Text and Image features, Vector models and Dimensionality Reduction

Akshay Shah
Kshitij Kashettiwar
Nikhil Agarwal
Ronak Tanna
Shivam Dhar
Shreyash Devan
Shubham Verma

Abstract

One of the classic problems in the web domain is to yield the best results as part of a user query that can be in form of multimedia. There are various techniques one can employ for Information Retrieval (IR) from huge sets of multimedia consisting of text, images, audio, video, etc. In this phase, we focus on implementing a search engine by applying the concepts of vector and tensor models, state of the art techniques of dimensionality reduction algorithms - Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Latent Dirichlet Allocation (LDA), and similarity/distance measures for textual and visual descriptors of multimedia on a publicly available dataset comprising of various models to represent text and images.
The search engines may yield varying results based on their model implementation. We strongly focus on the use of proven similarity metrics like Cosine similarity and Euclidean distance that work well with text and images respectively, to yield best results in the given time frame.

Keywords

Information retrieval, textual descriptor, visual descriptor, search engine, similarity, vector space model, tensor, latent semantics, dimensionality reduction, eigen vectors, tensor-decomposition.

INTRODUCTION

In this phase, we experiment with the dataset provided by [1] B. et al, with respect to three entities - location, user and images where each user has captured a set of images at certain locations and has tagged them. It also provides two categories of information - some of the features are extracted for text (image tags, title) like TF, DF, TF-IDF scores forming a set of textual descriptors and various models for images like CN, CM, HOG, etc which are extracted from the images and form a set of visual descriptors. These descriptors help in creating a feature vector for each entity in the dataset and enable comparison between them by computing a similarity score using metrics (that work well with text and images) like Cosine similarity and Euclidean distance.

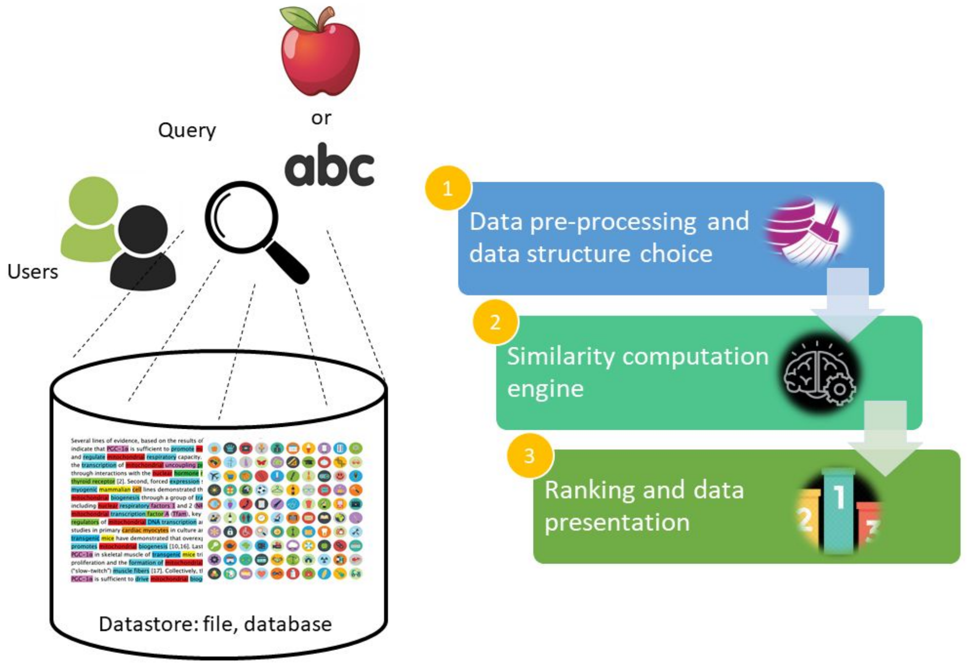

Query
or
abc
Users
Datastore: file, database
1
Data pre-processing and data structure choice
2
Similarity computation engine
3
Ranking and data presentation

Fig 1. shows a brief idea of the project - given a user query, finding out the top matching objects based on the similarity metrics computed.

Terminology

Dataset: The information set consisting of objects and their features.
Object: A particular entity in the dataset. Eg: image, location, user.
Model: A collection of features. Eg: CN (color naming histogram), TF (term frequency).
Feature: A description of object in dataset extracted using Information Retrieval techniques like different colors in the CN, center of moments for CM model.
Descriptor: A collection of models. Eg: textual descriptors - TF, DF.
Similarity: A metric that determines how similar the two entities in the dataset are based on different features associated with them.
Distance: Degree of closeness or separation between the two entities in the dataset. The more the distance, lesser is the similarity.
Vector: A data representation for an object in terms of its features.
Tensor: A data representation used to describe relation between vectors and scalars, i.e objects and their features. Eg: user, location, image - terms in common.
Dimensionality reduction: A technique that is employed for reducing the variates under consideration and obtain a set of principal components (or variables) that are hidden in the dataset.
Latent Concepts: Features or concepts that are hidden in the data, and that are identified via a dimensionality reduction technique.

Goal Description

The overall goal of the phase is to get familiarized with the various techniques of performing faster search in a multidimensional space.

- Task 1 considers user/image/location term vector space and identifies top-k latent semantics for the corresponding term space using the user-selected dimensionality reduction algorithm. Each latent semantic is presented in the form of term-weight pairs in decreasing order of their weight.
- Task 2 is an extension of Task 1 and considers the reduced user-concept space (or) image-concept space (or) location-concept space and given a userID, imageID, or locationID, finds 5 most similar users, images and locations via projections onto the reduced concept space of the term vector space chosen by the user in Task 1.
- Task 3 deals with user queries in the space of images across all locations and a particular visual descriptor model. Given an input image ID and a visual descriptor model, the top 5 image IDs and location IDs have to be found, by applying dimensionality algorithms specified by the user.
- Task 4 aims at deriving top 5 similar location from the dataset : set of files across all locations having the same visual descriptor model, containing features extracted for all images taken at a location, given a location id, model and a dimensionality reduction algorithm choice, solely on the basis of corresponding features derived by the models - CM, CM3x3, CN, CN3x3, CSD, GLRLM, GLRLM 3x3, HOG, LBP, LBP3x3.
- Task 5 aims at deriving top five similar locations from the dataset : set of files across all locations and models, containing features extracted for all images taken at a location, given a location id and a dimensionality reduction algorithm.
- Task 6 aims at discovering hidden/latent patterns in the location data by applying dimensionality reduction via a Singular Value Decomposition on a location-location similarity matrix.
- Task 7 aims at creating a tensor; this tensor is composed out of the count of common terms between users, images and locations. Next we run a CP decomposition upon this tensor for ‘k’ latent semantics and further group these users, images and locations into k independent sets.

Assumptions

- **Preprocessing**
    1. Remove any duplicates for visual descriptor dataset. Eg: If same imageID is present in a model location file twice, we remove one of the instances.
    2. Latest values of textual descriptors taken into account in case any duplicate user/image/location is found.
    3. Add missing objects to dataset. The data structure used requires all images in a location to be present across all models in the visual descriptor dataset. Any instance missing is imputed with mean of the respective feature values in the dataset.
    4. The LDA library used takes non negative input for processing. For this reason, we add the least negative value across the features for all the instances in the visual descriptor dataset. This ensures relative change in all the features and hence is the best way to handle such data.

- **Development**
    1. The very first assumption is that the dataset is a correct representation of facts. All the features extracted across various models will give efficient results for a query search engine. Our similarity engine is based on the assumption that data is noise free and doesn't have outliers.
    2. The input is part of the dataset and the query results include the input instance as well.
    3. Input k to any task given by the user is a positive integer.
    4. The sklearn decomposition library used for dimensionality reduction gives fit transformed results for the three algorithms PCA, SVD and LDA i.e given a value of k latent semantics, it returns the reduced (object X latent semantics) dataset which is further used for similarity computation. This holds for task 3, 4 and 5.
    5. While computing the image-image similarity Euclidean distance is used so as to preserve the distances between the two, assuming that this similarity metric would be a good measure for their closeness - this holds for task 3, 4 and 5.
    6. In case of data matrix with high features in range of 20k, we need to sparsify the matrix before passing to SVD, assuming that original semantic space remain preserved for term-weights pairs. This holds for task 2.

<u>Task 1:</u>

1. The basic assumption is that we have chosen TF-IDF scores as weights for the user-term vector space, the image- term vector space, and location-term vector space.

<u>Task 3:</u>

1. While computing the similar locations given a image ID, all the images of all locations are taken into consideration and similarity is computed. The assumption while choosing the similar location is that the maximum image-image score for this location is chosen.

Task 4:

1. As we compute image-image similarity, we need to find an image in other location which would be most similar to each image in input location. We consider maximum function to extend this behavior. Let’s consider location id 1 which has 300 images. For each image in location 1 we find the most similar image in location 2. Then to define similarity score for the two locations we compute the average of all the pairwise similarities computed. This gives a good estimation of how similar two locations are to each other.

Task 5:

1. The feature set for an image is considered as an amalgam of features across all the models. This results into an image being represented as a function of features across models i.e. image_id is represented as a vector with high dimensions consisting of feature values for all the visual descriptor models CM, CN, HOG, etc.
2. The similarity computation assumes that the top hidden features from the multidimensional dataset approximates the reduced dataset correctly.
3. After applying dimensionality reduction on the dataset, the model context will be lost and we will not be able to evaluate similarity measures considering individual models. Hence, we assume that euclidean distance will be a good measure for similarity computations.
4. As we compute image-image similarity, we need to find an image in other location which would be most similar for each image in input location. We consider maximum function to extend this behavior. Let’s consider location id 1 which has 300 images. For each image in location 1 we find the most similar image in location 2. Then to define similarity score for the two locations we compute the average of all the pairwise similarities computed. This gives a good estimation of how similar two locations are to each other.

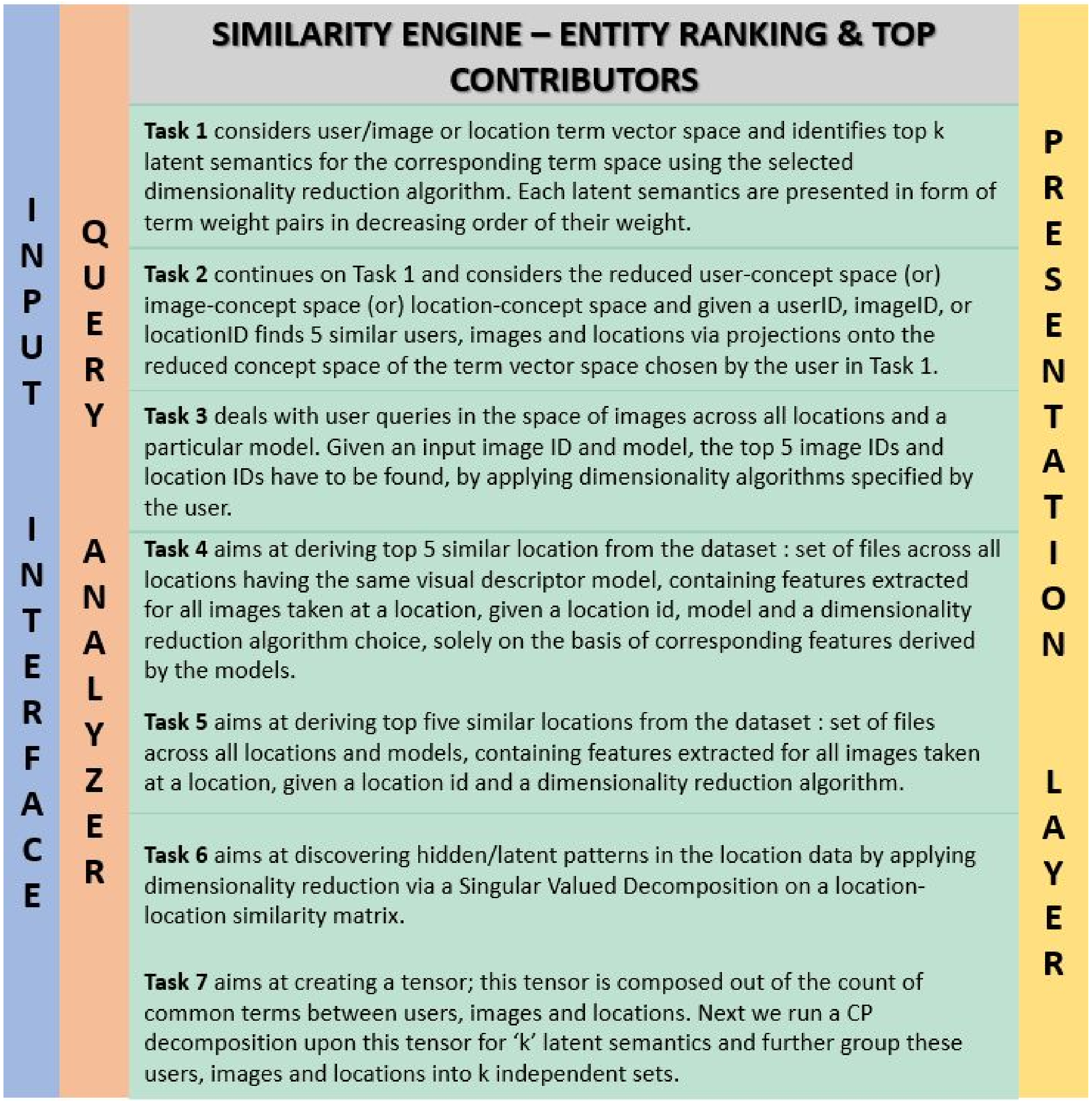
Task 6:

1. The feature set considered for computing the similarity matrix is the textual descriptors of the location data. The assumption here is that the common terms in the locations give a good representative of how similar the locations are.
2. We assume the choice of TF-IDF score to be used as basis for similarity computation would be the best as it would combine the object (location) description power and the object (location) discrimination power of a given term into a single score.

Task 7:

1. While computing a CANDECOMP/PARAFAC (CP) decomposition, we assume that the decomposition obtained is similar to the original data, and not much variance is lost during the process.
2. For grouping of various users, images and locations we use k-means clustering.
    a. **k-means** assumes that variance of the distribution of each attribute (variable) is spherical;
    b. all variables have the same variance
    c. the prior probability for all **k** clusters are the same

DESCRIPTION OF PROPOSED SOLUTION / IMPLEMENTATION



Task 1 considers user/image or location term vector space and identifies top k latent semantics for the corresponding term space using the selected dimensionality reduction algorithm. Each latent semantics are presented in form of term weight pairs in decreasing order of their weight.

Task 2 continues on Task 1 and considers the reduced user-concept space (or) image-concept space (or) location-concept space and given a userID, imageID, or locationID finds 5 similar users, images and locations via projections onto the reduced concept space of the term vector space chosen by the user in Task 1.

Task 3 deals with user queries in the space of images across all locations and a particular model. Given an input image ID and model, the top 5 image IDs and location IDs have to be found, by applying dimensionality algorithms specified by the user.

Task 4 aims at deriving top 5 similar location from the dataset : set of files across all locations having the same visual descriptor model, containing features extracted for all images taken at a location, given a location id, model and a dimensionality reduction algorithm choice, solely on the basis of corresponding features derived by the models.

Task 5 aims at deriving top five similar locations from the dataset : set of files across all locations and models, containing features extracted for all images taken at a location, given a location id and a dimensionality reduction algorithm.

Task 6 aims at discovering hidden/latent patterns in the location data by applying dimensionality reduction via a Singular Valued Decomposition on a location-location similarity matrix.

Task 7 aims at creating a tensor; this tensor is composed out of the count of common terms between users, images and locations. Next we run a CP decomposition upon this tensor for ‘k’ latent semantics and further group these users, images and locations into k independent sets.

Fig 2. describes the various components of the search engine

TASK 1

This task deals with computing the top-k latent semantics in the user-term vector space, image-term vector space and the location-term vector space. Given a choice of vector space and choice of algorithm as defined by user for dimensionality reduction, we have to determine top-k latent semantics. Value of k is provided as user input.

A global user tag list is created and the corresponding score value(weight) is stored in matrix format. Depending on the algorithm chosen by the user: SVD or PCA, we perform eigen decomposition either on Feature-Object matrix or on Co-variance matrix of the feature- object matrix in order to find the k-hidden semantics.
Based on user's choice of the dimensionality reduction algorithm, we determine the top-k latent semantics from the textual descriptors of users, image & location and represent it in the form of feature-weight pairs.

DIMENSIONALITY REDUCTION ON DATASET
We are using the sklearn library for using the truncated SVD, PCA and LDA algorithms.

DATA PROCESSING AND STORAGE
Textual descriptors for all the three vector spaces are processed and stored in the form of a common feature space. For PCA, data has been converted into standard scaler before applying the dimensionality reduction algorithm. Output i.e. feature-term pair values are arranged in decreasing order of their weights and stored in a file.

TASK 2

This task deals with user queries in the user term space, image term space and the location term space. Given an input userID, imageID or locationID, top 5 similar users,images and locations for given input entity id, need to be fetched and corresponding similarity scores need to be shown along with the respective entity.

DATA PROCESSING AND STORAGE
The dataset comprises of 3 files devsetTermsPerUser.txt, devsetTermsPerImage.txt and devsetTermsPerPOI.wFolderNames.txt which represents the textual descriptors of the users,images and locations respectively. A global term vocabulary is used for all the entities thus getting a common feature space.
.
SIMILARITY COMPUTATION
Cosine similarity between the TF-IDF weights of the terms is used for similarity computation. For a given user selection, we project the data points of the other entities apart from input onto reduced feature space obtained with considerable less number of features and use the the input vector to directly compare between other subsequent vectors and compute cosine similarity. Cosine similarity is considered in the sense that we need to find how the terms are related on the basis of TF-IDF score and position in the vector space instead of the magnitude of the score.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

CHOICE OF SCORE IN THE TEXTUAL DESCRIPTORS

The choice of score for consideration during similarity computation is the TF-IDF score of any given term as it combines the object description power given by the term frequency for each entity separately and the object discrimination power given by the document frequency.

DIMENSIONALITY REDUCTION ON DATASET
Dimensionality reduction algorithm is specified by the user, it can be PCA, SVD or LDA. The document term matrix obtained above is converted to a numpy 2D array and then PCA, SVD or LDA is applied. We are using the sklearn library for using the SVD, PCA and LDA algorithms. For each dimensionality reduction algo, the U matrix is obtained by projecting the original data points onto new latent semantics ie oxf multiplied by fxk (feature-concept matrix) so that we can get correct projection of points in terms of strength of hidden semantics.

QUERY PROJECTION OF INPUT VECTOR ONTO REDUCED ENTITY FEATURE SPACE
We have the leverage to project the input query vector onto the reduced feature space of other objects since the original feature space for all i same by using the following formula as explained by [4] Furnas et al.

$$q_k = q^T U_k \Sigma_k^{-1}$$

In the above equation, the following are considerations
q - 1xf - Input vector
U - fxk - feature - concept latent semantics
Σ - kxk - strength of latent semantics
qk -1xk - query vector in the projected latent space

For given userId, we project the input user vector onto the reduced space of images and locations which is obtained by the either of 1 algorithm chosen by user(SVD/PCA/LDA). Similarly for given imageId, we project the input image vector onto the reduced space of users and locations and finally for locationid , we project the input location vector onto the reduced space of users and images.

QUERYING AND RESULT GENERATION

In case of finding similar objects for same entity id, we directly compute cosine similarity between the input vector (in the reduced semantics space ie 1xk) and the vectors of same object type. For eg, userid - direct computes similar 5 users using above method.
However, in case of finding cross similar objects, the query projection vector is computed using the above query projection method and similar cross entities are computed for each non user input entity. For eg, given userid, this will compute 5 most similar images and locations. Similarly for other subsequent user input, the similarity can be computed and corresponding scores can be paired and displayed.

TASK 3

This task deals with user queries in the space of images across all locations and a particular model. Given an input image ID and model, the top 5 image IDs and location IDs have to be found, by applying dimensionality algorithms specified by the user.

DATA PROCESSING AND STORAGE
For each location there are 10 visual models(CM, CM3x3, CN, CN3x3, CSD, GLRLM, GLRLM3x3, HOG, LBP and LBP3x3). When a model is specified by the user, all the location files of this model are opened and the vectors of all images are are appended to make a matrix. Here in the matrix rows represent image ID and the columns represent the features of the images for the model specified. The indices(start index and end index) of all locations are maintained.

DIMENSIONALITY REDUCTION ON DATASET
Dimensionality reduction algorithm is specified by the user, it can be PCA, SVD or LDA. The matrix obtained above is converted to a numpy 2D array and then PCA, SVD or LDA is applied. We are using the sklearn library for using the SVD, PCA and LDA algorithms. This gives us the image ID to k-semantics matrix.

SIMILARITY COMPUTATION
Euclidean distance is used for similarity computation. Image to image similarity is computed for all images across all locations. The image-image score for all locations are maintained.

QUERYING AND RESULT GENERATION
Once the dimensionality reduction is done the task is to find top 5 image IDs and location IDs. Given an input image ID, similarity is computed with all images across all locations. Here the start and end index of all locations are maintained. The similarity score is sorted and the top 5 image IDs are returned. When computing the top 5 locations, scores obtained are sorted separately for each location. The maximum score of each location is taken into consideration and sorted again to give the top 5 location IDs.

TASK 4

This task deals with user queries in the space of images across all locations and a particular model. Given an input location and model, top five similar locations need to be fetched, by applying dimensionality reduction algorithms as specified by the user.

DATA PROCESSING AND STORAGE

This query focuses on the visual descriptors of the images. For each location and image, there are 10 general purpose visual models namely, CM, CM3x3, CN, CN3x3, CSD, GLRLM, GLRLM3x3, HOG, LBP and LBP3x3. Each model file for a particular location consists of a particular number of images and each imageID is followed by the some descriptor feature values for all the specific visual model. Query 4 specifies what visual model and what dimensionality reduction algorithm to use for finding the most similar locations. During runtime, when a model, locationID and dimensionality reduction algorithm is specified by the user, the particular model file for that location is opened from the preprocessed dataset and a list of vectors of all the visual descriptors for each image is maintained. We then append the list for each other location in the database. In this way, the list will contain all the locations' images in it with the given location at the first position each time.

DIMENSIONALITY REDUCTION ON DATASET

Further, the task of dimensionality reduction is then accomplished with the help of 3 choices of algorithms viz, SVD, PCA and LDA. The input to these algorithms is a 2 dimensional numpy array. For this, the final list created in the above data processing step is converted to a numpy array before it is given as input to the chosen algorithm. This matrix now contains all the images as its rows and all the feature values extracted from the particular specified model as its columns (n x m). We are using the sklearn library for using the SVD, PCA and LDA algorithms.

SIMILARITY COMPUTATION

Euclidean distance is used for similarity computation. Image to image similarity is calculated across locations with respect to input location, at last an average score is associated with the two locations considered. The following model drives the similarity score between two locations:

$$\text{Similarity(location1, location2)} = \frac{\sum_{\forall \textit{image in location}} \max(\Delta_{\text{similarity}}(\text{image}_{\text{location1}}, \text{image}_{\text{location2i}}))}{\text{Number of pairwise image similarity computations between two locations of a given model}}$$

Using this metric, top five similar locations are fetched.

QUERYING AND RESULT GENERATION

After the task of dimensionality reduction, we are left with a low dimension (k) data matrix with the same number of rows i.e images. Our task now is finding the top 5 locations similar to the given location. This involves image by image comparison of the given location with all the other locations in the data matrix. When one image from the given location is compared with all the images of location B, we maintain a list which will include the maximum similarity score for a particular image-image similarity. In this way, this list will contain the maximum similarity scores for each image in the given location in comparison with a certain location B. After we get the scores for all the images in location B, we take the average over all the scores to get a final similarity score for that location B. This score is then added to a map which maintains a key, value order of the location name and the final similarity score of that location. This is then done for all the other locations in the dataset. Thus, a map of all the locations and their respective similarity scores with the input location is now created. The final query result involves printing the top 5 locations and their scores from the map which will have the given location at the first position with a similarity score of 1.

TASK 5

This task deals with user queries in the space of images across all locations and models. Given an input location, top five similar locations need to be fetched, by applying dimensionality reduction algorithms as specified by the user.

DATA PROCESSING AND STORAGE

The dataset comprises of a model - location file eg. *angel_of_the_north_CM.csv,* having images with corresponding features. In this step, image features are extracted from all the model - location files and stored in a map. Also, we use a map to store the location indices for the set of images to track image pairs for every location.

DIMENSIONALITY REDUCTION ON DATASET

Based on the user's algorithm choice, switcher executes the corresponding dimensionality reduction algorithm on the dataset and returns instances (image ids) with k latent features. This is further used for finding location similarity with respect to the input location.

SIMILARITY COMPUTATION

Euclidean distance is used for similarity computation. Image to image similarity is calculated across locations with respect to input location, at last an average score is associated with the two locations considered. The following model drives the similarity score between two locations:

$$\text{Similarity}(\text{location1}, \text{location2}) = \frac{\sum_{\forall \textit{image in location}} \max(\Delta_{\text{similarity}}(\text{image}_{\text{location1}}, \text{image}_{\text{location2i}}))}{\text{Number of pairwise image similarity computations}}$$

between two locations

Using this metric, top five similar locations are fetched.

TASK 6

This task deals with computing a location-location similarity matrix and identifying k latent concepts in the location data. Once these are identified, we describe the k latent semantics with location name-weight pairs for each semantic sorted in the decreasing order of weights.

CHOICE OF THE DATASET FOR SIMILARITY MATRIX COMPUTATION

Our choice of dataset was the textual descriptors of the locations which provide the terms of the locations and their associated term frequency, document frequency and TFIDF scores. The dataset was chosen because of the intuition that the intersecting terms in two given locations give a good idea of how similar the locations are as they are taken from textual descriptions of the images in that location.

CHOICE OF SCORE IN THE TEXTUAL DESCRIPTORS

The choice of score for consideration during similarity computation is the TFIDF score of any given term as it combines the object (location) description power given by the term frequency and the object (location) discrimination power given by the document frequency.

CHOICE OF SIMILARITY METRIC BETWEEN LOCATIONS

We choose the similarity metric as the Cosine Similarity score between the TFIDF values of the intersecting terms of two given locations.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

where A is the vector of TFIDF scores of terms in location A and B is the vector of TFIDF scores of terms in location B. Since the terms that are in location A, but not in location B have a TFIDF value of 0 in B's vector and the terms that are in location B but not in location A have a TFIDF value of 0 in A's vector, we are only considering the intersecting terms between the two locations as the dot product of a non-intersecting term would result in a value of 0.

The intuition behind this is that we care more about how aligned the TFIDF scores of the intersecting terms are with respect to their positions in the vector space rather than how they compare in terms of the strength of the score (magnitude). The Cosine Similarity score perfectly captures what we want.

DATA PROCESSING AND STORAGE

The dataset comprises of a single file -- devsetTermsPerLocation.txt which represents the textual descriptors of the locations. A global term vocabulary is constructed apart from location specific term dictionaries. Each term in a location specific term dictionary consists of the term, its associated term frequency score, its document frequency score and its TFIDF score.

SIMILARITY MATRIX COMPUTATION

- Now that we have the location-specific term dictionaries constructed, we consider every location as a potential query location and every other location as a potential target location for the query location.
- We then compute the similarity score between two locations as the Cosine Similarity measure between the TFIDF scores of the intersecting terms.
- Once we repeat this for every location in the dataset, we get a self-similarity matrix that is symmetric and square, has all values ranging between 0 and 1, and has its diagonal elements as 1 showing 100% similarity as the query location and target locations are the same.

LATENT CONCEPT IDENTIFICATION

Once the similarity matrix is computed, we perform a Singular Value Decomposition (SVD) of it, and get 2 factor matrices that represent the same eigenvectors (as the data matrix is square and symmetric). This implies that we can choose the first ‘k’ columns of the U matrix or the first ‘k’ rows of the Vt matrix as the latent semantics in our data. These semantics represent the basis vectors in the new “concept” space.

LATENT SEMANTICS AS LOCATION-WEIGHT PAIRS

The problem statement also required the latent semantics to be represented as location name-weight pairs sorted in the decreasing order of weights. To achieve this:

- We multiply the data matrix (similarity matrix) with the U matrix reduced to consisting of only the first ‘k’ columns.
- If O is the number of objects and F is the number of features in the data matrix & O is the number of objects and ‘k’ is the number of features in the U matrix, the projection gives us an Oxk matrix. This columns of this matrix gives us the latent semantics weighted in terms of the original features in the concept space.
- For ease of implementation, we take the transpose of this matrix -- the rows now represent the latent semantics weighted in terms of the original features in the concept space.
- We sort a row while maintaining the positions of indices during the sorting and make use of a helper to map location IDs to their titles and print out weights to the console as Location Name-Weight pairs for the given latent semantic. We repeat the same procedure for each row of this matrix.

We have now achieved the task of finding ‘k’ latent patterns in the data and have also determined the Location Name-Weight pairs sorted in decreasing order of weights for each latent semantic that was identified.

TASK 7

This task deals with creating a tensor; this tensor is composed out of the count of common terms between users, images and locations. Next we run a CP decomposition upon this tensor for ‘k’ latent semantics and further group these users, images and locations into k independent sets.

DATA PROCESSING AND STORAGE

The dataset we deal here with comprises of textual descriptors only. This data is read from an independent data file for each vertical such as user, images and locations. These files are parsed using Python3 and each row is stored in a class object. These class objects of all users are then further stored in another class containing a dictionary of these objects with their id (userId in case of users) as key and terms and their textual descriptor values(like TF, DF) as sets of objects. These class objects and dictionary is then similarly computed for image and location textual data.

BUILDING A TENSOR

Python3 has been used as our programming tool for this step, with no extra libraries involved. To compute the intersection between terms used by images, users and locations, we use the sets present in our dictionaries prepared in the previous step, and take an intersection of these sets to obtain the common set, given an image, location and user. Given our test data, this computes a tensor of (30, 530, 8912).

Initially, our tensor build operation was taking around 1hr 15 mins; we have brought this computation down to around 5 mins. This was done by looking more closely at the data we had, and with the use of efficient data-structures. A re-ordering of the way we ran for-loops for tensor computation was done. Looking at the data, we noted that locations had the most number of tags, followed by users and then images. Previously, we were running users at our outermost loop and locations at the innermost, this was heavy computation, since we were performing intersection of large sets at our inner-most loop. Hence, we changed this, we now had locations at our outermost loop and images as our inner-most; thus a set-intersection between users and locations would now reduce our set so much that the inner loop now became faster. This had a tremendous impact on our computation time and we reduced computation time by 1 hr 10 min.

DIMENSIONALITY REDUCTION ON DATASET

For applying dimension reductionality, we have used CP - decomposition here, and this was made possible using the tensorly [5] library for Python3.

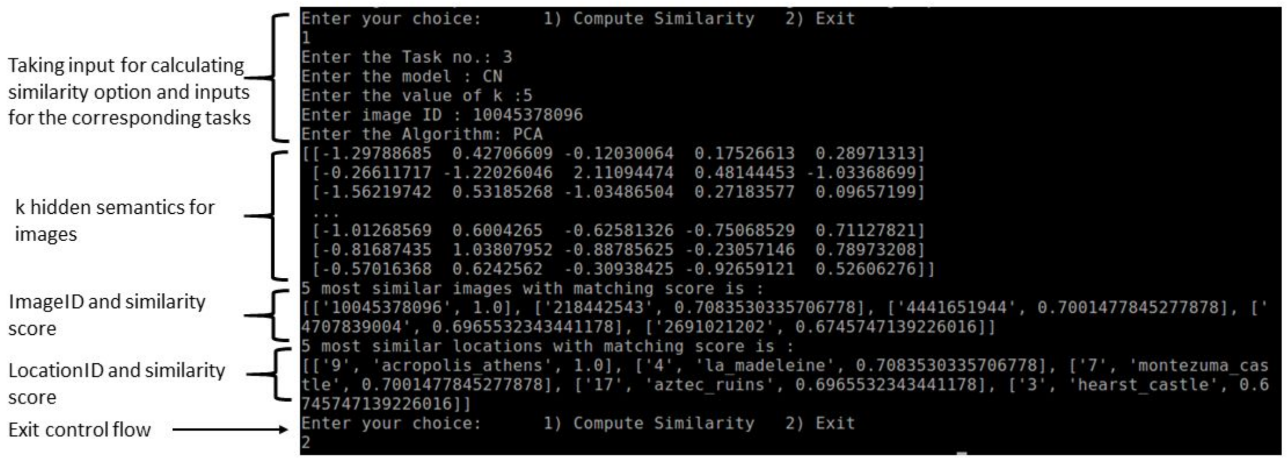
SIMILARITY COMPUTATION

K-means from the sklearn [6] library is used to find k common groups of users /locations and images on the reduced space obtained after having applied tensor decomposition successfully.

INTERFACE SPECIFICATION

COMMON INPUT INTERFACE

The driver program is run to invoke each of the tasks. The choice is between exiting the program and running the tasks. Once the user makes the choice of running the tasks, the user inputs the Task Number (1-7). Subsequently, the inputs specific to each task are provided as described below.


Taking input for calculating similarity option and inputs for the corresponding tasks
k hidden semantics for images
ImageID and similarity score
LocationID and similarity score
Exit control flow

```
Enter your choice:      1) Compute Similarity   2) Exit
1
Enter the Task no.: 3
Enter the model : CN
Enter the value of k :5
Enter image ID : 10045378096
Enter the Algorithm: PCA
[[-1.29788685  0.42706609 -0.12030064  0.17526613  0.28971313]
 [-0.26611717 -1.22026046  2.11094474  0.48144453 -1.03368699]
 [-1.56219742  0.53185268 -1.03486504  0.27183577  0.09657199]
 ...
 [-1.01268569  0.6004265  -0.62581326 -0.75068529  0.71127821]
 [-0.81687435  1.03807952 -0.88785625 -0.23057146  0.78973208]
 [-0.57016368  0.6242562  -0.30938425 -0.92659121  0.52606276]]
5 most similar images with matching score is :
[['10045378096', 1.0], ['218442543', 0.7083530335706778], ['4441651944', 0.7001477845277878], ['
4707839004', 0.6965532343441178], ['2691021202', 0.6745747139226016]]
5 most similar locations with matching score is :
[['9', 'acropolis_athens', 1.0], ['4', 'la_madeleine', 0.7083530335706778], ['7', 'montezuma_cas
tle', 0.7001477845277878], ['17', 'aztec_ruins', 0.6965532343441178], ['3', 'hearst_castle', 0.6
745747139226016]]
Enter your choice:      1) Compute Similarity   2) Exit
2
```

Fig 3. Shows the input interface and the related control flow for an example Task 3.

Task 1

INPUT INTERFACE

The driver program is run to take user query for this task.

Vector-space choice for performing the operation and dimensionality reduction choice is left to the user. Number of latent features, k, is provided by us in the form of user input.

DATA PRESENTATION

- Top-k latent semantics obtained, after performing the user-defined dimensionality reduction algorithm, in the corresponding(user/image/location) term space.
- These semantics are arranged in descending order of their term(feature)-weight pairs and the output is stored in a text file.

Task 2

INPUT INTERFACE
The driver program is run to take user query for this task. The corresponding entity id is taken as a user input depending on user, image or location space selection along with the dimensionality reduction algorithm.

DATA PRESENTATION

We need to display the following results
Top 5 similar user ids and their scores, images and their scores and locations and their scores with respect to the input (anyone of user_id,image_id,location_id)

Task 3

INPUT INTERFACE
The driver program is run to take user query for this task. While choosing computation for similarity and Task 3, we provide image ID for which top five similar image IDs and locations IDs need to be fetched. The number of hidden semantics, k is also taken as user input along with the dimensionality algorithm that needs to be run on the dataset.

DATA PRESENTATION

We display the following results -
- Top k latent semantics for the input image ID based on the algorithm chosen.
- Top five similar image IDs and location IDs along with the score with respect to the input image ID for the given model.

Task 4

INPUT INTERFACE

The driver program is run to take user query for this task. While choosing computation for similarity and Task 4, we provide location id for which top five similar locations need to be fetched. The number of hidden semantics, k is also taken as user input along with the dimensionality algorithm that needs to be run on the dataset.

DATA PRESENTATION

We display the following results -
- Top k latent semantics for the input location - based on the algorithm chosen

- Top five similar locations along with the score with respect to the input location for the given model

Task 5

INPUT INTERFACE

The driver program is run to take user query for this task. While choosing computation for similarity and Task 5, we provide location id for which top five similar locations need to be fetched. The number of hidden semantics, k is also taken as user input along with the dimensionality algorithm that needs to be run on the dataset.

DATA PRESENTATION

We display the following results -

- Top k latent semantics for the input location - based on the algorithm chosen
- Top five similar locations along with the score with respect to the input location

Task 6

INPUT INTERFACE

Once the driver program is run and this task is chosen, the input ‘k’ (an integer value) is taken from the user, representing the number of latent concepts to be found.

DATA PRESENTATION

We display the results as the following:

```
Latent Semantic: 1
        Location Name: <location_name> | Weight: <weight> (highest)
        Location Name: <location_name> | Weight: <weight> (second-highest)
        ….
        Location Name: <location_name> | Weight: <weight> (lowest)


Latent Semantic: 2
        Location Name: <location_name> | Weight: <weight> (highest)
        Location Name: <location_name> | Weight: <weight> (second-highest)
        ….
        Location Name: <location_name> | Weight: <weight> (lowest)

Latent Semantic: <Latent Semantic k>
        Location Name: <location_name> | Weight: <weight> (highest)
        Location Name: <location_name> | Weight: <weight> (second-highest)
        ….
        Location Name: <location_name> | Weight: <weight> (lowest)
```

Task 7

INPUT INTERFACE

The driver program is run to take user query for this task. While choosing computation for similarity and Task 7, we then provide k, (the number of latent dimensions and also the number of groups) as the only user input.

DATA PRESENTATION

K- similar groups of user, location and image groups are displayed in order.

SYSTEM REQUIREMENTS

- Programming Language: Python 3
- Libraries
    - NumPy (http://www.numpy.org/)
    - SciPy (https://www.scipy.org/)
    - SciKit Learn (http://scikit-learn.org/)
    - Tensorly (http://tensorly.org)
    - Pandas (https://pandas.pydata.org/)

These can be installed using the following command -

```
pip3 install -lv scipy==1.1.0
```

One driver program needs to be run - `driver.py` and this connects to all the tasks (written as modules).

`python3 driver.py` -> *to be run from the project directory submitted*

OBSERVATIONS

IMAGE - IMAGE SIMILARITY AND EFFECT OF K

To study the effect of K on image search engine, we kept parameters - model and imageID same for all the three dimensionality reduction algorithms and varied K from 2 to 10.

Here, we considered CN model and following are the results on applying **PCA** -

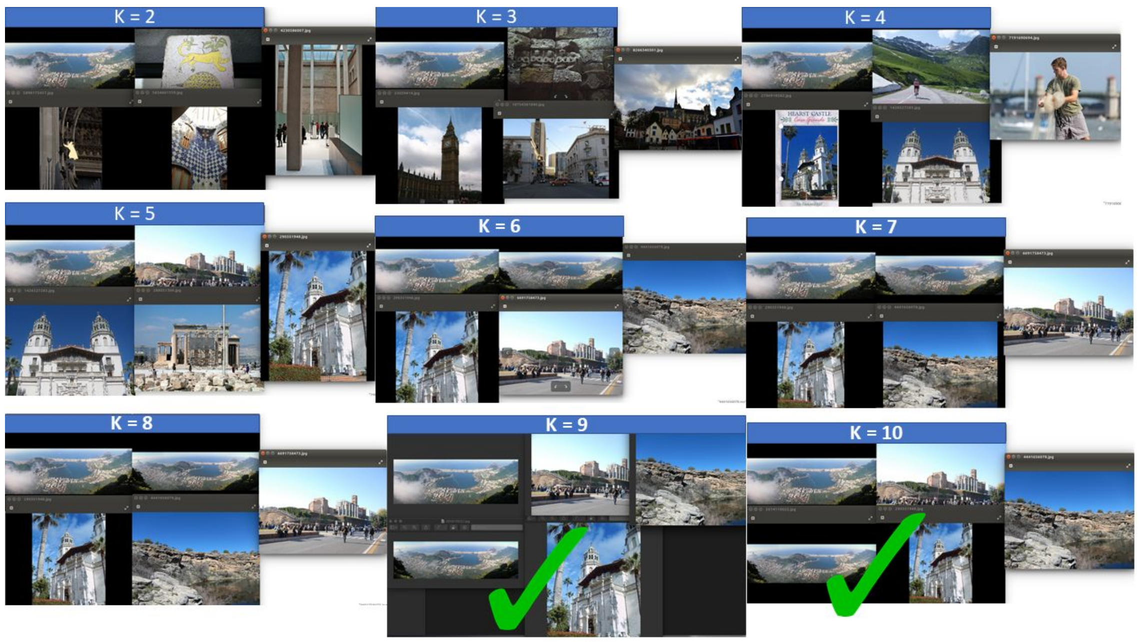


Fig.4 shows the effect of K on CN model when PCA is applied. The sub-images tick marked green are similar query results obtained for different values of k.

Each query result in the above image can be interpreted in the way that - eg. the result as shown in the top leftmost corner: the sub-image at top left is the input query image (which also is the most similar result or an equal match), the sub-image below this is the second most similar image and likewise the last sub-image is the most dissimilar in the top 5 results shown. As can be seen from the image, for k=9 and k=10 and the given query we are getting same results. Interestingly, this signifies -

- ❏ We can assume k=9 as the inherent dimension for the dataset.
- ❏ The results are in compliance with exact and partial query matching techniques, which implies that using PCA for CN model will guarantee a good image search engine for the given dataset.

Same analysis was done using **LDA** as dimensionality reduction algorithm, the results are as summarised in the image below -

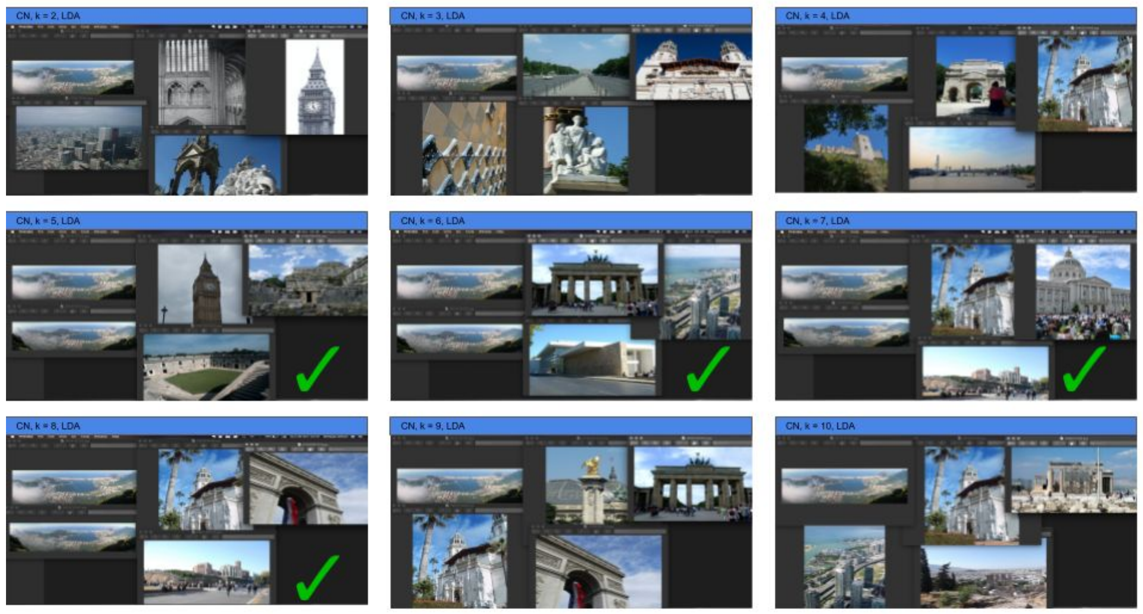


Fig.5 shows the effect of K on CN model when LDA is applied. The sub-images tick marked green are similar query results obtained for different values of k.

Here, we observe that the exact query matching started from k = 5 and the images retrieved for consecutive values of k were same, but had a different positional arrangement on the top 5 rankometer.

Next, we observe the variation of k for **SVD**. It clearly indicates, the results are similar post k equals 6. Hence, for CN model the inherent dimensionality found using SVD is 6. The partial matches show a good visual similarity with the query image.

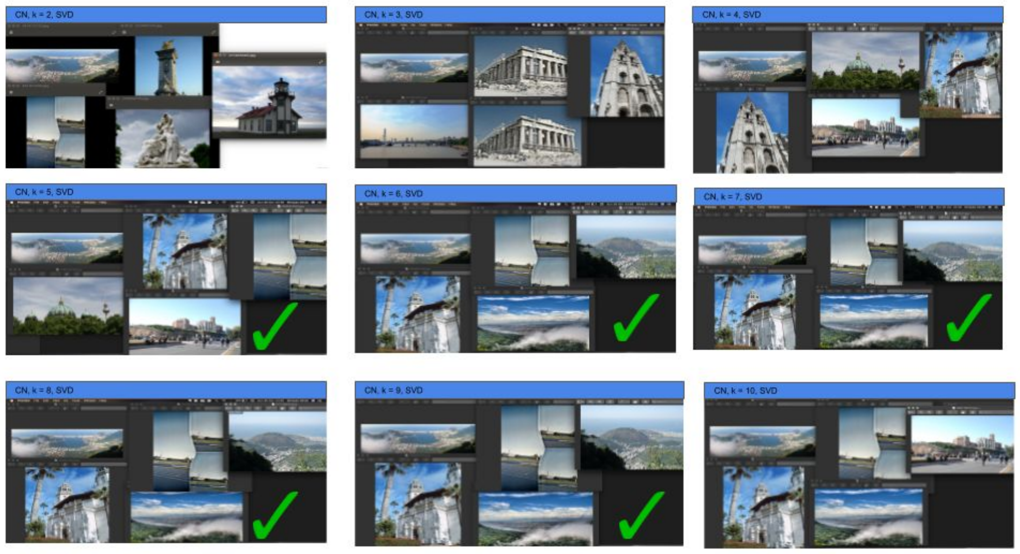


Fig.6 shows the effect of K on CN model when SVD is applied. The sub-images tick marked green are similar query results obtained for different values of k.

IMAGE - IMAGE SIMILARITY ACROSS VISUAL DESCRIPTOR MODELS

To study the effect of model on image search engine, we kept parameters - K (number of hidden semantics) and imageID same for all the three dimensionality reduction algorithms and ran the search engine for all the visual descriptor models - CN, HOG, LBP, CSD, etc.

Here, we considered k = 5 and ran search considering **SVD** as dimensionality algorithm - (the sub images shown in the image below can be interpreted in the same ways as in the above observation)

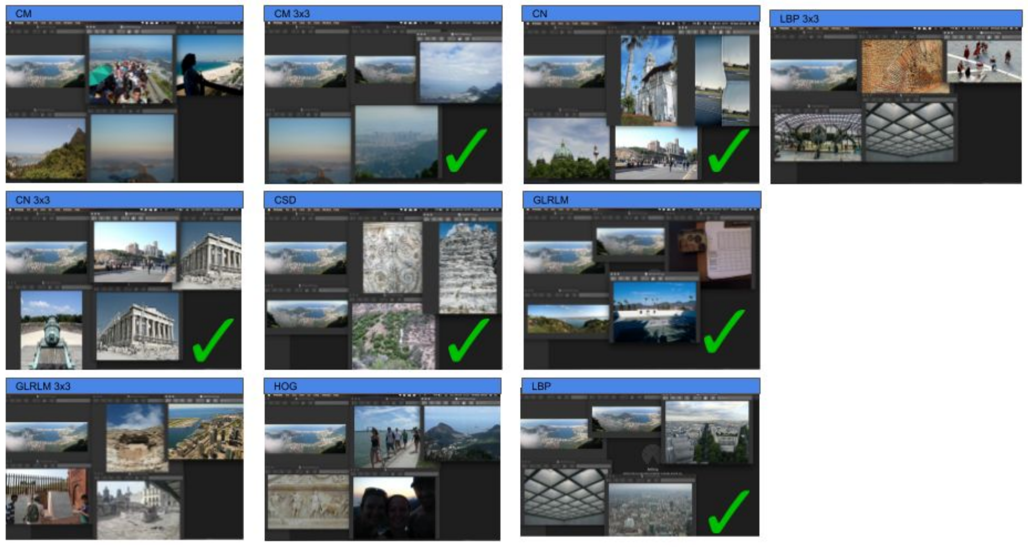


Fig.7 shows how different models evaluate based on given value of K and imageID using SVD. The green tick marked sub-images correspond to models showing good results.

We observe that -

- The models - CN, CN 3x3, CM 3x3, GLRLM, LBP and CSD gave good results in terms of exact and partial query match techniques.
- This observation above implies that Euclidean distance as a similarity measure works quite well with these models.

We ran similar analysis for **PCA** and **LDA** and the results are summarized in the images below.

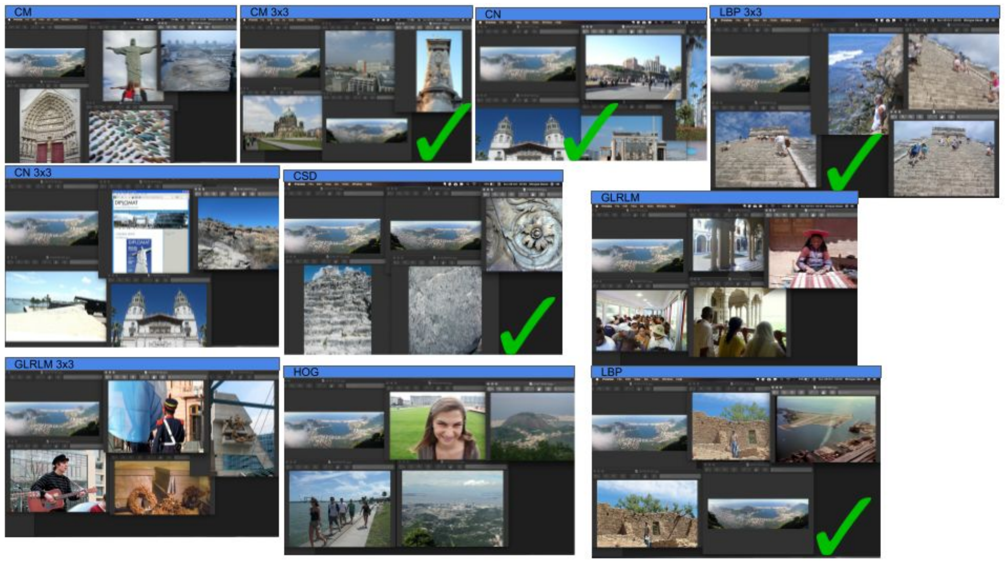


Fig.8 shows how different models evaluate based on given value of K and imageID using PCA. The green tick marked sub-images correspond to models showing good results.

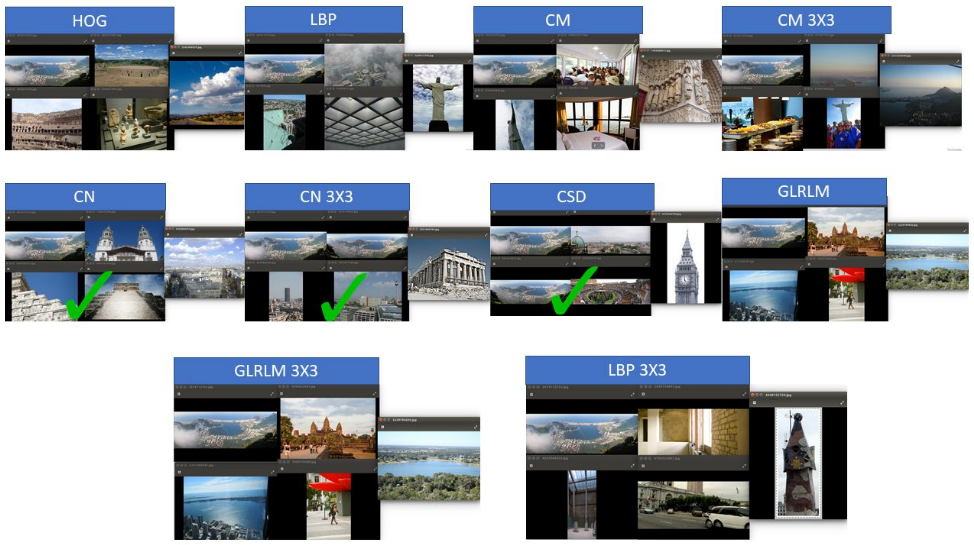


Fig.9 shows how different models evaluate based on given value of K and imageID using LDA. The green tick marked sub-images correspond to models showing good results.

CHANGING THE SIMILARITY METRIC TO EVALUATE QUERY RESULTS

We built a search engine implementation which used intersection similarity as the metric for query matches instead of Euclidean distance.

For same value of k = 5, same input imageID and HOG model, *Euclidean distance based search gave better results than intersection similarity*. The figure below captures the top matching result for both the implementations -

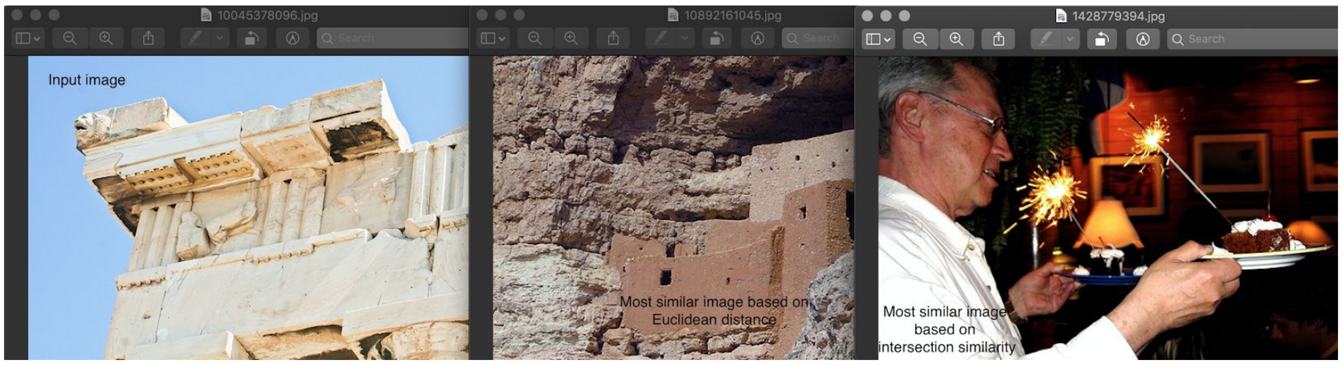

10045378096.jpg
Input image
10892161045.jpg
Most similar image based on Euclidean distance
1428779394.jpg
Most similar image based on intersection similarity

Fig.10 shows the top similar image for an Euclidean based similarity model and intersection similarity based model. The sub image at the centre is a better partial match than the one at the right for the input image (on the left), for HOG model.

HOG which represents a histogram of oriented gradients handles spatial contexts well. With this premise, the results obtained did not add value to the visual similarity of the image retrieved as for the end user it would be difficult to perceive visual similarity for the two images.
Since, the results were not that great we chose to build Euclidean distance based similarity engine across all the models.

TEXTUAL DESCRIPTOR BASED LOCATION - LOCATION SIMILARITY

For the purpose of evaluation of our approach in computing the similarity matrix, we assume that locations that are both of the same "kind" would show up as similar. This would mean different things based on how the terms are used in the respective locations. Two locations may be shown up as similar because of their structural characteristics (Cathedrals, Castles, etc.,) or their physical characteristics like where they are located (Both in the city of Rome).

We evaluate the accuracy of similarity matrix in defining the similarity by observing similar locations for a given location and cross-referencing with our intuitions about what locations could be great candidates for the given location.

For instance, a location that is a cathedral would have another cathedral location shown among the most similar locations and a location that is a castle would have another location that is a castle shown among the most similar locations. We could also potentially have a location shown up as most similar if they are in the same cities, because the user tags might describe the location a lot more among them.

Post similarity matrix computation, we observe that:

- Amiens Cathedral's most similar location apart from itself was Berlin Cathedral (Both Cathedrals).

- Chartres Cathedral’s most similar location apart from itself was Amiens Cathedral (Both Cathedrals).
- Montezuma Castle’s most similar location apart from itself was Hearst Castle (Both Castles).
- Colesseum’s most similar location apart from itself was Ara Pacis (Both located in Rome).

These results determined a positive evaluation of the approach that we have taken for computing the similarity matrix.

RELATED WORK

This section discusses about past and current work in the direction of similarity based ranking of entities. There has been a lot of activity around the search domain - query time optimisation, relevant query results retrieval using relevance feedback from users and machine learning algorithms, and so on; and we have implemented a similarity based retrieval tool for various entities given an ecosystem. Interpretation of similarity changes with the application - some might say that images having same color based on the colors extracted are more similar than images of the same object in two different filters eg. A grey shaded image of an apple and a mango may be similar in one application whereas a grey shaded apple and a sepia apple would be similar in the other.

User - user search is a classic example of one of the components of collaborative filtering based recommendation systems, where similar users are found based on the number of similar items of interest for both as explained by [2] Melville et al.

Image search moved to a dimension of semantics where image recognition is not the only way to decide similarity between the images as discussed in [3] G. et al.

Over the period of time, different models have come up, each model having different features of an image extracted and used for image similarity. Eg. Local binary pattern (LBP) yields good results while comparing image textures whereas CN has proven to be a good model when it comes to color match. Similarly, for text different models have shown their efficiency eg. word2vec is a tool built on the textual descriptors like TF, DF, TF-IDF and is widely used for word embeddings.

CONCLUSION

This was an exploratory phase where we used different dimensionality reduction algorithms to work with multimedia, find similarity between entities in the reduced space and evaluate our search engine implementation and ranking model, augmenting our knowledge base towards mechanics of multimedia and web world.

We used textual and visual descriptors effectively to compute similarity measure, eg. for text we used TF-IDF which is a good descriptor and a good discriminator and hence gave good results for location - location similarity computation model which was based on the idea of ranking the entities based on similar composition. Similarly, we used Euclidean distance as the similarity metric to compute image - image similarity and it gave good results.

At a broader level, the takeaway from the task was to appreciate the nuances of building a similarity ranking based search engine which given a input text/image would fetch the top matching or similar texts/images. We were exposed to different modeling techniques for text using vectors and tensors and images representing different dimensions for a given term in text corpus or an image.

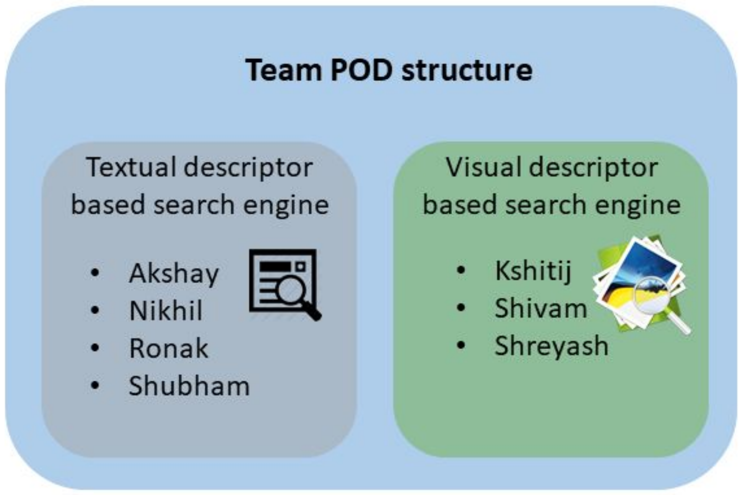
BIBLIOGRAPHY

[1] B. Ionescu, A. Popescu, M. Lupu, A.L. Ginsca, H. Muller, Retrieving Diverse Social Images at MediaEval 2014: Challenge, Dataset and Evaluation, MediaEval Benchmarking Initiative for Multimedia Evaluation, vol. 1263, CEUR-WS.org, ISSN: 1613-0073, October 16-17, Barcelona, Spain, 2014.
[2] Melville P., Sindhwani V. (2011) Recommender Systems. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA.
[3] G. Chechik, V. Sharma, U. Shalit (2010), Large Scale Online Learning of Image Similarity Through Ranking, Journal of Machine Learning Research 11, 1109-1135.
[4] G.W. Furnas, S. Deerwester, S.T. Dumais, T.K. Landauer, R.A. Harshman, L.S. Streeter and K.E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. Original in Proceedings of SIGIR 1998, 469-472
[5] http://tensorly.org/stable/modules/generated/tensorly.decomposition.parafac.html
[6] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

APPENDIX

List of Images:

Individual Contributions:


Team POD structure
Textual descriptor based search engine
Akshay
Nikhil
Ronak
Shubham
Visual descriptor based search engine
Kshitij
Shivam
Shreyash

Akshay Shah : Brain-stormed and worked upon an algorithm, fetching top 5 similar users,images and locations for given input entity id from the textual descriptor vector-space.

Nikhil Agarwal : Worked on creating a tensor based off the common terms present in the images-locations-users textual descriptor vector space; applying CP-decomposition on the tensor and returning k independent groups of users, images and locations using k-means clustering.

Ronak Tanna : Computed algorithms for searching similar locations based on the textual descriptors vector-space and applied SVD to reduce this space further and get k latent location name-weight pairs.

Shubham Verma : Formulated an algorithm to find top-k latent semantics within a given textual descriptor vector space by employing dimensionality reduction techniques such as PCA, SVD and LDA.

Kshitij Kashettiwar : Applied PCA, SVD and LDA dimensional reductionality techniques to the visual descriptors vector space and retrieved top 5 image IDs and location IDs given a user query.

Shivam Dhar : Crafted an algorithm to compute the 5 most similar locations given a query location; this was done upon the visual descriptors space by applying dimensionality reduction algorithms.

Shreyash Devan : Constructed an algorithm to get 5 most similar locations based on a query location and a particular color model. Dimensionality reduction techniques were applied to hasten the search and increase efficiency.