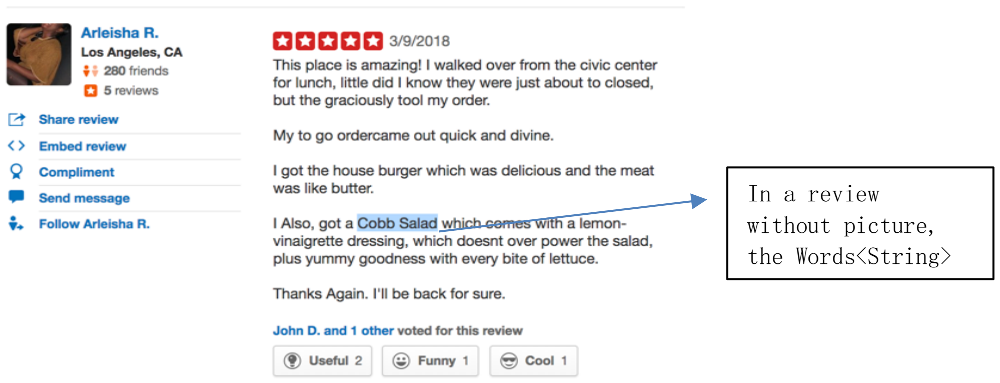
ETHZ-FOOD-101	101’000 images/101 food categories	
UPMC-FOOD-101	Same 101 food categories as ETHZFOOD-101 but with different images	The images of UPMCFOOD-101 are recipe images, in which each has the additional textual information
UNICT-FD889	3,583 images / 889 distinct dishes	The UNICT-FD889 dataset are used for Near Duplicate Image retrieval (NDIR)
UEC-Food100	100 categories	Japanese food dataset
UEC-Food256	256 categories	Japanese food dataset
Indian Food Dataset_1,2,3,4	100 categories each 2000 images	Indian food dataset

To train our food classification algorithm, we use the well-known dataset ETHZ-Food-101. It is a public dataset of 101 food categories, with 101’000 images. However, for Chinese dishes, Food-101 is not sufficient due to its limited categories classification. Compared to other types of food such as American fast food and Italian food, it is more difficult to recognize the images of Chinese dish. First, the images of the same Chinese dish may appear differently as different ingredients and cooking method make it hard to differentiate even for human vision. Second, the noise of images of Chinese dishes is hard to model because of complex noise and a variety of backgrounds, for example dim light, vapor environment, strong reflection, various utensils of Chinese dishes such as color, shape. Thus, as compliment to Food-101 dataset, we use ChineseFoodNet, Indian food dataset_1,2,3,4 to train our classification algorithm specifically regarding to Chinese restaurant (Yelp has restaurant label such as Chinese, American(New), Indian and Mediterranean etc.).

5.2.2 how to solve the problem

5.2.2.1 algorithml2

The pictures on the restaurant webpage are pic-comment pairs. For example, a picture of garlic fries with comment “Garlic Fries”, a picture of mushroom swiss burger with comment “Mushroom Swiss Burger (bun is a bit dry. Its meat and mushrooms were good”. Under most circumstance, pictures are uploaded by contributors without a comment. We make them into a Hashmap for certain restaurant with the picture the comment<String> as the key and ID<Integer> as the value. As mentioned above, quite a few of the comments are empty. Even if not empty, a picture and the comment may have not substantially correlated. For example, a comment “Hurrah it’s my birthday” for a picture of a pasta dish only increases the noise-level rather than providing informative features.



3/9/2018
This place is amazing! I walked over from the civic center for lunch, little did I know they were just about to closed, but the graciously tool my order.

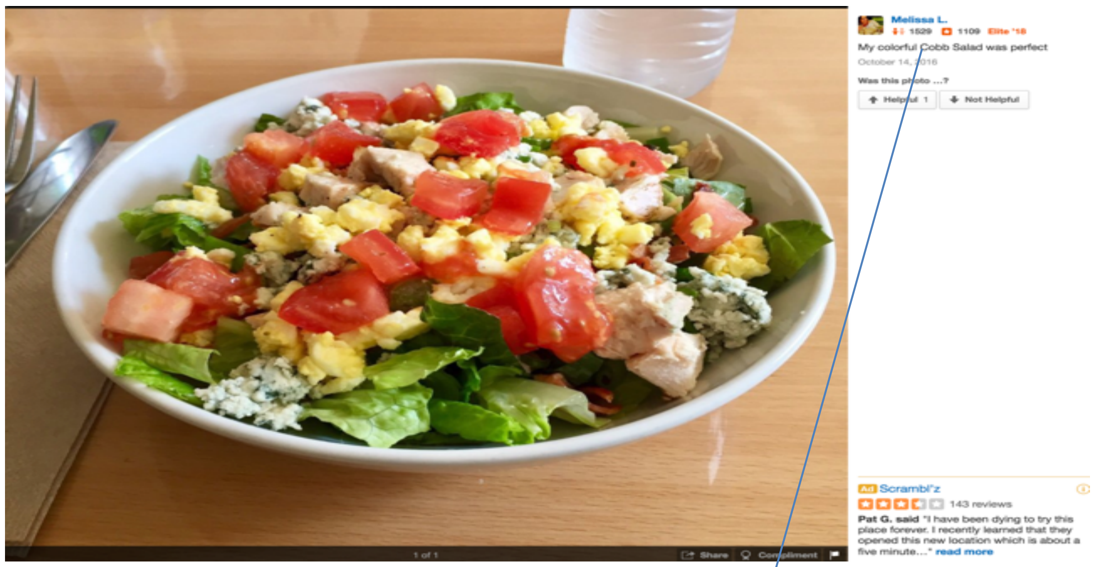
My to go ordercame out quick and divine.

I got the house burger which was delicious and the meat was like butter.

I Also, got a Cobb Salad which comes with a lemon-vinaigrette dressing, which doesnt over power the salad, plus yummy goodness with every bite of lettuce.

Thanks Again. I'll be back for sure.

John D. and 1 other voted for this review


Melissa L.

1529 1109 Elite '16

My colorful Cobb Salad was perfect

October 14, 2016

Was this photo ...?

Ad Scrambl'z

143 reviews

Pat G. said "I have been dying to try this place forever. I recently learned that they opened this new location which is about a five minute...." read more

Comment attached to a picture: My Colorful Cobb Salad was perfect!

Our framework addresses the problem of recommending images for each review in three major steps.

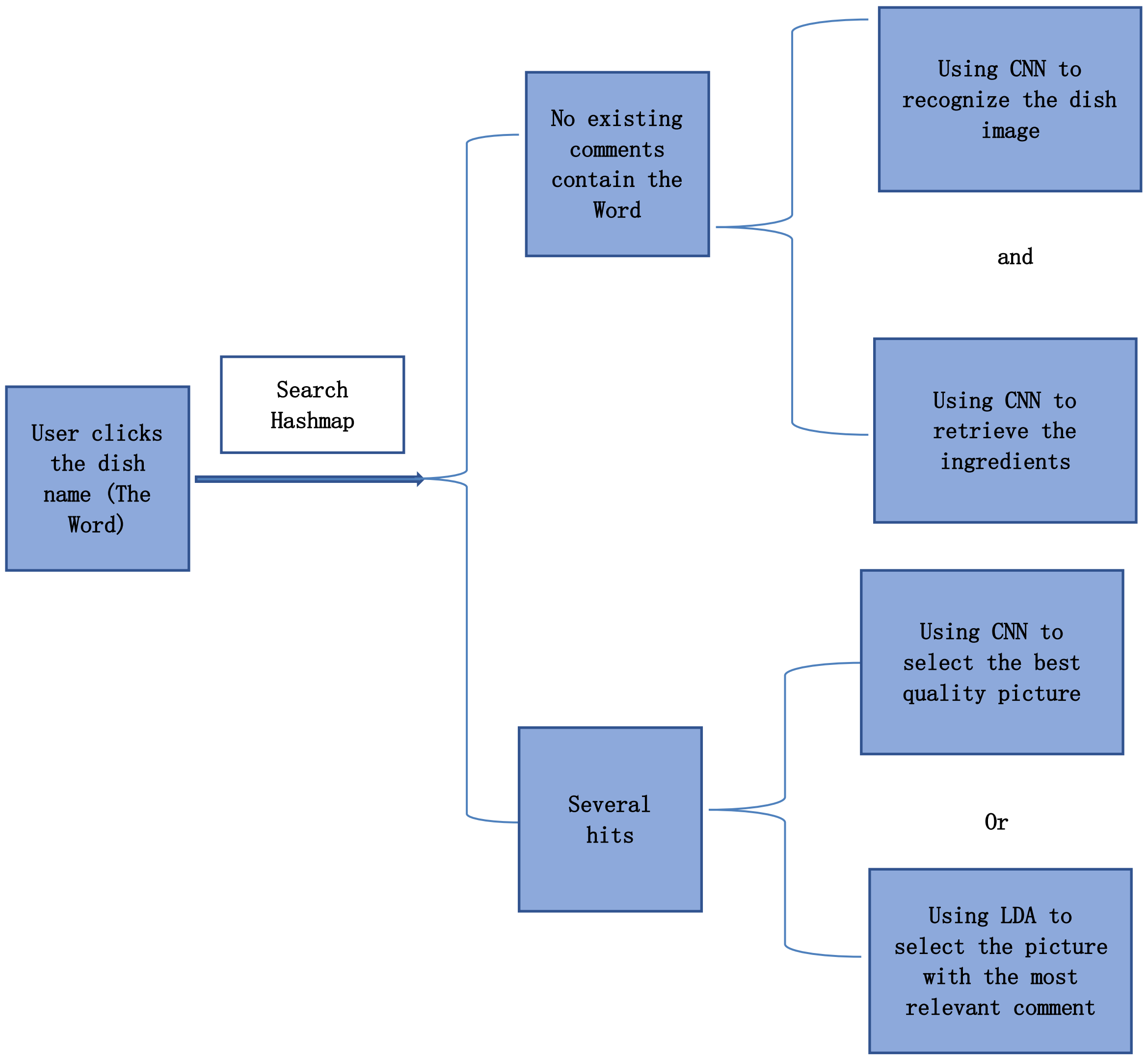
First, we collect the words<String> which the user thinks would be a dish name in a review. we iterate through the Hashmap to check which comment contains the words<String>. There might be several hits. We rank the corresponding pictures using certain criteria.

Second, we select the best pictures among which whose comments all contain the words<String>. Here we have two criteria to implement: (1) the pictures with highest quality in terms of field depth, contrast and alignment; (2) the comments have the most relevant topic terms with the review.

Third, if no existing comment contains the words<String>, we generate name tag for each picture using food recognition techniques until the classification coincides with the words<String>.

Forth, we retrieve ingredients for the words<String> using food ingredient recognition techniques.

Below is the road map for our methodology.


User clicks the dish name (The Word)
Search Hashmap
No existing comments contain the Word
Using CNN to recognize the dish image
and
Using CNN to retrieve the ingredients
Several hits
Using CNN to select the best quality picture
Or
Using LDA to select the picture with the most relevant comment

a. Image classification

We use a Convolutional Neural Network (CNN) image classifier to obtain class probabilities for all images in the test set (all images in a certain restaurant).

Convolutional Neural Network algorithms require that all of the images have the same dimension and are shaped as a square. We will resize the images so that the smallest dimension of the image is 64 or 224 pixels, and then crop the image in the other dimension to obtain a 64-by-64-pixel or

224-by-224-pixel image. We will test and implement CNN models using two Python libraries based on the Theano deep-learning library: Keras, and Lasagne. Keras and Lasagne provide high-level functions
for deep learning algorithms, including convolution, pooling, and fully-connected layers, as well as backpropagation and optimization routines, whereas Theano provides the back- end of the computation and includes GPU support.

We will use a number of different CNN models to evaluate their accuracy. One of these models was based on the CIFAR10 data while the others were designed to work with the ImageNet data: VGG-16, VGG-19 and GoogleNet. The CIFAR10 model is relatively simple, with only 11 layers. The VGG-16 model adds four convolutional layers and one fully-connected layer, which significantly increases the complexity of the model. The VGG-19 and GoogleNet models add even a larger number of layers, consisting of 19 and 22, respectively. We also used MATLAB 's Bag-of-Features with SVM classification algorithm as a baseline. We used six-fold cross-validation for evaluation of all these approaches.

b. Image quality scoring model

The three most important features of a photo are depth of field, color contrast and alignment. The depth of field measures how much of the image is in focus. Using a “shallow” depth of field can be an excellent way to distinguish the subject of an image from its background. In many cases, the most beautiful images of a given restaurant were very sharply focused on a specific entrée.

In particular, we found that a good proxy for quality is whether a photo was taken by a digital single-lens reflex camera, or **DSLR**. These cameras give the photographer more control over which parts of the image are in focus, by adjusting the lens type and aperture size. Further, DSLR sensors are larger and more sensitive to light, allowing great photos to be taken in even very dim situations. Finally, people who regularly use DSLR cameras may have more experience and skill in capturing higher quality images.

Training our model on such photos allows it to learn important photo features and recognize great photos even when they are not taken by a DSLR camera.

We will try several methods of training this model. Initially, we will collect 100,000 DSLR and non-DSLR images to use as positive and negative labels, respectively, and feed these into a model known as AlexNet, which was created by researchers at the University of Toronto in 2012. To improve the accuracy of this model, we will train an additional model with more than ten times the previous amount of training data. Finally, we will test a model called GoogLeNet, which was developed by researchers at Google in 2014 and achieved state of the art performance by having significantly deeper layers than previous top-of-the-line models.

In each of these cases, we will further evaluate the model against a dataset of thousands of images manually evaluated by Yelp engineers, which consisted of only those images which we could confidently say were very good or very bad. We assume that with each iteration, our ability to correctly identify good and bad photos will improve.

c. Topic modeling and review enhancement

We leverage Latent Dirichlet Allocation (LDA) to model the topics of the reviews. We obtain the topic with the highest probability and select the top t representative terms of that topic, regardless if they appear on the review or not. For each review, we recommend the top ϕ images based on the presence of the t representative terms in the review and in the comments of the subset of images available only for the business for which the review was written. An image is ranked higher for a particular review if a representative term is present both in the image comment and in the review, compared to an image which contains the representative term only in its comment.

We start by selecting images using representative terms that are present in both the review and the image comment. If ϕ images cannot be found, we select images for which comment contain representative terms but the review does not. This process ensures that the image selection is not solely driven by overlaps between a review and a comment, rather reviews and image comment without any overlap may become candidates for potential mapping due to the use of topical terms during the ranking.

d. Ingredient recognition model

We will build the ingredient recognition model based on our dataset, because it has clean recipe information. For ingredient information, we will manually filter out stop words and commonly used units like spoon and jar.

5.2.2.2 language used

To perform the image recognition the language used here is Python library Keras and Lasagne. Keras and Lasagne are high-level neural network API which is capable of running on top of TensorFlow. It supports

Convolutional network. The only difference between them is Keras is less flexible and less extensible than Lasagne.

6. Implementation

6.1 Code

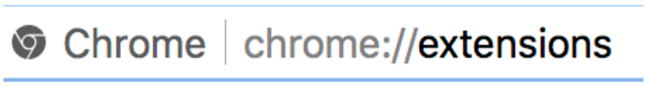
6.1.1 Food image category and quality recognition

1) download Python module: pillow, h5py, scipy, matplotlib, haul, tensorflow (1.2.0), keras(1.2.2). The last two has strict version requirement.
2) make sure the following file is properly stored: food_classes.txt, model4b.10-0.68.hdf5 and model68-2.7.hdf5. The latter two are ready-trained classification model to be implemented.
3) Run imageDownLoader.py, it will call imageRecognition.py and qualityRecognition.py.
4) The result will give the URL of the image, the classification of the image and the quality index. The classification and quality index will directly stored in the database, instead of been printed.

6.1.2 Chrome extension

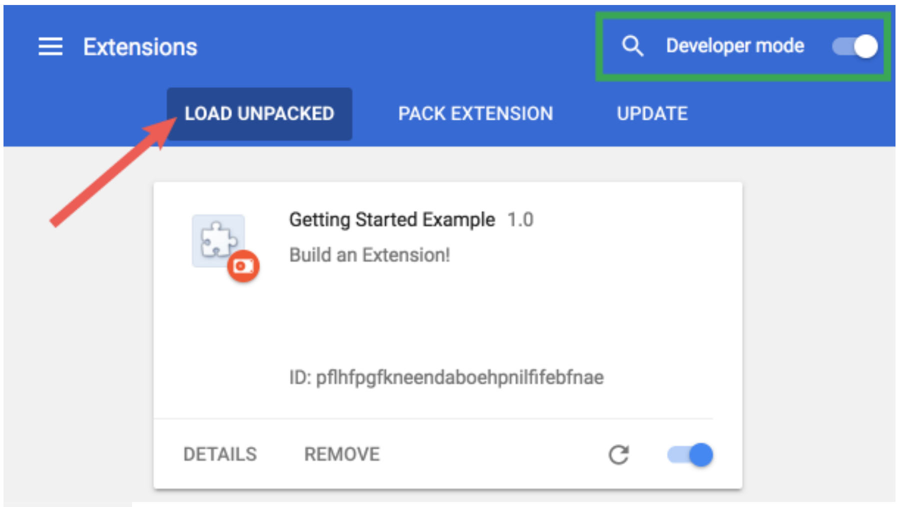
1. upload extension package

1) open “chrome://extensions”



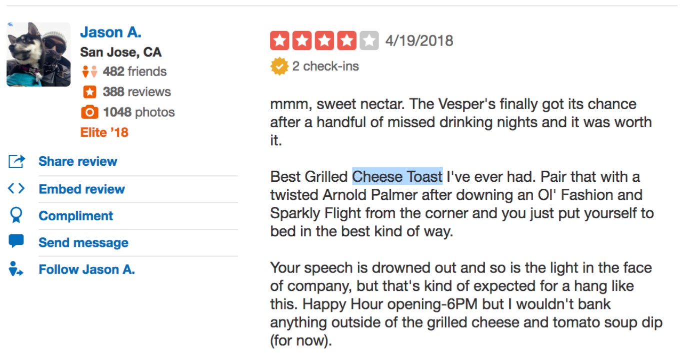
Chrome | chrome://extensions

2) load unpacked “preview”



this icon next to url bar shows that the extension is installed properly.

2. open http://yelp.com and click into a restaurant, then select any dish name in review area, then the corresponding image shows up.


Jason A.
San Jose, CA
482 friends
388 reviews
1048 photos
Elite '18

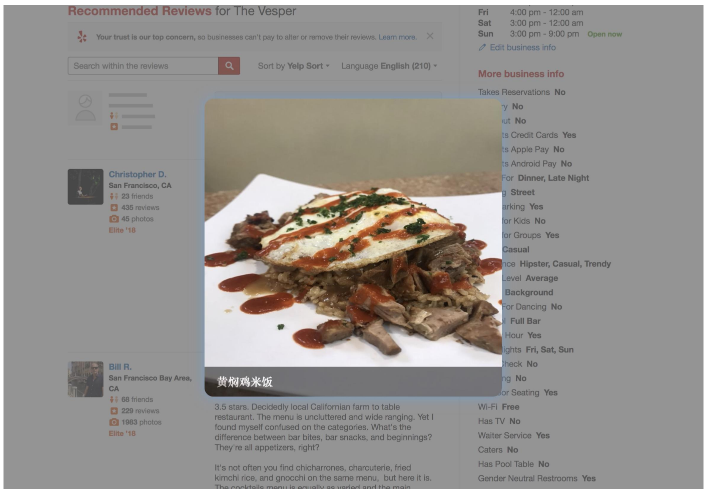
Share review
Embed review
Compliment
Send message
Follow Jason A.

4/19/2018
2 check-ins

mmm, sweet nectar. The Vesper's finally got its chance after a handful of missed drinking nights and it was worth it.

Best Grilled Cheese Toast I've ever had. Pair that with a twisted Arnold Palmer after downing an Ol' Fashion and Sparkly Flight from the corner and you just put yourself to bed in the best kind of way.

Your speech is drowned out and so is the light in the face of company, but that's kind of expected for a hang like this. Happy Hour opening-6PM but I wouldn't bank anything outside of the grilled cheese and tomato soup dip (for now).



6.1.3 Image preprocess

```
import scipy.misc
```

```
from scipy.misc import imresize

scipy.misc.Imresize(*args, ** kwds)
```

6.1.4 Database and Server

To run web crawl, fetch haul with pip:

```
pip install haul
```

Type below command in the terminal

```
python imageDownLoader.py
```

To run server, fetch these libraries with pip:

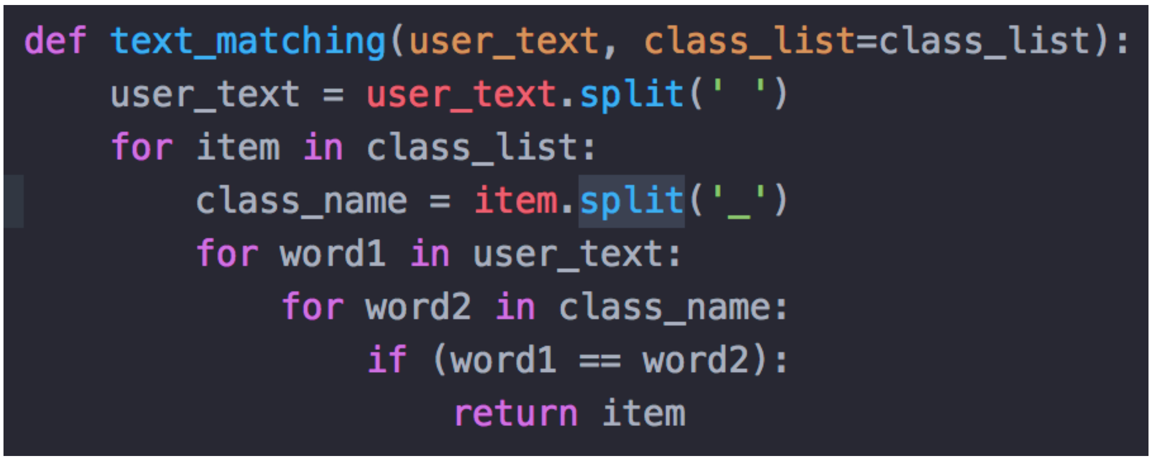
```
pip flask
pip dataset
```

Type below command in the terminal

```
python app.py
```

6.1.5 Mapping

We defined a function called text_matching, which takes the text selected by the user as input, and map it into(output) one of the 101 classes in food-101 dataset.

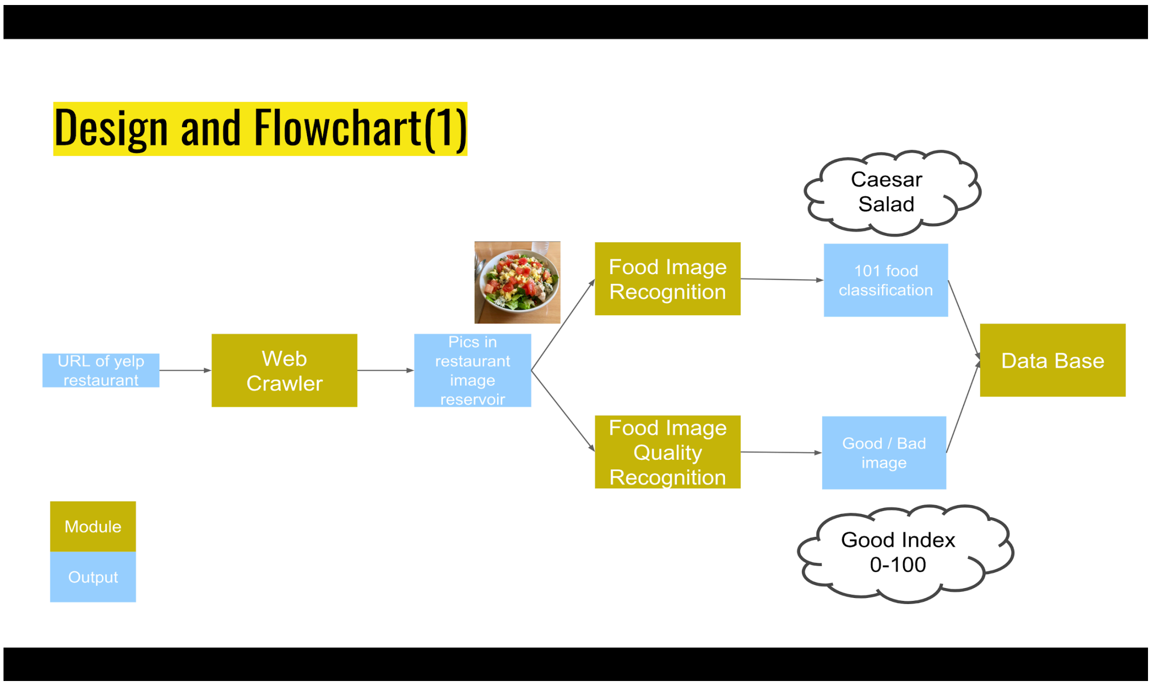


```
def text_matching(user_text, class_list=class_list):
    user_text = user_text.split(' ')
    for item in class_list:
        class_name = item.split('_')
        for word1 in user_text:
            for word2 in class_name:
                if (word1 == word2):
                    return item
```

6.2 Design document and flowchart

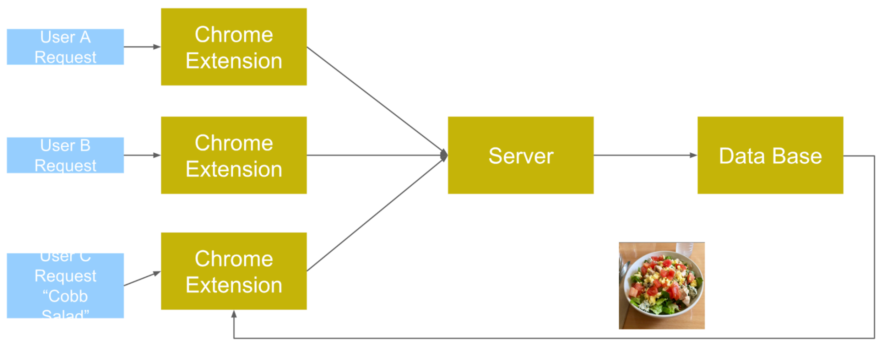
Following is the design and flowchart of our whole project:

we have 4 modules in the first part: Web Crawler, Food image recognition, food image quality recognition and Data Base. When the URL of a specific restaurant is provided, the web crawler will connect the web page and crawl all the pics in the restaurant image reservoir. Then, our food image recognition and food image quality recognition system will recognize their classes and quality one by one. Afterwards, all the information about the pic such as: restaurant business id, image id, image URL, image category and quality rate are installed in the database. This process should be done before the users get access to the restaurant page because the process takes very long time to accomplish.



The second part has 3 module: Chrome extension, Server and DataBase. When a users selects a dish name in his Chrome, the plugin will send the request to the server. The server responds by searching the business ID in Yelp and using the business ID to check the database of certain restaurant with the business ID. From the database, we find the category that mapping the dish name. From that category we find the best quality image and return that image to the users.

Design and Flowchart(2)



7. Data analysis and discussion

7.1 Output generation

7.1.1 Food image recognition

We tried to retrain the Google InceptionV3 model, it is pretrained on ImageNet. We want to train it on the Food-101 data. However, due to the limit of computer hardware, instead of training 101 classes, we tried to train 10 classes (1000 pics for each class and total 10,000 pics). We use 10 crops per example and taking the most frequent predicted class. We use Stochastic Gradient Descent (SGD) with a quickly decreasing learning schedule. The hardware condition is Nvidia GeForce GTX 1080/8 GB of memory, 16 GB of system RAM, as well as a 6-core Intel Core i7. It is running 64-bit Ubuntu 16.04 and using the Anaconda Python distribution.

However, each epoch (32) of training went 50 minutes, and it is too long for the repetitive work. We decide to borrow an InceptionV3 model pretrained on Food-101 data to recognize the food image. The training source code is on the website: http://blog.stratospark.com/creating-a-deep-learning-ios-app-with-keras-and-tensorflo.html . The trained model is available at following website.

https://s3.amazonaws.com/stratospark/food-101/model4b.10-0.68.hdf5

We only list the code we call the model to do the recognition work in Appendices 10.1.1.

The version requirement of the recognition process is as follows:

- Tensorflow 1.2.0 (Strict)
- Keras 1.2.2 (Strict)
- h5py
- scipy
- matplotlib
- haul

We are able to achieve 86.97% Top-1 Accuracy and 97.42% Top-5 Accuracy

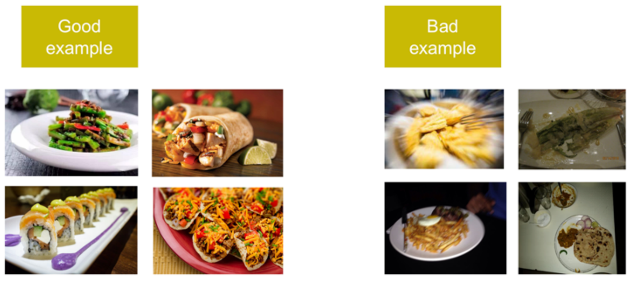
7.1.2 Food image quality recognition

7.1.2.1 Food image quality model training

As food image quality recognition is quite a new idea, we want to train our own quality recognition model. We retrain the Google InceptionV3 model which is pretrained on ImageNet. The logic behind it is that we use the weights and bias parameters got from training ImageNet as initial value and train most of the layers all over again. It is a solution of transfer learning.

In terms of image preprocessing, for each picture with .jpg suffix, either smaller than (299, 299, 3) or larger than (299, 299, 3), we resize the shortest edge to 299, and adjust the other edge proportional accordingly. We use 10 crops per image with each cropped image with size (299, 299, 3), because InceptionV3 model only accepts image size (299, 299, 3). The crop method is get 10 (299,299,3) images from the position of upper left, upper right, lower left, lower right, center, etc of the original image. Among the 10 images, we obtain the average of the probability that the image is belong to a good pic and using this average as the Goodness Index of the photo.

We create our own image quality dataset with 100 good photos and 100 bad photos. The selection is inevitably subjective, so we ask other friends to trim our selection.

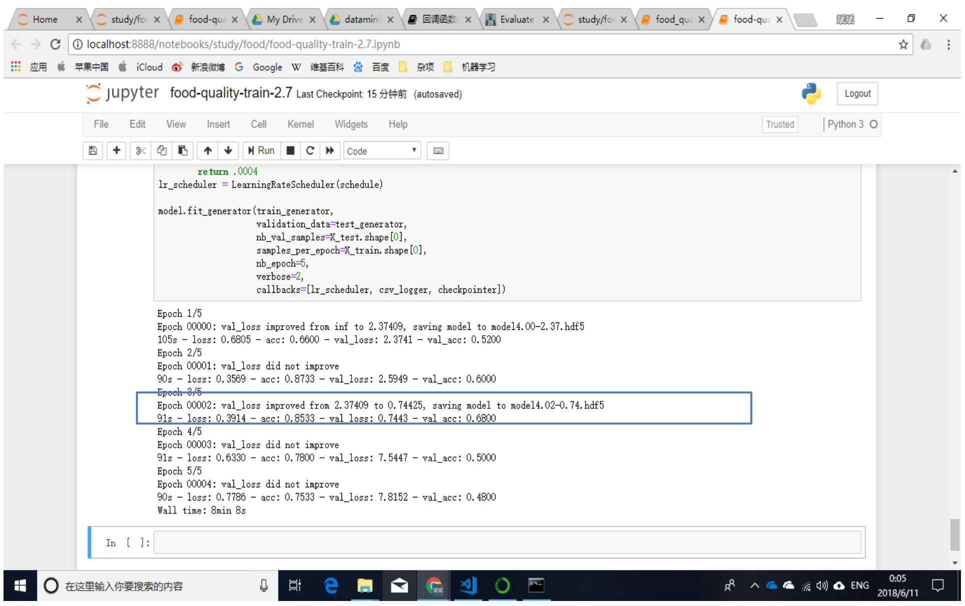


Normally speaking, the good food images are those of shallow field of depth, good color contrast and good alignment. The bad food images are those of characteristics such as out of focus, blur or dim background.

We set the training set to be 80% of the total image, test set to be the rest 20%. We are able to achieve **68% Accuracy**. 5 total epochs of training went 30 minutes. The version requirement of the image quality training process is as follows:

- Python 3.6
- Tensorflow 1.2.0
- Keras 1.2.2

Following is the result of the training process and we only load the model of accuracy 68% after epoch 3/5 to use in our following image quality recognition process.



```
        return .0004
lr_scheduler = LearningRateScheduler(schedule)

model.fit_generator(train_generator,
                    validation_data=test_generator,
                    nb_val_samples=X_test.shape[0],
                    samples_per_epoch=X_train.shape[0],
                    nb_epoch=5,
                    verbose=2,
                    callbacks=[lr_scheduler, csv_logger, checkpointer])
```

```
Epoch 1/5
Epoch 00000: val_loss improved from inf to 2.37409, saving model to model4.00-2.37.hdf5
105s - loss: 0.6805 - acc: 0.6600 - val_loss: 2.3741 - val_acc: 0.5200
Epoch 2/5
Epoch 00001: val_loss did not improve
90s - loss: 0.3569 - acc: 0.8733 - val_loss: 2.5949 - val_acc: 0.6000
Epoch 3/5
Epoch 00002: val_loss improved from 2.37409 to 0.74425, saving model to model4.02-0.74.hdf5
91s - loss: 0.3914 - acc: 0.8533 - val_loss: 0.7443 - val_acc: 0.6800
Epoch 4/5
Epoch 00003: val_loss did not improve
91s - loss: 0.6330 - acc: 0.7800 - val_loss: 7.5447 - val_acc: 0.5000
Epoch 5/5
Epoch 00004: val_loss did not improve
90s - loss: 0.7786 - acc: 0.7533 - val_loss: 7.8152 - val_acc: 0.4800
Wall time: 8min 8s
```

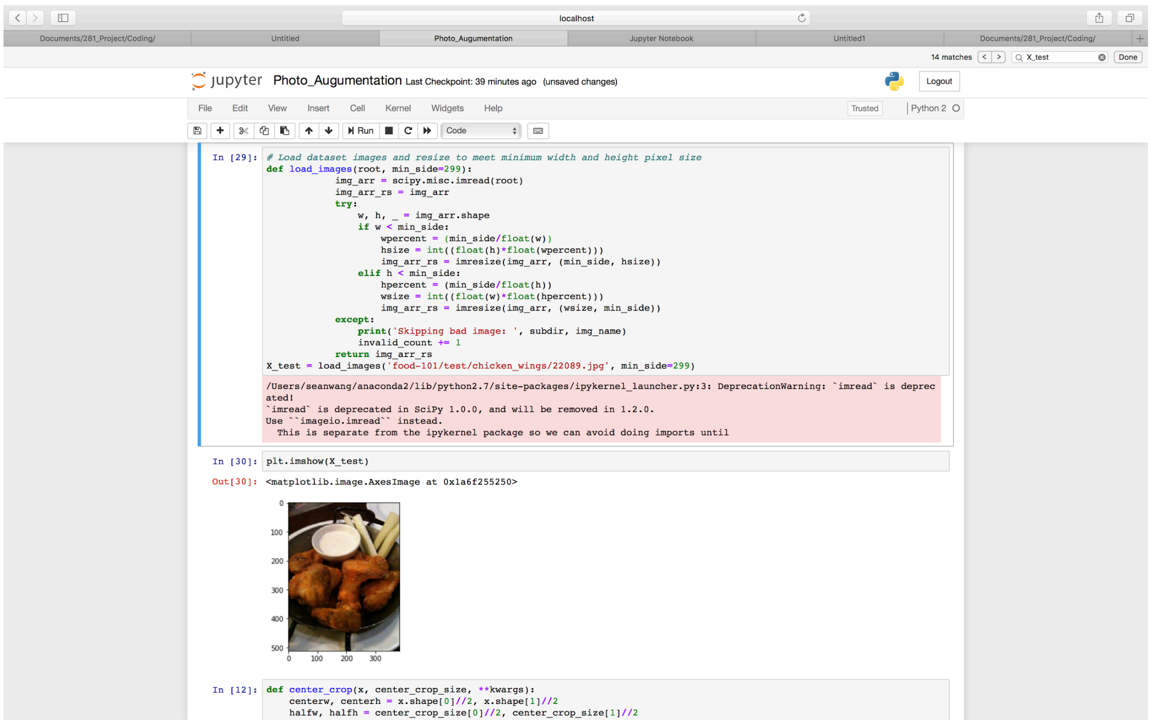
localhost:8888/notebooks/study/food/food-quality-train-2.7.ipynb
jupyter food-quality-train-2.7 Last Checkpoint: 15 分钟前 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
In [ ]:
在这里输入你要搜索的内容
ENG
0:05
2018/6/11

7.1.2.2 Food image quality recognition

We generate of the recognition model in the file of *model68-2.7.hdf5*, We only list the code we call the model to do the recognition work in Appendices 10.1.2.

7.1.3 Food image Pre Processing

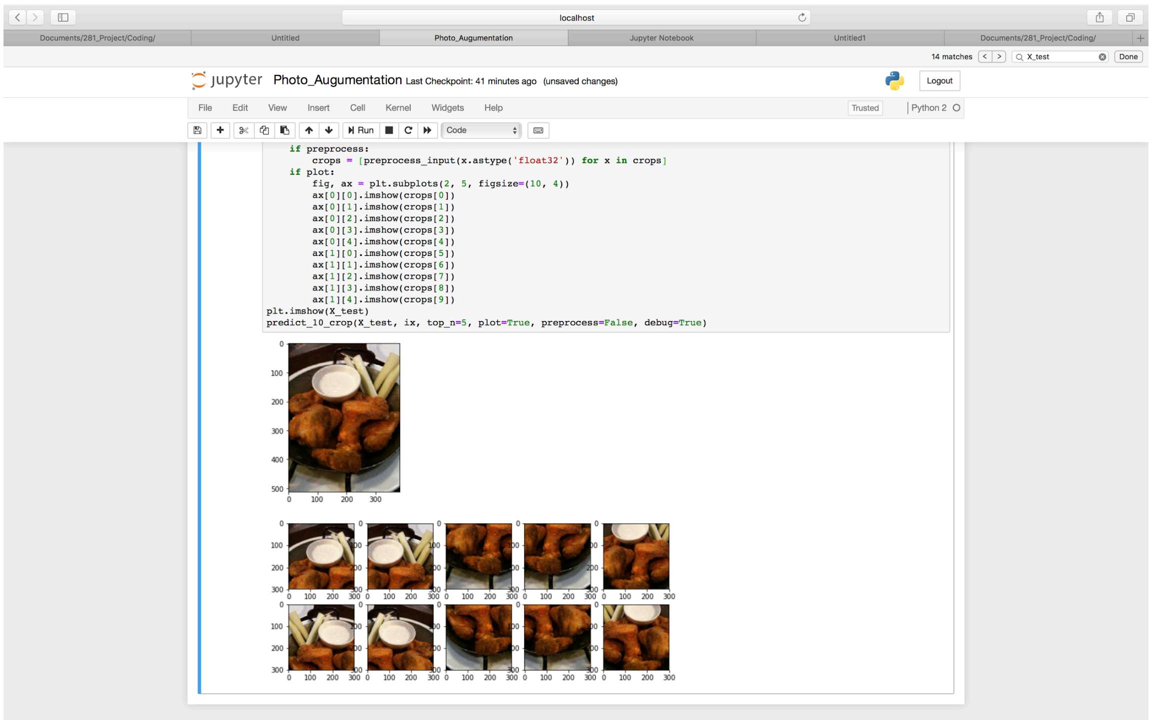
In order to feed our images to Google Inception, we need to preprocess the images to optimize size first. There are a lot of ways to resize the photo by 299 * 299. One of the most simple ways is to call the preprocessing_funtion which is directly provided by Keras library. However, simply resizing the photo may make photo to be cropped out directly or distort the shape of the original image during the preprocessing process. In order to get rid of these negative factors, we wrote our own resizing function to process the photos. In this case, we have used the scipy.misc.imresize which is provided by matlab open library to resize the image. The scipy.misc. library provide the shape function to directly access the width and height of each image. After we got the width and height of the image, the easiest way is to proportionally enlarge the image when the size of image does not satisfy the minimal size 299 * 299.



```
In [29]: # Load dataset images and resize to meet minimum width and height pixel size
         def load_images(root, min_side=299):
                     img_arr = scipy.misc.imread(root)
                     img_arr_rs = img_arr
                     try:
                         w, h, _ = img_arr.shape
                         if w < min_side:
                             wpercent = (min_side/float(w))
                             hsize = int((float(h)*float(wpercent)))
                             img_arr_rs = imresize(img_arr, (min_side, hsize))
                         elif h < min_side:
                             hpercent = (min_side/float(h))
                             wsize = int((float(w)*float(hpercent)))
                             img_arr_rs = imresize(img_arr, (wsize, min_side))
                     except:
                         print('Skipping bad image: ', subdir, img_name)
                         invalid_count += 1
                     return img_arr_rs
         X_test = load_images('food-101/test/chicken_wings/22089.jpg', min_side=299)

         /Users/seanwang/anaconda2/lib/python2.7/site-packages/ipykernel_launcher.py:3: DeprecationWarning: `imread` is deprec
         ated!
         `imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
         Use ``imageio.imread`` instead.
           This is separate from the ipykernel package so we can avoid doing imports until

In [30]: plt.imshow(X_test)

Out[30]: <matplotlib.image.AxesImage at 0x1a6f255250>

In [12]: def center_crop(x, center_crop_size, **kwargs):
             centerw, centerh = x.shape[0]//2, x.shape[1]//2
             halfw, halfh = center_crop_size[0]//2, center_crop_size[1]//2
```

At the beginning, we would iterate the every sub folders under the food 101 directory. Resizing function load every image from each sub folder and check each of them by single time. The function appends every image which is already processed and load to the memory for training the model at the end.

Any photos of width or height smaller than min_size will be resized. We could take proper-sized crops during image augmentation. The function will iterate every image under the subfolder within food 101 dataset and check if the size of photo smaller than the 299, then it enlarges the photo by getting the ratio of the proportion of width or height which not satisfy the minimal size 299 first. Then we multiply this ratio to unchosen height or width. Later, we assume that every image of width and height are above or at least equal to 299. Since google inception v3 only accepts 299 * 299. We still need to crop the photo to feed to our model. So we decide to crop the image to following crops : Upper Left, Upper Right, Lower Right, Lower Left, Lower Right, Center. Using the CNN model, it gives each tag to each crop image. we count the total high frequency index by the count. We will get the most frequent tag from these ten photos and decide which this photo belongs to which tag. We will see these ten photos here by processing the single photo.



```
    if preprocess:
        crops = [preprocess_input(x.astype('float32')) for x in crops]
    if plot:
        fig, ax = plt.subplots(2, 5, figsize=(10, 4))
        ax[0][0].imshow(crops[0])
        ax[0][1].imshow(crops[1])
        ax[0][2].imshow(crops[2])
        ax[0][3].imshow(crops[3])
        ax[0][4].imshow(crops[4])
        ax[1][0].imshow(crops[5])
        ax[1][1].imshow(crops[6])
        ax[1][2].imshow(crops[7])
        ax[1][3].imshow(crops[8])
        ax[1][4].imshow(crops[9])
plt.imshow(X_test)
predict_10_crop(X_test, ix, top_n=5, plot=True, preprocess=False, debug=True)
```

7.2 Output analysis

7.2.1 Food recognition output analysis

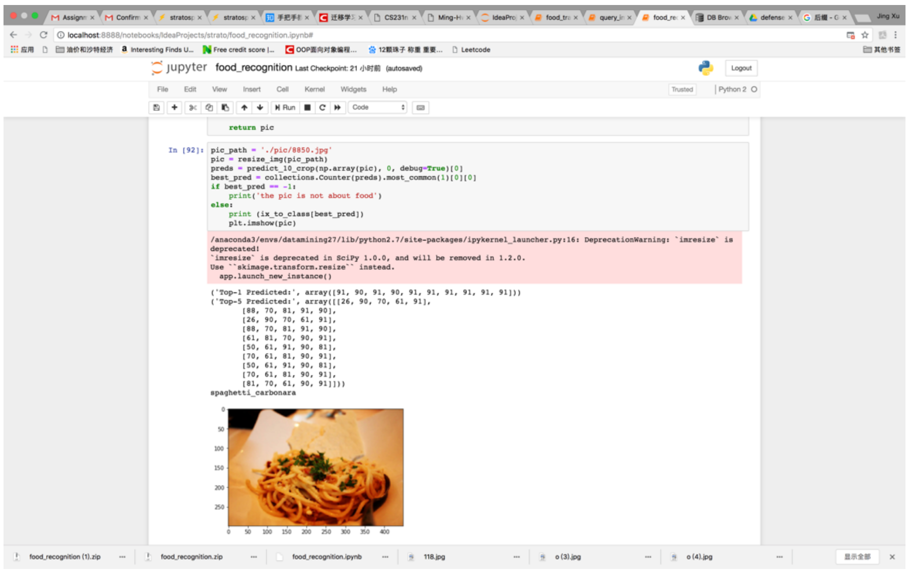
We have already discussed the output of the training part in above paragraphs. Here we discuss about the output in the recognition process.

The recognition process receives a local path of a image and output a 101 classification to be stored in the database. Normally, the result is shown as following:

```
array([91, 90, 91, 90, 91, 91, 91, 91, 91, 91]))
```

8 out of the 10 sub-images give the classification as `spaghetti_carbonara`, 2 out of the 10 sub-images give the classification as `spaghetti_bolognese`. So the final

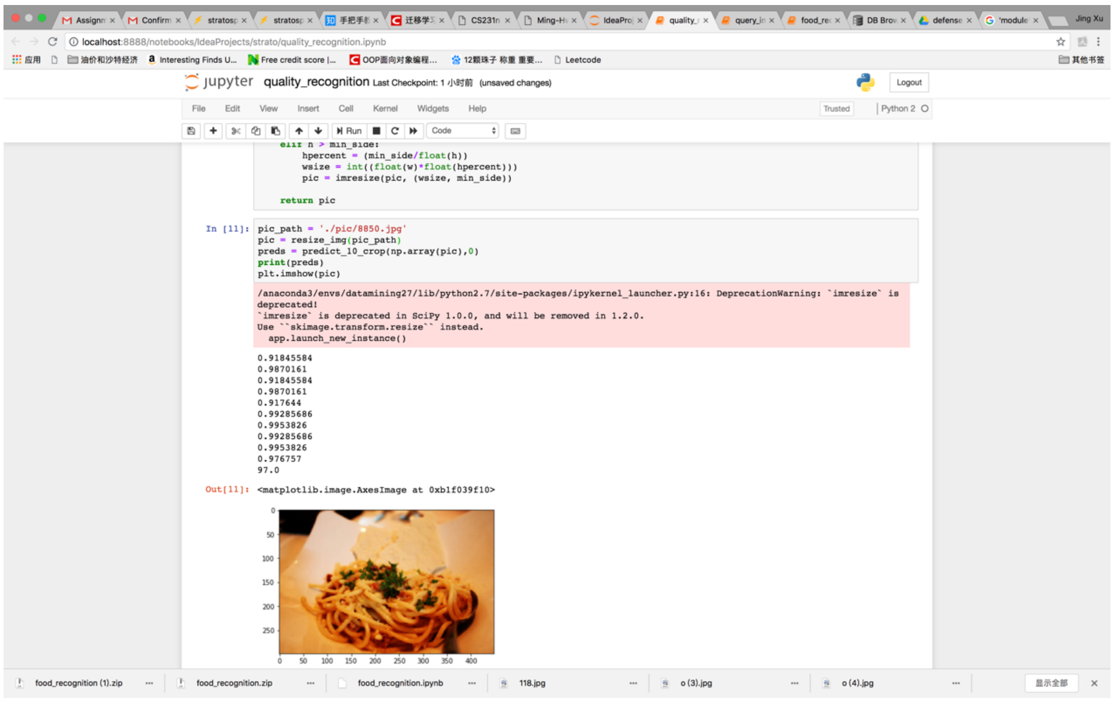
classification of the image belongs to spaghetti_carbonara which is quite right and shown in the last line of the output.



```
    return pic

In [92]: pic_path = './pic/8850.jpg'
         pic = resize_img(pic_path)
         preds = predict_10_crop(np.array(pic), 0, debug=True)[0]
         best_pred = collections.Counter(preds).most_common(1)[0][0]
         if best_pred == -1:
             print('the pic is not about food')
         else:
             print (ix_to_class[best_pred])
             plt.imshow(pic)

/anaconda3/envs/datamining27/lib/python2.7/site-packages/ipykernel_launcher.py:16: DeprecationWarning: `imresize` is
deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``skimage.transform.resize`` instead.
  app.launch_new_instance()

('Top-1 Predicted:', array([91, 90, 91, 90, 91, 91, 91, 91, 91, 91]))
('Top-5 Predicted:', array([[26, 90, 70, 61, 91],
       [88, 70, 81, 91, 90],
       [26, 90, 70, 61, 91],
       [88, 70, 81, 91, 90],
       [61, 81, 70, 90, 91],
       [50, 61, 91, 90, 81],
       [70, 61, 81, 90, 91],
       [50, 61, 91, 90, 81],
       [70, 61, 81, 90, 91],
       [81, 70, 61, 90, 91]]))
spaghetti_carbonara
```

7.2.2 Food image quality recognition output analysis

The recognition process receives a local path of the image and output a pic goodness index to be stored in the database. Normally, the result is shown as following:



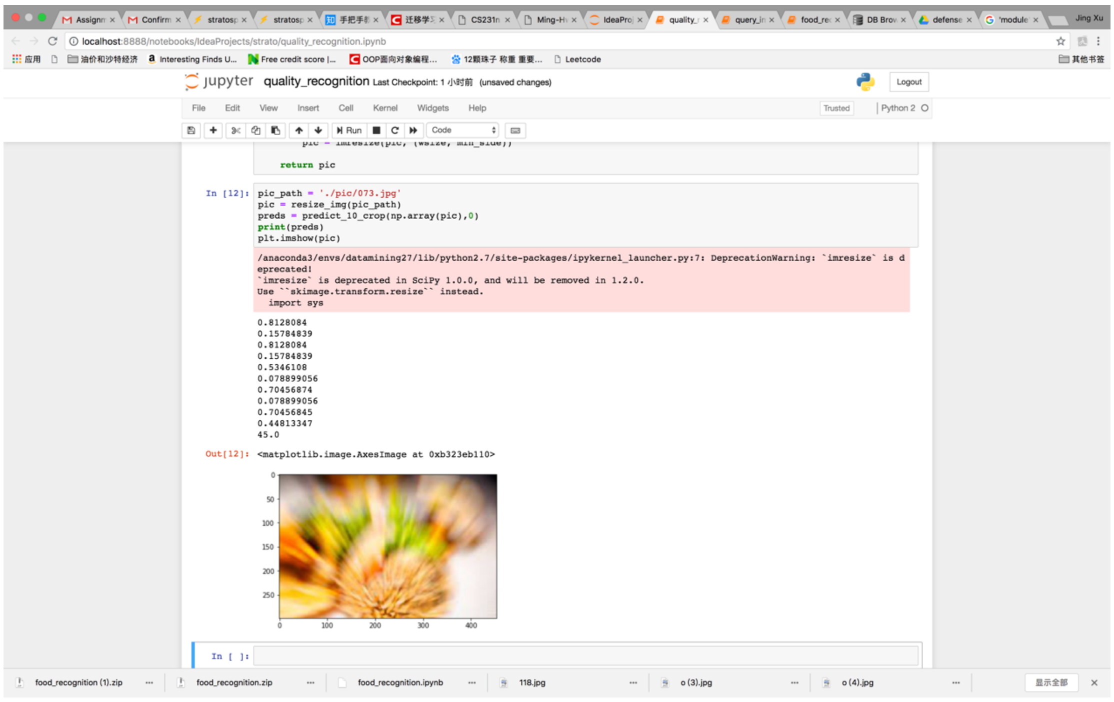
```
    elif h > min_side:
        hpercent = (min_side/float(h))
        wsize = int((float(w)*float(hpercent)))
        pic = imresize(pic, (wsize, min_side))

    return pic

In [11]: pic_path = './pic/8850.jpg'
         pic = resize_img(pic_path)
         preds = predict_10_crop(np.array(pic),0)
         print(preds)
         plt.imshow(pic)

/anaconda3/envs/datamining27/lib/python2.7/site-packages/ipykernel_launcher.py:16: DeprecationWarning: `imresize` is
deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``skimage.transform.resize`` instead.
  app.launch_new_instance()

0.91845584
0.9870161
0.91845584
0.9870161
0.917644
0.99285686
0.9953826
0.99285686
0.9953826
0.976757
97.0

Out[11]: <matplotlib.image.AxesImage at 0xb1f039f10>
```

We can see the probability being a good photo of the 10 crops are as follows, the average is 0.97, so the output gives the goodness index as an integer 97.0.

```
0.91845584
```

```
0.9870161
0.91845584
0.9870161
0.917644
0.99285686
0.9953826
0.99285686
0.9953826
0.976757
```

We can see a bad photo as comparison,


localhost:8888/notebooks/IdeaProjects/strato/quality_recognition.ipynb
jupyter quality_recognition Last Checkpoint: 1 小时前 (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 2
Run
Code

```
    return pic
```

In [12]:
```
pic_path = './pic/073.jpg'
pic = resize_img(pic_path)
preds = predict_10_crop(np.array(pic),0)
print(preds)
plt.imshow(pic)
```

```
/anaconda3/envs/datamining27/lib/python2.7/site-packages/ipykernel_launcher.py:7: DeprecationWarning: `imresize` is d
eprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``skimage.transform.resize`` instead.
  import sys
0.8128084
0.15784839
0.8128084
0.15784839
0.5346108
0.078899056
0.70456874
0.078899056
0.70456845
0.44813347
45.0
```

Out[12]:
```
<matplotlib.image.AxesImage at 0xb323eb110>
```

We can see the probability of being a good photo of the 10 crops are as follows, the average is 0.45, so the output gives the goodness index as an integer 45.0. which according to our criteria is a bad photo.

```
0.8128084
0.15784839
0.8128084
0.15784839
0.5346108
0.078899056
0.70456874
0.078899056
0.70456845
0.44813347
45.0
```

7.3 Compare output against hypothesis

The various hypothesis or assumptions made initially are validated against the output generated. The model is tested against two basic test cases as shown below:

TEST CASE NO:	TEST CASE EXPLANATION	EXPECTED OUTPUT	OUTPUT OBTAINED
1	Selected word doesn’t match with the words in database.	Nothing will be displayed.	Nothing is displayed.
2	Selected word maps with the words in the database.	A pop-up displaying the image of the food.	A pop-up displays the image of the food.

When there is no map between the selected word and the words in the database the model doesn’t display any images. There will be no map mainly due to two sub-cases:

a. When the selected word is not a food

b.When the food name is not in the Food-101 dataset.

In both the above cases the designed model will not display any images to the user. The first sub-case may arise if the user selects a word which is not a food at all. For example, the user might select the word “student” which is not a food name. The second sub-case arises if the user selects a word which is not in the Food-101 dataset. For example, some of the indian foods aren’t included in the Food-101 dataset when they are selected it will result in zero mapping and will result in no image.

In case of testcase 2, when there is a map between the selected word and the words in the database either partially or completely, the model will display the image corresponding to that word from the database. The map function will product a complete mapping when the word selected exactly matches with the word in the database, For example, if the selected word is chicken wings which exactly matches with the word in the database. The map function will result in partial mapping when the word selected matches partially. For example, if the selected word is “orange chicken wings”, only chicken wings are mapped and will result in displaying the image of chicken wings and not exact image of “orange chicken wings”.

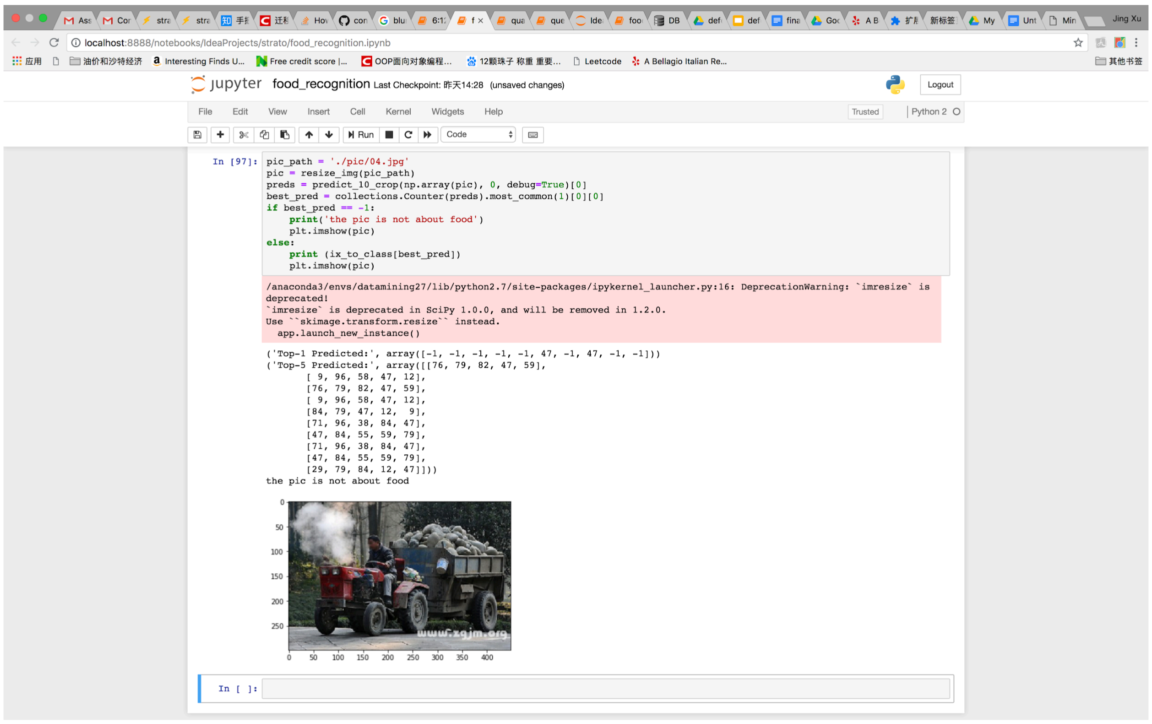
When we compare the last two columns of the table we can conclude that the designed model is able to meet the hypothesis stated before designing the model.

7.4 Abnormal case explanation

We will encounter such cases that the input image is not about a dish. if we don’t handle this exception, the recognition model will give the pic a food classification even if the probability is very low. so we set a threshold value. if the probability is lower than the threshold, we don’t reckon that it is a pic about food and return the classification as “Unknown”.

```
y_pred = model.predict(np.array(crops))
predsprob = np.max(y_pred,axis = 1)
preds = np.argmax(y_pred, axis=1)
for a in range(0,10):
  if predsprob[a] <= 0.3:
    preds[a] = -1
```

The code shows that if the probability of a subimage most likely belong to certain classification is lower than 0.3, we says the subimage belongs to the classification -1. So if the majority of the 10 subimages belongs to -1, we can claim the image is not about food. The demo shows a image about tractor. Even 2 out of 10 subimage claim it is a garlic-bread, 8 out of 10 subimages claims it is not about food. So our error handler returns error message to say it is not about food.


localhost:8888/notebooks/IdeaProjects/strato/food_recognition.ipynb
jupyter food_recognition Last Checkpoint: 昨天14:28 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 2
Run
Code
www.zgjm.org

```
In [97]: pic_path = './pic/04.jpg'
         pic = resize_img(pic_path)
         preds = predict_10_crop(np.array(pic), 0, debug=True)[0]
         best_pred = collections.Counter(preds).most_common(1)[0][0]
         if best_pred == -1:
             print('the pic is not about food')
             plt.imshow(pic)
         else:
             print (ix_to_class[best_pred])
             plt.imshow(pic)
```

```
/anaconda3/envs/datamining27/lib/python2.7/site-packages/ipykernel_launcher.py:16: DeprecationWarning: `imresize` is
deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``skimage.transform.resize`` instead.
  app.launch_new_instance()
```

```
('Top-1 Predicted:', array([-1, -1, -1, -1, -1, 47, -1, 47, -1, -1]))
('Top-5 Predicted:', array([[76, 79, 82, 47, 59],
       [ 9, 96, 58, 47, 12],
       [76, 79, 82, 47, 59],
       [ 9, 96, 58, 47, 12],
       [84, 79, 47, 12,  9],
       [71, 96, 38, 84, 47],
       [47, 84, 55, 59, 79],
       [71, 96, 38, 84, 47],
       [47, 84, 55, 59, 79],
       [29, 79, 84, 12, 47]]))
the pic is not about food
```

```
In [ ]:
```

8. Conclusions and recommendations

8.1 Summary and conclusions

The designed model basically assists the user in knowing how the food will look like by just selecting the food image name. This model basically consists of two parts. The first part involves image recognition and determining the good quality photos,The image recognition model used here is inceptionV3 which has been borrowed by the opensource and the model also includes CNN model for determining good quality photos from bad quality photos to display good photos. the second part is the chrome extension which is basically the front end of the designed model. The user selects the name of the food from this UI created.

8.2 Recommendations for future studies

1. This model can be further developed in many ways such as:
2. One can include ingredient recognition along with image recognition.
3. One can include information about other restaurants where similar kind of food is served.
4. The UI can be made even more beautiful.
5. One can extend the model that recognizes more food categories apart from Food-101.
6. One can extend the model to improve the mapping issues.
7. One can ask help from Yelp system to further develop the model.
8. One can also use map reduce to scrawl pictures then classify and qualify them.

9. Bibliography

1. Chu W T, Lin J H. Food image description based on deep-based joint food category, ingredient, and cooking method recognition[C]//Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on. IEEE, 2017: 109-114.

2. Chu W T, Lin J H. Food image description based on deep-based joint food category, ingredient, and cooking method recognition[C]//Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on. IEEE, 2017: 109-114.

3. Bossard L, Guillaumin M, Van Gool L. Food-101–mining discriminative components with random forests[C]//European Conference on Computer Vision. Springer, Cham, 2014: 446-461.

4. Barranco R C, Rodriguez L M, Urbina R, et al. Enhancing Yelp Data with Deep Learning and Information Reuse[C]//Information Reuse and Integration (IRI), 2017 IEEE International Conference on. IEEE, 2017: 452-461.

5. Chen X, Zhu Y, Zhou H, et al. ChineseFoodNet: A large-scale Image Dataset for Chinese Food Recognition[J]. arXiv preprint arXiv:1705.02743, 2017.

6.https://engineeringblog.yelp.com/2016/11/finding-beautiful-yelp-photos-using-deep-learning.html

7. Hassannejad, Hamid, et al. "Food image recognition using very deep convolutional networks." *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*. ACM, 2016.

8. O'Hara, Stephen, and Bruce A. Draper. "Introduction to the bag of features paradigm for image classification and retrieval." *arXiv preprint arXiv:1101.3354* (2011).

9. Nielsen, Michael A. *Neural networks and deep learning*. Determination Press, 2015.

10. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

11. Bag of Words, Wikipedia

12. Support Vector Machine, Wikipedia

13. Y. LeCun, Y. Bengio, and G.Hinton. Deep learning. Nature, 521(7533):436-444, 2015.

14..https://www.quora.com/What-are-the-differences-between-Keras-and-Lasagne-in-detail

15..https://keras.io/

16..http://lasagne.readthedocs.io/en/latest/modules/layers.html

17.http://blog.stratospark.com/deep-learning-applied-food-classification-deep-learning-keras.html

10.Appendices

10.1 Program source code with documentation

food image recognition model implementation

```
import matplotlib.image as img
import numpy as np
from scipy.misc import imresize
import collections
from keras.applications.inception_v3 import preprocess_input
from keras.models import load_model

model = load_model(filepath='./model4b.10-0.68.hdf5')

def center_crop(x, center_crop_size, **kwargs):
 centerw, centerh = x.shape[0]//2, x.shape[1]//2
 halfw, halfh = center_crop_size[0]//2, center_crop_size[1]//2
 return x[centerw-halfw:centerw+halfw+1,centerh-
halfh:centerh+halfh+1, :]

def predict_10_crop(img, ix, top_n=5, plot=False, preprocess=True,
debug=False):
 flipped_X = np.fliplr(img)
 crops = [
   img[:299,:299, :], # Upper Left
   img[:299, img.shape[1]-299:, :], # Upper Right
   img[img.shape[0]-299:, :299, :], # Lower Left
   img[img.shape[0]-299:, img.shape[1]-299:, :], # Lower Right
   center_crop(img, (299, 299)),

   flipped_X[:299,:299, :],
   flipped_X[:299, flipped_X.shape[1]-299:, :],
   flipped_X[flipped_X.shape[0]-299:, :299, :],
   flipped_X[flipped_X.shape[0]-299:, flipped_X.shape[1]-299:, :],
   center_crop(flipped_X, (299, 299))
 ]
 if preprocess:
   crops = [preprocess_input(x.astype('float32')) for x in crops]

 y_pred = model.predict(np.array(crops))
 predsprob = np.max(y_pred,axis = 1)
 preds = np.argmax(y_pred, axis=1)
 for a in range(0,10):
   if predsprob[a] <= 0.3:
     preds[a] = -1
 top_n_preds= np.argpartition(y_pred, -top_n)[:,-top_n:]
 return preds, top_n_preds

class_to_ix = {}
ix_to_class = {}
with open('./food_classes.txt', 'r') as txt:
 classes = [l.strip() for l in txt.readlines()]
 class_to_ix = dict(zip(classes, range(len(classes))))
 ix_to_class = dict(zip(range(len(classes)), classes))
 class_to_ix = {v: k for k, v in ix_to_class.items()}
sorted_class_to_ix =
collections.OrderedDict(sorted(class_to_ix.items()))

def resize_img(img_path, min_side=299):
 pic = img.imread(img_path)
 w, h, _ = pic.shape
 if w < min_side:
   wpercent = (min_side/float(w))
   hsize = int((float(h)*float(wpercent)))
   pic = imresize(pic, (min_side, hsize))
 elif h < min_side:
   hpercent = (min_side/float(h))
   wsize = int((float(w)*float(hpercent)))
   pic = imresize(pic, (wsize, min_side))

 elif w > min_side:
   wpercent = (min_side/float(w))
   hsize = int((float(h)*float(wpercent)))
   pic = imresize(pic, (min_side, hsize))
 elif h > min_side:
   hpercent = (min_side/float(h))
   wsize = int((float(w)*float(hpercent)))
   pic = imresize(pic, (wsize, min_side))

 return pic

def recognize(pic_path):
 pic = resize_img(pic_path)
 preds = predict_10_crop(np.array(pic), 0)[0]
 best_pred = collections.Counter(preds).most_common(1)[0][0]
 if best_pred == -1:
   return "Unknown"
 else:
    return ix_to_class[best_pred]
```

food image quality recognition implementation

```
import matplotlib.image as img
import numpy as np
from scipy.misc import imresize
import collections
from keras.applications.inception_v3 import preprocess_input
from keras.models import load_model

model = load_model(filepath='./model4b.10-0.68.hdf5')

def center_crop(x, center_crop_size, **kwargs):
 centerw, centerh = x.shape[0]//2, x.shape[1]//2
 halfw, halfh = center_crop_size[0]//2, center_crop_size[1]//2
 return x[centerw-halfw:centerw+halfw+1,centerh-
halfh:centerh+halfh+1, :]

def predict_10_crop(img, ix, top_n=5, plot=False,
preprocess=True, debug=False):
 flipped_X = np.fliplr(img)
 crops = [
   img[:299,:299, :], # Upper Left
   img[:299, img.shape[1]-299:, :], # Upper Right
   img[img.shape[0]-299:, :299, :], # Lower Left
   img[img.shape[0]-299:, img.shape[1]-299:, :], # Lower Right
   center_crop(img, (299, 299)),

   flipped_X[:299,:299, :],
   flipped_X[:299, flipped_X.shape[1]-299:, :],
   flipped_X[flipped_X.shape[0]-299:, :299, :],
   flipped_X[flipped_X.shape[0]-299:, flipped_X.shape[1]-
299:, :],
   center_crop(flipped_X, (299, 299))
 ]
 if preprocess:
   crops = [preprocess_input(x.astype('float32')) for x in
crops]

 y_pred = model.predict(np.array(crops))
 predsprob = np.max(y_pred,axis = 1)
 preds = np.argmax(y_pred, axis=1)
 for a in range(0,10):
   if predsprob[a] <= 0.3:
     preds[a] = -1
 top_n_preds= np.argpartition(y_pred, -top_n)[:,-top_n:]
 return preds, top_n_preds

class_to_ix = {}
ix_to_class = {}
with open('./food_classes.txt', 'r') as txt:
 classes = [l.strip() for l in txt.readlines()]
 class_to_ix = dict(zip(classes, range(len(classes))))
 ix_to_class = dict(zip(range(len(classes)), classes))
 class_to_ix = {v: k for k, v in ix_to_class.items()}
sorted_class_to_ix =
collections.OrderedDict(sorted(class_to_ix.items()))

def resize_img(img_path, min_side=299):
 pic = img.imread(img_path)
 w, h, _ = pic.shape
 if w < min_side:
   wpercent = (min_side/float(w))
   hsize = int((float(h)*float(wpercent)))
   pic = imresize(pic, (min_side, hsize))
 elif h < min_side:
   hpercent = (min_side/float(h))
   wsize = int((float(w)*float(hpercent)))
   pic = imresize(pic, (wsize, min_side))

 elif w > min_side:
   wpercent = (min_side/float(w))
   hsize = int((float(h)*float(wpercent)))
   pic = imresize(pic, (min_side, hsize))
 elif h > min_side:
   hpercent = (min_side/float(h))
   wsize = int((float(w)*float(hpercent)))
   pic = imresize(pic, (wsize, min_side))
 return pic

def recognize(pic_path):
 pic = resize_img(pic_path)
 preds = predict_10_crop(np.array(pic), 0)[0]
 best_pred = collections.Counter(preds).most_common(1)[0][0]
 if best_pred == -1:
   return "Unknown"
 else:
   return ix_to_class[best_pred]
```

food image recognition model training

```
import matplotlib.pyplot as plt
import matplotlib.image as img
import numpy as np
from scipy.misc import imresize

%matplotlib inline

import os
from os import listdir
from os.path import isfile, join
import shutil
import stat
import collections
from collections import defaultdict

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets

import h5py
from keras.utils.np_utils import to_categorical
from keras.applications.inception_v3 import preprocess_input
from keras.models import load_model

from keras.applications.inception_v3 import InceptionV3
from keras.applications.inception_v3 import preprocess_input,
decode_predictions
from keras.preprocessing import image
from keras.layers import Input

import tools.image_gen_extended as T
import multiprocessing as mp

num_processes = 6
pool = mp.Pool(processes=num_processes)

class_to_ix = {}
ix_to_class = {}
with open('good-bad-food/meta/classes-2.txt', 'r') as txt:
    classes = [l.strip() for l in txt.readlines()]
    class_to_ix = dict(zip(classes, range(len(classes))))
    ix_to_class = dict(zip(range(len(classes)), classes))
    class_to_ix = {v: k for k, v in ix_to_class.items()}
sorted_class_to_ix =
collections.OrderedDict(sorted(class_to_ix.items()))

print(class_to_ix)
print(ix_to_class)

if not os.path.isdir('./good-bad-food/test') and not
os.path.isdir('./good-bad-food/train'):

    def copytree(src, dst, symlinks = False, ignore = None):
        if not os.path.exists(dst):
            os.makedirs(dst)
            shutil.copystat(src, dst)
        lst = os.listdir(src)
        if ignore:
            excl = ignore(src, lst)
            lst = [x for x in lst if x not in excl]
        for item in lst:
            s = os.path.join(src, item)
            d = os.path.join(dst, item)
            if symlinks and os.path.islink(s):
                if os.path.lexists(d):
                    os.remove(d)
                os.symlink(os.readlink(s), d)
                try:
                    st = os.lstat(s)
                    mode = stat.S_IMODE(st.st_mode)
                    os.lchmod(d, mode)
                except:
                    pass # lchmod not available
            elif os.path.isdir(s):
                copytree(s, d, symlinks, ignore)
            else:
                shutil.copy2(s, d)
    def generate_dir_file_map(path):
        dir_files = defaultdict(list)
        with open(path, 'r') as txt:
            files = [l.strip() for l in txt.readlines()]
            for f in files:
                dir_name, id = f.split('/')
                dir_files[dir_name].append(id + '.jpg')
        return dir_files

    train_dir_files = generate_dir_file_map('good-bad-
food/meta/train.txt')
    test_dir_files = generate_dir_file_map('good-bad-
food/meta/test.txt')


    def ignore_train(d, filenames):
        print(d)
        subdir = d.split('/')[-1]
        to_ignore = train_dir_files[subdir]
        return to_ignore

    def ignore_test(d, filenames):
        print(d)
        subdir = d.split('/')[-1]
        to_ignore = test_dir_files[subdir]
        return to_ignore

    copytree('good-bad-food/images', 'good-bad-food/test',
ignore=ignore_train)
    copytree('good-bad-food/images', 'good-bad-food/train',
ignore=ignore_test)

else:
    print('Train/Test files already copied into separate folders.')

%%time

# Load dataset images and resize to meet minimum width and height
pixel size
def load_images(root, min_side=299):
    all_imgs = []
    all_classes = []
    resize_count = 0
    invalid_count = 0
    for i, subdir in enumerate(listdir(root)):
        imgs = listdir(join(root, subdir))
        class_ix = class_to_ix[subdir]
        print(i, class_ix, subdir)
        for img_name in imgs:
            img_arr = img.imread(join(root, subdir, img_name))
            img_arr_rs = img_arr
            try:
                w, h, _ = img_arr.shape
                if w <= h and w < min_side:
                    wpercent = (min_side/float(w))
                    hsize = int((float(h)*float(wpercent)))
                    #print('new dims:', min_side, hsize)
                    img_arr_rs = imresize(img_arr, (min_side, hsize))
                    resize_count += 1
                elif h <= w and h < min_side:
                    hpercent = (min_side/float(h))
                    wsize = int((float(w)*float(hpercent)))
                    #print('new dims:', wsize, min_side)
                    img_arr_rs = imresize(img_arr, (wsize, min_side))
                    resize_count += 1
                elif w <= h and w > 400:
                    wpercent = (400/float(w))
                    hsize = int((float(h)*float(wpercent)))
                    #print('new dims:', min_side, hsize)
                    img_arr_rs = imresize(img_arr, (400, hsize))
                    resize_count += 1
                elif h <= w and h > 400:
                    hpercent = (400/float(h))
                    wsize = int((float(w)*float(hpercent)))
                    #print('new dims:', wsize, min_side)
                    img_arr_rs = imresize(img_arr, (wsize, 400))
                    resize_count += 1
                all_imgs.append(img_arr_rs)
                all_classes.append(class_ix)
            except:
                print('Skipping bad image: ', subdir, img_name)
                invalid_count += 1
    print(len(all_imgs), 'images loaded')
    print(resize_count, 'images resized')
    print(invalid_count, 'images skipped')
    return np.array(all_imgs), np.array(all_classes)

X_test, y_test = load_images('good-bad-food/test', min_side=299)

%%time
X_train, y_train = load_images('good-bad-food/train', min_side=299)
```

```python
from keras.utils.np_utils import to_categorical

n_classes = 2
y_train_cat = to_categorical(y_train, nb_classes=n_classes)
y_test_cat = to_categorical(y_test, nb_classes=n_classes)

%%time

# this is the augmentation configuration we will use for training
train_datagen = T.ImageDataGenerator(
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=False,  # set each sample mean to 0
    featurewise_std_normalization=False,  # divide inputs by std of
the dataset
    samplewise_std_normalization=False,  # divide each input by its
std
    zca_whitening=False,  # apply ZCA whitening
    rotation_range=0,  # randomly rotate images in the range
(degrees, 0 to 180)
    width_shift_range=0.2,  # randomly shift images horizontally
(fraction of total width)
    height_shift_range=0.2,  # randomly shift images vertically
(fraction of total height)
    horizontal_flip=True,  # randomly flip images
    vertical_flip=False, # randomly flip images
    zoom_range=[.8, 1],
    channel_shift_range=30,
    fill_mode='reflect')
train_datagen.config['random_crop_size'] = (299, 299)
train_datagen.set_pipeline([T.random_transform, T.random_crop,
T.preprocess_input])
train_generator = train_datagen.flow(X_train, y_train_cat,
batch_size=32, seed=11, pool=pool)

test_datagen = T.ImageDataGenerator()
test_datagen.config['random_crop_size'] = (299, 299)
test_datagen.set_pipeline([T.random_transform, T.random_crop,
T.preprocess_input])
test_generator = test_datagen.flow(X_test, y_test_cat, batch_size=32,
seed=11, pool=pool)

def reverse_preprocess_input(x0):
    x = x0 / 2.0
    x += 0.5
    x *= 255.
    return x

%%time
@interact()
def show_images(unprocess=True):
    for x in test_generator:
        fig, axes = plt.subplots(nrows=8, ncols=4)
        fig.set_size_inches(8, 8)
        page = 0
        page_size = 32
        start_i = page * page_size
        for i, ax in enumerate(axes.flat):
            img = x[0][i+start_i]
            if unprocess:
                im = 
ax.imshow( reverse_preprocess_input(img).astype('uint8') )
            else:
                im = ax.imshow(img)
            ax.set_axis_off()
            ax.title.set_visible(False)
            ax.xaxis.set_ticks([])
            ax.yaxis.set_ticks([])
            for spine in ax.spines.values():
                spine.set_visible(False)

        plt.subplots_adjust(left=0, wspace=0, hspace=0)
        plt.show()
        break
X_test.shape[0]
X_train.shape[0]
%%time

from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D, 
GlobalAveragePooling2D, AveragePooling2D
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, CSVLogger, 
LearningRateScheduler, ReduceLROnPlateau
from keras.optimizers import SGD
from keras.regularizers import l2
import keras.backend as K
import math

K.clear_session()
```

```
base_model = InceptionV3(weights='imagenet', include_top=False,
input_tensor=Input(shape=(299, 299, 3)))
x = base_model.output
x = AveragePooling2D(pool_size=(8, 8))(x)
x = Dropout(.4)(x)
x = Flatten()(x)
predictions = Dense(n_classes, init='glorot_uniform',
W_regularizer=l2(.0005), activation='softmax')(x)

model = Model(input=base_model.input, output=predictions)

opt = SGD(lr=.01, momentum=.9)
model.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])

checkpointer = ModelCheckpoint(filepath='model4.{epoch:02d}-
{val_loss:.2f}.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('model4.log')

def schedule(epoch):
    if epoch < 10:
        return .01
    elif epoch < 28:
        return .002
    else:
        return .0004
lr_scheduler = LearningRateScheduler(schedule)

model.fit_generator(train_generator,
                    validation_data=test_generator,
                    nb_val_samples=X_test.shape[0],
                    samples_per_epoch=X_train.shape[0],
                    nb_epoch=5,
                    verbose=2,
                    callbacks=[lr_scheduler, csv_logger,
checkpointer])
```

chrome extension front-end

```
function getSelected() {
    if (window.getSelection) {
        return window.getSelection();
    } else if (document.getSelection) {
        return document.getSelection();
    } else {
        var selection = document.selection &&
document.selection.createRange();
        if (selection.text) {
            return selection.text;
        }
        return false;
    }
    return false;
}

function textSelectedHandler() {
    var txt;
    if (document.selection) {
        txt = document.selection.createRange().text
    } else {
        txt = window.getSelection() + '';
    }
    if (txt) {
        console.log(txt);
        document.getElementById('')
    }
}

var lastX = 0;
var lastY = 0;

function initContentScript() {
    var feeds = document.getElementsByClassName("review-list");
    if (feeds.length > 0) {
        console.log("load feed success");
    } else {
        console.log("load feed failed");
    }

    $(document).ready(function() {
        $popup = $('<span class="popup-tag hidden"></span>');
        $popup.append('<img id="dish-info" />')
            .append('<span id="dish-name"></span>');
        $mask = $('<span id="mask" class="hidden"></span>');
        $('.review-list').prepend($popup).prepend($mask);

        function setStyle() {
            $popup.removeClass('hidden').animate({
                opacity: 1
            }, function() {});
            $mask.removeClass('hidden').animate({
                opacity: 0.7
            }, function() {});
            H = $(window).height();
            W = $(window).width();
            $popup.css({
                'left': (W - $popup.width()) / 2,
                'top': (H - $popup.height()) / 2,
            });
        }

        function postData(text) {
            phoneNumber = $.trim($('.biz-phone').text());
            url = 'http://127.0.0.1:5000/recognize';
            data = {
                'phone_number': phoneNumber,
                'text': text
            }
            $.post(url, data, function(res) {
                var img_url = res.result[0].image_url ||
res.result[0].default_url;
                var dish_name = res.result[0].dish_name;
                $('#dish-info').attr('src', img_url);
                $('#dish-name').text(dish_name);
            });
        }

        $(document).on('click', function(e) {
            var text = $.trim(getSelected());
            if (text == '' && e.target ==
document.getElementById('mask')) {
                $popup.animate({
                    opacity: 0
                }, function() {
                    $(this).addClass('hidden');
                });
                $mask.animate({
                    opacity: 0
                }, function() {
                    $(this).addClass('hidden');
                });
                // $('#dish-info').attr('src', '');
            }
        });
        $('.review-list').on('mouseup', function(e) {
            var text = $.trim(getSelected());
            if (text != '') {
                setStyle();
                postData(text);
            }
        });
    });

    // $('.review-list').mousedown(function(event) {
    //   lastX = event.clientX;
    //   lastY = event.clientY;
    //   console.log(lastX);
    //   console.log(lastY);
    // });

    $('.review-list').mouseup(function(event) {
        var selection = getSelected();
        selection = $.trim(selection);
        if (selection != '') {
            var offset = $('.review-list').offset();
            console.log(event);
            console.log(JSON.stringify(offset));
            var top = offset.top;
            var left = offset.left;
            console.log(event.clientY);
            console.log(event.clientX);
            $("span.popup-tag").css("display", "block");
            $("span.popup-tag").css("top", event.clientY +
event.offsetY);
            $("span.popup-tag").css("left", event.clientX - left);
            $("span.popup-tag").text(selection);
        } else {
            $("span.popup-tag").css("display", "none");
        }
    });

    //
    // document.addEventListener('mouseup', textSelectedHandler);
    // document.addEventListener('dbclick', textSelectedHandler);
}

initContentScript();
```

text matching/mapping

```
def text_matching(user_text, class_list=classes):
   print(type(user_text))
   user_text = user_text.split(' ')
   for item in class_list:
       class_name = item.split('_')
       for word1 in user_text:
           for word2 in class_name:
               if word1 == word2:
                   return item
   return ''
```

DataBase manager

import dataset

```
DATABASE_URL = 'sqlite:///DataSet/DataBase/Photos.db'


class DbManager(object):

   def __init__(self):
       self.db = dataset.connect(DATABASE_URL)
       self.tbl = self.db.get_table('photos')

   def connect(self, url=DATABASE_URL):
       self.db = dataset.connect(url)
       return self.db

   def get_table(self, table_name):
       table = self.db.get_table(table_name)
       return table

   def update_or_insert(self, key, business_id, url, category,
rate):
       data = dict(pic_id=key, business_id=business_id,
url=url, category=category, rate=rate)
       self.tbl.upsert(data, ['pic_id'])

   def find_with_business(self, business_id):
       results = self.tbl.find(business_id=business_id)
       return list(results)
```

```
    def find_with_category(self, business_id, category):
        results = self.tbl.find(business_id=business_id,
category=category, order_by='-rate')
        return list(results)

    def find_one(self, key):
        result = self.tbl.find_one(pic_id=key)
        return result

    def insert(self, key, business_id, url, category, rate):
        data = dict(pic_id=key, business_id=business_id,
url=url, category=category, rate=rate)
        self.tbl.insert(data, ['pic_id'])
```

Web Crawler

```
import haul
import requests
import os
import re
import imageRecognition
from dbmanager import DbManager
import shutil
import qualityRecognition

root = 'DataSet/Images'

root_url = 'https://www.yelp.com/biz_photos/a-bellagio-italian-
restaurant-campbell-2?'
tab = '&tab=food'
business_id = 'O0R8TkEE2eWDnp9xOOjHBQ'

cursor = 30
total_page = 1

db_manager = DbManager()


def load_image():
    for i in range(0, total_page):
        start = cursor * i
       page_url = root_url + 'start=' + str(start) + tab
       result = haul.find_images(page_url)
       image_paths = []
       for i, url in enumerate(result.image_urls):
           if not os.path.exists(root):
               os.mkdir(root)
           if not re.match(r'^https?:/{2}\w.+$', url):
               continue
           components = url.split("/")
           last_component = components[-1].split('.')
           pic_id = components[-2]
           extension = last_component[-1]
           if extension != 'jpg':
               continue
           r = requests.get(url)
           r.raise_for_status()
           parent_path = root + '/' + pic_id
           if not os.path.exists(parent_path):
               os.mkdir(parent_path)
           path = root + '/' + pic_id + '/' + components[-1]
           image_paths.append(path)
           with open(path, "wb") as f:
               f.write(r.content)
           food_class = imageRecognition.recognize(path)
           rate = qualityRecognition.recognize(path)
           # print('image_url :' + url)
           # print('food class:  ' + food_class)
           # print('rate: ' + str(rate))
           db_manager.update_or_insert(pic_id, business_id,
url, food_class, rate)


def delete_cache():
   file_list = os.listdir(root)
   for f in file_list:
       file_path = os.path.join(root, f)
       if os.path.isfile(file_path):
           os.remove(file_path)
           print file_path + " removed!"
       elif os.path.isdir(file_path):
           shutil.rmtree(file_path, True)
           print "dir " + file_path + " removed!"
```

```
load_image()
delete_cache()
```

Server

```
from flask import Flask, request, jsonify
from dbmanager import DbManager
from yelp import Yelp
import re

app = Flask(__name__)
default_url = 'https://www.yelpblog.com/wp-
content/themes/yelpblog/images/yelp-avatar.png';

empty_result = [
   {
       'status': 200,
       'id': '',
       'image_url': '',
       'dish_name': '',
       'default_url': default_url
   },
]

yelp = Yelp()
db_manager = DbManager()
classes = []

with open('./food_classes.txt', 'r') as txt:
   classes = [l.strip() for l in txt.readlines()]


@app.route('/')
def hello_world():
   return jsonify('hello, how are u')


@app.route('/recognize',  methods=['POST'])
def response():
   phone_number = request.form.get('phone_number', '')
```

```
    food_name = request.form.get('text', '')
    phone = convert(phone_number)
    business = yelp.search_business(phone)
    business_id = business['id']
    category = find_category(food_name)
    if category == '':
        return jsonify({'result': empty_result})
    photos = db_manager.find_with_category(business_id,
category)
    if len(photos) == 0:
        return jsonify({'result': empty_result})
    return json_response(photos[0], food_name)


def convert(phone_number):
    res = re.sub('[ ()-]', '', phone_number)
    return '+1' + res


def json_response(photo, dish_name):
    result = [
        {
            'status': 200,
            'id': photo['pic_id'],
            'image_url': photo['url'],
            'dish_name': dish_name,
            'default_url': default_url
        },

    ]
    return jsonify({'result': result})


def find_category(food_name):
    name_str = food_name.lower()
    return text_matching(name_str)


def text_matching(user_text, class_list=classes):
    print(type(user_text))
    user_text = user_text.split(' ')
    for item in class_list:
        class_name = item.split('_')
        for word1 in user_text:
            for word2 in class_name:
                if word1 == word2:
                    return item
    return ''


if __name__ == '__main__':
    app.run()
```

Yelp

```
import requests
import json
from urllib.parse import quote

API_KEY =
"uPwnMrsg1PknTUUG8j3e0wjGdb2Cec0Zh1Y8Hc6mSEDSZVvDTc295dbfS9uOa9
oHyFUia0qTHJ8stBKHvKg93jtumdEkKjExJxfPykhb1G3q8jf2k0xFDFm2AF8WW
3Yx"
API_HOST = 'https://api.yelp.com'
SEARCH_PATH = '/v3/businesses/search/phone'


class Yelp(object):
    @staticmethod
    def request(host, path, api_key, url_params=None):

        url_params = url_params or {}
        url = '{0}{1}'.format(host, quote(path.encode('utf8')))
        headers = {
            'Authorization': 'Bearer %s' % api_key,
        }

        print(u'Querying {0} ...'.format(url))
        response = requests.request('GET', url, headers=headers,
params=url_params)
        return response

    def search_business(self, phone_number):
        url_parameters = {
            'phone': phone_number
        }

        response = self.request(API_HOST, SEARCH_PATH, API_KEY, 
url_parameters)
        if not response.ok:
            return ""
        content = json.loads(response.text)
        business = content['businesses'][0]
        return business
```

10.2 Input/output listing

food_classes.txt

```
apple_pie         eggs_benedict         onion_rings
baby_back_ribs          escargots             oysters
baklava                 falafel               pad_thai
beef_carpaccio          filet_mignon          paella
beef_tartare      fish_and_chips        pancakes
beet_salad        foie_gras             panna_cotta
beignets          french_fries          peking_duck
bibimbap          french_onion_soup           pho
bread_pudding     french_toast          pizza
breakfast_burrito  fried_calamari             pork_chop
bruschetta        fried_rice                  poutine
caesar_salad      frozen_yogurt         prime_rib
cannoli                 garlic_bread          pulled_pork_sandwich
caprese_salad     gnocchi               ramen
carrot_cake       greek_salad                 ravioli
ceviche                 grilled_cheese_sandwich         red_velvet_cake
cheesecake        grilled_salmon        risotto
cheese_plate      guacamole             samosa
chicken_curry     gyoza                 sashimi
chicken_quesadilla  hamburger                 scallops
chicken_wings     hot_and_sour_soup           seaweed_salad
chocolate_cake          hot_dog               shrimp_and_grits
chocolate_mousse   huevos_rancheros           spaghetti_bolognese
churros                 hummus                spaghetti_carbonara
clam_chowder      ice_cream             spring_rolls
club_sandwich     lasagna               steak
crab_cakes        lobster_bisque        strawberry_shortcake
creme_brulee      lobster_roll_sandwich                              sushi
croque_madame     macaroni_and_cheese                                tacos
cup_cakes         macarons              takoyaki
deviled_eggs      miso_soup             tiramisu
donuts            mussels               tuna_tartare
dumplings         nachos                waffles
edamame                 omelette
```

model4b.10-0.68.hdf5

Model of food image recognition

model68-2.7.hdf5
Model of food image quality recognition