Yelp Restaurant Photo Classification

Yixin Cai

Abstract

This project is about attaching labels to restaurants based on the photos uploaded by users. In this paper we have explored different methods to extract image features, extract restaurant features and classify restaurants. For the training set, the F1 score considering all labels is 0.7306. The mean F1 score is 0.69719. The mean F1 score on hidden test set is 0.69059.

1. Introduction

Nowadays with the extensive use of smart phones equipped with high resolution cameras, more people are taking photos to record their daily life. Consequently many mobile apps are developed for people to post these photos and share their experiences. With the massive amount of user photos and comments, companies began to look for machine learning tools that can analyze the data on a large scale within a reasonable time. These tools should also have good performance so that the companies can identify user demand and provide better service.

As a result, I decide to do this project of Yelp restaurant photo classification. In fact, this project is inspired by a Kaggle competition posted by Yelp (https://www.kaggle.com/c/yelp-restaurant-photo-classification) half a year ago. The data provided include a set of user-uploaded images of restaurants, and the goal is to identify the attributes of these restaurants. The competition has concluded on 4/12, so I was not able to participate in it. But I was still able to download data of training and test photos, and use them to explore different feature extraction and image classification methods.

2. Related works

2.1 Review of previous work

Since the competition has concluded, some top 10 teams have posted their solution. Here are the links to solutions posted by the first place (http://blog.kaggle.com/2016/04/28/yelp-restaurant-photo-classification-winners-interview-1st-place-dmitrii-tsybulevskii/), the fourth place and the seventh place (https://www.kaggle.com/c/yelp-restaurant-photo-classification/forums/t/20340/solution-sharing/119373#post119373). The winner used different CNN models (Inception-V3, Inception-BN and ResNet) to extract image features. He used feature pooling, fisher vectors and VLAD descriptor to extract business features. Finally he uses gradient boosting, logistic regression and neural network for classification.

In addition, there are many academic papers describing different methods for classifying images. Oren Boiman, Eli Shechtman and Michal Irani talks about an improved method for using nearest neighbor to classify images[1]. Yunpeng Li, David Crandall and Daniel Huttenlocher has also introduced a SVM method to classify images of landmarks[2]. There are also other classification methods like random forest and ferns which is elaborated by Anna Bosch, Andrew Zisserman and Xavier Munoz[3].

2.2 Difference between my approach and winners' approach

The major difference is that I used HOG (trained on all data) and SIFT (trained on partial data because of time constraint) features, while the top 10 teams all used CNN features. This is because I do not have enough time to setup and train CNN features. When I started the project, no solution was released and my plan is to use HOG and SIFT features, which was supposed to be covered in lectures. It was not until late week 7 that I saw winners' solutions and realized that CNN works best. At that time I have not finished training all data with HOG feature, I decided to prioritize HOG feature extraction and the implementation of classification methods. Since I had zero prior knowledge about CNN, my plan is to explore CNN features in the last week. Unfortunately I had some issue with installing and configuring CaffeNet, and consequently I do not have enough time to use CNN features in my project.

3. Data

3.1 Train and test images

The data set is released by Yelp. The link to data is https://www.kaggle.com/c/yelp-restaurant-photo-classification/data. In the training set, there are 234,842 photos corresponding to 2,000 restaurants, where 4 are thrown away because they do not have corresponding labels. For test set, there are 237,152 photos and 10,000 businesses. I originally thought there were 1,048,575 test photos because the file provided has 1,048,575 photo-restaurant pairs. The reason there are more photo-restaurant pairs than actual photos is because the test data assumes that a user could mistakenly submit multiple photos for the same restaurant, and there are multiple restaurants using the same photo because they belong to the same chain business.

Since the attributes of the test businesses are not released, I trained and tested using only the training set. The training data is split into 80% for training and 20% for test. The test set is used once in the end and the result is submitted to Kaggle.

3.2 Labels

There are 9 Boolean attributes associated to businesses, and they are

good_for_lunch	*good_for_dinner*	*takes_reservations*
outdoor_seating	*restaurant_is_expensive*	*has_alcohol*

has_table_service *ambience_is_classy* *good_for_kids*

Since a restaurant can have multiple labels and all the labels consider different aspects, I decided to classify each label separately.

4. Technical Approach

4.1 Three step solution

There are three steps to solve this problem. The first step is to extract features from each image. Some images from the dataset are shown in Figure 1. We can see that the images are mainly food images, but there are also images about the restaurant. So the extracted image feature should be able to deal with different kinds of objects in the scene, including food, drinks, menu, human, tables, building and other information.

Figure 1. Sample images in training set

The second step is to get the feature for restaurant based on the image features. Here we want to find a way to combine all image features without either losing too much precision or introducing too much noise.

The third step is to classify the restaurants based the business features. The result of the classification algorithm should be stable and should introduce a result that is well above random guess.

4.2 Failure of two-step solution

In the beginning of this project I tried to use a two-step solution, which is to skip the second step of extracting business-level features. The two step solution directly assign business labels to images and classify businesses based on the average predict labels of each image. At first I thought this should produce a similar result, but the result after using 3,000 photos (80%

train and 20% test) has overall F1 score of only 0.58055. The reason for the low F1 score is that many restaurant labels are not related to the image. For labels like “takes_reservations”, the example images are usually about food which does not help classification. So I decided to avoid using this two-step approach.

4.3 Image level feature extraction

I have tried to extract both HOG features and SIFT BoW features from the images. I was not able to test CNN feature because of the time limit.

4.3.1 HOG features

To extract HOG features, the images are converted to grayscale and resized to 256*256. Then I used HOG function from Python scikit-image library, which is roughly the same as our implementation in Problem Set 3. The parameters are 8 for orientations, 16*16 for sliding window, and 1*1 for cells per block. As a result, the feature vector for each image is 16*16*8 = 2048.

I was running feature extraction locally, and the total time used to extract features for the training set is 7.5 hours. The total time for test set is 8.5 hours.

4.3.2 SIFT features

I started with SIFT Bag of Words feature extraction that I implemented in Problem Set 4. The parameters I used are pyramid depth of 3, and 20 SIFT words chosen randomly.

Unfortunately, with this implementation I was not able to finish extracting features for all data. The extraction rate of is about 2000 photos per hour locally on my machine, and it would need more than 10 days to extract all train and test features.

As a result, I only trained on a small subset of images with 3,000 photos (1117 businesses). The F1 score is 0.63772, which is higher than the result of HOG features (0.58055). So this indicates that if I could fully train the given data with SIFT BOW features, it is definitely possible that I can improve the overall score.

4.4 Business-level feature extraction

As mentioned before, the corresponding photos for each business tend to be noisy. In order to capture all features and reduce noise, I decided to use average pooling for features. For each restaurant, I take all corresponding image-level features and average them to get business level features. There are definitely other ways to handle the features, and I will discuss it in the final section.

4.5 Classification algorithms

In the beginning of this project I planned to use a linear classifier for the baseline model, but I decided to skip it and start with SVM because SVM generally works better than a linear model in classification problem, and there is already good SVM implementation available in Python.

4.5.1 SVM

I used SVM with a linear kernel and RBF kernel. I have also tuned the C value to improve performance.

4.5.2 K nearest neighbors

I decided to try K nearest neighbors mainly because of the paper of Oren Boiman, Eli Shechtman and Michal Irani, which support the use of KNN to classify images. Since there are only 2,000 restaurants in training set, the prediction time is very short, and I was able to tune the parameters within a short time. I have tuned different K values for different labels in order to get a good result.

4.5.3 Neural network

Finally a simple neural network was implemented for classification. This network has one hidden layer, which is fully connected with the input layer of business vectors. The activation function is hyperbolic tangent. I tuned the size of the hidden layer from 100 to 1000. Learning rate and regularization factor are also tuned.

I was expecting that the neural network can improve the result, but the reality the result of neural network is nowhere close to SVM and KNN. The discussion will be in the next section.

4.6 Evaluation metrics

For evaluating classification results, F1 is a reasonably good metric because we want to minimize both labels assigned to wrong restaurants and restaurants with missing labels.

4.7 Code

All code I have written and used is in Dropbox. The link is https://www.dropbox.com/sh/1b0809kv8d4d6kz/AADGVSDEdc1AIGOl4iX4OYNua?dl=0.

5. Results and discussion

4.1 SVM

Linear SVM has generally a good result. The results for different C values are shown in

Figure 2. For most labels, changing C value does not affect the classification result much. But for labels of Good_for_lunch, Restaurant_is_expensive and Ambience_is_classy, a tiny C value caused the result to drop significantly. This means the separating plane should be very small. It makes sense because for these three labels, they are only 17%, 21% and 20% of total restaurants, and allowing more misclassified labels would cause all restaurants to be classified as negative examples.

On the other hand, the result of non-linear kernel is actually significantly lower than the linear kernel. For the three labels mentioned above, it tends to classify all restaurants as negative

examples. I believe this is caused by the small separating hyperplane and there are way too many features in the higher order SVM kernel.

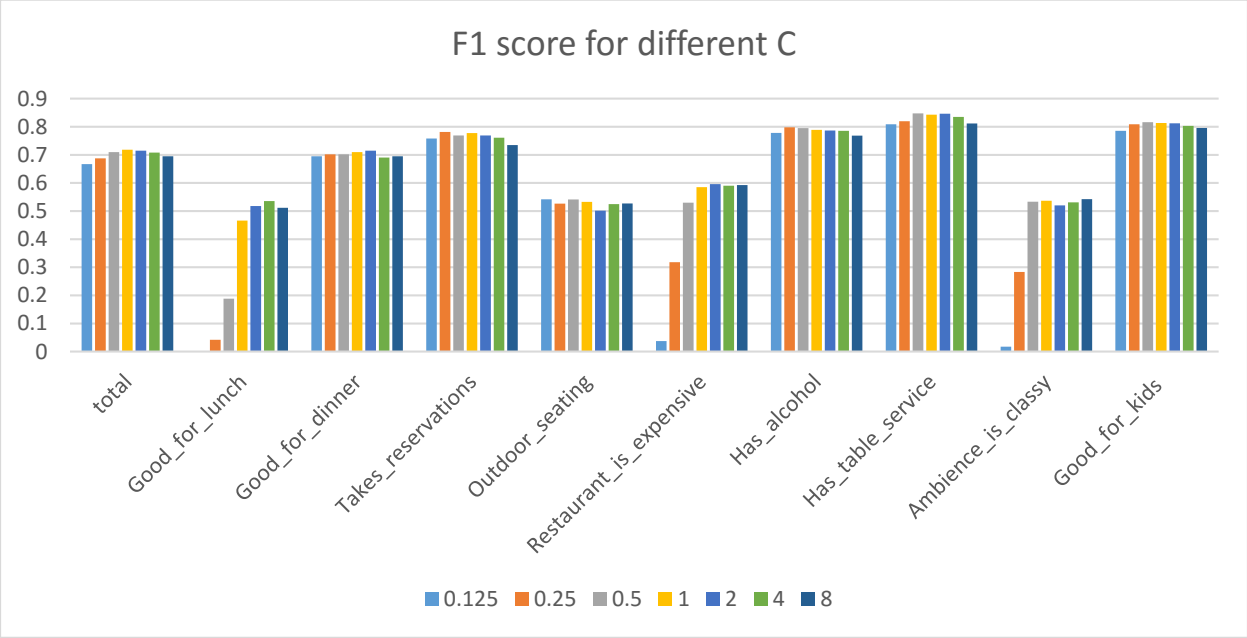

Figure 2. SVM F1 score for all labels with different C value

4.2 KNN

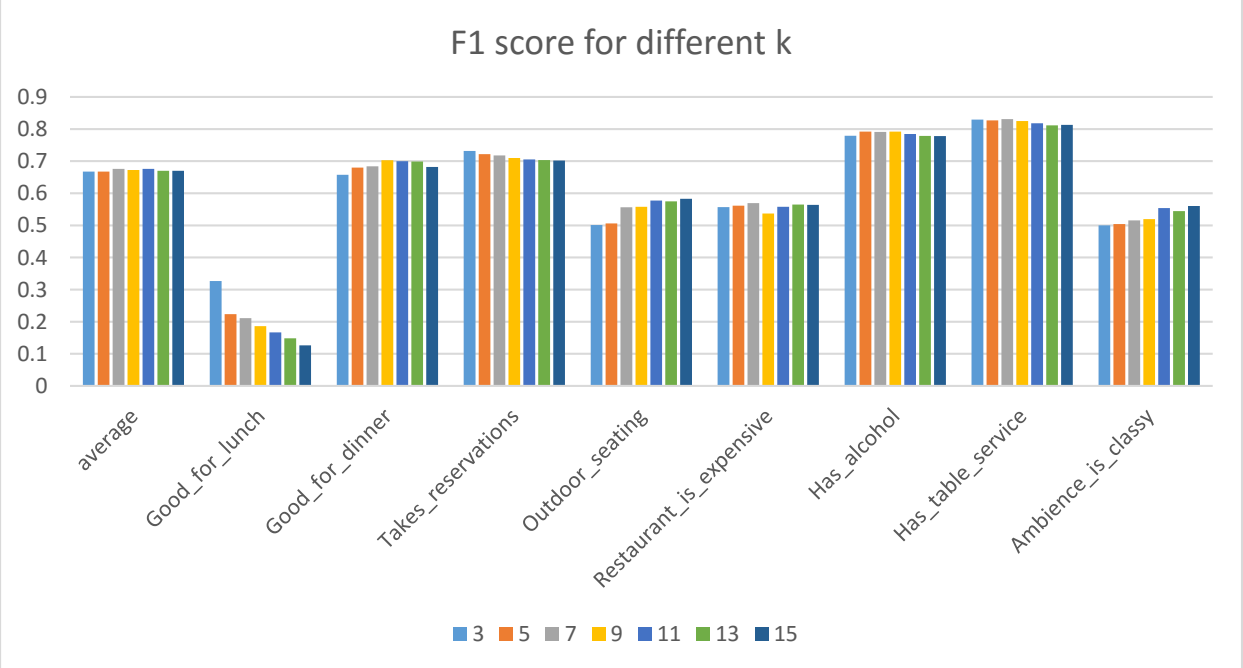

Figure 3. KNN F1 score for all labels with different K value

After tuning the K value, the resulting F1 scores are shown in Figure 3. In fact, the result of KNN is well above my expectation. For labels of Restaurant_is_expensive and Ambience_is_classy, KNN has a constantly better performance than SVM. Also if we consider the consistency of result regarding different parameters, the performance of KNN is more stable than SVM.

4.3 Neural Network

Although I have spent a week implementing and tuning the neural network, the result is significantly worse than the result of linear SVM and KNN. In fact, the result is very similar to SVM with higher order kernel. For labels that belongs to most restaurants, it tends to classify all restaurants as positive example, and vice versa.

The major reason for the pitfall in neural network is that because the major functionality of neural network is to introduce higher-order features, and higher-order features based on HOG features do not work well. A solution might be to feed feature-wise comparison rather than the raw features to the network. We can rank the features based on the comparison result and give labels based on ranking.

Another problem is that the random initialization of a huge number of parameters and the relatively small training example lead to multiple local minimums that changes the resulting F1 score all the time. Since this project is about classifying business rather than images, feeding the image-level features does not improve the performance much, though it solves the problem of multiple local minimum.

4.4 Final F1 score

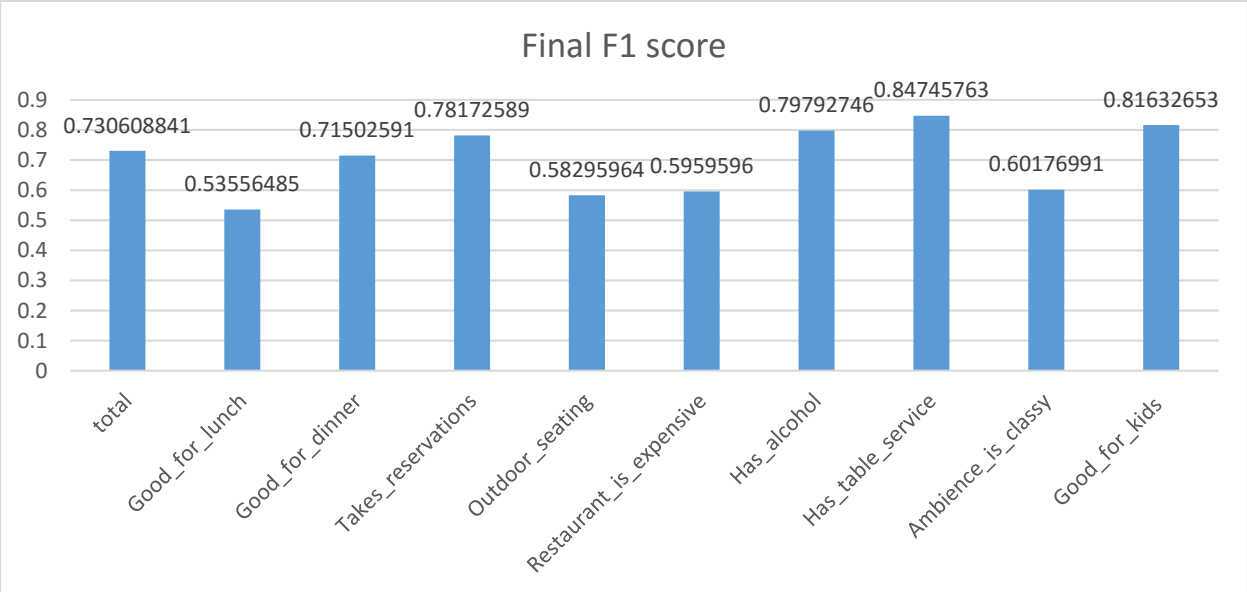

Figure 4. Final F1 scores for individual classes

good_for_lunch	good_for_dinner	takes_reservations
SVM(C: 4)	SVM(C: 2)	SVM(C: 0.25)
outdoor_seating	restaurant_is_expensive	has_alcohol
KNN(K: 15)	SVM(C: 2)	SVM(C: 0.25)
has_table_service	ambience_is_classy	good_for_kids
SVM(C: 0.5)	KNN(K: 19)	SVM(C: 0.5)

Table 1. Parameters for individual classes

The best F1 score is given in Figure 4 with corresponding parameters shown in Table 1. Since I only used HOG features to train the overall dataset due to time constraint, I do not expect the result to be very high. The average F1 score for all labels is 0.69719.

I have also done cross validation by spilting training data with different 80% and 20%. The resulting mean F1 score is 0.69403, which is very close to the final F1 score. So the classification methods I used is not overfitting on the training data.

The F1 score for the hidden test set from Kaggle is 0.69059. This score is well above the benchmark for random guess, which is 0.44692. However, there is still a long way from the result of the first place, which is 0.83177. The top solution has more complicated feature extraction and classification methods as discussed in previous sections. On the other hand, my solution regarding business feature extraction and classification is very close to the solution of the 4th place, which has F1 score of 0.83076. The difference mainly lies on the fact that the 4th team extracted features from a fine-tuned pre-trained Inception V3 network.

5. Conclusions and Future work

5.1 Conclusions of the current model

For me, the lack of prior knowledge about CNN is a huge disadvantage in this project. If I can start over again, I would directly jump to CNN feature extraction. I would test with different CNN features from various CNN open source implementation rather than spending time testing with HOG and SIFT features.

Also another mistake that I made is to run everything locally, which takes enormous time even with the simplest HOG features. I should be prepared with this massive training and testing time.

Although I have spent a lot of time on different classification methods, the result was not significantly changed. In fact, the most time should be put on feature extraction rather than classification. An interesting finding is that if we consider the time for model training and testing, KNN actually has the characteristics of zero training time, short prediction time and good performance. This made KNN a great method to get the preliminary result in image classification problem.

5.2 Possible future improvement

5.2.1 Image-level feature extraction

As mentioned above, CNN would definitely improve the final result by a huge amount. I might take CS231N in order to get myself more familiar with CNN.

Another point is that although my score is higher than the benchmark of using color features (0.64598), I have not used any color-related features. Adding color-related featured might improve the final result.

5.2.2 Business-level feature extraction

An approach that I have not tried is to compare two restaurants, one with the label and one without, and generate features based the difference of business-level features. In the prediction stage, all restaurants can be ranked based the label of their feature difference. Since the labels we feed to SVM can be either positive or negative, depending on how we take the difference, we should be able to solve the issue with current SVM that some labels have too many negative examples than positive examples, making classification one-sided. This method should also work pretty well with Bag of Words features, and I believe if I could finished extracting BOW SIFT features, I could get a boost from it.

5.2.3 Classification

SVM and KNN both work reasonably well for this project. However, I noticed that no one has tried to examine the relation between two labels. So it is possible to use some conditional probability to adjust the result of labeling. This may be able to capture characteristics like “restaurants providing alcohol is not good for kids”.

Another finding is that F1 score is generally low for labels with few positive restaurants, especially for label good_for_lunch. From Figure 2 and 3 we can see that to get the best result for this label, C value should be large and K value should be small. Since for SVM we generally want to feed in equally distributed positive and negative examples, using the feature difference as mentioned above should lead to a better result.

Reference

[1] O. Boiman, E. Shechtman and M. Irani. *In Defense of Nearest-Neighbor Based Image Classification. Computer Vision and Pattern Recognition*. CVPR 2008.

[2] Yunpeng Li, David Crandall and Daniel Huttenlocher. *Landmark Classification in Large-scale Image Collections*. 2009, IEEE. 1957-1964.

[3] Anna Bosch, Andrew Zisserman and Xavier Munoz. *Representing shape with a spatial pyramid kernel*. 6th ACM international conference on Image and video retrieval. 401-408.