Predicting the Outcome of Food Re-Inspections in Chicago

Ali Pelczar and Mike Feldman

Abstract

Thousands of food inspections are conducted each year in Chicago. Establishments that fail an inspection must cease operations until they address the violations and pass a re-inspection. We use comment text from the original, failed inspection to build models that predict the outcome of food re-inspections in Chicago in order to allocate inspectors' time more efficiently. A relatively small and noisy dataset and severe class imbalance hamper the effectiveness of prediction. In neural net models, recall rates range from 0.44 to 0.83 and precision ranges from 0.14 to 0.19. These models might benefit from alternative preprocessing methods or from the inclusion of non-text features such as inspection history.

Introduction

The city of Chicago conducts nearly 20,000 food inspections per year, of which 78 percent result in a passing evaluation ("Food Inspections" n.d.). These inspections cover any establishment that handles or serves food, including restaurants, bars, food trucks, bakeries, grocery stores, schools, and more. They are intended to investigate and prevent cases of foodborne illness, which impact an estimated 48 million people in the United States per year ("Burden of Foodborne Illness: Findings" 2018).

An inspection may be conducted for several reasons. Canvass inspections are the most common and are routine, unannounced visits. New businesses that are applying for a food license must request an inspection before they begin operating.

Additionally, an inspection may be conducted in response to a complaint or suspected case of food poisoning. The regulations that an establishment must meet are identical in each case. There are many potential violations, such as improper food storage, insufficient employee training, or evidence of rodents. Minor violations may be corrected at the time of the inspection, but for serious violations or those that cannot be corrected immediately, the establishment must cease operations until the violations have been addressed. At that time, the establishment must request a re-inspection before reopening (“Understand Health Code Requirements for Food Establishments” n.d.).

Using inspection data made public by the city of Chicago, we leveraged the comments noted by the inspector during a failed inspection to build binary classification models that predict the outcome of the resulting re-inspection. The comments include both the text of the violated health code rule as well as direct observations from the inspector, and comments generally range from two or three sentences to a lengthy paragraph. The dataset also includes information about the establishment, including name, location, and type (e.g. restaurant, bakery, bar), as well as the date and type of inspection.

Food inspections take up resources, in particular the time available to the inspectors, and it is wasteful to conduct inspections of an establishment that repeatedly fails. The city could use these results to target their resources more efficiently, such as by requiring that establishments take more time to address the violations if they are unlikely to pass re-inspection. While predictions should not replace the inspections, they may help inspectors prioritize their caseload and flag establishments that are likely to struggle to correct the identified violations.

Related Work

In the early 2010s, researchers began to explore how social media postings could be analyzed to monitor public health phenomena, such as flu or food-poisoning outbreaks (Sadilek, Kautz, and Silenzio 2012; Aramaki, Maskawa, and Morita 2011). From this literature emerged another strain of studies that investigated how restaurant reviews, in particular those on the platform Yelp, could be used to predict results of restaurant health inspections. This literature is largely focused on efficient deployment of resources by local governments, rather than the issue of identifying emerging threats to public health. Given the limited resources of local governments to dispatch inspectors, online reviews are hypothesized to be effective predictors of health inspection results and that they can be used in models to prioritize restaurants for inspection.

The first empirical study that learned a mapping between text in restaurant reviews and hygiene inspection records used data from Seattle (Kang et al. 2013). Using Yelp reviews written between 2006 and 2013 for establishments, they build support vector machine (SVM) models with accuracy ranging from 61 to 81 percent depending on the classification threshold. Their methods have since received serious criticism, including retaining only inspections that resulted in either a perfect or very poor score and using nonrandom sampling methods to undersample the majority class (Altenburger and Ho 2019). Correcting these methodological weaknesses resulted in models in which the Yelp reviews offered no predictive power. Nonetheless, this work inspired similar efforts using Yelp reviews and inspection records in Boston (Wang,

Balasubramani, and Cruz 2017), San Francisco, and New York (Schomberg et al. 2016).

While no study using Yelp review data has been conducted in Chicago, social media and machine learning methods have been leveraged to identify establishments at risk of violating food codes. In 2013, the city launched a website where users could submit complaints of foodborne illness. An algorithm also monitored Twitter for tweets within Chicago that were related to food poisoning; a staff member then followed up with the user to identify the establishment. However, the inspections triggered by these two avenues identified serious violations at the same rate as business-as-usual methods (Harris et al. 2014).

Another, more successful undertaking by the city of Chicago analytics team was using a variety of open data sources to prioritize annual canvass inspections, such as inspection history, recent temperatures, nearby 311 complaints, and the length of time since last inspection (“Food Inspection Forecasting” n.d.). These results are used to establish the order in which establishments are inspected, so that higher-risk establishments receive inspections earlier. When using this approach, 69% of inspections identified critical violations in the first half of the time period study, compared to 55% using normal procedures. While this does not eliminate inspections at low-risk establishments, it can help violations be identified earlier.

Solution and Our Work

Data Preparation

We downloaded data from the city of Chicago open data portal on food inspections, and included all records from the beginning of the dataset in January 2010 through March 31, 2021, resulting in nearly 220,000 inspections. From these we extracted about 41,500 records that were labeled as a re-inspection.

These re-inspections were then matched to the inspection at the same establishment that occurred most recently prior to the re-inspection. Records were dropped if a one-to-one match could not be created and if the inspection and re-inspection were separated by more than six months. We also restricted our data to matches in which the original inspection had a failing outcome. The largest elimination was for original inspections with an outcome of pass with conditions. While this outcome may require a re-inspection, it represents minor violations that were corrected at the time of the inspection, and the establishment is not closed pending a follow up. The policy implications, therefore, are different for these cases. These steps resulted in 37,206 matched re-inspections. An additional 652 records were dropped because the comments field for the failed inspection contained no text.

Next we split this dataset into training and test sets. Because our models use past data to predict future outcomes, this split was not random; before splitting, we sorted the dataset on inspection date to ensure the most recent records appeared in the test set.

We then developed three types of feature encodings from the text of the inspector comments field in the training set. First, we developed bag of words (BOW) encodings using relative word frequencies, with a minimum vocabulary frequency of 1000. We also experimented with a lower minimum frequency. The length of the

document was also used as a feature. We used GLoVE to generate continuous bag of words (CBOW) embeddings. Finally, we used the CountVectorizer function from scikit-learn to extract two-grams, with a minimum frequency of 500, and developed bag of two-grams (BOTG) encodings using the relative two-gram frequencies.

We did not remove stop words or perform stemming during this process. Given that each document is relatively short and the phenomenon we are trying to model could possibly be influenced by small subsections of the document (e.g., a particular rule failure or one aspect of a rule failure), we concluded that removing stop words and stems could have resulted in the deletion of useful information.

Our final step was to split our training set into a training and validation set. Similar to the train/test split, this split was executed to ensure the validation set included the later records. Note that the re-inspection outcomes are even more imbalanced than the inspection outcomes in general; only 8.8% of re-inspections resulted in a failure. We oversampled failures in the training set to account for this.

Goals

Given that a model predicting all inspections to pass would achieve an accuracy of 91%, this is not an effective measure of model performance. Instead, we sought to minimize the tradeoff between precision and recall while maximizing each value.

Benchmark Models

To establish a performance baseline, we first trained models using Naive Bayes, logistic regression, and SVM. As expected, the Naive Bayes models produced mediocre

results. We used a grid search to tune the parameters of the logistic regression and SVM models. The BOW models generally performed the worst, and CBOW models the best, although there were no substantive differences among the model specifications. Tuning the parameters of logistic regression and SVM had little effect; the differences among the performance metrics were at the third decimal point.

Neural Network Models

We developed feed forward neural networks (FFNN) and convolutional neural networks (CNN). While we considered building long short-term memory (LSTM) recurrent neural networks, we decided that an LSTM was likely not appropriate for the task at hand, given that our textual data is not marked by long-range dependencies.

The FFNNs use binary cross entropy loss and the Adam optimizer. Parameter tuning focused on the optimizer learning rate, the number of layers, the dimension of the hidden layers, and the method of addressing class imbalance.

A learning rate 0.001 was ultimately used, as higher learning rates tended to result in the model not learning and simply predicting all inputs to be of the same class. A lower learning rate slowed training without improving model performance. Models include a single hidden layer. Additional layers worsened model performance across all metrics, likely due to overfitting. Both a high (32 or 48) and a very low (8) dimension for the hidden layers had a slight negative impact on performance, while a dimension of 16 appeared to perform best. Using Tanh or ReLU as the hidden layer nonlinearity had little impact.

We also explored the use of weighted loss instead of oversampling the minority class. This would give greater weight to predictions for failed re-inspections. Several weight values between 5 and 15 were considered. Lower weights increased accuracy at the expense of recall and precision, as this produced models closer to a null model of predicting all inspections to pass. Models with higher weights, meanwhile, tended to perform poorly on all metrics. These models were also prone to overfitting after about five epochs. A weight of 10, which generally represents the true class imbalance, produced results most similar to those seen with oversampling. The best FFNN used BOW, one hidden layer with a dimension of 16, and a Tanh nonlinearity.

The CNNs used binary cross entropy loss and the AdamW optimizer. An embedding layer of dimension 3000 was also used, along with two-dimensional convolution filters[1] and one-dimensional max pooling layers (Kim 2014; Trevett 2021). Parameter tuning focused on the optimizer learning rate, the method of addressing class imbalance, the number and size of the filters, the activation function, and the dropout rate.

A learning rate of 0.00005 was found to be optimal, as higher learning rates typically resulted in precision and recall values of less than 0.1 and accuracies of 0.8 to 0.9 after only 2 to 3 epochs. These were clear signs of overfitting because the majority class represents 91 percent of the dataset. Various combinations of filter sizes from 2 to 6 were tried, and ultimately inclusion of filters of size 2, 3, 4, and 5 was found to most

[1] Although our convolutional filters are two-dimensional, the implementation using Pytorch is fundamentally similar to a one-dimensional implementation. We unsqueeze our input tensors so that they have a channel dimension of 1 before entering the convolutional layer. The height of the filter is then a hyperparameter k and the width is the embedding dimension. A one-dimensional implementation would simply not unsqueeze the input tensors; the filter would then have a depth equal to the embedding dimension and width equal to a hyperparameter k. This two-dimensional architecture was informed by Ben Trevett's Convolutional Sentiment Analysis.

improve performance. The optimal number of filters was found to be 10 per filter size (i.e. 40 total). We tried increasing the number of filters up to 250, but found that generally, an increased number of filters was associated with overfitting and reduced performance. We also found that a Leaky ReLU activation function and dropout rate of 0.5 were associated with best performance on the validation set. Note that other activation functions (ReLU, tanh) performed marginally worse, while lower dropout rates were associated with overfitting.

We also attempted an implementation of focal loss to address class imbalance, in lieu of oversampling the minority class. Focal loss was originally introduced as a means to address the extreme foreground-background class imbalance encountered during training of image recognition CNN models (Lin et al., n.d.). It reshapes cross entropy loss by downweighting the loss assigned to well-classified examples and focusing training on a sparse set of 'hard' examples. When using focal loss, we experimented with weights ranging from 0.05 to 0.25. Similar to our results when exploring weighted loss in FFNNs, a weight of 0.1 performed best, but was similar to our results when using binary cross entropy loss and oversampling.

Thus per the above, the best CNN used binary cross entropy loss, a 0.00005 learning rate, 10 filters each of size 3, 4, and 5, a Leaky ReLU activation function, and 0.5 dropout rate.

Results and Analysis

The test set metrics of our best models are presented below for benchmark models (Table 1) and neural net models (Table 2). Overall, we found our models highly

sensitive to overfitting, with a sharp tradeoff between precision and recall. Our best models generally exhibited fairly low precision values of between 0.13 and 0.19 and higher recall values of above 0.45. Note that the convolutional neural net, the model with the highest precision value (0.19), was also the model with the lowest recall value (0.44) among those presented. The highest accuracy level was obtained by our worst model, a support vector machine with 2-gram features, which simply learned to always predict that an establishment would pass reinspection. Most of our attempts to add complexity to our neural net models, which included adding more layers and higher dimensionality, resulted in overfitting, with accuracy levels of greater than 0.8 and near-zero levels of precision and recall.

Table 1. Results from benchmark models

Model Type	Input Type	Parameters	Accuracy	Precision	Recall	F1 Score
Naive Bayes	BOW	N/A	0.36	0.13	0.83	0.23
	CBOW	N/A	0.67	0.17	0.49	0.26
	2-grams	N/A	0.44	0.14	0.78	0.24
Logistic Regression	BOW	L2 penalty; C = 1.0	0.54	0.15	0.64	0.24
	CBOW	L2 penalty; C = 0.7	0.44	0.14	0.78	0.24
	2-grams	L2 penalty; C = 1.0	0.54	0.15	0.64	0.24
Support Vector Machines	BOW	C = 0.3	0.37	0.14	0.83	0.23
	CBOW	C = 0.3	0.45	0.14	0.76	0.24
	2-grams	C = 0.7	0.88	0.00	0.00	0.00

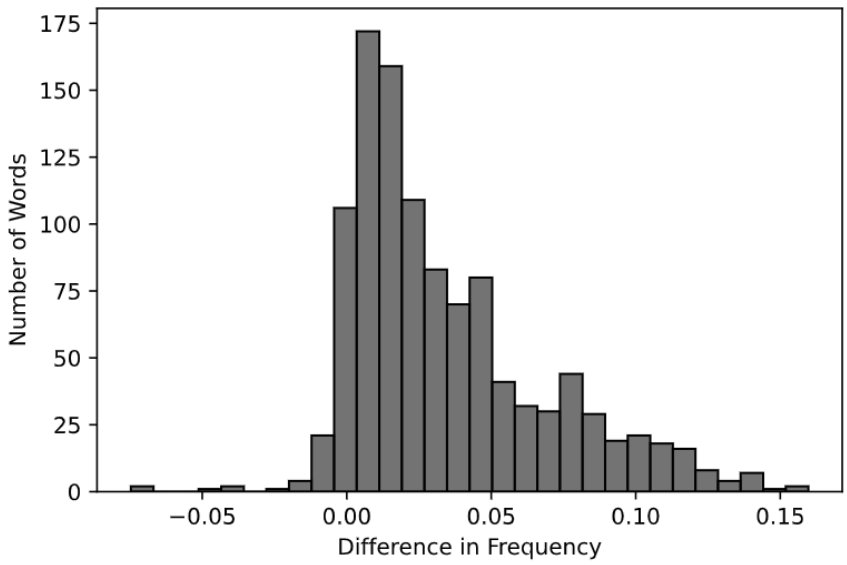
We believe our models' middling results and sensitivity to overfitting derive largely from the limitations of the data. Our dataset is relatively small at approximately 36,000 total observations, and after splitting into train and test sets, our models have fewer than 30,000 observations to train on. This is compounded by the imbalance of the data, as only 9 percent of establishments in the data fail reinspection. This means our models have fewer than 3,000 unique observations from which to learn what encodes reinspection failure. Further, the text for each observation may not impart a significant amount of information, as inspection comments can range from a few sentences to a lengthy paragraph, but typically not longer. We addressed the imbalance issue by attempting both oversampling and weighted loss functions, but this does not fix the underlying sparsity of text data from failed reinspections. Even if there is some signal in inspector comments that can help predict reinspection results, there may not be enough information in our dataset to overcome the noise of a small sample.

Table 2. Results from neural net models

Model Type	Input Type	Parameters	Accuracy	Precision	Recall	F1 Score
Feed-Forward Neural Net	BOW	BCE loss; 0.001 lr; one hidden layer with dimension 16; 10 epochs	0.57	0.16	0.62	0.25
	CBOW		0.37	0.14	0.83	0.23
	2-grams		0.57	0.15	0.55	0.23
Convolutional Neural Network	n/a	BCE loss; 0.00005 lr; 10 filters each of size 2, 3, 4, 5; Leaky ReLU; 0.5 dropout rate; 8 epochs	0.72	0.19	0.44	0.27

The noise becomes clear when we examine the frequency with which BOW vocabulary words appear in observations corresponding to each class. These rates tend to be similar between passed and failed re-inspections. In Figure 1, we present the difference between classes in the proportion of observations in our training set containing at least once instance of each word. A 0 indicates that the word appears with equal frequency in each class and positive values indicate that it appears more often in failed re-inspections. Most words appear at about the same rate and no words appear with notably greater frequency in either class. The words with the greatest magnitude of difference (Table 3) are not, to a human eye, strong indicators of either outcome.

Figure 1. Difference in the appearance of each vocabulary word between failed and passed re-inspections.



This issue likely arises from the similar language used for violations of differing severity. Consider the following two inspection comments, both of which violate the same code:

- 18. no evidence of rodent or insect outer openings protected/rodent proofed, a written log shall be maintained available to the inspectors - comments: must provide pest control log book.
- 18. no evidence of rodent or insect outer openings protected/rodent proofed, a written log shall be maintained available to the inspectors - comments: all necessary control measures shall be used to effectively minimize or eliminate the presence of rodents, roaches, and other vermin and insects on the premises of all food establishments, in food-transporting vehicles, and in vending machines. observed evidence of improper pest control,mice droppings. more than 25 mice droppings behind ice freezer,storage racks. recommend pest service. must clean/sanitize all areas.

It should be clear which of these establishments passed its re-inspection and which did not. Nonetheless, words that we might expect to be strong signals, such as “rodent” or “pest control”, appear in contexts that are quite unequal in severity. This offers another clue as to why our models struggled to deliver strong predictions.

Table 3. Words with the greatest difference in frequency.

Word	Frequency, Failed	Frequency, Passed	Difference
water	0.59	0.45	0.14
hot	0.44	0.31	0.14

facilities	0.43	0.28	0.15
through	0.45	0.29	0.16
low	0.62	0.46	0.16

Suggestions for Future Work

We made a number of decisions early in the process of developing these models, and revisiting them could improve model performance. We retained stop words because we worried that removing them would remove important information. However, they may also increase noise in the model, so a potential next step would be to explore models without them. We kept the legal text of the violations for the same reason, but filtering that portion out and using only the direct observations by the inspector may reduce noise. Using other types of embeddings or longer n-grams is another possibility. We could find clues or ideas on how to improve the preprocessing steps by looking for patterns among the examples that the models tend to predict correctly and examples that the models tend to struggle with. Unfortunately, although our dataset is small by neural net standards, it is still large enough that it would be difficult to sort through manually.

Adding non-text features could also improve model performance. In particular, these models may benefit from the same features that have already been identified as useful during the project conducted by Chicago's analytics team. These features, such as inspection history, time of year, and type of establishment, may supplement these models or even provide greater predictive power than the text. Features specific to

re-inspections, such as the time between the failed inspection and the re-inspection, could also be included.

Description of Effort

The bulk of our efforts were devoted to tuning model parameters in search of the optimal settings. Because there is no previous work that uses analogous data, we spent less time reading research papers, although we did draw some inspiration and lessons from research work using Yelp reviews. We also conducted some background research on how to develop CNNs and which parameters to include because they were not covered as extensively in class or part of a homework assignment.

Much of the data cleaning and preprocessing steps leveraged skills that we already had, whether through prior coursework or early homework assignments. Because we used a separate dataset rather than a dataset included in Pytorch, we learned how to build our own data loader and the associated Dataset class required to be behind a data loader. Building the FFNNs also enabled us to use a codebase developed through homework assignments. Those functions were adapted for binary prediction. We also researched the options for loss functions, and ultimately used binary cross entropy with logits because it has a weight parameter.

Developing the CNNs required the most research and new code. We used legacy Pytorch libraries, which meant altering the data loader setup that we created for FFNNs. These models included more parameters and more options to understand and tune. We adapted our CNN architecture from a tutorial by Ben Trevett (2021). Across all

model types, the focus of our effort was on tuning parameters and iterating on different model specifications rather than writing code.

Ali handled the initial data clean and Mike was responsible for text preprocessing. He built early versions of the basic models. Ali refined them further, built Pytorch data loaders, and built and tuned the FFNNs. Mike completed all the steps necessary for the CNNs. Both team members contributed to the development of the presentations and reports.

Bibliography

Altenburger, Kristen M., and Daniel E. Ho. 2019. “Is Yelp Actually Cleaning Up the Restaurant Industry? A Re-Analysis on the Relative Usefulness of Consumer Reviews.” In *The World Wide Web Conference*, 2543–50. San Francisco CA USA: ACM. https://doi.org/10.1145/3308558.3313683.

Aramaki, Eiji, Sachiko Maskawa, and Mizuki Morita. 2011. “Twitter Catches The Flu: Detecting Influenza Epidemics Using Twitter.” In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 1568–76. Edinburgh, Scotland, UK.: Association for Computational Linguistics. https://www.aclweb.org/anthology/D11-1145.

“Burden of Foodborne Illness: Findings.” 2018. Centers for Disease Control and Prevention. November 19, 2018. https://www.cdc.gov/foodborneburden/2011-foodborne-estimates.html.

“Food Inspection Forecasting.” n.d. City of Chicago. Accessed May 26, 2021. https://chicago.carto.com/builder/c2b6567c-d9b0-4fe6-97cf-f7134baf3fd4/embed.

“Food Inspections.” n.d. Chicago Data Portal. Accessed May 26, 2021. https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5.

Harris, Jenine K., Raed Mansour, Bechara Choucair, Joe Olson, Cory Nissen, and Jay Bhatt. 2014. “Health Department Use of Social Media to Identify Foodborne Illness — Chicago, Illinois, 2013–2014.” *MMWR. Morbidity and Mortality Weekly Report* 63 (32): 681–85.

Kang, Jun Seok, Polina Kuznetsova, Michael Luca, and Yejin Choi. 2013. “Where Not to Eat? Improving Public Policy by Predicting Hygiene Inspections Using Online Reviews.” In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1443–48. Seattle, Washington, USA: Association for Computational Linguistics. https://www.aclweb.org/anthology/D13-1150.

Kim, Yoon. 2014. “Convolutional Neural Networks for Sentence Classification,” September. http://arxiv.org/abs/1408.5882.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. n.d. “Focal Loss for Dense Object Detection,” 9.

Sadilek, Adam, Henry Kautz, and Vincent Silenzio. 2012. “Modeling Spread of Disease from Social Interactions.” In *Proceedings of the International AAAI Conference on Web and Social Media*, 6:8.

Schomberg, John P., Oliver L. Haimson, Gillian R. Hayes, and Hoda Anton-Culver. 2016. “Supplementing Public Health Inspection via Social Media.” *PLOS ONE* 11 (3): e0152117. https://doi.org/10.1371/journal.pone.0152117.

Trevett, Ben. 2021. “Pytorch Sentiment Analysis.” GitHub. March 12, 2021. https://github.com/bentrevett/pytorch-sentiment-analysis.

"Understand Health Code Requirements for Food Establishments." n.d. City of Chicago. Accessed May 26, 2021. https://www.chicago.gov/content/city/en/depts/cdph/provdrs/healthy_restaurants/svcs/understand_healthcoderequirementsforfoodestablishments.html.

Wang, Zhu, Booma Sowkarthiga Balasubramani, and Isabel F. Cruz. 2017. "Predictive Analytics Using Text Classification for Restaurant Inspections." In *Proceedings of the 3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics*, 1–4. Redondo Beach CA USA: ACM. https://doi.org/10.1145/3152178.3152192.