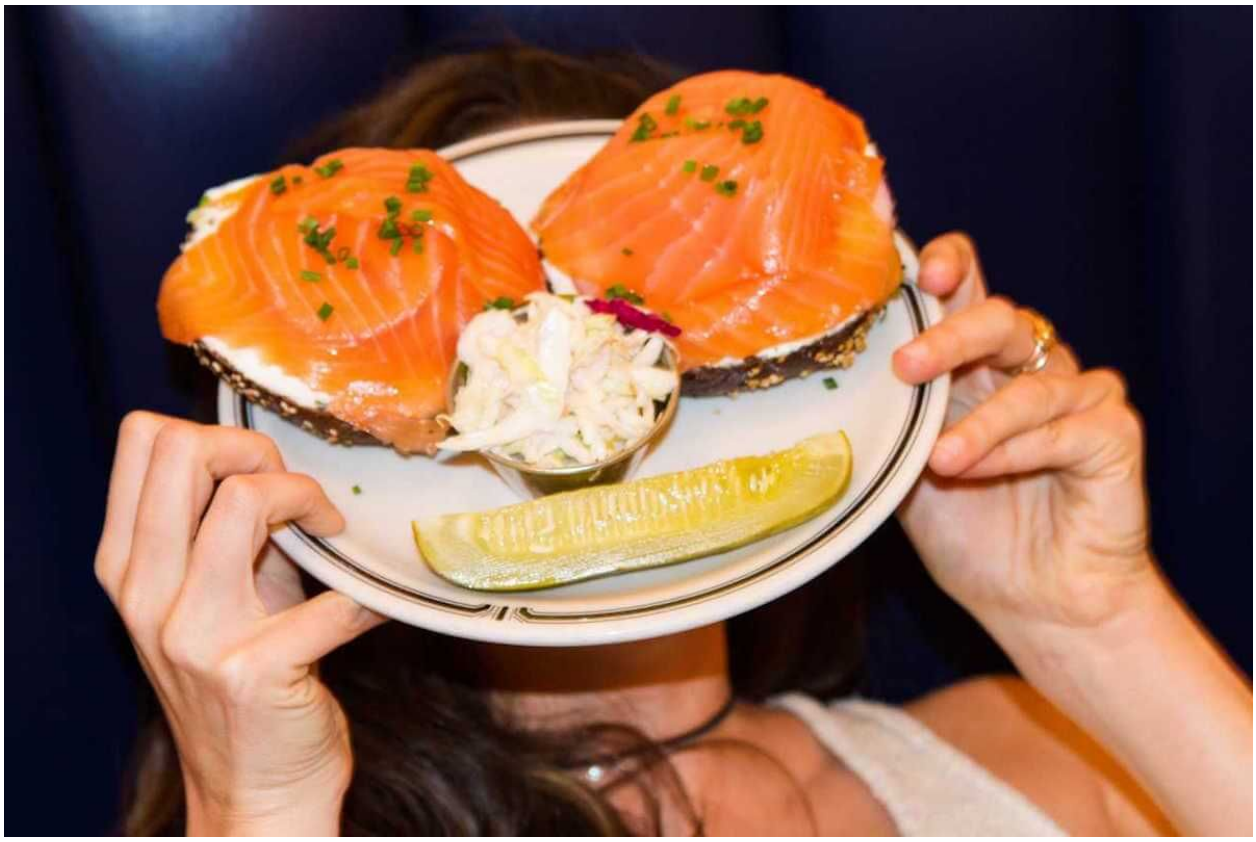
Springboard AI/ML Career Track

Restaurant Rating using Sentiment Analysis



Introduction

Textual data - messages, reviews, articles etc represents unstructured data which represents a great challenge to the NLP domain. In this article we will go through the challenges of interpreting text data from restaurant reviews. We will be using the “yelp dataset”, which is a subset of the yelp business, reviews and user data. We will primarily focus on the reviews. This is a good time to also make the readers aware that we are using pyspark to extract and preprocess the data. Along the course of this article you will notice snippets of code that employs pyspark.

The goal of the model we are building is to rate restaurants based on the reviews

from customers. We will be using sentiment analysis on the text data to assign a value to each of the reviews.

Dataset and goal

The yelp dataset contains a combination of user and business data in json format. Figure.1 shows the breakdown of the review json file, we are interested in the “text“ which basically is the review posted by the user tagged by “review_id”.

Figure.1

```
root
 |-- _corrupt_record: string (nullable = true)
 |-- business_id: string (nullable = true)
 |-- cool: long (nullable = true)
 |-- date: string (nullable = true)
 |-- funny: long (nullable = true)
 |-- review_id: string (nullable = true)
 |-- stars: double (nullable = true)
 |-- text: string (nullable = true)
 |-- useful: long (nullable = true)
 |-- user_id: string (nullable = true)
```

The dataset covers various business types, we are interested in restaurant reviews and this brings us to our first preprocessing step. In order to isolate restaurant data from rest we will use the business json file which has “categories” attribute that can be used to filter restaurant businesses like shown in the below snippet of code.

```
business_df = spark.read.format('json').option("inferSchema", True).load(YELP_BUSINESS_PATH)

restaurant_business = business_df.where('categories like "%Restaurant%"')
```

The next step was to group the reviews associated with a restaurant using the “business_id” attribute which is unique to a business, this allows us to apply our model on individual businesses.

Model

As mentioned earlier in the introduction we want to rate restaurants based on sentiment scores of reviews from customers. We want to provide separate ratings for the food served and customer service. There are two reasons to separate the ratings - 1) having separate ratings is useful for business to help them understand where they are lacking and 2) typically review texts have mixed opinions on both food and service, this could pollute the sentiment score.

In order to achieve this we need to separate sentences that refer to food from service related contexts. Our model achieved this by turning each of the sentences(after sentence tokenization) to

sentence embeddings and applying a clustering algorithm on top of it. The universal sentence encoder(from tensor flow hub) was used to encode the sentences. The idea here is that sentence embeddings with similar context will be closer to each other and we could use the euclidean distance to create two separate clusters. After running experiments between kmeans and agglomerative clustering, the agglomerative clustering was chosen for higher precision. The agglomerative clustering algorithm is a bottom up hierarchical clustering algorithm. This approach yielded a successful result by employing the combination of universal sentence encoder and agglomerative clustering algorithm(with euclidean affinity).

At this point we have classified each of the sentences as either food related or not food related. But before we apply the sentiment analysis on the sentences we grouped the sentences back(still separating food and non-food sentences) to their original review. This was done based on empirical analysis that the sentiment scores were more consistent with the review ratings when they were grouped to their original review.

For sentiment score there were multiple options - NLTK, TextBlob, BERT, Flair. Flair(character-level LSTM model) has the ability that it can predict sentiment for out of vocabulary words. This is quite useful for our project since we are trying to evaluate sentiment of unedited text data from yelp users which is bound to have a lot of typos.

Experimental Results

```
'rice noodle hybrid dish',
'Italian items',
'Bangkok',
'arugula flatbread',
'BBQ burger',
'jammy Buggars',
'entree fish',
'flash frying',
'oily side',
'tender cubes chicken',
'broiler',
'Buffalo Wild Wings',
'flour tortillas',
'Spicy Chicken Sandwich',
'boar pasta w',
'Canadian beef',
'butternut squash',
'amazing flavors',
'Shrimp noodle dish',
'tablespoon scoop',
```

We tried to expand on the results of clustering food and non-food related sentences to perform entity recognition on food related nouns with the goal to capture names of dishes from the menu. Idea was if we can tag nouns in all the sentences we can encode them and apply clustering on top of it like we did for sentences to create distinct groups. But

the result had poor precision as can be seen in the snapshot of the result above. One of the problems that was observed is that the names of dishes varied based on cuisines, for example tagging something like *hot and sour soup* from chinese restaurant menu requires custom noun parsers. Without a custom noun parser we would only capture *soup* from *hot and sour soup.* For example NLTK noun parser with custom format """NP: {(<JJ><CC>)?<JJ>?<NN.*>+}""" had to be used to capture *hot and sour soup*.

We continued to explore other methods of categorizing sentences based on dishes that were referenced. One of the approaches that provided a reasonable result was using category nouns like seafood to identify names of seafood dishes. The steps involved encoding the category name and filter nouns that have favorable euclidean distance.

Deploying strategy

The deployment strategy follows the steps used to deploy on the Paperspace platform on which most of the development and experimentation was performed. In brief the steps involve saving the model, registering the saved model with Paperspace Gradient and then deploying the registered model. The last two steps are done using the gradient cli after publishing the training and inference code in github. To register the model with gradient we run "*gradient experiments run singlenode*" with the path specified to save the model using the script to train the model. The registered model is then deployed using the following two commands - *"gradient deployments create"* && "*gradient deployments start"*. Once the model is deployed we can use the endpoint url generated by the above two steps to access our model using REST api.

Conclusion/Learnings

The experiments in general indicated that hierarchical clustering yielded the best results in creating the necessary distinction between higher dimensional representation of unstructured data such as text data. The universal sentence encoder worked out of the box without any retraining to project the sentences to higher dimensional vector representation.

One thing that remains to be explored are ways to perform NER to detect food items. Since different cuisines have their unique dish names it is challenging to create a generalized NER model to cover all cuisines.

With respect to the development platform Paperspace does provide the flexibility to select the workstation for our experiments but in terms of support and documentation it lacks in comparison to platforms such as GCS or AWS.