CSE 515: Multimedia and Web Databases
Project Report: Phase 2
Group 18, Fall 2018

Group Members

1. Aditya Kanchivakam Ananth
2. Aniket Rajendra Dhole
3. Shatrujit Singh
4. Siddharth Pandey
5. Siddhesh Sudesh Narvekar
6. Vasisht Sankaranarayanan

Abstract

Volume, variety and velocity are the characteristics of data being generated in today's world. These characteristics make it increasingly challenging to make sense of the data and find similarity between objects in datasets. Using dimensionality reduction, we can up have algorithm that is polynomial instead of exponential. In this project, we try to determine similarity between users, images and locations using a dataset which consists of textual and visual descriptors of images which have been uploaded by users to Flickr. The similarity between items is determined by doing a latent semantic analysis using the techniques such as Principal Component Analysis, Singular Valued Decomposition and Latent Dirichlet allocation.

Keywords

- TF – Term Frequency
- DF – Document Frequency
- IDF – Inverse Document Frequency
- TF-IDF – Term Frequency-Inverse Document Frequency
- CM – Global Color Moments on HSV Color Space
- CN – Global Color Naming Histogram
- HOG – Global Histogram of Oriented Gradients
- LBP – Global Locally Binary Patterns on Gray Scale
- CSD – Global Color Structure Descriptor
- GLRLM – Global Statistics on Gray Level Run Length Matrix
- Code 3x3 – Spatial Pyramid Representation (CN3x3, CM3x3, LBP3x3, GLRLM3x3)

- PCA – Principal Component Analysis
- SVD – Singular Value Decomposition
- LDA – Latent Dirichlet Allocation
- Tensor
- CP Decomposition – CANDECOMP and PARAFAC Decomposition

Introduction

- **Terminology**
    - Term Frequency (TF): Term Frequency is a measure of the weight of a given keyword *k* in a given document *d*. It is mathematically defined as

$$tf(k,d) = \frac{count(k,d)}{size(d)}$$

    - Document Frequency (DF): Document Frequency is a measure of the total number of documents in the database *D* containing the given keyword *k*. It is mathematically defined as

$$df(k,D) = \frac{number\ of\ documents\ containing(k,D)}{number\ of\ documents(D)}$$

    - Inverse Document Frequency (IDF): Inverse Document Frequency is an inverse measure of DF. It is mathematically defined as

$$idf(k,D) = \log\left(\frac{number\ of\ documents(D)}{number\ of\ documents\ containing(k,D)}\right)$$

    - Term Frequency-Inverse Document Frequency (TF-IDF): TF-IDF weight of the keyword *k* for document *d* in database *D* combines the aspect of TF and IDF:

$$tf_{idf(k,d,D)} = tf(k,d) * idf(k,D)$$

    - Principal Component Analysis (PCA): Principal component analysis, also known as the Karhunen-Loeve, KL, transform is a linear transform, which optimally decorrelates the input data. In other words, given a data set described in a vector space, PCA identifies a set of alternative bases for the space along which the spread is maximized. The goal of the PCA transform is to identify a set of alternative dimensions for the given data space, such that the covariance matrix of the data along this new set of dimensions is diagonal. This is done through the process of eigen decomposition, where the square matrix, S, is split into its eigenvalues and eigenvectors.

- Singular Value Decomposition (SVD): Singular value decomposition (SVD) is a technique for identifying a transformation that can take data described in terms of n feature dimensions and map them into a vector space defined by *k ≤ n* orthogonal basis vectors. *If A is an m x n matrix with real values. Then, A, can be decomposed into*

$$A = U\Sigma V^T$$

  where

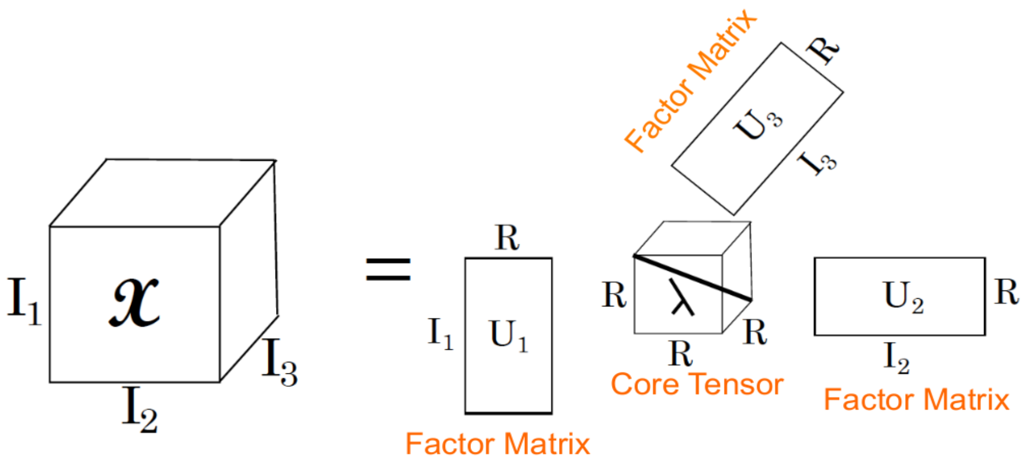
  1. *U is a real, column-orthonormal m x r matrix, such that* $UU^T = I$
  2. *∑ is an r x r positive valued diagonal matrix, where r <= min(m, n) is the rank of the matrix A*
  3. V^T *is the transpose of a real, column-orthonormal r x n matrix, such that* $VV^T = I$

  The columns of U, also called the left singular vectors of matrix A, are the eigenvectors of the m x m square matrix, AA^T*. The columns of V, or the right singular vectors of A, are the eigenvectors of the n x n square matrix,* A^TA*. ∑[i, i] > 0, for 1 ≤ i ≤ r, also known as the singular values of A, are the square roots of the eigenvalues of* AA^T *and* A^TA*.*

- Latent Dirichlet Allocation (LDA): Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA assumes the following generative process for each document w in a corpus D:
  1. Choose N ∼ Poisson(ξ).
  2. Choose θ ∼ Dir(α).
  3. For each of the N words :
     1. Choose a topic z_n∼ Multinomial(θ).
     2. Choose a word w_n from p($w_n|z_n$,β), a multinomial probability conditioned on the topic z_n.

- Tensor: A tensor is a generalization of matrices whereas a matrix is essentially a two-dimensional array, a tensor is an array of arbitrary dimension. Thus, a vector can be thought of as a tensor of first order, an object-feature matrix is a tensor of second order, and a multisensor data stream (i.e., sensors, features of sensed data, and time) can be represented as a tensor of third order. The dimensions of the tensor array are referred to as its modes. For example, an M× N × K tensor of third order has three modes: M columns (mode 1), N rows (mode 2), and K tubes (mode 3). These 1D arrays are collectively referred to as the fibers of the given tensor. Similarly, the M× N × K tensor can also be considered in terms of its M lateral slices, N horizontal slices, and K frontal slices: each slice is a 2D array (or equivalently a matrix, or a tensor of second order).

- CANDECOMP and PARAFAC decompositions (CP Decomposition):



I_1 ~ Users (length of Users)

I_2 ~ Images (length of Images)

I_3 ~ Locations (length of Locations)

R ~ K (Rank)

U_1 ~ Users x K

U_2 ~ Images x K

U_3 ~ Locations x K

λ ~ K x K x K

- **Goal Description**
  - **Task 1:** Implement a program which lets the user choose among
    1. A user-term vector space
    2. A image-term vector space, or
    3. A location-term vector space,

    and then, given, a positive integer value, k, identifies and reports the top-k latent semantics/topics in corresponding term space using

    1. PCA,
    2. SVD, or
    3. LDA

    Each latent semantic should be presented in the form of term-weight pairs, ordered in decreasing order of weights.

  - **Task 2**: Extend the above program such that, after the top-k latent semantics are identified, given
    1. A user ID,
    2. An image ID, or

    3. A location ID,

    The system also identifies the most related 5

        1. User IDs,
        2. Image IDs and
        3. Location IDs

        using these k latent semantics (list also the matching scores).

- **Task 3**: Implement a program which, given a visual descriptor model (CM, CM3x3, CN, CN3x3, CSD, GLRLM, GLRLM3x3, HOG, LBP, LBP3x3) and value "k",
    1. First identifies (and lists) k latent semantics using
        1. PCA,
        2. SVD, or
        3. LDA,

        On the corresponding image feature space, then

    2. Given an image ID, identifies the most related 5
        1. Image IDs and
        2. Location IDs

        Using these k latent semantics (list also the matching scores).

        As before, the choice of the dimensionality reduction algorithm to be used is left to the user.

- **Task 4**: Implement a program which, given a location ID, a visual descriptor model (CM, CM3x3, CN, CN3x3,CSD,GLRLM,GLRLM3x3,HOG,LBP, LBP3x3), and value k,
    1. First identifies (and lists) k latent semantics corresponding to the given location using
        1. PCA,
        2. SVD, or
        3. LDA

        (as before, the choice of the dimensionality reduction algorithm to be used is left to the user), and then

    2. Identifies and lists the most related 5 locations using these k latent semantics (list also the matching scores).

- **Task 5:** Implement a program which, given a location ID and value k,
    1. First identifies (and lists) k latent semantics corresponding to the given location using

        1. PCA,
        2. SVD, or
        3. LDA

    Applied across all visual descriptor models (as before, the choice of the dimensionality reduction algorithm to be used is left to the user), and then

    2. Identifies and lists the most related 5 locations using these k latent semantics (list also the matching scores).

- **Task 6:** Implement a program which, given a value k,
    1. Creates a location-location similarity matrix,
    2. Performs SVD on this location-location similarity matrix, and
    3. Reports the top-k latent semantics.

       Each latent semantic should be presented in the form of location (name)-weight pairs, ordered in decreasing order of weights.

- **Task 7**: Implement a program which, given a value k,
    1. Creates an user-image-location tensor (based on number of terms shared by all three),
    2. performs rank-k CP decomposition of this tensor, and
    3. created k non-overlapping groups of users, images, and location based on the discovered latent semantics.

- **Assumptions**
    - All terms are keywords.
    - The term frequencies have been defined differently in the dataset.
        - Term frequency: Count of the term in a document.
        - Document frequency: Count of the documents in the database containing the term.
        - TF-IDF: This has been set to the value given by a division of term frequency by document frequency.

<u>Description of the Proposed Solution/Implementation</u>

Task 1:

The user decides one of the vector spaces: user-term, image-term or location-term, based on which the appropriate input file is read. Each user and image in the respective input file has a user id/image id and information related to each term for all terms of a user/image: term value, term frequency (tf), document frequency (df) and tf-idf. Each location in the first location input file has a location id and location title. The second location input file has the location title and information related to each term for all terms of a location: term value, term frequency (tf), document frequency (df) and tf-idf. The location title is fetched on basis of the location id given as an input and this location title is used to get the terms and their frequencies. Based on the input dimensionality reduction model requested at run time, one of the following program branches is executed:

- Model= PCA: The data from the input file is parsed into a Pandas data frame with the user id/image id/location title as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. Then a scikit learn PCA decomposition is performed to reduce the dimensions to user input 'k' and these 'k' latent semantics are then sent to output with the semantics described in terms of the weights of the original dimensions (terms).
  For each latent semantic, the terms and their weights are then modelled into a Python dictionary with the term as the key and the weight as the value. This dictionary is then sorted in descending order of the term weights and this sorted term-weight pair is then sent to output for each of the 'k' latent semantic.

  Example for User-Term when 'k'=1:

Latent Semantic/Term	0.2	0.2m	베를린	브란덴부르크개선문
0	0.000027	0.000027	0.000027	0.000046

  Latent Semantic: 0
  [('브란덴부르크개선문', 0.000046), ('베를린', 0.000027), ('0.2m', 0.000027), ('0.2', 0.000027)]
  The 4 features (terms) of the input data are used to describe the one latent semantic requested by user. Then, these terms are sorted for the latent semantic in descending order of their weights.

- Model= SVD: The data from the input file is parsed into a Pandas data frame with the user id/image id/location title as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. Then a scikit-learn SVD decomposition is performed to reduce the dimensions to user input 'k'. This decomposition produces 3 matrices
  1. U: Indicates the objects in terms of the latent features
  2. Sigma: Indicates the strength of the latent features
  3. V^T: Indicates the k-latent features in terms of the old features.

  For each latent semantic, we fetch the term weights from the V^T matrix. The terms and their weights are then modelled into a Python dictionary with the term as the key and the weight as the value. This dictionary is then sorted in descending order of the term weights and this sorted term-weight pair is then sent to output for each of the 'k' latent semantic.

- Model= LDA: To perform LDA, the genism library in python has been used.

For LDA, the data is stored in a structure of the following format:

Users = { usr1 :{ "wd": [1, 2, 3]} , usr2 :{ "wd2": [1, 2, 1] , "wd3": [1, 2, 1]}} (Similarly for images and locations)

LDA operates on a list of documents to find topics. The number of documents in our case will be the number of users/images/locations respectively based on input and the document for each entity is populated with the associated terms of the entity where each term occurs 'tf' number of times in the document.

Example:

For Users, the document list will look like this:

Users	Document
U1	[a,a,a,b,b,c,d,d,e]
U2	[a,b,b,c,d,e,r,s,z]
U3	[f,a,a,a,d,d,d,d,d,e]
U4	[c,d,d,e,e,e,e,s,a]
U5	[f,q,s,d,a,a,a,g]

U1 above has the terms a,b,c,d,e in his term-space and their tf's are [3, 2, 1, 2, 1], hence the repetition of words in the document.

A word->ID mapping dictionary is made out of the global term-space where each unique word is mapped to an auto-generated ID. This dictionary will be used by the LDA computation function.

The document-term matrix is prepared where document represents user/image/location. The matrix contains the tf values. This document term matrix is a bag of words representation and for each document contains tuples of the form (word_ID, TF). Since this structure is not explicitly available, it had to be created using the tf values in order to pass it to lda.

This matrix is passed to the lda function with number of topics as a parameter along with certain hyperparameters (eg. number of passes) and the necessary document-topic and topic-term matrices are got.

The topic-term matrix has the distribution of the terms in each topic (weighted) which is required.

Task 2:

This task is an extension of Task 1. After the top 'k' latent semantics are identified, these latent features are then used to get an object-latent semantic matrix where each object (user, image or location) is described in terms of the 'k' latent features. Based on the input dimensionality reduction model requested at run time, one of the following program branches is executed:

- Model= PCA: We create data frames for all objects i.e. users, images and locations which are then integrated together using a union set of the features for all objects. The user inputs an object ID (user ID, image ID or location ID). Then, the cosine distance between the given object ID and all the object IDs is computed and 5 objects with least cosine distance from each category (5 from user space, 5 from image space and 5 from location space) are sent to output along with the distance between the input object and the respective object. Euclidean distance is used to compute the relativeness among the objects as Euclidean distance between similar objects is less than the Euclidean distance between dissimilar objects.

  Example for Location-term when 'k'=2

Latent Semantic/Term	0.2	0.2m	베를린	브란덴부르크개선문
0	0.000027	0.000027	0.000027	0.000046
1	0.000016	0.000008	0.000027	0.000046
2	0.000027	0.000016	0.000008	0.000027

Similar Users

User IDs	Euclidean Distance
22890158@N08	132.065
66204286@N00	137.898
64379474@N00	138.98
65613910@N08	140.903
77759596@N00	152.009

Similar Images

Image IDs	Euclidean Distance
4472413588	147.89
4472159392	147.90
4472535510	148.23
4472419172	148.90
4471758695	148.99

Similar Locations

Location IDs	Euclidean Distance
big_ben	0.0000
cn_tower	61.90
bok_tower_gardens	64.23
Cabrillo	68.90
Colosseum	71.99

- Model= SVD: We create data frames for all objects i.e. users, images and locations which are then integrated together using a union set of the features for all objects. The user inputs an object ID (user ID, image ID or location ID). Then, the cosine distance between the given object ID and all the object IDs is computed and 5 objects with least cosine distance from each category (5 from user space, 5 from image space and 5 from location space) are sent to output along with the distance between the input object and the respective object. Euclidean distance is used to compute the relativeness among the objects as Euclidean distance between similar objects is less than the Euclidean distance between dissimilar objects.

- Model= LDA: After the required 'k' topics are identified, these topics are used are then used to get a document-topic matrix where each object (user, image or location) is described in terms of the 'k' latent features (topics). Then, based on the input vector space, the required data-frame is considered and converted into a structure to find the necessary distances.

  The resultant structure looks like this:

[[.02,.2,.43,.114], [.02, .4, .6, .7] …..] where each of the sub list is a list of contributing values for that particular entity. The sub-list for the given entity is compared against all the other sub-lists to find distances and sort. Euclidean distance measure is used. The top nearest entities based on this is output.

Task 3:

Each location in the first input file has a location id and location title. The location title is fetched on basis of the location id given as an input. The second input file has the visual descriptors for each of the images whose file name was fetched from the location title of the first input file. The image file names are computed on basis of the location title and the model that was provided as an input by the user. Based on the input dimensionality reduction model requested at run time, one of the following program branches is executed:

- Model= PCA: The data from the input file is parsed into a Pandas data frame with the image id/location title as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. Then a scikit learn PCA decomposition is performed to reduce the dimensions to user input 'k' and these 'k' latent semantics are then sent to output with the semantics described in terms of the weights of the original dimensions (terms).

An image ID is taken as an input from the user, and the cosine distances between all the image IDs and the input image ID is calculated using the above calculated latent semantics and the 5 images with least distances are returned as output as the most related 5 image IDs with the cosine distance returned as the matching score. The first 5 unique locations with the least cosine distance among all the images are returned as an output as the most related 5 location IDs with the cosine distance returned as the matching score.

- Model= SVD: The data from the input file is parsed into a Pandas data frame with the image id/location title as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. Then SVD decomposition is performed to transform the data into three matrices which represent images in terms of latent features (U), strength of latent features (Σ) and latent features versus the original features(V^t).
  An image ID is taken as an input from the user, and the euclidean distances between all the image IDs and the input image ID is calculated using the above calculated images in terms of latent features (U) matrix and the 5 images with least distances are returned as output as the most related 5 image IDs with the Euclidean distance returned as the matching score. The first 5 unique locations with the least Euclidean distance among all the images are returned as an output as the most related 5 location IDs with the Euclidean distance returned as the matching score.

- Model= LDA: In this approach, we used the Latent Dirichlet Allocation function from the Scikit-learn library to perform the decomposition and identify the k latent semantics. The detailed procedure that was followed was:
  First, given a visual descriptor model, we aggregate the data for all the image files from that particular model into a single data frame for further processing. The data is normalized so that all the columns have values in the range [0,1].
  In the second step, we pass this data frame to the Latent Dirichlet Allocation function present in the Scikit-learn library. We obtain two key matrices from the LDA Decomposition. Let us assume we have n images, k topics and m features. The matrices obtained are:
  - An n x k [image, topic] matrix. The rows of this represent individual images and the columns of this represent the probability that an image belongs to a given topic. In other words, we get the probability distribution information for the images assigned to topics. For example, consider we have, N = 3, m = 3 and k = 3. Then consider the table below.

IMAGE TOPIC MATRIX

Image	Topic 1	Topic 2	Topic 3
Image 1	0.5	0.3	0.2
Image 2	0.6	0.3	0.1
Image 3	0.2	0.3	0.5

Probability that image 1 belongs to topic 1 is 0.5, topic 2 is 0.3 and topic 3 is 0.2

- A k x m [topic, feature] matrix. The rows of this matrix represent the k latent topics discovered and the columns represent the contribution of the different features to each topic.
  For example, below:

TOPIC FEATURE MATRIX

Topic	Feature 1	Feature 2	Feature 3
Topic 1	50	40	30
Topic 2	90	10	40
Topic 3	70	50	100

The strength of feature 1's contribution to topic 1 is 50, Feature 2's contribution to topic 1 is 40 and feature 3's contribution to topic 1 is 30.

Now the problem can be divided into 2 distinct parts:

1. Finding out the latent semantics of the image space
2. Returning the 5 most similar images and locations to the given image.

For the first problem, we took the following approach:

1. After obtaining the [topic, feature] matrix as described above, we list down all the rows of this topic feature matrix as our k latent semantics. This is because each row in this matrix represents a different topic, and each column represents a contribution of a feature to a topic, so we can obtain a description of the topics in terms of the original features.

   In the above example, our output would be:

   Topic 1 -> {50, 40, 30}

   Topic 2 -> {90, 10, 40}

   Topic 3 -> {70, 50, 100}

For the second problem, the approach followed was as follows:

Finding the 5 most similar images:

1. We consider the [image, topic] matrix in our data. This represents the distribution of topics among the images. Now, since this is our reduced feature space, we will compute the distances among the images in this new space and then sort them.

   Then we return the 5 images with the shortest distance to given image.

2. Example, consider the below matrix

Image	Topic 1	Topic 2	Topic 3
Image 1	0.5	0.3	0.2
Image 2	0.6	0.3	0.1
Image 3	0.2	0.3	0.5

Distance [Image1, Image2] = $\sqrt[2]{(0.5-0.6)^2+(0.3-0.3)^2+\ (0.2-0.1)^2} =$

$\sqrt{(0.1)^2+0^2+(0.1)^2}$ = $\sqrt{0.01+0+0.01} =\ \sqrt{0.02} = 0.141$

Finding the 5 most similar Locations:

1. Again, we consider the [image, topic] matrix in our data. Here, a separate mapping is maintained from the images to locations, which can help us identify which image belongs to which location.
2. After sorting the distances to the different images as seen in the previous example, we consider each of the images and add its corresponding location to the result set if that location has not been seen so far. This process is continued till we have our 5 target locations.
   For example, consider we have the following 10 images and 6 locations. Here, we first compute the distance of Image 1 from every image and sort them in ascending order.

   Distance of image from source image, sorted in ascending order:

Image	Distance
Image 1	0
Image 2	0.5
Image 3	0.7
Image 4	0.9
Image 5	1
Image 6	1.4
Image 7	1.6
Image 8	1.7
Image 9	2
Image 10	2.5

Suppose, we also have the following image to location mapping;

Image	Location
Image 1	2
Image 2	1
Image 3	1
Image 4	3
Image 5	3
Image 6	6
Image 7	6
Image 8	5
Image 9	6
Image 10	7

So, we iterate through the above images till we get 5 distinct locations. The unique locations are then picked and returned as the result.

- In our case, the answer is SET (2,1,1,3,3,6,6,5) = {2,1,3,6,5}

Task 4:

Each location in the first input file has a location id and location title. The location title is fetched on basis of the location id given as an input. The second input file has the visual descriptors for each of the images whose file name was fetched from the location title of the first input file. The image file names are computed on basis of the location title and the model that was provided as an input by the user. Based on the input dimensionality reduction model, one of the following program branches is executed:

- Model= PCA: The data from the input file is parsed into a Pandas data frame with the image id and location title as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. Then a scikit learn PCA decomposition is performed to reduce the dimensions to user input 'k' and these 'k' latent semantics are then sent to output with the semantics described in terms of the weights of the original dimensions (terms).
  A location ID is taken as an input from the user, and the cosine distances between all the location IDs and the input location ID is calculated using the above calculated latent semantics and the 5 locations with least distances are returned as output as the most related 5 location IDs with the cosine distance returned as the matching score.

- Model= SVD: The data from the input file is parsed into a Pandas data frame with the image id and location title as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. Then SVD decomposition is performed to transform the data into three matrices which represent images in terms of latent features (U), strength of latent features (Σ) and latent features versus the original features(Vt). Now the matrix which represents images in terms of latent features (U) is grouped according to locations.

A location ID is taken as an input from the user, and the cosine distances between all the location IDs and the input location ID is calculated using the above calculated images in terms of latent features (U) and the 5 locations with least distances are returned as output as the most related 5 location IDs with the cosine distance returned as the matching score.

- Model= LDA: In this approach, we used the Latent Dirichlet Allocation function from the Scikit-learn library to perform the decomposition and identify the k latent semantics. The detailed procedure that was followed was:

  First, given a visual descriptor model, we aggregate the data for all the image files from that particular model into a single data frame for further processing. The data is normalized so that all the columns have values in the range [0,1].
  In the second step, we pass this data frame to the Latent Dirichlet Allocation function present in the Scikit-learn library. We obtain two key matrices from the LDA Decomposition. Let us assume we have n images, k topics and m features. The matrices obtained are:

  1. An n x k [image, topic] matrix. The rows of this represent individual images and the columns of this represent the probability that an image belongs to a given topic. In other words, we get the probability distribution information for the images assigned to topics. For example, consider we have, N = 3, m = 3 and k = 3. Then consider the table below.

IMAGE TOPIC MATRIX

Image	Topic 1	Topic 2	Topic 3
Image 1	0.5	0.3	0.2
Image 2	0.6	0.3	0.1
Image 3	0.2	0.3	0.5

Probability that image 1 belongs to topic 1 is 0.5, topic 2 is 0.3 and topic 3 is 0.2

2. A k x m [topic, feature] matrix. The rows of this matrix represent the k latent topics discovered and the columns represent the contribution of the different features to each topic. For example, below:

TOPIC FEATURE MATRIX

Topic	Feature 1	Feature 2	Feature 3
Topic 1	50	40	30
Topic 2	90	10	40
Topic 3	70	50	100

The strength of feature 1’s contribution to topic 1 is 50, Feature 2’s contribution to topic 2 is 40 and feature 3’s contribution to topic 1 is 30.

Now the problem can be divided into 2 distinct parts:

1. Finding out the latent semantics of the image space
2. Returning the 5 most similar locations to the given location.

For the first problem, we took the following approach:

After obtaining the [topic, feature] matrix as described above, we list down all the rows of this topic feature matrix as our k latent semantics. This is because each row in this matrix represents a different topic, and each column represents a contribution of a feature to a topic, *so we can obtain a description of the topics in terms of the original features.*

In the above example, our output would be:

Topic 1 -> {50, 40, 30}

Topic 2 -> {90, 10, 40}

Topic 3 -> {70, 50, 100}

For the second problem, the approach followed was as follows:

<u>Finding the 5 most similar Locations:</u>

Again, we consider the [image, topic] matrix in our data. Here, we have *a master data frame* that is containing the data of all the image files belonging to *a particular visual descriptor model*. The data frame consists of the following.

<u>Image ID column</u>**:** A column to store the image id for each image

<u>Topic columns:</u> Columns representing the topics discovered. Each column represents the probability of the image to belong to that topic.

<u>Location name column:</u> A column which indicates the location name for a given image id.

Example,

<u>Image</u>	<u>Topic 1</u>	<u>Topic 2</u>	<u>Location</u>
Image 1	0.5	0.5	A
Image 2	0.4	0.6	A
Image 3	0.1	0.9	A

To compute the 5 most similar locations the following steps were followed.

1. We extracted all the data corresponding to our base/source location in a single base data frame. This would contain all information about the images in our base location. For example, if our inputs were
   Visual descriptor model: CM
   Location: Angkor_wat

   Then all the data from the file Angkor_wat.CM would be present in the base data frame. The data frame above is constructed based on a *location_name* column, which indicates which location a particular image file belongs to.

2. Next, we iterate through each of the locations in our database and perform the following steps:

   A. Create a data frame storing the information *for all the images from that particular location.*
   B. For every image in the source location, find out the nearest image in the target location based on a distance measure. The Euclidean distance metric was used.
   C. Sum up these distances to obtain a measure of the total distance between the two locations.

   For example, consider we have data corresponding to 2 locations A and B as follows.

   Location A:

Image ID	Topic 1	Topic 2	Location
Image1	0.4	0.6	A
Image2	0.9	0.1	A

   Location B:

Image ID	Topic 1	Topic 2	Location
Image3	0.7	0.3	B
Image4	0.5	0.5	B

   Now for location A, we will try to find the nearest image in Location B:

   Distance [Image1, Image3] = $\sqrt{0.3^2 + 0.3^2}$ = $\sqrt{0.09 + 0.09}$ = $\sqrt{0.18}$ = 0.424

   Distance [Image1, Image4] = $\sqrt{0.1^2 + 0.1^2}$ = $\sqrt{0.01 + 0.01}$ = $\sqrt{0.02}$ = 0.14

   Hence, we can see that the closest image to Image1 in Location B:

Min (0.424,0.14) = 0.14
Thus, Image 4 is the closest.

Distance [Image2, Image3] = $\sqrt{0.2^2 + 0.2^2}$ = $\sqrt{0.04 + 0.04}$ = $\sqrt{0.08}$ = 0.28

Distance [Image2, Image4] = $\sqrt{0.4^2 + 0.4^2}$ = $\sqrt{0.16 + 0.16}$ = $\sqrt{0.32}$ = 0.565

Hence, we can see that the closest image to Image1 in Location B:
Min (0.28,0.565) = 0.28
Thus, Image 3 is the closest.

Thus, the distance between the two locations can be computed as:
Distance [Image1, Image4] + Distance [Image2, Image3] = 0.14+0.28 = 0.42

- Using the procedure defined above**,** *we compute the distances from our input location to all other locations and sort the locations in ascending order of distance. Then we fetch the first 5 locations and display them.*

Task 5:

Each location in the first file has a location id and location title. The location title is fetched on basis of the location id given as an input. The second input file has the visual descriptors for each of the images whose file name was fetched from the location of the first input file. The image file names are computed by concatenating all available model names to the location title fetched from the first file. Based on the input dimensionality reduction model, one of the following program branches is executed:

- <u>Model= PCA:</u> The data from the input file is parsed into a Pandas data frame with the image id, location title and visual descriptor model as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. A list of all visual descriptor models is fetched from the data frame. Then, for each model, a scikit learn PCA decomposition is performed to reduce the dimensions to user input 'k' and these 'k' latent semantics are then sent to output with the semantics described in terms of the weights of the original dimensions (terms). Thus, 10 matrices are sent to output, one for each visual model and each matrix having latent semantics described in terms of original features.
  Based on each of the latent semantics, cosine distance is calculated between the locations and 5 locations with the least cosine distance from the given location are sent to the output with the cosine distance as the matching score.

- <u>Model= SVD:</u> The data from the input file is parsed into a Pandas data frame with the image id, location title and visual descriptor model as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. A list of all visual descriptor models is fetched from the data frame. Then SVD decomposition is performed to transform the data into three matrices which represent images in terms of latent features (U), strength of latent features (Σ) and latent features versus the original features(Vt). Then we group the resulting matrix with images (U) in terms of latent features according to visual descriptor model. For each data frame belonging to a model, we group with location and then distance is calculated between given location and all other locations. Based on each of the latent semantics, Euclidean distance is calculated between the locations and 5 locations with the least cosine distance from the given location are sent to the output with the cosine distance as the matching score.

- Model= LDA: The data from the input file is parsed into a Pandas data frame with the image id, location title and visual descriptor model as the index and the terms (features) describing the item as columns of the data frame. The value of the terms not used by an item are set to zero for calculation purposes. A list of all visual descriptor models is fetched from the data frame. Then, for each model, a LDA with gensim is performed to reduce the dimensions to user input 'k' and these 'k' topics are then sent to output with the semantics described in terms of the weights of the original dimensions (terms). Thus, 10 matrices are sent to output, one for each visual model and each matrix having topics described in terms of original features. Based on each of the latent semantics, Euclidean distance is calculated between the locations and 5 locations with the least Euclidean distance from the given location are sent to the output with the Euclidean distance as the matching score.

Task 6:

- Creating location-location similarity matrix: We created the location-location similarity matrix using location-terms data frame. Location-term data frame was created using the terms that represent the respective location (TF values of each term/word). Dot product of location-term and transpose of location-term matrices resulted in the location-location similarity matrix.

- Performing SVD: We made use of NumPy's linear algorithm SVD. It requires the data frame as an input. Using NumPy's SVD we fetch the U, sigma and V matrices.
    - U ~ Representation of Objects in terms of k latent semantics
    - sigma ~ Core matrix – importance of k latent semantics
    - V ~ Representation of k latent semantics in terms of original features

- Displaying the k latent semantics: We need to represent the latent semantics in terms of original features (i.e. features of original data frame). V matrix is the representation of k latent semantics in terms of original features. So, we display the top k rows of the V matrix obtained.

Task 7:

We read and store the data in the following format:

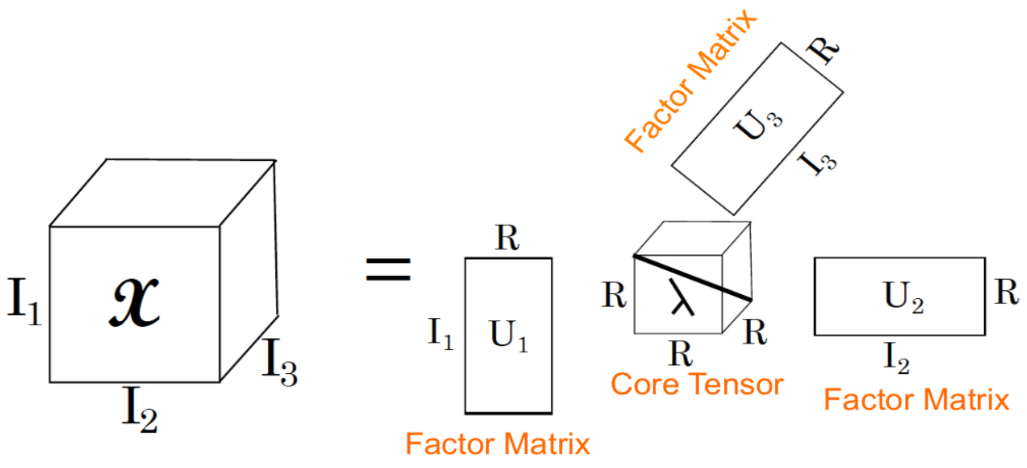
```
dictionary_of_users = {
user_1: set(set_of_words for user_1),
.
.
user_i: set(set_of_words for user_i)
}
dictionary_of_images = {
image_1: set(set_of_words for image_1),
.
.
image_j: set(set_of_words for image_j)
}
dictionary_of_locations = {
location_1: set(set_of_words for location_1),
.
.

location_k: set(set_of_words for location_k)
}
```

- Creating a tensor:

Each element in the tensor represents the relation between user_i, image_j and location_k. The value of the element is the total number of shared words between them, so we used set intersection to populate the value in each element of tensor using the data created earlier.

- Performing CP Decomposition:



I1 ~ Users (length of Users)
I2 ~ Images (length of Images)
I3 ~ Locations (length of Locations)
R ~ K (Rank)
U1 ~ Users x K
U2 ~ Images x K
U3 ~ Locations x K
Lambda ~ K x K x K

We made use of tensorly for CP decomposition. parafac from tensorly.decomposition takes the tensor and rank as input parameters, performs CP decomposition using ALS and provides factor matrices as output.

- Displaying Factor matrices: We have displayed each of the factor matrices obtained from the decomposition. Also displayed are the shape (dimensions) of each of the factor matrices obtained.

System Requirements

- Operating System: Windows x64 architecture
- Python version: 3.7
- Packages: NumPy, Pandas, gensim, sklearn, tensorly

Installation

- We have created a zip file which contains program codes for each task along with another program that can make a sub process call to each of the above-mentioned programs.
- Install NumPy: pip install numpy
- Install Pandas: pip install pandas

- Install genism: pip install genism
- Install sklearn: pip install sklearn
- Install tensorly: pip install tensorly

Execution Instructions

- Tasks:
  - Command: python tasks.py
  - Enter choice of task: 1, 2, 3, 4, 5, 6, 7 or 8
- Task 1: Enter <<vector_space model k>> : user pca 5
- Task 2: Enter <<vector_space model k itemID>> : location svd 3 10
- Task 3: Enter <<visual_descriptor_model model k imageID>> : cm3x3 lda 4 9067739127
- Task 4: Enter <<location_id visual_descriptor_model model k>> : 10 cn svd 8
- Task 5: Enter <<location_id model k>> : 10 pca 6
- Task 6: Enter << k >> : 10
- Task 7: Enter the CP- rank: 5

Conclusion

We have implemented a set of program codes which decompose data matrices of different object feature vector spaces to provide set of latent semantics. These set of latent semantics are then used to calculate similarity measures between the objects and output similar or related objects. We have also implemented tensor creation and CP-decomposition.

Bibliography

- Data Management for Multimedia Retrieval, K. Selçuk Candan and Maria Luisa Sapino, Cambridge University Press, 2010
- David M. Blei, Andrew Y. Ng and Michael I. Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Research 3 (2003) 993-1022
- Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. SIAM Review, 51(3):455–500, September 2009

Appendix

- Aniket and Vasisht have implemented SVD from task 1 to 6.
- Siddharth and Aditya have implemented LDA from task 1 to 5.
- Shatrujit has implemented PCA from task 1 to 5.
- Siddhesh has implemented task 7.