Applications of Machine Learning Techniques to Resale Based e-Commerce Websites

by

Dileep Bodanki

(Under the Direction of Thiab R. Taha)

Abstract

Spam is still a widely prevalent problem in the internet. In this project, we applied supervised learning techniques to grapple with the spam problem for resale-based e-commerce (classified ad) websites such as Craigslist, eBay and Amazon. By leveraging the existence of structured information on these websites, we showed that we can build a better spam detection technique compared to traditional anti-spam techniques that are meant for e-mail and social network ecosystems. After scraping more than 20,000 posts from Craigslist, we tried out various supervised learning algorithms and showed that it is possible to build classifiers that can have up to 98% true positive rate with less than 2% false positives. Moreover, the technique proposed and the features we developed lay the foundation for further work beyond spam detection such as automated tagging for these e-commerce websites.

Index words: Supervised classification, Logistic Regression, Random Forests, Machine Learning, Natural language processing

Applications of Machine Learning Techniques to Resale Based e-Commerce Websites

by

Dileep Bodanki

B.Tech., GITAM University, 2010

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

MASTER OF SCIENCE

Athens, Georgia

2016

Applications of Machine Learning Techniques to Resale Based E-commerce Websites

by

Dileep Bodanki

Acknowledgments

I am greatly thankful to my major advisor, Dr. Thiab R. Taha who helped me throughout the period of my course. He is a wonderful teacher and helped me every step of the way during the completion of my thesis. I like to thank my committee members Dr. Suchendra M. Bhandarkar and Dr. Hamid R. Arabnia for their support. I like to thank my colleagues Phani Vadrevu and Sriram Akella for helping me during this time. Finally, I like to thank my family for their support throughout all these years.

Contents

List of Figures

Chapter 1

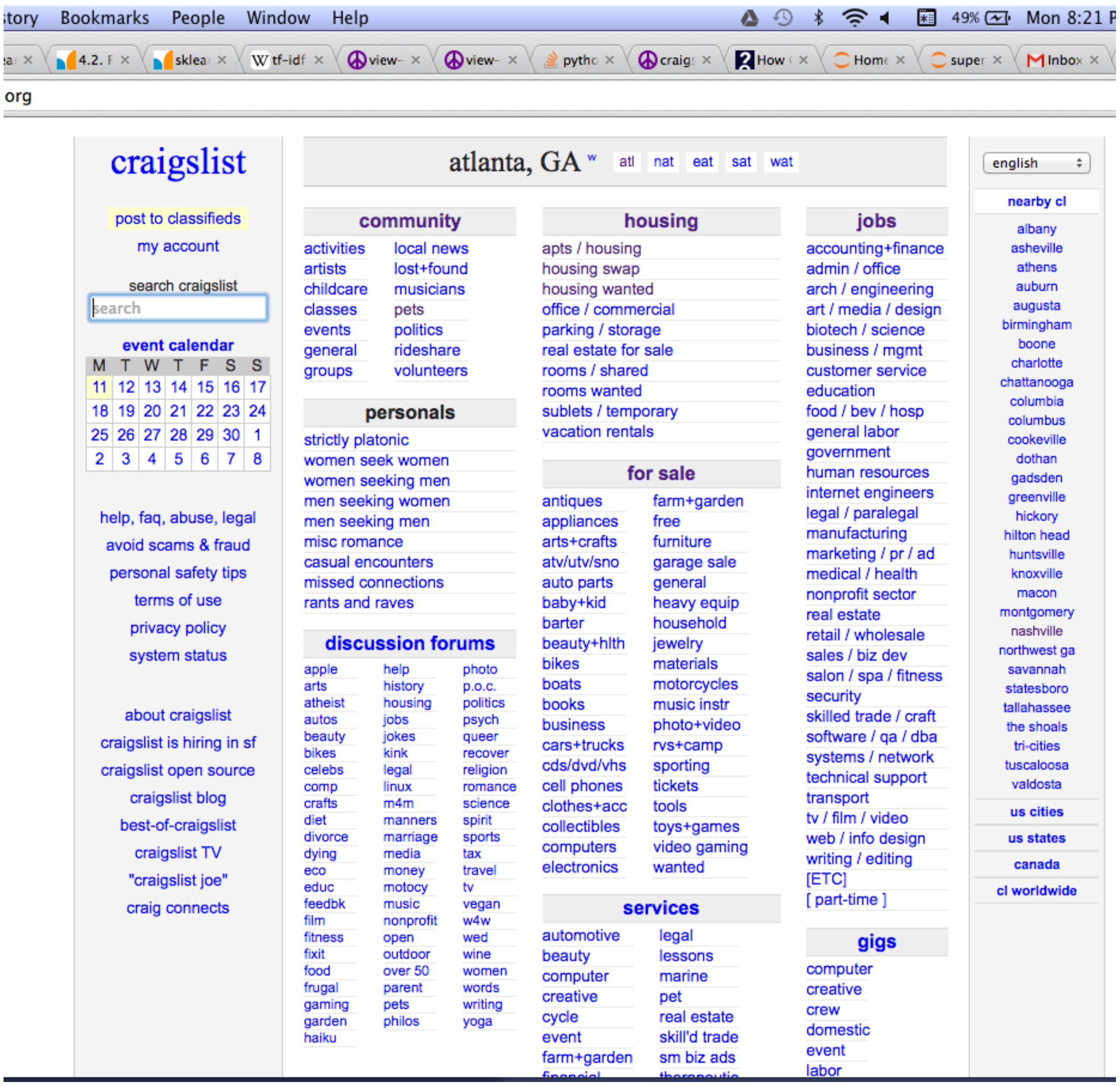
Introduction

Electronic commerce or generally referred to as e-commerce, is the use of electronic medium to carry out sales and purchases. E-commerce has grown vastly in the last decade, nowadays most of the commercial transactions are carried out through e-commerce. The clients of e-commerce are called cyber-consumers. The e-commerce domain which was originally introduced to buy and sell products via internet has extended its reach and in the current day it is used for purposes like renting a house, selling used phones or even purchasing services like nanny for the kids. Day by day more businesses are migrating their operations onto the Internet. In the near future, the lines between ”conventional” and ”electronic” commerce will become increasingly blurred. E-commerce applications are two types: the first one where the sellers are business units and the second one where the sellers are general population. In the first type, the access to the applications is restricted and the applications are well maintained by professionals and there is a very little chance of spam posts, whereas in the second type, where the regular people can sell the products, there is a high chance of posts being spam or the posts not being properly catalogued. It is this problem that we will tackle in our research.

There are a lot of e-commerce websites that allow regular people to sell goods and services. Cragislist, eBay, Amazon and OfferUp are some popular examples. Most people use these websites to sell used goods, advertising real estate rentals and advertise or request services (such as a driver or nanny). These resale based e-commerce websites are also often referred to as classified ad websites. For our research, we have selected Craigslist (http://www.craigslist.com) as the website of our choice for building an automated spam detection system. Craigslist is an online classifieds website divided into different city areas. Users post advertisements in categories including ”For Sale”, ”Services”, ”Housing”, ”Jobs”, ”Community” and ”Personal”. Craigslist divides its ads into several different categories. Anyone can post an ad with or without a Craigslist account. Creating an account allows you to easily access all of your postings to revise or delete them. Without an account, you receive an email after posting with links to modify the ad. User clicks on the ”Post to Classifieds” link at the upper left corner of each page on the Craigslist website to create an ad. Then the user should fill in the template fields including ”Posting Title”, ”Posting Body” and additional details such as ”Price” and ”Condition”. The ad appears in the category on the selected city's Craigslist page. Figure 1.1 shows the various categories under which people can post ads on the Craigslist webpage.

If users are interested in browsing the Craigslist ads, they need to select the closest city on the right side of the Craigslist page. The main page of one's city shows the ads organized under main categories. Subcategories appear below to help users find the specific items they want. For example, the ”For Sale” section includes categories to sell antiques, appliances, books, furniture and jewelry. Users need to click on the desired category to see a list of ads posted from newest to oldest. Users have the option of sorting these ads by price instead of chronological order. They can also see the ads from the selected category represented as expandable info-bubbles on a map based on the buyer's location.

Craigslist is gaining popularity every day because it provides an opportunity for common people to sell their goods and services. This inadvertently creates problems as well. Craigslist



tory Bookmarks People Window Help 49% Mon 8:21 P

4.2. F | sklea | tf-idf | view- | view- | pytho | craigs | How | Home | super | Inbox

org

craigslist

post to classifieds

my account

search craigslist

search

event calendar

M	T	W	T	F	S	S
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

help, faq, abuse, legal
avoid scams & fraud
personal safety tips
terms of use
privacy policy
system status

about craigslist
craigslist is hiring in sf
craigslist open source
craigslist blog
best-of-craigslist
craigslist TV
"craigslist joe"
craig connects

atlanta, GA w atl nat eat sat wat

community

activities, artists, childcare, classes, events, general, groups, local news, lost+found, musicians, pets, politics, rideshare, volunteers

personals

strictly platonic, women seek women, women seeking men, men seeking women, men seeking men, misc romance, casual encounters, missed connections, rants and raves

discussion forums

apple, arts, atheist, autos, beauty, bikes, celebs, comp, crafts, diet, divorce, dying, eco, educ, feedbk, film, fitness, fixit, food, frugal, gaming, garden, haiku, help, history, housing, jobs, jokes, kink, legal, linux, m4m, manners, marriage, media, money, motocy, music, nonprofit, open, outdoor, over 50, parent, pets, philos, photo, p.o.c., politics, psych, queer, recover, religion, romance, science, spirit, sports, tax, travel, tv, vegan, w4w, wed, wine, women, words, writing, yoga

housing

apts / housing, housing swap, housing wanted, office / commercial, parking / storage, real estate for sale, rooms / shared, rooms wanted, sublets / temporary, vacation rentals

for sale

antiques, appliances, arts+crafts, atv/utv/sno, auto parts, baby+kid, barter, beauty+hlth, bikes, boats, books, business, cars+trucks, cds/dvd/vhs, cell phones, clothes+acc, collectibles, computers, electronics, farm+garden, free, furniture, garage sale, general, heavy equip, household, jewelry, materials, motorcycles, music instr, photo+video, rvs+camp, sporting, tickets, tools, toys+games, video gaming, wanted

services

automotive, beauty, computer, creative, cycle, event, farm+garden, financial, legal, lessons, marine, pet, real estate, skill'd trade, sm biz ads, therapeutic

jobs

accounting+finance, admin / office, arch / engineering, art / media / design, biotech / science, business / mgmt, customer service, education, food / bev / hosp, general labor, government, human resources, internet engineers, legal / paralegal, manufacturing, marketing / pr / ad, medical / health, nonprofit sector, real estate, retail / wholesale, sales / biz dev, salon / spa / fitness, security, skilled trade / craft, software / qa / dba, systems / network, technical support, transport, tv / film / video, web / info design, writing / editing, [ETC], [part-time]

gigs

computer, creative, crew, domestic, event, labor

english

nearby cl

albany, asheville, athens, auburn, augusta, birmingham, boone, charlotte, chattanooga, columbia, columbus, cookeville, dothan, gadsden, greenville, hickory, hilton head, huntsville, knoxville, macon, montgomery, nashville, northwest ga, savannah, statesboro, tallahassee, the shoals, tri-cities, tuscaloosa, valdosta

us cities

us states

canada

cl worldwide

Figure 1.1: Craigslist homepage showing different categories.

allows user to post ads without even requiring a user account creation. This makes Craigslist a very popular target for spammers. Also, since everybody is not a savvy computer user, the cyber-consumers are bound to make mistakes thus resulting in their ads to fall in the wrong categories. This eventually makes their ads invisible to the intended users. Such wrongly posted ads along with spam posts cause other users to spend more time looking at the wrong ads. The success of applications like Craigslist depends on how well users can navigate and find the things they are looking for. If this objective is hindered, it will eventually result in the failure of the business. Further, spam posts tend to have malicious content which can harm the website's naive users, alienating them from the website. In order to prevent all of this, we propose to use supervised learning techniques to train different classifiers which can effectively differentiate between these spam posts and benign posts and alert the website staff.

We organize this thesis as follows. In Chapter 2, we give a background of spam and spam detection research work, discuss the intuition and motivate the need for our project. In Chapter 3, we give an architectural overview of our work. In Chapter 4, we discuss the details of various modules that are part of our system's architecture. Our work has two main technical components: data scraping and supervised learning. We discuss both of these components in detail in Chapter 5. Chapter 6 clarifies the entire end-to-end process with the help of a case study. Evaluation setup and the experimental results are discussed in Chapter 7. In Chapter 8, we describe how our work makes a unique contribution in the light of closely related research work in this domain. In Chapter 9, we discuss directions in which our work can be extended in the future. We conclude this thesis with Chapter 10. Appendix A contains some code samples from our project code.

Chapter 2

Motivation and Background

Unsolicited messages sent using electronic messaging systems are referred to as spam. These messages are commonly related to advertising and are sent out on a mass scale. Spam has been a major problem since the internet era had begun. Spam is very prevalent in a number of media over the Internet: emails, instant messaging systems (such as Yahoo chat, AOL, GTalk), social networks (such as Facebook and Twiiter), online classified ads, and web blogs. Nowadays, spam has spread to channels beyond internet. Mobile phone messaging spam (SMS spam) and VOIP spam using robot calls are few examples of this. All media which allow users to send messages without a significant cost per message are open to spam attacks.

In 2001, The Internal Market Commission appointed by the European Union (E.U.) determined [2] that spam emails cost Internet users about 10 billion Euros every year all over the world. In 2007, the California government found [1] that spam costs U.S. companies more than $ 13 billion dollars every year. There are a number of ways in which spam causes damage. For example, it wastes human productivity, as reading through the spam posts always results in a direct loss of valuable work time. Also, computer and network resources are wasted for processing these spam messages. Cases in which the user falls for one of

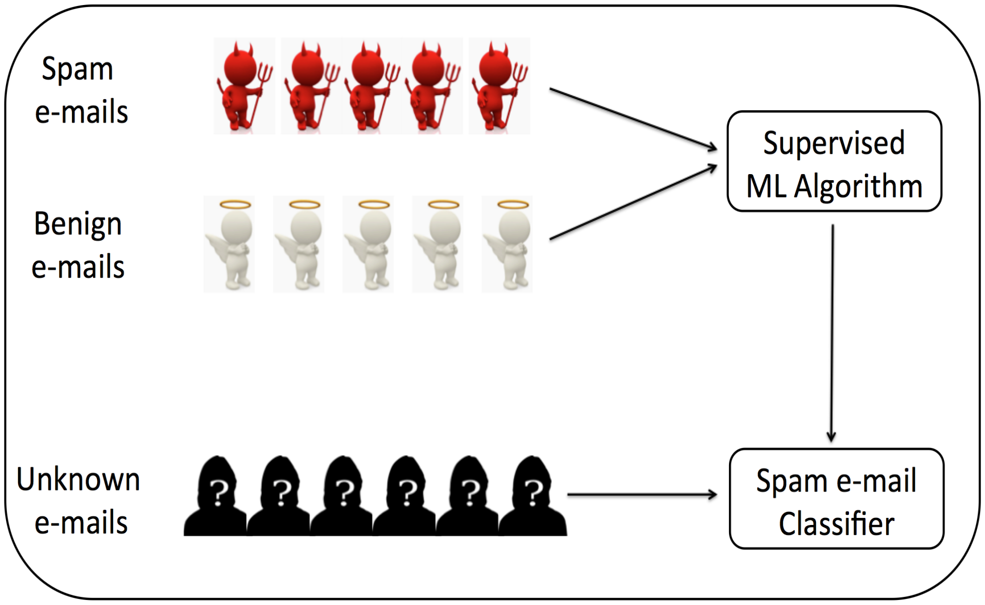


Figure 2.1: A traditional supervised learning based spam-detection system

the tricks employed by the spammers and buys their advertised fake products or software, it would often result in direct monetary damage for the user. If the spam message lures the user into downloading malicious software, then identity theft, financial and intellectual property theft, espionage are all possible. Spammers often make use of the fact that the cost of sending messages in this modern world is very low. Researchers have found that even though their conversion rates (i.e. the percentages of users who click on a spam ad) are as low as 1 in 12,000,000 [15], spammers still seem to make a profit out of it.

Over the years, security researchers have developed a number of techniques to detect spam. [14], [24] and [11] are few examples of the many research works done in this domain. The research work spans multiple categories such as e-mails, social networks and web blogs. Nevertheless, not much work was done specializing in the domain of detecting spam posts on resale based e-commerce websites. These works will be discussed in detail in Chapter 8.

One thing that is common to most of these systems is the presence of a supervised learning system. Consider the example of a spam email detection system. Figure 2.1 depicts how such a traditional supervised learning based spam-detection system works. As with all supervised learning systems, there are two phases: the first is a training phase in which the system learns to differentiate between spam and benign emails and the second is a deployment phase in which the system is put into use to detect spam emails. For the training phase, a significant number of spam and benign email samples are collected and are fed to a pre-configured supervised learning algorithm. The output of this phase, as can be seen in the Figure 2.1 is a classifier model. This model can then be used in the deployment phase. Any unknown email can then be classified as spam or benign by making use of the classifier model. But, note that the benign emails fed to the algorithm are often a mixed variety consisting of various kinds of emails. For example, one might have personal emails from friends and family, work related emails coming from colleagues, promotional emails like coupons from websites that the user has signed up for, feed containing snippets of information from websites such as Twitter, Quora and Facebook etc. The categories of emails mentioned here are all unique in their own way and the structure and the content of these emails looks drastically different from one another. As a result, feeding them all as one ”benign” category of emails to the supervised learning algorithm limits the learning capacity of the resulting classifier. Further worsening the results is the fact that even the spam posts are not all of one kind, spam posts tend to cover many different topics such as illegal pharmaceuticals, pornography, fraud and cheap retail products, phishing attempts etc.

Our intuition behind this work lies in the fact that e-commerce websites afford something to spam detection that other spam attractors don't: structured data. For example, as we discussed in Chapter 1, how the data in e-commerce websites is divided into various categories. The Figure 1.1 also shows how Craigslist website maintains more than 100 categories for all its content. Because of the presence of all this structured data, we can do something different from a traditional spam detection system such as the one discussed

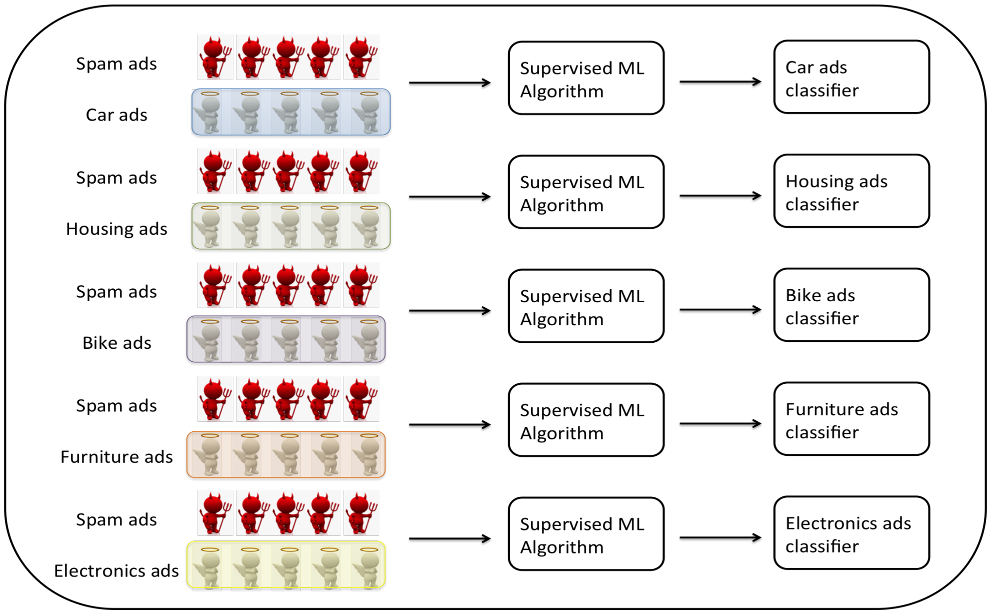


Figure 2.2: Our proposed spam-detection system for classified ad websites

above. Figure 2.2 shows how this can be done. Instead of having one classifier that can differentiate between all benign and spam messages, we instead opt for training several classifiers one for each category of posts. For example, the "Car" category will have its own classifier different from the "Housing" category's classifier. This way all the data that is fed to the algorithm when training the classifiers would be homogenous. Our main idea is that it should be very easy to train a classifier to identify one specific category of posts at a time, instead of training one classifier that can tell if a post belongs to a multitude of benign categories or it is a spam. Our experiments reveal that this is in fact the case. By using multiple natural language processing techniques, we have been able to build classifiers that can accurately ($> 90\%$ accuracy) identify the category to which a given post belongs.

.

Chapter 3

Architectural Overview

The spam filtering system as explained before is a supervised learning system. As with any supervised learning system, it has two separate modules: a training module and a testing/deployment module. However, the two modules share a lot of the architectural components. Hence, we try to describe them together here. Also, our spam filtering system is built outside the web servers, and includes a web crawler that is capable of periodically crawling the websites. While it is sometimes useful to have a spam filtering system deployed inside the web server, there are some compelling reasons to use a crawler based spam filtering system instead of deploying one inside a web server such as:

- A crawler based system can be made more generic. We built our crawler using the popular Scrapy project that makes it easily adaptable to other websites that we have not yet evaluated.

- Access to the back end database is not needed. This enables independent security researchers to use our system on a variety of e-commerce websites to study patterns of spam.

Figure 3.1 depicts the various architectural components of the system. As can be seen

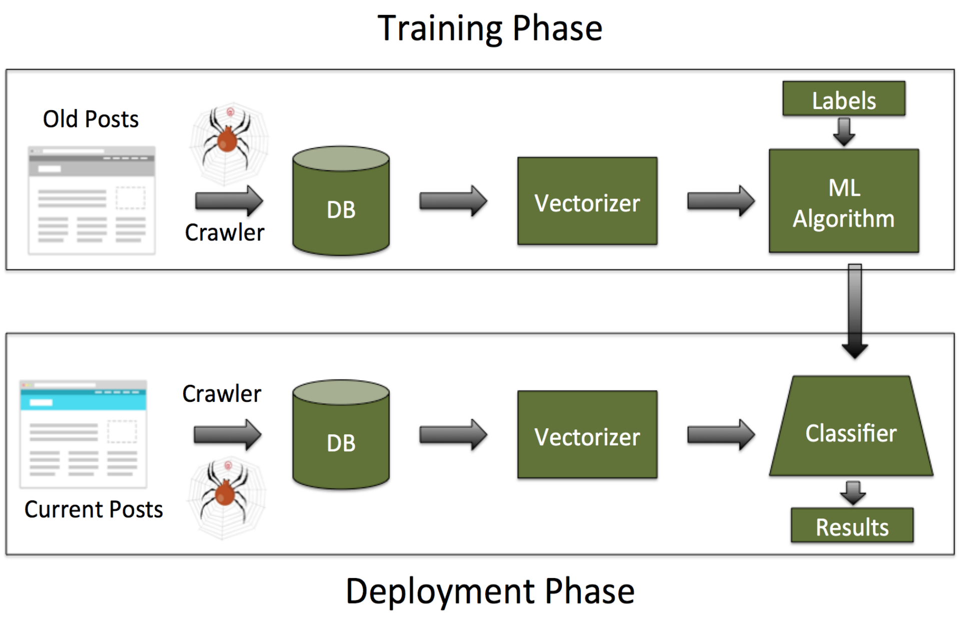


Figure 3.1: Architecture Overview

in the Figure 3.1, there is a crawler which can periodically crawl the website for data and convert the data into a structured format. The data is then stored in the backend database. This process repeats until the end of the training period. At the the end of the training period, the stored text data is fed to the vectorizer module which converts it into a machine understandable format. For all the data collected during the training phase, labels should be collected accordingly indicating which data is benign and which of it is spam. The labels along with the data from the vectorizer are then sent to a machine learning algorithm which outputs a classifier. This concludes the training phase.

The classifier from the training phase is placed on the deployment module. In the deployment phase, all the data being posted on the websites is sent to the classifier after being passed through the crawler, database and the vectorizer. If the classifier detects any spam posts then they are marked as spam and stored in another database table.

Chapter 4

Architecture Details

In this chapter, we present the details about each of the system modules that were mentioned in Chapter 3.

4.1 Website

The system we built is intended to be used for e-commerce websites. We picked Craigslist (http://www.craigslist.com) as the website of our choice as it is widely popular and also representative of the kind of portals that would be suitable for the proposed study. Also, the popularity would mean, a lot of data would be available which is necessary in order to train the classifiers properly. Craigslist contains advertisements posted by people at no cost. There are various categories such as ”housing”, ”for sale”, ”community”, ”jobs”, etc.

4.2 Crawler

The crawler is the module that scrapes the websites and returns structured data from them. It needs to do this on a periodic basis. We built the crawler using the Scrapy open source project in Python. We chose to use Scrapy because it enables building of crawlers that can be easily adapted to other websites as well. The crawler collects the following data fields: "title of advertisement", "body of the advertisement". We chose the "housing" and "for sale" sections of Craigslist for reasons that we will explain later.

4.3 Database

We chose MongoDB [4] as the back end database to store the data. MongoDB is an open source cross platform NoSQL database. However, using a NoSQL database is not a hard set requirement and the backend database can easily be switched to a relational SQL based databases such as ORACLE, MySQL, or PostgreSQL without much change to the code. But MongoDB is very suitable for storing document oriented data which makes it very suitable for our work. In order for the rest of the code to communicate with the MongoDB database, we have used the default Python driver called PyMongo.

4.4 Learning Module

The learning module is the heart of the system. We used the Scikit Learn open source Python library for this module. Specifically, we tried out various supervised learning algorithms such as random forests, SVM and Logistic regression. After comparing the results, we figured that the logistic regression made the best sense in terms of both accuracy and simplicity.

After getting a good number of posts for both positive and negative classes, we fed this data to a TF-IDF based vectorizer [21, 10]. We used N-grams [5] as part of the vectorization process in order to provide more contextual information. The TF-IDF based vectorizer and N-grams will be discussed in detail in Section 5.2.1. Also, the standard stop words for English language that were available as part of the Scikit Learn library were used. The vectorizer object was also stored to be later used in the deployment phase. The vectorized training data along with the class labels were then fed to the classifier in order to obtain a trained model.

4.5 Classifier

The trained classifier can be used for quickly classifying posts as spam or not. In order to do this we need to vectorize the text based data that is scraped. This is done using the stored vectorizer object that was saved during the learning phase. The results from the classifier are then stored in a separated database table.

Chapter 5

System Details

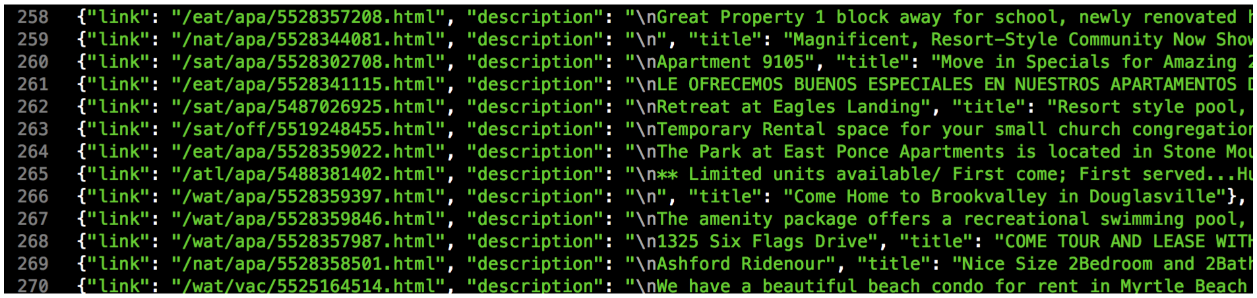
Our project involves a typical supervised learning workflow. Like any other such workflow, it can be divided into two parts:

- Data scraping
- Supervised learning

In the rest of this chapter, we will explain in detail about these two parts of our work.

5.1 Data Scraping

The data scraping part of our work mainly consists of the crawler that we already discussed in Chapter 4. We coded this part using the Python language. In this part of the work, we have also used Scrapy [8], which is an open source framework for extracting data from the web. Scrapy's modular architecture allows us to easily adapt this code for other websites as well. For this, scrapy uses independent pieces of code that are called spiders. Though we use only one spider for crawling Craigslist, other spiders can be added quickly in the future without affecting any of the existing codes.



```
{"link": "/eat/apa/5528357208.html", "description": "\nGreat Property 1 block away for school, newly renovated
{"link": "/nat/apa/5528344081.html", "description": "\n", "title": "Magnificent, Resort-Style Community Now Sho
{"link": "/sat/apa/5528302708.html", "description": "\nApartment 9105", "title": "Move in Specials for Amazing
{"link": "/eat/apa/5528341115.html", "description": "\nLE OFRECEMOS BUENOS ESPECIALES EN NUESTROS APARTAMENTOS
{"link": "/sat/apa/5487026925.html", "description": "\nRetreat at Eagles Landing", "title": "Resort style pool,
{"link": "/sat/off/5519248455.html", "description": "\nTemporary Rental space for your small church congregatio
{"link": "/eat/apa/5528359022.html", "description": "\nThe Park at East Ponce Apartments is located in Stone Mo
{"link": "/atl/apa/5488381402.html", "description": "\n** Limited units available/ First come; First served...H
{"link": "/wat/apa/5528359397.html", "description": "\n", "title": "Come Home to Brookvalley in Douglasville"},
{"link": "/wat/apa/5528359846.html", "description": "\nThe amenity package offers a recreational swimming pool,
{"link": "/wat/apa/5528357987.html", "description": "\n1325 Six Flags Drive", "title": "COME TOUR AND LEASE WIT
{"link": "/nat/apa/5528358501.html", "description": "\nAshford Ridenour", "title": "Nice Size 2Bedroom and 2Bat
{"link": "/wat/vac/5525164514.html", "description": "\nWe have a beautiful beach condo for rent in Myrtle Beach
```

Figure 5.1: Few scraped posts in JSON format

We have converted all this scraped data into a JavaScript Object Notation (JSON) format which is an open-standard format [12] that has key-value pairs. Figure 5.1 shows a screenshot of a few of the posts that have been scraped. MongoDB, being a document-oriented database, was easily able to save all this JSON data without having to do any more conversions. In total, we had collected about 20,000 posts from our target website. More details on the kind of posts we have collected will be discussed in Chapter 7.

5.2 Supervised Learning

This is the machine learning part of the project. Our current problem is to be able to determine whether a user submitted post is spam or not. This is a typical classification problem in machine learning where one class is spam (positive class) and the other is benign (negative class). Our goal is to be able to build a system that can determine if any user submitted post is spam or not.

One way to solve this problem is by using a supervised learning approach which is what we have used in our project. In this approach, there exists a learning phase during which our system "learns" about the characteristics of spam and benign posts after being provided with some labeled posts (i.e. some spam and benign posts as examples). Using these samples, the system will be able to learn how the posts look like in a particular category. In reality, this learning phase consists of building a statistical model which is simply a 1D array of fitted parameters. Once this model is built, we would be able to classify new posts into either spam or benign. Our supervised learning work can be further divided into two parts:

- Vectorization
- Classification

5.2.1 Vectorization

Vectorization is the process of converting human readable text into machine understandable arrays of numbers. This step is vital to all natural language processing (NLP) applications like ours. This involves a lot of steps which are standard part of NLP applications [20]. These steps will be described below. Note that vectorization is common to both the learning phase as well as the deployment phase.

Preprocessing

The goal of the preprocessing step is to remove all noise from the input text that might otherwise negatively affect our ability to differentiate between the positive and negative classes.

Stop Words One way to do this is to remove all words referred to as the ”stop words” in the language. For example, in the English language words such as ”is”, ”and”, ”are” are very common words that are present in all texts regardless of the context. We cannot differentiate between spam and benign posts based on the difference in relative frequencies of these words. Hence, it would be better to remove these words to keep them from confusing our classifiers. We do this by using ”stop words” list that comes with the Scikit library for the English language.

Cut-off Cut-off specifies a fraction such that if the given word is present for less than that much proportion of documents (called document frequency) then that word is ignored. Sometimes, a whole number signifying an absolute count is used as a cut-off value instead. Cut-off ensures that the number of distinct words that are passed on to the classifier becomes much less and only the important words are passed on to the classifier. Unimportant words that exist only rarely do not add any value to our discerning ability. We used a cut-off value of 2

Maximum Document Frequency Similar to cut-off, if a word is present in all the documents irrespective of its class, then it doesn’t make much sense to pass it to the classifier. So, we set a threshold value for the maximum document frequency. We used a maximum document frequency of 80%. Given that we maintained an equal proportion of negative and positive samples, it is reasonable to assume that any word that is present in more than 80% of all posts does not add much value to our classifier.

Tokenization

Depending on the NLP problem that we are dealing with, there might be classes of words that are not relevant to the solution we are seeking. In our case, posts might contain words depicting prices like ”$ 100”, or phone numbers like ”678-315-0163”. These might not add much value to deciding whether or not a post is a relevant to the category or a spam post. In order to filter out all these, we use a regular expression that is matched against all the words. All the non-matching words are filtered out in this phase. The regular experssion we have used is mentioned in Appendix A.2

N-grams

Sometimes, the presence or absence of a single word is not enough. It would be practical to have more context. For example, the word ”not bad” has very different meaning from the word ”bad”. In order to show this difference, it is customary to use N-grams. This means that all contiguous word combinations in the text from 1 to N words are used as tokens. This is better illustrated in the case study in Chapter 6. There is always trade-off between processing speed versus the improvement in results when choosing a suitable value for N. Based on a small sample set, we conducted a few simple experiments and chose the value of N to be 3.

TF-IDF Vectorization

Finally, we need to convert the obtained tokens into arrays of numbers. The simplest way to do this is by taking counts or proportion of each token in the document (called term frequency) and using them. This is a simple approach and is called a count vectorizer. But, this will unnecessarily give excess importance to tokens (higher value) that might be present in documents of all classes. This is similar to the problem we had when we discussed the ”Maximum document frequency” previously. As a solution, we normalize these term frequency values by using the inverse document frequencies. This way we give more weightage to those tokens that have moderate document frequencies. This approach is called the Term Frequency - Inverse Document Frequency (TF-IDF) vectorization.

Here, we provide the formal definition for TF-IDF metric. Assume that the post (or document) we are interested in is d and token (or term) that we are interested in is t. The term frequency $\mathrm{TF}(t, d)$ is simply the frequency of t in the document d (denoted by $f_{t,d}$).

$$\mathrm{TF}(t, d) = f_{t,d}$$

Inverse document frequency is a function of the term across the entire document set. It denotes how rare this term is across all documents (denoted by set D). This is done by logarithmically scaling the inverse fraction of posts (or documents) that contain the relevant term. Hence the term ”Inverse document frequency”. In other words, we divide the number of documents that contain the term by the total number of documents, take the inverse of it and then use its logarithm.

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

Notice that when the term is present in all documents, it will have an IDF of 0 whereas when it is present in only a minor fraction of documents it will have a high IDF. We thus use IDF to to qualify (mutiply) the value provided by the term frequency. Some words such as ”buy”, ”sell” etc might present in all the posts regardless of the category. Hence, they will add little value to the analysis. Using IDF will help us to decrease value for such tokens.

$$\text{TF-IDF}(t, d) = \mathrm{TF}(t, d)\,.\,\mathrm{IDF}(t, D)$$

We have used this TF-IDF method as part of our work. At the end of all these stages, we will be able to transform the set of posts that we scraped from the classifieds website to a matrix of numbers that can then be fed to a classification algorithm. All the steps that we have mentioned in this preprocessing subsection were performed using Scikit Learn, which is a machine learning library for Python. In particular, we have used its TF-IDF Vectorizer class [7] for all of the preprocessing steps mentioned here. All the parameter that we have used are mentioned in Appendix A.2

5.2.2 Classification

Once we transform the dataset into a vectorized format, we were able to test it against many of the supervised learning algorithms that have been proposed. Names of all the algorithms that we have tried out have been listed in Section 7.2. But before vectorization or the classification process begins, the dataset should first be divided into a train set and a test set. We split this into a specific ratio maintaining equal number of spam and benign posts as mentioned in Section 7.2. Once the train and tests are obtained, the data from the train set is vectorized and the vectorizing model is also saved. Using the vectorized train set, all the algorithms we considered were used to build different models. We have used logistic regression, support vector machines (SVM with a linear kernel) and random forests algorithms. All of these were implemented in the Scikit Learn machine learning library [6]. Sample source codes depicting how the various models were trained and tested are in Appendices A.3, A.4. After building the training models, we used the vectorizer model on the text data from the test set to vectorize it. We tried out this vectorized test set on different trained models and the results were reported.

Chapter 6

Case Study

In this chapter, we give a complete walk through of how user posts are classified with the aid of two examples. We consider a user classifier that is trained to recognize user posts related to "Housing" posts in this study. This is the same classifier that we later use to quantitatively evaluate our system in Chapter 7. Specifically, we use the logistic regression classifier for this. Consider the following posts:

- **Benign post**: "Young professional seeking room with garage parking for $400/month, utilities included!"
- **Spam post**: "Buy Premium insurance for your car or bike for just $10 a month!! Hurry!"

6.1 Tokenization

The first step is preprocessing and tokenization as described in Chapter 5.2.1. The tokens produced for the posts are as follows:

- **Benign post**: ['young', 'professional', 'seeking', 'room', 'garage', 'parking', 'month', 'utilities', 'included', 'young professional', 'professional seeking', 'seeking room', 'room garage', 'garage parking', 'parking month', 'month utilities', 'utilities included', 'young professional seeking', 'professional seeking room', 'seeking room garage', 'room garage parking', 'garage parking month', 'parking month utilities', 'month utilities included']

- **Spam post**: ['buy', 'premium', 'insurance', 'car', 'bike', 'just', 'month', 'hurry', 'buy premium', 'premium insurance', 'insurance car', 'car bike', 'bike just', 'just month', 'month hurry', 'buy premium insurance', 'premium insurance car', 'insurance car bike', 'car bike just', 'bike just month', 'just month hurry']

We can see in both posts how special characters such as '$' and the digits are removed due to the regular expression filtering. Also, we can see how words such as "for", "your", "with" are filtered out as part of the English stop words. Next, we can also see that each token consists of up to 3 words. For example, there are tokens like: "parking month utilities", "premium insurance car" etc. This is part of the 3-gram generation process.

6.2 Vectorization

The next step is to convert the two token lists into a number array format using the vectorizer model which is one of the outputs from the learning stage. This outputs two vectors with as many as 356,073 elements in each one. The high number of elements is due to the usage of 3-grams and large number of training samples. But, most of the elements in these two vectors are zeros. The number of non-zero elements is only 24 for the benign post and 21 for the spam post. That is why such vectors are referred to as sparse vectors.

6.3 Classification

We can then feed the two sparse vectors to the classifier (that was trained to recognize "Housing" posts as already mentioned). After this is done, the benign post gives out a score of 0.93. This indicates that there is a 93% chance that this post correctly belongs to the "Housing" category. On the other hand, the classifier gives a score of 0.13 for the spam post. This shows that the classifier only has 13% confidence that this is a legal post. In other words, it has 87% confidence that this post is spam. This is how the classifiers we have trained can recognize between genuine posts and spam posts. Because of the huge separation in the confidence scores for the two classes, even a naive threshold value like 0.5 would do the job. In practice, we can choose a threshold on a quantitative basis, by comparing the true and false positive values and various thresholds. This is done for a sample algorithm in Chapter 7.2.

Chapter 7

Experiments

In this chapter, we discuss the experiments that we have conducted to measure the performance of the system qualitatively. We have tried out many different machine learning algorithms and measured their accuracy.

7.1 Experimental Dataset

We collected a total of approximately 20,000 posts under the categories apartments, rent, rooms wanted, bikes, cars, motorcycles, boats and RV's. We then divided the posts into two categories: training-data and testing-data. We used an algorithm to split the posts in the ratio 4:1 so that 80% of the posts fall under training-data and 20% of them fall under testing-data.

In order to train the classifier, we needed both spam and non-spam posts. As it is very hard to collect good number of spam posts, we instead chose to simulate the spam posts. In order to do this, we considered all posts from "housing" section as negative examples. For simulating positive examples, we chose posts from the "for sale" section. As these posts are widely different from the posts of the "housing" section, we reason that this might be a good set to simulate spam posts.

7.2 Experimental Results

We have used different supervised learning algorithms. We used logistic regression and support vector machines (using linear kernels). We have also used random forests (with 10 and 100 trees) as they have proven to be successful in previous research [13]

In machine learning, results are evaluated by using counts of true and false positive. A false positive in our case, would be a benign post that has be wrongly classified as a spam post. If a spam post is correctly classified as a spam post, then, its a true positive. It is customary in literature to compare various supervised learning techniques using measures like false positive rate (fall-out), true positive rate (recall) and accuracy. These are defined as follows:

$$\text{False positive rate (fall-out)} = \frac{\text{\# benign posts classified as spam}}{\text{\# benign posts}}$$

$$\text{True positive rate (recall or detection rate)} = \frac{\text{\# spam posts classified as spam}}{\text{\# spam posts}}$$

$$\text{Accuracy} = \frac{\text{\# posts correctly classified as spam/benign}}{\text{\# posts}}$$

We have computed these measures for all the algorithms that we have tried and listed them in Table 7.1. Based on our computation, the 4 classifiers perform very well in terms of these 3 measures. In particular, the Logistic Regression and SVM algorithms perform really well in classifying more than 99% of the samples correctly. The classifiers actually do not directly classify the samples. Instead they give a value between 0 and 1 that determines the probability of a sample belonging to the positive class (i.e. spam). Then, a preset threshold value is used to determine which class the sample should belong to. This is determined suitably by the machine learning library (scikit-learn, in our case). Usually, this value is simply 0.5. While this value may work many times, that is not always the case. Some times, we might be sensitive to false positives. In those cases we use a higher threshold. In other cases, we might not want to miss any spam posts at all. In those cases it would be preferable to have a lower threshold value.

So, just considering the accuracy, true and false positive rates at a particular preset value might not be the best way to compare different algorithms. In order to make a fair comparison among different supervised classification techniques, we need to be able to measure the performance of the spam classifiers at all the threshold values. For this, we use the Receiver Operating Characteristic (ROC) curve that has been shown in Figure 7.1. This is a curve obtained by plotting false positive rate (on Y-axis) and true positive rate (on X-axis) at different thresholds. Therefore, the curves on the top can be considered to be performing better when compared to those at the bottom. In order to make a fair comparison at all threshold values, we may consider the area under this curve (AUC). The closer this value is to 1, the better the performance of the algorithm is, at all the threshold values. We have also mentioned these AUC values in Table 7.1. Again, from the table, we can see all the algorithms perform really well with SVM being the best performing algorithm with an AUC of 0.9882.

Table 7.1: Table showing classification results

Algorithm	True Positive Rate	False Positive Rate	Accuracy	AUC
Logistic regression	99.13%	1.45%	93.47%	0.9880
SVM (linear kernel)	99.05%	1.59%	94.09%	0.9882
Random forests (10 trees)	96.89%	5.18%	92.56%	0.9825
Random forests (100 trees)	98.17%	3.03%	93.36%	0.9862

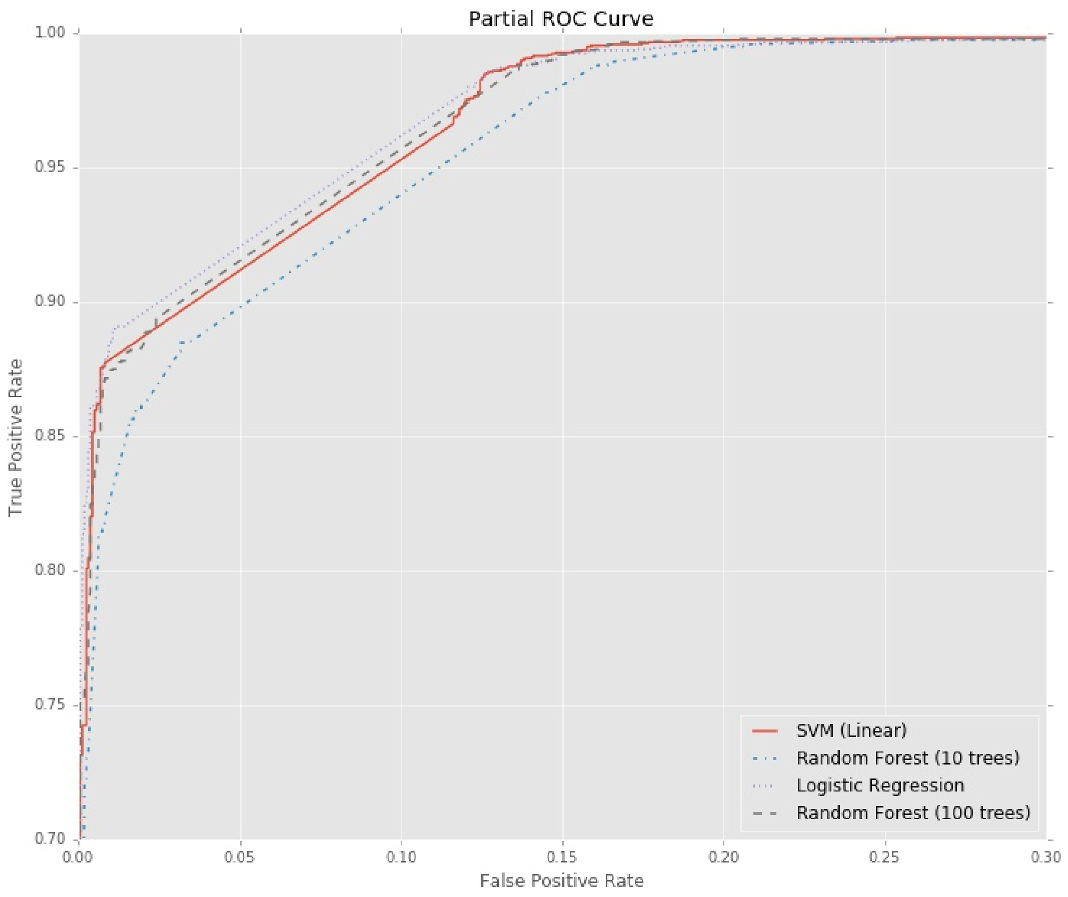


Figure 7.1: Partial ROC curve.

Chapter 8

Related Work

8.1 Web/Email Spam Detection

A lot of research was done in detecting spam posts in user generated content on the Internet. Emails have received a major share of attention over the last decade. Blanzieri et al [11] conducted a survey of many popular supervised learning techniques used over the years to detect e-mail based spam. Zhou et al [24] propose to solve the email spam problem as a three-way decision approach in order to improve the accuracy and reduce the false positive rate.

In recent years, more research is being done on detecting spam in user generated web content. For example, Mishne et al [19] detect spam comments in blog posts by constructing language models of blog posts and comments. Using such methods, they have achieved an accuracy of about 83% in detecting spam comments. Kolari et al [16] used local and link-based features to build SVM models for detecting spam blogs (or splogs). More recent work has begun to focus on social media networks such as Twitter and FaceBook. Xin et al [14] worked on a data mining based approach to detect spam on social media networks. McCord et al [17] used NLP and user-based features to build a Random Forest classifier that can detect spamming Twitter accounts. Alex Hai Wang [23] has done spam Twitter post detection using graph-based features.

But, as shown in the work by Tran et al [22], most of these traditional approaches will not work well when used for spam detection on classified web pages. In our research, we propose to use some characteristics that are inherent and specific to classified web pages in order to improve spam detection on these kind of domains.

8.2 Classified Spam Detection

The work by Tran et al [22] was specialized in detecting spam on classified websites. To the best of our knowledge, this is the only other work that has been published in the field of spam detection in classified websites. They have built a spam detection classifier that can detect spam posts in Craigslist. They argue that in order to improve over traditional content or NLP based classifiers we need to use additional features. They have proposed and used features such as time of ad posting, presence of URLs or e-mail addresses in the posts. Interestingly, they have also used domain-specific features such as the market price of the item being posted. They have shown an improvement of over 50% in F-1 scores (F-1 score is a measure of a classifier's accuracy and it ranges between 0 and 1 [3]). But, a major limitation of their work is that they rely on external source based features which is hard to generalize.

On the other hand, the major contribution of our work is that we propose the use of the natural categorization that exists in classifieds websites in order to train different classifiers. For example, we create one classifier for detecting Housing posts, another for detecting Automobile posts. Because, the posts in each category are all homogenous, regular NLP based classifiers can be trained in a much better way to recognize the benign posts. We can see this from the performance results presented in Section 7.2. The work done by Tran et al is complementary to ours in this regard and we can use some of the additional generalizable features that they proposed to make our classifiers even better.

Another work in this field of classified web site scams is done by McCormick et al [18] in 2013. Unlike the above work, McCormick et al. focus on detecting fraud posts. Fraud posts look very similar to legitimate posts and the fraud is done usually by phone or via e-mail. Thus, these fraud posts are very different from spam posts and require much more detailed information. For this, the authors used private information from the web server running the classifieds page such as age of the user account posting the ad, the IP address of the poster etc. Our work instead focuses on spam posts which are much more in number as noted by the authors themselves in their work.

Chapter 9

Future Work

9.1 Gathering Spam Data

In our work, we have used one benign category of posts (the ”Cars for Sale” section posts) to simulate the positive samples. We had to follow this approach mainly because it is hard to obtain a significant number of original spam posts required in order to train a classifier properly. Nevertheless, it would be good to obtain such samples to get a better idea of how well the system that we have proposed works in detecting spam in real life. In order to do this, one can use the following approaches.

- Collecting such spam posts from Craigslist manually is very labor intensive. One can instead take the help of the category classifiers similar to those we have proposed to aid in the collection process. For example, say a user wants to collect a significant number of spam posts from ”Housing” category. The user should first train a classifier using ”Housing” posts as positive examples and other posts from all other categories as negative examples. Then, the user can use this category classifier to automatically weed out examples that are very likely related to ”Housing”. Only those remaining posts can be manually analyzed to collect spam posts.

- Another approach is to simulate spam posts by using actual spam content from other media. For example, there are several organization that have been collecting a number of spam emails for many years now. They make the email datasets publicly available for the use of researchers [9]. We can use the content from these emails as positive examples in order to train our classifiers.

9.2 Automated Categorization

Our results show that we can accurately differentiate between various categories using only the language based features. This make it possible to develop a system that can automatically tag posts. We need to check if given a post, is it possible to determine what category it belongs to by running it all by the classifiers. Although the results from this project indicate that this is possible, it would be interesting to do this comprehensively by taking a number of categories into consideration.

Another interesting approach would be to use proven topic modeling techniques like NMF (Non-negative Matrix factorization) or LDA (Latent Dirichlet allocation) in order to automatically generate micro-categories. The intent is to be able to divide the macro category data into smaller micro categories. Websites like Craigslist can use this to automatically generate micro-category tags and advertise them in their main pages. As these micro-categories tend to be very dynamic and vary temporally and geographically, generating them using an automated approach seems to be a viable solution. One can evaluate these micro-category results manually by choosing a random selection of posts and see if the auto-generated categories make sense.

Chapter 10

Conclusion

In this work, we have shown that it is possible to build a spam detection system that is specialized for classified ad websites. Our main idea was to make use of the structured data present in these websites by building multiple classifiers that identify benign posts belonging to each website category. Using Craigslist as a representative example, we scraped about 20,000 posts for some macro categories (such as housing, electronics, automobiles etc.) and stored them in a queryable database (MongoDB). We then tried out supervised classification techniques using Python's Scikit Learn library to train a classifier for recognizing a specific category of posts. Our evaluation shows that this classifier performs accurately ($> 99\%$ accuracy) with very less false positives ($< 1.5\%$).

Bibliography

[1] California business and professions code. http://www.spamlaws.com/state/ca.shtml.

[2] Data protection: ”junk” e-mail costs internet users 10 billion a year worldwide. http://europa.eu/rapid/press-release_IP-01-154_en.htm?locale=en.

[3] F1 score. https://en.wikipedia.org/wiki/F1_score.

[4] Mongodb. https://docs.mongodb.com/.

[5] N-grams. https://en.wikipedia.org/wiki/N-gram.

[6] Scikit learn: Supervised learning methods. http://scikit-learn.org/stable/supervised_learning.html.

[7] Scikit learn: Tf-idf vectorizer. http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[8] Scrapy framework. http://doc.scrapy.org/en/latest/intro/overview.html.

[9] Spam archive. http://untroubled.org/spam/.

[10] Tf-idf. https://en.wikipedia.org/wiki/Tf?idf.

[11] Enrico Blanzieri and Anton Bryl. A survey of learning-based techniques of email spam filtering. *Artif. Intell. Rev.*, 29(1):63–92, 2008.

[12] Tim Bray. The javascript object notation (json) data interchange format. RFC 7159, RFC Editor, March 2014.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] Xin Jin, Cindy Xide Lin, Jiebo Luo, and Jiawei Han. Socialspamguard: A data mining-based spam detection system for social media networks. *PVLDB*, 4(12):1458–1461, 2011.

[15] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 3–14. ACM, 2008.

[16] Pranam Kolari, Akshay Java, Tim Finin, Tim Oates, and Anupam Joshi. Detecting spam blogs: A machine learning approach. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1351. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[17] Michael Mccord and M Chuah. Spam detection on twitter using traditional classifiers. In *International Conference on Autonomic and Trusted Computing*, pages 175–186. Springer, 2011.

[18] Alan Matthew McCormick and William Eberle. Discovering fraud in online classified ads. In *FLAIRS Conference*, 2013.

[19] Gilad Mishne, David Carmel, and Ronny Lempel. Blocking blog spam with language model disagreement. In *AIRWeb 2005, First International Workshop on Adversarial Information Retrieval on the Web, co-located with the WWW conference, Chiba, Japan, May 2005*, pages 1–6, 2005.

[20] Anand Rajaraman and Jeffrey David Ullman. Data mining. In *Mining of Massive Datasets*, pages 1–17. Cambridge University Press, 2011. Cambridge Books Online.

[21] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[22] Hung Tran, Thomas Hornbeck, Viet Ha-Thuc, James Cremer, and Padmini Srinivasan. Spam detection in online classified advertisements. In *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality*, pages 35–41. ACM, 2011.

[23] Alex Hai Wang. Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–10. IEEE, 2010.

[24] Bing Zhou, Yiyu Yao, and Jigang Luo. A three-way decision approach to email spam filtering. In *Advances in Artificial Intelligence, 23rd Canadian Conference on Artificial Intelligence, Canadian, AI 2010, Ottawa, Canada, May 31 - June 2, 2010. Proceedings*, pages 28–39, 2010.

Appendix A

Code Samples

A.1 Spider Code for Automobile Posts

```
class MySpider(CrawlSpider):

    name = "craigs"

    allowed_domains = ["craigslist.org"]


#     Bikes, Motorcycles, Cars, RVs, Boats

    start_urls = ["http://atlanta.craigslist.org/search/bia",

                  "http://atlanta.craigslist.org/search/mca",

                  "http://atlanta.craigslist.org/search/cta",

                  "http://atlanta.craigslist.org/search/rva",

                  "http://atlanta.craigslist.org/search/boo"]

    rules = (

        Rule(SgmlLinkExtractor(allow=(),

            restrict_xpaths=('//a[@class="button next"]',)),

            callback="parse_items", follow= True),
    )


    def parse_items(self, response):
        hxs = HtmlXPathSelector(response)
        titles = hxs.select('//span[@class="pl"]')
        links = []
        for titles in titles:
            item = CraigslistItem()
            link = titles.select("a/@href").extract()[0]
            links.append(link)
        for link in links:
            item = CraigslistItem()
            item['link']=link
            yield Request(urlparse.urljoin(response.url, link),
                meta={'item':item},callback=self.parse_listing_page)
    def parse_listing_page(self,response):
        hxs = HtmlXPathSelector(response)
        item = response.request.meta['item']
        item['title'] = hxs.select(
            '//span[@id="titletextonly"]/text()').extract()[0]
        item['description'] = hxs.select(
            '//section[@id="postingbody"]/text()').extract()[0]
        yield item
```

A.2 TF-IDF Vectorizer

```
vectorizer = sklearn.feature_extraction.text.TfidfVectorizer(
    ngram_range=(1,3),
    token_pattern=u'(?u)\\b[a-zA-Z][a-zA-Z]+\\b',
    stop_words='english',
    min_df=2,
    max_df=0.8)
```

A.3 Running the Classifiers and Measurements

```
def run_test(title, classifier, test_X, test_labels):
    print title
    predictions = classifier.predict(test_X)
    tps = np.sum(np.logical_and((predictions==test_labels),
                (test_labels == np.ones(predictions.size))))
    ps = np.sum(test_labels == np.ones(predictions.size))
    fps = np.sum(np.logical_and((predictions!=test_labels),
                (test_labels == np.ones(predictions.size))))
    ns = np.sum(test_labels != np.ones(predictions.size))
    tpr = tps / (ps * 1.0)
    fpr = fps / (ns * 1.0)

    test_scores = [x[1] for x in classifier.predict_proba(test_X)]
    fpr, tpr, thresholds = sklearn.metrics.roc_curve(test_labels, test_scores)
    accuracy = classifier.score(test_X, test_labels)
    auc = sklearn.metrics.roc_auc_score(test_labels, test_scores)
    print "TPR:", tpr, "FPR:", fpr
    print "Accuracy:", accuracy
    print "AUC Score:", auc
    return fpr, tpr, thresholds
```

A.4 Training and Testing with the Classifiers

```
def train_and_test_classifiers(train_X, test_X, test_labels):
    #    Logistic Regression
    lr = sklearn.linear_model.LogisticRegression(random_state=0)
    classifier = lr.fit(train_X, train_labels)
    title = "Logistic Regression"
    roc_curve[title] = run_test(title, classifier, test_X, test_labels)

    # SVM
    title = "SVM (Linear)"
    classifier = svm.SVC(kernel='linear', C=1, probability=True,
                         random_state=0).fit(train_X, train_labels)
    roc_curve[title] = run_test(title, classifier, test_X, test_labels)

    # Random Forests
    title = "Random Forest (10 trees)"
    rf = sklearn.ensemble.RandomForestClassifier(random_state=0)
    classifier = rf.fit(train_X, train_labels)
    roc_curve[title] = run_test(title, classifier, test_X, test_labels)

    title = "Random Forest (100 trees)"
    rf = sklearn.ensemble.RandomForestClassifier(random_state=0, n_estimators=100)
    classifier = rf.fit(train_X, train_labels)
    roc_curve[title] = run_test(title, classifier, test_X, test_labels)
```