CTRimages

CS4624 Capstone Project

Virginia Tech, Blacksburg

Kurban Hakimov, Andrew Hartwell, RobertUlmet

Clients: Kiran Chitturi, Seungwon Yang

Date: May 8th, 2013

Table of Contents

Abstract.....3

User's Manual.....4

Developer's Manual.....5

Requirements.....5

Concepts.....5

Data Inventory.....7

Screen Dumps.....7

Lessons Learned.....8

Timeline.....8

Observations.....8

Acknowledgements.....9

References.....9

Abstract

The goal of this project focused on the creation of a computer program capable of dissecting web pages pertaining to significant historical events (namely human and natural disasters) and extracting the photographs pertaining to said events to be archived and displayed in online galleries. The preservation of historical media isn't a new concern, but the challenge has never been on the magnitude that we face today. The sheer amount of photographs being generated daily presents an impossible task for manual recording, and thus the idea to use a computer program to do the work automatically was born.

The project consisted of three distinct phases. The first phase was to manually parse the Crisis, Tragedy, and Recovery Network database web pages pertaining to the "Brazilian Nightclub Fire," and to collect raw data of the related photographs embedded in the web pages. The purpose of collecting this sample data was to 1. Help construct an algorithm for our computer program to extract photographs from HTML pages and 2. Help us to best design and choose the right tools for our image gallery.

The second phase was to construct the actual computer program to process web pages and extract photographs. Exactly how we produced this program is described in detail in the developer's manual. The third and final phase was to construct our online gallery that would hold the extracted photographs of these historical events. Drupal provided the framework we needed for this and for display purposes our gallery was filled with the parsed images of the nightclub fire event. Again we go into more detail on our design decisions in the developer's manual. We hope that this project will serve as a shining example of what must be done to preserve history in the information age.

User's Manual

How to use the script:

script usage:
type “ctrfilter”

The “ctrfilter” bash script will find all .htm and .HTML files in the directory (and subdirectories) of the **ctrfilter** file and run the proper scripts to download the pictures.

The parser script takes in one HTML page at a time, and outputs any suitable urls and alt tag texts to their respective text files.

After all HTML files have been parsed, the filter will apply the filter to all images in the output text files from the parser. The filter will output to a directory called album. The filter script downloads the images that pass the filter, so it requires an internet connection to run. **The filter criteria is set in the *filter_images.py* file.**

To see a demo of the gallery: http://ctrimages.x10.mx/

How to use the gallery:

Documentation on how to install and use the gallery can be found on its home page:

http://vacilando.net/bg

Developer's Manual

Requirements:

Python 2.7.x - http://www.python.org/download/

Python Imaging Library - http://www.pythonware.com/products/pil/

Concepts

From the start of this project our clients had a very clear set of goals they wished for us to accomplish. We were to first write a computer program capable of processing HTML pages for their relevant image files. We were then to create an online gallery to hold these extracted images that would provide a lasting photographic record of the event. To get us going we were given a dump of HTML documents from the Crisis, Tragedy, and Recovery Network database pertaining to the Brazilian nightclub fire. We were also given the suggestion to use the python library Beautiful Soup (http://www.crummy.com/software/BeautifulSoup/) to help with the parsing of the HTML files. The rest of the challenge was left to us using any tools or methods we deemed necessary.

Phase 1 was a manual examination of these web pages in order to understand how relevant media is usually organized in an HTML page, as well as a to determine some of the properties that would be suitable for an online gallery. We organized this data by hand into an excel spreadsheet (link in Data Inventory). We weren't sure which data would become relevant but we decided it was best to record anything we felt could possibly be relevant in the design of both our script and the gallery.

The data collected included the URL, title and alt tags, width and height of the image, and physical location within the web page. We thought it would be useful to use the location information, but this information was better suited for a much more advanced script that was really not possible within the constraints of our project.

Phase 2 began with us first trying to use Beautiful Soup to parse the HTML pages so that we could find all the images in a file. After running into quite a few complications with Beautiful Soup such as returning parse errors for certain types of HTML files we turned to a simpler approach using regular expressions. We used the regular expressions to locate image tags and then grabbed the proceeding URLs and alt tags.

The next step was to create a filter that would screen out irrelevant images from the parser's output. This was the part that we struggled the most with because our examination of the images in our test dump of HTML pages revealed that we had very little metadata to work with. There are little to no restrictions on what kind of information can be put in image tags, and finding trends in relevant images that would absolutely disallow irrelevant ones was no easy feat.

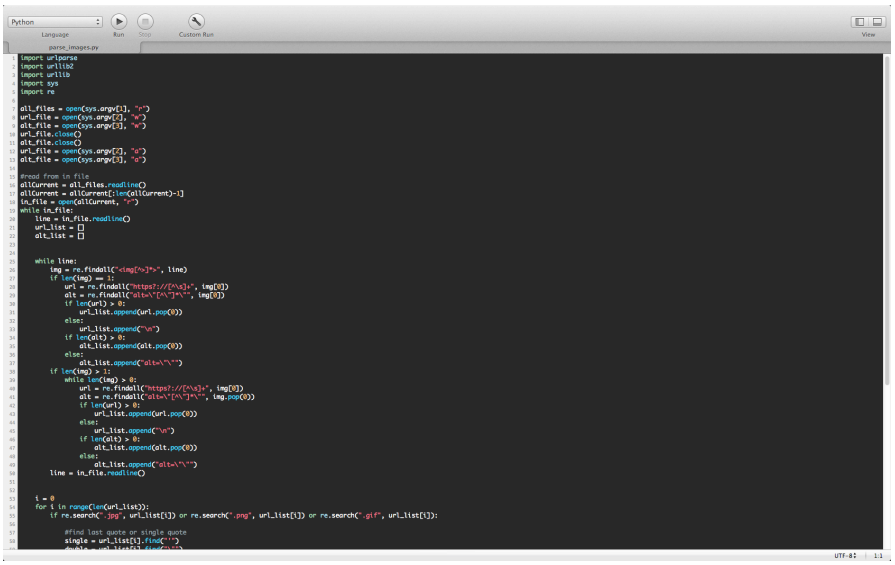
After examining what made images suitable and not for our gallery, we decided to filter by URL, file type, dimensions, and keywords in the alt tag. We then spent some time testing our filter by trial and error using different keywords and banned URLs to find what gave the best results. With our current configuration images must be a jpg, png or gif, they must have dimensions greater than 150 pixels, they must not be hosted by any of our list of common ad hosting servers and they must have an alt tag that contains at least 2/3 of our keywords. We made sure all of these criteria were easily customizable so that future galleries could be customized to produce the best results for any particular event. After an image passes all of the above criteria, it is downloaded and stored in an album folder in preparation for being added to our gallery. Because we used a very strict filter, most images fail at least one of our criteria, so we only accept a small percentage of images. Out of the ~3200 images that were in our original sample set, only 90 passed the filter. However, because the filter is so strict, it removes all irrelevant images. The only unresolved problem we had was that we could not remove duplicate images because there is no way to tell the content of two images based on the metadata alone. Therefore, the script has no way of telling if one url contains the same image as a different one.

Phase 3 initiated with a large amount of research and experimentation with existing Drupal galleries and how we could combine them with our Python scripts. After installing and thoroughly testing every Drupal gallery we could get our hands on we settled on a module called Brilliant Gallery (http://drupal.org/project/brilliant_gallery). Combining Brilliant Gallery with the module Lightbox2 (http://drupal.org/project/lightbox2) gave us the results we were looking for – our script would output the relevant images into a folder and Brilliant Gallery would pull the images from this folder and create a gallery. For display purposes we then filled our gallery with the images extracted by our script and the settings we used for Phase 2. The gallery can be viewed at http://ctrimages.x10.mx.

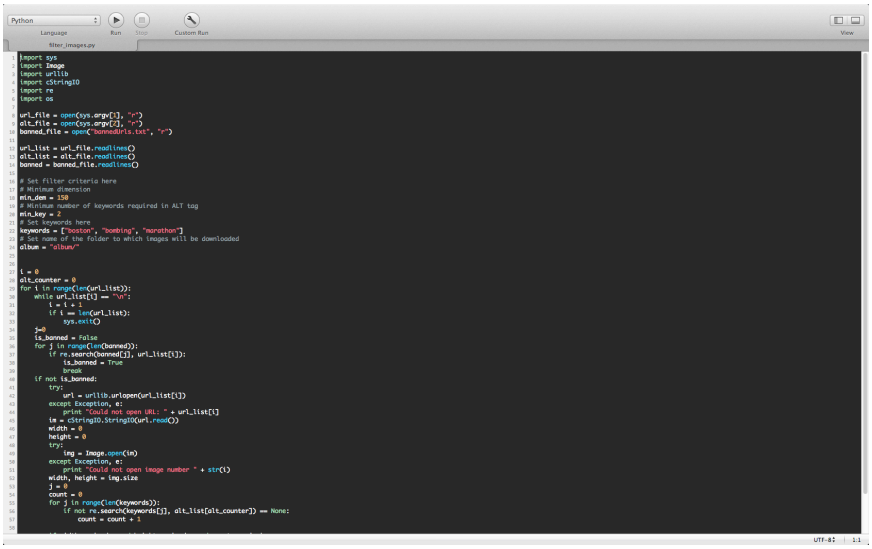
Data Inventory

The sample compilation of web pages given to us by the client is contained in https://www.dropbox.com/sh/3we2gyomd2n6grk/4cZrkQjMTB. This will give you access to all of the same data we were given at the outset of the project. The documentation of the image properties we extracted from the web pages can be viewed in the file https://docs.google.com/spreadsheet/ccc?key=0AgB9NzrwlMoHdEJWMnNpZk52YkVtbVRzUGhiakNsV2c&usp=sharing. This will give you exactly the information that we obtained from the web pages for use in designing our program as well as our gallery. The completed gallery can be viewed at http://ctrimages.x10.mx/?q=node/3.

Screen Dumps



```
import urlparse
import urllib2
import urllib
import sys
import re

all_files = open(sys.argv[1], "r")
url_file = open(sys.argv[2], "w")
alt_file = open(sys.argv[3], "w")
url_file.close()
alt_file.close()
url_file = open(sys.argv[2], "a")
alt_file = open(sys.argv[3], "a")

#read from in file
allCurrent = all_files.readline()
allCurrent = allCurrent[:len(allCurrent)-1]
in_file = open(allCurrent, "r")
while in_file:
    line = in_file.readline()
    url_list = []
    alt_list = []


    while line:
        img = re.findall("<img[^>]*>", line)
        if len(img) == 1:
            url = re.findall("https?://[^\s]+", img[0])
            alt = re.findall('alt=\"[^\"]*\"', img[0])
            if len(url) > 0:
                url_list.append(url.pop(0))
            else:
                url_list.append("\n")
            if len(alt) > 0:
                alt_list.append(alt.pop(0))
            else:
                alt_list.append("alt=\"\"")
        if len(img) > 1:
            while len(img) > 0:
                url = re.findall("https?://[^\s]+", img[0])
                alt = re.findall('alt=\"[^\"]*\"', img.pop(0))
                if len(url) > 0:
                    url_list.append(url.pop(0))
                else:
                    url_list.append("\n")
                if len(alt) > 0:
                    alt_list.append(alt.pop(0))
                else:
                    alt_list.append("alt=\"\"")
        line = in_file.readline()


    i = 0
    for i in range(len(url_list)):
        if re.search(".jpg", url_list[i]) or re.search(".png", url_list[i]) or re.search(".gif", url_list[i]):

            #find last quote or single quote
            single = url_list[i].find("'")
```



```
import sys
import Image
import urllib
import cStringIO
import re
import os

url_file = open(sys.argv[1], "r")
alt_file = open(sys.argv[2], "r")
banned_file = open("bannedUrls.txt", "r")

url_list = url_file.readlines()
alt_list = alt_file.readlines()
banned = banned_file.readlines()

# Set filter criteria here
# Minimum dimension
min_dim = 150
# Minimum number of keywords required in ALT tag
min_key = 2
# Set keywords here
keywords = ["boston", "bombing", "marathon"]
# Set name of the folder to which images will be downloaded
album = "album/"


i = 0
alt_counter = 0
for i in range(len(url_list)):
    while url_list[i] == "\n":
        i = i + 1
        if i == len(url_list):
            sys.exit()
    j=0
    is_banned = False
    for j in range(len(banned)):
        if re.search(banned[j], url_list[i]):
            is_banned = True
            break
    if not is_banned:
        try:
            url = urllib.urlopen(url_list[i])
        except Exception, e:
            print "Could not open URL: " + url_list[i]
        im = cStringIO.StringIO(url.read())
        width = 0
        height = 0
        try:
            img = Image.open(im)
        except Exception, e:
            print "Could not open image number " + str(i)
        width, height = img.size
        j = 0
        count = 0
        for j in range(len(keywords)):
            if not re.search(keywords[j], alt_list[alt_counter]) == None:
                count = count + 1
```

Criteria
File type (.jpg, .png, .gif)
Banned urls
Image size (width > min && height > min)
ALT tag text (alt tag must contain keywords)

Features
Customizable filter criteria
No manual filtering required
Filters all images in directory
100% accuracy

Lessons Learned

Timeline

February - Initial Drupal set-up and image documentation

March - Filter brainstorming and script parser writing

April - Finish parser script and write filter script

May - Perfect filter and improve Drupal gallery

Observations

Phase 1: We noticed that image tags were used in a multitude of ways depending on the host. Some would use the alt tag only for the image title, while others would use alt and title for the image header. This is why we decided to use the alt tag rather than the title

tag: many hosts would leave it blank, and when they did fill it in it usually was an exact copy of the title tag. When it came to acceptable image files we found that location within the HTML played some role whether the image was relevant or not. Almost all images coupled in the same section of the title story were relevant images. Also any sections starting with "related" such as related photos, related videos, related media, and related stories contained acceptable images. This information proved to be too in-depth for our simple parser design, and will have to be kept in mind for someone wishing to design a much more sophisticated parser.

Phase 2: After research we found that there are many third party modules in Python that worked with Images. The first we tried was Beautiful Soup which after extensive testing did not perform as we had expected. We ended up using the PIL library as it provided the tools we were looking for and worked consistently with our Regular Expressions.

Phase 3: The first thing we noticed was that there was a plethora of image gallery modules available for Drupal. We tested many of them thoroughly but had not found one we liked until we tried Brilliant Gallery with the Lightbox2 module installed. Our clients may decide to use another gallery but Brilliant Gallery has our recommendation.

Acknowledgements

We would like to thank our wonderful clients Kiran Chitturi and Seungwon Yang for creating such an important and inspired project for our team to tackle, as well as always being flexible and patient with our schedules and needs along the way.

We would also very much like to thank Dr. Edward Fox for hosting this course and guiding our work as best he can, as without such a motivated and supportive instructor this project could have been a painful experience rather than the inspired endeavor that it was.

References

PIL Documentation - http://www.pythonware.com/library/pil/handbook/index.htm

Brilliant Gallery Documentation - http://vacilando.net/bg

If the URL has moved the user should simply use a search engine and search for the terms listed above.