Due: 03.May.2015 23.55 PM

CMPE 230 - Spring 2015 Project 2

Introduction (Due Date: 3 May 2015 23:55 PM)

In this project you are going to implement the scripts required for a basic image gallery with tagging abilities. The system will be given one or more images and in the end it will produce lists of the images with the appropriate tags.

Requirements

In order to complete this project you will need the following:

1- **A Linux Machine** (ex. Ubuntu) For some reason if you use another operating system for development, make sure the scripts that you create also work correctly on Ubuntu.
2- An Amazon AWS account
   a. http://aws.amazon.com
3- AWS EC2 command line interface tools
   a. http://docs.aws.amazon.com/AWSEC2/latest/CommandLineReference/set-up-ec2-cli-linux.html
4- An imagga.com API user.
   a. Go to https://imagga.com/auth/signup
   b. Sign-up for a free (named hacker) account. The code related to imagga will be provided for you with the description. You will need your own imagga API keys.
   c. Go to https://imagga.com/profile/dashboard (You can see the API keys here)
5- Perl, WWW::Curl package for Perl (On Ubuntu you can do the following to install it.)
   a. sudo apt-get install libcurl4-gnutls-dev
   b. sudo cpan WWW::Curl

Project Details

Consider the scenario that I, as a user, have multiple photographs on my local computer. I want to categorize these images so that I can easily find what I want and see some other examples from the internet. For instance, I want to be able to get the names of the photographs that are related to the sky as well as see what the internet has to offer on this topic. The problem is that I may have too many photographs to go over all of them and to the categorization manually. Therefore an automation is needed to make this task feasible.

As mentioned before you are going to implement a basic image gallery with tagging abilities. Basically, the user will point your system to a file or a directory and it will produce labels or tags for the photograph(s) and store those labels on the file system. Furthermore it will also produce some "related images" for each of the labels. As an added benefit you will also store the images and the labels on a central server. This way I will be able to access the images anywhere I want in accordance with the cloud storage scheme.

Expected Output

You will need to implement several components, as separate Perl scripts. The names of the Perl scripts should be as defined here. The command line arguments that each script can get are described in the Usage sections.

1) A script to start an Amazon AWS instance
   a) Name: startaws.pl
2) A script to upload file(s) pointed to by the user to the AWS instance
   a) Name: uploadaws.pl
   b) Usage1: uploadaws.pl filename1 filename2 ... filenameN -o awsdirname

        i) This should upload all those files to the awsdirname directory within the started AWS instance
    c) Usage2: uploadaws.pl dirname -o awsdirname
        i) This should upload all the files in the local directory (including subdirectories) to the awsdirname directory within the started AWS instance.
3) A script to produce top n image labels (on the AWS instance using the imagga API)
    a) Name: labelimage.pl
    b) Usage: labelimage.pl 5 image_name
        i) This will run the imagga script with the image named "image_name" and return top 5 tags ordered by their confidence values.
4) A script to search for related images to a given tag and return top n results.
    a) Name: searchlabels.pl
    b) Usage1: searchlabels.pl 12 "sky"
        i) This will return the URL's of 12 different images with the keyword "sky" using any search engine.
5) A script to download file(s) pointed to by the user from the AWS instance to your local computer.
    a) Name: downloadaws.pl
    b) Usage1: downloadaws.pl filename1 filename2 ... filenameN -o localdir
        i) This should download all those files from the started AWS instance to the local directory called localdir
    c) Usage2: downloadaws.pl dirname -o localdir
        i) This should download all the files in the directory (including subdirectories) from the AWS instance to the local directory localdir
6) A script to stop an AWS instance
    a) Name: stopaws.pl
7) A script that enables the user to control the gallery. It can get various different types of options. The list image options should list the full paths of the images.
    a) Name: gallery.pl
    b) Usage 1: gallery.pl --listlabels
        i) This will return all the unique labels that your current image gallery contains.
    c) Usage 2: gallery.pl --listimages
        i) This will list every image currently within your gallery.
    d) Usage 3: gallery.pl --listimages "labelname"
        i) This will list every image tagged with the label "labelname" currently within your gallery.
    e) Usage 4: gallery.pl --listimages --similar image_name
        i) This will list every image tagged with the same label(s) that the image with "image_name" is tagged.
    f) Usage 5: gallery.pl --listsamples "labelname"
        i) List all the sample images with the label "labelname" that the user has downloaded.
    g) Usage 6: gallery.pl --add "dirname"
        i) Adds the files in "dirname" to the gallery.

Important Note

Since we are using Amazon and Imagga API's for this project, you shouldn't hard code the required API keys and Secret Keys in your scripts. The scripts that require those should read it from an external file. The file should be a text file (you may also use a .pl file and include it in the other required scripts) and its format should be as follows.

```
AWS_ACCESS_KEY=<your-aws-access-key-id>
AWS_SECRET_KEY=<your-aws-secret-key>
IMAGGA_ACCESS_KEY=<your-imagga-access-key-id>
IMAGGA_SECRET_KEY=<your-imagga-secret-key>
```

Implementation Details

For 1 and 6 you will need to use Amazon AWS command line interface and your account details. You will have to keep track of your started instances and their connection details for use in the other scripts. You can for example have an additional script to get running instance id's.

For 2 and 5 you should also use tools like SFTP and/or SCP. The important thing to keep in mind here is that the downloaded results might have to be added to an existing gallery. For example, the user might have a directory which contains hundreds of images on which they have used the service and thus have a labels list file and images list file. Then they add one more image to the directory and run the service on this single image. This script should merge the new result with the previous results and NOT overwrite them.

For 3 you will be provided with a script that generates tags (with confidence values) for a single given image. Using this script you are expected to produce labels for the images that were uploaded by the user. The output of this script will be two text files: a file that lists the labels in the gallery, a file that lists the images and their labels in the gallery. Keep in mind that for a single image tens of tags may be given by the generator script, you have to select how many of them you will use, depending on confidence levels.

For 4 you will write a script that searches for images for each label you have in your gallery. You may use your favorite search engine, however using well-known engines such as Google, Yandex and Bing is suggested. You may use an API or the regular web page to perform the search. Once the search is complete grab the top n images related to the label by either downloading (downloading them may be required for the last project that you are going to implement using QT) them or their URL's. We will call these n images "samples".

For 7 your script will be able to perform several tasks:

List Labels : List the labels in the gallery

List Images : Print the full paths of the images in your gallery.

List Images with label < label > : Print the full paths of the images < optionally with the label "label" >

List Images with labels similar to < image > : Print the full paths of the images < optionally with the same label(s) as "image" >

List Samples with the label < label > : Print the full paths of the samples < optionally for the "label">

Add < path > : Add the files in directory "path" to the gallery.

You may use two separate files for your gallery (in addition to downloaded sample images). First one may list your gallery images with the tags generated. The second one may include the tags and the paths to the downloaded sample images. It is up to you how you implement this part, but the functionality should be as follows. After running

```
gallery.pl --add "dirname"
gallery.pl --add "dirname2"
gallery.pl --listimages --similar testimg.jpg
```

you should get the list of all images that have similar labels to testimg.jpg that were within the folders dirname or dirname2.

Notes:

1- You will be given an API Key and an API Secret once you sign up to imagga.com. You will need to insert your credentials into the script (imagesearch.pl) given to you for 3.

2- You do not need to display the images for this part of the project. You need to print their names to the command line.
3- If required feel free to produce additional files in your scripts (keep the number reasonable though)
4- Since we are using the free option for Imagga, you shouldn't use the service during this project with very large number of files (2000 image limit per month).
5- The free API service also doesn't support tagging with multiple files at the same time. Therefore to tag multiple files (files within a folder for example) you need to run the script within a loop.

What to submit:

1- Your (well written and commented) code.
2- A brief report that describes the implementation you have done and where you discuss why we use an Amazon AWS instead of doing the labeling on the local computer. Keep this report BRIEF: no more than 2 pages, 1 page is preferred.

IMPORTANT:

It is highly probable that you will use the scripts you create for this project on the last project as well. Therefore, do your best to complete this project.