# CSE 515: Multimedia and Web Databases

# Phase 2

# "Dimensionality Reduction of Features"

[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 reduce the dimensionality curse by feature reduction methods such as PCA/SVD and LDA model. The project deals with the efficient retrieval of data and reducing the data to carry out similarity measure among different types of users and 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, Euclidean, PCA, SVD, LDA, Cosine, NoSQL, Tensor

## 1. Introduction

Mankind 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. Thus, it is beneficial to reduce the dimensions of the dataset to makes things less complicated. In this phase of project, our aim is to implement the dimensionality reduction techniques taught in class, such as SVD, PCA, LDA and CP decomposition.

### 1.1 Singular Value Decomposition

SVD is directly applied on *Object-Feature* matrix itself.

$S = PCP^{-1}$, *where S* is original object-feature matrix, *C* is diagonal matrix of eigenvalues and *P* is an orthonormal matrix consisting of the eigenvectors of *S*.

### 1.2 Principal Component Analysis

For a given dataset, PCA uses co-variance matrix and identifies alternative bases along which spread is maximum. It is based on Eigen Decomposition.

### 1.3 Latent Dirichlet Allocation

LDA tries to find a decision boundary around each cluster of a class. It then projects the data points to new dimensions in a way that the clusters are as separate from each other as possible and the individual elements within a cluster are as close to the centroid of the cluster as possible.

### 1.4 NoSQL Database

The NoSQL database is especially designed to handle the unstructured data. The project had high amounts of unstructured data and NoSQL handles the storage and retrieval of such data efficiently. It is not relational database. The records are stored in form of collections and documents in NoSQL database. We selected MongoDB for the purpose.

## 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.

## 3. Problem Description

This phase of project has 5 tasks which deals with dimensionality reduction and then selection top latent semantics and finding most related entities (users/locations/images).

The detailed specification of every task is on the following:

**Task 1**

Implement a program which lets the user to choose among

1. User-term vector space
2. Image-term vector space, or
3. Location term vector space,

and then, given, a positive integer value, k, identifies and reports the top-k latent semantics/topics in the 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 task1 such that, after the top-k latent semantics are identified, given

1. User ID,
2. Image ID, or
3. Location ID,

the system also identifies the most related 5

- ➢ User IDs,
- ➢ Image IDs, and
- ➢ 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", first identifies (and lists) k latent semantics using

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

on the corresponding image feature space, then given an image ID, identifies the most related 5 image IDs and 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, 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 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, 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 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, creates a location-location similarity matrix, performs SVD on this location-location similarity matrix, and 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, creates a user-image-location tensor (based on number of terms shared by all three), performs rank-k CP decomposition of this tensor, and creates k non-overlapping groups of users, images, and locations based on the discovered latent semantics.

## 4. Problem Solutions

### 4.1 Libraries used

**For SVD**

- Initially we thought of using Truncated-SVD, but the library generates an error when we give the value of input *k* (latent semantics) equal to *n* (total features)). Hence it holds a tight bound that k<n.
- Hence, we went with the scipy library function linalg() which behaves in the same way as gives the same output as Scikit learn SVD library but with handling the previous error mentioned.

**For PCA**

- Scikit-learn – It calculates the covariance matrix and then processes the eigen decomposition, depending on the shape of the input data and the number of components to extract. We are using StandardScaler() to standardize the data points and make the computations easier.
- We can specify number of components to keep (say *k*) and it gives us transformed dataset along these k components. It is much efficient and gives results in considerable time, thus we are using this library.

**For LDA**

- Scikit-learn – It calculates the features in terms of the latent semantics, similar to PCA and SVD but in a probabilistic environment. We use the LatentDirichletAllocation() in this case. It is based on conditional probability and the distribution is Dirichlet.
- We specify number of components to keep (say *k*) and it gives us transformed dataset along in these k components.

**For Tensors**

- We are using the ‘tensorly’ package library to perform tensor decomposition. This package allows us to perform different tensor decomposition starting from CP to Tucker decomposition. Parafac function takes n dimensional array and rank as input and returns the n factor matrices corresponding to each dimension.

**Numpy and Pandas**

- We have used numpy for built in calculation functions such as min(), mean() and the similarity measures Euclidean() and cosine().
- For such a large-scale data, simply retrieval from the MongoDB database and creating the matrix using the loops was causing performance and memory issues for us and insisted our choice to use pandas to create DataFrames in form of rows and columns to create the required matrices.

## 4.2 Task Approach Explanation

**Task 1**

Initially, we created a list of the distinct terms used by all the users to get the common users and a list of all the different/uncommon users. For this, we made a DataFrame of our json file (described in Readme.txt) related to user/image/location. We extracted the words used by different users/images/locations and put them in a set so that we don't get repeated words, and make a list of this set. Similarly, we extracted all the users from the list. This list of words will represent the columns of our base dataset with users/images/locations ID as row Index. To fill the matrix with data, we need the TF_IDF value for each word used by a user/image/location. We used the TF_IDF value as it covers aspects of distribution of word across a document as well as across multiple documents (TF and DF). We created 2 dictionaries, one representing all the words used by a user/image/location, another representing the TF_IDFs of those words at the same index. To fill the matrix, we run a loop for all the terms in our term superset and an inner loop over all the users/images/locations to see whether that term is used by the user/image/location. If the term has been used by the user/image/locations then we extract the index of the term from the corresponding dictionary and using this index to extract the TF_IDF from the other dictionary. The output from this loop is the feature to object matrix. We do the transpose to convert it into object to feature matrix. We have used Scikit libraries for all three operations of SVD, PCA and LDA. In case of SVD or PCA, we have used matrix described by $V^t$. This matrix is representing our latent features in terms of word used by users/images/locations.

To represent the top k latent semantics in the form of term-weight pair, we sorted the term weights on individually for each latent semantic (k new lists will be there representing each latent semantic). We print this sorted list to show term-weight pairs in descending order of their weights.

**Task 2**

After reducing the dimensions using PCA/SVD/LDA algorithm based on the user's input and identifying the top k latent semantics for given space and k value in Task 1 will extend the program to identify the top 5 objects i.e. USER_ID/IMAGE_ID/LOCATION_ID for a given space based on user's input.

Steps to identify top 5 objects i.e. USER_ID/IMAGE_ID/LOCATION_ID among each space with respect to the given object:

In the Task 1, given input is object-feature matrix where values are TF-IDF (reason mentioned in Task 1) which acts as discriminator, reduced the dimensionality of the features for given space and identify top k latent semantics (features)

In Task 2, we had to choose the corresponding values of the object, identify top 5 similar objects among each space. After getting the reduced dimensionalities of the chosen space and given corresponding space's object ID, we need to find the values of the corresponding ID using any dimensionality reduction algorithm.

To find the top 5 similar objects among each space. Please find below steps to find top 5 similar objects among each space:

- After dimensionality reduction of the chosen space, multiply the latent-semantic-feature ($V^t$) matrix with each space's corresponding object-feature matrix which will give the new matrix where each space's objects will be represented in k-dimensionality.
- We need to find top 5 similar objects among each space with respect to give object(user/location and image)ID. Hence, find the similarity between given object and other objects. We are using Cosine Similarity measure algorithm because Cosine similarity makes sense in a way that the angle matter for the terms more as similar context terms can have different TF but high similarity Euclidean distance though they mean the same. Hence, cosine will capture such nuances and is preferred in case of the terms. (same reason for task 3 as well)

- Identify top 5 scores and print the corresponding objects for all the spaces.

**Task 3**

Dimensionality reduction algorithms used: PCA/SVD/LDA

*Similarity/distance algorithm:* Cosine Similarity in this case because Cosine similarity makes sense in a way that the angle matter for the terms more as similar context terms can have different TF but high similarity Euclidean distance though they mean the same. Hence, cosine will capture such nuances and is preferred in case of the terms.

Dimensionality reduction of features:

Input: Matrix (object-feature)

1st dimension – objects as Images

2nd dimension –Features as given model and its corresponding values

Output: Matrix (objects-latent_semantic)

1st dimension – objects as Images

2nd dimension –k latent semantics and its corresponding values

To identify top 5 similar Images for given Image:

- Retrieve corresponding Image's values from the object-latent_semantic matrix. Find the similarity between this and other images using Cosine Similarity (the reason is same as mentioned above in Task 2). Then Retrieve top 5 similar images which are similar to given Image

To identify top 5 similar locations for given Image:

- Retrieve corresponding Image's values from the object-latent_semantic matrix. Find the similarity between each location's images with given image and find the mean of all the similarity scores of that location. Retrieve top 5 similar locations which are similar to a given Image based on similarity score.

**Task 4**

*Aim:* To find k Latent Semantics for the given Location ID using PCA/SVD/LDA for a given Visual Descriptor Model. Also, to find the 5 most similar location.

*Storage of data:* MongoDB for efficient retrieval *Approach:* We created the data matrix of images (8423) x scores of all the locations (30) as the *master data matrix*. Also, we created a data matrix for the query location in form of images (~300) x scores (Model based length) where scores are the features and images are the objects as per the input of the location ID. The matrix is constructed by extracting the images and scores data from MongoDB as per location and the scores are distributed into different rows by using the dataframe in pandas. We have used pandas as it has proved to be an efficient and faster retrieval of the data from the database. To generate the Latent Semantics of the query location, the query location data matrix is passed to the PCA/SVD/LDA function as selected by the user for eigen decomposition. As the data matrix gets called to any of the feature reduction function, the number of components are generated using the K value and the corresponding library. For PCA and LDA, we have used the scikit learn library and for SVD we have used the scipy library. The reason for using these specific libraries is mentioned in the previous section.

Negative Values issue in LDA: In the case of LDA, the negative values have been dealt by doing a global shift on the minimum value of the matrix and transformed into a positive matrix as LDA doesn't accept negative values.

The components generated from the dimensionality reduction function gives us the features (Scores) in terms of Latent semantics of the query location for a given model which satisfies the first requirement of Task 4. The feature matrix (V transpose is projected onto the original matrix to get the new feature space. (reduced features)

For the second part of task 4, the features x latent semantics matrix (Resultant from the above) is projected on the master data matrix of images x scores to represent all the images in terms of the query location's latent semantics. Hence, we are representing all the images as per location in terms of latent semantics of the query location. This brings the new master data matrix on the same dimension as the query location components. We compute the similarity between the query location matrix (300 x K) and the new master data matrix (Images x K) using the Cosine similarity and store the mean value as per location in a dictionary. Then we take out the top 5 similarity scores from the dictionary to get the top 5 most related locations to the given query location.

The *Cosine similarity measure:* is being used because Cosine similarity covers the composition of the features(scores), on how the scores are different from each other for an image. How the differences in score signifies the differences in images. Since image is a composition of scores in this case, we compare the Model of the query location with the same Model of all the other 30 locations to know how the images of the model are contributing with their scores composition using the cosine similarity. It takes into account the spatial relationships between the visual descriptors of the image.

**Task 5**

*Aim:* To find k Latent Semantics for the given Location ID using PCA/SVD/LDA across all the Visual Descriptor Models. Also, to find the 5 most similar location.

*Approach:* This is similar to the approach of Task 4 except the number of models increases in this scenario. We created an original data matrix for the query location in form of images (~300) x scores (Model based length) where scores are the features and images are the objects as per user input of the location ID. The matrix is constructed by extracting the images and scores data from MongoDB as per location and distributed the scores into different rows using the dataframe in pandas. The query location matrix is constructed across all the 10 Models. To generate the Latent Semantics of the query location across the 10 models, the query location data matrix is passed as model by model to the PCA/SVD/LDA function as selected by the user for eigen decomposition. The resultant Latent Semantics of the query location is stored in a dictionary for further comparisons with all the models of all the locations.

As the data matrix gets called to any of the feature reduction function, the number of components were generated using the K value and the corresponding library. The components generated from the dimensionality reduction function gives us the features (Scores) in terms of Latent semantics of the query location on a *particular model* as pointed by Task 5. The feature matrix (V transpose is projected onto the original matrix to get the new feature space with reduced features of the particular model.

For the second part of task 5, the features x latent semantics matrix (Resultant from the above) is projected on the original Scores matrix of other locations model by model to represent their images in terms of latent semantics of the particular model. This brings the score matrix of the location on the same dimension as the query location components. We compute the similarity between the query location matrix (300 x K) and the new score matrix of the other locations using the Euclidean distance and store the mean value as per location in a dictionary model_similarity. We calculate the similarity between the corresponding models (CM-CM and so on) and take the mean value of the similarity to store the final result. We have taken the 'mean' of the similarities as mean covers all the data point's(model's similarity contribution for a location) contribution towards the value. Then we take out the top 5 similarity scores from the Model_Similarity dictionary to get the top 5 most related locations to the given query location.

The *Euclidean similarity measure:* is being used because Euclidean distance works well with the dense matrices. Here, the size of matrix is large and iterates over all the ten visual models. Within that, the location loop iterates 30 times each and then corresponding computations takes place. Additionally, we had also noticed in Phase 1 and in this task itself, that the Euclidean distance computations occur faster than the Cosine similarity without having a significant impact on the similarity. And the distance between two scores made sense in a way that there are two pixels on the space and we use their spatial coordinates to compute the distance in an easy formulated way.

**Task 6**

*Aim:* To work with the textual descriptors of the location data and apply the dimensionality reduction algorithm to identify the top k latent semantics of the locations data. Unlike the other tasks where we use object – feature matrix for dimensionality reduction, here we are using the object-object similarity matrix for dimensionality reduction.

Goal of the task is given a 'k' value, create a location to location similarity matrix, perform SVD (singular valued decomposition) on this matrix and report the top k latent semantics in the form of location- weight pairs in the descending order. The approach to achieve as mentioned below. Fetch all the distinct or unique list of terms used by all the locations from Mongodb and to construct location term matrix find the TF_IDF value for these terms for all locations, if the term is not present in the location update the value to '0' else the actual TF_IDF value of the term. Why we are using TF_IDF? Because this value represents the how good/ discriminator the feature(term) across the document.

Once location -Term matrix D is constructed, we need to find the location to location similarity matrix D(Dt) using Euclidean distance. Why Euclidean Distance? Reason for using the Euclidean distance, we had noticed from Phase 1 and in task 4 that the Euclidean distance computations occur faster than the Cosine similarity without having a significant impact on the similarity.

After finding the location-location similarity matrix apply Singular Valued Decomposition (SVD) for this matrix. For SVD we are using the SciPy package library(linalg). This will accept the 2 D array and returns the 3 matrices U, S, $V^T$ (U- represents the n objects in terms of k latent semantics. S represents the eigen values in decreasing order. $V^t$ represents the k latent semantics explained in terms of n objects)

Note: U and V matrices are identical here as our input matrix is location - location symmetric matrix. To report the top k latent semantics, we are taking first k rows from U matrix and ordering each latent semantic in the descending order of name- weight pair.

**Task 7**

*Aim*: To experiment with the multi-modal and multi-dimensional data, apply dimensionality reduction algorithm on this data to find the latent dimensions and report the data in terms of identified latent semantics.

*Approach:* For this task, a tensor is a 3-dimensional array of location-user-image entities each of these entities are modes and they represent one dimension. Given a k value, create a user-Location-image tensor based on the number of common terms shared across all the 3 modes. Approach to build a tensor is, loop through all the users and get the terms used by each user, loop through all the location for each location get the terms used by each location, then get the common terms between location and user, get all the images, for each image get the terms used by image and find the common terms again across location-image and user. Get the count of common terms shared across 3 modes and build the 3D array which is tensor of order 3. Once tensor is ready, apply the Parafac /CP decomposition. Rank means the number of latent semantic to return after the decomposition. In this task, we input a 3-dimensional array user-location- image and the rank=k (accepted as input in the task). CP decomposition is Higher order SVD. The principle behind this decomposition is SVD only change is it operates on higher order matrices. As we learnt in class this decomposition is basically not accurate. The data obtained will not represent the original data perfectly. To reduce the error, it uses the ALS (alternate least square) algorithm. Once we Identify the factor matrices for each dimensions, we have to report the k non overlapping groups of users-location-images. To find the k non-overlapping groups we are applying the K-mean clustering, which basically means that, it clusters the factor matrices into different groups of user/locations and images in the new space based on their distribution. We are using the sklearn package library, which has the function to accept the n-dimensional factor matrix and number of clusters required and returns back the cluster numbered from 0-n.

Based on the data returned from this library we are displaying the respective cluster of the users/locations/users.

## 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 | scikit-learn | 0.20.0 |
| 4 | pandas | 0.23.4 |
| 5 | tensorly | 0.4.2 |

- *Execution Steps:*

  *Refer: Readme.MD*

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

## 7. Conclusion

We have successfully implemented all the 7 tasks as per problem specification of this phase with good understanding of dimensionality reduction and object 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 SVD, PCA, LDA, CP-Decomposition along with k-means clustering and multiple similarity measures. 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 data matrix is formed for given objects and features to dump it in the feature reduction algorithms such as PCA/SVD/LDA. We got to know on how the transformation happens using these packages, how these packages work. We saw the application side by the practical implementation of the theoretical concepts in class on how to deal with dimensionality curse provided the given challenges. We faced the issues in terms of memory, or constructing such a huge matrix, along with the algorithm package constraints on how it behaves differently from each other but with a common purpose of reducing the dimension. We found ways of handling the same by Pandas for constructing the matrix, and managing the loops efficiently.

## Bibliography

1. https://en.wikipedia.org/
2. https://stackabuse.com/implementing-lda-in-python-with-scikit-learn/
3. https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.linalg.svd.html
4. http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
5. K Selcuk Candan & Maria Luisa Sapino, "*Data Management for Multimedia Retrieval*", Cambridge.org

## *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. | Research on libraries to be used and its output | All |
| 4. | Task1 | Vishal, Jagriti |
| 5. | Task2 | Akshay, Shrikrishna |
| 6. | Task3 | Akshay, Jagriti |
| 7. | Task4 | Anik |
| 8. | Task5 | Anik, Jagriti |
| 9. | Task6 | Manoj |
| 10. | Task7 | Manoj, Vishal |
| 11. | Testing | All |
| 12. | Readme | Shrikrishna, Akshay |
| 13. | Project Report | All |

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