INTRODUCTION TO WEB SCIENCE- FALL 2014

Assignment Ten

Sybil Melton

December 10, 2014

Contents

List of Tables

Listings

1 Blog Retrieval

Choose a blog or a newsfeed (or something similar as long as it has an Atom or RSS feed). It should be on a topic or topics of which you are qualified to provide classification training data. In other words, choose something that you enjoy and are knowledgable of. Find a feed with at least 100 entries.
Create between four and eight different categories for the entries in the feed.
Download and process the pages of the feed as per the week 12 class slides.

1.1 Solution

To find a blog, I used the "next blog" functionality to randomly choose one until I found one with a technology topic and more than 100 entries. The particular blog I chose is titled "The Invisible Things Lab's blog." The description is Kernel, Hypervisor, Virtualization, Trusted Computing and other system-level security stuff, which I have knowledge of from work and my studies at ODU. I used requests, feedparser, and BeautifulSoup to get the blog pages and parse the feed. I retrieved the first four pages and saved the entries in a list, to be filtered later. The content still had some markup language and illegal XML characters, so I used the same functions I created for assignment nine to remove them. Listing 1 is the code used to accomplish the first part of this task. [4]

```
entries = []
blog = 'http://theinvisiblethings.blogspot.com/'
path = '/feeds/posts/default'
query_arg = {'alt' : 'atom'}
udata = urllib.urlencode(query_arg)

full_url = blog+path+"?"+udata
r = requests.get(full_url)
f=feedparser.parse(full_url)
entries = f['entries']
bsdata = r.text
soup = BeautifulSoup(bsdata)
next = soup.find('link', rel='next')
for x in range(1,4):
  n= next.get('href')
  f=feedparser.parse(n)
  entries.extend(f['entries'])
  r = requests.get(n)
  bsdata = r.text
  soup = BeautifulSoup(bsdata)
  next = soup.find('link', rel='next')

_illegal_xml_chars_RE = re.compile(u'[\x00-\x08\x0b\xa9\xae\x0c\x0e-\x1F\uD800-\uDFFF\uFFFE\uFFFF]')

def remove_tags(text):
    cleanr =re.compile('<.*?>')
    cleantext = re.sub(cleanr,'', text)
    cleantext = re.sub("\n", ' ', cleantext)
    cleantext = cleantext.strip()
    cleantext = cleantext.replace(u'\u201c', '')
    cleantext = cleantext.replace(u'\u201d', '')
    cleantext = cleantext.replace(u'\u2019', '\'')
    cleantext = cleantext.replace('-&gt;', '')
    return _illegal_xml_chars_RE.sub(' ', cleantext)

for x in range(0,100):
  entries[x]['content'][0]['value'] = remove_tags(entries[x]['content'][0]['value'])
  entries[x]['title'] = remove_tags(entries[x]['title'])
```

Listing 1: Retrieve Blog

As I read the entries, I chose announcements, security, hardware, os, and paper as the categories. Announcements was for general announcements. Security was for any entry about cybersecurity. Hardware entries discussed chipset and DRAM. Os handled any general articles about operating systems that did not fit into security, and paper handled any blogs that just posted white papers or slides from conferences.

2 Classification

Manually classify the first 50 entries, and then classify (using the fisher classifier) the remaining 50 entries. Report the cprob() values for the 50 titles as well. From the title or entry itself, specify the 1-, 2-, or 3-gram that you used for the string to classify. Do not repeat strings; you will have 50 unique strings.
Create a table with the title, the string used for classification, cprob(), predicted category, and actual category.

2.1 Solution

My program set up the fisher classifier and database. The database was “feed.db” and is included with this report. Then feedfilter was used to manually classify the first 50 entries, train the classifier, and guess a classification for the remaining 50. Listing 2 is the code used to accomplish the next task. [4]

```
cl=docclass.fisherclassifier(docclass.getwords)
cl.setdb(’feed.db’)

feedfilter.read(entries, cl)
```

Listing 2: Set Up Classifier

I modified feedfilter.py to classify the entries. I changed the read function to accept my list of entries, so the list could be split between the first and second 50 entries. The ‘summary’ token was changed to the content value, because I found some entries were cut off because of the length. The title and content were concatenated before sent to the classifier. As the loop progressed, it added my classifier and manually assigned categories to the list for each entry. Then the train function from docclass.py was called and was not modified. Listing 3 is the modified code in feedfilter.py to accomplish this. [4]

```
  for x in range(0, 50):
    print
    print ’-----’
    # Print the contents of the entry
    print ’Title:     ’+entries[x][’title’].encode(’utf-8’)
    print
    print entries[x][’content’][0][’value’].encode(’utf-8’)

    # Combine all the text to create one item for the classifier
    fulltext= entries[x][’title’] + ’, ’ + entries[x][’content’][0][’value’]

    # Ask the user to specify the correct category and train on that
    tr=raw_input(’Enter classifier: ’)
    entries[x][’classifier’] = tr
    cl=raw_input(’Enter category: ’)
    entries[x][’actual’] = cl
    classifier.train(fulltext,cl)
```

Listing 3: First 50 Classification

Next the second 50 entries had to classified by me and the classifier. Additionally, the cprob had to be saved, so I modified docclass.py classify function to return it when a guess was given. This allowed me to save both the predicted category and cprob for each in the entries list. Listing 4 is the additional code in feedfilter.py to accomplish this. [4]

```
# classify the final 50 and get the guess and cprob
    for x in range(50, 100):
    print
    print ’-----’
    # Print the contents of the entry
    print ’Title:     ’+entries[x][’title’].encode(’utf-8’)
    print
    print entries[x][’content’][0][’value’].encode(’utf-8’)

    # Combine all the text to create one item for the classifier
    fulltext= entries[x]['title'] + ', ' + entries[x]['content'][0]['value']

    # Ask the user to specify the correct category
    cl=raw_input('Enter category: ')
    entries[x]['actual'] = cl
    # Print the best guess at the current category
    guess = classifier.classify(fulltext)
    entries[x]['pred'] = str(guess[0])
    entries[x]['cprob'] = round(guess[1], 3)
    print 'Guess: '+ str(guess[0])
```

Listing 4: Second 50 Classification

In order to create the table, the title, classifier strings, predicted category, actual category, and cprob were sent to “table.txt.” The output file was then converted to “table.tex,” in order to create table 2.1. Listing 5 is the code to output the data. [4]

```
#write results to file, for table creation
tab = open('table.txt', 'w', 0)
tab.write('Title\tClassifier\tPredicted\tActual\tcprob()\n')
for x in range(0, 50):
  tab.write(entries[x]['title'] + '\t' + entries[x]['classifier'] + '\t' + ' '+ '\t' + entries[x]['actual']+'\n')

for x in range(50, 100):
  tab.write(entries[x]['title'] + '\t' + ' ' + '\t' + entries[x]['pred']+'\t' + entries[x]['actual']+'\t'+str(entries[x]['cprob'])+'\n')

tab.close()
```

Listing 5: Table Data

Table 2.1: Classifier Data

TITLE	CLASSIFIER	PREDICTED	ACTUAL	cprob()
Qubes R3/Odyssey initial source code release	Join us		announcements	
Announcing Qubes OS Release 2!	bug fixes		announcements	
Physical separation vs. Software compartmentalization	topics discussed		paper	
Qubes OS R2 rc2, Debian template, SSLed Wiki, BadUSB, and more...	updates		announcements	
Qubes OS R2 rc1 has been released!	improvements		announcements	
Shattering the myths of Windows security	Windows security		security	
Qubes R2 Beta 3 has been released!	beta		announcements	
Windows 7 seamless GUI integration coming to Qubes OS!	Support Tools		os	
Thoughts on Intel's upcoming Software Guard Extensions (Part 2)	memory bus		hardware	
Thoughts on Intel's upcoming Software Guard Extensions (Part 1)	DRAM		hardware	
Qubes OS R3 Alpha preview: Odyssey HAL in action!	configuration template		announcements	
Introducing Qubes Odyssey Framework	hardcoded		announcements	
Qubes 2 Beta 2 has been released!	drivers		announcements	

Table 2.1: Classifier Data

TITLE	CLASSIFIER	PREDICTED	ACTUAL	cprob()
Converting untrusted PDFs into trusted ones: The Qubes Way	PDFs		paper	
Qubes 2 Beta 1 with initial Windows support has been released!	windows support		os	
How is Qubes OS different from...	linux BSD		os	
Introducing Qubes 1.0!	introducing Qubes		announcements	
Qubes 1.0 Release Candidate 1!	release candidate		announcements	
Some comments on "Operation High Roller"	two-factor authentication		security	
Windows support coming to Qubes!	coming to		announcements	
Qubes Beta 3!	installation guide		announcements	
Thoughts on DeepSafe	firewall		security	
Trusted Execution In Untrusted Cloud	confidentiality		security	
Exploring new lands on Intel CPUs (SINIT code execution hijacking)	processors		hardware	
Playing with Qubes Networking for Fun and Profit	infrastructure		os	
Qubes Beta 2 Released!	proud to announce		announcements	
Anti Evil Maid	disk encryption		security	
Interview about Qubes OS	interview about		announcements	
My SSTIC 2011 slides	security		security	
From Slides to Silicon in 3 years!	presentation		announcements	
USB Security Challenges	software attacks		security	
(Un)Trusting the Cloud	encrypted		security	
The App-oriented UI Model and its Security Implications	security by isolation		security	
Following the White Rabbit: Software Attacks Against Intel VT-d	publish		paper	
The Linux Security Circus: On GUI isolation	sniff all		security	
Why the US "password revolution" won't work	multi-factor authentication		security	
Qubes Beta 1 has been released!	new features		announcements	
Partitioning my digital life into security domains	compromise		security	
My documents got lost/stolen [offtopic]	SSL cert		security	
Update on Qubes	Stay tuned		announcements	
Qubes Alpha 3!	milestone		announcements	
ITL is hiring!	looking to hire		announcements	
On Thin Clients Security	desktop security		security	
(Un)Trusting your GUI Subsystem	depriviliged		security	
Qubes, Qubes Pro, and the Future...	the future		announcements	
The MS-DOS Security Model	attacker exploiting		security	
Skeletons Hidden in the Linux Closet: r00ting your Linux Desktop for Fun and Profit	malicious		security	
Qubes Alpha 2 released!	are here		announcements	
Disposable VMs	non-sensitive		security	
On Formally Verified Microkernels (and on attacking them)	verification attempts		security	

Table 2.1: Classifier Data

TITLE	CLASSIFIER	PREDICTED	ACTUAL	cprob()
Evolution		announcements	announcements	1.0
Remotely Attacking Network Cards (or why we do need VT-d and TXT)		announcements	security	1.0
Introducing Qubes OS		announcements	announcements	1.0
Priorities		security	security	1.0
Another TXT Attack		announcements	security	1.0
Evil Maid goes after TrueCrypt!		announcements	security	1.0
Intel Security Summit: the slides		announcements	paper	1.0
About Apple's Security Foundations Or Lack Of Thereof...		security	security	1.0
PDF signing and beyond		security	security	1.0
Vegas Toys (Part I): The Ring -3 Tools		announcements	security	1.0
Black Hat 2009 Slides		announcements	paper	1.0
Interview		announcements	announcements	0.983
Virtualization (In)Security Training in Vegas		announcements	announcements	1.0
Quest to The Core		announcements	announcements	1.0
More Thoughts on CPU backdoors		security	security	1.0
Thoughts About Trusted Computing		announcements	security	1.0
Trusting Hardware		security	security	1.0
The Sky Is Falling?		security	security	1.0
Attacking SMM Memory via Intel CPU Cache Poisoning		announcements	security	1.0
Independent Attack Discoveries		announcements	security	1.0
Attacking Intel TXT: paper and slides		announcements	paper	0.997
Nesting VMMs, Reloaded.		announcements	announcements	0.997
Closed Source Conspiracy		security	security	1.0
Why do I miss Microsoft BitLocker?		security	security	1.0
Attacking Intel Trusted Execution Technology		security	security	1.0
Microsoft executive "rebuts" our research!		announcements	security	1.0
Xen 0wning Trilogy: code, demos and q35 attack details posted		announcements	security	1.0
The three approaches to computer security		security	security	1.0
Teamwork Crediting		security	security	1.0
Intel patches the Q35 bug		announcements	announcements	1.0
Attacking Xen: DomU vs. Dom0 consideration		announcements	security	1.0
Our Xen 0wning Trilogy Highlights		security	announcements	1.0
0wning Xen in Vegas!		announcements	announcements	1.0
Rafal Wojtczuk joins Invisible Things Lab		announcements	announcements	1.0
1984?		announcements	announcements	0.999
Vegas Training 2008		announcements	announcements	1.0
Research Obfuscated		security	hardware	1.0

Table 2.1: Classifier Data

TITLE	CLASSIFIER	PREDICTED	ACTUAL	cprob()
The Most Stupid Security News Ever		security	security	1.0
The RSA Absurd		security	security	1.0
Kick Ass Hypervisor Nesting!		security	os	1.0
Razor-Thin Hypervisors		announcements	announcements	1.0
Thoughts On Browser Rootkits		announcements	security	1.0
Tricky Tricks		announcements	security	1.0
Virtualization Detection vs. Blue Pill Detection		announcements	announcements	1.0
We're ready for the Ptacek's challenge!		announcements	announcements	1.0
Invisible Things Lab, Bitlocker/TPM bypassing and some conference thoughts		announcements	security	1.0
Understanding Stealth Malware		announcements	security	1.0
The Human Factor		security	security	1.0
The Game Is Over!		security	security	1.0
Handy Tool To Play with Windows Integrity Levels		announcements	os	1.0

3 PERFORMANCE

Assess the performance of your classifier in each of your categories by computing precision, recall, and F1. Note that the definitions of precisions and recall are slightly different in the context of classification; see:
http://en.wikipedia.org/wiki/Precision_and_recall#Definition_.28classification_context.29
and
http://en.wikipedia.org/wiki/F1_score

3.1 SOLUTION

For a classifier, the expected prediction will be positive if labelled correctly and negative if not. The observation will be true or false, as determined by external judgement. A true positive (TP) is an entry labelled correctly. A false negative (FN) would be an entry that was not labelled, but it should have been. This classifier did not have any FN. A false positive (FP) was incorrectly labelled. [2] Precision, or positive prediction value, was calculated as:

$$\text{Precision} = \frac{tp}{tp + fp}$$

The recall, or sensitivity, was calculated as:

$$\text{Recall} = \frac{tp}{tp + fn}$$

The F1 score, or accuracy, was calculated as:

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Listing 6 is the code used to perform the calculations. [2], [3]

```python
tp = 0
fn = 0
fp = 0
for x in range(50, 100):
  if entries[x]['pred'] == entries[x]['actual']:
    tp += 1
  elif not entries[x]['pred'] and entries[x]['actual']:
    fn += 1
  else: #labelled incorrectly
    fp += 1

precision = float(tp) / (tp + fp)
recall = float(tp) / (tp + fn)

f1 = 2 * ((precision * recall) / (precision + recall))
```

Listing 6: Calculations

The classifier precision was only .58 and because there were no FN, the recall score was 1. This provided an F1 score of 0.734. I believe this was due to the lack of training for the paper and os categories. There were far fewer entries that belonged here, so they were not classified correctly by the classifier. Perhaps if there had been more trained entries, it would have been higher.
Finally, all entries and data were output to “entries.txt” to be included in this report. [1] Listing 7 in section 4 is the entire classify.py program and listing 8, inthe same section is feedfilter.py. The file docclass.py, as used, is included with this report.

4 Python Code

```python

import urllib
import feedparser
from bs4 import BeautifulSoup
import requests
import docclass
import feedfilter
import re

entries = []
blog = 'http://theinvisiblethings.blogspot.com/'
path = '/feeds/posts/default'
query_arg = {'alt' : 'atom'}
udata = urllib.urlencode(query_arg)

full_url = blog+path+"?"+udata
r = requests.get(full_url)
f=feedparser.parse(full_url)
entries = f['entries']
bsdata = r.text
soup = BeautifulSoup(bsdata)
next = soup.find('link', rel='next')
for x in range(1,4):
  n= next.get('href')
  f=feedparser.parse(n)
  entries.extend(f['entries'])
  r = requests.get(n)
  bsdata = r.text
  soup = BeautifulSoup(bsdata)
  next = soup.find('link', rel='next')

_illegal_xml_chars_RE = re.compile(u'[\x00-\x08\x0b\xa9\xae\x0c\x0e-\x1F\uD800-\uDFFF\uFFFE\uFFFF]')

def remove_tags(text):
    cleanr =re.compile('<.*?>')
    cleantext = re.sub(cleanr,'', text)
    cleantext = re.sub("\n", ' ', cleantext)
    cleantext = cleantext.strip()
    cleantext = cleantext.replace(u'\u201c', '')
    cleantext = cleantext.replace(u'\u201d', '')
    cleantext = cleantext.replace(u'\u2019', '\'')
    cleantext = cleantext.replace('-&gt;', '')
    return _illegal_xml_chars_RE.sub(' ', cleantext)

for x in range(0,100):
  entries[x]['content'][0]['value'] = remove_tags(entries[x]['content'][0]['value'])
  entries[x]['title'] = remove_tags(entries[x]['title'])

cl=docclass.fisherclassifier(docclass.getwords)
cl.setdb('feed.db')

feedfilter.read(entries, cl)

#write results to file, for table creation
tab = open('table.txt', 'w', 0)
tab.write('Title\tClassifier\tPredicted\tActual\tcprob()\n')
for x in range(0, 50):
  tab.write(entries[x]['title'] + '\t' + entries[x]['classifier'] + '\t' + ' '+ '\t' + entries[x]['actual']+'\n')

for x in range(50, 100):
  tab.write(entries[x]['title'] + '\t' + ' ' + '\t' + entries[x]['pred']+'\t' + entries[x]['actual']+'\t'+str(entries[x]['cprob'])+'\n')

tab.close()

# compute Precision, recall, f1
# tp is labelled correctly, fn is not labelled but should have been, fp is incorrectly labelled
tp = 0
fn = 0
```

```
fp = 0
for x in range(50, 100):
  if entries[x]['pred'] == entries[x]['actual']:
    tp += 1
  elif not entries[x]['pred'] and entries[x]['actual']:
    fn += 1
  else: #labelled incorrectly
    fp += 1

precision = float(tp) / (tp + fp)
recall = float(tp) / (tp + fn)

f1 = 2 * ((precision * recall) / (precision + recall))

# print the entries to file
out = open('entries.txt', 'w', 0)
for e in entries:
  print>>out, e

out.write('\n----------------------\n')
out.write("Precision: " + str(precision)+'\n')
out.write("Recall: " + str(recall)+'\n')
out.write("F1: " + str(f1)+'\n')
out.close()

print "Precision: " + str(precision)
print "Recall: " + str(recall)
print "F1: " + str(f1)
```

Listing 7: Complete Classifier

```
import feedparser
import re

# Takes a filename of URL of a blog feed and classifies the entries
def read(entries,classifier):
# classify the first 50 and train the classifer
  for x in range(0, 50):
    print
    print '-----'
    # Print the contents of the entry
    print 'Title:     '+entries[x]['title'].encode('utf-8')
    print
    print entries[x]['content'][0]['value'].encode('utf-8')

    # Combine all the text to create one item for the classifier
    fulltext= entries[x]['title'] + ', ' + entries[x]['content'][0]['value']

    # Ask the user to specify the correct category and train on that
    tr=raw_input('Enter classifier: ')
    entries[x]['classifier'] = tr
    cl=raw_input('Enter category: ')
    entries[x]['actual'] = cl
    classifier.train(fulltext,cl)

# classify the final 50 and get the guess and cprob
    for x in range(50, 100):
    print
    print '-----'
    # Print the contents of the entry
    print 'Title:     '+entries[x]['title'].encode('utf-8')
    print
    print entries[x]['content'][0]['value'].encode('utf-8')

    # Combine all the text to create one item for the classifier
    fulltext= entries[x]['title'] + ', ' + entries[x]['content'][0]['value']

    # Ask the user to specify the correct category
    cl=raw_input('Enter category: ')
    entries[x]['actual'] = cl
    # Print the best guess at the current category
    guess = classifier.classify(fulltext)
    entries[x]['pred'] = str(guess[0])
    entries[x]['cprob'] = round(guess[1], 3)
    print 'Guess: '+ str(guess[0])
```

Listing 8: Modified Feed Filter

References

[1] Alex Martelli. Python: Write a list to a file. http://stackoverflow.com/questions/899103/python-write-a-list-to-a-file. Accessed: 2014-12-7.

[2] Precision and recall. http://en.wikipedia.org/wiki/Precision_and_recall#Definition_.28classification_context.29. Accessed: 2014-12-10.

[3] F1 score. http://en.wikipedia.org/wiki/F1_score. Accessed: 2014-12-10.

[4] Toby Segaran. *Programming Collective Intelligence.* O'Reilly Media, 2007.