CSE 515: Multimedia and Web Databases

Phase 3

"Graph Analysis, Clustering, Indexing and Classification of multimedia database objects"

[1]Anik Pait (1211225753), [2]Vishal Kumar (1215200480), [3]Akshay Bhandari (1215188273), [4]ShriKrishna Bankar (1215211322), [5]Manoj Yellojirao (1215201117), [6]Jagriti Verma (1215173531)

Abstract: As the data is humungous in today's world and keeps on increasing exponentially day by day, our project team has dealt with large-scale data distribution to index the images by applying variety of classification and clustering algorithms such as K means, PageRank algorithm and Personalized PageRank algorithm. The project deals with the efficient retrieval of data and indexing the data to carry out similarity among different types of the visual models of their images at the social networking sites like Flickr. The data is stored in NoSQL database (huge data set) and the project showcases how the required data is retrieved for the necessary operations and how the data has been managed and implemented for certain similarity calculations.

Keywords: data, retrieval, similarity, PageRank, PPR, K means, NoSQL, Clustering, kNN, Teleportation, Graph, Indexing

1. Introduction

The world produces millions of gigabytes of data per day in the form of text, images, videos etc. Thus, efficient storage and retrieval of this data is very crucial part. Text data being easy to manipulate can be easily handled but the real challenges occurs when we deal with multimedia objects as it cannot be compared directly like text data.

Multimedia objects usually have multiple features, which are often represented in the form of multi-dimensional vector space with each feature becomes one dimension. As dimensions goes on increasing, searching becomes tedious task and it complicates the overall application. Indexing holds a key aspect to rank the multimedia objects based on the query similarity. Either classification or clustering algorithms are performed to achieve the similarity with the query. This process of displaying the similar images with respect to the query image is called indexing.

1.1 K Means Clustering

K means clustering is a NP hard problem on cluster analysis on the data. It partitions 'n' number of samples into k partitions/ clusters in which each observation stays with the mean of the cluster with the nearest distance to it. In other words, K-means is an iterative method of clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

1.2 PageRank Algorithm

PageRank algorithm indexes or ranks the web pages in the web based on the content and the links related to the spent by a user on a node determines the PageRank of the node. The algorithm works iteratively on the following formula:

$$\text{PR(Node)} = (1\text{-d}) + \text{d}^* \sum_{k=0}^{n} \frac{\text{PR(incoming node k)}}{\text{No of outgoing edges of k}}$$

Where PR refers to PageRank of the node and d refers to the damping factor which has been taken as d = 0.85 (as mentioned in the paper) for all the tasks.

1.3 Personalized Page Rank (PPR)

Personalized pageRank (topic sensitive page rank) is modified version of pageRank where we can bias the probability of random jumps according to topic or query. If we look at problem formulation in context of matrix, the teleportation vector or personalization vector can be assigned with more probability for the query topics [6]. Personalized page rank can be calculated as

rank = ((1-d) * M * rank) + (d * p)

where, d is damping factor usually set as 0.85 as per many research papers, M is column stochastic normalized probability matrix, p is personalized teleportation matrix with biased weight for topics

1.4 k-nearest neighbors' classification

It is also knows as kNN classification, where objects are classified based on labels of k-nearest neighbors of the object. Nearest neighbors can be identified using any distance/similarity measure. We are using cosine similarity in this phase of the project.

1.5 Locality Sensitive Hashing

Locality Sensitive Hashing is used to reduce the dimensionality curse by creating buckets where the similar objects with higher probability are mapped to the same 'buckets' using the hash function. It aims to maximise the collision between two similar items. It is useful in finding the nearest neighbors with a good approximation.

2. Assumptions

- Pre-requisite knowledge of MongoDB and Python programming
- The distance measure selected for each of the tasks for similarity is precise mathematically.
- Data provided is consistent in its form (example column1 of file represents the image -id) so that the files can be directly parsed into database by building a common parsing logic.
- Data provided is pre-computed and we are supposed to use as is.
- In task 2, Number of clusters in which we want to divide the given graph is greater than or equal to two. (Being 1 simply means we are not performing any clustering mechanism).

3. Problem Description

This phase of project has 6 tasks which deals with the indexing structures and clustering algorithms of the image-image graph.

The detailed specification of every task is on the following:

Task 1

In task 1, we need to implement an image - image similarity matrix using an appropriate similarity measure. Using the image - image similarity graph, we need to create a graph such that each image has K outgoing edges to K similar images. The output of this task would act as an input for the further tasks.

Task 2

Given a directed graph of images, we need to divide the graph in small groups/clusters for better visualization or categorization as per our need (or need for some future application). We are required to cluster the images so that the images that are present in the same cluster are related in a visual manner (not just randomly assigned to a cluster). The clustering algorithm is based on choice.

Task 3

Given a graph of images, (from task 1), we need to apply PageRank algorithm and visualize the K most dominant images of the graph where K is supplied by the user. This gives the idea on which nodes/images are dominating in the dataset. We need to use the output of task 1 for this task. The PageRank algorithm would be computed by using the mathematical formula as proposed in the paper by Page and Brin.

Task 4

In this task given an image -image graph, user supplied 3 image ids and K value. We need to apply the personalized page rank algorithm to

report the K most relevant images in terms of the 3 user supplied image ids. where image -image graph is of dimension n*k matrix where k<=n, k denotes the most similar or nearest neighbor image for each image.

Task 5

Implement LSH, given no. of Layers (L) and no. of bits (k) values, and chosen any distance function, create an in-memory index structure for the vector space which is creating different buckets of similar images if they have same hash codes. Hash code is generated by doing dot product of a data point and random generated planes whose number is same as the number of bits given as an input. Random generated planes divide the whole data set into smaller spaces. After creating an in-memory index structure, given no of top similar images (t) and image id, find the top similar t images. Also, find no of unique and overall images referred for finding those top t similar images.

Task 6

In this task, we need to implement k-nearest neighbor (kNN) and personalized page rank (PPR) based classification algorithms for labeling rest of the images in entire dataset using a given set of input image – label pair list (in the form of file). We need to use image – image similarity matrix for kNN and output graph of task 1 for PPR based classification algorithm.

4. Problem Solutions

4.1 Implementation Details

Task 1

***Aim*:** The task 1 focuses on generating an image - image similarity graph and getting the K most outgoing edges and images of the graph.

Approach.

- First construct the image-image similarity matrix using the VD. We are using the cosine similarity to find the distance between the object to object.
- We are taking the k nearest neighbors for each of the images. Then we are constructing the n*n matrix where each image has weight for k images and remaining images are filled with zero weight.
- The matrix was used to get out the most K outgoing edges by sorting the columns of the image image similarity matrix in a descending order and taking out the top K columns using the lambda function, the K most outgoing edges to the images were displayed.
- The resulting image image graph was stored locally to use it for the future tasks. (stored in graph.json)

***Why Visual Descriptors*:** The Visual descriptors consists of 10 visual models and the 300 images are distributed over the 10 models which keeps the number of dimensions constant compared to way higher number of dimensions considering the terms. The terms have dynamic nature as they may change time to time and they increase the probability of dimensionality curse even if majority of the images might not have that term causing in a sparse matrix. The term act as labels which doesn't give a mathematical relevance to our calculations compared to visual descriptors which form as composition to the image.

Why Cosine Similarity: Since our target was to capture the composition of the image formed by the visual model scores so that they stay semantically similar to each other as presented visually, Cosine similarity works best for the purpose of capturing the similarity matrix between the images. Hence, cosine similarity was used to compute the image-image similarity matrix.

Conclusion: Hence, we generated an image image graph of N images x K outgoing edges to K images and the graph is stored for the further tasks.

Task 2

Aim: The task 2 focuses on using the output graph of task 1 and form clusters using the two distinctive clustering algorithms. The clustering algorithms used by us are *K Means Clustering* and *Spectral Clustering*.

Approach:

- For ***K Means Clustering***:

- *Representing graph space in vector space:* There are various ways to represent graphs in terms of a matrix. Our approach is based on the method to construct weighted adjacency matrix described in [5]. We can define weighted adjacency matrix A_{ij} for a directed graph G with n vertices as $n \times n$ matrix. Let set of nodes be $V = \{1,2,\ldots,n\}$ and set of directed edges E. Additionally, let weight for the edge $(i,j) \in E$ be ω_{ij}.
  $A_{ij} \coloneqq \omega_{ij}$, if $(i,j) \in E$
  $A_{ij} \coloneqq 0$, otherwise
- *Initial cluster formation:* General form of k-means algorithm starts by forming the initial clusters by picking up random points from the dataset and treating them as centroids. We then assign all the points in dataset to a cluster based on the distance of the point from each centroid. Though it is one of the simplest method for initialization it results in additional time till the means converge. To tackle this problem, we have decided to use max-a-min algorithm to initialize the cluster. Max-a-min, finds k farthest points in the dataset, where k is the number of clusters to be formed. This ensures an approximate equal distribution of points in the all the clusters.
- *Cluster refinement:* Once all k clusters are initialized k-means tries to refine clusters iteratively by updating the centroid of each cluster as means of each cluster and then redistributing all the points in datasets according to their distances with newly found centroids. This process is repeated until centroids become stable.

- For ***Spectral Clustering***:

  - The mechanism of spectral clustering starts with the adjacency matrix which is later used to create the Laplacian matrix, where we directly calculated the Laplacian matrix using our given graph (this graph gives us the information about connections between different images which is same as the information provided by adjacency matrix). The rules to calculate Laplacian matrix are as follow –
    1. If neighbor - put '-1'.
    2. If not connected - put '0'.
    3. For self - put 'k'

  - It is nature of Laplacian matrix that the rows sum would be '0'. We calculate Eigen-Values and Eigen-Vectors for this Laplacian matrix and find the second smallest Eigen-value, this Eigen-value represents the Eigen-vector with which the division of graph is easiest, i.e. represent weakest connections, representing cheapest cut. Note to calculate the Eigen-Values and Eigen-Vectors, we have used the 'Numpy linear algebra eigen' library.
  - We take this Eigen vector and sort it in descending order (Image names are maintained in the index).
    The portioning mechanism –
    Even though we concluded that applying 'K-means' gives the most accurate results(as it considers the impact of 3rd, 4th largest eigen-values and so on.), We decided not to implement it as we already used 'K-means' as our other clustering method.

We formed our own set of rules of cluster division. The rules are as following –

- The initial division (in two clusters) is done on the basis that whether the Eigen-vector element for a certain image is positive (or equal to '0') or negative. We put Images with values greater than or equal to zero in one group and images with negative values in another group.
- We compare the length of both groups and keep only the one which is smaller, the larger one will be used for further division. This larger group is maintained and is not defined as any cluster, call it 'non-cluster' group.
- If only, two clusters were there i.e. only one needs to be formed now – we simply put larger one in another group, as nothing else can be done.

If, more than one clusters are still there to be formed then –

- We compare the length of all groups that have been already formed along with the non-cluster group. We do this to get the group having maximum number of datapoints(images).
- If the group is 'non-cluster' group, then we form further cluster using this group only.

Further clustering method –We look for the element(image) having 'Eigen-vector' element closest to the mean of all 'Eigen-vector' elements related to images in this this group. Once we find such elements, we can create 2 clusters, one containing images till the mean-element image, 2nd cluster containing other

images. We keep the one with smaller number of images and mark the other one as 'non-cluster' group and the iteration continues till the required number of clusters have been formed.

Conclusion: The proposed solution created groups of equal weights and images of the cluster had visual similarity.

Task 3

Aim: To get the K most dominant images from the image image graph of task 1 using the PageRank Algorithm.

Approach:

- The image image graph of N x K was retrieved from task 1 output for this task. The teleportation vector was initialized with 1/N (N: Number of images) as assumed to be the initial pagerank of the nodes and stored in page dictionary.
- For each image/node, the incoming nodes were identified using the graph (the 'keys' in the dictionary graph.json) were identified as the incoming nodes if the particular node whose pagerank is yet to be identified appears in the 'value' of the same dictionary (graph.json) and the pagerank was calculated iteratively using the mathematical formula (no matrix method was used).
- The pagerank values get updated as the iteration moves and the updated pagerank values are used by the images. The outgoing edges for each image was taken as K.
- Once, the convergence occurs with an error rate of 0.00001, the final pagerank value of the particular node gets stored in the 'p_rank' dictionary. As the length of 'p_rank' dictionary becomes equal to the original length of the page dictionary, indicates convergence has occurred and the loop ends.
- The Pagerank scores are displayed and the top K most images are taken using lambda function. The images are visualized in the visualize function

Damping Factor: It explains the concept of imaginary surfer will eventually stops clicking any link. The d will signify the probability with which the surfer will continue to click on the link. Generally, the damping factor is 0.85. this value can be changed, this will significantly vary the PageRank of a node. The damping factor has been taken as 0.85 (as mentioned on the original paper by Page and Brin) as damping factor reduces the effect of giving too much weight to the links only. Different applications use different damping factor values according to the research or experiment on the specific domain.

Conclusion: The Pagerank computation was done applying the algorithm and the most dominant K images were visualized. The images had certain number of incoming links which varied with each image.

Task 4

Aim: To find K related images using the personalized Pagerank algorithm for specified three specified image Id's.

Approach*:*

- The primary difference between the PR and PPR is that random walks are completely based on the given input images.
- We are constructing the probability matrix using the similarity matrix by normalizing the similarity matrix across the columns so that probability of traversing of each node to each remaining K nodes sum up to 1.
- Teleportation vector used for the finding the PR of a node is a 1-dimensional array of n elements with probability for given 3 images are equally divided to 1/3 and remaining images are having 0 value.
- For finding the PageRank of a node we are just looping through the all nodes and considering the number of incoming edges for a node and summing up the all the probability of all the of incoming edges and multiplying with the damping factor of 0.85 and adding it to the (1-d) * the teleportation vector until convergence is reached.
- Convergence is a factor which signifies the tolerance of error. It is calculated using the minimum of the difference between the current page rank of a node and the previous page rank of a node. The values are generally set according to the application semantics.
- We are using the value as 0.000001 as the page rank value calculated for a node is very small. Once the convergence has achieved, we are just sorting the page rank of all the node in

the descending order of their values. And selecting the top K user specified values.

- Random walks in PPR is purely based on the input images, it will check the how well the other images are connected to the given input image ids. The visualization happened in the same as described in task 3.

Conclusion: The K most relevant images related to the given image ID's were identified and visualized.

Task 5

Aim: *To implement* LSH, given no. of Layers (L) and no. of bits (k) values, and chosen any distance function, create an in-memory index structure for the vector space which is creating different buckets of similar images if they have same hash codes.

Approach*:*

- Consider the whole visual data with all the models and form the matrix. After fetching above matrix as input, generate the random planes in order to divide the whole vector space in smaller spaces as data points present in the smaller spaces will likely to be similar.
- Generate hash code for each data point with respect to the random planes. For each iteration, the generated random planes will be different. Hash codes will be generated based on the orientation of the data point with respect to the layers. If data point is on the right side of the plane then it will give 1 or else 0.
- Make clusters for each hash code and if the hash code of the two data points matches then add them into the same hash code bucket. Likewise, buckets of hash codes will be generated.
- For a query image if, find the respective vector and generate the hash code. Now check this hash code belongs to which bucket by comparing the existing hash codes of the respective buckets. If matches, then retrieve all the data points of that bucket because they are supposed to be similar as they have the same hash code which implies belongs to the same space after division.
- After retrieving all the data points of the bucket where query image belong and this will repeat for each layer and the bucket values will be different as the planes are randomly generated. Take the all the images for each layer where the query image belongs and then find the top t similar images using Euclidean Distance. Also, find the number of the unique and overall images that were referred for finding top t similar images.

Conclusion: The t most relevant images were identified and visualized using the LSH method.

Task 6

Aim: Classify all the images in dataset using user given training data (input – label pairs provided in csv file). Algorithms to be used are k-nearest neighbor (kNN)based algorithm and personalized page rank (PPR) algorithm. For kNN, the value of k will be accepted from user at runtime.

Approach:

- For kNN:

We are using entire image – image similarity matrix. Similarity measure used is cosine similarity.

  - Accept the value of k and input csv file containing image – label pairs.
  - For every image in dataset (except input images which are already labelled in input csv file), find k nearest images using an image – image similarity matrix.
  - Out of k nearest images, find the label having maximum count and assign that label to current image.

How to break the ties: There can be multiple labels with same image count. One common strategy to resolve ties is to consider odd value of k. Also, as same image can have multiple labels in input file, considering odd value for k will not always serve the purpose. Here, we are giving higher weightage to the label of image with highest similarity score.

- For PPR:

  - Initialize damping factor = d = 0.85, threshold error maxerr = 1.0e-8, initial pageRank as 1/N where N is total number of images in dataset.

- Graph graph is NxN column stochastic normalized probability matrix which is modified version output of task1, with jump probability based on similarity score where there is outgoing link and zero in all other cases.
- For each label from training(input) dataset, extract all image ids having that label. Let image_count be the total number of images having current label.
- While matrix does not get converged (new_rank – old_rank < mazerr),
  - Initialize teleportation vector or personalized vector of dim 1xN with (1/image_count) for each image having current label and zero for all other values.
  - Calculate page rank using formula:

    rank_new = (1-d * graph * rank_old) + (d * personalized_vector)

- Each image will have PPR score with respect to every label in training dataset. Assign label to image having highest PPR score.

Conclusion: We successfully applied the kNN and Personalized PageRank algorithm to label the images in dataset.

5. Interface Specifications

We are using Command Line interface to run all our python files. The pythons can be run by the following syntax:

C:/path/ pythonFileName.py

6. System Requirements and Installations

Install below software/tools as a pre-requisite of this Phase. Please follow readme file for installation and execution of program.

Python

- Version 3.7.0
- Installation steps: https://realpython.com/installing-python/
- Download Link: https://www.python.org/downloads/windows/

MongoDB

- MongoDB community edition is also available for free to download and use
- Version: v4.0.1
- Installation Steps: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/
- Download Link: https://www.mongodb.com/download-center/community

GitBash

- Bash shell is used only to convert raw data from devset to uploadable format into mongoDB.
- Version 2.18.0
- https://git-scm.com/download/

Notepad++ IDE for python (user choice)

- Version: 7.5.8
- Installation steps and download link: https://notepad-plus-plus.org/download/v7.5.8.html

Python Libraries

Sr. No	Library	Version
1	pymongo	3.7.2
2	numpy	1.15.3
3	pandas	0.23.4

- *Execution Steps:*
  Refer: Readme.MD

- Operating System:
  Windows 10, Mac, Ubuntu, Linux

7. Conclusion

We have successfully implemented all the 6 tasks as per problem specification of this phase with good understanding of index structures and image similarity concepts. All the tasks were perfectly designed which was practical implementation of concepts was taught in class. This helped us to get deep insight of the topics

such as PageRank Algorithm, Personalized PageRank algorithm, K means clustering, Spectral Clustering and Locality Sensitive Hashing. Dealing with such a huge data was challenging given the system constraints, thus code optimization and properly handling memory allocation was crucial part. We learnt that how the image graph is formed for given set of images and features to index and cluster them using the appropriate algorithms. We saw the application side by the practical implementation of the theoretical concepts in class on how to deal with indexing and clustering provided the given challenges. We faced the issues in terms of memory or constructing such a huge image graph considering the visual descriptors. We found ways of handling the same by Pandas for constructing the graph as json object. Also, convergence was an issue for iterative algorithms and error rate had to be balanced to get the convergence.

Bibliography

1. https://en.wikipedia.org/
2. S. Brin and L. Page, "*The anatomy of a large-scale hypertextual Web search engine"*. Computer Networks and ISDN Systems 30: 107117, 1998"
3. J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. *Automatic multimedia cross-modal correlation discovery*. In KDD, pages 653-658, 2004
4. Alexandr Andoni and Piotr Indyk, "*Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions*" Communications of the ACM, vol. 51, no. 1, 2008, pp. 117-122
5. Masaki Aida, Chisa Takano, and Masayuki Murata. "*Oscillation Model for Network Dynamics Caused by Asymmetric Node Interaction Based on the Symmetric Scaled Laplacian Matrix."* FCS 2016, pp. 38–44, July 2016.
6. Taher H.Haveliwala Stanford University Computer Science Department, Topic-*Sensitive PageRank*
7. Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, Pinar Duygulu, Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213, U.S.A, *In Proceedings of the 10th ACM SIGKDD Conference* 2004, Seattle, WA, August 22-25, 2004

8. Appendix

We have worked as a group on all the tasks sharing our conceptual and implementation ideas throughout this phase and helping each other in understanding every task. On higher level, below is the snapshot of tasks we divided amongst ourselves, but not tightly bound as almost everyone was involved in every task.

Sr. No	Task	Team Member
1.	Creating MongoDB from data files	Akshay, Shrikrishna
2.	DB querying and Data retrieval	Manoj, Anik, Vishal
3.	Visualisation	Shrikrishna
4.	Task1	Vishal,Akshay
5.	Task2	Akshay, Vishal
6.	Task3	Anik
7.	Task4	Manoj
8.	Task5	Jagriti
9.	Task6	Shrikrishna
10.	Testing	All
11.	Readme	Jagriti, Anik
12.	Project Report	All

Everyone's contribution was remarkable and was a great help clearing each other's roadblocks!