Craigslist Post Classifier: Identifying the category of a Craigslist post

Vaishak Salin, Vishal Krishna, Dhawal Joharapurkar
Department of Computer Science and Engineering, Manipal University, Manipal, Karnataka
Email:vaishak.salin@learner.manipal.edu, vishal.krishna@learner.manipal.edu, dhawal.manoj@learner.manipal.edu

***Abstract*—Craigslist is a prevalent platform for local classified advertisements. It has nine prominent sections and each of these sections is divided into subsections called categories. For example, the services section has categories such as: beauty, automotive, computer, household, etc. To predict the category under which a Craigslist post should be listed, we pre-processed the heading of the post, and used a numerical statistic method known as term frequency-inverse document frequency (tf-idf) as our weighting factor, which is used in information retrieval and text mining. Then, a bag-of-words model is applied where the occurrence of each word is used as a feature for training our classifier. For the classifier, we used a machine learning technique known as support vector machine, a supervised learning model with associated learning algorithms that analyze data and recognize patterns. Finally, we performed rotation estimation to estimate how accurately our predictive model would generalize to an independent data set.**

I. Introduction

Craigslist is a place with local classifieds and forums which are community moderated, and largely free, on commodities like jobs, housing, goods, services, romance, local activities, advice, etc. There are different sub-domains of Craigslist, for each country they operate in. Each country has listings organized by sections. Under each section, there's further subdivision by categories.

We use supervised learning techniques to classify a Craigslist post into different categories. We use text mining techniques like text normalization[5] and term frequency-inverse document frequency for preparing the data for classification. We have built a classifier using Support Vector Machine using a linear kernel that takes as input the heading, city and section of a given advertisement, to predict the category under which it should be listed[2].

Our classifier has achieved 81% accuracy on the test data. Craigslist could implement a recommender system that predicts the category under which the post must be filed while a user is posting an advertisement, using our classifier. This sort of recommendation system is a very handy tool for an organization like Craigslist which sees an average of 50 billion posting per month as it improves usability and it is very scalable.

II. Data

Our dataset has been obtained from the Craigslist sites for a set of sixteen different cities (such as NewYork, Mumbai, etc.). The dataset has records from four sections 'for-sale', 'housing', 'community' and 'services' and a total of sixteen categories from the above sections. The categories are: activities, appliances, artists, automotive, cell-phones, childcare, general, household-services, housing, photography, real-estate, shared, temporary, therapeutic, video-games and wanted-housing. Each category belongs to only one section. Given the city, section and heading of a Craigslist post, we have to predict the category under which it was posted.

The first line in the dataset is an integer N. N lines follow, each line being a valid JSON object. The following fields of raw data, given in JSON:

- city (string, ASCII) : The city for which this Craigslist post was made.
- section (string, ASCII) : for-sale / housing / community / services.
- heading (string, UTF-8) : The heading of the post.

A sample record looks as follows:

{*"city" : "singapore", "section" : "for-sale", "heading" : "Panasonic ,2doors fridge(238L)($220 with delivery+1mth warranty)"*}

A total of approximately 20,000 records have been provided to us, proportionally represented across these sections, categories and cities. The format of training data is the same as input format but with an additional field category, the category in which the post was made. A separate test dataset of 15,370 records have also been provided to us, on which we have tested our final model.

III. Methods

A. Support Vector Machine

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

In our project, we used a support vector machine equipped with a linear kernel of LinearSVC package of Scikit-Learn which is a python wrapper for the liblinear implemented by Machine Learning Group at National Taiwan University.

scikit-learn[3] (formerly scikits.learn) is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

B. *Bag of Words Model*

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, text is represented as the bag (multi-set) of its words, disregarding grammar and even word order but keeping multiplicity. It is commonly used in methods of document classification, where the frequency of occurrence of each word is used as a feature for training a classifier.

Eg: For the two sentences:
numbr bedroom at eaton place
wanted numbr bedroom apt oct numbr

We construct a dictionary which has ten distinct words.

{*"numbr": 1, "bedroom": 2, "at": 3, "eaton": 4, "place": 5, "wanted": 6, "apt": 7, "oct": 8*}

Using the indexes of the dictionary, each sentence is represented by a 8-entry vector:
[1, 1, 1, 1, 1, 0, 0, 0]
[2, 1, 0, 0, 0, 1, 1, 1]
Where each number in the vector is the frequency of occurrence of the corresponding word in the sentence (histogram representation). Instead of using frequency of occurrence as a weighing factor, it is more common to weigh terms using tf-idf[4].

C. *tf-idf*

Tf-idf[1], short for term frequency-inverse document frequency[1], is a numerical statistic that reflects how important a word is to a document in a collection or corpus and is a common technique used in information retrieval and text mining. The tf-idf[4] value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

IV. Methodology

1) Pre-processing the text data:
   a) Replace special symbols with spaces
   b) Normalize the text. Eg: To treat all numbers the same we find and replace numbers with 'numbr'
2) Compute extra features
   a) Count the number of special characters other than spaces
   b) Count the occurrence of numbers
   c) Count the number of upper case characters
3) Add the features from the dataset
4) Use tf-idf[4] as the weighting scheme
5) Use bag of words to compute feature matrix
6) Apply Support Vector Machine with a linear kernel on the feature matrix
7) Analyze the model

In our project, we use text normalization[5] techniques to cleanse the dataset to apply a classification algorithm, namely, support vector machine to classify each data-point into a category for posting on Craigslist. We remove all the special characters from the data as these aren't really a deciding factor for classification. Since they're inconsequential, we nullify their effect them by removing them. We treat all numbers that occur equally as these usually represent the price or quantity of the product, which doesn't really help us decide the category of a post on Craigslist. We normalize it by using the word 'numbr' for any digit or number that occurs.

We need more features in order to achieve better classification from our learning algorithm and hence we apply the following techniques. As we've removed the special characters, we're losing features. To negate that, we count the number of special characters we've removed and use that as a feature to mark the frequency of occurrence of special characters. We do as above, for numbers. We also count he number of capitalized numbers in the heading of each posting.

After we've extracted features from the text under the heading of each post, we import the other features given to us and form our normalized dataset.

Now, to emphasize certain words, we use a numerical statistic method that reflects how important each word is to a document in a corpus, called the term frequency-inverse document frequency[1]. This method, has two components: term frequency and inverse document frequency[4]. Term frequency, essentially, pegs the raw frequency to the maximum raw frequency of any term in the document.

$$tf(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{\max\{f(w,d) : w \in d\}}$$

Inverse document frequency[1], is a measure of how common or rare the word is, across the corpus. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

TABLE I. Classification Report

classes	precision	recall	f1-score	support
activities	0.76	0.62	0.68	282
appliances	0.74	0.97	0.84	1053
artists	0.85	0.62	0.71	354
automotive	0.86	0.93	0.90	904
cell-phones	0.98	0.91	0.94	1325
childcare	0.80	0.96	0.87	963
general	0.75	0.70	0.72	740
household-services	0.87	0.84	0.85	1043
housing	0.78	0.43	0.55	291
photography	0.96	0.85	0.90	1048
real-estate	0.83	0.76	0.80	758
shared	0.72	0.78	0.75	1801
temporary	0.68	0.69	0.69	1175
therapeutic	0.98	0.99	0.99	1605
video-games	0.97	0.85	0.91	965
wanted-housing	0.84	0.84	0.84	1063
avg / total	0.84	0.84	0.84	15370

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with:

- N: total number of documents in the corpus
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $tf(t, d) \neq 0$. If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the formula to $1 + |\{d \in D : t \in d\}|$.

Then tfidf is calculated as

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

A high weight in tf-idf[4] is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf-idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf-idf[1] closer to 0.

We then use a bag of words model, to vectorize our dataset and prepare it for the application of learning algorithms. We train our model, which is a Support Vector Machine, a Machine Learning algorithm, enabled with a linear kernel to learn to classify future posts based on the training dataset. We then test our model on the testing dataset and obtain an accuracy score, which is an indicative of how well our learning model has scaled to new datapoints.

V. Results

Table I shows the results on the 15,370 test data points. We have precision, recall and f1-scores for each class and the number of records that were classified under each class under support. Precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. The f1-score is computed as follows:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

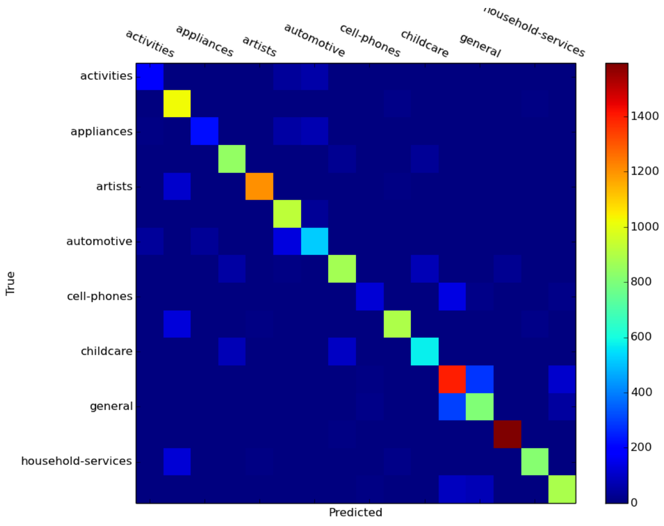

Fig. 1. Confusion matrix

We observe that post under 'cell-phones' and 'therapeutic' categories are classified with high accuracy while postings under 'activities' and 'housing' are misclassified much more than other categories. The model performs considerably well overall as indicated by the confusion matrix shown in Fig 1.

VI. Conclusion

In this project, we used SVM for predicting the categories of a Craigslist post and achieved an accuracy of approximately 81%. Apart from using normal bag of words textual features, we also applied the tf-idf[4] weighting scheme to be able to train a better model. Our results show that it is quite practical for Craigslist to implement a recommender system, which recommends categories under which the post must be filed based on what the user is posting.

In our future work, we would like to extend our text normalization[5] and add more meaningful and context-specific features, especially to improve classification of categories such as activities and housing. We have made an assumption in our project that a advertisements cannot belong to multiple categories, which is not true with real world data. So, we propose to extend this model to be able to predict multiple classes for a single data point.

References

[1] Hinrich Schtze AChristopher D. Manning, Prabhakar Raghavan. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[2] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. *Informatik LS8*, 1998.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Anand Rajaraman and Jeffrey David Ullman. Mining of massive datasets. *Cambridge University Press*, 2011.

[5] A.; Chen S.; Kumar S.; Ostendorfk M.; Richards C. Sproat, R.; Black. Normalization of non-standard words. *Computer Speech and Language*, 2001.