Capstone Writeup

Title

Yelp Gross-o-meter

Fangzhao Alex An

Summary

This project predicts and assigns a “cleanliness score” to restaurants in Austin, TX based on the text of their Yelp reviews. The model is trained on restaurant inspection scores.

This project predicts the inspection score of a restaurant with an R^2 value of ~0.35, and suffers mainly from a small dataset.

Problem Statement

As a Yelp user, I want to see a ‘cleanliness’ score that I can associate with each restaurant page that can affect my decision to dine there, but Yelp does not display health inspection scores for many of its businesses. Normally, I can get an impression of the cleanliness by scrolling through reviews to see if there are any big red flags like "bugs in soup" or "unwiped tables.” This project aims to do this automatically, by doing NLP on review text and comparing to health inspection scores.

The end product is a webapp where a user provides the name/address of a restaurant and gets back a predicted score from my model, along with a few statistics and representative reviews that could inform their decision on where to eat.

Project

Data sources

* Austin restaurant Yelp reviews from Yelp academic dataset
* Austin health inspection scores: <https://data.austintexas.gov/Health-and-Community-Services/Food-Establishment-Inspection-Scores/ecmv-9xxi>
* Yelp Fusion API used to connect the two datasets

Gathering and processing the data

Yelp has inspection scores for some of the restaurants, so I tried scraping Yelp with the Python libraries requests and BeautifulSoup4, but this got me IP banned for a month after scraping only ~50 pages. So instead, I took data available from the city of Austin for the health inspection scores.

I limited the Yelp academic dataset to only restaurants located in Austin, TX. To match the restaurants between the two datasets, I queried the Yelp Fusion API using the name and address of each restaurant in the health inspection dataset. In all cases, the name/address was sufficient for the API to return only one result, from which I took the primary key for the Yelp review dataset.

This left me with a dataset of 570614 reviews for 2506 restaurants with 6617 total inspections.

Naive model - fully supervised

Initially, I tried a fully supervised bag of words approach by assigning the inspection score to *each* review. I cleaned the review text with NLP techniques (removing stop words, removing non-English reviews, lemmatization) and vectorized the text (CountVectorizer, TfidfTransformer) using monograms and bigrams. Then, I tried training a variety of estimators (trying both regression and rounding the scores for classification) like linear models, random forests, logistic regression, etc. but none did well on separate test data. (R^2 < 0.1 for regression, accuracy < 0.3 for classification).

This naive approach doesn’t work. The human approach of looking for rare “red flags” in reviews like mentions of bugs and dirty tables is more intuitive, and so I decided to split the model into two steps:

1. **Feature engineering/individual review model**: Converts the text in each review into features that can be aggregated.
2. **Aggregate model**: Aggregates features over all reviews of each business. Train a regressor on this to predict inspection scores.

Feature engineering / Individual review model

In order to accurately judge reviews, I need to accurately judge words (ignoring contextual models). This can be done by training a word2vec model, and then aggregating the word vectors into a review vector.

I first tried to train a word2vec model on the entire dataset of 500,000 reviews. But this runs into the same problem with too few reviews discussing cleanliness. I then tried to train a model on reddit comments from a few relevant threads where health inspectors discussed their grossest experiences. This was a very small dataset which also did not work. So, I turned to pre-trained models, using specifically the glove-twitter-200 word2vec model with a >1M large vocabulary trained on >2B tweets. This is easily accessible with gensim.

My initial approach was to take the cosine similarity between each word and a representative word/group of words that indicate dirtiness:

DIRTY\_WORDS = ('disgusting','smelly','rotten','nasty','gross','dirty','undercooked','moldy','puke','sick','unhealthy')

PEST\_WORDS = ('rat','bugs','cockroach','fly','ant','flea','insect','infestation','infest')

But after trying many strategies, it turns out that a simple mean of the individual word vectors for each review gives the best performance.

The output of the feature engineering step was 203 features for each review: 200 components of the glove vector, and three extra features indicating the cosine similarity to DIRTY\_WORDS, similarity to PEST\_WORDS, and the maximum similarity between the two. These features were averaged across all words of each review.

I want to try using pre-trained contextual models like BERT or ELMo but I don’t believe that these will work too well since the signal I am looking for is so specific (cleanliness), and to specialize one of these would also require extensive training.

Aggregate model

In this second step, I again tried several ways to combine the vectors (mean, median, average of 10% ‘dirtiest’ reviews, etc.), but found that an average over all review features for business features gave the best performance. So in the end, the input to my regressor were the 203 features described above (200 from glove, 3 cosine similarity).

I trained a random forest model to predict the cleanliness score using these 203 features, and got an R^2 value of 0.35 with a test-train split. In training this model, I found that it tended to easily overfit the small dataset (2506 restaurants), and would benefit primarily from a much larger dataset.

Deliverables

My github: <https://github.com/alexanphd/yelp-cleaners>

Streamlit dashboard

I collated all of my code into a python file and made a simple streamlit dashboard. This takes a few minutes to start up because it loads in the large glove-200 model.

Graphical user interface, application

Description automatically generated

I have allowed the dashboard to take in a restaurant name and address, and I then query the yelp Fusion API to search for a matching yelp business. To get the reviews, I then match up the business with the Yelp academic dataset and find reviews from there. Unfortunately, this means that any input comes from the dataset that I trained my model on.

The dashboard outputs a predicted inspection score, along with a percentile and a plot of where the restaurant inspection score compares to all predicted inspection scores. I also run the model on all reviews and output the best and worst reviews.

Chart, histogram

Description automatically generated

As an alternative, a user can also put in a raw review string, and the dashboard will spit out an inspection score and a percentile. This does not work well on one review, unfortunately.

Graphical user interface, text, application

Description automatically generated