Project Phase 1: Measuring similarity based on textual and visual descriptors

Arizona state university

CSE 515 Multimedia and Web Databases (Fall 2018)

Team members:

Vishal Kumar(1215200480)

Akshay bhandari(1215188273)

Anik pait(1211225753)

Jagriti verma(1215173531)

Manoj Kumar yellojirao(1215201117)

Shrikrishna banker(1215211322)

Abstract:

Phase 1 deals with working on textual and visual descriptors of different entities to compare them using these same descriptors, these descriptors are stored in different forms of files like txt, XML and CSV. We create a system which is able to retrieve these textual and visual descriptors for different entities and work over them to find entities similar to each other. The similarity between 2 entities is based on the study of different forms of similarity measures like different Pnorm distances, cosine distance e.t.c. Challenge was to create these similarity measures in a most efficient way possible to us by designing an algorithm which has less time complexity and yet being able to work with large amount of data.

Keywords:

TF, DF, TF-IDF, CM, CM3x3, CN, CN3x3, CSD, GLRLM, GLRLM3x3, LBP, LBP3x3, HOG, K, model, user, image, location, distance.

Introduction

Terminology:

Location : Refers to the position in a structure, or index

Similarity : Refers to the value of the distance calculated.

Model : Refers to a Textual or Visual feature.

Problem specification:

This Phase includes 5 sub task, each task basically deals with finding entities similar to some particular entity based on some particular features which are common between them, these feature specifically refers to Textual or Visual descriptors. Detailed problem specification of each task is given below:

Task 1: Given a users USER ID, a category of textual descriptor (either one among TF, DF and TF-IDF), and an integer value K, find K similar users with their distances to that input user based on the given category of textual descriptor. Also return top 3 contributors (words which contribute most for this similarity).

Task 2: Given an Images IMAGE ID, a category of textual descriptor (either one among TF, DF and TF-IDF), and an integer value K, find K similar images with their distances to that input image based on the given category of textual descriptor. Also return top 3 contributors (words which contribute most for this similarity).

Task 3: Given a locations LOCATION ID, a category of textual descriptor (either one among TF, DF and TF-IDF), and an integer value K, find K similar images with their distances to that input image based on the given category of textual descriptor. Also return top 3 contributors (words which contribute most for this similarity).

Task 4: Given a locations LOCATION ID, a category of visual descriptor (either one among CM, CM3x3, CN, CN3x3, CSD, GLRLM, GLRLM3x3, HOG, LBP, LBP3x3), and an integer value K, find K similar locations with their distances to that input location based on the given category of textual descriptor. Also return top 3 contributors in the form of image-image pair (3 image to image pair which contributes most towards being similar.)

Task 1: Given a locations LOCATION ID, a list of visual descriptor ("ALL" among CM, CM3x3, CN, CN3x3, CSD, GLRLM, GLRLM3x3, HOG, LBP, LBP3x3), and an integer value K, find K similar locations with their distances to that input location based on all textual descriptors. Also return individual contribution of each visual descriptor.

Assumption:

1. We are assuming that the similarities between different entities can be expressed in the for m of distance calculated over their textual or visual descriptors, for sake of simplicity we are using distances that has been mentioned in the class.
2. Value of K is smaller than number of common users.

Description of proposed solution:

Solution to Task 1, 2 and 3 is similar to each other. Task 4 and 5 addition on that.

For Task 1, 2 and 3:

Logic is to first extract all the words used by query user/image/location then over our complete data search of all uses/images/locations who has at least 1 common word with query user. This is done by running a loop on the list of words used by query user/image/location and under each loop, we make mongo queries to find users/images/locations from the data who has used the word over which we are running the loop, doing so we store user/image/location names of all common users/images/locations in a set so that if a user/image/location comes again it won't make a duplicate entry. Then we run loop for all users/images/locations in this set under this loop we run loop for all the words used by query user/image/location inside this loop we make mongo queries to extract textual descriptor values as per the given model for both the query user/image/location and the user/image/location with which we want to compare we then the Euclidean distances between all the terms used by both uses, if there is a word that a user/image/location has not used then its textual descriptor value is taken as 0, which is justified as their contribution in Euclidean distance will be large as they are not common. For each user/image/location we store this distance in a list and later extract minimum k distances with corresponding user/image/location ID of the compared user/image/location.

To find top 3 contributing terms We again run a loop but only for closest k user/image/location under this loop there is loop for every word used by query user/image/location. We calculate difference of textual descriptor between 2 users/images/locations and store them in a dictionary with user/image/location in comparison with it's key. After this loop of k closest users/images/locations is complete we extract 3 min value of difference among all k user/image/location.

This is the complete logic behind task 1,2 and 3.

For Task 4 and 5:

Logic for Task 4:

We first extract the visual descriptor value as per the given location and model over which we want to work, for this we first make mongo query to call whole data and then extract document related to query location from appropriate collection. In this document under given model we run loop to extract visual descriptors of each image related to that location. We then access all other distinct documents (different locations data) by running a loop under the complete that set which contains 30 documents for 30 different locations. For each location we store these visual descriptors in a dictionary, this dictionary has location names as their key and value of visual descriptors (along with image names) as its value (it is a 2D dictionary). We also extract and store these location names in a list. Then we run loop for each location under this loop we run loop for each image of query location and then we make mongo queries to extract the visual descriptor values of these images and perform cosine operation on a pair of image (image being compared). We keep aggregating these cosine values until the inner loop is complete then we take average of that aggregated sum by dividing it by product number of images in location & number of images in location 2. Once we have compared one location with another total number of image to image comparison are 90,000 (saying that each location has 300 images.) we store this distance corresponding to location name in a dictionary and extract min K with their location names.

The we again run a loop for each location in k closest location (to get top 3 contributing image to image pairs.) under this loop we run loop for each image in query location and calculate & store cosine of each image to image comparison along their image to image names then store all these values in dictionary with location in comparison (from k closest location) as the key. Then we run loop 3 times over this dictionary to extract top 3 image to image pairs contributing to its similarity with query location and store this data in another dictionary(a smaller dictionary), thus we got top 3 contributing image to image pairs.

Logic for Task 5:

Logic for task 5 is exactly the same for task for it is just that while calculating distance between two different locations we run a outer loop for different model i.e. we are comparing each location with another for all different models, there will be a little changes in the inner code to manage the storage of cosine values calculated (so that they are not lost), once a location to location comparison is done there will be a total of 300x300x10 image to image comparisions i.e. 900,000 individual cosine values. Further logic to normalize and store in a dictionary is similar. Then we extract minimum 3 values with location names. This is how we get k closest location.

During this above task we also maintain a dictionary with model name as key and value as another dictionary which has individual distance of different locations with query location for that given model. We use this dictionary to represent individual model contribution for a location to location comparison. We just iterate through this dictionary and find locations(belonging to k closest locations) under each model and extract its contribution from there.

System requirements:

Note that I created my programs using spyder IDE but, it is completely safe to run the files from command prompt, you do not need to have any IDE to run my files. But MongoDB is required.

Please read Read_me.txt to understand the conversion of text, CSV and XML files to JSON files for MongoDB storage. It also includes the instructions to use our JSON files and gives description of my database and collection names.

Pre-requisite for my solution:

1. Python 3.7 (our base programming language):

Download available at - https://www.python.org/downloads/

2. MongoDB: This is my database management system.

Download available at - https://www.mongodb.com

3. Numpy and Scipy libraries.

Installation using pip command

Executing files:

please make sure you have python 3.7 installed with numpy and scipy as additional libraries.

installing numpy and scipy:

1. pip install numpy

2. pip install scipy

installing pip using CMD:

python get-pip.py

Execution of solution files:

1. Open command prompt.

2. go to location where you have these solution file usind CD command (CD <File Path>).

3. To run program for Task 1,2 or 3, execute the following command

```
python Task123.py
```

4. To run program for Task 4 & 5, execute the following command

```
python Task45.py
```

The program will ask for your desired inputs and will give corresponding appropriate output.

Sample Input & output:

Note that I have included separate Input/output files for each task (html files, as they were easy to export from the IDE). I just included them because they look better then screenshots of command prompt, but command prompt gives exactly same output (visual terms). So you may use any. Following are taken html files. I'm including only 2 sample outputs in the report(I have included separate output files for each location):

Sample output for Task 2:

```
In [15]: runfile('C:/Users/rockers_vn/Desktop/MWDB project/project_phase1_Task123.py',
wdir='C:/Users/rockers_vn/Desktop/MWDB project')
==========================INPUT START============================

Specify the Task numer: 2

Enter the image Id: 2951846628

Enter the model you want to use: TF_IDF

Enter the value of K: 4
==========================INPUT END============================
==========================OUTPUT START============================

K closest Images are:

Name: 2951861710 Distance: 0.04381489176351859

Top 3 contributing terms are:

acropoli
athen
museum

----
Name: 2951841806 Distance: 0.04714045207910317

Top 3 contributing terms are:

acropoli
athen
greece
```

```
----
Name: 2950994233 Distance: 0.04714045207910317

Top 3 contributing terms are:

acropoli
athen
greece

----
Name: 12045021564 Distance: 0.04739404489645185

Top 3 contributing terms are:

greece
parthenon
museum

----
==========================OUTPUT END============================
```

Sample output for Task 4:

```
In [10]: runfile('C:/Users/rockers_vn/Desktop/MWDB project/project_phase1_Task45.py',
wdir='C:/Users/rockers_vn/Desktop/MWDB project')
==========================INPUT START============================

Enter the task number: 4

Enter the Location number: 7

Enter the model you want to use: CN

Enter the value of K: 3
==========================INPUT END============================
Running on model: CN


===== QUERY LOCATION: montezuma castle=====

==============TASK 4 - OUTPUT START=====================

LOCATION NAME: aztec ruins
SCORE: 0.041803259031425684

Top 3 Image-Image pairs are:

4438568385 2976189
7381767396 2976189
7381767396 6494673379
```

```
---
LOCATION NAME: colosseum
SCORE: 0.042568906970878326

Top 3 Image-Image pairs are:

4438568385 2976189
7381767396 2976189
7381767396 6494673379


---
LOCATION NAME: agra fort
SCORE: 0.04286717927794232

Top 3 Image-Image pairs are:

4438568385 2976189
7381767396 2976189
7381767396 6494673379


---

Time Taken for Task 4: 66.9614188671112 seconds

==============TASK 4 - OUTPUT END=====================
```

Conclusion:

All team members were successfully able to complete all task in given period, we all applied same logic just implementation methods were different which gave slightly different results in some cases, sometimes 1,2 entities from k were different, sometimes distance values were slightly different. For task 5, I used cosine distance while some team members switched back to Euclidean.

Time taken in executions:

1. For Task 1,2 and 3: 3-9 seconds.
2. For Task 4 : 60 – 70 seconds (see the bottom of the output for Task 4)
3. For Task 5 : around 11-15 mins with cosine calculation as distance, otherwise around 3 mins. with Euclidean distance.
   I checked for both cases but finalized with cosine distance (using same code for Task 4 & 5.)

Overall, we successfully completed all the five given tasks in given period of time and got the desired results.

Appendix :

During Implementation phase, we all worked together in providing logic and helped each other with coding syntax and errors.

Initially we divided work in the following manner:

- Akshay and Krishna wrote scripts to convert all base files to JSON files for MongoDB usage.
- Manoj worked with MongoDB queries.
- I with Anik and Jagriti worked looked for connection, search and retrieval between MongoDB and python.

We all were new to MongoDB that is why we had to divide this task of understanding and using MongoDB.

Each Team member was a great help.