Metrics of successful sites and companies

January 8, 2017

In [9]:
```
#First we import the libraries we will need
import urllib
import urllib2
import time
import os
from bs4 import BeautifulSoup
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [10]:
```
#First of all we need to find all the name of the sites that belong to for
#The information needed from the below link
url = "http://www.zyxware.com/articles/4344/list-of-fortune-500-companies-
list_company_number =[]
list_company_name = []
list_company_website = []
list500_sites = []
list500_names = []
list500_num = []
list500_url = []
```

In [11]:
```
#In order to extract the needed informations we will create 3 lists. The f
#second one will contain the name of the company and the 3rd one will cont
#For achieving this purpose we will create a funstion that will in its tur
#In order to know if the function worked we will ask it to return the firs
def websites (url):
    from time import time # I used it to see how much time it does to run
    start = time ()
    browser = urllib2.build_opener()
    browser.addheaders = [('User-agent', 'Mozilla/5.0')]
    response = browser.open(url)# this might throw an exception if someth
    myHTML = response.read()
    soup = BeautifulSoup(myHTML,"lxml")
    o = 0
    td_list =[]
    for row2 in soup.html.body.findAll('td'):
```

```
            td_list.insert(o, row2)
            o = o + 1
        a = 0
        b = 1
        c = 2
        list_numbering = 0
        for i in range (0,500):
            num = str(td_list[a])
            company = str(td_list[b])
            site = str(td_list[c])
            c_num = re.findall('>(.+?)</td>',num)
            c_num = str(c_num[0])
            c_name = re.findall('>(.+?)</td>',company)
            c_name = str(c_name[0])
            c_site = re.findall('">(.+?)</a>',site)
            c_site = str(c_site[0])
            list_company_number.insert(list_numbering,c_num)
            list_company_name.insert(list_numbering,c_name)
            list_company_website.insert(list_numbering,c_site)
            a = a + 3
            b = b + 3
            c = c + 3
            list_numbering =  list_numbering + 1
        end = time ()
        duration = round (end - start, 1)
        minutes = round (duration /60, 1)
        print 'The lists are ready in ', duration, ' seconds'
        print 'The lists are ready in ', minutes, ' minutes'
```

In [12]:
```
# After creating the function we should now test that it actually works co
websites (url)
```

```
The lists are ready in  1.2  seconds
The lists are ready in  0.0  minutes
```

In [13]:
```
#Try to validate each page url #pip install validators
import validators
nv = 0
for num in range(len(list_company_website)):
    line = 'http://' + str(list_company_website[num])
    x = validators.url(line)
    if x != True:
        nv = nv +1
print "The validation is complete! There were" , nv, "not valid pages"
```

```
The validation is complete! There were 0 not valid pages
```

In [14]:
```
#def list_company_HTML (list_company_website,list_company_name,start,end)
import time
browser2 = urllib2.build_opener()
browser2.addheaders = [('User-agent', 'Mozilla/5.0')]
for i in range (0,500):
    k = str(i + 1)
    lc = str(list_company_website[i])
    lc = lc.replace("'","")
    lc = lc.replace("[","")
    lc = lc.replace("]","")
    lcn = str(list_company_name[i])
    lcn = lcn.replace("'","")
    lcn = lcn.replace("[","")
    lcn = lcn.replace("]","")
    url2= 'http://' + lc
    #print (url2)
    list500_names.insert(i,lcn)
    list500_url.insert(i,lc)
    list500_num.insert(i,k)
    if i == 118 or i == 464:#The site 118(119) has a problem and the whole
        #when I run it so we will thing of this site as a not downloadable
        list500_sites.insert(i,0)
        #print ("The site " + k + " has NOT been downloaded!")
    else:
        #an exception might be thrown, so the code should be in a try-exce
        try:
            response2=browser2.open(url2)
        except Exception: # this describes what to do if an exception is t
            list500_sites.insert(i,0)
            print ("The site " + str(i) + " has NOT been downloaded!")
            continue
            #read the response in html format. This is essentially a long
        myHTML2=response2.read()
        list500_sites.insert(i,myHTML2)
        #wait for 2 seconds
        time.sleep(2)
        #print ("The site " + k + " has been downloaded!")
    #print "We saved: ",str(i + 1)," sites!"
    #print (len(list500_names),list500_names)
```

```
The site 14 has NOT been downloaded!
The site 15 has NOT been downloaded!
The site 37 has NOT been downloaded!
The site 62 has NOT been downloaded!
The site 90 has NOT been downloaded!
The site 97 has NOT been downloaded!
The site 127 has NOT been downloaded!
The site 135 has NOT been downloaded!
```

```
The site 141 has NOT been downloaded!
The site 161 has NOT been downloaded!
The site 164 has NOT been downloaded!
The site 209 has NOT been downloaded!
The site 216 has NOT been downloaded!
The site 239 has NOT been downloaded!
The site 242 has NOT been downloaded!
The site 275 has NOT been downloaded!
The site 306 has NOT been downloaded!
The site 326 has NOT been downloaded!
The site 363 has NOT been downloaded!
The site 414 has NOT been downloaded!
The site 424 has NOT been downloaded!
The site 441 has NOT been downloaded!
The site 481 has NOT been downloaded!
```

In [15]:
```
#As we can see there is one site that hasn't been downloaded in order
#to keep track of the sites that we could not download
#we will create a new list that we will keep them all together there
not_d = []
not_d_n = []
num = []
def not_downloadables (list500_names,list500_sites):
    met = 0
    for i in range(len(list500_names)):
        if list500_sites[i] == 0:
            ct = list500_names[i]
            not_d.insert(met,ct)
            not_d_n.insert(met,str(i))
            num.insert(met,met)
            met = met + 1
```

In [16]:
```
#Now we will run the function to see which sites havent been downloaded
not_downloadables (list500_names,list500_sites)
d = {'company' : pd.Series(not_d, index=[num]),
     'number' : pd.Series(not_d_n, index=[num])}
nd = pd.DataFrame(d)
nd
```

Out[16]:

	company	number
0	Costco	14
1	Fannie Mae	15
2	Target	37
3	HCA Holdings	62
4	Nike	90
5	Tesoro	97

```
6            Arrow Electronics    118
7             Emerson Electric    127
8                   AutoNation    135
9           Southwest Airlines    141
10                     Southern    161
11    American Electric Power    164
12                       Loews    209
13                  PBF Energy    216
14                Toys "R" Us    239
15          Dominion Resources    242
16             Global Partners    275
17             PayPal Holdings    306
18                  News Corp.    326
19                    Williams    363
20              Tractor Supply    414
21                      Ameren    424
22  Old Republic International    441
23           St. Jude Medical    464
24    Raymond James Financial    481
```

In [17]:
```
#Now we will perfom a reliability test for the text that is
#included in the html code of the company's page
from pattern import metrics
readability = []
rdb = []
```

In [18]:
```
def readable (list500_names,list500_sites):
    for i in range (len(list500_names)):
            myHTML = list500_sites[i]
            if myHTML == 0:
                readability.insert(i,"n/a")
                rdb.insert(i,"n/a")
            else:
                a = metrics.flesch_reading_ease(myHTML) * 100
                a = round (a, 1)
                if a > 90:
                    readability.insert(i,"Very easy")
                    rdb.insert(i,6)
                elif a > 80:
                    readability.insert(i,"Easy")
                    rdb.insert(i,5)
                elif a > 70:
                    readability.insert(i,"Fairly easy")
                    rdb.insert(i,4)
                elif a > 60:
                    readability.insert(i,"Standard")
                    rdb.insert(i,3)
                elif a > 50:
```

```
                readability.insert(i,"Fairly difficult")
                rdb.insert(i,2)
            elif a > 30:
                readability.insert(i,"Difficult")
                rdb.insert(i,1)
            else:
                readability.insert(i,"Very Confusing")
                rdb.insert(i,0)
        print "The function is completed!"
```

In [19]:
```
readable (list500_names,list500_sites)
```

```
The function is completed!
```

In [21]:
```
d1 = {'company' : pd.Series(list500_names, index=[list500_num]),
      'url' : pd.Series(list500_url, index=[list500_num]),
      'Readability' : pd.Series(readability, index=[list500_num]),
      'Readability_index' : pd.Series(rdb, index=[list500_num])}
fre = pd.DataFrame(d1)
fre.tail(3) #we see the first 3 in the data frame
```

Out[21]:

	Readability	Readability_index	company	url
498	Very Confusing	0	NVR	www.nvrinc.com
499	Very Confusing	0	Cincinnati Financial	www.cinfin.com
500	Very Confusing	0	Burlington Stores	www.burlingtonstores.com

In [53]:
```
#Retreiving the social media from each site
#First create empty lists for the ones that
#we will need to calculate
sm_f = []
sm_t = []
sm_i = []
sm_p = []
sm_y = []
sm_l = []
sm_nm = []
nm = []
sm_url = []
```

In [54]:
```
#Then create a function that will feel in those
#lists so as to make the data frame later on
def socialmedia (list500_sites,list500_names,list500_url):
    from time import time
```

```
# I used it to see how much time it does to run the function
start = time ()
for i in range(len(list500_names)):
        myHTML = list500_sites[i]
        sm = ['facebook.com','twitter.com',
              'instagram.com','pinterest.com',
              'youtube.com','linkedin.com']
        if myHTML == 0:
            #print(str(i),"no")
            sm_nm.insert(i,list500_names[i])
            nm.insert(i,i)
            sm_url.insert(i,list500_url[i])
            sm_f.insert(i,'n/a')
            sm_t.insert(i,'n/a')
            sm_i.insert(i,'n/a')
            sm_p.insert(i,'n/a')
            sm_y.insert(i,'n/a')
            sm_l.insert(i,'n/a')
        else:
            for index in range(len(sm)):
                x = sm[index]
                social = re.findall(x,myHTML)
                if (len(social) > 0):
                    if x == 'facebook.com':
                        answerf = 'TRUE'
                    if x == 'twitter.com':
                        answert = 'TRUE'
                    if x == 'instagram.com':
                        answeri = 'TRUE'
                    if x == 'pinterest.com':
                        answerp = 'TRUE'
                    if x == 'youtube.com':
                        answery = 'TRUE'
                    if x =='linkedin.com':
                        answerl = 'TRUE'
                else:
                     if x == 'facebook.com':
                        answerf = 'FALSE'
                     if x == 'twitter.com':
                        answert = 'FALSE'
                     if x == 'instagram.com':
                        answeri = 'FALSE'
                     if x == 'pinterest.com':
                        answerp = 'FALSE'
                     if x == 'youtube.com':
                        answery = 'FALSE'
                     if x =='linkedin.com':
                        answerl = 'FALSE'
```

```
                    sm_nm.insert(i,list500_names[i])
                    nm.insert(i,i)
                    sm_url.insert(i,list500_url[i])
                    sm_f.insert(i,answerf)
                    sm_t.insert(i,answert)
                    sm_i.insert(i,answeri)
                    sm_p.insert(i,answerp)
                    sm_y.insert(i,answery)
                    sm_l.insert(i,answerl)
            end = time ()
            duration = round (end - start, 3)
            minutes = round (duration /60, 1)
            print 'The lists are completed in ', minutes, ' minutes'
            print 'The lists are ready in ', duration, ' seconds'
```

In [55]:
```
#Now we will run the function for the 25 first sites for starters
socialmedia (list500_sites,list500_names,list500_url)
```

```
The lists are completed in  0.0  minutes
The lists are ready in  0.259  seconds
```

In [58]:
```
#Finally we create the data frame with the elements we found
d2 = {'company' : pd.Series(sm_nm, index=[nm]),
     'facebook' : pd.Series(sm_f, index=[nm]),
      'twitter' : pd.Series(sm_t, index=[nm]),
    'instagram' : pd.Series(sm_i, index=[nm]),
     'pinterest' : pd.Series(sm_p, index=[nm]),
    'youtube' : pd.Series(sm_y, index=[nm]),
     'linkedin' : pd.Series(sm_l, index=[nm]),}
social_media = pd.DataFrame(d2)
social_media.tail(3) #we see the first 3 in the data frame
```

Out[58]:

	company	facebook	instagram	linkedin	pinterest	twitter	youtube
497	NVR	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
498	Cincinnati Financial	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
499	Burlington Stores	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE

In [59]:
```
#Create the lists we will need for the data frame
l_nm = []
l_ex = []
l_in = []
l_t = []
nm = []
l_url = []
```

```
In [60]: #create the function that will calculate the different type of links
         def links (list500_sites,list500_names,list500_url):
             from time import time
             # I used it to see how much time it does to run the function
             start = time ()
             for num in range(len(list500_names)):
                     myHTML = list500_sites[num]
                     if myHTML == 0:
                         l_nm.insert(num,list500_names[num])
                         l_ex.insert(num,'n/a')
                         l_t.insert(num,'n/a')
                         l_in.insert(num,'n/a')
                         nm.insert(num,num)
                     else:
                         href = re.findall('href',myHTML)
                         external = re.findall('href="https:',myHTML)
                         ex = (len(external))
                         alllinks = (len(href))
                         internal =  (len(href) - len(external))
                         l_nm.insert(num,list500_names[num])
                         l_ex.insert(num,ex)
                         l_t.insert(num,alllinks)
                         l_in.insert(num,internal)
                         nm.insert(num,num)
             end = time ()
             duration = round (end - start, 3)
             minutes = round (duration /60, 1)
             print 'The lists are ready in ', minutes, ' minutes'
             print 'The lists are ready in ', duration, ' seconds'

In [61]: #Run the function in order to find the external,
         #internal and total links of each site
         #For now we are running for the first 25 sites only
         links (list500_sites,list500_names,list500_url)

The lists are ready in  0.0  minutes
The lists are ready in  0.088  seconds


In [63]: #Create a dataframe so as to be able to see
         #the results of the function we run
         d3 = {'company' : pd.Series(l_nm, index=[nm]),
               'external' : pd.Series(l_ex, index=[nm]),
               'internal' : pd.Series(l_in, index=[nm]),
              'total links' : pd.Series(l_t, index=[nm])}
         sites_links = pd.DataFrame(d3)
         sites_links.tail(3) #we see the first 3 in the data frame

Out[63]:
         company external internal total links
497          NVR        5       29          34
```

```
498  Cincinnati Financial        3       73          76
499     Burlington Stores       16      168         184
```

In [70]:
```
#The initial lists we will need in order
#to calculate the loading time
lt_nm = []
lt_time = []
nm = []
lt_url = []
```

In [71]:
```
#the function that will calculate the loading time
def loadtime (list_company_website,list500_names,list500_url):
    from time import time
    browser2 = urllib2.build_opener()
    browser2.addheaders = [('User-agent', 'Mozilla/5.0')]
    for num in range(len(list500_names)):
        lc = str(list_company_website[num])
        lc = lc.replace("'","")
        lc = lc.replace("[","")
        lc = lc.replace("]","")
        url2 = 'http://' + lc
        if num == 118 or num == 464:
            #The site 118(119) has a problem and the whole code
            #is stacking when I run it so we will thing of this
            #site as a not downloadable
            lt_nm.insert(num,list500_names[num])
            lt_time.insert(num,'n/a')
            nm.insert(num,num)
            lt_url.insert(num,list500_url[num])
            #print ("The site " + str(num) + " has NOT been loaded!")
        else:
            try:
                response2 = browser2.open(url2)
            except Exception:
                lt_time.insert(num,'n/a')
                lt_nm.insert(num,list500_names[num])
                nm.insert(num,num)
                print ("The site " + str(num)+ " has NOT been loaded!")
                continue
            start_time = time()
            myHTML2 = response2.read()
            end_time = time()
            response2.close()
            l_t = round(end_time-start_time, 3)
            #in order to be more readable we rounded the time
            loadt = str(l_t)
            lt_nm.insert(num,list500_names[num])
            lt_time.insert(num,loadt)
```

```
                    nm.insert(num,num)
                    lt_url.insert(num,list500_url[num])
                    #print ("The site " + str(num) + " has been loaded!")
            print "The function is completed!"
```

In [72]:
```
#running the function for the first 25 sites
loadtime (list_company_website,list500_names,list500_url)
```

```
The site 14 has NOT been loaded!
The site 15 has NOT been loaded!
The site 37 has NOT been loaded!
The site 62 has NOT been loaded!
The site 90 has NOT been loaded!
The site 97 has NOT been loaded!
The site 127 has NOT been loaded!
The site 135 has NOT been loaded!
The site 141 has NOT been loaded!
The site 161 has NOT been loaded!
The site 164 has NOT been loaded!
The site 204 has NOT been loaded!
The site 209 has NOT been loaded!
The site 216 has NOT been loaded!
The site 242 has NOT been loaded!
The site 275 has NOT been loaded!
The site 306 has NOT been loaded!
The site 326 has NOT been loaded!
The site 363 has NOT been loaded!
The site 414 has NOT been loaded!
The site 424 has NOT been loaded!
The site 441 has NOT been loaded!
The site 481 has NOT been loaded!
The function is completed!
```

In [73]:
```
#creating the data frame with the loading times
d4 = {'company' : pd.Series(lt_nm, index=[nm]),
      'loading time' : pd.Series(lt_time, index=[nm])}
loading_time = pd.DataFrame(d4)
loading_time.head(3) #we see the first 3 in the data frame
```

Out[73]:

	company	loading time
0	Walmart	0.449
1	Exxon Mobil	5.376
2	Apple	0.021

In [74]:
```
#Find out how many and what type of images each site has
#first we create the initially empty lists
p_p = []
p_d = []
```

```
p_jpg = []
p_jpeg = []
p_gif = []
p_tif = []
p_tiff = []
p_bmp = []
p_jpe = []
p_nm = []
p_tt =[]
nm = []
p_url = []
```

In [75]:
```
#Then we create the function that will explore
#the html pages and search for the images
def images (list500_sites,list500_names,list500_url):
    from time import time # I used it to see
    #how much time it does to run the function
    start = time ()
    for num in range(len(list500_names)):
            myHTML = list500_sites[num]
            image = ['.png','.dib','.jpg','.jpeg',
                     '.bmp','.jpe','.gif','.tif','.tiff']
            totalnumber = 0
            if myHTML == 0:
                p_nm.insert(num,list500_names[num])
                p_p.insert(num,'n/a')
                p_d.insert(num,'n/a')
                p_jpg.insert(num,'n/a')
                p_jpeg.insert(num,'n/a')
                p_gif.insert(num,'n/a')
                p_tif.insert(num,'n/a')
                p_tiff.insert(num,'n/a')
                p_bmp.insert(num,'n/a')
                p_jpe.insert(num,'n/a')
                p_tt.insert(num,'n/a')
                nm.insert(num,num)
                p_url.insert(num,list500_url[num])
            else:
                for index in range(len(image)):
                    x = image[index]
                    photo = re.findall(x,myHTML)
                    if x == '.png':
                        p = str (len(photo))
                    if x == '.dib':
                        d = str (len(photo))
                    if x == '.jpg':
                        jpg = str (len(photo))
                    if x == '.jpeg':
```

```
                                jpeg = str (len(photo))
                            if x == '.gif':
                                gif = str (len(photo))
                            if x == '.tif':
                                tif = str (len(photo))
                            if x == '.tiff':
                                tiff = str (len(photo))
                            if x == '.bmp':
                                bmp = str (len(photo))
                            if x == '.jpe':
                                jpe = str (len(photo))
                            totalnumber = len(photo) + totalnumber
                        total = str (totalnumber)
                        p_nm.insert(num,list500_names[num])
                        p_p.insert(num,p)
                        p_d.insert(num,d)
                        p_jpg.insert(num,jpg)
                        p_jpeg.insert(num,jpeg)
                        p_gif.insert(num,gif)
                        p_tif.insert(num,tif)
                        p_tiff.insert(num,tiff)
                        p_bmp.insert(num,bmp)
                        p_jpe.insert(num,jpe)
                        p_tt.insert(num,total)
                        nm.insert(num,num)
                        p_url.insert(num,list500_url[num])
            end = time ()
            duration = round (end - start, 3)
            minutes = round (duration /60, 1)
            print 'The lists are ready in ', minutes, ' minutes'
            print 'The lists are ready in ', duration, ' seconds'

In [76]: #Then we run the function for the first 20 sites for now
         images (list500_sites,list500_names,list500_url)

The lists are ready in  0.1  minutes
The lists are ready in  3.341  seconds


In [77]: #Finally we create a dataframe in order to see the results of the functio
         d5 = {'company' : pd.Series(p_nm, index=[nm]),
               '.png' : pd.Series(p_p, index=[nm]),
               '.dib' : pd.Series(p_d, index=[nm]),
               '.jpg' : pd.Series(p_jpg, index=[nm]),
               '.jpeg' : pd.Series(p_jpeg, index=[nm]),
               '.bmp' : pd.Series(p_bmp, index=[nm]),
               '.jpe' : pd.Series(p_jpe, index=[nm]),
               '.gif' : pd.Series(p_gif, index=[nm]),
```

```
            '.tif' : pd.Series(p_tif, index=[nm]),
            '.tiff' : pd.Series(p_tiff, index=[nm]),
            'total images' : pd.Series(p_tt, index=[nm])}
        images_types = pd.DataFrame(d5)
        images_types.head(3) #we see the first 3 in the data frame
```

Out[77]:

	.bmp	.dib	.gif	.jpe	.jpeg	.jpg	.png	.tif	.tiff	company	total images
0	0	0	55	153	153	63	46	10	0	Walmart	480
1	0	0	1	0	0	16	2	4	0	Exxon Mobil	23
2	0	0	9	0	0	0	2	0	0	Apple	11

In [78]:
```
#Now we will find the different dimensions that each site uses
#initially we create the empty lists we will need
nm = []
s_comp = []
s_dimensions = []
s_times = []
s_tt_dif_dim = []
ht = [] #list of different heights in each case
wt = [] #list of different widths in each case
h_w = [] # combinations of height and width
dif_size = []
un_size = []
s_url = []
```

In [79]:
```
#With the below function we will gather
#in a variable all the different dimensions
#and in another one all the times that each
#dimension occures for each html code
def find_dif_sizes (list_company_website,list500_names,list500_url):
    from time import time # I used it to see how much time it does to run
    start = time ()
    for num in range(len(list500_names)):
            nm.insert(num,num)
            s_comp.insert(num,list500_names[num])
            s_url.insert(num,list500_url[num])
            myHTML = list500_sites[num]
            if myHTML == 0:
                s_dimensions.insert(num,0)
                s_times.insert(num,0)
            else:
                soup = BeautifulSoup(myHTML, "lxml")
                # we create 2 local variables so as to gather the
                #different dimensions and occurencies  of each page sepera
                s_dimensions_local = []
                s_times_local = []
                hw = 0
                # we use it for the lists of height and width
```

```
# find all the img in the first site html.Since in some
#cases either the height or the width is missing we would
#like to keep only the ones that have both dimensions
for tag in soup.find_all('img'):
    h = tag.attrs.get('height', None)
    w = tag.attrs.get('width', None)
    #we use if to check which ones have both
    if h != None:
        if w != None:
            ht.insert(hw,h)
            wt.insert(hw,w)
            hw = hw + 1
hw2 = 0
for l in range(len(ht)):
    h_w_c = ht[l] + 'x' + wt[l]
    #we create a str with the form (300x300)
    #so as to be more easily to read later on
    h_w.insert(hw2,h_w_c)
    #we put it in a new list
    hw2 = hw2 + 1
if h_w == []:#we check if there are not any dimensions ava
    nm.insert(num,num)
    s_comp.insert(num,list500_names[num])
    s_dimensions.insert(num,0)
    s_times.insert(num,0)
if h_w != []:#now we continue with the cases
    #where the dimensions are indeed available
    from collections import Counter
    hw_unique = Counter(h_w)
    hw_unique2 = str(hw_unique)
    #the unique different dimensions for the specific site
    #Due to the fact that we are talking about
    #a list we have to split the parts we need
    split1 = hw_unique2.split('{')
    a = split1[1]
    split2 = a.split('}')
    b = split2[0]
    split3 = b.split(',')
    finalsplit = []
    fs = []
    z = 0
    m = 1
    j = 0
    z1 = 0
    m1 = 1
    #each of the items in split3 has a form '300x300 : 15
    #and in order to create the dataframe we have
    #to split this form and keep the informations in diffe
```

```
                            for numb in split3:
                                oldstring = numb
                                newstring = oldstring.replace("'", "")
                                new = newstring.replace("'","")
                                string = new.replace(" ","")
                                finalstring = string.split(':')
                                #the finalstring is a list that contains the dimen
                                #and the occurencies in order toseperate in differ
                                #lists we create an additional loop
                                for xx in range(len(finalstring)):
                                    ax = finalstring[xx]
                                    if 'x' in ax:
                                        s_dimensions_local.insert(z1,finalstring[x
                                        z1 = z1 + 1
                                    else:
                                        s_times_local.insert(m1,finalstring[xx])
                                        m1 = m1 + 1
                            #Now we can add to the lists the parts we created so a
                            #to have them all gathered together
                            s_dimensions.insert(num,s_dimensions_local)
                            s_times.insert(num,s_times_local)
                end = time ()
                duration = round (end - start, 3)
                minutes = round (duration /60, 1)
                print 'The lists are ready in ', minutes, ' minutes'
                print 'The lists are ready in ', duration, ' seconds'

In [80]: #Run the function for the first 20 sites
         find_dif_sizes (list500_sites,list500_names,list500_url)

The lists are ready in  1.4  minutes
The lists are ready in  81.813  seconds


In [81]: #Find the unique different image dimensions and put them on a list
         def unique_dif_sizes (s_dimensions,list500_names):
             ds = 0
             for num in range(len(list500_names)):
                 asw = s_dimensions[num]
                 if asw != 0 :
                     for s in range(len(asw)):
                         ss = asw[s]
                         dif_size.insert(ds,ss)
                         ds = ds + 1
             dsu = 0
             for i in dif_size:
                 if i not in un_size:
                     un_size.insert(dsu,i)
```

```
                    dsu = dsu + 1
            print(un_size)
```

In [82]:
```
#Run the function unique_dif_sizes
unique_dif_sizes (s_dimensions,list500_names)
```

```
['15x75', '44x556', '1x1', '800x1200', '24pxx133px', '21pxx173px', '49x49', '50x45'
```

In [83]:
```
#The lists we will need for the next function
t_f_s = []
ttf = []
nm = []
com = []
```

In [84]:
```
#Function in order to check whether or not each
#company has these dimensions
def dimensions_per_company (un_size,list500_names):
    from time import time
    # I used it to see how much time it does to run the function
    start = time ()
    #t_f_s.insert(0,un_size)
    #ttf.insert(0,t_f_s)
    for num in range(len(list500_names)):
        #print(str(num))
        s1a = s_dimensions[num]
        #dimensions of site num
        where = [] #empty list
        wh = 0
        haveornot = []
        for er in range (len(un_size)):
            if s1a != 0 :
                for sizea in s1a:
                    if sizea == un_size[er]:
                        where.insert(wh,str(er))
                        wh = wh +1
                        break
            if str(er) in where:
                haveornot.insert(er,True)
            else:
                haveornot.insert(er,False)

        t_f_s.insert(num,haveornot)
        ttf.insert(num,t_f_s)
        nm.insert(num,num)
        com.insert(num,list500_names[num])
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
```

```
            print 'The lists are ready in ', minutes, ' minutes'
            print 'The lists are ready in ', duration, ' seconds'
```

In [85]:
```
#Run the function dimensions_per_company
dimensions_per_company (un_size,list500_names)
```

```
The lists are ready in  0.1  minutes
The lists are ready in  3.699  seconds
```

In [86]:
```
#Create an initial dataframe where we will add the sizes later on
d6 = {'company' : pd.Series(com, index=[nm])}
sizess = pd.DataFrame(d6)
sizess.head(3)
```

Out[86]:

	company
0	Walmart
1	Exxon Mobil
2	Apple

In [89]:
```
#Now we want to break the variable t_f_s
#in order to add the columns to the dataframe
#Finally we create the data frame with the elements we found
def final_dimensions_dataframe (un_size,t_f_s,list500_names):
    from time import time
    # I used it to see how much time it does to run the function
    start = time ()
    for q in range(len(un_size)):
        names = un_size[q]
        var = []
        for num in range(len(list500_names)):
            a = t_f_s[num]
            var.insert(num,a[q])
        sizess[names] = pd.Series(var, index=sizess.index)
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'
```

In [90]:
```
#Run the function final_dimensions_dataframe
final_dimensions_dataframe (un_size,t_f_s,list500_names)
```

```
The lists are ready in  0.0  minutes
The lists are ready in  0.414  seconds
```

In [91]:
```
sizess.tail(3)
```

Out[91]:

	company	15x75	44x556	1x1	800x1200	24pxx133px	21pxx173p
497	NVR	True	True	True	True	True	Tru
498	Cincinnati Financial	True	True	True	True	True	Tru
499	Burlington Stores	True	True	True	True	True	Tru

	49x49	50x45	29x29	...	120x120	46x46	318x460	370x630	75x171	105x530
497	True	True	True	...	True	True	False	False	False	False
498	True	True	True	...	True	True	True	True	True	True
499	True	True	True	...	True	True	True	True	True	True

	781x1800	50x100	79x126	130x176
497	False	False	False	False
498	True	True	False	False
499	True	True	True	True

[3 rows x 706 columns]

In [92]:
```
#Now we would like to find the words in the text
#and the unique words of each html page
#First of all we need to have a dictionary with which
#we would check if the word we found truly exists
#The dictionary is available in the internet from a
#github acount from where we are going to read it
url_dictionary = "https://raw.githubusercontent.com/dwyl/english-words/mas
browser = urllib2.build_opener()
browser.addheaders = [('User-agent', 'Mozilla/5.0')]
response = browser.open(url_dictionary)
html_dictionary = response.read()
html_dictionary
dicti = str(html_dictionary)
#dicti
```

In [93]:
```
#dict_new = dicti.split("\\n")
dict_new = dicti.split("\n")
dict_new[49]
#the first 49 parts are not words so we have to remove them from the list
```

Out[93]: 'a1'

In [94]:
```
dict_final = []
df = 0
for i in range (50,len(dict_new)):
    forfinal = dict_new[i]
    forfinal = forfinal.replace("'","")
    dict_final.insert(df,forfinal)
    df = df + 1
dict_final[0]
#This is the original dictionary with which we will check each word
```

```
Out[94]: 'aa'

In [95]: #And now we will find each html file which words has inside
         empty = []
         wordsin = []
         ocin = []
         def html_which_word (list500_names):
             from time import time
             # I used it to see how much time it does to run the function
             start_t = time()
             for num in range(len(list500_sites)):
                     line = list500_sites[num]
                     if line == 0:
                         wordsin.insert(num,0)
                         ocin.insert(num,0)
                     else:
                         wordcount={}
                         simeiastiksis = ["/",".",",","=",">","<","?","|",":"
                                          ,"_","]","[","$","&","%","(",")","{"
                                          ,"}","'",";","\\","-","!","+","#","="
                                          ,"@","^","*","'"]
                         for ss in range(len(simeiastiksis)):
                             simeio = simeiastiksis[ss]
                             line = line.replace(simeio, " ")
                         for word1 in line.split():
                             word1 = word1.lower()
                             if word1 in dict_final:
                                 if word1 not in wordcount:
                                     wordcount[word1] = 1
                                 else:
                                     wordcount[word1] += 1
                         wordsin_local = []
                         wl = 0
                         ocin_local = []
                         for k,v in wordcount.items():
                                 #print (k,v)
                                 wordsin_local.insert(wl,str(k))
                                 ocin_local.insert(wl,str(v))
                                 wl = wl + 1
                         wordsin.insert(num,wordsin_local)
                         # final list with all the words in each site
                         ocin.insert(num,ocin_local)
                         #final list with all the occurencies of the words in each
                     #print('The site', str(num +1), ' has been checked')
             end_t = time()
             total_t = round(end_t - start_t,3)
             total_ = round(total_t / 60,1)
             print('finished ',str(x) ,' sites in: ', str(total_),' minutes')
```

```
In [96]: html_which_word (list500_names)

('finished ', 'True', ' sites in: ', '197.6', ' minutes')


In [97]: #Create the dataframe for the words and unique words
         d7 = {'company' : pd.Series(list500_names, index=[nm])}
         wordss = pd.DataFrame(d7)
         wordss.head(3)

Out[97]:        company
         0      Walmart
         1  Exxon Mobil
         2        Apple

In [98]: #Create the two lists we will need in order to make the dataframe
         l1 = []
         l2 = []
         for num in range(len(list500_names)):
                 line = list500_sites[num]
                 if wordsin[num] == 0 :
                     l1.insert(num,'n/a')
                     l2.insert(num,'n/a')
                 else:
                     total_words = len(wordsin[num])
                     occurencies = ocin[num]
                     l1.insert(num,total_words)
                     count = 0
                     for a in occurencies :
                         if a == '1':
                             count = count + 1
                     l2.insert(num,count)
         wordss['total_words'] = pd.Series(l1, index=sizess.index)
         wordss['unique_words'] = pd.Series(l2, index=sizess.index)
         wordss.head(3)

Out[98]:        company total_words unique_words
         0      Walmart        1646          443
         1  Exxon Mobil         801          154
         2        Apple         397          123

In [99]: #In order to validate the html code we will use the w3 validator
         #We will validate each url and then we will open the url of the validatior
         #so as to extract the errors, the info warnings and the non-document-erro
         #First we create the empty lists we would use later on
         num_errors = []
         num_info_warnings = []
         num_non_doc = []
         nm = []
```

```
num_open_page = []
empty = ""
```

```
In [100]: #Then we create the function that will pull the informations we want
          def html_validation (list500_url,list500_names):
              from time import time # I used it to see how much time it does to run
              start = time ()
              for num in range(len(list500_names)):
                  line = list500_url[num]
                  url_check = "https://validator.w3.org/nu/?doc=https://" + line
                  browser = urllib2.build_opener()
                  browser.addheaders = [('User-agent', 'Mozilla/5.0')]
                  response = browser.open(url_check)
                  html_check = response.read()
                  html_check
                  check = str(html_check)
                  er = 0
                  err = 0
                  errr = 0
                  e = False
                  if check != empty:
                      e = True
                      soup = BeautifulSoup(check,"lxml")
                      o = 0
                      keyf = []
                      for row in soup.html.body.findAll('div'):
                          keyf.insert(o,row)
                          o = o + 1
                      #print(len(keyf),list500_url[num], "site number: ", str(num))
                      if len(keyf) != 0:
                              keyfin = str(keyf[2])
                              #the elements we need is in the 2nd div of the code
                              dol= re.findall('class="error"',keyfin)
                              er = er + len(dol)
                              doll= re.findall('class="info warning"'
                                               ,keyfin)
                              err = err + len(doll)
                              dolll= re.findall('class="non-document-error io"'
                                                ,keyfin)
                              errr = errr + len(dolll)
                  num_errors.insert(num,er)
                  num_info_warnings.insert(num,err)
                  num_non_doc.insert(num,errr)
                  nm.insert(num,num)
                  num_open_page.insert(num,e)
              end = time ()
              duration = round (end - start, 3)
              minutes = round (duration /60, 1)
```

```
        print 'The lists are ready in ', minutes, ' minutes'
```

In [101]:
```
#Now we will run the function we created
html_validation (list500_url,list500_names)
```

```
The lists are ready in  37.0  minutes
```

In [102]:
```
#After the checks we will create the dataframe with the informations we w
d8 = {'company' : pd.Series(list500_names, index=[nm]),
      'The_page_opened' : pd.Series(num_open_page, index=[nm])
      ,'number_of_errors' : pd.Series(num_errors, index=[nm]),
      'number_of_warning' : pd.Series(num_info_warnings, index=[nm])
      ,'non-document-error' : pd.Series(num_non_doc, index=[nm])}
html_val = pd.DataFrame(d8)
html_val.head(3)
```

Out[102]:

	The_page_opened	company	non-document-error	number_of_errors	number_of_warning
0	True	Walmart	0	879	2
1	True	Exxon Mobil	0	55	29
2	True	Apple	0	14	6

In [103]:
```
#The next step is to take some informations from the fortune 500 site for
#In order to achieve that we should open the pages for each one of the si
#Since there is a pattern in the way the pages are named it shouldn't be
#Firstly we should create the pattern with which we will download the pag
#By running the code we can see that the names of each comany are not
#written exactly as we have saved them
#So we do need to alter the names first in order for the below function t
```

In [104]:
```
#creating a new list with alterations in order for the names
#to match the ones that fortune 500 uses so that we can download the html
list_company_name_new = []
for num in range (0,500):
    cn = list_company_name[num]
    cn = cn.replace(" ", "-")
    cn = cn.replace("&", "")
    cn = cn.replace("'", "")
    cn = cn.replace(".", "-")
    cn = cn.replace("amp;", "")
    company = cn.lower()
    list_company_name_new.insert(num,cn)
```

In [105]:
```
fortune_pages = []
def fortune500 (list_company_name_new):
```

```
    from time import time # I used it to see how much time it does to run
    start = time ()
    for num3 in range (0,500):
        i = str (num3 +1)
        companyname =  list_company_name_new[num3]
        browser = urllib2.build_opener()
        #because i work from different computers with different
        #pyhton version some commands are not recognizable in each versio
        browser.addheaders = [('User-agent', 'Mozilla/5.0')]
        site_fortune = "http://beta.fortune.com/fortune500/"+companyname+
        page_fortune = browser.open(site_fortune)
        html_fortune = page_fortune.read()
        #print("fortune page for company: ", list_company_name_new[num3],
        fortune_pages.insert(num3, html_fortune)
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'
```

In [106]:
```
#Run the function we created
fortune500 (list_company_name_new)
```

```
The lists are ready in  21.6  minutes
The lists are ready in  1295.917  seconds
```

In [107]:
```
#Now that we have opened the url we are going to extract
#some informations that we need from them
#In order to do that initially we have to create
#the variables we will need
keyf =[]
per =[]
rev_dol = []
rev_per = []
prof_dol = []
prof_per = []
assets_dol = []
assets_per = []
tse_dol = []
tse_per = []
mar_dol = []
mar_per = []
market = []
nm = []
ln = []
urln = []
empty = []
```

```
In [108]: def fortune_metrics (list_company_name,list_company_website):
              x = 0
              for n in range (0,500):    #we put 25 for testing
                  nm.insert(x,x)
                  ln.insert(x,list_company_name[n])
                  urln.insert(x,list_company_website[n])
                  files = fortune_pages[x]
                  soup = BeautifulSoup(files,"lxml")
                  o=0
                  for row in soup.html.body.findAll('tbody'):
                      keyf.insert(o,row)
                      o=o+1
                  keyfin = keyf[0]
                  #the elements we need is in the first tbody of the code
                  data = keyfin.findAll('td')

                  one = str(data[0])
                  # revenue
                  two = str(data[1])
                  # revenue in dollars we need to extract this
                  revdol= re.findall('>\$(.+?)</td>',two)
                  #we keep only the numbers
                  if revdol[0] != empty:
                      w = revdol[0]
                      a = w.replace("[", "")
                      r = a.replace("]","")
                      rev_dol.insert(x,r)
                  else:
                      rev_dol.insert(x,'not available')
                  tria = str(data[2])
                  # revenue in percentage we need to extract this as well
                  revper= re.findall('>(.+?)%</td>',tria)
                  #we keep only the numbers
                  if revper != empty:
                      w = revper[0]
                      a = w.replace("[", "")
                      r1 = a.replace("]","")
                      rev_per.insert(x,r1)
                  else:
                      rev_per.insert(x,'not available')
                  four = str(data[3])    # profit
                  five = str(data[4])
                  # profit in dollars we need to extract this
                  profdol= re.findall('>\$(.+?)</td>',five)
                  #we keep only the numbers
                  if profdol != empty:
                      w = profdol[0]
                      a = w.replace("[", "")
```

```
        p = a.replace("]","")
        prof_dol.insert(x,p)
    else:
        prof_dol.insert(x,'not available')
    six = str(data[5])
    # profit in percentage we need to extract this as well
    profper = re.findall('>(.+?)%</td>',six)
    #we keep only the numbers
    if profper != empty:
        w = profper[0]
        a = w.replace("[", "")
        p1 = a.replace("]","")
        prof_per.insert(x,p1)
    else:
        prof_per.insert(x,'not available')
    seven = str(data[6]) #assets
    eight = str(data[7]) #assets in dollars we need to extract this
    assetsdol= re.findall('>\$(.+?)</td>',eight)
    #we keep only the numbers
    if assetsdol != empty:
        w = assetsdol[0]
        a = w.replace("[", "")
        ass = a.replace("]","")
        assets_dol.insert(x,ass)
    else:
        assets_dol.insert(x,'not available')
    ten = str(data[9]) #Total Stockholder Equity ($M)
    eleven = str(data[10])
    #Total Stockholder Equity ($M) in dollars we need to extract this
    tsedol= re.findall('>\$(.+?)</td>',eleven)
    #we keep only the numbers
    if tsedol != empty:
        w = tsedol[0]
        a = w.replace("[", "")
        ts = a.replace("]","")
        tse_dol.insert(x,ts)
    else:
        tse_dol.insert(x,'not available')
    thirteen = str(data[12]) # market value
    fourteen = str(data[13])
    # market value in dollars we need to extract this
    mardol= re.findall('>\$(.+?)</td>',fourteen)
    #we keep only the numbers
    if mardol != empty:
        w = mardol[0]
        a = w.replace("[", "")
        mar = a.replace("]","")
        mar_dol.insert(x,mar)
```

```
                else:
                    mar_dol.insert(x,'not available')
                x = x + 1
            print "The function is complete!"
```

In [109]:
```
fortune_metrics (list_company_name,list_company_website)
```

```
The function is complete!
```

In [110]:
```
d9 = {'company' : pd.Series(ln, index=[nm]),
      'Revenues $' : pd.Series(rev_dol, index=[nm]),
      'Revenues %' : pd.Series(rev_per, index=[nm]),
      'Assets $' : pd.Series(assets_dol, index=[nm]),
      'Total Stockholder Equity $' : pd.Series(tse_dol, index=[nm]),
      'Market value $' : pd.Series(mar_dol, index=[nm])}
fort500 = pd.DataFrame(d9)
fort500.head(3)
```

Out[110]:

	Assets $	Market value $	Revenues $	Revenues %	Total Stockholder Equity
0	199,581	215,356	482,130	-0.7	80,54
1	336,758	347,129	246,204	-35.6	170,81
2	290,479	604,304	233,715	27.9	119,35

	company
0	Walmart
1	Exxon Mobil
2	Apple

In [111]:
```
fort500.merge(html_val, left_on='company', right_on='company', how='outer
fort500.head(3)
```

Out[111]:

	Assets $	Market value $	Revenues $	Revenues %	Total Stockholder Equity
0	199,581	215,356	482,130	-0.7	80,54
1	336,758	347,129	246,204	-35.6	170,81
2	290,479	604,304	233,715	27.9	119,35

	company
0	Walmart
1	Exxon Mobil
2	Apple

In [112]:
```
result = pd.merge(fort500, html_val, how='inner', on=['company', 'company
result2 = pd.merge(social_media, fre, how='inner', on=['company', 'compar
result3 = pd.merge(wordss, sizess, how='inner', on=['company', 'company']
result4 = pd.merge(images_types, loading_time, how='inner', on=['company'
result5 = pd.merge(result, sites_links, how='inner', on=['company', 'comp
result6 = pd.merge(result5, result2, how='inner', on=['company', 'company
result7 = pd.merge(result6, result3, how='inner', on=['company', 'company
final3 = pd.merge(result7, result4, how='inner', on=['company', 'company'
final3.head(3)
```

Out[112]:

	Assets $	Market value $	Revenues $	Revenues %	Total Stockholder Equity
0	199,581	215,356	482,130	-0.7	80,54
1	336,758	347,129	246,204	-35.6	170,81
2	290,479	604,304	233,715	27.9	119,35

	company	The_page_opened	non-document-error	number_of_errors	\
0	Walmart	True	0	879	
1	Exxon Mobil	True	0	55	
2	Apple	True	0	14	

	number_of_warning	...	.dib	.gif	.jpe	.jpeg	.jpg	.png	.tif	.ti
0	2	...	0	55	153	153	63	46	10	
1	29	...	0	1	0	0	16	2	4	
2	6	...	0	9	0	0	0	2	0	

	total images	loading time
0	480	0.449
1	23	5.376
2	11	0.021

[3 rows x 740 columns]

In [113]:
```
final3.to_csv('total_500.csv', sep=';')
```

In [114]:
```
data500 = pd.read_csv("total_500.csv", sep=';')
```

In [115]:
```
data500.head(3)
```

Out[115]:

	Unnamed: 0	Assets $	Market value $	Revenues $	Revenues %	\
0	0	199,581	215,356	482,130	-0.7	
1	1	336,758	347,129	246,204	-35.6	
2	2	290,479	604,304	233,715	27.9	

	Total Stockholder Equity $	company	The_page_opened	non-document-e
0	80,546	Walmart	True	
1	170,811	Exxon Mobil	True	
2	119,355	Apple	True	

	number_of_errors	...	.dib	.gif	.jpe	.jpeg	.jpg	.png	.tif	.ti
0	879	...	0	55	153	153	63	46	10	
1	55	...	0	1	0	0	16	2	4	
2	14	...	0	9	0	0	0	2	0	

	total images	loading time
0	480	0.449
1	23	5.376
2	11	0.021

[3 rows x 741 columns]

In [ ]: