PURDUE

KRANNERT

SCHOOL OF MANAGEMENT

Project Report – Tagging Free and General Items on Craigslist

Prepared for –

MGMT 590, Analyzing Unstructured Data, Spring 2021

Prepared by –

Shah Ishita, Tiwari Diksha, Lyu Xinxin

Yeh Kai-Wei, Mukhopadhyay Sandeep

Table of Contents

Background

Craigslist is a free platform accessible for every user to find any items including cars, furniture, and cellphones. However, Craigslist has been notoriously known for the lack of management especially in categorization for different products. One of the most common issue which could discourage customers from making purchase is that they need to waste a plentiful of time to search for desired products due to the lack of proper categorization or filter out irrelevant items. Although the cheap prices and the diversity in products, customers are generally frustrated when mentioning their purchase experience on Craigslist. With more and more customers moving online since the Covid-19, the management team thus decide to optimize the current categorization especially for top categories like furniture, automobiles, cellphones and appliances to further enhance Craigslist's competency in the industry.

Broad categories like “general” and “free” items have greater number of products which can be classified better under other pre-existing categories to improve customer experience (refer Figure 1 in Appendix)

Project Objectives

Since categorization is the major reason for most customers’ complaints, we decided to come up with an automatic categorization algorithm to help customers classify their items to each of the right category. Instead of searching and finding categories manually, customers could simply follow classification recommendations from the platform when uploading their product information to Craigslist. The new algorithm should be able not only make an accurate classification for each item to the common categories on which this model will be trained but also to identify if a given item does not belong to that broader category.

- Business objective - Improve customer experience by minimizing misclassification of items into broad categories
- Research objective - Correctly classify items from general and free stuff into top categories of furniture, electronics, automobiles, and cellphones

Data analysis

Methodology

The first step we will do is to create a training dataset which would include product information of furniture and other common item such as automobiles, cellphones, and appliances. We will try to access these data in a metropolitan market, such as New York City, to gather larger enough training data size. We will use these respective categories and the associated category labels as our training model with an aim to make correct classifications. Finally, we will test the usefulness of our model with data on more broader categories like general items and free stuffs where categorization would be helpful for customers in getting the right items. (refer Figure 2 in Appendix for overview of the data analysis process)

Data collection

In an effort towards data collection, we have created Spiders using Scrapy for some of these categories. We used the post title and post description as input fields in our model which were later analyzed using NLP analysis techniques (refer Figure 2 in Appendix for sample scraping code)

Input Methods

Next, we decided to use the data in the following methods:

1. Use Title as the sole predictor.
2. Use Text as the sole predictor.
3. Use Title and Text together as predictors.
4. Use Title and Text together as predictors. Additionally, a list of weights was applied to the predictions got by using Title and Text separately. The weights were applied to the Title predicted values while (1 – weight) were applied to the Text predicted values.

The said weights are given below:

```
alpha = [0.52,0.525,0.53,0.54, 0.55, 0.58, 0.60, 0.63]
```

Pre-processing

The initial pre-processing applied was mostly common in all the above-mentioned input methods. The data was imported as a Dataframe using the pandas package. Next the unwanted variables were dropped, and the predictor variable(s) was tokenized using the NLTK package. For example, the Title column was tokenized in the following manner:

For the 3rd and 4th input methods, Text and Title were joined together and put into one list. Thereafter, we used the for loop to apply tokenization on the list.

Next, similar code was used to lemmatize the variables. The lemmatizer was applied to the variables using a for loop. Thereafter, Tf-Idf was used to assign weights to the variables with minimum document frequency of 2. We also removed stop words and single letter words in the preprocessing loop.

Lastly, the training data was split into training data and testing data in the following manner:

The ratio kept for training and testing is 70:30. With this our data was ready for machine learning.

Model training

We used the following models on our training data:

Logistic Regression	Random Forests	Naïve Bayes
Support Vector Machine	Decision Tree	Neural Network

The following results were observed:

Input\Model	Logistic Regression	Random Forests	Naïve Bayes	Support Vector Machine	Decision Tree	Neural Network
Title	0.9471	0.9389	0.9450	0.9455	0.9018	0.8483
Text	0.9384	0.9277	0.9196	0.9374	0.8499	0.2952
Title & Text	0.9545	0.9423	0.9440	**0.9620**	0.9380	0.8585
Title & Text (weighted)	0.9579	0.945	0.945	0.9606	0.9155	0.8677

Support Vector Machine proved to show the greatest accuracy and the Title & Text (unweighted) input method proved to be the best input method.

Validation

Test Data Acquisition & Evaluation

To test the accuracy of our algorithm, we decided to use two datasets from General and Free class items. Different from other predefined classes on Craiglist such as the Furniture dataset we used for training, more diversity of items existed in General and Free class. For example, we can find furniture items, hand-made cookies, and even free soil from a user's garden at the first page in the Free class (see Figure 4). For our classification algorithm, it would be best case to see how our models could handle heterogeneous inputs in the actual world.

We firstly labeled each observation manually in the two classes in order to generate testing benchmarks. There would be five labels we would assign to each observation including the current four and the 'None' category which contain items outside the four categories. For our accuracy test, we would then compare our model prediction results on our test dataset and the actual labels in it. Meanwhile, we filtered out 'None' category during the testing since our machine could not automatically identify items outside the four categories due to limitations of our training datasets. Next, we defined our accuracy score to be the fraction of the occurrence of correct predictions out of the total length of our valid testing dataset.

Testing Result

In the result (see table 5), we achieved the maximum score of 93.59% by the SVM model. The logistic regression model is the second optimist one which achieves an accuracy score of 92.95%. Tree series models, including both decision tree and random forest models, were constant in 80ish % accuracy scores. However, the neural network model failed to demonstrate its potentials with the depressing 69.23% accuracy score. We believed the limited size of the training dataset which only contained roughly 7000 observations.

Insight on the "None" category

We also attempted to develop other algorithms to identify the ''None'' item. The assumption for such an extra algorithm was a unique pattern, such as an even distribution number, in probability distribution as our machines trying to make classification. As a result, we set a cutoff threshold of the maximum probability score of each observation to be 0.25-0.3, which suggested a chance of even distribution. However, we noticed that the machine would force itself to make a classification given the fact the algorithm could not identify the extra category. The randomness existed in the decision making across observations rather than in probability distribution in each data entry. Eventually we failed to achieve a better result but the maximum 0.5 accuracy score.

The attempt was a good lesson for us that we should always predict and test from what we have trained. The machine could not identify any other object beyond the scope while it was not worth it to develop any other methods outside from increasing the training dataset to achieve such a goal.

Conclusions & Future implications

Our model has achieved a promising outcome to identify most of the 4 categories, appliance, automobiles, cellphones, and furniture among messy data we had from Craigslist. Yet, we should still think about to include more data for our training dataset to increase the accuracy of scores of our models and to handle more diversity in data, a situation we could expect in future practices. It could be as well beneficial to include more data besides the 4 major categories we were using to train the model to identify any other

items which do not belong to the four. With a larger and more diversified contents, we could as well adapt more advanced models such as LSTM. However, potential extra costs in the increasing demand of computation powers would be an important tradeoff to evaluate before the Craigslist make a step forward.

Furthermore, Craigslist may also consider conducting an unsupervised learning to optimize its current classifications of items. Currently, there were 45 classes existed on the websites with an excessive amount of redundancy occurring. Craigslist could use the result from the unsupervised classification to see how they could decrease the diversity in classes to enhance users' friendliness through providing a more straight-forward classification rule of each item on shelves.

Appendix

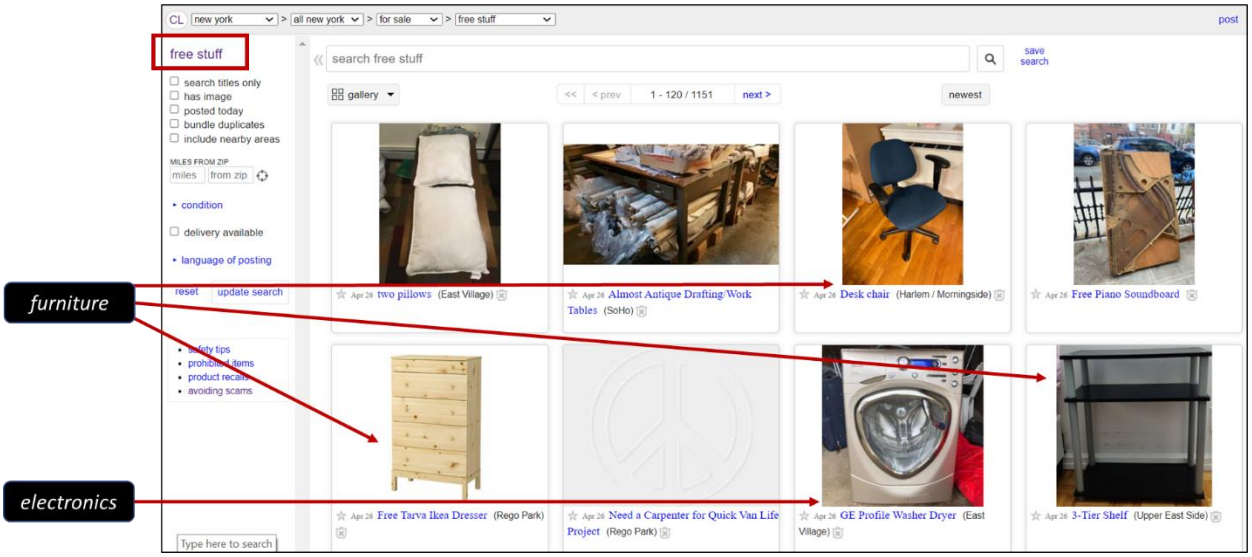

Figure 1 - Background

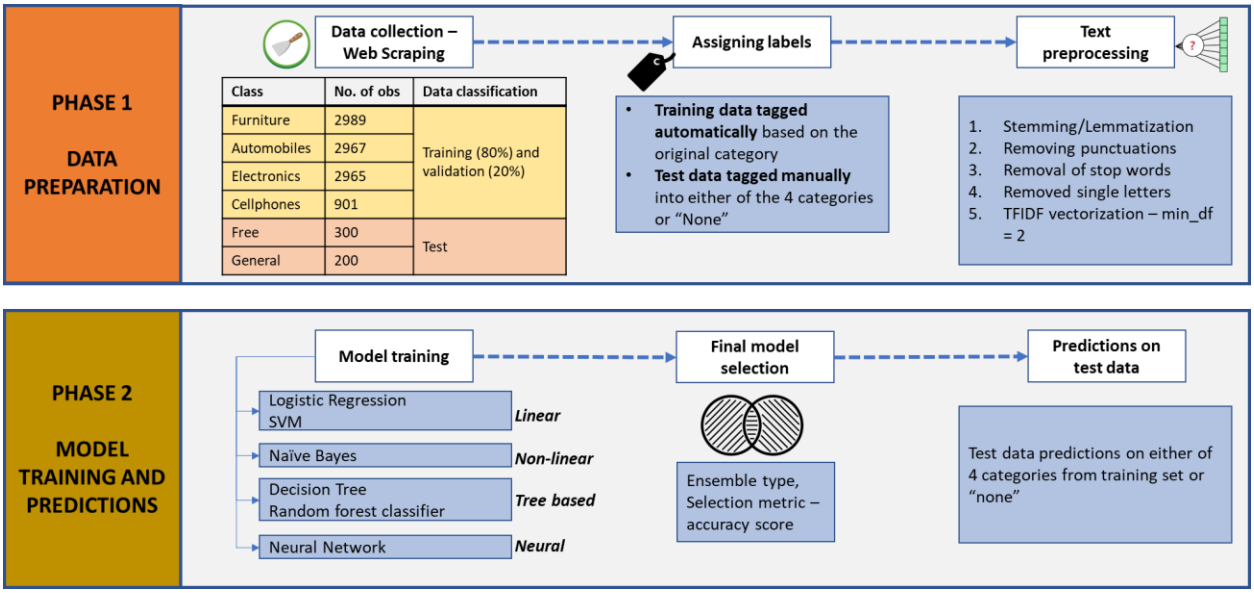

Figure 2 – Data analysis process

furniture.py

```python
import scrapy
from scrapy import Request

class FurnitureSpider(scrapy.Spider):
    name = 'furniture'
    allowed_domains = ['craigslist.org']
    start_urls = ['https://newyork.craigslist.org/d/furniture/search/fua']

    def parse(self, response):
        deals = response.xpath('//div[@class="result-info"]')
        for deal in deals:
            # extract_first, if not, may mess up the results
            title = deal.xpath('h3[@class ="result-heading"]/a[@class="result-title hdrlnk"]/text()').extract_first()
            city = deal.xpath('span[@class ="result-meta"]/span[@class="result-hood"]/text()').extract_first("")[2:-1]
            price = deal.xpath('span[@class ="result-meta"]/span[@class="result-price"]/text()').extract_first("")

            lower_rel_url = deal.xpath('*/a[@class="result-title hdrlnk"]/@href').extract_first()
            lower_url = response.urljoin(lower_rel_url)

            yield Request(lower_url, callback=self.parse_lower, meta={'Title': title, 'City': city, 'Price': price})

        next_rel_url = response.xpath('//a[@class="button next"]/@href').extract_first()
        next_url = response.urljoin(next_rel_url)
        yield Request(next_url, callback=self.parse)

    def parse_lower(self, response):
        text = "".join(line for line in response.xpath('//*[@id="postingbody"]/text()').extract())
        # because there may be multiple texts
        response.meta['Text'] = text
        yield response.meta
```

Figure 3 – Sample code for scraping

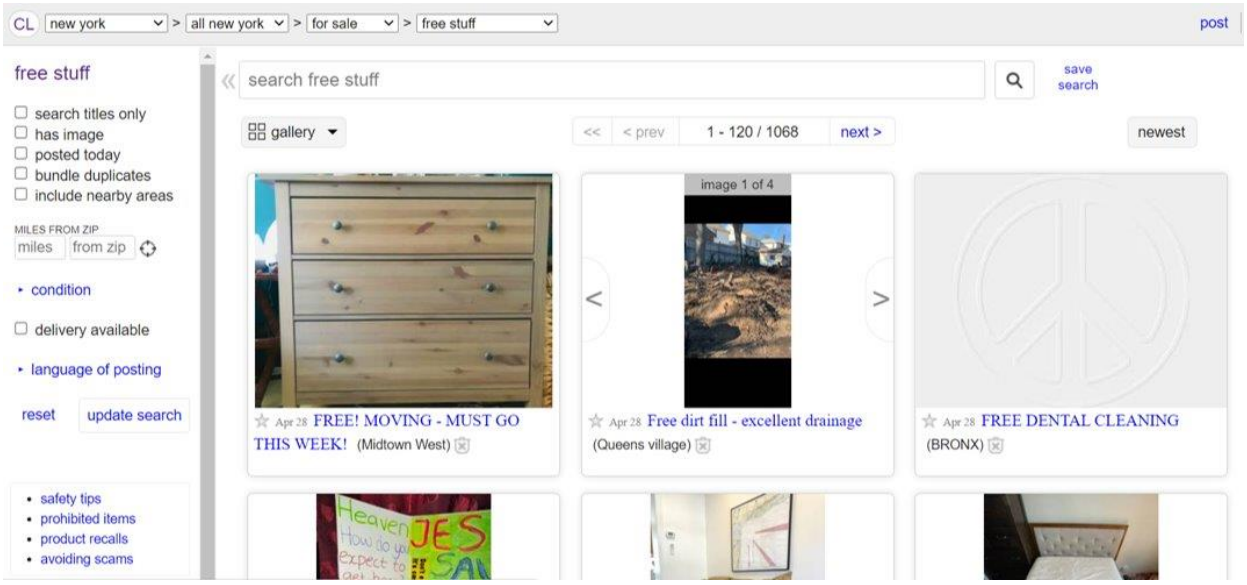

Figure 4 A Screenshot from Craiglist Free Class

Input\Model	Logistic Regression	Random Forests	Naïve Bayes	Support Vector Machine	Decision Tree	Neural Network
Test Result	0.9295	0.8782	0.6154	**0.9359**	0.8141	0.6923

Table 6 Testing Accuracy Scores of the Adjusted Models

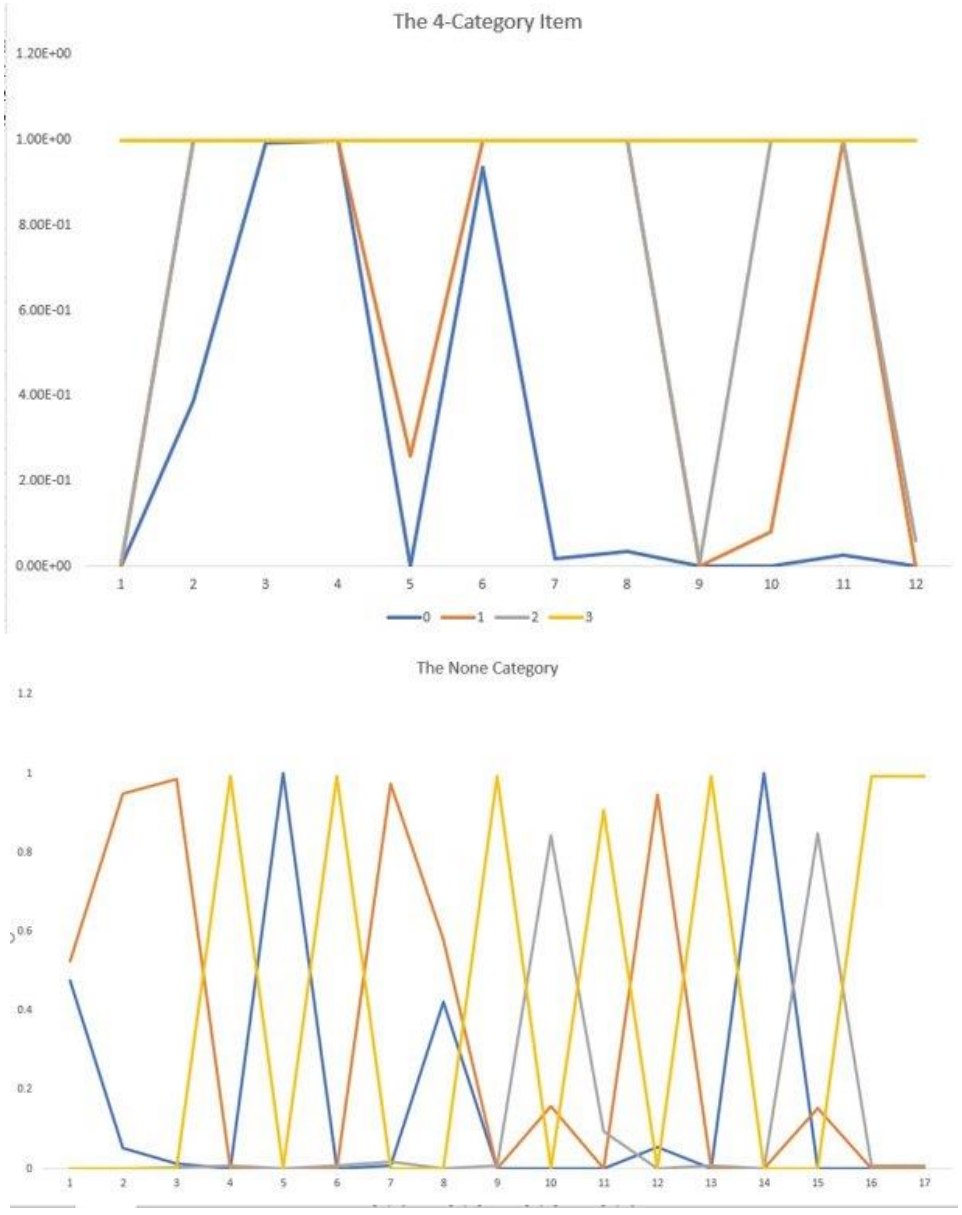

Figure 6. The Prediction Probability Distribution of the None and the 4 Category Item

Input\Mode	Logistic Regression	Random Forests	Naïve Bayes	Support Vector Machine	Decision Tree	Neural Network
Processed	0.4833	0.4567	**0.5**	0.4867	0.4233	0.36

Table 7. Testing Accuracy Scores of the 6 Models