SI 650 Information Retrieval Final Report

Peiyan Wang
School of Information
peiyanw@umich.edu

Zhengyang Zhao
School of Information
zzyang@umich.edu

1. Introduction

When people travel or move to a new country, there may be many vegetables and fruits they have never seen before. To find out how to make a dinner with the unseen vegetables, they may need to take a picture of the ingredients, upload it to google image, get the name of the vegetable from the retrieved results, and finally search for the recipe. To simplify these steps, we build a search engine to help people to decide what kind of meal they would like to have based on the ingredients by only taking a picture. To realize this, we build a CNN classifier to recognize the vegetable in the input image and modify the returned label as a search query to generate recipes for users. The ranking method we have approached is BM25 and apply NDCG@10 as the evaluation metric to our search engine.

2. Data

2.1 Data source

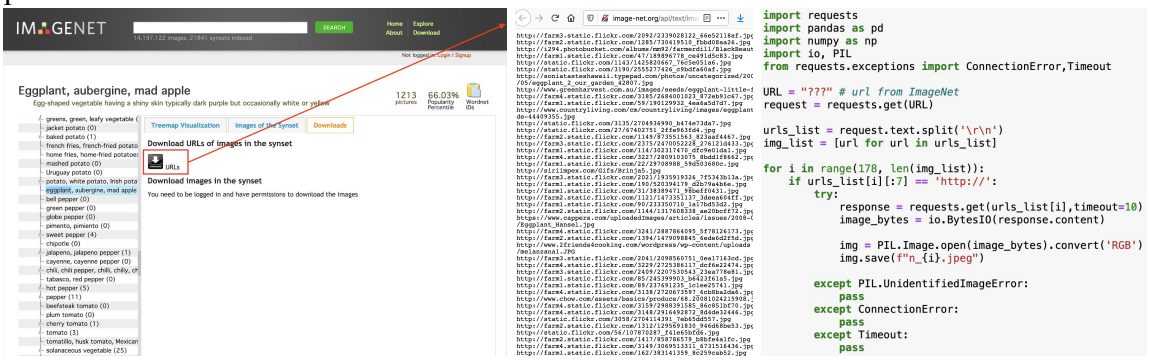
Vegetable image source
ImageNet (http://www.image-net.org/synset?wnid=n07735510#)
This is a public image dataset platform that provides image urls and thumbnails to researchers.

Recipe text data source
https://github.com/stevenjson/CuisineClassifying/tree/master/Data
This is a dataset published in Github and is originally designed for cuisine classification. We only use part of this dataset that includes all recipe text files.

2.2 Image Data obtaining

We first manually copied all contents in recipes text files from Github and then manipulated the text file from .txt to .csv with two columns - 'recipe_text' and 'recipe_id'. From this combined file, we selected overall 21 types of vegetables as part of our queries for the search engine, which includes lettuce, green onions, cabbage, pepper, broccoli, spinach, green peas, asparagus, cucumber, bok choy, celery, ginger, garlic, zucchini, tomato, mushroom, eggplant, oat, carrot, squash, parsley. Since these folders were compressed, we built a model to uncompress folders and renamed these folders with their vegetable names.

ImageNet is a huge image resource dataset that contains hundreds of categories of objects from vegetables to animals. By registering an account for ImageNet, we are able to download tar files for each kind of object. However, we noticed that not all vegetables that we manually selected from recipes text are available to download. Instead, we used io, requests, pillow packages to obtain an image and save images as jpeg in their corresponding vegetable folder. During the process of request, we also noticed that not all of the links are direct to the correct pages; some of them have false responses, and some of them don't have contents on the page. Therefore, we skipped these urls that cause errors and saved the rest of them in folders.



2.3 Image Data Preprocessing

Since image data all have different sizes, we use the cv2 package to read in and resize the image as 64X64X3 (Height, Width, RGB channels) to generate new images with the same size, then store them in a NumPy array, which is assigned to X as our features. We also normalize the values in X by dividing 255 to speed up training with more stable gradients. Since CNN cannot take in categorical variables, we transform the label Y, which is vegetable names, from text to a NumPy array using the OneHotEncoder package.

2.4 Recipe Data Description

There are duplicates in recipes therefore we drop these and reset indexes. The final recipes file with CSV format has a shape of 1410 rows (or recipes) and 2 columns ('recipe_text' and 'recipe_id'). Below are the first 5 samples of recipes.csv.

	recipe_text	recipe_id
0	2 onions , chopped 2 cloves garlic , minced 1 ...	1
1	1 tablespoon olive oil 1 (3 pound) roasting ch...	2
2	1 sweet potato , peeled and cubed 1 medium egg...	3
3	1 onion , sliced 2 cloves garlic , minced (opt...	4
4	2 tablespoons olive oil 2 medium onions , chop...	5

The numbers of images in each vegetable folder are listed below. We noticed that, on average, the number of images for each category is around 1000. While there are some outliers - ginger, garlic, and oat. These vegetables are having less than 200 images on average, which might result in low accuracy in CNN classifiers.

	vege_name	image_counts
0	lettuce	1361
1	green onions	1165
2	cabbage	934
3	pepper	1378
4	broccoli	1224
5	spinach	1108
6	green peas	1296
7	asparagus	1661
8	cucumber	1268
9	bok choy	1026

	vege_name	image_counts
10	celery	1020
11	ginger	66
12	garlic	163
13	zucchini	1093
14	tomato	1405
15	mushroom	1100
16	eggplant	495
17	oat	104
18	carrot	762
19	squash	758
20	parsley	490

3. Related work

There are two main problems in our project - image classification and text ranking method. For image classification, we found one article using the same dataset as their image data resource in the research. The research used deep convolutional neural networks to classify images in the ImageNet. The main purpose of this research is to find a model with better performance dealing with larger datasets and the problem of overfitting.[1] The difference between our project and the paper is the size of our dataset. Since the huge scale of the dataset, they trained image data on multiple GPUs, while we only took 21 categories of images instead of 1000. As a result, there is no need for us to take memories into consideration. What we choose to do is to focus on the CNN classification, and how they apply it to their research.

We also noticed that many other methods are built based on CNN. For example, VGG16 [2] is to apply an architecture with very small convolution filters to increase depths that eventually lead to higher accuracy. Also, there is another model, a compressed convolutional neural network model, also known as Residual Squeeze VGG16 [3], which is built based on VGG16. With this method, the size of the proposed model is much smaller than VGG16 and has a faster training time, with same-good performance meanwhile.

Besides deep learning, image classification can also be done using Random Forests and Ferns.[4] This is a multi-method classifier with a focus on the region of interest (ROI) and shape or edge distribution. Compared with multi-method SVM, the Random Forests and Ferns multi-method is easier to do training and testing. The results showed that the performance on ROI increased by about 5% and 10% improvement over the Caltech-256, an improvement to the Caltech 101 dataset, the Pascal Visual Object Classes datasets.

For the text retrieval method, we found that a Linear Least Squares Fit (LLSF) is introduced to estimate the similarity between terms of different vocabularies.[5] The relevance judgments of LLSF mapping is based on human assessments in the training set. In another word, researchers need to judge the relevance of query and documents manually - similarly, for our project would be vegetable names and recipes. While we doubted the implementability of this method with limited time for a course project.

We also found a paper with a similar topic, based on what we want to do in our project. The distinction in the paper is that they use food images, the final outcome to be precise, as their input data, and generate recipes from these images.[6] While we only take in images with a single type of food ingredients. The study also involves user studies, taking qualitative results into consideration, and re-train their best predicted model. Based on the sample results presented in the paper, the F1 score is below 0.5, which is much lower than we thought it would be.

4. Methods

4.1 Build Vegetable Recognition Deep Learning Models

To optimize and evaluate our CNN model, we split the whole dataset into the train(72%), validation(18%), and test(10%) datasets. Accuracy is used as the evaluation matrix to evaluate the performance of the classification model. We applied image data augmentation using ImageDataGenerator to generate transformed images, such as rotation and horizontal flipping, to prevent overfitting.

We tried one dummy classifier predicting the most frequent class of vegetables and four CNN classifiers to compare their performance. The first CNN model is built from scratch. It contains 4 convolutional layers followed by max-pooling layers and two fully connected layers. The other three CNN models are transferred based on VGG16 which is a convolutional neural network model that achieves top5 test accuracy in ImageNet. It has 5 convolutional blocks and 3 fully connected layers as shown in figure 4.1.1. For the three VGG16 transferred classifier, we import the 'imagenet' weight and (1) freeze the weight of the 5 convolutional blocks and add our own fully connected layers (2) or unfreeze the weight of the last one conv block, or unfreeze the weight of the last two conv blocks.

During the training process, we store the models as h5 files and save the weights of the best model with the least validation loss as hdf5 files. We use the models with the saved weights on the test dataset to compare the performance of the 5 models. To classify the new input images, we choose the best model with saved weights.

We use libraries including NumPy, sklearn, and Keras to build the image classifiers. Matplotlib and seaborn libraries are used to visualize the results.

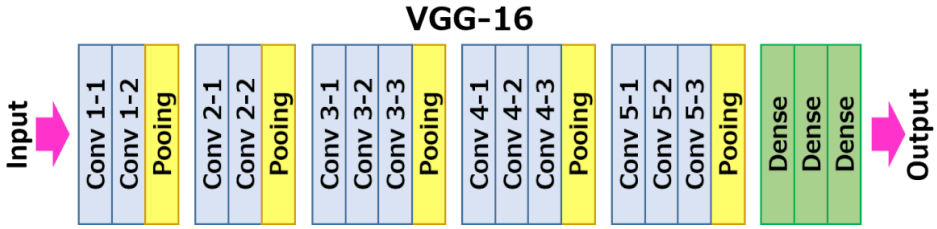


Figure 4.1.1 The structure of the VGG16 classifier.

4.2 Ground truth annotation

In order to make the information retrieval system more complex, we modify the vegetable recognition result returned from the CNN model as queries instead of using the ingredient directly. Queries are like *How to make broccoli salad?,* and *What part of the green onion can be*

used? We annotated 23 queries as our ground truth data to measure the performance of the information retrieval system. The relevance is marked as 0, 1, or 2 as shown in figure 4.2.1. 0 means the recipe is not relevant to the query at all. 1 means the recipe could partially answer the query while 2 means the recipe could mostly match the query. The ground truth annotations are stored in a CSV file with query id, recipe id, and the relevance rating as shown below.

	query_id	recipe_id	rating
0	1	274	2
1	1	294	2
2	1	295	2

Figure 4.2.1 The first 5 rows of the ground truth annotation table.

4.3 Build an Information retrieval system

We build inverted indexes of 1410 recipes to support users' queries via the metapy package. We choose BM25 as our ranking model. We tried two packages, metapy and rank_bm25, to determine the rankings. Stop words are moved because they are not distinguishable for information retrieval. For the metapy package, we use unigram to extract features from the recipes. The filter is default-unigram-chain. NDCG@10 is used to evaluate the performance of our information retrieval system.

4.4 Implement Spell Check in input query

We used *pyspellchecher* package in our queries. By typing in a sentence, the function we built below will take in the query and check whether there are misspelled words, and provide the most likely correct words. Then we replaced misspelled words with the correct ones and presented the corrected sentence back to the user. A sample interactive interface is presented as follows.

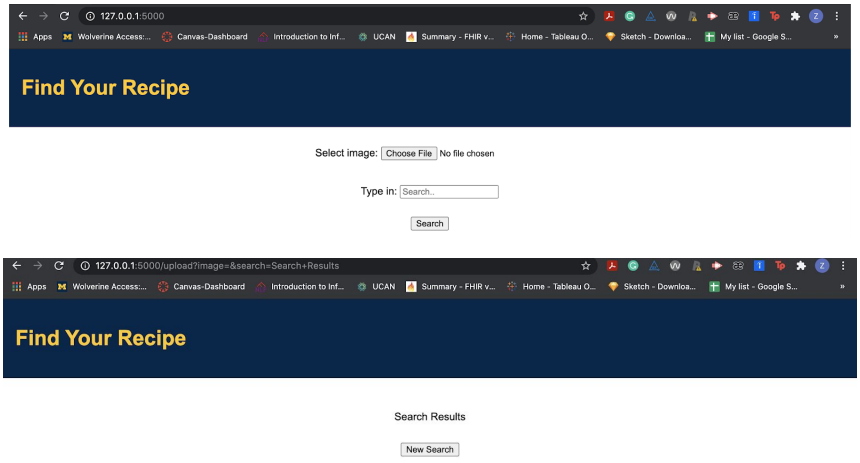
```
def check(query):
    spell = SpellChecker()
    if query:
        words = query.split(' ')
    else:
        words = []
    my_list = words.copy()
    misspelled = spell.unknown(words)

    for word in misspelled:
        corrected = spell.correction(word)
        my_list[my_list.index(word)] = corrected

    return ' '.join(my_list)
```

```
Type in your query...
thiss is a testt
this is a test

Type in your query...
this is annoother testt
this is another test

Type in your query...
hello  thiss is my houuse
hello a this is my house

Type in your query...
todday is a beautifull daay
today is a beautiful day
```

4.5 Build a user interface

In order to build a user interface, we write a simple user interface script allowing users to upload images to the input folder and get the returned vegetable name. Then users could input their queries and the script would return top 10 relevant recipes.

To build a more user-friendly interface, we decided to use Flask to implement our classification and retrieve engine. The expectation would be that users are able to upload an image file and type in a query based on the classification result. After clicking the *Search* button, a result page would come out, presenting 3 recipes for users to choose from.



5. Evaluation and Results

5.1 VGG16 model tuning last two conv blocks has the best performance

We tried one dummy classifier and 4 CNN classifiers to recognize the images. We found that the VGG16 model tuning the last two conv blocks has the best performance with accuracy as high as 0.56, which increased 86% from the performance of the dummy classifier. The performance of the 5 models are shown in figure 5.1.1. The accuracy score and loss function of the best model changed over epochs as shown in figure 5.1.2. The best model began overfitting after 13 epochs.

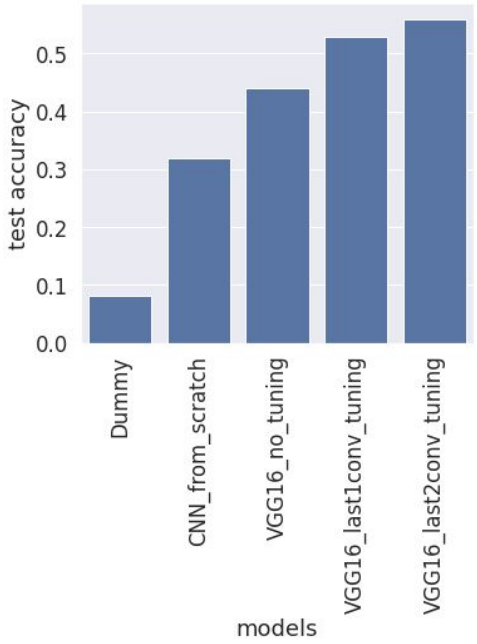


Figure 5.1.1 The performance of the 5 image classification models

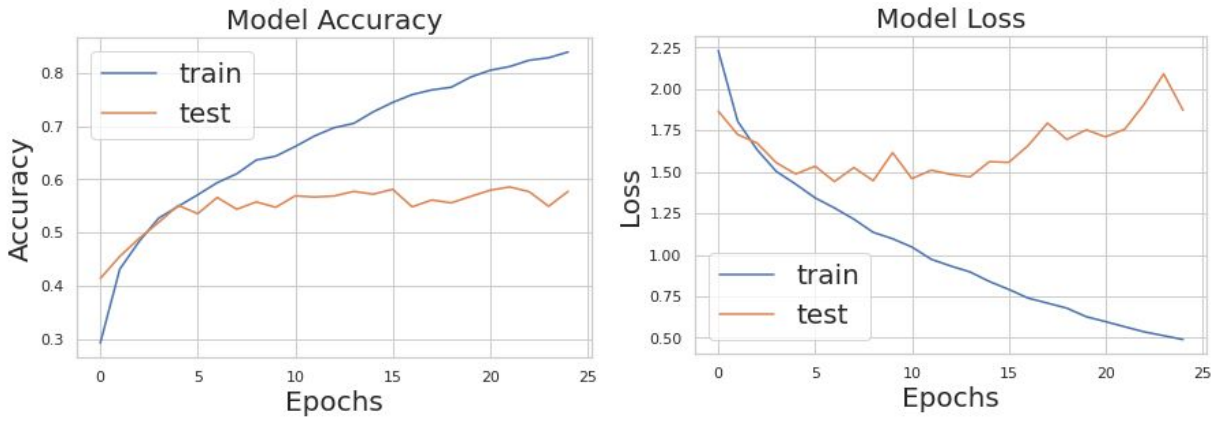


Figure 5.1.2 The accuracy (left) and loss (right) of the best VGG16 tuning model.

5.2 The basic BM25 model has a good performance to retrieve recipes

After the users get the classification result of the vegetable image, they can type in the relevant query. BM25 ranking model will return the top 10 results. We compared the performance of metapy and rank_bm25 package and found that the performance of metapy is much better than that of rank_bm25. The average NDCG@10 score using metapy is 0.94 while the score is only 0.85 using rank_bm25. The NDCG@10 scores of the first 16 queries via metapy are shown in figure 5.2.1, which looks good enough. Therefore, we did not try other ranking algorithms.

	query	query_id	NDCG@10
0	How can I cook Lettuce?	1	1.000000
1	What part of the green onion can be use?	2	1.000000
2	How to make stir-fried pork and cabbage?	3	1.000000
3	pork and pepper stir fry	4	0.448261
4	How to make broccoli salad	5	0.935648
5	How to cook spinach and meat	6	0.751444
6	Fried noodles with green peas	7	1.000000
7	How to make oven-Roasted Asparagus	8	1.000000
8	How to make cucumber salad	9	0.751444
9	stir fry meat and Bok Choy	10	1.000000
10	smached potato	11	1.000000
11	minced tomato	12	1.000000
12	minced garlic	13	1.000000
13	minced ginger	14	1.000000
14	ginger powder	15	1.000000
15	garlic powder	16	1.000000

Figure 5.2.1 The NDCG@10 score of the annotated queries using the metapy package

6. Discussion

Our best image classifier has an accuracy score of only 0.56. Although the performance increased by 86% compared to the dummy classifier, it may not satisfy the users in real practice. Some possible reasons would lead to such a low accuracy score. (1) The dataset is imbalanced. 14 out of 20 kinds of vegetables have more than 900 images while 3 kinds of them have images less than 100. (2) The quality of the images is not good enough. After observing the misclassified images, we found that the vegetables only occupy a small proportion of the image in some cases. A large proportion of the image is people or other stuff instead of vegetables. (3) The model begins to overfit over 10 epochs although we applied data augmentation. As we see from figure 5.1.2, the loss of the training dataset keeps decreasing while the loss on the validation dataset begins to go up after 13 epochs, which means the model is overfitting on the training dataset. Solutions to this problem include adding more image data, utilizing regularization such as dropout layers.

The performance of the BM25 model via metapy package looks good, which is 0.94 on average, for the annotated queries. Reasons may include (1) The queries are relatively simple although we have modified them. Because the keyword in the query is the vegetable, the ranking model could easily find out the recipes containing that word. But we also notice that the BM25 ranking model has bad performance on some queries, such as *pork and pepper stir fry, how to cook spinach and meat*. That is because the recipes containing pork and pepper are making Wonton with ground pork and pepper, which are not the correct results. For the latter one, the recipes containing meat are using *pork*, *beef*, or *shrimp* but not contain *meat* directly. So the BM25 ranking model may have bad performance on queries with extra meanings than words.

7. Conclusion

In our project, we build a search engine with images as input and output recipes. Users can upload an image of the vegetable and get the classification result. Then they can modify the result as queries and the relevant recipes are returned by our search engine. We tried 4 CNN classifiers to recognize the vegetables and found that the transfer learning algorithm, VGG16 with the last two conv blocks tuning, had the best performance. BM25 ranking model was used in our information retrieval system and had a good performance. In the future, more image data would be collected to optimize the classification model. More complex queries and other ranking models would be added to our information retrieval system.

8. Other Things We Tried

(1) Image dataset from TensorFlow

When we seek the image dataset, we found that TensorFlow has an imagenet_resized dataset with different image sizes. However, there are limited kinds of vegetables in that dataset. Therefore, we downloaded the original image dataset from Imagenet.

(2) Not data augmentation

In the project progress report, we tried CNN models with no data augmentation. The VGG16 model with no tuning classifier began overfitting after 5 epochs while the best accuracy score on the test dataset is less than 0.34. In contrast, the best accuracy score of the same model could be as high as 0.44 when data augmentation was applied.

(3) Query with only the name of the vegetables

In the project progress report, we use the name of the vegetable as queries directly. The average NDCG@10 is 0.966, which means the information retrieval system is relatively simple.

9. Next Step

If we have more time, we would devote more time to the following aspects.

(1) More images and balanced dataset

Since the classifiers have an overfitting problem, we would add more images, especially for those vegetables with less than 100 images.

(2) More complex queries

We would try more complex queries to test our information retrieval system, such as adding the country to search for the recipes.

(3) More recipes

We chose the recipe text from a Github repo. However, the recipes are sometimes hard to understand because they do not have titles and need more sentence breaks. We should use other sources for our recipes, like allrecipes.com. Because it has more structured texts and easier to read. A web crawler would be used to extract the texts.

(4) Try other ranking models

When we try the queries with no words in common with the recipes, the BM25 ranking model may have a bad performance. We will need to try other ranking models such as a language model or PLSI.

(5) Build a more user-friendly interface

We first tried a simple user interaction interface through the command line. We want to fill the model and methods in the flask framework added with some modern CSS styles and designs so that users are able to interact with the retrieval engine like using a real web application. The difficulty now we have is not being able to get the image file through request in the flask route.

10. Reference

[1] Krizhevsky, A., Ilya Sutskever and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks.” CACM (2017).

[2] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition.” Int. Conf. on Learning Representations, San Diego, CA, 2015.

[3] Qassim, Hussam, Abhishek Verma and David Feinzimer. “Compressed residual-VGG16 CNN model for big data places image recognition.” *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (2018): 169-175.

[4] Bosch, Anna, Andrew Zisserman and X. Muñoz. “Image Classification using Random Forests and Ferns.” *2007 IEEE 11th International Conference on Computer Vision* (2007): 1-8.
[5] Yiming Yang and Christopher G. Chute. 1994. An example-based mapping method for text categorization and retrieval. *ACM Trans. Inf. Syst.* 12, 3 (July 1994), 252–277. DOI:https://doi.org/10.1145/183422.183424
[6] Salvador, A., M. Drozdzal, Xavier Giró-i-Nieto and A. Romero. “Inverse Cooking: Recipe Generation From Food Images.” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019): 10445-10454.