Yelp Multiclass Image Classification

Julian Yeh
DSC
UCSD
cyyeh@ucsd.edu

Edwin Huang
DSC
UCSD
edh021@ucsd.edu

Sabrina Ho
DSC
UCSD
ssh026@ucsd.edu

Howard Lin
CSE
UCSD
hol119@ucsd.edu

I. Abstract

AI has been powering so many different ways to help humans on searching, discovering, and eventually getting us what we want, from speech recognition for translating spoken words into text to facial recognition for online payment. And visual search is one of them. With visual search, we can take an image and search for what we want. This is an important feature that is being used in many applications in life such as online shopping.

In our research, we attempt to classify images into its corresponding classes using the 2019 Yelp online challenge dataset. We performed and produced data visualizations for exploration and extracted a variety of features such as a bag-of-words on captions and images as vectors. With all the extracted features, we tested and evaluated a range of models to solve our multiclass classification problem. The most successful model was a Random Forest classifier trained on the top 50 principal components with a testing score of 0.965.

II. Introduction

These days, pictures are a part of everyones life. Everyone takes pictures whenever and wherever. When we see something we like, we take a picture of it or when we go out to eat, cameras always eat first.

This year Yelp provided datasets on images, business, users, and reviews to challenge their competitors to classify whether an image is beautiful or not. With this inspiration, we decided to classify images. But instead of classifying its beauty, we are predicting an images label. We did this through using different features and models.

III. Dataset Exploration

A. Data Collection

This dataset collected by Yelp contains data on photos, businesses, users, and reviews. The files we used in this predictive task were:

yelp_photos/photo.json:

– contains **200,000** entries

The following are the attributes of this dataset:

- business_id: the business id
- caption: the caption of the photo
- label: the label of the photo (inside, food, outside, drink, menu)
- photo_id: the unique photo id

yelp_photos/photos/:

– contains **200,000** entries

yelp_dataset/business.json:

– contains **192,609** entries

The following are the attributes of this dataset:

- address: the address of the business
- attributes: a dictionary contains attributes about the business
- business_id: the unique business id
- categories: a list of all categories the business belongs to
- city: the city where the business is located in
- hours: the hours of the business
- is_open: binary value of whether or not the business is opened

- latitude: the latitude of the business
- longitude: the longitude of the business
- name: the name of the business

B. *Data Analysis*

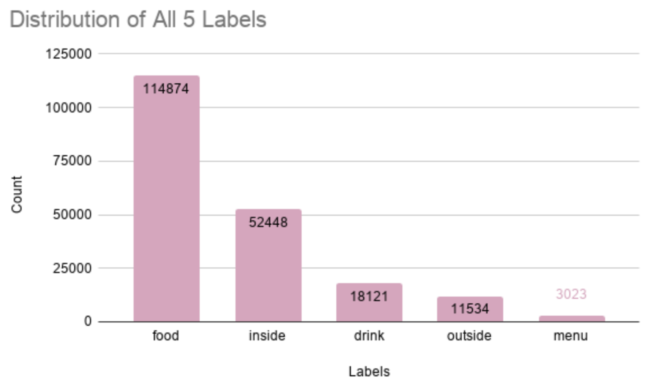

Fig. 1. Distribution of labels in data set.

*1) **Label Distribution:*** The chart above displays the distribution of the labels among the data set in yelp˙photos/photo.json. There are 200,000 unique photo˙ids and more than half (114874) of the images are labeled as food. From the distribution, there is an obvious imbalance between the classes. The second largest group of labels consist of about 52448 images, which is not even half of the images labelled as food. Furthermore, the images that are labelled as drink (18121), outside (11534), or menu (3023) are not even half of the images labelled as inside. Therefore, in our procedure, we will have to take the heavily imbalanced data into consideration in order to classify the image labels.

*2) **Captions:*** In the photo.json data set, not all rows of data contain image captions. More than half of the original data set contains no image caption (53.9%). We decided to discover whether the imbalance between labels will also have an affect on the imbalance between images with and without caption with different labels. We discovered that apart from food images, all other images under other labels have more images without captions.

*3) **t-SNE:*** This is the graph of all 2 dimensional vectors converted from original image vectors of 2240 dimensions using t-SNE (t-distributed Stochastic Neighbor Embedding.) This is a nonlinear dimensionality reduction technique used to visualize high dimensional vectors in a low dimensional space. In this case, we reduced 2240 dimensional vectors into 2 dimensional

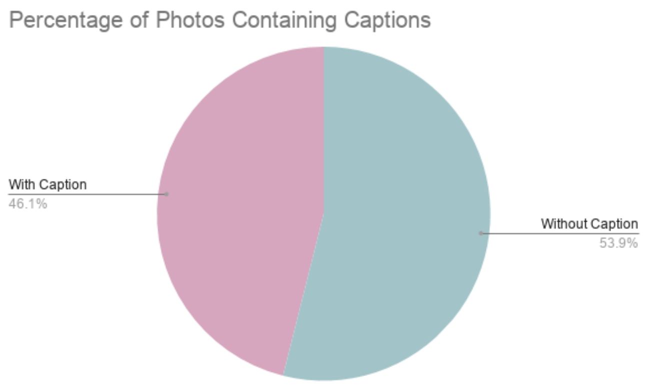

Fig. 2. Percentage of Photos containing Captions.

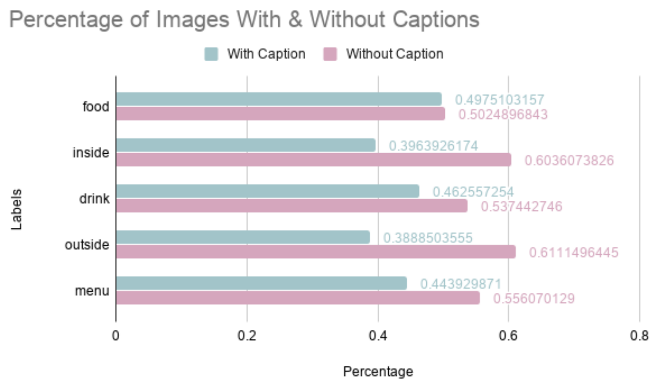

Fig. 3. Percentage of Photos containing Captions by Label.

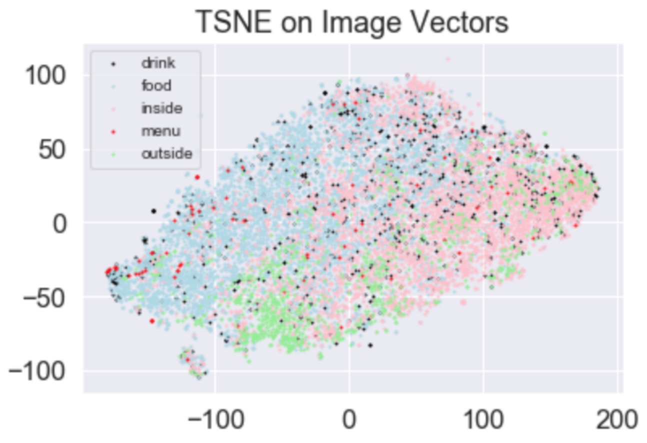

Fig. 4. t-SNE of Image Vectors by Label

space. From the graph, it is clear that the main three clusters are food (light blue), inside (pink) and outside (light green) while the images labeled as drink and menu did not form obvious clusters.

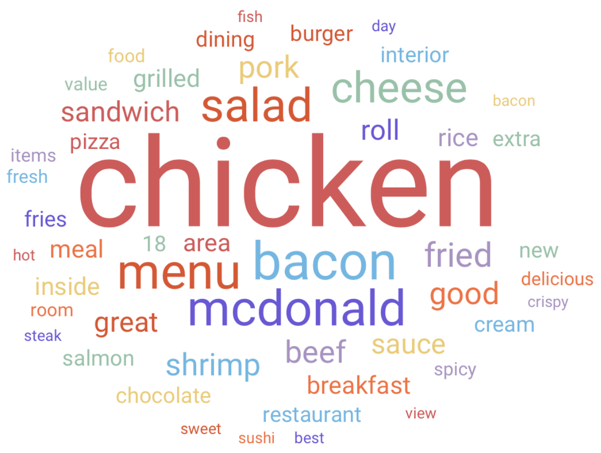

Fig. 5. Word Cloud of Top 50 Words

4) ***Top Occurring Words:*** This is the word cloud of the top 50 words appearing in the captions. The chart is created using CountVectorizer to retrieve the top 50 words among the captions. Furthermore, common stop words like the, a and and are removed from the CountVectorizer. As seen from the word cloud, chicken occurred the most frequently. After looking into the dataset, around 97% of the images are labeled as food. Most captions that include food names or food sources are labeled as food. Another example is menu which also occurred many times. On many occasions, the images with menu in the captions are labeled as food 311 times and as menu 277 times. After looking into the dataset, captions like the inside menu, may 2012 tasting menu or Foie gras custard - Kaiseki menu are not labeled as menu but as outside, food and drink respectively. This may be caused by confusing user interface or incorrectly input by users but we decided to use the original labels without manually changing the ones that are confusing.

5) ***Star Ratings:*** The histogram above shows the distribution of all star ratings across all businesses in the dataset *business.json*. From it, we can see that it is a right skewed distribution with

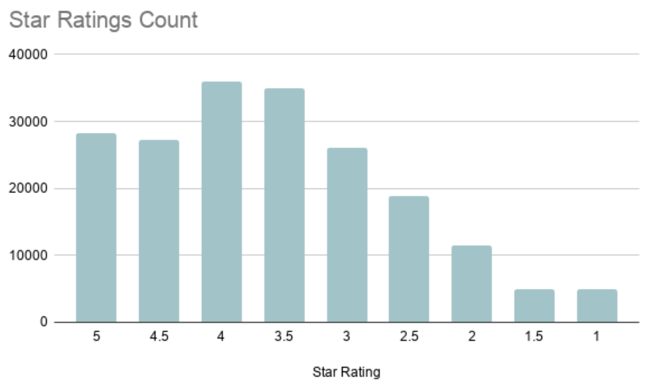

Fig. 6. Star Ratings Count Across All Businesses

$$mean \approx 3 < median \approx 3.5 < mode \approx 4$$

.

IV. Predictive Task

A. Baseline Model

As the data analysis shown above, we can see from that the distribution histogram, the label food has the highest percentage among all five labels and menu has the lowest. Therefore, we designed the baseline model to predict the label of a given image if the caption includes the word related to the label, and predict as food if the caption is empty since our dataset has approximately 57% labeled as food. Additionally, if the caption contains words related to more than one label, we will prioritize the label with the lowest percentage of the distribution. Lets say, given an image with the caption I like the interior of this place! we will then classify this image with the label inside. The accuracy of our baseline model is 0.60763.

B. Metrics

For model evaluation, we will be using balanced accuracy as the metric to measure the performance of our models. According to the data visualization of the distribution in the previous section, it is obvious that the classes are heavily imbalanced. The label food was classified at least twice more than other labels. Therefore, we decided to use the balanced accuracy in order to consider the class imbalance.

The formula for balance accuracy is the following. It is the average of recall obtained on each label. The formula is

$$balanced\_accuracy = \frac{1}{2}(TPP+TNN) = \frac{(TPR+TNR)}{2}$$

TPR represents the true positive rate which is the ratio of actual positives that are accurately identified whereas TNR represents the true negative rate which is the ratio of actual negatives that are correctly identified.

C. Data Processing

*1) **Data Imbalance Solution:*** In order to combat class imbalance of our dataset, we used over sampling as a technique to ensure classes are balanced. Prior to using oversampling, our distribution of data was heavily towards the label food. After using oversampling, we were able to ensure that all the labels have the same number of images in our training set (114874) and testing set (38,000). We performed random oversampling using RandomOverSampler from the imblearn library. As aforementioned, we then trained our models using training sets with more balanced labels.

This method of random oversampling samples more data from the minority class. For example, in our case, the selection of samples will consist of more images under the menu class when compared to the food class. However, even though this technique leads to a more balanced distribution of data, it may be more likely to lead to overfitting and perhaps some useful data may be dropped.

*2) **Image Dimension Reduction:*** The original images downloaded from the dataset are of various sizes. We wanted to ensure that images are comparable so we changed the size of some images to a specific dimension to stay consistent. We used opencv library and the imread function to read in images as vectors and resize the images. The resulting vectors are of 2240 dimensions for all images. Having a vectorized representation of the images allows us to perform numerical computations and compare / contrast between various images.

*3) **Caption text processing:*** We have also incorporated image captions as part of our features in hopes that they convey meaningful information on what the images are. The way we perform text preprocessing is with converting texts to lowercase, remove the punctuation and remove stopwords before using CountVectorizer from sklearn. The reason why we remove unnecessary symbols or repetitive words is because we only wanted to extract key features from the captions. Stopwords like the or a do not contribute much to our model since most captions will include these words.

D. Feature Extraction

*1) **Image Vectors:*** We first had to get a numerical representation of the images. After resizing the images, we were able to retrieve a vector of 2240 dimensions for each of the images in our dataset. During training and testing, we used these 2240 dimensional vectors as input to our model. Having a numerical representation of the images are handy when we wanted to to fast numerical computation and comparisons.

*2) **Dimensionality Reduction:*** Due to the image vectors living in high dimensions (2240), Principal Component Analysis (PCA) was used to construct a feature matrix of lower dimension. PCA is a statistical method that finds the principal components or vectors that explain most of the variance of the data. The goal here is to attempt to represent the original data with as few components, or dimensions, as possible. Due to memory issues, only components ranging from 1-800 were considered during the gridsearch process to find the optimal number of components.

*3) **Bag-of-Words:*** After processing the text, a bag-of-words feature matrix was built using the sklearn's CountVectorizer. We used grid search to find the best hyperparameter for max_features as the main hyperparameter to tune. We performed a grid search on values from 10,000 - 47,054 to discover the optimal vocabulary size for the model.

V. Model

A. Training/Testing Splits

Throughout the entire study, the dataset was split into training and testing sets using a 75/25 split.

B. Algorithm Selection

Three classical supervised algorithms were considered for this study as follows:

*1) **Naive Bayes:*** The model is based on Bayes theorem which assumes independence of the predictors or features. This means the model makes an assumption that each feature contributes equally and independently to the outcome of our predictions. Specifically, this study uses sklearn's implementation of the Gaussian Naive Bayes learner.

*2) **Logistic Regression:*** Logistic regression is another probabilistic classification model. However, it uses a sigmoid function to transform the result to return a probability that is mapped to two or more classes. Again, sklearn's implementation was used for this study.

*3) **Random Forest:*** Random forest classifier is an ensemble learner that is built upon multiple decision trees to generate its predictions. Each individual decision tree in the algorithm will produce an output. The output that is generated the most frequently will be the final label decision. For the random forest algorithm, we also decided to use the implementation from sklearn.

The process of selecting the best classifier among the three was straightforward. Since the images were already in vector forms, each of the classifiers were trained on the vectorized dataset with no additional features and with default parameters. The classifier with the highest balanced accuracy score on the testing set was used to train on more complex features later on. Table I below summarizes the selection results.

Algorithm	Training Score	Testing Score
Naive Bayes	0.427	0.426
Logistic Regression	0.548	0.546
Random Forest	0.999	0.949

TABLE I

ALGORITHM SELECTION

C. Features Testing

After selecting the optimal classifier (Random Forest), two different sets of features were used to see which one gives better predictions. The first set of features were the PCA components with varying numbers of components, and the second set of features were the bag-of-words with varying sizes of vocabulary. Table II and III displays the resulting scores from the first and second set of features respectively.

Components	Training Score	Testing Score
1	0.985	0.888
50	0.999	0.952
400	0.999	0.944
800	0.999	0.937

TABLE II

FEATURE: PCA COMPONENTS

D. Tuning

At this point, it was clear that the best model to use was the Random Forest classifier trained on the top 50 principal components. In order to beat the original top test score of 0.952, we attempted to find the best parameters for the classifier using a brute-force approach. By using sklearn's GridSearchCV function, several values were tested for the following parameters: class_weight, max_features, and n_estimators. Finally, based on the gridsearch results, the optimal classifier had the following parameters:

Vocab Size	Training Score	Testing Score
10000	0.547	0.531
20000	0.550	0.534
30000	0.550	0.533
40000	0.550	0.533
47054	0.550	0.533

TABLE III

FEATURE: BOW

$$class\_weight = balanced$$

$$max\_features = log2$$

$$n\_estimators = 100$$

Table IV summarizes the performance of the best model.

Model	Training Score	Testing Score
RF w/ PCA	1.000	0.965

TABLE IV

BEST MODEL

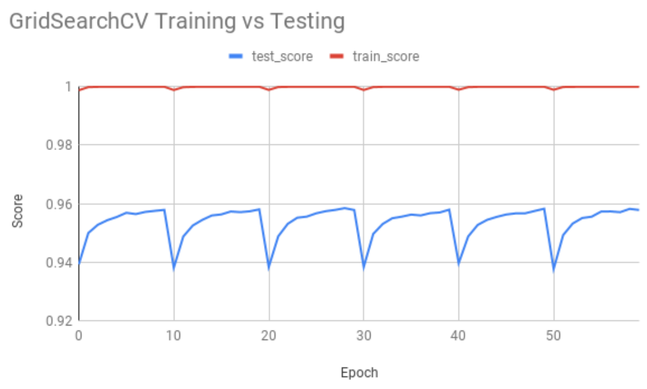

Fig. 7. GridSearch Training Score vs. Testing Score

The line chart above is the training score versus testing score on the gridsearch results. From the graph, it is obvious that there are six cycles, which are the six different combinations of the parameters *class_weight*, *max_features*, and *n_estimators* as mentioned above.

E. Additional Approach

1) ***Cosine Similarity:*** In addition to using traditional learning algorithms, we've also implemented a custom model using cosine similarity as the main feature. The similarity score is the calculation of the angle between the two feature vectors. If two images have 1 as their cosine similarity, this means the two images are identical. Cosine similarity between X image and Y image is the dot product between the two vectors of features divided by the magnitude of the product between X and Y. This method allows us to generate a similarity score between 0 and 1 inclusive, which is used in our model to generate prediction of labels. The equation is as the following:

$$cosine\_similarity = cos(\theta) = \frac{A \cdot B}{\parallel A \parallel \parallel B \parallel}$$

This similarity score allows us to discover some of the most similar pictures in our training set. The way we constructed this model is by first randomly selecting 5 representative images of each class from the training set. To select these images, we first filtered out businesses that are the most popular. Our definition of a popular business is one with more than 33 reviews and has 4.8 or greater ratings. The reason why we used these specific numbers is that 33 is the average number of reviews and 4.8 is the 80th percentile of the ratings for all businesses in the business dataset. From the filtered businesses, we grouped the training data by labels then randomly selected a representative photo within each label.

After retrieving the representative images of each class, we iterated through the test set and calculated the cosine similarity of the image with all the 5 representative images. Then according to these similarity scores, we simply recorded the maximum similarity and the class this representative images belong with and classified this image with the corresponding label. The reason why we selected only one image to represent each label was to reduce the runtime. It turned out that the resulting balanced accuracy was around 0.294, which is a lot lower than all the other models.

VI. Literature

The dataset we used came from the Yelp Restaurant Photo Classification competition on Kaggle that started back in 2014. Given the nature of the data, it is trivial to conclude that the task is an image classification problem. It was originally used to predict attribute labels for restaurants using user-submitted photos[1].

Similar research has indicated that the current state-of-the art methods used to tackle image classification revolves around variations of the Convolutional Neural Network (CNN). Specifically, ResNet was able to achieve a top-5 error of 3.6% on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2015[2]. This motivated the study to compare how traditional feature extraction methods and algorithms would fare against the new methods that are more towards deep learning. Surprisingly, our best model performed exceptionally well scoring-wise, but there were several flaws that are noted in the following section.

VII. Results & Conclusion

From Table I, we can see that the Random Forest classifier outperformed the other two classifier by a significant margin while having an near-perfect training score. This was an early indication that the model might be over-fitting or our approach had a severe flaw. Nevertheless, we proceeded along to testing out different features.

As shown from Tables II and III, the model heavily favored training on PCA-reduced components than the bag-of-words matrix. At first glance, this came as a surprise since we thought the captions were extremely indicative of what the images were portraying. However, it made sense after noticing that a lot of the images had no captions attached to them.

Going back to what was mentioned towards the end of the previous section, the optimized model had a perfect score of 1.0 on the training set and a really high score of 0.965 on the test set (Table IV). The first reaction was to blame it on over-fitting to the training data, but there are two reasons to why it should not have been the case. First, cross-validation was performed as part of the gridsearch process with multiple validation sets to ensure that the model is able to generalize to unseen data. Second, if the model DID over-fit to the training data, then it should have performed poorly on the testing set. Instead, it had the best testing score out of all the models. Still, we were not convinced that our model was actually able to perform well for this image classification problem, otherwise, there would not have been a need for CNNs to emerge.

To figure out what went wrong, we had to take a step back to before we even began training models. Recall that the original data was highly imbalanced towards the 'food' label. This led to the application of the oversampling technique in order to balance the data. The oversampling technique was basically randomly re-sampling data from all classes/labels until all classes are balanced. We then split the data into training and testing sets afterwards. This implies that the training data might have leaked information to the testing data since there are repeated data instances and the same instance could have appeared in both sets. The supposedly high scores of the best model is now irrelevant since the model is guaranteed to make a correct prediction on the test set of the instance is also a part of the training set. Thus, this study cannot conclude whether or not traditional machine learning methods are able to compete with with modern-day deep learning methods on image classification problems.

References

[1] Yelp Restaurant Photo Classification. Kaggle, www.kaggle.com/c/yelp-restaurant-photo-classification.

[2] He, Kaiming et al. ”Deep Residual Learning For Image Recognition.” arXiv.org. N. p., 2015. Web. 2 Dec. 2019.

[3] Yelp Restaurant Photo Classification. Kaggle, www.kaggle.com/c/yelp-restaurant-photo-classification.

[4] Peng, Haomin et al. ”Yelp Restaurant Photo Classification.” Cs229.stanford.edu. N. p., 2019. Web. 2 Dec. 2019.