Capstone Project #2: What's for "Free" on Craigslist?

Final Report

Date: November 17, 2017

Introduction and Objective

Craigslist was founded in 1995 and has become one of the most powerful sources for classified advertisements found on the web today. The site is commonly used when searching for new and used products, jobs, and can even be used as a service to meet other people.

One of the most interesting aspects of the Craigslist site is its simplicity in design and user experience. It is fairly simple to set up an advertisement and I have often utilized the site when selling items that I no longer need (typically when I moving to/from apartments). Within minutes a user can list an item for sale.

One of my pain points in the user experience, however, lies in the browse functionality of some of the categories. Oftentimes products are misclassified or the category itself is too broad to be useful. To test this theory this I will explore the "For Sale/Free" category and leverage unsupervised learning techniques to cluster and explore the actual products advertised.

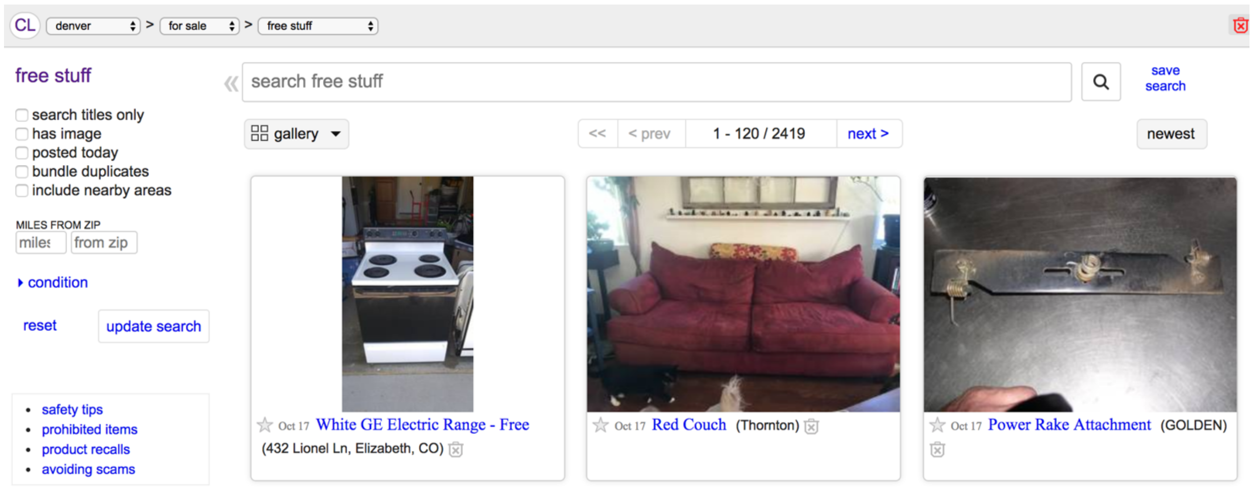


Photo above: screenshot taken of the Denver Craigslist "free stuff" category that shows a variety of products with no ability to filter on a particular category.

Target Audience and Why

My target audience will initially be the savvy online shopper. In my opinion, Craigslist users tend to be deal seekers so having more detailed information on what is offered for "Free" may be of particular interest. This would allow users to understand what is commonly offered for

“Free” and may sway them to consider Craigslist before purchasing something in those categories. E.g. if “moving boxes” are commonly offered for free why would anyone purchase them?

Craigslist has individual sites that are based on metro area since it uses a “meet-in-person” model. Everything is local meaning that no transactions occur online (e.g. exchange of money for shipment of goods). This translates to a localized market of goods and services where offerings may drastically differ across metros. It may also be of interest to compare the findings of the model for Denver metro to another metro in a different area of the country.

Dataset Acquisition

Data was scraped from the Denver, Boulder, Colorado Springs, and Fort Collins Craigslist sites via the Python package “Scrapy” which was initially built to scrape Craigslist sites for job postings and is easily modifiable to handle other Craigslist categories. The four cities represent what I will refer to as the “Greater Denver Metro” area.

Creating a Scrapy spider for the “free stuff” page was an easy task. I simply had to modify the appropriate URL and then find the appropriate HTML tags and variables to modify in order to return the appropriate results.

After scraping a single page of data I needed to modify the script so that I could scrape as many pages of results that exist. In the Greater Denver Metro Craigslist sites this ended at 38 pages and 4,060 results. Each result was returned as a separate CSV file and wrangled via Pandas.

‘Glob’ was used to consolidate the 38 CSV files into a single Pandas data frame.

Dataset #1 (Greater Denver Metro “free stuff”) features only two columns:
1. ‘titles’ – features the text description for each listing title in the “free stuff” category.
2. ‘timestamp’ – features the time and date that the listing was posted to the site.

Initial Dataset:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4060 entries, 0 to 119
Data columns (total 2 columns):
titles       4060 non-null object
timestamp    4060 non-null object
dtypes: object(2)
memory usage: 95.2+ KB
```

Great to see that I have 4,060 records of data but there are a number of steps required to wrangle and cleanse the data before I can put it to good use.

Data Wrangling

Steps required to cleanse and modify the data into a necessary format for analysis.

1. Reset the index of the final data frame since multiple CSV files were loaded with records having the same index numbers.
2. Drop duplicate records which removed 133 records. I removed only duplicates that matched title and timestamp. I also considered removing duplicates where ONLY the title matched but found that these were primarily different listings (e.g. microwave).
3. Convert columns to proper types. 'titles' to string and 'timestamp' to an an actual timestamp object.
4. Convert all words to lowercase for future analysis.
5. Remove all characters and numbers.
6. Remove all stop words. Leverage NLTK 'stopwords' as the basis for the list. Add the word 'free' for this analysis since almost all of the listing titles are for free items. Parse each word in the titles list and if in the stop words list then remove, otherwise keep in the final list of titles to analyze.

Final Dataset:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3927 entries, 0 to 4059
Data columns (total 2 columns):
titles       3927 non-null object
timestamp    3927 non-null datetime64[ns]
dtypes: datetime64[ns](1), object(1)
memory usage: 92.0+ KB
```

Exploratory Data Analysis

Most of the initial data analysis was regarding the shape of the words and their associated counts. More advanced analysis will require unsupervised machine learning models to be created.

Understanding the words found in the postings:

```
count    2585.000000
mean        4.517602
std        13.473507
min         1.000000
25%         1.000000
50%         1.000000
75%         3.000000
max       274.000000
Name: Count, dtype: float64
```

Although we have 3,927 unique listings, the listings only contain 2,585 unique words. The average word is found 4.51 times with a min of 1 count and max of 274 counts! One notable point of caution is the high standard deviation at 13.47, this likely means that my data is skewed in some manner.

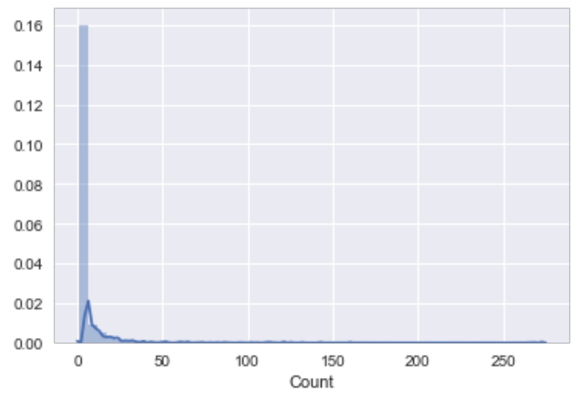
	COUNT	TOP 10 WORDS
	274	wood
	267	couch
	160	tv
	143	boxes
	132	chair
	125	dirt
	121	pallets
	121	curb
	115	moving
	68	firewood

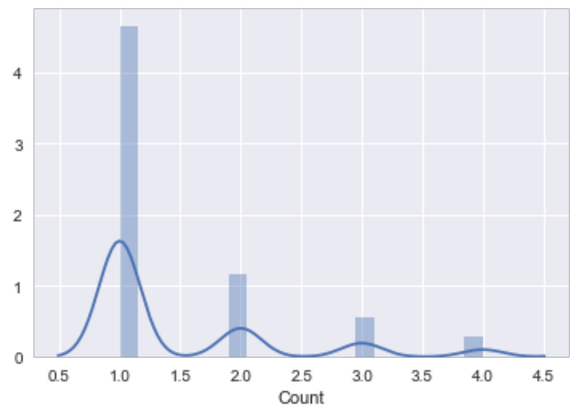
	COUNT	BOTTOM 10 WORDS
	1	ability
	1	newspaper
	1	newer
	1	neutral
	1	network
	1	net
	1	nepro
	1	needing
	1	necklaces
	1	mtb

Charts above: 'wood' was found 274 times while 'ability' was only found 1 time in 3,927 unique listings.

Interesting findings already! It appears that the top item listed is "wood" which I assume to be related to "firewood" (which is the 10th most frequently found word). The words "couch" and "tv" are ranked 2nd and 3rd respectively and make me wonder if I should consider looking on Craigslist for these items before I purchase them next!

It also appears that there are many infrequent words with a count of 1. 57% of the words are found only one time! Analyzing the distribution of the word frequency shows that there are actually many words with a count less than 5.





Charts above: frequency of all words (left) along side the frequency of words found fewer than 5 times (right).

The distribution of the words found less than 5 times highlights that the data is skewed by infrequently found words:

```
count    2122.000000
mean        1.473139
std         0.825960
min         1.000000
25%         1.000000
50%         1.000000
75%         2.000000
max         4.000000
```

Wow, 2,122 of the 2,585 words (~82%) are found less than 5 times in the 3,927 unique listings! The inverse is also quite interesting with 463 of the 2,585 words being found 18.4 times in the 3,927 unique listings.

Machine Learning

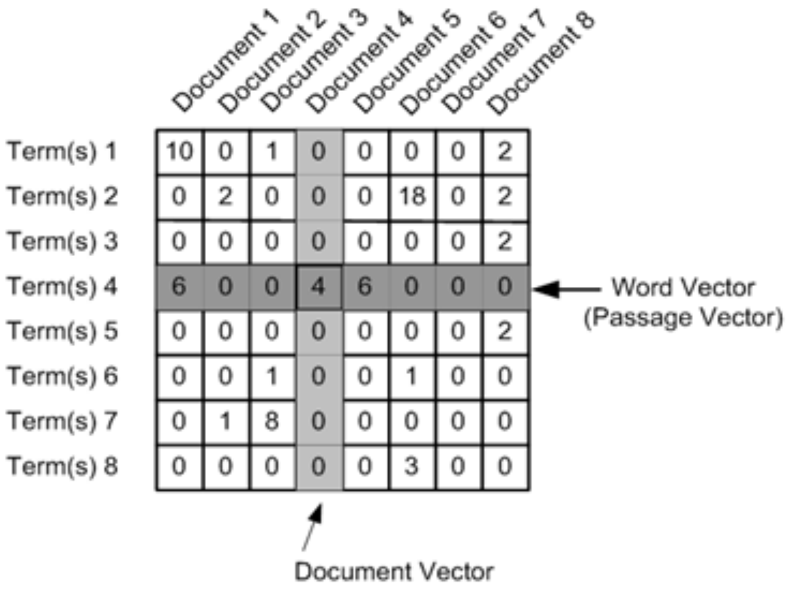
The primary goal of the machine learning (ML) section is to identify the best algorithm to properly classify the "free stuff" listings on Craigslist. This is an unsupervised ML exercise and will focus on testing and understanding key parameters as relates to the exploratory data analysis findings above.

A few comments before diving into the ML model creation. The dataset that acquired is relatively small (3,927 unique records). In addition to the small sample size, it appears that the data is heavily skewed towards infrequent words. This potentially means that the data will need to be transformed before iterating over different parameters in the ML section.

Understanding Text Frequency

Text analysis was performed with the term frequency-inverse document frequency (tf-idf) algorithm and the vectorizers were then transformed to the tf-idf matrix.

Steps to Build the best TF-IDF Matrix:
1. Count word (term) occurrences by document.
2. Transform into a document-term matrix.
3. Apply the term frequency-inverse document frequency weighting.
4. Perform sensitivity analysis on frequency weighting as well as some of the other key parameters.



	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Visual above: each unique term and document has its respective vector in the tf-idf matrix.

A couple of things to note about the key tf-idf matrix parameters:

- Max_df: this is the maximum frequency within the listings a given term can have to be used in the tf-idf matrix. E.g. If the term is in greater than X% of the listings it probably carries little meaning.
- Min_idf: this is the minimum frequency within the listings a given term can have to be used in the tf-idf matrix. This could be an integer (e.g. 5) and the term would have to be in at least 5 of the documents to be considered. If “min_df” was set too low (e.g. 0.0) it ended up clustering on almost irrelevant terms.
- N grams: accounts for term relativity. As an example, setting this value equal to “2” would mean that I consider unigrams and bigrams, e.g. “leather” and “leather couch” which are both of importance.
- Stop Words: this topic was discussed earlier. The only customization made for this model was to include the word “free” in the stop words list as it was found in nearly all listings.
- Sublinear_tf: Setting this value to “true” provides a log10 transformation of the data, which in my case was sorely needed given the high skewness towards infrequent words.

Creating the TF-IDF Vectorizer & Matrix

For the initial ML model I created a Train/Test dataset split of 80% with fixed “random_state” for reproducibility.

The initial model was created without much regard for document frequency and intended to be a baseline for understanding the full tf-idf matrix shape.

Input:
```
vectorizer = TfidfVectorizer(max_df=1.0, min_df=0.0, sublinear_tf=True,
                             ngram_range=(1,3), stop_words=stoplist)
X = vectorizer.fit_transform(train)
X.shape
```

Output:
```
(3141, 10149)
```

To create the initial tf-idf matrix I am using the "train" dataset of 80% of actual data available and creating the baseline matrix. The output shows that the matrix contains roughly 3,141 unique listings with 10,149 terms of those unigrams/bigrams/trigrams found of those 3,141 listings.

Performing a brief sensitivity analysis on the "min_df" parameter shows that it has a *significant* impact on the size of the matrix.

Input:
```
vectorizer = TfidfVectorizer(max_df=1.0, min_df=0.005, sublinear_tf=True,
                             ngram_range=(1,3), stop_words=stoplist)
X = vectorizer.fit_transform(train)
X.shape
```

Output:
```
(3141, 125)
```

Adjusting the "min_df" parameter so that the word must be found in 0.5% of the documents reduced our terms result set from 10,149 to 125! That is a significant difference and once again shows just how skewed the data set is towards infrequent terms.

Clustering

Using the tf-idf matrix, there are a number of clustering algorithms to better understand the hidden structure within the data set. The clustering algorithm of choice for this model was "K-means". K-means initializes with a pre-determined number of clusters. Each observation is assigned to a cluster (cluster assignment) so as to minimize the sum of squares. Next, the mean of the clustered observations is calculated and used as the new cluster centroid. Then, observations are reassigned to clusters and centroids recalculated in an iterative process until the algorithm reaches convergence.

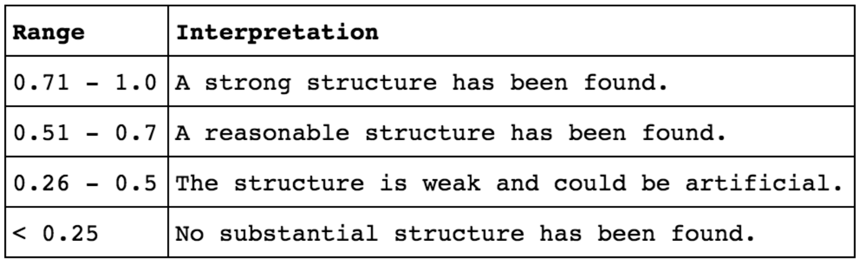
To better understand the best number of clusters ("k") to elect, sensitivity analysis was performed on various cluster values to determine associated model scores.

K-means scores where the initial vectorizer min_df=0.0:

K Value	Sum-of-Squares Score	Silhouette Score
2	-3,068	0.014
3	-3,008	0.027
4	-2,979	0.034
5	-2,963	0.036
6	-2,919	0.043
7	-2,869	0.057
8	-2,856	0.058
9	-2,849	0.057

Chart above: iterating over the number of clusters "k" to determine their associated model scores..

The "Sum-of-Squares Score" shows that the model score improves with each increasing value of "k". The silhouette score is computed on every data point in every cluster. The silhouette score (SS) ranges from -1 (a poor clustering) to +1 (a very dense clustering) with 0 denoting the situation where clusters overlap. Some criteria for the silhouette coefficient is provided in the table below (courtesy of the Springboard curriculum).



Range	Interpretation
0.71 - 1.0	A strong structure has been found.
0.51 - 0.7	A reasonable structure has been found.
0.26 - 0.5	The structure is weak and could be artificial.
< 0.25	No substantial structure has been found.

Chart above: understanding the "silhouette score" ranges and their meaning.

The actual results of the baseline model show that the SS ranges from 0.01 – 0.05 depending on the number of clusters, which means that no substantial structure has been found.

Changing the min_df to 0.005 yields different results:

K Value	Sum-of-Squares Score	Silhouette Score
2	-2,273	0.134
3	-2,201	0.145
4	-2,128	0.163
5	-2,081	0.181
6	-1,981	0.202
7	-1,994	0.191
8	-1,893	0.223
9	-1,861	0.233

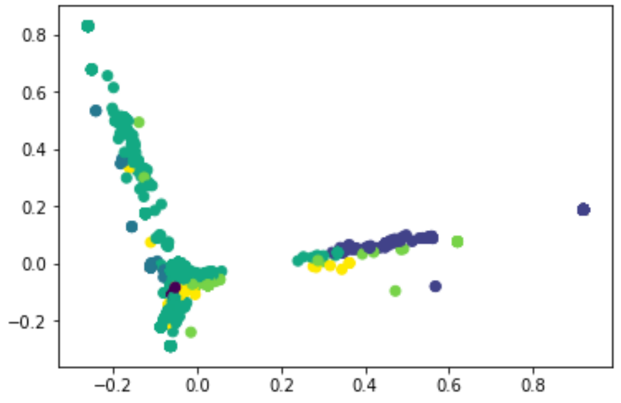
Chart above: increasing the minimum document frequency yields better model scores.

With min_df = 0.005 we can see that the scores improve, however are still fairly weak in their relevance. At this point k=6 was chosen to continue with the modeling results given it has a slight inflection point in the scores. In addition to looking at model scores already let's create a visualization at k=6 to understand if anything is noticeable visually.

Visualizing the Results

Principal components analysis (PCA) was leveraged to reduce the dimensionality and plot the data points in 2-D.

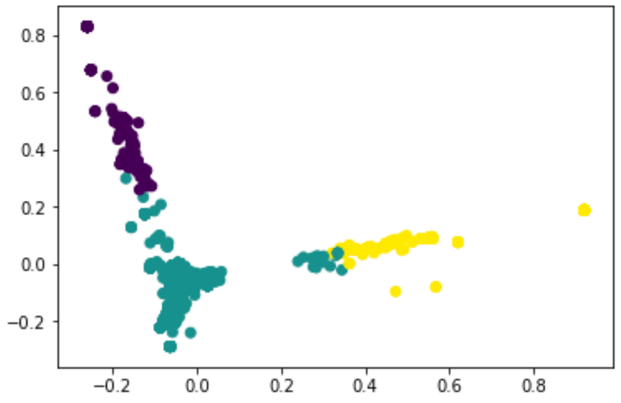
Scatter plot with k=6:



Visual above: 2-D scatter plot of the train data set and k=6 clusters.

Interesting to see the supposed cluster on the right-hand side of the chart with multiple colors of dots next to each other. Clearly the model is having a difficult time differentiating the vectors in this space.

Scatter plot with k=3:



Visual above: 2-D scatter plot of the train data set and k=3 clusters.

At k=3 the clusters visually appear more relevant, however, yellow and green dots are still very close together on the right-hand side of the page. Looking back at the scores for k=3, the SS is 0.145 which means a weak structure.

Finally, let's look at the top words found in each cluster at k=3:

Cluster 1	**Cluster 2**	**Cluster 3**
Wood	Firewood	Couch
Tv	Wood	Leather
Pallets	Pallets	Leather couch
Chair	Today	Loveseat
Boxes	Come	Recliner
Desk	Scrap	Chair
Dirt	Yard	Reclining
Stuff	Small	Sofa
Table	Mulch	Brown
Moving	Pallet	Bed

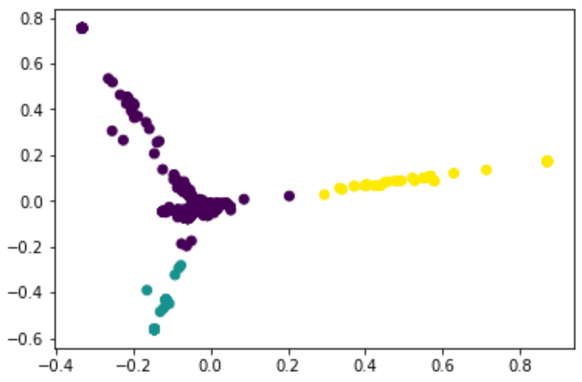
Visual above: 2-D scatter plot of the train data set and k=6 clusters.

A few key observations:

- As expected, some of the words are present across clusters. This was to be expected given the low silhouette scores and the visual of the many different colored dots close together.
- Only one bigram is present, meaning that the non-unigram words were not as important as initially thought.
- An attempt at naming the clusters might be:
  - Cluster 1 would be named "Moving Items".
  - Cluster 2 would be named "Scrap Materials".
  - Cluster 3 would be named "Furniture".

Using the Test Data

Running the same model (k=3, min_df=0.005) shows similar results to the training set. The silhouette score for this model was 0.155.



Visual above: 2-D scatter plot of the test data set and k=3 clusters.

Results Summary

- The data set acquired for the Greater Denver Metro “Free Stuff” analysis was heavily skewed by infrequent terms. Even a log 10 transformation was not able to properly redistribute or normalize the data.
    - This does not mean it is “bad data”, only that the majority of the postings in the “free stuff” section represent very few terms.
- With a Silhouette Score of 0.14 on the training set and 0.15 on the test set, the K-means clustering model provided weak (at best) structure for the designated clusters.
- In this analysis of the “Greater Denver Metro” Craigslist sites, there was an abundance of items in the Craigslist “free stuff” that was classified into three categories:
    - Category 1 is “Moving Items”.
    - Category 2 is “Scrap Materials”.
    - Category 3 is “Furniture”.

Recommendations to the Craigslist Team & Future Improvements

- Craigslist site administrators should create and enable “tags” or “categories” to assist in filtering the “free stuff” section.
- Perform unsupervised learning on the broader, national data set of all Craigslist site for better understanding of what is offered for free across the country.
- There could be a similar analysis done regarding miscategorized or mislabeled listings where the goal would be to find the outlying data points to understand their value.