Multimedia and User Credibility Knowledge Extraction

Deliverable 1.2
Report on New Data Collection

Document Information

Delivery date:	31/09/2013
Lead partner:	UAIC
Author(s):	Lenuta Alboaie, Mihai Lupu, Adrian Popescu, Adrian Iftene, Pinar Sahin, Allan Hanbury
Participant(s):	TUW, CEA, Bilkent
Workpackage:	1
Workpackage title:	Data collection
Workpackage leader:	UAIC
Dissemination Level:	PU – Public

History of Versions

Version	Date	Status	Author (Partner)	Description/Approval Level
0.1	18/06/2013	Draft	UAIC	First draft
0.2	29/09/2013	Draft	UAIC	Second Draft

Abstract

This document identifies the existing data collections for image search and associated meta-data. It also looks beyond the set of existing collections at tools or resources that will allow us to collect the data necessary for the project.

Deliverable Context
/*CHIST-ERA Full Proposal Form page 12 of 39 */

WP1 – Data Collection
The objectives of this work package are to create a data collection framework which will fuel concerned work packages (WP2 to WP6) with necessary raw data and, given that many of the data used are personal, to establish clear ethical rules concerning the information processing in order to preserve users' privacy.

Task T1.2 - New Data Collection -
The complex nature of the project objectives requires the mobilization of different multimedia data sources in order to mine all necessary user-related knowledge. Since the data access policies of social networks change frequently, the first task will be to evaluate which are the available sources at the beginning of the project. With over 6 billion photos available (http://news.softpedia.com/newsImage/Flickr-Boasts-6-Billion-
Photo-Uploads-2.jpg/), Flickr is one of the largest photo repositories on the Web and will constitute the main source of visual information used in the project. CEA LIST has extensive experience in crawling this data source and, will work in close collaboration with BILKENT in order to download images and textual metadata for as many as 1,000,000 Flickr users.
The output of this task is a data collection module that crawls data sources and pre-processes them into a common format, usable for the different information extraction tasks. Given the dynamic nature of social networks data, emphasis will be put on creating a data collection framework, which updates the information needed. The amount of data collected is calibrated in order for the consortium to test the extraction of new knowledge from large amounts of heterogeneous data.
UAIC will coordinate the data collection effort but, in order to speed up the process, it will be distributed among all partners. Interchangeable XML based formats will be used in this process.

Table of Contents

1. Introduction

The output of this task is a data collection module that crawls data sources and pre-processes them into a common format, usable for the different information extraction tasks. Because Flickr is one of the largest photo repositories on the Web, it is used in MUCKE as the main source of visual information.

In Section 2 we will present how information collection was a coordinate effort of all consortium members in order to speed the whole process. Details regarding the technical process are given in Sections 2.4 and 2.5. In Section 3 we describe the links between concepts (Wikipedia concepts) used for image annotation and the downloaded images in order to obtain a flexible corpus directory structure for MUCKE.

The data was collected using Flickr's public APIs and conforming with the Flickr Community Guidelines.

2. New data collection

The complex nature of the project objectives requires the mobilization of different multimedia data sources in order to mine all necessary user-related knowledge. With over 6 billion of photo uploads, Flickr is one of the largest photo repositories on the Web and will constitute our main source of visual information. Consortium members have extensive experience in crawling this data source and will download images and textual metadata for as many as 1,000,000 Flickr users. Given that Flickr contains a mix of personal and social relevant data, focus will be put on downloading the latter type, which is useful for information extraction tasks. UAIC coordinates the data collection effort but, in order to speed up the process, it will be distributed among all partners.

2.1. Architecture

The download process was planned as follows:

- CEA generated lists of files that need to be downloaded, and a copy of these lists resided on each machine that did the download;
- When a new list is provided, TUW updates the Task Distribution Service (TDS) to respond to requests for this list;
- For each list, the downloader queries the TDS and receives back a task id, and a pair of numbers indicating the start and end position in the list;
- Then, it passes these parameters to the download script (perl) provided by CEA and, after the download is over, creates an archive with the images indicated in the download task;
- After the download task is finished, the resulted archive is verified in order to ensure that no error has occurred and that it contains the 10000 images that had to be obtained in the corresponding task;
- When all download tasks are finished and verified, the archives from all four partners are collected and stored on the server provided by UAIC.

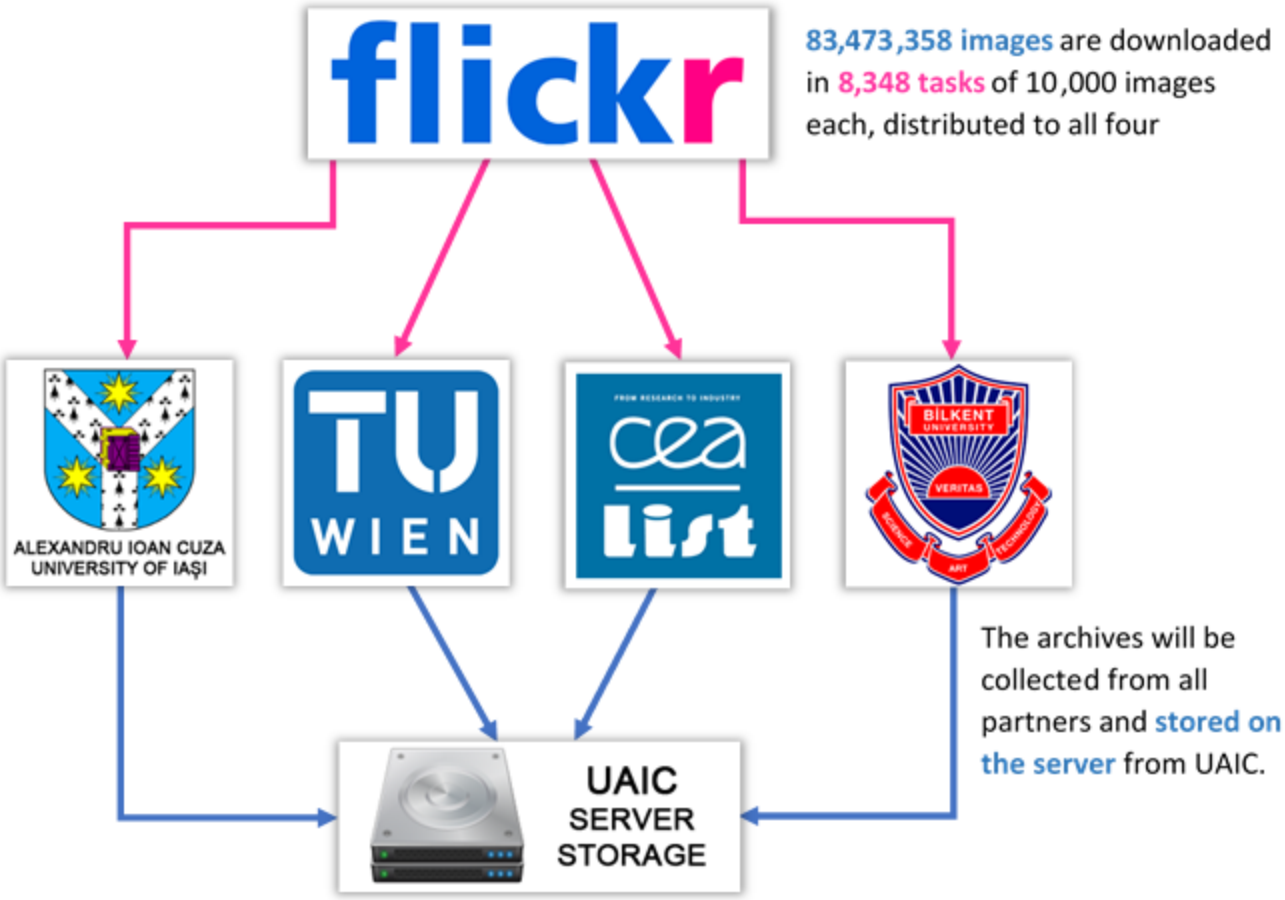


Figure 1. How the download process works

2.2. Data downloading

Each member of the consortium set up several machines to handle the download process. The list of files generated by CEA to be downloaded is called “imageListBigUnique”. It has a size of 5.95 GB and each line contains the name and the URL of an image to be downloaded from Flickr. This list was present on every machine and was used by the download scripts in order to find the images corresponding to a task. For example, the images for task 111 were found in the list at the lines numbered 1110000 through 1119999.

During the download process, some statistics were provided at http://stutomcat.ifs.tuwien.ac.at:8080/MuckeDownloadTasker in the form of two charts: one for ongoing and one for completed tasks. These charts were dynamically updated, thus enabling us to check our tasks and see the overall status of the download process at any time.

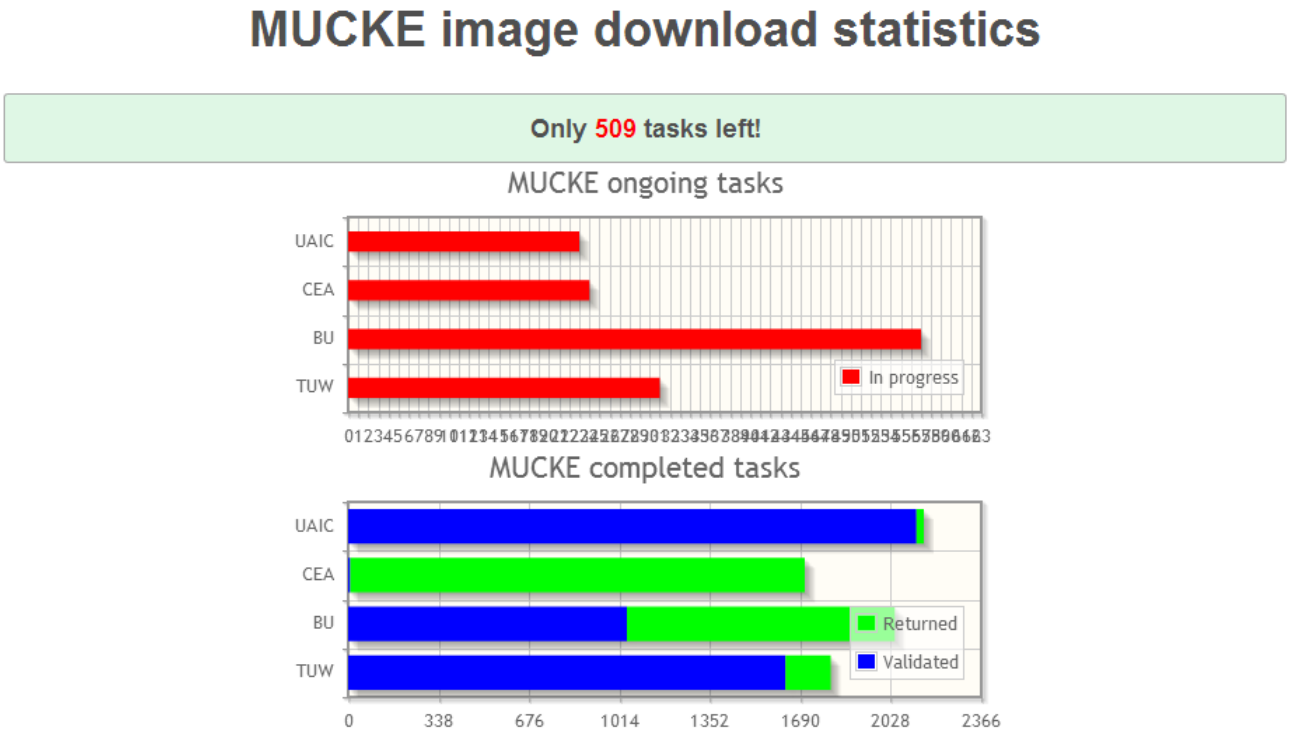


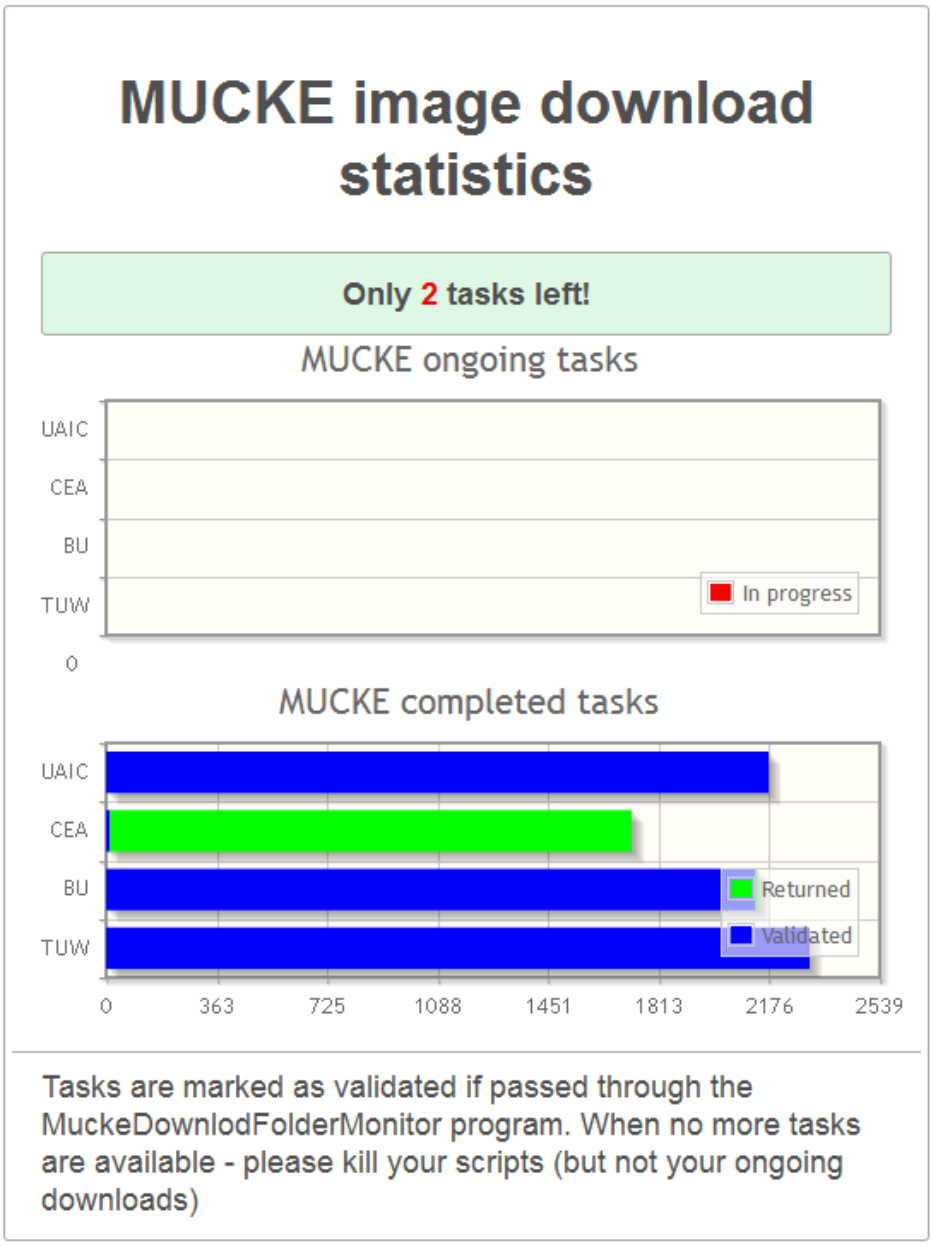
Figure 2. Sample of image download statistics during the download process.

2.1. Data source

Flickr is an image and video hosting website, web services suite, and online community. It boasts more than 6 billion images being hosted on its servers. Flickr provides a filtering system that enables its members to mention the types of the photographs they upload, and it also lets users search for pictures in the same manner. It comes with a complex API (http://www.flickr.com/services/api), which can be accessed through REST as well as SOAP, with a vast documentation and API Kits for every modern programming language.

One of the great features of Flickr is how it organizes the images that its users submit. They are asked to add tags to their photos, enabling other users to easily find images related to a particular topic. Flickr was one of the early implementers of tag clouds and is also considered a good example of effective use of folksonomy. Other than tags, a Flickr image also has some other metadata, such as title, owner, date taken, date uploaded, views. There is also the possibility to access the Exif data of an image, which includes, but is not limited to, camera, exposure, aperture, focal length, date and time it was created, orientation, color space.

Both private and public image storage are provided and there is the posibility to either release images under certain common usage licenses (http://www.flickr.com/creativecommons) or label them as “all rights reserved”. The images we have downloaded are all public and fall under a Creative Commons license.



http://stutomcat.ifs.tuwien.ac.at:8080/MuckeDownloadTasker/

2.2. Task Distribution Service (TDS)

For downloading all the data needed for MUCKE image repository, a Task Distribution Service (TDS) was created. The images were filtered according to a list of Wikipedia concepts ranked by frequency of occurrence in Flickr. For test purposes, a 200 concept list was used, called *imagelistSmall*. The complete Wikipedia concept list used in the download task was used to generate the *imageListBigUnique* file.

The final repository has **83.473,358** images, and therefore an incremental download process needed to be implemented in order to ensure the correctness and completeness of the

algorithm and to be able to monitor the overall task. By correctness we refer to the fact that all images are successfully transferred from Flickr servers, while completeness stands for ensuring that all images were downloaded as intended and that the algorithm has stopped when this condition was met. In order to implement the previously mentioned constraints, the TDS distributes to each downloading agent a **download task.** Each download task refers to a total of 10000 images to be transferred from Flickr servers onto the agent's local storage. After the task was finished, the agent ensured that the process has ended by marking the task as "done" and that it complies with the above constraints by marking it as "verified".
The TDS resides at http://stutomcat.ifs.tuwien.ac.at:8080/MuckeDownloadTasker and has two main actions: getting new tasks and indicate tasks completion.

2.2.1. Get new task

To reserve a new download task, the following HTTP request needs to be issued to the TDS. The request requires two parameters, separated by '|':

- *username* - for recording purposes only; to identify the downloading agent, the institution acronym (UAIC/TUW/CEA/BU) was used optionally followed by the machine identifier; the IP of the request was also recorded.
- *image list name* - to know from which file were the image ids taken for download

Get new task example:

```
curl -v -H "Accept: text/plain" -H "Content-type: text/plain" -X POST -d 'user1|imagelistBigUnique' http://stutomcat.ifs.tuwien.ac.at:8080/MuckeDownloadTasker/resources/Task/getNew
```

The service replies with 5 parameters, also separated by '|':

- *task id* - internal number to keep track of tasks - requested when indicating task completion
- *start position* - the line from which to start reading image identifiers to download
- *end position* - the line until which to read image identifiers for download
- *username* - a confirmation of the correct recording of the given username
- *image list name* - a confirmation of the correct image list identification

Reply to previous request example:

```
HTTP "1 | 100 | 199 | user1 | imagelistBigUnique"
```

2.2.2. Indicate task completion

When the download script finished a new archive was created containing the requested 10000 images and an associated log file. The next required step was to notify the TDS of the task status. This service request contains three parameters:

- *username* - for recording purposes only
- *task id* - the task to be marked as completed
- *image list name* - to know from which file were the image ids taken for download

Example of completions status notification:

```
curl -v -H "Accept: text/plain" -H "Content-type: text/plain" -X POST -d 'user1|1|imagelistBigUnique' http://stutomcat.ifs.tuwien.ac.at:8080/MuckeDownloadTasker/resources/Task/done
```

The response of this request simply states that the task status was updated.

2.2.3. Putting it all together

The download script resided at (File:DownloadImages.sh). The script identifies downloads based on the task id and the image list used in the download.
File:MyMuckeDownloader.sh uses the above to fetch a new task, download the data, and mark the task as completed when it is so. It takes three parameters: the image list to use, the folder where to put the images, the folder where to put the logs. In the end, the archive of all the images is created in the folder where this script is run.

2.2.4. Moving and verification

The verification step is accomplished using a java jar file (File:MuckeDownloadFolderMonitor.jar) which can be run:

```
java -jar MuckeDownloadFolderMonitor.jar <folder>
```

given that the File:Config.xml is in the same folder as the jar file.
The app is written in Java, using JVM 1.7 and is OS independent. It will attempt to use native file system monitoring functions, but where the OS rejects that, it will poll the given folder every 10 minutes.
It will monitor the indicated folder and check the zip files in that folder according to three criteria:

1. they conform to the naming convention (mucke_<listname>_<taskid>_<startno>_<endno>.zip)
2. they contain 10000 entries
3. they are at least 1MB large

If all of the above are satisfied it will move the file to the (newly created) <folder>/verified folder and will inform the server that the file has been verified.
If any of the above conditions fail it will move the file to the (newly created) <folder>/erroneous folder and will inform the server that the file has been detected as erroneous. The server will then release the task corresponding to this file (based on the naming convention) to be downloaded again.
If the server returns an error (e.g. the task to be marked as verified must be in status 'returned' then the file is moved to the (newly created) <folder>/limbo folder, as all errors returned by the server must have an unusual cause.
The server has of course been updated to hand the new verification notification.
The service accepts POST requests with message of type

```
<user> | <image list name> | <task id> | <verification status>
```

at the URL

http://stutomcat.ifs.tuwien.ac.at:8080/MuckeDownloadTasker/resources/Task/verified

The <verification status> is either 'ok' or 'error'. Anything else will return an error message from the server.

2.3. Data Download Tool Description

To download the Flickr images, we have used two versions of Perl scripts: one for Windows and one for Linux machines, the main difference being the way the download is carried out. At runtime, they receive 6 parameters corresponding to the following fields:

```
$listFile = @ARGV[0]; #list of images that contains IDs and associated URLs
$taskId = @ARGV[1]; #task id
$minPos = @ARGV[2]; #position of the first image to download
$maxPos = @ARGV[3]; #position of the last image to download
$outDir = @ARGV[4]; #name of the output directory
$logDir = @ARGV[5]; #log directory to store logs of the download
```

At first, it opens the images list file containing the IDs and associated URLs and it browses to the line corresponding to the position of the first image to download. After checking whether the file exists, it downloads the image and it prints the completion status in the task logo file. Further, it browses to the next line corresponding to a new image and it continues the download process until the line corresponding to the position of the last image is reached. The script makes sure that no more than 1 image per second is downloaded in order to comply with Flickr download policies.

```
#counter for the list of images
$counter = 0;
#open the list of images open  LF, $listFile or die "can't open $listFile\n";
while(defined($line = &lt;LF&gt;))
{
      if($counter &gt;= $minPos && $counter &lt;= $maxPos)
      {
              chomp($line);
              print "$line\n";
              @parts = split(/\t/, $line);
              $photoFile = "$outDir/@parts[0]";
              #if the file exists, test that it is not empty
              $photoSize = 0;
              if(-e $photoFile)
              {
              $filesize = stat($photoFile)-&gt;size;
              #remove the file if it is empty
              if($filesize == 0){unlink($photoFile);}
              }
              #if the file does not exist
              if(!(-e $photoFile))
              {
              #the while loop ensures that an image is searched until it can be
downloaded
              #or until at least $maxTries tries were unsuccessful
```

```
              #this can be useful
              $inWhile = 0;
              $tries = 0;



              while($inWhile == 0 && $tries &lt; $maxTries)
              {
                     $timeBefore = time();
                     #linux version
                     $toExec = "wget -q -t 0 -T 10 -O $photoFile -U IE5
\'@parts[1]\'";
                     `$toExec`;
                     #windows version
                     #$status = is_success(mirror(@parts[1], $photoFile));
                     $timeAfter = time();
                     $timeDifference = $timeAfter - $timeBefore;
                     #make sure that we do not download more than 1 image per
second - to comply with Flickr download policies
                     if($timeDifference &lt; 1){sleep 1;}
                     $statPhotoFile = stat($photoFile);
                     if($statPhotoFile) {
                          $filesize = $statPhotoFile-&gt;size;
                     if($filesize &gt; 0){$inWhile++;}
                     }
                     $tries++;
              }
              #write to the log file
              if($tries &lt; $maxTries)
              {
                     print LOG "@parts[0] ok in $tries tries\n";
              }
              else
              {print LOG "@parts[0] could not be downloaded correctly\n";
              }}
       }
       $counter++;
       }
       close LF;
```

After all the images from this task have been downloaded, a zip file containing these pictures and the logo file is being created.

```
$zip = Archive::Zip-&gt;new();
#add all the files to the zip opendir OD, $outDir or die "cant open out dir for
#reading $outDir\n";
@images = readdir(OD);
closedir OD;
foreach $img (@images)
{
    if($img =~ m/\w/){$zip-&gt;addFile("$outDir/$img");}
}
$zip-&gt;addFile($logFile);
unless ( $zip-&gt;writeToFileNamed($zipDir) == AZ_OK )
{
       die 'write error';
}
```

3. MUCKE Corpus

The MUCKE corpus contains metadata and images based on Wikipedia concepts that are often used to annotated Flickr images.
Wikipedia concepts are ranked using the number of corresponding Flickr images which is divided by the log of incoming Wikipedia links in order to penalize very common concepts.

3.1. File description

The file *conceptsRankedFlickrBig.txt* contains a list of Wikipedia concepts ranked by frequency of occurrence in Flickr. The top 200 concepts of this list are represented in the current corpus.
The file *imageListBigUnique*.txt contains a list of images that correspond to the Wikipedia concepts. Names are composed of the image Flickr ID and of the corresponding Flickr user ID, separated by underscore.

3.2. Mucke Corpus Directory Structure

A. *conceptMetadata* - directory that contains the Flickr metadata associated to the Wikipedia concepts. To make sure that there will not be too many subdirectories in conceptMetadata when all concepts will be represented, intermediary subdirectories were created using the first two digits from the #CONCEPT_ID in conceptsRankedFlickrBig.txt. In each of these intermediary subdirectories, we created a final directory dedicated to each Flickr concept. The #CONCEPT_ID in conceptsRankedFlickrBig.txt is used to names these final directories. For each concept, downloaded images are taken from the file "1" and their names can be reconstituted from the "owner" and "id" fields of the XML files.
B. *images* - directory that contains the image files downloaded from Flickr. To accommodate a large number of images, subdirectories are created using the first four digits of the image name from imageList.txt were created. Then, in each of these intermediary subdirectories, images are stored with their name from imageListBigUnique.txt.

Flickr Metadata File example:

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
<photos page="1" pages="2" perpage="500" total="896">
     <photo id="4912140037" owner="11201698@N00" secret="5789c5046e"
server="4079" farm="5" title="Hercule Poirot's Chrismas, Agatha
Christie" ispublic="1" isfriend="0" isfamily="0" datetaken="2010-01-15
14:00:20" datetakengranularity="0" tags="collins agathachristie
herculepoirot fontanabooks alexisorloff romanspoliciers
vintagedetectives vintageagathachristiebookcovers"
dateupload="1282383421" views="225" />
     <photo id="4570779334" owner="35524174@N04" secret="3bc942575a"
server="4050" farm="5" title="Hercule Poirot en attendant l' Express
Orient" ispublic="1" isfriend="0" isfamily="0" datetaken="2010-05-01
09:30:32" datetakengranularity="0" tags="man thessaloniki
orientexpress shootingpeople θεσσαλονίκη
```

```
herculepoirotenattendantlexpressorient
herculepoirotwaitingtheorientexpress expressorient
authenticorientexpresswagon" dateupload="1272796121" views="201" />
```
……

The following parameters were extracted using the Flickr public API:

- photo id – unique Flickr id
- owner – unique Flickr uploaded id
- title – Flickr image title
- datetaken – image create date
- tags – Flickr users added tags
- dateupload – Flickr upload date
- views – image number of unique views

4. Conclusion

The objective of this work package was to create a data collection framework that will offer the necessary resources for next work packages. We have shown in section 2.2, 2.3, 2.4 the activities and methodologies that were used in order to obtain a large and a well organized image collection and associated metadata. We also make the observation that the images we have downloaded are all public and fall under a Creative Commons license.