Caleb Scott - CS 373 – Lab 2

Strategy

My strategy for creating a URL classifier was based on determining which fields or sets of fields had the greatest impact on a site's trustworthiness. I devised a system that treats everything as malicious until it passes enough rules to get above an established point threshold. After examining the training file and the hints provided in the lab instructions the two attributes I determined to be the most significant are first the domain age followed by the alexa rank. The first thing I did was save all of the available variables for each record. I determined the most important variables by modifying the original readcorpus.py program to print out a value, such as domain_age_rank and the malicious_url rating for each record in the train.json file and redirected the output to a file. My program printed the domain_age_rank and the malicious_url with pythonprog.py > domain_age_rank.

A few lines of output are:

```
621 malicious_url 0
9 malicious_url 1
5647 malicious_url 0
52 malicious_url 1
```

So, I used:

```
grep "malicious_url 1" domain_age_rank > domain_age_rank_1
grep "malicious_url 0" domain_age_rank > domain_age_rank_0
sort -n domain_age_rank_0 >domain_age_rank_0_sorted
sort -n domain_age_rank_1 >domain_age_rank_1_sorted
```

Looking at the output I can see that for this example all ages that are in the negative and are less than 365 days old are malicious. I basically determined statistical ranking manually for each of the rules I implemented and used this strategy for all of the potential rules. Additionally, this could have been taken a step further and used machine learning to go through the training file to assign weights based from that I adjusted the weighting rules for each variable based on how prevalent and easily sortable each option is. The domain age is worth the most, with alexa ranking second.

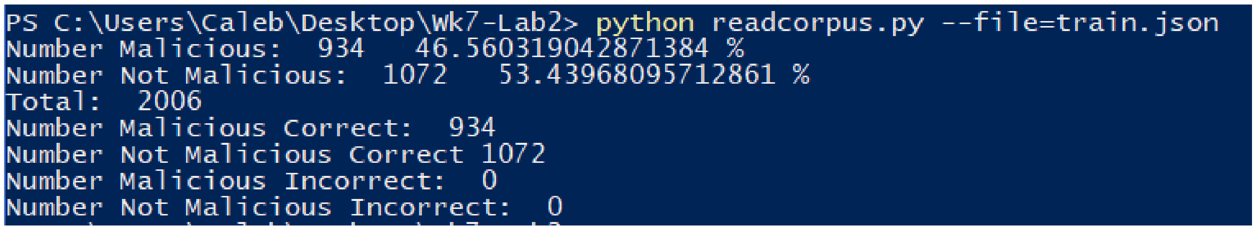
From there I added additional rules with lower weighting as the data was less clear; for example, when examining geographic location of a site 50% of Russian hosted sites were malicious. So, I don't want to automatically exclude all Russian sites but they must be treated with more suspicion and not be given as many points as a site hosted from the US for example. This same concept I used for alexa rank, domain age, file extension, geographic location, number of domain tokens, query info, top level domain and a compound rule. In only one case did I not follow my standard points pattern where known malicious qualities are worth 0 points and unknown or possibly safe/malicious attributes are worth a few points and known safe attributes are worth more. This is the compound rule combining if the domain age was less than 1 year old and whether the domain token included a keyword from a list I devised of major internet companies like PayPal, Apple, Amazon, etc... That rule will either give no points or subtract 5 points from the total in order to weight it as more malicious. All of the rules are explained more in depth in the following section. From there it was primarily experimentation to tune the right weightings and measuring the performance against the training file and the classify file.

Rules

Domain Age

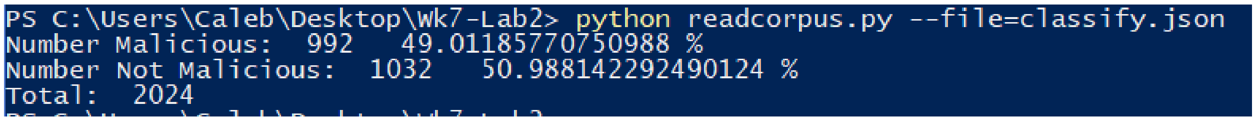
The domain age was by far the most important classification for this particular set of data because I was able to determine that I could perfectly sort the training file with a single rule based on the age of the domain. If anything < 366 days was determined to be malicious I would get the following results on the training and classify files.

Training.json



```
PS C:\Users\Caleb\Desktop\Wk7-Lab2> python readcorpus.py --file=train.json
Number Malicious:   934    46.560319042871384 %
Number Not Malicious:   1072    53.43968095712861 %
Total:   2006
Number Malicious Correct:   934
Number Not Malicious Correct 1072
Number Malicious Incorrect:   0
Number Not Malicious Incorrect:   0
```

Classify.json

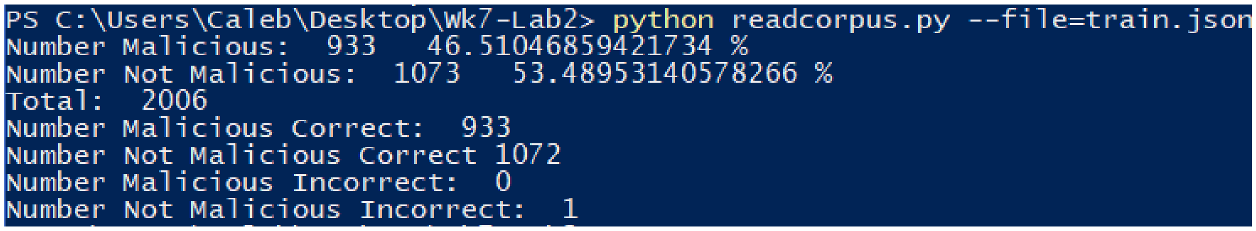


```
PS C:\Users\Caleb\Desktop\Wk7-Lab2> python readcorpus.py --file=classify.json
Number Malicious:   992    49.01185770750988 %
Number Not Malicious:   1032    50.988142292490124 %
Total:   2024
```

Even though I don't know for sure how well it performed on the classify file the % malicious vs % not malicious should be a pretty good indicator that it was close

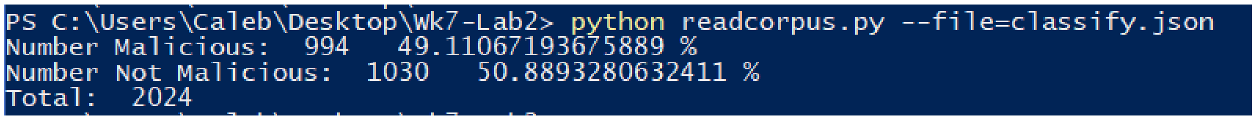
Results

Final Results Training File:



```
PS C:\Users\Caleb\Desktop\Wk7-Lab2> python readcorpus.py --file=train.json
Number Malicious:   933    46.51046859421734 %
Number Not Malicious:   1073    53.48953140578266 %
Total:   2006
Number Malicious Correct:   933
Number Not Malicious Correct 1072
Number Malicious Incorrect:   0
Number Not Malicious Incorrect:   1
```

Final Results Classify File:



```
PS C:\Users\Caleb\Desktop\Wk7-Lab2> python readcorpus.py --file=classify.json
Number Malicious:   994    49.11067193675889 %
Number Not Malicious:   1030    50.8893280632411 %
Total:   2024
```

Code

```
#!/usr/bin/python


import json, sys, getopt, os
```

```python
def usage():
  print("Usage: %s --file=[filename]" % sys.argv[0])
  sys.exit()

def main(argv):

  #####################
  #Data Import/Parsing#
  #####################

  file=''
  resultsFile= open("results.txt","w")

  myopts, args = getopt.getopt(sys.argv[1:], "", ["file="])

  for o, a in myopts:
    if o in ('-f, --file'):
      file=a
    else:
      usage()

  if len(file) == 0:
    usage()

  corpus = open(file, encoding='latin1')
  urldata = json.load(corpus, encoding="latin1")

  maliciousThreshold = 28
  #Threshold for domain age only
  #maliciousThreshold = 15

  numMalicious = 0
  numNotMalicious = 0
  numTotal = 0
  numMalCorrect = 0
  numMalIncorrect = 0
  numNotMalCorrect = 0
  numNotMalIncorrect = 0
  Company_List = ['amazon', 'facebook', 'ebay', 'paypal', 'coinbase', 'google', 'apple', 'microsoft', 'twitter', 'youtube', 'yahoo']

  for record in urldata:

    # Do something with the URL record data...
    malURL = record["malicious_url"]
    host_len = record["host_len"]
    fragment = record["fragment"]
    url_len = record["url_len"]
    default_port = record["default_port"]
    domain_age_days = record["domain_age_days"]
    tld = record["tld"]
    num_domain_tokens = record["num_domain_tokens"]

    #source: https://stackoverflow.com/questions/51796525/python-parsing-json-
file-to-access-values-returning-typeerror
    if(record["ips"] is not None and "geo" in record["ips"][0]):
      geo = record["ips"][0]["geo"]
    if(record["ips"] is not None and "ip" in record["ips"][0]):
      ip = record["ips"][0]["ip"]
    if(record["ips"] is not None and "type" in record["ips"][0]):
      typeVar = record["ips"][0]["type"]
    url = record["url"]
    alexa_rank = record["alexa_rank"]
    query = record["query"]
    file_extension = record["file_extension"]
    registered_domain = record["registered_domain"]
    scheme = record["scheme"]
    path = record["path"]
    path_len = record["path_len"]
    port = record["port"]
    host = record["host"]
    num_path_tokens = record["num_path_tokens"]
    domain_token_list = []
    path_token_list = []
    numTotal += 1

    for x in range(num_domain_tokens):
      domain_token_list.insert(x,record["domain_tokens"][x])
      #print (domain_token_list[x])

    for x in range(num_path_tokens):
      path_token_list.insert(x,record["path_tokens"][x])
      #print (path_token_list[x])

      ####################
      #URL Classification#
      ####################

    ###########Alexa Ranking###############
    if (alexa_rank == None) or (int(alexa_rank) >= 1000000):
      alexa_rank_score = 0
    elif (int(alexa_rank) >= 100000) and (int(alexa_rank) < 1000000):
      alexa_rank_score = 3
    elif (int(alexa_rank) >= 10000) and (int(alexa_rank) < 100000):
      alexa_rank_score = 4
    elif (int(alexa_rank) >= 1000) and (int(alexa_rank) < 10000):
      alexa_rank_score = 8
    elif (int(alexa_rank) >= 1) and (int(alexa_rank) < 1000):
      alexa_rank_score = 10
    else:
      alexa_rank_score = 0

    #print ('Alexa Rank Score: ',alexa_rank_score)
    #######Alexa Ranking End#############

    ######Domain Age#####
    if (int(domain_age_days) < 365):
      domain_age_days_score = 0
    else:
      domain_age_days_score = 15
    #print(domain_age_days_score)
    ####Domain Age End##

    ##############File Extension####################
    if (file_extension == 'exe') or (file_extension == 'it') or (file_extension
== 'de') or (file_extension == 'rar'):
      file_extension_score = 0
      #print (file_extension_score)
    elif (file_extension == 'cgi'):
      file_extension_score = 2
      #print (file_extension_score)
    elif (file_extension is None) or (file_extension == 'swf') or (file_extension
== 'jsp') or (file_extension == 'php'):
      file_extension_score = 3
      #print (file_extension_score)
    else:
      file_extension_score = 5
      #print (file_extension_score)
    ####File Extension End###################

    ######Location#####
    if (geo == 'RU'):
      geo_score = 0
      #print (geo_score)
```

```python
    elif (geo == 'CN'):
      geo_score = 2
    else:
      geo_score = 3
      #print (file_extension_score)
    ##Location End#######

    ####################Domain Tokens#######
    if (num_domain_tokens > 7):
      num_domain_tokens_score = 0
    else:
      num_domain_tokens_score = 2
    ######Domain Tokens End#####################

    ###Query#########
    if (str(query).startswith('cgi-bin') and (query is not None)):
      query_score = 0
    elif (str(query).startswith('cmd=') and (query is not None)):
      query_score = 0
    else:
      query_score = 5
    ###Query End########

    ######TLD##########
    if (tld == 'gov'):
      tld_score = 5
    elif ((tld == 'net') or (tld == 'jp') or (tld == 'eu') or (tld == 'fr') or
(tld == 'com') or (tld == 'org')):
      tld_score = 3
    else:
      tld_score = 0
    ################TLD END#########

    ###################Compound Rules######
    if ((int(domain_age_days) < 365) and
bool(set(domain_token_list).intersection(Company_List))):
      rule_1 = -5
    else:
      rule_1 = 0
    #########Compound Rules End#########

    #print (bool(set(domain_token_list).intersection(Company_List)))

    ##################Compound Rules End###
```

```
  ########Total Score#################
  totalScore = alexa_rank_score + domain_age_days_score + file_extension_score
+ geo_score + num_domain_tokens_score + query_score + tld_score + rule_1
  #totalScore = domain_age_days_score
  if (totalScore < maliciousThreshold):
   #print ('Total Score: ',totalScore,' Malicious')
   numMalicious += 1
   #print (url,', 1')
   resultsFile.write(url + ", 1\n")
   if (malURL == 1):
    numMalCorrect += 1
   else:
    #print (totalScore)
    #print(url)
    numMalIncorrect += 1
  else:
   #print (url,', 0')
   resultsFile.write(url + ", 0\n")
   #print ('Total Score: ',totalScore,' Not Malicious')
   numNotMalicious += 1
   if (malURL == 0):
    numNotMalCorrect += 1
   else:
    numNotMalIncorrect += 1
    #print(totalScore)
    #print(url)
  ########Total Score End#######
 print ('Number Malicious: ',numMalicious,' ',numMalicious/numTotal*100,'%')
 print ('Number Not Malicious: ',numNotMalicious,'
',numNotMalicious/numTotal*100,'%')

 print ('Total: ',numTotal)
 if (sys.argv[1] == '--file=train.json'):
  print ('Number Malicious Correct: ',numMalCorrect)
  print ('Number Not Malicious Correct',numNotMalCorrect)
  print ('Number Malicious Incorrect: ',numMalIncorrect)
  print ('Number Not Malicious Incorrect: ',numNotMalIncorrect)
 corpus.close()
 resultsFile.close()
if __name__ == "__main__":
 main(sys.argv[1:])
```