Multimedia and Web Databases Project Report

Phase II

Instructor
K. Selçuk Candan

Submitted by
Sharma Vartika

ARIZONA STATE UNIVERSITY

22 October, 2017

Contents

Abstract

Generally, we do not feed a large number of multimedia features extracted from the web database directly into an algorithm because they are very costly, and they slow down the computations. Further issues of storage space and communication bandwidth concerns, motivate media compression algorithms as well as coding and encoding algorithms. For the multimedia data management using content-based and object-based query processing, and how media objects affect users, we require better understanding of the underlying perceptive and cognitive processes in media processing. For most of the media object types, there are multiple features that one can be used for indexing and retrieval of the media objects. In this project, we will study and implement such techniques for media transformation, so that we can retrieve and decorrelate our input data. Some techniques implemented in this project phase are Principal Component Analysis, Cosine Similarity, Singular Value Decomposition, CP Decomposition and PageRank.

Keywords PageRank, Principal Component Analysis, Singular Value Decomposition, Tensor decomposition

1 Introduction

We implement various techniques for reducing the number of dimensions in the raw data. Principal component analysis is one method of reducing the number of dimensions in the raw data. To capture as much information from data as possible in a low number of dimensions, we find a basis of principal components. Each principal component is the vector along which variance is maximized, conditioning on it being orthogonal to all preceding principal components. [1] In linear algebra, the singular-value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigen decomposition of a positive semi-definite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any m x n matrix via an extension of the polar decomposition. It has many useful applications in signal processing and statistics. Latent Dirichlet allocation (LDA) is a topic model that generates topics based on word frequency from a set of documents. LDA is particularly useful for finding reasonably accurate mixtures of topics within a given document set. In multi-linear algebra, the tensor rank decomposition or Canonical Polyadic decomposition (CPD) may be regarded as a generalization of the matrix singular value decomposition (SVD) to tensors.

1.1 Goal description (problem specification)

In this project report we will be using the Term Frequency-Inverse Document Frequency (TF-IDF) weighted tag vector for different terms in a document (from the phase one of the project). Tags are classified and information is retrieved according to different criteria such as all the movies an actor has played in, all movies of a given genre, all the movies watched by a particular user and the difference between two genres. Hence, we do the classification of the documents using a vector-based model for document representation. Then we will implement the various techniques for reducing the number of dimensions mentioned above.

For our project, we are given the $IMDB$ and $MovieLens$ Data including mlmovies(movieId; movieName; genres), mltags(userId; movieId; tagid; timestamp), mlratings(movieId; userId; imdbId; rating; timestamp), genome-tags(tagId; tag), movie-actor (movieId; actorId; actorMovieRank), imdb-actor-info (actorId; name; gender), mlusers(userId) in the form of csv files.

1.2 Terminology

Similarity Matrix A similarity Matrix or similarity function is a real-valued function that quantifies the similarity between two objects. Although no single

definition of a similarity measure exists, usually such measures are in some sense the inverse of distance metrics.

Objects A database object is any defined object in a database that is used to store or reference data. Some examples of database objects include actors, movies etc. These are the objects which are described using features like genres, tags, timestamps etc.

Features Features are attributes related to the objects which describe the objects of a database. Some examples of features include tags, genres, timestamps etc.

Latent Semantics/ Topics Latent semantics are the hidden features or unobserved features in a database unlike the observed features like tags, genres etc. These can be usually by applying some kind of decomposition on the object-feature or object-object or feature-feature data matrices. They give information about the relationships between the objects.

Tensors A data tensor is a 3-dimensional representation of a database. Tensor is mainly useful when the database has a temporal dimension that cannot be captured in a single snapshot. Similar to vector decomposition, tensor can be decomposed to obtain factor matrices.

TF-IDF Term Frequency-Inverse Document Frequency is a text mining technique used to categorize the given set of documents. TF-IDF classifies documents into categories inside the documents. [2] This gives us knowledge of the reviews.

K-means Clustering K means Clustering classifies a set of observations into k clusters using the k-means algorithm. The algorithm attempts to minimize the **Euclidian distance** between observations and centroids. Several initialization methods are included.

Degrees of Membership The membership function of a fuzzy set is a generalization of the indicator function in classical sets. In fuzzy logic, it represents the degree of truth as an extension of valuation. Degrees of truth are often confused with probabilities, although they are conceptually distinct, because fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition.

1.3 Assumptions

For our implementation, we will take the following assumptions:
1) In Task 1a, the number of movies in the given genre must be greater than the number of Latent Semantics needed.
2) If a user has rated a movie (mlratings.csv), then the user has has also watched the movie.
3) If no user has rated the movie, then its rating is 0.
4) Ratings are taken as integers 1,2,3,4,5.
5) For Task 3 PageRank implementation,

$$T = aM + (1 - a)\left[\frac{1}{N}\right]$$

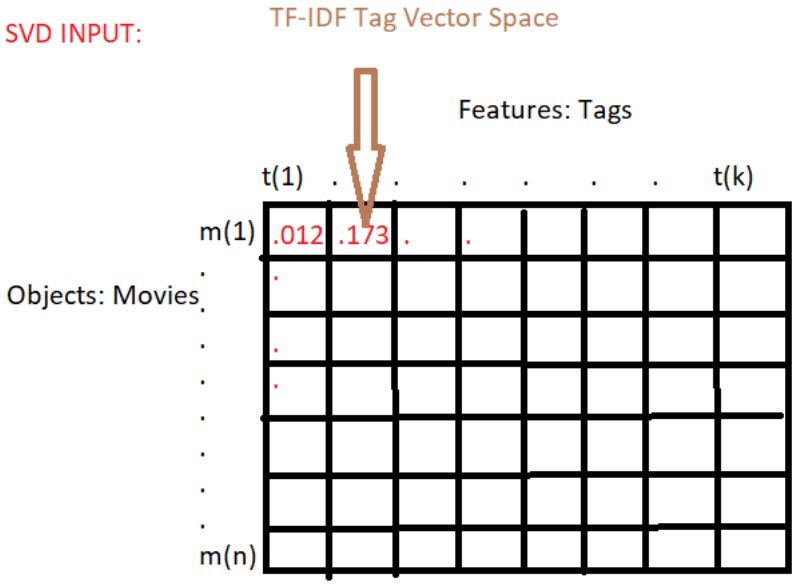
Here, random surfer follows one of the available links with the probability $a = 0.85$ and we take random surfer probability that represents the probability with which the random walk through the graph will deviate from its edges and instead jump randomly to any node in the graph as, $rsp = 1 - a = 0.15$
6) For Task 4 PageRank Implementation, α value is 0.15 because we wish to give more weight to the Seed Vectors.
7) The value of parameter α in LDA implementation is '1' and it follows a symmetric distribution. By assuming so, the peak of Dirichlet distribution will be in the center.

2 Description of the proposed solution/implementation

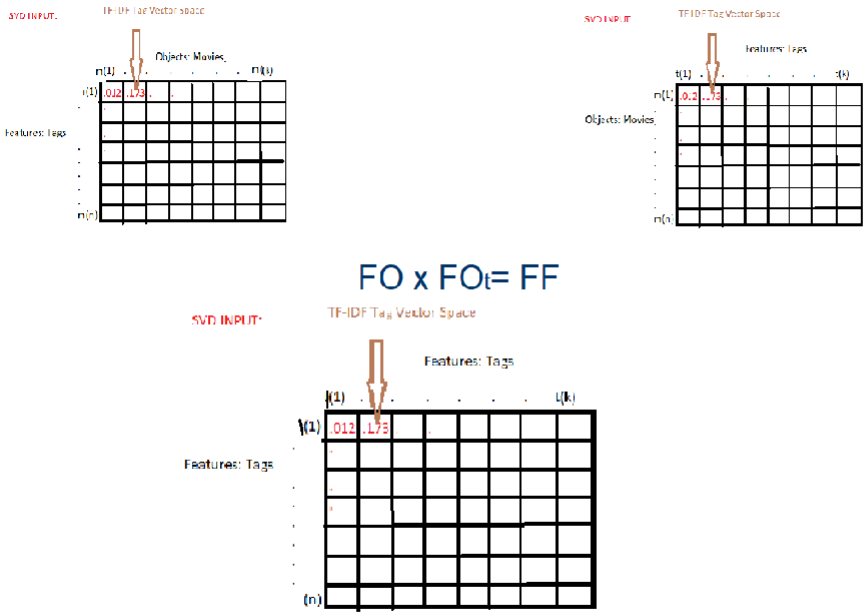
2.1 First Task

In the first task, we implement a program which, given a genre, identifies and reports the top-4 latent semantics/topics using PCA in TF-IDF space of tags(/actors), SVD in TF-IDF space of tags(/actors), and LDA in the space of tags(/actors). Principal component analysis (PCA) is usually explained via an eigen-decomposition of the covariance matrix. However, it can also be performed via singular value decomposition (SVD) of the data matrix X.

For the SVD task, our input is the Object-Feature Matrix obtained by taking TF-IDF values of the movies for a given genre. Therefore, for a given genre/actor we will extract the movies as Objects and TF-IDF tag vectors as features.



Using this matrix in the function $SVD(Matrix)$, for a given genre, we will identify and report the top-4 latent semantics. Similarly for the PCA part, we will use a Feature-Feature Correlation Square Matrix as input to the PCA function as shown in the figure.



Here, the Feature Feature matrix or S is an nxn symmetric, square matrix with real values. Then S can always be decomposed into:

$$S = PCP^1$$

The eigenvectors of the covariance matrix S define bases such that the pairwise correlations have been eliminated. Moreover, the eigenvectors with the largest eigenvalues also have the greatest discriminatory power and thus are more important for indexing.

LDA is a generative Bayesian probabilistic model for collections of discrete data. It allows the observed objects in a dataset to be represented by a random mixture of hidden topics that are unobserved [3]. Each object in the dataset is assigned a set of topics by LDA and that object can be viewed as a mixture of various topics assigned to all the objects in the entire dataset. The topic variable is obtained by dirichlet distribution. Problem Specification (Task 1a): Given a genre, identify the top-4 latent semantics/topics using LDA in the space of tags Here, we assume that a movie is not considered in the corpus, if a tag is not assigned to that movie (i.e., if the user has not watched a movie, that movie has no tags and hence it is not considered in the corpus for LDA). We implement LDA on the dataset by using an existing package, gensim in Python.

The input to the program is a genre. When the user inputs a genre (example War), all the movies which belong to that genre War are extracted from the file *mlmovies.csv*. Once we have the movies related to War, the tags of those movies are extracted from *mltags.csv*. This information is represented in a matrix X. As known, LDA is a probabilistic model. So, the input to the LDA function must be a count of the number of times a tag is assigned to a movie. X is converted into a dictionary and every tag is represented by a unique token id. The matrix is now a corpus represented by only its features (here, tags) and their counts (occurrences of every tag). To this corpus, LDA is performed and we get the top-4 latent semantics/topics. The topics are represented by all the tags that form the corpus. We get the importance (weightage) of a tag in that topic.

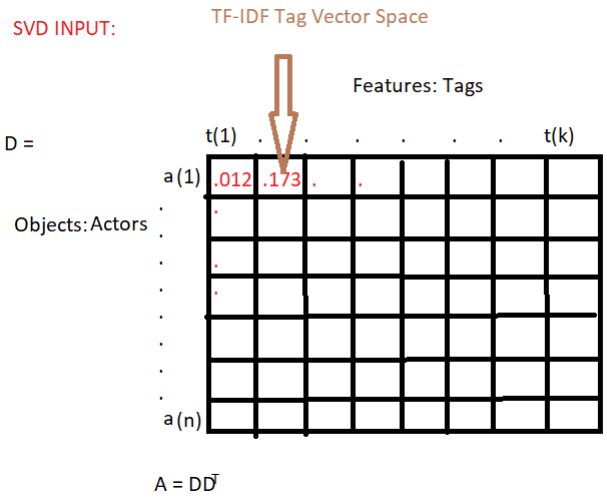
Next, we have to implement a program which, given an actor, finds and ranks the 10 most similar actors by comparing actors' TF-IDF tag vectors and top-5 latent semantics (PCA, SVD, or LDA) in the space of tags. Also, implement a program which, given a movie, finds and ranks the 10 most related actors who have not acted in the movie, leveraging the given movie's TF-IDF tag vectors and top-5 latent semantics (PCA, SVD, or LDA) in the space of tags.

2.2 Second Task

We implement a program which reports the top 3 latent semantics in the actor space, underlying the actor-actor similarity matrix. First, the actor-actor similarity matrix is computed using dot product similarity of the object-feature matrix and its transpose. Then, SVD is performed on the obtained actor-actor similarity matrix to decompose the similarity matrix. We retrieve the top 3 latent semantics from the obtained result of the SVD decomposition.

For the SVD part, our input is the actor-actor similarity matrix obtained by taking the dot product similarity of the actor-feature matrix and its transpose. Therefore, for a given movie, we will extract its actors as objects and its TF-IDF tags as features.

Using this matrix in the Calcsvd(Matrix) function in the svd implementation, for a given actor, we will obtain the top 3 latent semantics.



Here, the Object-Object matrix or D is an n n symmetric, square matrix with real values. Then D can always be decomposed into

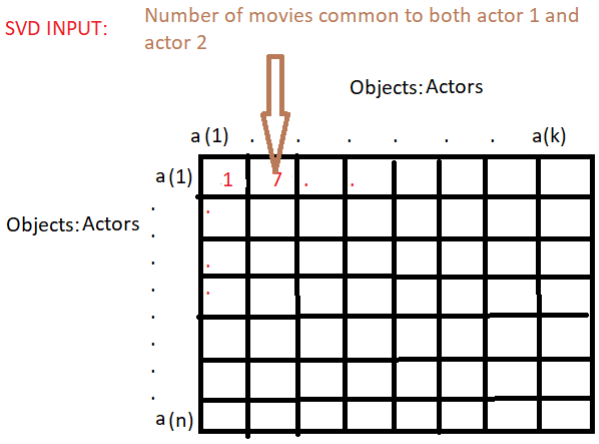
$$D = USV^T$$

From the V^T, we retrieve the top 3 rows of the matrix to get the top 3 latent features. This describes, in terms of features, how two actors are related to each other.

After getting the top 3 latent semantics, we create 3 non-overlapping groups to partition the actors based on their degrees of memberships to the 3 semantics. By using K-means clustering algorithm, we can group the actors into their corresponding latent semantic groups.

Next, we implement a program which reports the top 3 latent semantics in the actor space, underlying the coactor-coactor similarity matrix. First, the coactor-coactor similarity matrix is computed by finding out the number of times two actors played in the same movie. These values are used to fill the coactor-coactor matrix Then, SVD is performed on the obtained coactor-coactor matrix to decompose it. We retrieve the top 3 latent semantics from the obtained result of the SVD decomposition.

Using the coactor-coactor matrix in the Calcsvd(Matrix) function, for a given actor, we will obtain the top 3 latent semantics.



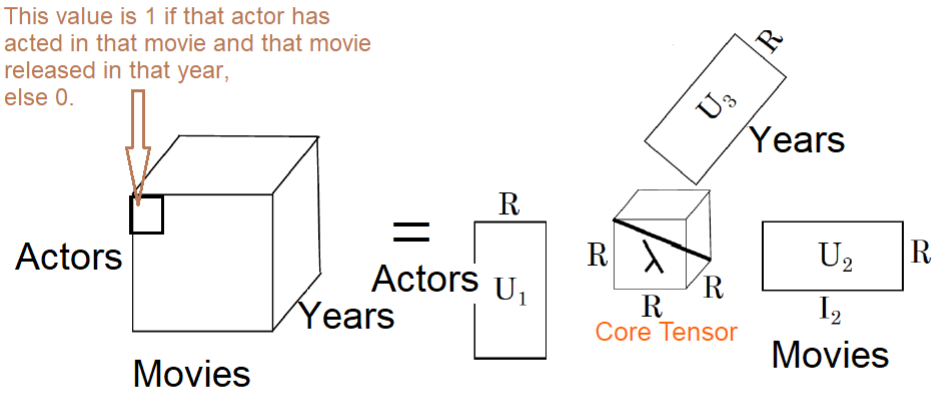
Here, the Object-Object matrix or D is an n n symmetric, square matrix with real values. Then D can always be decomposed into

$$D = USV^T$$

From the V^T, we retrieve the top 3 rows of the matrix to get the top 3 latent features.

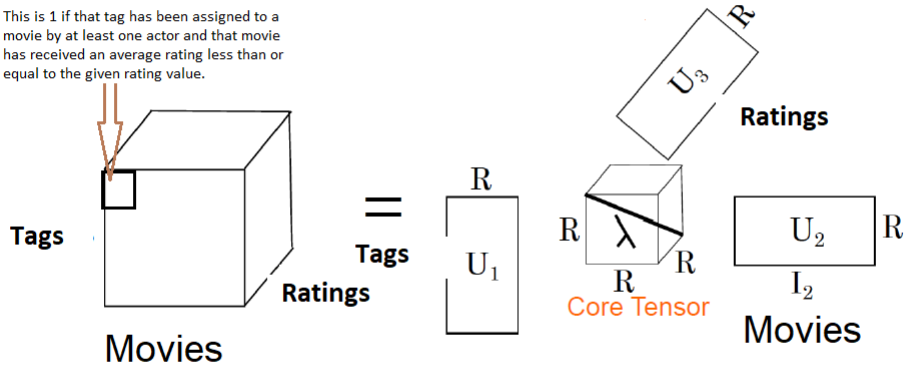
After getting the top 3 latent semantics, we create 3 non-overlapping groups to partition the actors based on their degrees of memberships to the 3 semantics. By using K-means clustering algorithm, we can group the actors into their corresponding latent semantic groups.

Next, we implement a program to report the top 5 latent semantics in terms of actor, movie and year underlying the actor-movie-year tensor. First, an actor-movie-year tensor is created in which the entries of the tensor are 1 if for any actor-movie-year triple if the given actor played in the stated movie and the movie was released in the stated year. Then, CP-decomposition is performed on this tensor with target 5 to obtain the top 5 latent semantics.



After obtaining the top 5 latent semantics, the actors, movies and years are grouped into non-overlapping groups based on their degrees of memberships to the latent semantics.

Next, we implement a program that reports the top 5 latent semantics in terms of tags, movies and ratings underlying the tag-movie-rating tensor. First, a tag-movie-rating tensor is created in which the entries are 1 for any tag-movie-actor triple if that particular tag is assigned to a movie by at least one user and the movie has received an average rating lower than the given rating value. All the other entries for all other triples are 0s.



After obtaining the top 5 latent semantics, the tags, the movies and the ratings are grouped into 5 non-overlapping groups based on their degrees of memberships to these latent semantics

To group the objects into non-overlapping groups for each subtask, we used k-means clustering algorithm. The kmeans() and kmeans2() of SciPy are implemented to group the objects.

2.3 Third Task

In subtask 3a, we implement Random Walk with ReStarts or Personalized PageRank (PPR) to identify the 10 most related actors in a given seed set, S, of seed actors, which are chosen by the users. We make use of an actor-actor similarity matrix that is obtained by applying dot product similarity to the actor-tag matrices. The goal is to find the most similar actors using PPR algorithm, which is why we create a similarity matrix for actors.

The input for the algorithm is the actor-actor similarity matrix obtained by incorporating dot product similarity for the actor-tag matrix and its transpose. This similarity matrix is M. We have alternatively taken P as a 1xN matrix containing 0s and a 1xN matrix containing 1s. S is the given seed set

containing the seed actors. Using the following Personalized PageRank formula, we compute the new P' matrix iteratively, for 100 iterations, until the difference between the P and P' matrix is less than 0.001.

$$P' = \alpha MP + (1 - \alpha)S$$

where, the random surfer follows one of the available links with the probability α, which is assumed to be 0.85 and we take random surfer probability that represents the probability with which the random walk through the graph will deviate from its edges and instead jumps randomly to any node with probability $1 - \alpha = 0.15$.

We retrieve the top 10 most related actors from the final P' matrix obtained after the final iteration.

In the subtask 3b, we implement Random Walk with 'ReStarts' or 'Personalized PageRank (PPR)' to identify the 10 most related actors in a given seed set, S, of seed actors, which are chosen by the users. We make use of an cocoactor-actor similarity matrix that is obtained by calculating the number of movies that two actors have in common. The goal is to find the most similar actors using PPR algorithm, which is why we create a similarity matrix for actors.

The input for the algorithm is the coactor-coactor similarity matrix by computing the number of movies in common for every pair of actors. The entries of the matrix are the number of movies for each pair of actors, if there are common movies, otherwise 0. This similarity matrix is M. We have alternatively taken P as a 1xN matrix containing 0s and a 1xN matrix containing 1s. S is the given seed set containing the seed actors. Using the following Personalized PageRank formula, we compute the new P' matrix iteratively for 100 iterations, until the difference between the P and P' matrix is less than 0.001.

$$P' = \alpha MP + (1 - \alpha)S$$

where, the random surfer follows one of the available links with the probability , which is assumed to be 0.85 and we take random surfer probability that represents the probability with which the random walk through the graph will deviate from its edges and instead jumps randomly to any node with probability 1- = 0.15.

We retrieve the top 10 most related actors from the final P' matrix obtained after the final iteration.

2.4 Fourth Task

In the fourth task, we are asked to implement a program which given all the information available about the set of movies a given user has watched, recommends the user five more movies to watch. We will use the Random Walk Algorithm (using PageRank) method for this task. The random walk algorithm uses the structure of the graph to return movies that matches the query and is also recommended by other users. A random walk over a network places a user at a random node in the network. The user then follows one of the outgoing links from this node at random bringing the user to a new node in the network. The user then repeats this process n number of times. Following the nodes that the user passed through gives a random path through the network. The random walk algorithm allows us to rank the importance of nodes in a network based on the structure of the network. As we see in PageRank, the most important nodes are those that have many in-links from other important nodes. With more important in-links, a node becomes more likely to be reached by a random walker, and therefore, is ranked as more important. In order to apply the random walk algorithm to rank movies in order of importance (for a given user), we create a network that best represents the data and make slight modifications to the algorithm which encode the important aspects of the problem that we are trying to solve.

We define a function $MovieRecommendation$ which takes the input as user id. We define seed vectors as the list of movies that user has watched. The random walk algorithm is implemented as Personalized PageRank where we take the tf-idf of movies in the tag-space:

It is defined by the formula:

$$P' = \alpha MP + (1 - \alpha)S$$

where, it is important to note that we take the value of α as 0.15 to give **more weight** to the seed vectors.

3 Interface specifications (with focus on the goal and problem specification)

We used the following libraries and functions for the implementation of our program:

1) **K-means Clustering** We used the kmeans and kmeans2 functions

of scipy in our tasks to cluster the objects into groups according to latent semantics. K-Means is one of the most popular "clustering" algorithms. K-means stores kk centroids that it uses to define clusters. A point is considered to be in a particular cluster if it is closer to that cluster's centroid than any other centroid. [4]

2) **SciPy** We used the scipy library in several tasks for the usage of kmeans, Euclidean distance, cosine distance ($fromscipy.spatialimportdistance$) [5]

3) **NumPy** [6] Functions from the NumPy library have been incorporated in the tasks involving SVD and similarity matrix

4) **Scikit-Learn** It is a simple and efficient tools for data mining and data analysis which is accessible to everybody, and reusable in various contexts. It is built on NumPy, SciPy, and matplotlib Open source, commercially usable - BSD license. We can use functions like PCA from this library. [7]

We use the following Software Distribution:
Python 3.3+ (on Anaconda Python Distribution)

Further, input and output for each tasks have been mentioned in the output file attached with the Code.

4 System requirements/installation and execution instructions

The project is implemented in Python (Anaconda Distribution) where we have made use of various libraries such as sciPy, csv, sqlite3 (for querying data from the csv files), math, numPy and many more. We need to install the Anaconda python distribution package on Windows 10 (alternatively, we can simply install the given libraries in Linux Terminal).

The first step is to create a Connection object that represents the database using the sqlite3 library. We then create a file where the data will be stored; in our case it is stored in *mwdproj.db*. Now, we create a cursor object, which allows us to interact with the database and add records. Here we use *SQL* syntax to create tables with text different fields.

We then run the python file $task1.py, task2.py$, etc. which contains implementation of all the given Tasks: For a user selected task, we can give the input (genre, actor, movie, etc.)and the given model (pca, svd, pagerank, etc.) to be used on that input.

5 Related Work

Recent years have witnessed intense development of randomized methods for low-rank approximation. These methods target principal component analysis (PCA) and the calculation of truncated singular value decompositions (SVD). The paper by A. Szlam et al. [8] presents an essentially black-box, fool-proof implementation for Mathworks' MATLAB, a popular software platform for numerical computation. Paper by Michael E. Wall et al. [9] describes SVD methods for visualization of gene expression data, representation of the data using a smaller number of variables, and detection of patterns in noisy gene expression data. In addition, it describes the precise relation between SVD analysis and Principal Component Analysis (PCA) when PCA is calculated using the covariance matrix, enabling the descriptions to apply equally well to either method. Ranking methods like PageRank assess the importance of Web pages based on the current state of the rapidly evolving Web graph. The dynamics of the resulting importance scores, however, have not been considered yet, although they provide the key to an understanding of the Zeitgeist on the Web. The paper by Klaus Berberich et al. [10] proposes the BuzzRank method that quantifies trends in time series of importance scores and is based on a relevant growth model of importance scores. We experimentally demonstrate the usefulness of BuzzRank on a bibliographic dataset.

6 Conclusion

In this report, we have taken on the audacious task of implementing various techniques for reducing the number of dimensions in the raw data. We have successfully implemented PCA, SVD and LDA for the tasks of the retrieving information given a certain Object such as genre, actor, movie, etc from various Object-Object, Object-Feature and Feature-Feature matrices. We also implement CP Tensor decomposition to calculate top latent semantics. Using PageRank, we are able to order search results so that more important and central movies are given preference. In experiments, this turns out to provide higher quality search results to users. The intuition behind PageRank is that it

uses information which is external to the movies themselves. Finally, PageRank may be a good way to help find representative pages to display for a cluster center. We have found a number of applications for PCA, SVD, LDA, CP Decomposition and PageRank, in addition to search. Also, we can generate personalized PageRanks which can create a view from a particular perspective. Overall, our experiments with reducing the number of dimensions in the raw data and searching through the structure of the graph is very useful for a variety of information retrieval tasks.

References

[1] K. Selçuk Candan and Maria Luisa Sapino. *Data Management for Multimedia Retrieval*. Cambridge University Press, 2010.

[2] Buckley C. Salton G. Term weighting approaches in automatic text retrieval. information processing and management, 1988, 24(5), 513-523.

[3] Ng A. Blei, D. and M. Jordan. "latent dirichlet allocation." journal of machine learning research, 2003, pp 993-1022.

[4] Scipy Community. K-means clustering and vector quantization. https://docs.scipy.org/doc/scipy/reference/cluster.vq.html.

[5] The Scipy community Copyright 2008-2009. Scipy. https://docs.scipy.org/doc/scipy-0.7.x/reference/.

[6] SciPy. Numpy. https://docs.scipy.org/doc/numpydev/user/quickstart.html.

[7] Mathieu Blondel Et al. Scikit-learn: Machine learning in python. http://scikit-learn.org/stable/.

[8] Mark Tygert Arthur Szlam, Yuval Kluger. An implementation of a randomized algorithm for principal component analysis, 2014. acm toms, 43(3): 28:1-28:14.

[9] Luis M. Rocha. Michael E. Wall, Andreas Rechtsteiner. Singular value decomposition and principal component analysis". in *A Practical Approach to Microarray Data Analysis*. d.p. berrar, w. dubitzky, m. granzow, eds. pp. 91-109, kluwer: Norwell, ma (2003). lanl la-ur-02-4001.

[10] M. Vazirgiannis K. Berberich, S. Bedathur and G. Weikum. Buzzrank . . . and the trend is your friend. in www06: Proceedings of the 15th international conference on world wide web, pages 937–938, new york, ny, usa, 2006. acm.