Undervalued Burritos of Yelp

Problem Statement

Reviews and overall star ratings on Yelp often contain irrelevant information to the specific question a potential customer may have. In particular, I am interested in finding restaurants with the highest quality burritos. Selecting the Mexican restaurant on Yelp with the highest overall star rating and the largest number of reviews could be a good place to start. However, this necessarily ignores whether the majority of reviews praise the quesadillas, the service, or the atmosphere. Perhaps the restaurants serving the best burritos are nowhere near the top of the overall star rating because they don't have a parking lot or vegetarian options (as an example).

This could be an invaluable method if successful because it will be extremely easy to generalize. The scope of this project is to find the best burritos, but other applications can benefit from similarly specified queries. Interested in yoga studios with the most popular teachers, laundromats with the fastest machines, or bars with the cheapest happy hours? The possibilities are limitless and hold great potential for identification of underrated establishments in all realms of business.

Dataset and Cleaning

Yelp, the massive review platform, is kind enough to publicly offer reviews for a huge swath of businesses across the country and world. Split into several large JSON files, you can filter review information based on user, business, category, etc. The immediate problem is the sheer size of the data. There are nearly 7 million reviews contained from close to 200,000 businesses.

The first step in getting my hands on the information I'm interested in is downloading the files. They come in a nested JSON format, and are quite large. The file containing all the reviews is almost 6 gb. This is too large to hold in memory so I have to parse through line by line and pull out the reviews I need. Because I am interested in burritos specifically, I need only the reviews for establishments that sell some kind of mexican or tex-mex food.

Thankfully, the JSON file that contains the information on all the businesses can be loaded into local memory and pushed into a dataframe. Once this is done, I iterate over the dataframe and look for the 'restaurant','mexican' and 'tex-mex' tags in the 'business category' dictionary associated with each business. I then pull out the restaurant name and unique restaurant ID (to separate out all the chains) and use that information to parse the reviews file line by line looking for reviews associated with the restaurants I am interested in. This reduction of data results in just over 4,000 restaurants of interest with almost 390,000 reviews. Still large, but much more manageable. There are a few other intricacies I need to complete to get all the information I need in one place, namely transferring information on restaurant name, location, and region from the businesses file.

The last step I need to take in cleaning the dataset is pulling out all the reviews that mention ‘burritos’. This set will be my validation set where I run sentiment analysis on specific sentences to find undervalued burritos. To pull out all the reviews that mention my word of interest, I can use pandas df.str.contains() method to return every review that has some variation of the word burrito in it. Saving this to a new dataframe and dropping the same rows from the large dataset (so I don’t train on these reviews) leaves me with a burritos dataframe of about 50,000 reviews and a finalized reviews dataframe of nearly 330,000 reviews.

Now that I have the reviews I’m interested in with all the information I’ll need to answer my question, I can proceed to some exploratory analysis.

Exploratory Data Analysis (Part 1)

My first goal is to make a model that can accurately predict what star rating is associated with a block of text. I’ll have to begin by taking a look at some numbers that describe my objects of interest. Below is a count of the number of reviews associated with each star rating. The number of 5 star reviews dwarfs every other star rating. This may lead to my model predicting everything as a 5 star review unless I balance the classes. Duly noted, but I’ll move on for now.

Figure 1: Number of reviews in my whole dataset, binned by stars

The next figure looks at the mean character count for all reviews by star rating. This plot looks a little more uniform, the concern being that if there were significantly more words for a certain type of review that a model might be much more successful at predicting that class. Interestingly enough, 5 star reviews have the lowest mean length. I would not have predicted this beforehand.

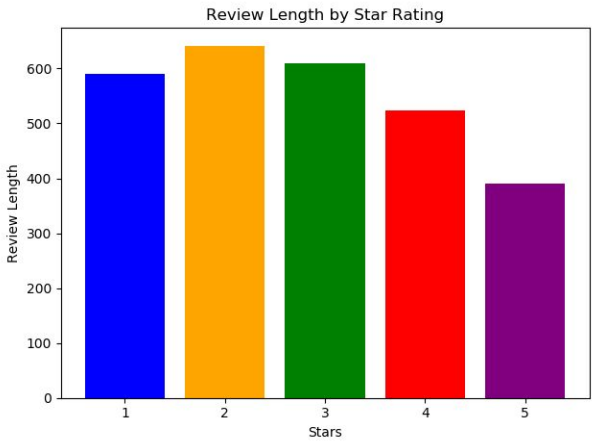

Figure 2: Mean length of reviews binned by stars

Now that I have a little intuition for how review length and count of star ratings are distributed, I want to next looks at the restaurants the make up the 330,000 reviews. Below is a list of the top 15 restaurants garnering reviews (for my purposes here, individual chain's reviews are all added together). As I suspected, chain restaurants make up the majority of the top 15, but not nearly to the extent that it might become problematic. Of all 330,000 reviews in my dataset, only about 2% are for a Chipotle. This is of interest to me because I sincerely doubt that the most undervalued burritos come from somewhere like Chipotle or Taco Bell. If these places made up the majority of my reviews I might have to come up with a different plan.

Chipotle Mexican Grill	6513
Tacos El Gordo	5283
Taco Bell	4504
Chili's	4062
Mesa Grill	3331
Nacho Daddy	3113
Cafe Rio	2913
Lindo Michoacan	2631
El Dorado Cantina	2529
Border Grill	2280
La Santisima	2074
Roberto's Taco Shop	2011
Joyride Taco House	1985
El Pollo Loco	1957
Barrio Queen	1954

Figure 3: Top 15 restaurants garnering reviews (here, all Chipotles and all Taco Bells)

The list above is also interesting because it seems to be biased towards restaurants in the American West. Specifically, I know 'Barrio Queen' is a well respected traditional mexican restaurant in Arizona, 'Tacos el Gordo' a highly regarded California mainstay, and 'Nacho Daddy' a Nevada nacho champion. I previously presumed the dataset from Yelp included a large swath of the nation but I better take a closer look at the distribution of states and regions to see what I'm working with.

Las Vegas	99034
Phoenix	65851
Scottsdale	25754
Charlotte	15847
Mesa	11925
Henderson	11528
Pittsburgh	10407
Tempe	9781
Chandler	9476
Gilbert	9409
Glendale	6136
Cleveland	5725
Madison	5203
Peoria	4794
North Las Vegas	3558

Figure 4: Top 15 cities represented

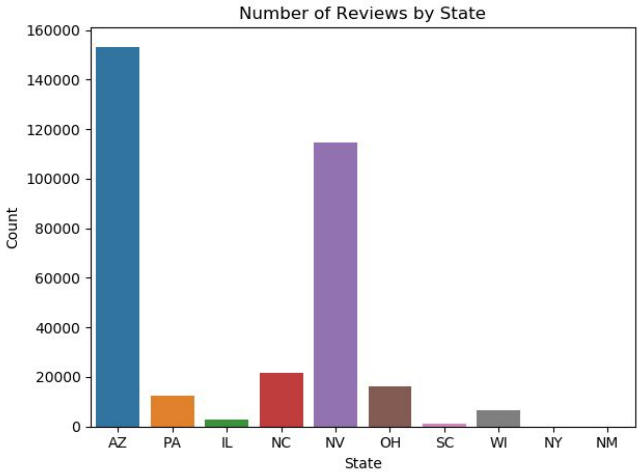

Figure 5: Number of total reviews binned by state

As the two images above confirm, I actually have a fairly random distribution of states. By far, Arizona has the most reviews as it seems that all their large cities have been included. After that, Las Vegas and the surrounding areas make up a large chunk of the reviews set. Moving forward, it might be prudent to split my analysis into different areas or regions so my undervalued burrito list is not a purely Arizona or Nevada heavyweight clash. For now I'll leave it as it will not influence my model building.

Word Tokenizing, Vectorization and a Baseline Model

Now that I understand my data (and some potential pitfalls) a little bit better, I can proceed to the second part of cleaning that goes with this project, natural language processing. I plan to use a simple bag of words model to represent all the reviews in a vectorized form that a model can understand.

To begin, I will work with the standard NLTK library, utilizing its corpus and built in tokenization to clean the plain text reviews. First I write a function that will allow me to use pandas df.apply() method on my reviews dataframe. The function is exceedingly simple, it takes the whole plain text review, breaks it up into a list of tokens (words in this case), and then removes every stopword contained in NLTK's stopword corpus. Applying this function to the whole dataframe takes about 30 minutes, or 5 milliseconds per review. This is impressively fast and shows how optimized the NLTK library must be. When finished, the reviews dataframe has a new column for the 'tokenized and cleaned text'. Of course, I still keep the plain text review so I can later use it for sentiment analysis.

After tokenizing and cleaning every review, I can now turn these lists of words into a sparse matrix of feature vectors. I will use Scikit-learn's CountVectorizer as a starting point because it is transparent and will offer a strong baseline. Instantiating a new CountVectorizer object and fitting the vectorizer to my list of documents returns a corpus of every word that occurs through the whole set. There are around 190,000 entries in the corpus.

Next, I can transform the list of documents into a list of feature vectors. The CountVectorizer does this in a straightforward manner, representing each review with a matrix of 190,000 columns denoting the number of times that specific word is contained in the given document. This is an incredibly sparse matrix as you might imagine. There are techniques to reduce the size of the corpus, as much of those features will have little bearing on the actual model, but this is just to get a baseline so I'll leave it for now.

Once the every document has been transformed into a feature vector, I can fit and run a model to see where the baseline stands. I'll begin with a Multinomial Naive Bayes, as online literature suggests this model as being particularly useful for working with text. I first separate my massive dataset into training and testing sets. Again, the features here are the vectors of word counts and the target is star rating. Fitting the model to the training data, and predicting labels from the testing data returns the classification report printed to the left.

	precision	recall	f1-score
1.0	0.64	0.74	0.69
2.0	0.41	0.29	0.34
3.0	0.45	0.33	0.38
4.0	0.50	0.46	0.48
5.0	0.75	0.84	0.79

Unfortunately this baseline model does not perform as well as I had hoped. However, I believe the biggest problem is the one detailed above (see figure 1), extremely imbalanced classes. It seems that because there are so many 5 star reviews relative to every other class (but particularly 2 and 3 star reviews), the model has learned to predict a 5 star review for any instance of language that isn't exceedingly negative. This gives fairly high recall for 5 star reviews, but lower precision. To adjust for the imbalance of number of reviews, I believe the classes can be combined without losing sight of the final goal of this project.

Making a new column in my reviews dataframe for a binary review classification is fairly straightforward. I construct a dictionary of star ratings (1-5) to 2 new classes, 'Good' for 5 and 4 star reviews and 'Bad/Neutral' for 1, 2 and 3 star reviews.

	precision	recall	f1-score
Good	0.90	0.93	0.91
Neutral/Bad	0.86	0.80	0.83

Performing the same steps as above, but changing my target variable to this new column yields the classification report to the left. As I expected, the classifier now performs significantly better when predicting our target variable. Because of the significantly reduced complexity, the classifier now just has to decide if a review is mostly positive or mostly negative. 9/10 times the classifier predicts a review as being positive, it is!

This is a fairly solid baseline, but can definitely be improved upon using techniques such as word-lemmatization, the inclusion of punctuation, or sentiment analysis. For now, I'll proceed on with the project and increment later if desired.

Sentiment Analysis

The second half of this project attempts to find out if I can use sentiment analysis to identify undervalued burritos, relative to star rating. Specifically, using the subset of reviews that mention the word 'burritos' I will pull out the sentence concerning the topic of interest and run sentiment analysis on it.

The first step in this process is splitting the sentence or sentences that contain the word 'burrito' (or analogs) from the rest of the review. To do this, I will first apply the custom function 'split_sentence_return_burrito'. This function checks for the presence of punctuation (some reviews do not have punctuation for whatever reason) then applies the regular expression re.split on sentence ending punctuation. This returns a list that can be iterated through to pull out the sentence or sentences that contain the word 'burrito'. The 'burrito_sentence' is then added to a new column of the burritos dataframe.

The second step in this process is running these collected sentences through a sentiment analyzer. There are many out of the box sentiment analysis APIs for various use cases and purposes. Originally I attempted to use Google's Natural Language API as it is highly regarded but the runtime on my machine was massive (~32 hours for sentiment analysis of every burrito sentence). The API I decided to use instead is VADER (Valence Aware Dictionary and Sentiment Reasoner). VADER is a "lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains" (https://github.com/cjhutto/vaderSentiment). The API is fast, clean and relatively straightforward.

The output, given a sentence, is a polarity dictionary with 4 measures, positive, negative, neutral, and compound. Positive, neutral and negative are percentages of words that fall into each class in a given sentence and add up to 1. The compound score is the more useful measure as a true sentiment score, offering a value from -1 (most negative) to 1 (most positive). The API calls this compound measure a "normalized weighted composite score." To get a sense for how this works, an example is reproduced below. There's a lot going on here, but the workflow is simple. I take an original text review, pull out the sentence(s) that have the word 'burrito', and run it through the sentiment analyzer. A polarity dictionary is returned where we see that this sentence has been correctly classified as positive and neutral with a fairly high compound score.

```
Original review: Just love, love love it here.  I love the cute beachy decor and the place is always clean.  The menu has a
great selection.  I just love the California Burrito - it is yummy.  We get it without onions.  My daughter loves the Cheese
Quesadilla.  We always get a side of fries.  They are now selling Hamburgers too - not sure why they added it but if you are
trying to please everyone in a group for me it is a great thing to add to the menu.  I love that you can call your order in
and it is ready for pick-up.  For a great mexican food - check this out.

Sentence to be analyzed: ['i just love the california burrito - it is yummy']

Polarity dictionary: {'neg': 0.0, 'neu': 0.656, 'pos': 0.344, 'compound': 0.6369}
```

Figure 6: Positive review and burrito sentiment classified by VADER Sentiment Analyzer

An example of a negative sentence is reproduced below. This review is obviously quite poor overall, and the sentence about the burrito isn't much better. The analyzer nails this one too, identifying the negative sentiment associated with 'fake egg' and 'not very satisfying'. The compound score, as expected, is quite low.

```
Original review: We split a breakfast burrito, which had a decent chorizo flavor, but the egg was fake and not very appetizi
ng. The salsa was not good at all and we were afraid to eat it after visual inspection. I also felt a bit sick afterwards an
d this was the only thing I ate. Chili's express might be a better choice for airport dining.

Sentence to be analyzed: ['we split a breakfast burrito, which had a decent chorizo flavor, but the egg was fake and not ver
y appetizing']

Polarity dictionary: {'neg': 0.196, 'neu': 0.804, 'pos': 0.0, 'compound': -0.631}
```

Figure 7: Negative review and burrito sentiment classified by VADER Sentiment Analyzer

So now that I have a sentiment analysis tool, I can plug in every burrito sentence 1 by 1 and get back a score that may be useful in ranking burritos. Again, the goal here is to identify those restaurants that may not have great star ratings overall but are consistently complimented on their burritos.

Finding the Best Burritos

As mentioned previously, one might expect star ratings to be a good starting spot for finding the best burritos. So let's begin there. Looking at all the reviews that mention burritos, grouping by restaurant and only keeping the restaurants that have >10 reviews(793 restaurants) gives me this list (reproduced below). The best burrito restaurant according to this ranking system is Pollos LaChuya in Tempe, Arizona with a mean star rating of 4.92 based on 12 reviews. However, their mean burrito sentiment score is fairly low, 0.16. Their burritos don't make anyone feel too strongly, apparently.

	Restaurant Name	Mean Sentiment Score	Mean Stars	Review Count
562	Pollos LaChuya	0.169492	4.916667	12
529	Cocina Madrigal	0.181900	4.875000	16
130	Garden Grill	0.298246	4.846154	13
78	El Frescos Cocina Mexicana	0.331225	4.812500	32
177	Kiss Pollos Estilo Sinaloa	0.282415	4.769231	26
386	La Purisima Bakery	0.222075	4.750000	12
557	El Cordobes	0.170783	4.750000	12
150	Del Yaqui	0.289464	4.727273	11
421	Humberto's Mexican Food	0.213009	4.727273	11
346	Tacos Kissi	0.232714	4.714286	14

Figure 8: Top burrito restaurants with more than 10 reviews sorted by star ratings

By changing the sort_values argument, I will next look at the top 10 restaurants based on mean sentiment score. The top 10 has completely changed. The new number 1 is Rosarita's Beach in Las Vegas, Nevada (although now closed - not a great review of this method) they garnered a mean sentiment score of 0.50. Their mean stars tell us this is a poor restaurant, but apparently their burritos make reviewers feel quite positive.

	Restaurant Name	Mean Sentiment Score	Mean Stars	Review Count
0	Rosarita's Beach	0.508306	3.375000	16
1	Mexquite Mexican Eatery	0.456758	4.000000	12
2	Betitos Mexican Food	0.455457	4.285714	14
3	Super B Burrito	0.454376	3.644444	135
4	Burrito's Juarez	0.449913	3.600000	15
5	Los Picos Parrilla Restaurant	0.446238	3.769231	13
6	Mexico City	0.433089	3.388889	18
7	Chronic Cantina	0.427884	3.684211	19
8	Jose' and Tony's Mexican Restaurant	0.423275	3.250000	12
9	Amigos Tacos	0.417572	3.916667	36

Figure 8: Top burrito restaurants with more than 10 reviews sorted by sentiment score

To dive further into this method, let's look at a few samples of burrito reviews from each of the top ranking restaurants. First are three reviews for Rosarita's Beach (top based on sentiment), the next three reviews are for Pollos LaChuya (top based on stars). What is immediately apparent is that the reviews for Rosarita's Beach have more features (words) to work with. This results in a more confident sentiment analysis. For example, the first review shown for Pollos LaChuya was labeled as totally neutral by the sentiment analyzer. While this is correct given the narrow question being asked, there is a high chance that a restaurant with 'phenomenal meats' also has phenomenal burritos. Alas, therein lies the limitation of this relatively simple approach to sentiment analysis.

```
Rosarita's Beach Burrito Reviews

["their breakfast burritos are a pretty good size and it's pretty hearty and very filling with eggs, potatoes, pico and your
choice of either ham or bacon", 'the breakfast burrito we had was better executed as far as the way the eggs and bacon are c
hopped and mixed, better seasoned, and pico is even different ', 'stars for having a better tasting burrito, good location,
good deal and quick service']

['really love the chicken tacos here and my gf loves the veggie burrito']

['my husband ordered the chicken asdo burrito, it was fresh, tasty and awesome\nfree fixings bar, excellent tortillas, clean
and nicely decorated for what it is, a taqueria']


Pollos LaChuya Burrito Reviews

['next time, i might try one of their phenomenal meats in a burrito']

["this is probably one of the best burritos i've ever had", 'i went back a second time just to try the asada burrito', 'the
service was exceptional just be prepared to wait a little longer than those chain burrito places']

["the only reason why i didn't give it a 5 star review is because there was quite a bit of gristle in the carne burrito i or
dered"]
```

However, this does not make the results of this method invalid. As an attempt to use sentiment analysis to yield potentially undervalued restaurants, working with sentences that contain a keyword of interest seems to hold promise. To improve performance and reliability I have some tweaks in mind. First, getting more reviews for each restaurant would definitely help. As illustrated above, sentences classified as totally neutral really hurt a restaurants ranking. Getting more reviews would work towards solving this problem because I could drop the reviews that don't really tell me anything about the quality of the burrito. Sentences such as ***"I think I want to try the burrito next time."*** are actually surprisingly common and weight down reviews that actually praise the quality of the burritos. The second tweak I have in mind is including more information for the sentiment analyzer to work with. Oftentimes, a review will include potentially useful information AFTER the sentence about the burrito, as shown in the following example: ***"I tried the burrito again. Last time it was terrible but with the new cook it's delicious."*** This review would get classified as totally neutral because all the useful information is contained in the second independent sentence. This would be tough to implement, but perhaps searching for a keyword such as 'it's' in the next sentence - hopefully referring to the burrito - could allow this extra information to be kept around when parsing sentences.

Conclusions