VECTOR-SPACE MAPPING OF MULTIMEDIA AND WEB DATABASE FEATURES

Suprada Ganesh Nadiger, Hari Kripa Omalur Chandran, Tinu Tomson, Jasmin George, Bosco Paul, and Akshay Muraleedharan Nair Santhy

ABSTRACT

This project describes a methodology to identify the latent semantics in the data using the dimensionality reduction techniques such as SVD, PCA and LDA. A methodology to find the similar actor/co-actors has also been defined. The dimensionality reduction techniques have also been applied on the tensor to find the hidden groupings and the latent semantics.
This project also describes a methodology to recommend movies for the users based on the movies that they have watched previously.

KEYWORDS: Multimedia, PCA, SVD, LDA TF, IDF, TF-IDF, CP, latent semantics, movie recommendation.

INTRODUCTION

Terminology

$TF_{(t,d)}$ - Term Frequency, which measures how frequently a term t occurs in a document d
$IDF_{(t,D)}$ - Inverse Document Frequency, which measures the importance of a term t within the set of documents D. The IDF will decrease the weight of terms which are common to all many documents within the set D.
$TF - IDF_{(t,d,D)}$ - Combined TF and IDF measure for a term t in a document d within the set of documents D.
PageRank - An algorithm used by google search engines to rank the importance of web pages [1].
Personalized PageRank - A modified version of PageRank to factor in a seeded list of initial pages [2].
Cosine similarity - Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them [4].
Tensor - Tensor is an N dimensional array.
CP - CP is a tensor decomposition technique used for mapping tensors from a high to lower dimensional space.
PCA - Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.
SVD - Singular-value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigen-decomposition of a positive semidefinite normal matrix (for

example, a symmetric matrix with positive eigenvalues) to any *mxn* matrix via an extension of the polar decomposition.

LDA- a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics.

Clusters - Group of similar objects

K Means- it is a method of vector quantization, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster

Goal Description

The goal of this project is to:

- Reduce the dimensionality of the sample data from MovieLens and IMDB by identify the latent semantics/topics using SVD, PCA and LDA techniques.
- Apply dimensionality reduction techniques on *actor-actor* similarity matrix, *co-actor - co-actor* matrix, actor-movie-year tensor, and tag-movie-rating tensor to identify the underlying latent semantics.
- Perform a random walk on *actor-actor* similarity matrix and *co-actor – co-actor* matrix to identify the 10 most related actor/co-actor.
- Implement a movie recommendation system which recommends the top 5 movies to a user based on the movies he/she has previously watched.

Assumption

(1) A user is considered to have watched a movie if they have either rated or tagged the movie.

(2) LDA is applied on document term matrix where only the number of times a term occurs is considered. The other weights of that term are not considered.

(3) To partition the actors into non-overlapping groups, it is assumed that the actors in actor similarity matrix are spread over 3 latent semantics, and to group them, we are using K Means clustering method.

(4) To partition the co-actors into non-overlapping groups, it is assumed that the co-actors in co-actor similarity matrix are spread over 3 latent semantics, and to group them, we are using K Means clustering method.

(5) PageRank is initialized to value 1. Personalized PageRank iteration is done for 300 times. The probability of random walker jumping back to seed actors is 0.15. Therefore, probability of transition is 0.85.

FORMULAS

- The Term Frequency (TF) is defined as

$$TF_{(t,d)} = \frac{Number\ of\ times\ a\ term\ t\ appears\ in\ a\ document\ d}{Total\ number\ of\ terms\ in\ the\ document}$$

- The Inverse Document Frequency (IDF) is defined as

$$IDF_{(t,D)} = \log\left(\frac{Total\ number\ of\ documents\ in\ the\ corpus}{Number\ of\ documents\ with\ term\ t\ in\ it}\right)$$

The IDF can further be used in to eliminate the features which are very common among all the objects.

$$TF - IDF_{(t,d,D)} = TF_{t,d} * IDF_{t,d}$$

- The cosine similarity is defined as

$$Similarity = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- The PageRank of a seeded actor *j* is defined as

$$= \frac{0.15}{number\ of\ seeded\ actor} + 0.85 * \sum_i PageRank(actor\ i) * TransitionMatrix(i,j)$$

- The PageRank of the other actor *j* is defined as

$$= \sum_i PageRank(actor\ i) * TransitionMatrix(i,j)$$

PROPOSED SOLUTION AND IMPLEMENTATION

Description of the Proposed Solution

(1) Task 1
This task implements a program that identifies the latent semantics/topics in the data using PCA, SVD and LDA decomposition. This is achieved in four subtasks.

- Task 1a - Is to identify and report the top-4 latent semantics/topics in space of tags, given a genre.
  (a) Filter all movies of a given genre.
  (b) Represent each movie in the space of tags using weighted TF/IDF values.
  (c) Apply both SVD and PCA algorithms on the previously defined matrix to identify top 4 latent semantics.
  (d) Represent each movie in space of tags using number of occurrences as weights and apply LDA algorithm on this to identify top 4 latent semantics.

- Task 1b - Is to identify and report the top-4 latent semantics/topics in space of actors, given a genre.
  (a) Filter all movies of a given genre.
  (b) Represent each movie in the space of actor using weighted TF/IDF values.
  (c) Apply both SVD and PCA algorithms on the previously defined matrix to identify top 4 latent semantics.
  (d) Represent each movie in space of actors using number of occurrences as weights and apply LDA algorithm on this to identify top 4 latent semantics.

- Task 1c – Is to identify and rank 10 most similar actors, given an actor and to identify latent semantics in space of tags.
  (a) Represent each actor in the TF/IDF space of tags.
  (b) Consider a vector of given actor and compare the vector with all other actor vectors to identify the 10 most similar actors using cosine similarity measure.
  (c) Apply PCA, SVD algorithms on the matrix defined in step(a) to identify top-5 latent semantics. Consider a vector of given actor from the resultant matrix and compare with all other actors using cosine similarity to identify the 10 most similar actors in the space of newly identified latent semantics.
  (d) Represent each actor in the space of tags using number of occurrences as weights. Apply LDA algorithm on this to identify the top-5 latent semantics.
  (e) Consider a vector of given actor in the new space (from the previous step) and compare with all other actors in the database using cosine similarity to identify the 10 most similar actors.

- Task 1d – Is to identify and rank the 10 most related actors who have not acted in the given movie
  (a) Represent all movies in the TF/IDF space of tags.
  (b) Represent all actors in the TF/IDF space of tags.
  (c) Consider a vector of given movie from the movie-tag matrix and compare with each row of actor-tag matrix using cosine similarity.

(d) Identify the top-10 actors who have not acted in the given movie from the resultant matrix from previous step.
(e) Apply PCA and SVD algorithms on a movie-tag matrix from step(a) and consider the vector of a given movie from the resultant matrix.
(f) Compare this vector with each row of actor-tag matrix from step(b) using cosine similarity.
(g) Identify the top-10 actors who have not acted in the given movie from the resultant matrix from previous step.
(h) Represent all movies in the space of tags and all actors in the space of tags using number of occurrences as weight.
(i) Apply LDA algorithm on a movie-tag matrix from previous step and consider the vector of a given movie from the resultant matrix.
(j) Compare this vector with each row of actor-tag matrix from step(h) using cosine similarity.
(k) Identify the top-10 actors who have not acted in the given movie from the resultant matrix from previous step.

(2) Task 2

- Task 2a – Is to create an actor-actor similarity matrix and perform SVD on that to identify top-3 latent semantics and to partition the actors into 3 non-overlapping groups based on their degrees of membership to these 3 semantics.
  (a) Represent each actor in the TF/IDF space of tags.
  (b) Create an actor-actor similarity matrix using cosine similarity between two actors.
  (c) Apply SVD on similarity matrix to identify 3 latent semantics sorted in the decreasing order of the weights. Library function sklearn.TruncatedSVD is used for this decomposition.
  (d) the decomposed actor-actor similarity matrix is fed into library function sklearn.KMeans[7], which will cluster the input data into new 3 groups.
  (e) Each label represents different cluster, map the cluster names back to the objects to display.

- Task 2b - Is to create a co-actor – co-actor similarity matrix and perform SVD on that to identify top-3 latent semantics and to partition the actors into 3 non-overlapping groups based on their degrees of membership to these 3 semantics.
  (a) Create co-actor – co-actor matrix such that (i,j)th column represents number of movies actor i and actor j have acted together.
  (b) Normalize each row of the matrix by its sum.
  (c) The values of row i represents the probabilities with which a random walker jumps to another actor from actor i.(normalized sum adds to 1)[3].
  (d) Apply SVD algorithm to the co-actor – co-actor matrix to identify the top-3 latent semantics.
  (e) Decomposed co-actor – co-actor matrix is normalized and fed into library function sklearn.KMeans[7], which will cluster the input data in the 3 new groups.

(f) Each label represents different cluster, map the cluster names back to the objects to display.

- Task 2c – Is to create an actor-movie-year tensor and perform cp-decomposition to identify the latent semantics and to partition actors, movies, and years into 5 non-overlapping group based on their degree of membership to the semantics.
  (a) Create actor movie year tensor using the following constraint
    - Set the value to one if, for any actor-movie-year triple the given actor played in the stated movie and the movie was released in the stated year
    - Set the value to zero otherwise.

  (b) Use "parafac" method of tensorly package to perform CP decomposition on the tensor. The decomposition will produce three factor matrices for actors, movies,and year in terms of five latent semantics.
  (c) Using the latent semantic values of each actor, movies and years, they are partitioned into five non-overlapping group. All the points in the reduced vector space that have maximum contribution to a latent semantic are grouped together (using the clustering technique in [5]) and hence forming five non-overlapping group.

- Task 2d – Is to create a tag-movie-rating tensor and perform cp-decomposition to identify the latent semantics and to partition actors, movies, and years into 5 non-overlapping group based on their degree of membership to the semantics.
  (a) Create tag-movie-rating tensor using the following constraint
    - Set the value to one if, for any tag-movie-rating triple the given tag was assigned to a movie by at least one user and the movie has received an average rating lower than or equal to the given rating value
    - Set the value to zero otherwise.

  (b) Use "parafac" method of tensorly package to perform CP decomposition on the tensor. The decomposition will produce three factor matrices for tags, movies, and ratings in terms of five latent semantics.
  (c) Using the latent semantic values of each tags, movies, and ratings, they are partitioned into five non-overlapping group. All the points in the reduced vector space that have maximum contribution to a latent semantic are grouped together (using the clustering technique in [5]) and hence forming five non-overlapping group.

(3) Task 3

- Task 3a – Is to create actor-actor similarity matrix and to identify the 10 most related actors by performing a random walk with ReStarts.
  (f) Represent each actor in the TF/IDF space of tags.
  (g) Create an actor-actor similarity matrix using cosine similarity between two actors.
  (h) Normalize each row of the matrix by its sum.
  (i) The values of row i represents the probabilities with which a random walker jumps to another actor from actor i.(normalized sum adds to 1)[3].

(j) This normalized matrix can be used as the transition matrix. TransitionMatrix(i,j) represents the probability with which random walker jumps from actor i to actor j. Pagerank of all actors is initialized to 1.
(k) Pagerank is then iteratively calculated using the formulas mentioned in formula section. The actors (except seed actors) with highest pagerank is the displayed.

▪ Task 3b – Is to create a co-actor – co-actor matrix based on the number of movies two actors acted in together and to identify 10 most related actors using RWR.
  (g) Create co-actor – co-actor matrix such that (i,j)th column represents number of movies actor i and actor j have acted together.
  (h) Normalize each row of the matrix by its sum.
  (i) The values of row i represents the probabilities with which a random walker jumps to another actor from actor i.(normalized sum adds to 1)[3].
  (j) This normalized matrix can be used as the transition matrix. TransitionMatrix(i,j) represents the probability with which random walker jumps from actor i to actor j. Pagerank of all actors is initialized to 1.
  (k) Pagerank is then iteratively calculated using the formulas mentioned in formula section. The actors (except seed actors) with highest pagerank is the displayed.

PageRank is calculated with 0.15 restart probability to seed actors and 0.85 transition probability. Number of iterations is 300. Similarity of an actor and co-actor with itself is set to zero.

(4) Task 4

This task implements a movie recommendation algorithm which recommends 5 movies to a user based on movies previously watched by them.

A user might like a particular movie because of its user rating (U), tags (T), actors (A), or genre (G). So, to recommend a list of movies to the user, one needs to consider all the above 4 aspects of the movie. Here we define a movie in the database in terms of {U, T, A, G} in a matrix as shown below.

$$\begin{array}{c|ccc|ccc|ccc|ccc|}
 & \mathbf{U_1} & \dots & \mathbf{U_i} & \mathbf{T_1} & \dots & \mathbf{T_j} & \mathbf{A_1} & \dots & \mathbf{A_k} & \mathbf{G_1} & \dots & \mathbf{G_l} \\
\mathbf{M_1} & * & * & * & * & * & * & * & * & * & * & * & * \\
\mathbf{M_2} & * & * & * & * & * & * & * & * & * & * & * & * \\
\dots & * & * & * & * & * & * & * & * & * & * & * & * \\
\mathbf{M_n} & * & * & * & * & * & * & * & * & * & * & * & *
\end{array}$$

A movie is defined in the user space by using ratings as weights. It is defined in the tag space by using weighted TF/IDF as weights. It is defined in the actor space using the actor rank as weights. Finally, it is defined in the genre space depending on whether the movie belongs to a specific genre(s) or not. Each aspect {U, T, A, G} in the matrix is normalized in the following way in order to give equal weight to all aspects considered.

▪ User rating is normalized by dividing all user ratings by the maximum possible rating which is 5.

- TF/IDF tag values are normalized by dividing all TF/IDF values by maximum TF/IDF value for a corresponding movie.
- Actor ranks are normalized by dividing each actor rank by the maximum rank of all actors who have acted in that movie.
- Genre values are not normalized since they are already represented in binary values.

To find movies to recommend, a similarity matrix is created which defines the similarity between movies watched by the target user (in rows) and all movies in the database (in columns). This matrix is computed using the cosine similarity between two movie vectors as shown below.

To find the top 5 movies that are similar to the movies the user has watched, the column values in the similarity matrix above are summed for each movie in the database. The 5 movies with the highest sum are recommended to the user.

Interface Specifications

The algorithms described above were implemented using Python 3.5 and 2.7. The scripts and functions were tested on Linux and Mac OS.

System Requirements

(1) Linux and Mac OS
(2) Python 3.5 and 2.7 with pandas, numpy, scipy, sklearn, tensorly, and genism packages installed.

Execution Instructions

Task1

Run the corresponding scripts to execute the corresponding tasks.
Task 1a:
Run the following scripts inside ‘task1a’ directory

```
python pca.py genreName
python svd.py genreName
python lda.py genreName
```

Task1b:
Run the following scripts inside ‘task1b’ directory

```
python pca.py genreName
python svd.py genreName
```

python lda.py genreName

Task 1c:
Run the following scripts inside ‘task1c’ directory

python similarity.py actorId
python pca.py actorId
python svd.py actorId
python lda.py actorId

Task 1d:
Run the following scripts inside ‘task1d’ directory

python related_actor.py movieId
python pca.py movieId
python svd.py movieId

<u>Task2</u>
Run the following scripts inside ‘task2a_b’ directory
Task 2a:
python aa_svd.py

Task 2b:
python cc_svd.py

Task 2c:
Inside task2c folder run the following commands. This task will only run on 2.7 version of python.

python actor_movie_year_tensor.py (this is for creating tensor)
python cp_decomposition.py (this is for performing cp decomposition, displaying the latent semantics and partitioning actors, movies and years)

Task 2d:
Inside task2d folder run the following commands

python tag_movie_rating.py (this is for creating tensor)
python cp_decomposition.py (this is for performing cp decomposition, displaying the latent semantics and partitioning tag, movies and ratings)

<u>Task3</u>
Inside task2d folder run the following commands

Run scripts to create coactor and similarity matrices

```
python actor_sim_matrix.py
python coactor_matrix.py
```

Run following sample code to get recommendations related to a seeded list of actors

```
python actor_recommend.py '1014988,1342347,1698048'
python coactor_recommend.py '1014988,1342347,1698048'
```

Task4

Inside task4 folder run the following commands

```
python recommend_movies.py USER_ID
```

SUMMARY/CONCLUSIONS

Task 1

Latent semantic analysis is very useful tool for dimensionality reduction. It helps to identify concepts that can represent the data with fewer dimensions. This is especially true when there is enough data to establish the correlations between the dimensions. PCA will not work very well if such a correlation cannot be established especially when data is sparse.

Task 2

Task 2a and 2b

Clustering is a technique to combine objects in a given space according to how close they are in the space. In the given data set, an actor - actor similarity matrix and co-actor – co-actor relationship matrix is created for all actors. A Cosine similarity measure is used for calculating the similarity between objects. These matrices are object-object matrices, hence SVD was applied to reduce the dimensions, and applied K Means algorithm to it to identify 3 clustering in the actor data in the latent dimensions created by the SVD.

Task 2c and 2d

Tensor decomposition is a useful technique in understanding the latent semantics of a data set and to cluster the data into related groups based on these semantics. It identifies the relation among the data points and groups them into specified number of clusters.

Task3

Applying Personalized pagerank on similarity matrix of actors gave similar actors with respect to TF-IDF values of their tags. Similarly, Personalized pagerank on co-actor matrix gives relevant actors with respect to number of movies acted together. It is helpful in understanding that pagerank can also be applied on weighted edge graphs, taking into consideration of the weights of the edges. The issue faced was due to very low number of tags, the data was very sparse and hard to verify the results.

<u>Task4</u>

This task implemented a strategy to recommend movies to the user based on the movies they have previously watched. All the factors of the movie such as user ratings, movie tags, actors and movie genre were taken into consideration to recommend new movies for the user to watch. The methodology is implemented using the provided data from IMDB and MovieLens web databases.

BIBLIOGRAPHY

[1] https://en.wikipedia.org/wiki/PageRank
[2] https://cs.stanford.edu/people/plofgren/bidirectional_ppr_thesis.pdf
[3] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery.In KDD, pages 653658, 2004.
[4] https://en.wikipedia.org/wiki/Cosine_similarity
[5] Tao Wu, Austin R. Benson, David F. Gleich. General Tensor Spectral Co-clustering for Higher-Order Data
[6] https://tensorly.github.io/stable/modules/generated/tensorly.decomposition.parafac.html
[7] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[8] http://scikit-learn.org/stable/auto_examples/text/document_clustering.html#sphx-glr-auto-examples-text-document-clustering-py
[9] https://radimrehurek.com/gensim/wiki.html
[10] Jean Kossaifi, Yannis Panagakis and Maja Pantic, TensorLy: Tensor Learning in Python, https://arxiv.org/abs/1610.09555.
[11] Bogers, Toine. (2010). Movie recommendation using random walks over the contextual graph.
[12] https://en.wikipedia.org/wiki/Principal_component_analysis.
[13] https://en.wikipedia.org/wiki/Singular-value_decomposition

APPENDIX A
SPECIFIC ROLES OF GROUP MEMBERS

The tasks were distributed among the group. The implementation details were discussed in the group but the actual implementation was carried out by individual group members as follows.

Task 1a, 1b (PCA) - Tinu Tomson, Jasmin George

Task 1a, 1b (SVD) - Tinu Tomson, Bosco Paul

Task 1a, 1b (LDA) - Akshay Muraleedharan Nair Santhy, Hari Kripa Omalur Chandran

Task 1c - Tinu Tomson

Task 1d - Tinu Tomson

Task 2a, 2b - Akshay Muraleedharan Nair Santhy

Task 2c, 2d - Jasmin George

Task 3 - Bosco Paul

Task 4 - Suprada Ganesh Nadiger, Hari Kripa Omalur Chandran

Report - Suprada Ganesh Nadiger, Tinu Tomson, Jasmin George, Akshay Muraleedharan Nair Santhy, Hari Kripa Omalur Chandran, Bosco Paul.