CSE 515 - MULTIMEDIA AND WEB DATABASES

PROJECT PHASE 2 REPORT

Mohan Maddula
Pranay Sai Dadi
Anvesh Reddy Koppela
Sravan Kumar Garipalli
Uma Sampath Mallampalli

Abstract
This phase of the project deals with the similarity measures calculation, performing different dimensionality reduction techniques on multi-dimensional feature space, discovery of the latent semantics from the epidemic simulation data sets. We also perform the K-nearest neighbor query with respect to the similarity measure and the latent semantics in the reduced space. This phase also deals with the mapping of objects into points in k-dimensional space preserving the distance between objects without knowing the original feature space. This phase of the project uses the epidemic simulation word, average, different files produced in the first phase of the project.

Keywords: Dynamic Time Warping (DTW), Feature Selection, Dimensionality Curse, Feature Significance, Data Distribution, Feature dependence, Intrinsic Dimensionality, Eigenvectors, Singular Value Decomposition (SVD), Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), Multidimensional Scaling (MDS), FastMap Algorithm.

Introduction

Similarity measures and their importance:

There are several measures to compute similarity between two epidemic simulation files. Some of them are metric and some of them are non-metric measures. We need to select one among the several measures which best suits the application requirements.

Different similarity measures include Euclidean measure, Dynamic Time Warping, Intersection Similarity, Quadratic Distance measures. In this phase we experiment with the above similarity measures to get k-nearest neighbors to an object or a query.

Feature Selection and Dimensionality Reduction:
Even though theoretically it is good to store as many features of the media as possible, in practical it is not feasible to do so, because of high indexing, storage and computation costs. Nearest neighbor search with more number of features becomes meaningless. So for efficient retrieval of the media data, we need to reduce the dimensionality of the data. To reduce the number of dimensions, there are lot of algorithms proposed by different computer scientists, such as Principle Component Analysis (PCA), Singular Value Decomposition (SVD), Probabilistic Aspect Model (PLSA), Latent Dirichlet Allocation (LDA), Multidimensional Scaling (MDS) and FastMap.

Depending on the application scenario, we need to select one algorithm to reduce the dimensionality of the data. Principal Component Analysis is used to capture the maximum variance of the data. If we want to preserve the dimensions along which variance is high, then PCA is used. But distances in the original space are not preserved. It under-estimates the distances among the data objects.

To preserve the distances among the objects in the original space, SVD is proposed. It maximally preserves the distances. Both the algorithms depend on the Eigen decomposition of the Input matrix, but they differ in the input matrix they accept.

The algorithms based on Eigen decomposition (PCA, SVD), don't take background knowledge into consideration. They are not statistically accurate. So Latent Dirichlet Allocation is used (LDA). It assumes a generative model for generation of epidemic simulation files by using the Expectation Maximization Concept to identify the latent topics from the available epidemic simulation files.

The above discussed algorithms, assumes that there is an initial set of features available to represent the given epidemic simulation files. So we will not be able to apply the above algorithms, if there is not clear method to extract initial set of features. Multidimensionality

Scaling and FastMap Algorithms are proposed to overcome this limitation. These algorithms require only distances among the epidemic simulation files to reduce them from some unknown space into the new space of user specified dimensions. FastMap resolves the high object mapping and query processing times introduced by MDS, by using the concept of selecting farthest pivot objects.

In this project we are dealing with the algorithms SVD, LDA, FastMap Algorithms to experiment with the epidemic simulation data sets for efficient retrieval and visualization of the data set.

Terminology

Euclidean distance: It measures the length between two points a and b in n -dimensional space.[3]

Euclidean Similarity: It is obtained using the Euclidean distance and is inversely proportional to the Euclidean distance. When Euclidean distance is high then the Euclidean similarity is low and vice versa.

DTW distance: It measures the distance between two temporal sequences which vary in time or speed.[4]

DTW Similarity: It is obtained using the DTW distance and is inversely proportional to the DTW distance. When DTW distance is high then the DTW similarity is low and vice versa.

Heat Map Visualization: It is the graphical representation of the data where the individual values are represented using colors. [5]

Latent Semantics: These are the hidden topics which are present in the simulation files, which are obtained by analyzing the relationship between set of simulations and the terms present in them.[6]

Dimensionality Reduction: It is the process of mapping objects from N-dimensional vector space to k- dimensional space where $k<<N$.

SVD: It is a technique for identifying a transformation that can take data described in terms of n feature dimensions and map them into a vector space defined by $k \leq n$ orthogonal basis vectors.[7]

LDA: It is a generative model for extracting latent topics from the epidemic simulation files using expectation maximization approach.

FastMap: It is the process of mapping objects into points in some k-dimensional space (k is user-defined), such that the dis-similarities are preserved.[2]

File 1 : The object 1 that has been selected from the user input. The object can be either of the simulation files or epidemic word files, epidemic word files average, epidemic word files difference.

File 2 : The object 2 that has been selected from the user input. The object can be either of the simulation files or epidemic word files, epidemic word files average, epidemic word files difference.

Goal Description

The project has 4 tasks to be accomplished.

- **Task 1 - Time Series Similarity**
  In this task we need to implement various similarity measures applied to multimedia data objects such as Euclidean Similarity, Dynamic Time Warping (DTW), Intersection Similarity and Quadratic Distance. Tasks 1a to 1e assumes that the simulation words occur independently i.e. the occurrence of one word will not affect the occurrence of another word. This assumption is lifted while computing the tasks from 1f to 1h where we compute the similarity between two words and include it in the similarity score.

  - **Task 1a**
    This task computes the Euclidean Distance between two given epidemic simulation files. Using Euclidean Distance we need to compute the similarity between two objects in multidimensional space.
    - If we consider epidemic simulation files as the objects in multidimensional space, this task measures the similarity between these two files.
    - If epidemic simulation files are considered analogous to the year in which the disease outbreak has occurred, this measure helps us to identify the years in which the years has follow the same pattern with respect to state and time.
    - Euclidean distance has an advantage that it is a similarity metric.
    - If epidemic simulation files are considered analogous to different simulation of diseases or years, by computing the Euclidean similarity we can predict the severity of the disease.
      - If the files are very similarity then there is no impact on the severity of the disease.
      - If the files are very dissimilar, then we can conclude that either the disease has either eradicated or became intense.

- **Task 1b**
  This task computes the Dynamic Time Warping (DTW) distance between two given epidemic simulation files. As mentioned in the paper “Exact indexing of dynamic time warping” by Eamonn Keogh, Chotirat Ann Ratanamahatana, Euclidean distance cannot index time series accurately among two different time phases.
  - DTW allows elastic shifting of time axis to matches sequences similar in shape even though they are out of phase in time axis [1]. Since, DTW distance is based on dynamic programming; it provides a reliable measure in comparing the similarity between two time series.
  - In this task, we compute the similarity between two epidemic simulation files using DTW distance measure.
  - If epidemic simulation files are considered analogous to different simulation of diseases or years, by computing the DTW similarity we can predict the severity of the disease.
    - If the files are very similarity then there is no impact on the severity of the disease even if they are out of phase.
    - If the files are very dissimilar, then we can conclude that either the disease has either eradicated or became intense.

- **Task 1c**
  The word simulation files are used to track the transmission of a disease. If the dataset iterations are considered analogous to days, depending on the window length, the epidemic word files determine the number of simulations observed. This task computes the unnormalized version of intersection similarity between two given word simulation files. It measures the degree of overlap of the two epidemic word files without considering the state and time.
  - If word simulation files are considered analogous to different simulation of diseases or years, by computing the word similarity we can infer how similarly the files are with respect to disease progress.
    - If the files are very similarity then the disease has not progressed
    - If the files are very dissimilar, then we can conclude that either the disease has eradicated or became intense.

- **Task 1d**
  The average word files are used to determine the impact of the disease on the neighboring states and the current state. This task computes the unnormalized version of intersection similarity between two given word simulation files. It measures the degree of overlap of the two average word files without considering the state and time.

        - If average word files are considered analogous to different simulation of diseases or years, by computing the average similarity we can predict how similarly the files are with respect to the impact of neighboring states on disease progress.
            - If the files are very similarity then the neighboring states have not impacted much on the disease progress.
            - If the files are very dissimilar, then we can conclude that the impact of neighboring states has been either eradicated or became intense.
    - **Task 1e**
      Since, the difference word files are used to measure the residual severity of the disease in the current state from its neighboring states. This task computes the similarity between two given difference simulation files. It measures the degree of overlap of the two different word files without considering the state and time.
        - If difference word files are considered analogous to different simulation of diseases or years, by computing the difference similarity we can predict how similarly the files are with respect to the residual impact of neighboring states on disease progress.
            - If the files are very similarity then the neighboring states have not impacted much on the disease progress.
            - If the files are very dissimilar, then we can conclude that the impact of neighboring states has been either eradicated or became intense.

    - **Task 1f**
      Among the various distance measures implemented in the previous tasks, this task identifies the similarities in the epidemic word files by the following measures.
        - the closeness between the state and time pairs of both the compared file words
        - The level of discrimination that 2 compared epidemic words in the files is in the database.

      Based on these measures, the similarity is constructed on the lines of the Quadratic distance, where A matrix maps each epidemic word's similarity from file 1 and file 2 by evaluating as mentioned before. The objects to be compared are treated as binary vectors containing the combined words of state, time and words and the overall similarity is computed.

    - **Task 1g**
      This task identifies the similarities in the epidemic average word files by the same measure following.

        - the closeness between the state and time pairs of both the compared file words
        - The level of discrimination that 2 compared epidemic average words in the files is in the database.

        Based on these measures, the similarity is constructed on the lines of the Quadratic distance, where A matrix maps each epidemic average word's similarity from file 1 and file 2 by evaluating as mentioned before. The objects to be compared are treated as binary vectors containing the combined words of state, time and words and the overall similarity is computed.

    - **Task 1h**

        This task identifies the similarities in the epidemic word difference files by the following measures.

        - the closeness between the state and time pairs of both the compared file words
        - The level of discrimination that 2 compared epidemic difference words in the files is in the database.

        Based on these measures, the similarity is constructed on the lines of the Quadratic distance, where A matrix maps each epidemic difference word's similarity from file 1 and file 2 by evaluating as mentioned before. The objects to be compared are treated as binary vectors containing the combined words of state, time and words and the overall similarity is computed.

- **Task 2**

    In this task we calculate the top K similar objects with respect to the given file. Since, different similarity measures have different levels of compactness, depending on the similarity measure; we get different top K similar objects with respect to the query.

- **Task 3**

    This task deals with the discovery of the hidden latent semantics from the set of epidemic simulation files and also performs the k-nearest neighbor search on set of epidemic simulation files for given query file with respect to the latent semantics. This task is mainly for the dimensionality reduction techniques like SVD, LDA. This gives the semantic scores of the files in non-increasing order for the r latent semantics. Also, this gives the k-nearest neighbors based on the semantic scores.

- **Task4**

    Analysis of Multimedia data involves preprocessing of the data for efficient retrieval and visualization of the data. One kind of pre-processing is feature-extraction from the data. Feature extraction can be done in two ways,

    - By relying on domain experts to give k-features in the data,

- By deriving feature extraction functions like PCA, SVD etc.

But It is always not an easy task to obtain the features using any of the above methods. If there are only distances between objects are given and no information about features is present, the above methods don’t work. We need algorithms which depend on only distances to map the objects into new dimensional space and make no assumptions about underlying features of the objects.

Multidimensional Scaling (MDS)is one such algorithm, but it is not widely used because of it’s inefficient performance issues for mapping the objects into new space and also for processing the query.

To solve the problems in the MDS algorithm, Christos Faloutsos and King-Ip (David) Lin proposed a new Algorithm called “FastMap”, which is based on MDS algorithm and reduces the high time complexity problem of both object mapping and query processing in the new space (k-dimensions) significantly.

Task 4a:

Given a set of objects and a distance metric, the main goal of this task is to map these set of objects using FastMap algorithm into new r-dimensional space, where r is specified by the user. The idea of this mapping is the Euclidean distances among the objects in the r-dimensional space matches the original distances, such that the mapping error is minimized. This reduction is useful in efficient retrieval and visualization of the data objects.

Input: A set of objects (say of count n) and distances among the objects (say n x n matrix) and required number of dimensions (r)

Output: Object representation in the reduced space (of r-dimensions), Mapping error

Task 4b:

This task is about processing the user query to retrieve the similar objects in the data set. To do so, since the user query is in different space, to compare it with the objects in the new r-dimensional space, we need to map the query object into the new r-dimensional space. Top k results according to the Euclidean distance metric are given to the user.

Input : A query object and the required number of results (say top k results)

Output : Top k results for a given query.

<u>Assumptions</u>

- All the input simulation files are assumed to be in the same directory and the file naming convention is assumed to be same as the naming convention used in the sample dataset provided (1.csv, 2.csv).
- The simulation files have uniform meta data structure in all the files. The number of rows and columns in each file are also assumed to be the same and the states are also assumed to be in alphabetical order.

- All the tasks assume that code for phase 1 is executed before executing the phase 2 task.
- We considered window = 8, shift = 5 and resolution = 4 for generating the word, difference and average files.
- Task 1f also takes as an input the location matrix, to determine the closeness between 2 states.
- In task 2, if two files have the same similarity scores with respect to the query, we output the first top K similar values after sorting the values.
- In task 3, for LDA to run we will have to put the supporting Gibbs Sampling dll and cpp files in place.
- In task 3, the number of k nearest neighbor's value should be less than the total number of documents.
- In task 3, the value of r entered should be less than the total number of features.
- In task 3, we did not implement Incremental SVD on the assumption that the query file might have new features.
- In task 3, it is assumed that the word files (word, average, difference) for the epidemic simulations and query file are already present.
- In task 3, if there are two scores at same value, we choose the first one.
- In task 3, it is assumed that number of iterations is kept as 500; seed is set to a value, alpha, beta values are set to optimal values in analysis.
- In task 3, all intermediate files getting generated would be deleted before continuing with the next steps.
- In task 3, the folder which has the query file should contain only one query file
- In task 4, to convert the similarity to distance we identified that distance is inversely proportional to similarity.
- In task 4, the distant measures used in mapping the objects used in the fast map implementation are non-negative, symmetric and obeys the triangular inequality.
- In task 4, FastMap algorithm assumes no outliers are present in the given data set of objects.
- In the task 4, distances among the objects in the reduced space are computed using Euclidean distance metric.
- In the task 4, while displaying the top results for a query, if there are any ties among the documents similarity, then first document is selected according to its name. Suppose, if there is a tie between 1.csv and 2.csv and one of these has to be returned, and then 1.csv is returned.
- In task 4, during query processing, we assume the number of dimension for query is equal to the number of dimensions for objects.

<u>Description of the proposed solution/implementation</u>

Below are the proposed solutions/implementations of the project, task level and the various analysis and approaches following by our team.

Task 1

- **Task 1a**
  In this task we compute the Euclidean similarity between two given epidemic simulation files. To compute the Euclidean similarity between two files f1 and f2 we have used the following formula.

$$sim_{Euc}(f_1, f_2) = \frac{1}{1 + AVG_{s_i \in states}\, \Delta_{Euc}(f_1.s_i, f_2.s_i)}$$

  - Here f1 and f2 corresponds to the two epidemic simulation files.
  - S_i corresponds to each state id with respect to the two epidemic simulation files.
  - The following formula is used to compute the Euclidean distance between two states.
    If we consider < x1, x2, x3 …. xn > as the simulation values for state si in simulation file f1 and <y1, y2, y3 …. yn> as the simulation values for state si in simulation file f2 then the Euclidean distance between these two states is computed as the two norms between the two vectors.

$$\text{Euclidean Distance} = \text{Sqrt}\ ((x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + (x_n - y_n)^2)$$

Pseudo Code

```
Read the data for the files f1 and f2
distance = 0;
For each state $s_i$
    distance = distance + euclidean distance between the two simulation vectors $f_1.s_i$, $f_2.s_i$ using the above equation
end
Calculate the average similarity for all the states in the files as distance / No of states
Calculate the similarity from average distance based on the formula described above
```

The pair wise distance between first five simulation files are computed as

File	1.csv	2.csv	3.csv	4.csv	5.csv
1.csv	1	1.94E-05	1.58E-05	2.36E-05	2.11E-05

2.csv	1.94E-05	1	3.57E-05	2.89E-05	1.54E-05
3.csv	1.58E-05	3.57E-05	1	1.99E-05	1.41E-05
4.csv	2.36E-05	2.89E-05	1.99E-05	1	1.83E-05
5.csv	2.11E-05	1.54E-05	1.41E-05	1.83E-05	1

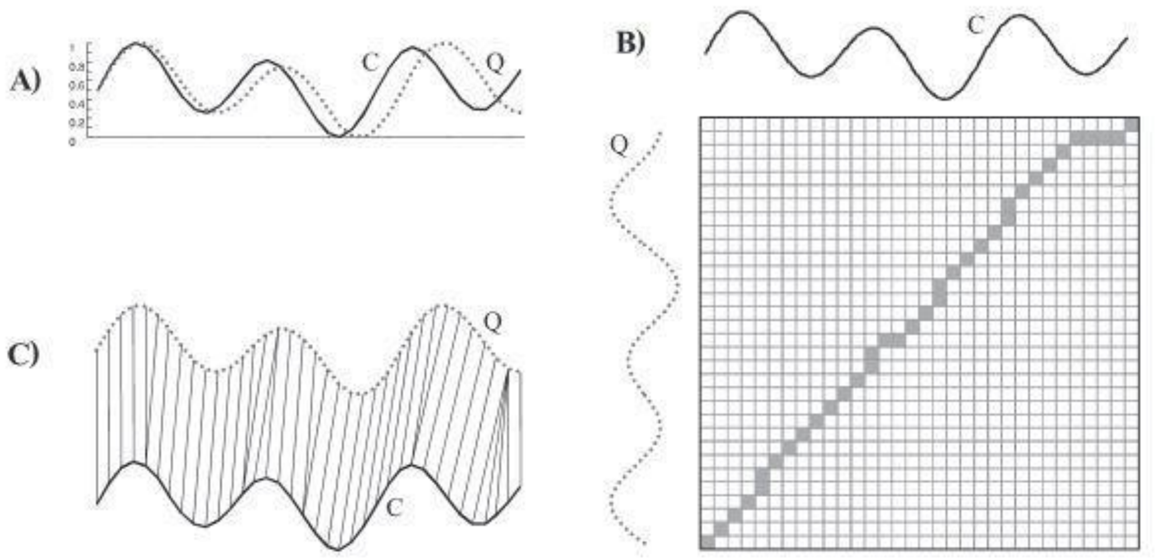
Observations

- We observed that the euclidean similarity is symmetric.
- Since we are dealing with time series, the euclidean similarity gives very less values.

- **Task 1b**
  In this task we compute the Dynamic Time Warping (DTW) similarity between two files f1 and f2 based on the following formula. The advantage of using DTW distance over euclidean distance is discussed in the goal description part.

$$sim_{DTW}(f_1, f_2) = \frac{1}{1 + AVG_{s_i \in states}\, \Delta_{DTW}(f_1.s_i, f_2.s_i)}$$

  - Here f1 and f2 corresponds to the two epidemic simulation files.
  - S_i corresponds to each state id with respect to the two epidemic simulation files.
  - To compute the DTW distance between two time vectors of state s_i with respect to the files f1 and f2 is computed using following procedure
    If we consider X = < x1, x2, x3 …. xn > as the simulation vectors for state si in simulation file f1 and Y = <y1, y2, y3 …. yn> as the simulation vector for state si in simulation file f2 then the DTW distance between these two states is computed using the mentioned pseudo code.
    - Distance Function for computing the similarity in DTW is distance $(x_i, y_i) = (x_i - y_i)^2$ where x_i and y_i correspond to the values of the instances i^th and j^th position in X and Y respectively.



The figure shows the computation of DTW distance between the sequences Q and C as shown in the figure [1]

Pseudo Code for DTW Distance

```
Obtain the number of instances for sequence X and Y. In our case the number of iterations will be the same (n).
Initialize the M(0,0) = 0 which corresponds to time 0 in sequence X and time 0 in sequence Y.
Construct an n x n matrix M where the i^th and j^th position correspond to the alignment between the points $x_i$ and $y_j$. The alignment is computed based on the function
            $distance(x_i, y_i) = (x_i - y_i)^2$
Construct the warping path that constitutes the minimum distance between two sequence which is computed using
        $M(i,j) = distance(x_i, y_i) + min([M(i-1,j)\ M(i,j-1)\ M(i-1,j-1)]);$
Return M(n,n)
```

Pseudo Code

```
Read the data for the files f1 and f2
distance = 0;
For each state $s_i$
    distance = distance + DTW distance between the two simulation vectors
    $f_1.s_i$, $f_2.s_i$ using the above described pseudo code
end
Calculate the average similarity for all the states in the files as distance / No of states
```

Calculate the similarity from average distance based on the formula described above

The DTW distance for first five simulation files are

File	**1.csv**	**2.csv**	**3.csv**	**4.csv**	**5.csv**
1.csv	1	2.25E-09	2.00E-09	1.33E-09	1.26E-09
2.csv	2.25E-09	1	3.68E-09	2.66E-09	1.33E-09
3.csv	2.00E-09	3.68E-09	1	3.08E-09	1.24E-09
4.csv	1.33E-09	2.66E-09	3.08E-09	1	2.33E-09
5.csv	1.26E-09	1.33E-09	1.24E-09	2.33E-09	1

Observations

- We observed that DTW similarity is symmetric.
- We observed that DTW similarity differs from euclidean similarity.

- **Task 1c**
  In this task we compute the unnormalized intersection similarity between two word simulation files f1 and f2.

  Approach 1: Constructing binary vectors w1 and w2

  - The *epidemic simulation word files* are considered for calculating the similarity.
  - All unique words ie; <win> are considered from each row of the epidemic word file <f, s, t, win>.
  - The epidemic word file 1 and epidemic word file 2 are iterated on all the rows to get the list of all the unique words $<win_i>$.
  - These unique words are maintained in the HashMap.

$$sim_{word}(f_1, f_2) = \vec{w_1}\vec{w_2}$$

w1 and w2 are the binary vectors of file1 and file2 respectively, the length of each w1 and w2 is equal to the total number of unique words combined in both the files.

The vector is represented as w_i=<b_{i1}, b_{i2}, $b_{i3,}$ ………, b_{in}>, where n is the total number of unique words combined in both the files.

The value of b_{ij} is 1 if $word_j$ is present in $file_i$

The value of b_{ij} is 0 if $word_j$ is not present in $file_i$

Pseudo Code

```
Read the word files file1 and file2
Get all the unique words from both files file1 and file2.
Compute binary vectors w1 and w2.
Calculate the dot product of the vectors w1 and w2.
Print the similarity value.
```

Consider for example there are 5 unique words combined in both file1 and file2.

	word1	word2	word3	word4	word5
w1	1	1	1	0	0
w2	0	1	1	1	1

- This implies that word1 is present in only file1; word 4 and word 5 are present in only file2; and word2 and word3 are present in both files file1 and file2.
- Now similarity between file1 and file2 are calculated using the formula mentioned above, which is the dot product of the two vectors w1 and w2.

Therefore $\text{sim}_{\text{word}}(\text{file1}, \text{file2}) = \langle 1, 1, 1, 0, 0\rangle . \langle 0, 1, 1, 1, 1\rangle$
$= (1*0)+(1*1)+(1*1)+(0*1)+(0*1)$
$= 0 + 1 + 1 + 0 + 0 = 2.$

The word similarity computed for first five word files are

File	**1.csv**	**2.csv**	**3.csv**	**4.csv**	**5.csv**
1.csv	11	2	2	3	3
2.csv	2	4	3	2	2
3.csv	2	3	4	2	2
4.csv	3	2	2	5	2
5.csv	3	2	2	2	6

Approach 2: Without constructing the binary vectors w1 and w2

In this approach we have eliminated the construction of binary vectors for the files to be compared. To compute the similarity we have computed the unique words in file f1 and unique words in f2 and calculated the intersection between them as

$$sim_{word}(f_1, f_2) = \vec{w_1}\vec{w_2}$$

- f1 and f2 corresponds to either word simulation file
- w_1 and w_2 corresponds to the unique words present in the files f1 and f2 without considering state and time.
- Similarity is measured as the intersection of unique words in the two sets $\mathbf{w_1}$ and $\mathbf{w_2}$.

  Consider the example where two files contain the following words

f1	a	b	d	a	b
f2	c	b	h	c	a

- w_1 = Unique words present in f1 are a, b, d
- w_2 = Unique words present in f2 are c, b, a, h
- Similarity = Intersection of w_1 and w_2 = 2

The word similarity computed for first five word files from given set of files is as follows

File	1.csv	2.csv	3.csv	4.csv	5.csv
1.csv	11	2	2	3	3
2.csv	2	4	3	2	2
3.csv	2	3	4	2	2
4.csv	3	2	2	5	2
5.csv	3	2	2	2	6

Observations:

- sim $_{word}$ (file1, file2) measure is nothing but the number of common words present in both file1 and file2.
- sim $_{word}$ (file1, file2) is symmetric ie; sim $_{word}$ (file1, file2) = sim $_{word}$ (file2, file1).

- **Task 1d**
  In this task we compute the unnormalized intersection similarity between two average simulation files f1 and f2.

  Approach 1: Constructing the binary vectors w1 and w2
  - The approach for this task is same as Task 1c with changes as mentioned below.
  - The *epidemic simulation average word files* are considered for calculating the similarity.
  - All unique average words i.e.; <win> are considered from each row of the epidemic average word file <f, s, t, win>.
  - The formula for calculating the sim_{avg_word} is as follows

$$sim_{avg\_word}(f_1, f_2) = \vec{w_{avg,1}} \vec{w_{avg,2}}$$

  - The logic for construction of both w_{avg1} and w_{avg2} are similar as mentioned in Task 1c.

  Approach 2: Without constructing the binary vectors w1 and w2
  The approach to this task is same as Task 1c with the changes as mentioned below. To compute the similarity we have computed the unique words in file f1 and unique words in f2 and calculated the intersection between them as

$$sim_{avg\_word}(f_1, f_2) = \vec{w_{avg,1}} \vec{w_{avg,2}}$$

  - f1 and f2 corresponds to average word file
  - $w_{avg,1}$ and $w_{avg,2}$ corresponds to the unique words present in the files f1 and f2 without considering state and time.
  - Similarity is measured as the intersection of unique words in the two sets $w_{avg,1}$ and $w_{avg,2}$.

The average similarity computed for first five average word files from given set of files is

File	**1.csv**	**2.csv**	**3.csv**	**4.csv**	**5.csv**
1.csv	32	5	5	8	8
2.csv	5	11	8	5	5
3.csv	5	8	11	5	5
4.csv	8	5	5	14	5
5.csv	8	5	5	5	17

Observations:

- sim $_{avg\_word}$ (file1, file2) is symmetric
  i.e; sim $_{avg\_word}$ (file1, file2) = sim $_{avg\_word}$ (file2, file1).

- **Task 1e**
  In this task we compute the unnormalized intersection similarity between two average simulation files f1 and f2.
  Approach: Constructing the binary vectors w1 and w2
  - The approach for this task is same as Task 1c with changes as mentioned below
  - The *epidemic simulation difference word files* are considered for calculating the similarity.
  - All unique difference words i.e.; <win> are considered from each row of the epidemic average word file <f, s, t, win>.
  - The formula for calculating the sim$_{diff\_word}$ is as follows

$$sim_{diff\_word}(f_1, f_2) = w_{\vec{diff},1} w_{\vec{diff},2}$$

  - The logic for construction of both w$_{diff1}$ and w$_{diff2}$ are similar as mentioned in Task 1c.

Observations:

- sim$_{diff\_word}$ (file1, file2) is symmetric
  i.e; sim$_{diff\_word}$ (file1, file2) = sim$_{diff\_word}$ (file2, file1).

- **Task 1f**
  This task obtains the files file 1, file 2 which are the epidemic simulation files and connectivity graph as specified in the user input. This function simulates the quadratic distance by constructing A matrix, a similarity measure by evaluating the words from the input file1 and file2. The final similarity measure is reported by evaluating the total similarity for the given input as mentioned below.

$$sim_{weighted\_word}(f_1, f_2) = \vec{w_1} A (\vec{w_2})^T$$

The A matrix is calculated by using the following measure as indicated below.

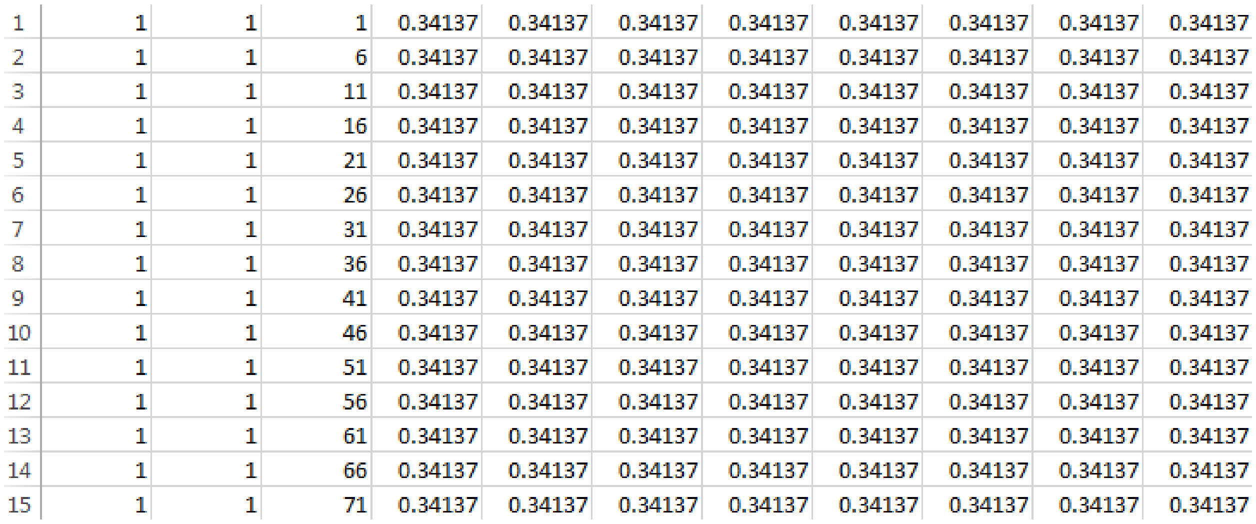
$$A[idx_{1,h}, idx_{2,j}] = A[\langle\langle f_1, s_h, t_h\rangle, \vec{win}_h\rangle, \langle\langle f_2, s_j, t_j\rangle, \vec{win}_j\rangle]$$

where <f$_1$,s$_h$,t$_h$> are the indexes extracted from each row and the win$_h$ is the word extracted from the file representing the corresponding index. Similarly the same nomenclature can be defined for the file 2.

Proposed Solution:

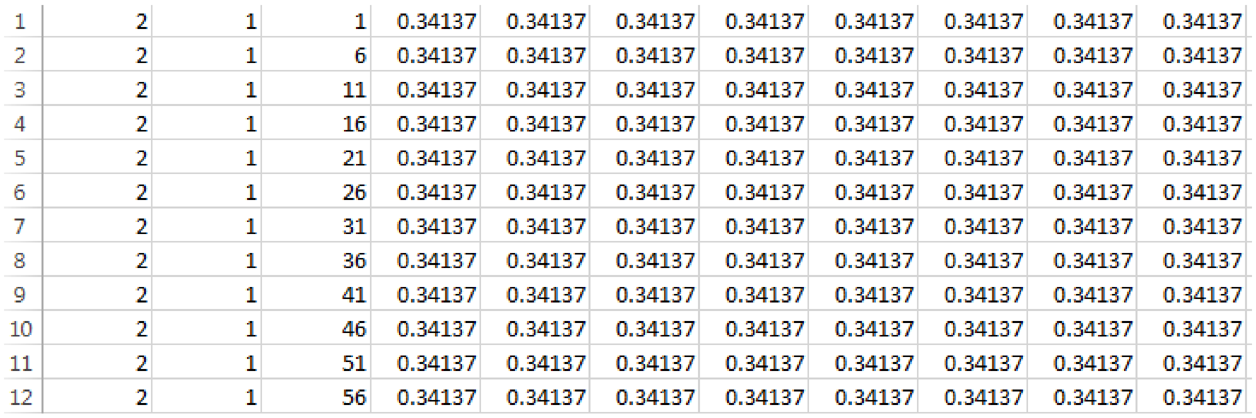
The file1 and file2 are read and respective matrices are constructed from each file, ignoring the file id from the index. Once the matrices are constructed, we have the respective words along with the state and time index values from each file. Then, the A matrix measure is constructed as indicated below.

Each row, column pair in the matrix represents the respective file 1 and file 2 rows. For example let us consider the file 1 as indicated in the below figure.



1	1	1	1	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
2	1	1	6	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
3	1	1	11	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
4	1	1	16	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
5	1	1	21	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
6	1	1	26	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
7	1	1	31	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
8	1	1	36	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
9	1	1	41	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
10	1	1	46	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
11	1	1	51	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
12	1	1	56	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
13	1	1	61	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
14	1	1	66	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
15	1	1	71	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137

and file 2 as indicated below



1	2	1	1	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
2	2	1	6	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
3	2	1	11	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
4	2	1	16	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
5	2	1	21	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
6	2	1	26	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
7	2	1	31	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
8	2	1	36	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
9	2	1	41	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
10	2	1	46	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
11	2	1	51	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137
12	2	1	56	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137	0.34137

The A matrix for these files would contain a total of 15*12 = 180 cells, with each cell containing a similarity measure. For example the (2,3) index of the matrix (index start from 1), would contain the similarity score evaluated for file 1 word 2 and file 2 word 3 respectively. Similarly the matrixes for all the indexes are computed.

Evaluation of A matrix:

As previously mentioned A matrix is calculated from the formula as below

$$A[idx_{1,h}, idx_{2,j}] = A[\langle\langle f_1, s_h, t_h\rangle, \vec{win}_h\rangle, \langle\langle f_2, s_j, t_j\rangle, \vec{win}_j\rangle]$$

where each entry in the matrix is evaluated how close the two state and time pairs are and how discriminating the words being considered are in the database.

Similarity Measure and Formulas used:

The evaluated entry in the A matrix should be a similarity metric. The closeness between two states can be measured by the shortest path between the two states which is a distance measure. Since the distance and similarity measures are inversely proportional and to avoid the divide by zero error we have converted the distance measure to similarity measure by using the formula

$$similarity\ measure = \frac{1}{1+distance\ measure}$$

The above stated formula is used for determining both the time and state pairs closeness similarity.

Our thought process in evaluating each metric!

State pair closeness:

The first similarity factor should determine the closeness between the states for a given words. Given an adjacency matrix representing the one hop neighborhood of the states, an undirected graph can be obtained we have considered various approaches like degree centrality and betweenness centrality if anything about closeness in a graph can be obtained. The degree centrality determines how connected the state is with the other states which can give a good measure in determining the risk of a state for getting contaminating its neighbors with the epidemic.[7] The betweenness centrality determines how many number of times the state can be traversed by a shortest path distance between any two other states.[7]

The closeness can also be determined by the shortest path between the two states that can be evaluated by Dijkstra's shortest path algorithm, which when given an input node, a destination node and an adjacency matrix would determine the shortest path between them.

We have decided to consider the shortest path between two states since they give a much meaningful measure of closeness. Since we already have the adjacency matrix representing the one hop neighborhood between the states as given by the location matrix from the user input and we also know that breadth first search algorithm has the same effect as Dijkstra's algorithm when the node weights all have weight values as 1 and the connectivity graph is an undirected graph, the breadth first search algorithm is decided to be applied between the state pairs in order to obtain the shortest distance between the two state pairs. We have used a built in function referred from mathworks networks[8] that takes the connectivity graph as input and instantly gives us the shortest path between two states. The distance measure is then converted to a similarity measure using the formula stated above.

Time Pair Closeness:
We also have the time iterations from the index which have values representing a one dimensional data, hence one norm or the Manhattan distance is decided to be used to determine the closeness measure between any given time values. We have evaluated the similarity measure from the obtained distance measure as defined in the formula stated before.

How is the discriminating factor of a word in the database being computed and our thought process?

The words in the file 1 and file 2 being compared should be a discriminating factor in the entire sample files (database) given. We have analyzed various approaches like getting the euclidean distance between the words, by considering each value in the window as a histogram value and then obtaining the euclidean distance.

Approaches like string kernel method by getting the words as bigrams converting them into vector space and then compare them by using cosine similarity were also considered during the initial brainstorming sessions.

We also have considered using edit distance if any meaningful discrimination could be obtained. We however have identified that the level of discrimination in the database for a given word is effectively given by the **IDF factor** [10] of a word. The IDF factor of a word is more if the word is less frequent and vice versa and hence will give the discrimination of words in the database.

Choosing a measure which collectively gives the discrimination of both the words is a challenge and we have tried various collective measures like getting the arithmetic means or harmonic mean for both the words. We have observed that although the geometric mean is a very good measure for the collective discrimination, the final similarity value is very small when compared with using the arithmetic means. This is because the many of the files have words which are common in the entire database and their idf values are zero. Since the harmonic mean has a multiplicative factor majority of the A matrix values becomes zero.

The similarity measure is a collective measure of the multiplication of state pair closeness, time pair closeness and the discriminativeness of both the words.
We have tried several approaches while computing the similarity. We will describe below the approaches we have followed and the significant performance gain we have achieved.

Approach 1: Initial Working version

The loop iterates through all the rows of file 1 and file 2 making it an n^2 algorithm. We obtain the word from each file and then convert them to string for getting the idf values of each word and then compute the cell value of the matrix. We have observed that the string conversion for every row in a file would take 0.000094 seconds to execute. Since the majority of the computations are performed in the inner loop. The total time taken to compute the similarity was around 721 seconds.

Approach 2: Significant improvement in Performance

We have observed that majority of time is consumed in converting the word to string and then obtain the idf value of the word. Since the computation is performed in the inner loop, we made a separate string matrix containing the file 1 and file 2 words before the loop execution. This step produced an enormous or drastic performance boost from 721 seconds to 40 seconds.

Pseudo Code

```
    Read the word files file1 and file2
    Get all the unique words and their idf scores.
    Extract the state, time indexes and the words from each file row wise.
    Compute the state pair closeness similarity by computing the shortest path
    Compute the time pair closeness similarity by performing 1 norm distance
    Calculate the average of the idf words being considered
    Calculate the product of the above scores and update the respective cells in A matrix
    The binary vectors w1 and w2 is a bit vector containing all ones.
    Compute the similarity by performing the multiplication of w1*A*w2'
    Print the similarity value.
```

Observations:

- Since the process iterates over each file and individually computes the similarity measure, the usual process of execution is quite high.
- We have also observed that the similarity measure is symmetric
- We have also observed that the similarity of the documents for the same files is not a minimum bound.

- **Task 1g**

  $$sim_{weighted\_avg\_word}(f_1, f_2) = \vec{w_{avg,1}} A (\vec{w_{avg,2}})^T$$

  This task also obtains the similarity measure by constructing the A matrix, however the similarity measure being computed in this task operates on the epidemic word average files.

  Observations:
  - Since the process iterates over each file and individually computes the similarity measure, the usual process of execution is quite high.
  - We have also observed that the similarity measure is symmetric
  - We have also observed that the similarity of the documents for the same files is not a minimum bound.

.

- **Task 1h**

  $$sim_{weighted\_avg\_word}(f_1, f_2) = \vec{w_{diff,1}} A (\vec{w_{diff,2}})^T$$

  This task also obtains the similarity measure by constructing the A matrix, however the similarity measure being computed in this task operates on the epidemic word difference files.

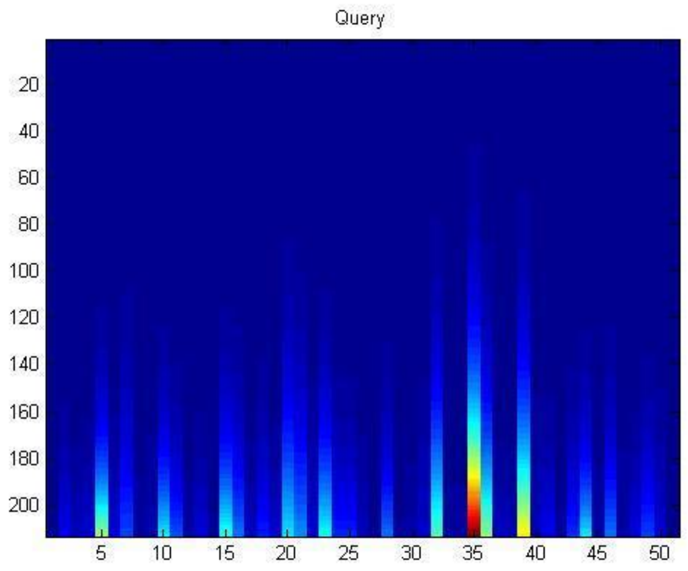
  Observations:
  - The same observations mentioned in task 1g are observed here.
  - In addition we also observe that the similarity score obtained in the previous task and the similarity scores obtained here contain the same value. This is because the same files across their databases reflect the same discrimination.

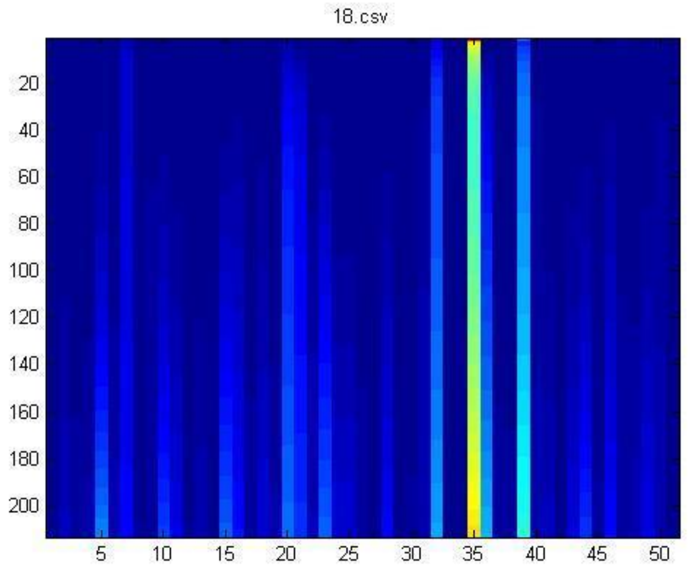
Task 2
In this task we compute the top K similar values with respect to the given query file. The similarity measure used for task 2 is one among the similarity measures in task 1.

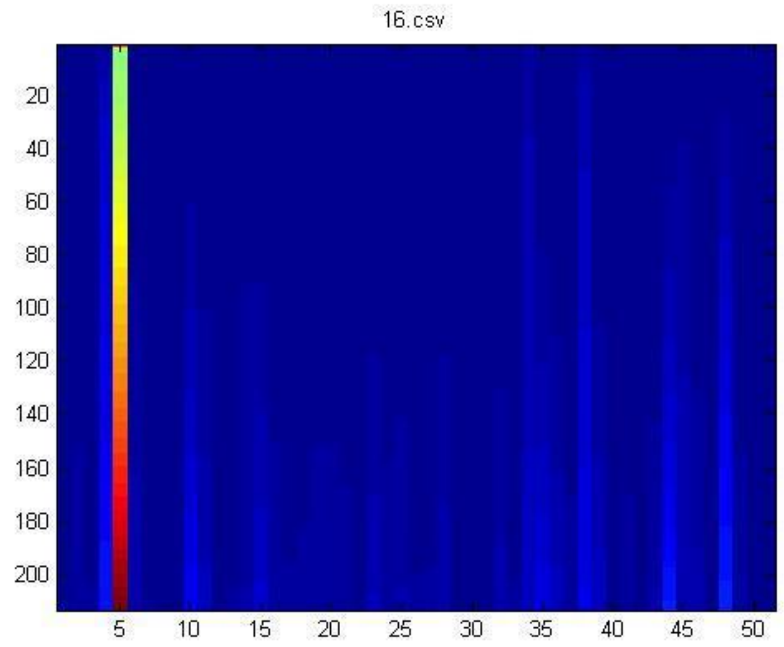
Pseudo Code

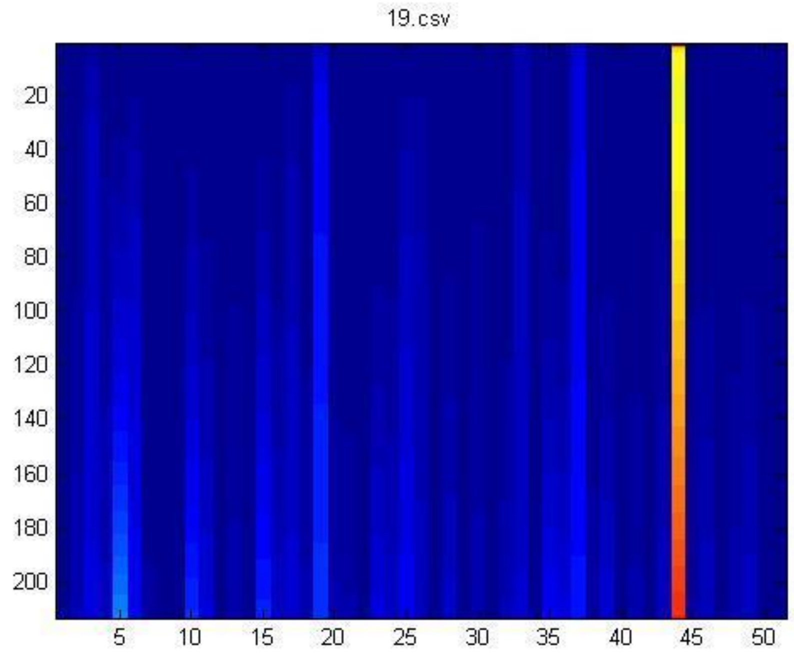
```
Read the query data and the dataset files path and the number of similar files required
For each file in the dataset files
    Compute the similarity between the query and the file
    Store the similarity values in a matrix along with the index for file name.
end
Sort the files in with respect to similarity values in descending order.
Return the top K similar files
```

The sample output for query word and top 3 similar files using euclidean distance is shown below. Here, the X axis represents the state id and y axis represents the simulation value. The top 3 similar files obtained for the query are 18.csv, 16.csv, 19.csv.









Task 3

Task 3 deals with the discovery of latent semantics for the set of input simulation files and also performing nearest neighbor search (set of simulation files) with respect to given query simulation file.

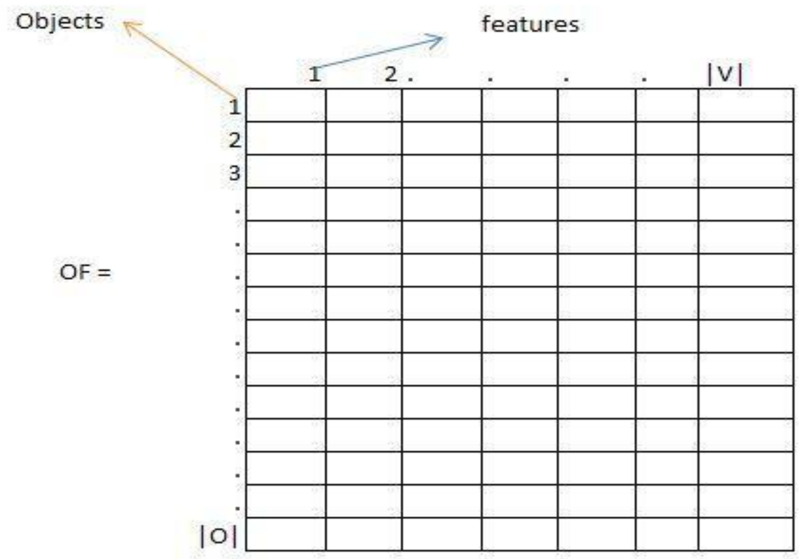
This latent semantic discovery is done in various methods mentioned below from Task 3a - 3c and the nearest neighbor search in Task3d-3f.

- **Task 3a**

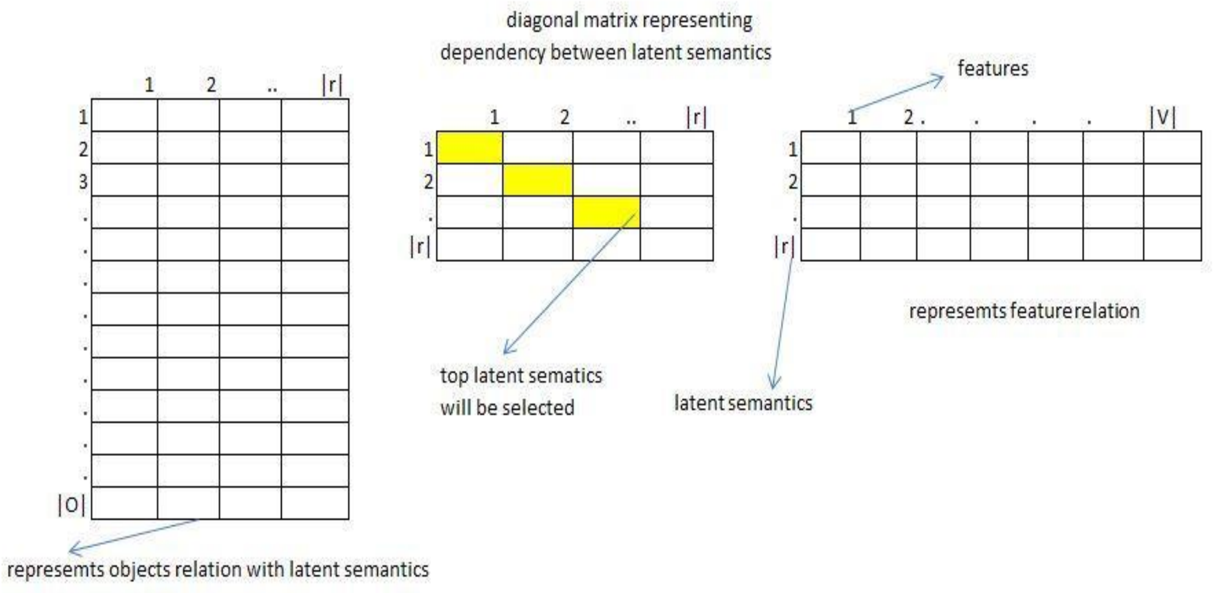
  This task involves identification of latent semantics using Singular value decomposition (SVD). SVD involves eigen decomposition of data matrix, in which each object is represented using set of features. In this task each simulation is considered as an object and to select the initial set of features we have assumed several approaches.

  Assume the initial data matrix is as followed, each row is an object represented using set of features. The data matrix is as follows:

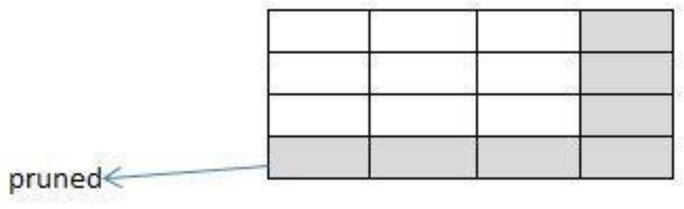
a) Each object is an epidemic word file with 'v' number of features.
b) For the particular file, we considered the binary vector value as the representation of the feature in that particular file.
c) The number of features is unknown for the given data, applying SVD gives us the most important features organized in order.
d) In context of epidemic simulations, we use SVD just to remove extra features considered while plotting data.
e) Also, the eigenvalues that are maximum represent the features to be considered while pruning the features with less eigenvalue.



If we perform eigen decomposition on the above matrix it would result the following matrices.


diagonal matrix representing dependency between latent semantics
features
represemts feature relation
top latent sematics will be selected
latent semantics
represemts objects relation with latent semantics
|O|
|r|
|V|

The matrix O x R is column independent and matrix R x R is diagonal and matrix R x V is row independent. so if we want to preserve the top k latent semantics of r independent semantics, then the R x R matrix is modified as followed.



Suppose, we have the following representation for the SVD function

[U S V] = SVD (Objects X Features)

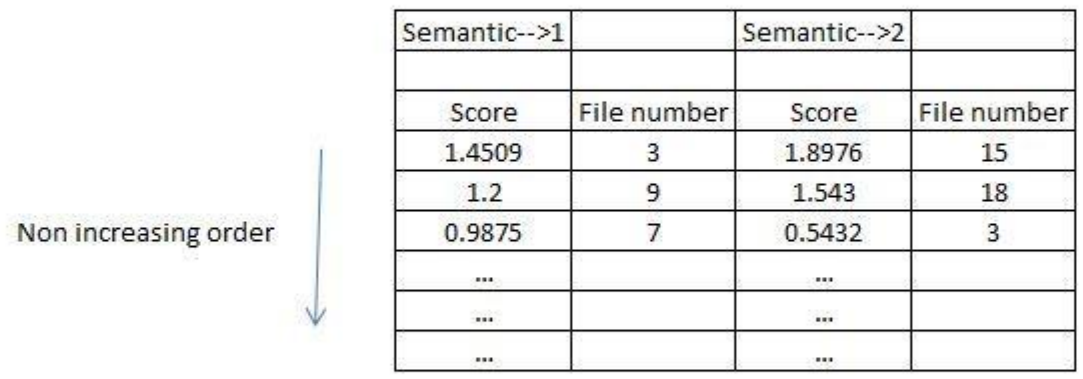
For getting the semantic strength of each object(in our case, epidemic word file), we will have to consider the U matrix. The corresponding row in the U matrix of a file gives its latent semantic score.

S is the eigen valued diagonal matrix. The values in this matrix are in decreasing order and hence when we need to take the top-r semantics, we can directly pick the first r columns of U matrix.

Coming to the implementation of this algorithm, we used the SVD function inbuilt in Matlab for computing the U, S and V matrices. The algorithm of the implementation of this task is as shown below:

1) Read the directory of epidemic word files, number of latent semantics user is interested in and the similarity measure.
2) Construct a HashMap with all the unique words from all the files as keys.
3) Compute the binary vectors of each file based on the unique words that file has.
4) Use the SVD function and give the binary vector matrix as input
5) Based on user's input of the number of latent semantics, display the semantic scores of each file and its score in non-increasing order

The sample output looks something like


Non increasing order

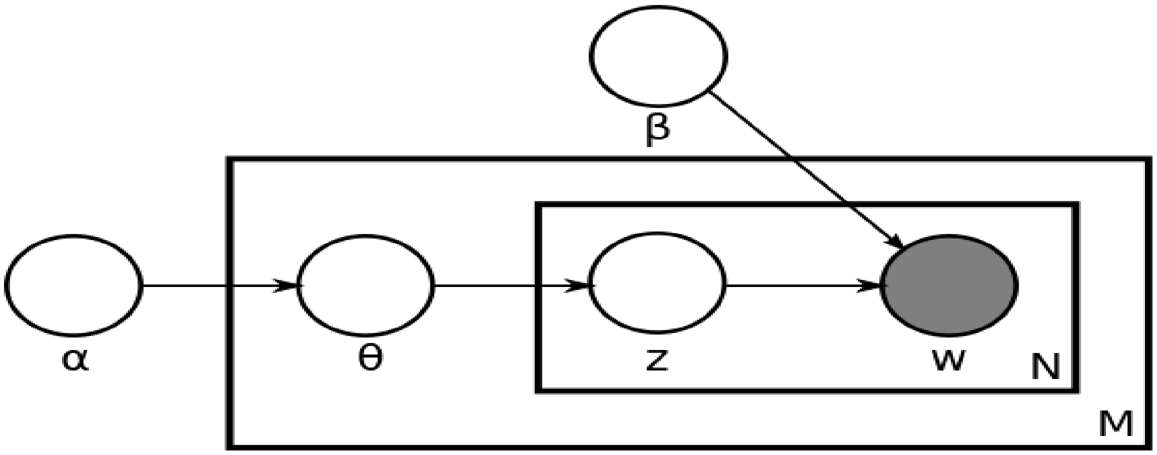
Semantic-->1		Semantic-->2	
Score	File number	Score	File number
1.4509	3	1.8976	15
1.2	9	1.543	18
0.9875	7	0.5432	3
...		...	
...		...	
...		...	

- **Task 3b**

In this task, we are to perform the LDA on the given epidemic word files and display the number of times a pattern in disease is observed in a topic. For implementing the LDA algorithm, computing the topics and scores of topics, we have used the Gibbs Sampling LDA[11].

On a broader picture, the main task of an LDA algorithm is to display the objects in terms of topics and display the features in terms of topics. In the case of epidemic simulations, the objects are the word files of simulation data collected across different timelines. The features being unknown for the given data, LDA computes the topics and their respective scores.

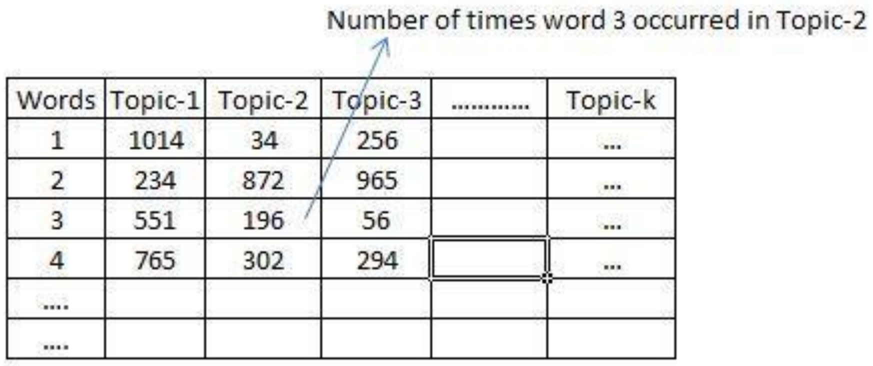
The LDA algorithm looks like:



Given a set of objects and features matrix, a typical LDA algorithm takes in three parameters namely the number of topics, alpha and beta. Number of topics is something that the user gives based on which a Poisson distribution is applied. Alpha and Beta values are for making the continuous distributions to Dirchlet distributions. Ideally,
The alpha value should be 50/Number of topics
Beta value should be 200/ Number of features

Also, we are giving the Term Frequency (TF) of each file and its corresponding words as an input to LDA algorithm, in turn getting the scores of topics.

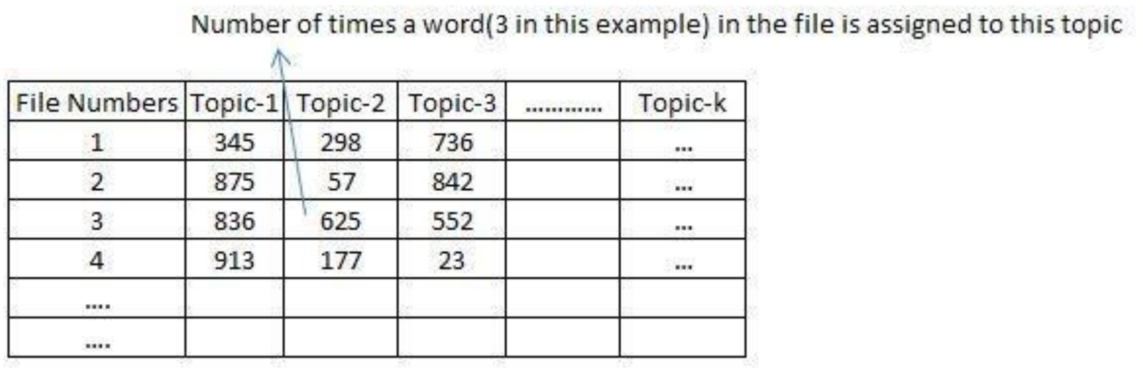
Based on these values, the Gibbs Sampling LDA[11] computes the topics and score of each topic. A typical LDA output for Words and Topics relation looks like:



Number of times word 3 occurred in Topic-2

Words	Topic-1	Topic-2	Topic-3	…………	Topic-k
1	1014	34	256		…
2	234	872	965		…
3	551	196	56		…
4	765	302	294		…
….					
….					

Also, as described above, the documents and topics matrix looks like


Number of times a word(3 in this example) in the file is assigned to this topic

File Numbers	Topic-1	Topic-2	Topic-3	…………	Topic-k
1	345	298	736		…
2	875	57	842		…
3	836	625	552		…
4	913	177	23		…
….					
….					

After we get these matrices, based on user's input we prune the least interesting topics and provide the user with the first 'r' topics.

Coming to the implementation of this task, find the algorithm below

1) Read the directory path where the epidemic word files are present and the number of topics the user is interested in.
2) Create a HashMap with all the unique words from all the files in the path provided by the user.
3) Compute the binary vectors of each file based on the unique words that file has.
4) Now compute the Term Frequency (TF) of each word in a particular file.
5) As we are using the Gibbs Sampling LDA[11], we have the following inputs and outputs to work on. Please find below a detailed explanation of what each field corresponds to, in the algorithm.

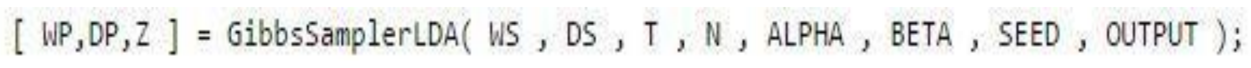
Inputs

a) Bag of words (In our case, the words from the epidemic word files)
b) Number of times that particular sequence of epidemic condition occurred in the file (TF in our case)

Outputs

a) Topic assignments to each keyword token
b) Counts of the number of times each word is assigned to a topic
c) Number of times each topic is assigned to a file

The function used to implement LDA looks like



```
[ WP,DP,Z ] = GibbsSamplerLDA( WS , DS , T , N , ALPHA , BETA , SEED , OUTPUT );
```

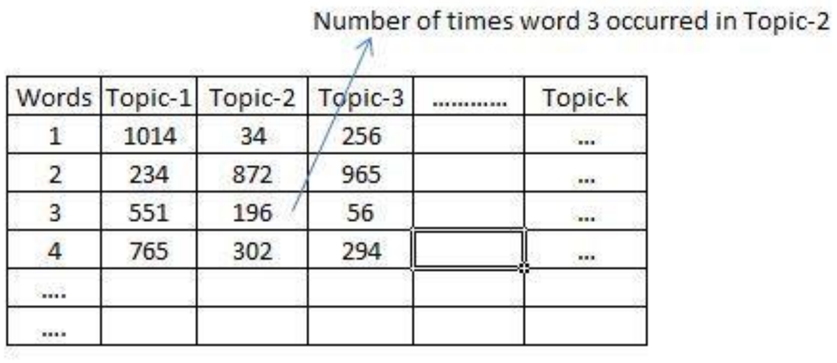
Let us now see the illustration of each field in this implementation:

Inputs:

WS → A 1 X N matrix of vectors which has the word and its corresponding number of word tokens
T → Number of topics
N → Number of iterations
Beta → 200 / Number of Features
Alpha → 50 / Number of topics
Seed → Random number generator
Output → 1 - No output, 2 - Iteration number, 3 - Show all output

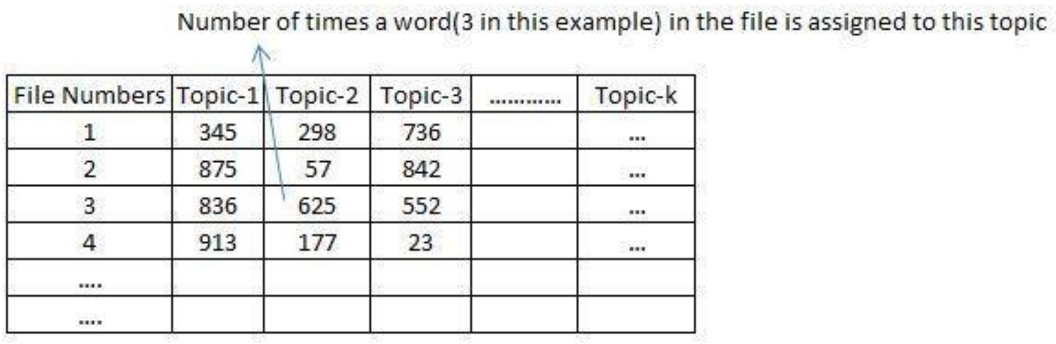
Outputs:

WP matrix:


Number of times word 3 occurred in Topic-2

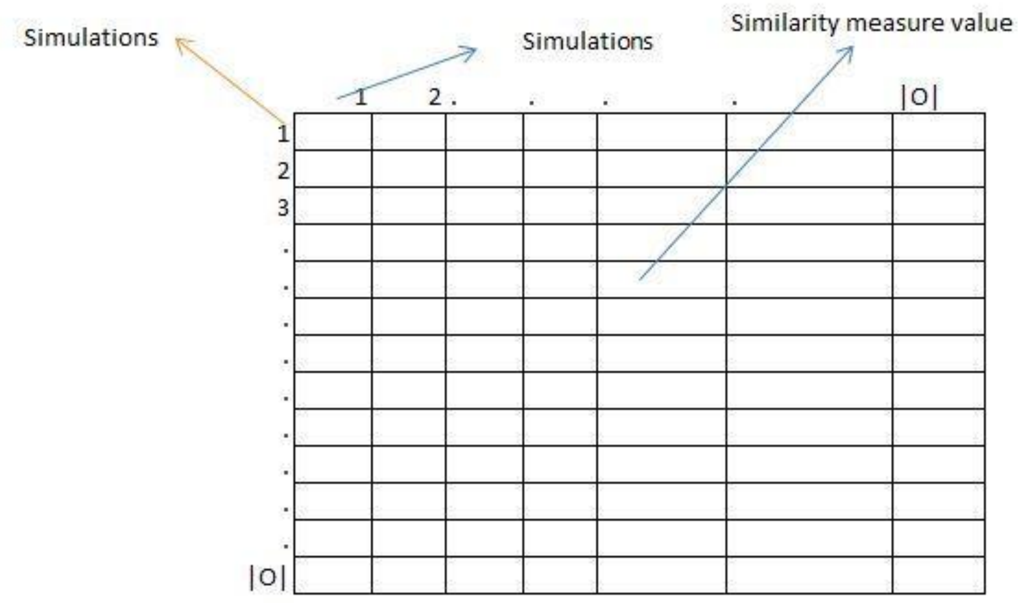
Words	Topic-1	Topic-2	Topic-3	....	Topic-k
1	1014	34	256		...
2	234	872	965		...
3	551	196	56		...
4	765	302	294		...
....					
....					

DP matrix:


Number of times a word(3 in this example) in the file is assigned to this topic

File Numbers	Topic-1	Topic-2	Topic-3	....	Topic-k
1	345	298	736		...
2	875	57	842		...
3	836	625	552		...
4	913	177	23		...
....					
....					

- **Task 3c**
  In this task, we have to implement the SVD algorithm on simulation-simulation similarity matrix instead of the traditional Object-Feature matrix. We are asked to take the similarity measures computed in the 1st question as input and make a simulation-simulation similarity matrix. Then, this matrix is given as an input to SVD algorithm.



After a matrix of this specification is computed, we will have to pass this matrix as input to SVD algorithm as

[U S V] = SVD (Simulations X Simulations)

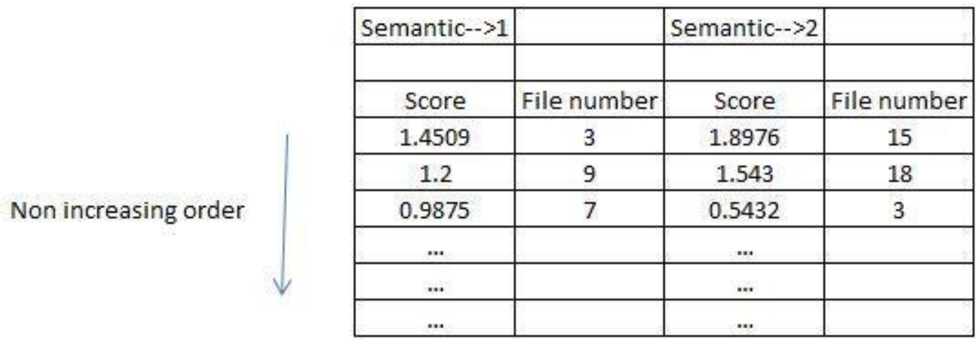
This implementation of SVD algorithm on simulation-simulation is similar to PCA algorithm except for PCA taking covariance matrix and here we are taking the similarity measure as the value.

The U matrix computed consists of the simulation and its similarity with the latent semantic. As S contains the eigenvalues in decreasing order, we prune the eigenvalues based on user's input

Coming to the implementation of this algorithm, we used the SVD function inbuilt in Matlab for computing the U, S and V matrices. The algorithm of the implementation of this task is as shown below:

1) Read the directory of epidemic word files and number of latent semantics user is interested in.
2) Construct a simulation-simulation similarity matrix based on user input of similarity measure
3) Pass this file as an input to the SVD function
4) Based on user's input of the number of latent semantics, display the semantic scores of each file and its score in non-increasing order

The sample output looks something like


Non increasing order

Semantic-->1		Semantic-->2	
Score	File number	Score	File number
1.4509	3	1.8976	15
1.2	9	1.543	18
0.9875	7	0.5432	3
...		...	
...		...	
...		...	

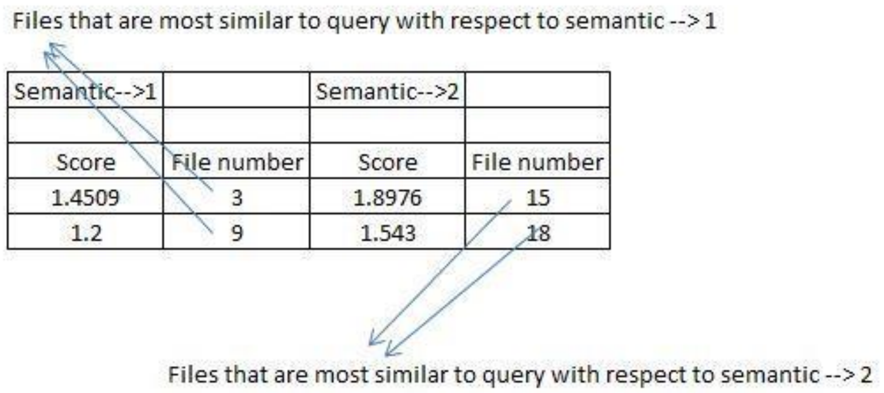
- **Task 3d**
  This task is similar to Task 3A except that the set of simulation files also includes a query file based on which we give the k nearest neighbors after computing the latent semantic scores of all the files.

  As this is similar to Incremental SVD, our first approach is to use the same set of object-latent semantics computed to all the simulation files prior in the Task 3a. But, if the query file that is coming in is not having the same set of features, then we cannot go ahead with Incremental SVD. So, we have opted an approach where we compute the SVD to all the simulation files and the query file all over again and then take the r latent semantics and their scores.

  The algorithm for this task is very similar to Task 3a except for getting the k most similar files as the query file. Find below the algorithm as to how we computed the k-most similar files to the query file.

  1) Get all the latent semantic scores of all the semantic files and the query file
  2) Take the score of query file and subtract all the scores from all the files
  3) Compute the absolute values of all the received values and sort them in increasing order.
  4) The first k files are the files which are the k-nearest neighbors of query file.

  The output of this task looks like



Files that are most similar to query with respect to semantic --> 1

Semantic-->1		Semantic-->2	
Score	File number	Score	File number
1.4509	3	1.8976	15
1.2	9	1.543	18

Files that are most similar to query with respect to semantic --> 2

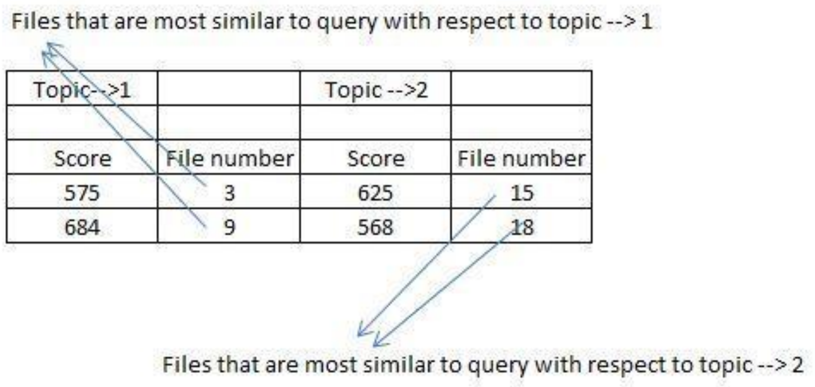
- **Task 3e**

  This task is similar to Task 3B except for the output. This task gives the k nearest neighbors of the query file in terms of scores of topics in the particular file. For getting the scores of the query file, we are giving all the word files along with the query files. The reason for implementing the whole algorithm is that the query file might have a set of new topics.

  The algorithm for this task is all same except for the computation of k nearest neighbors of query which is illustrated below:

  1) Get the scores of all the topics of simulation files and query file
  2) Take the score of query file and subtract the scores of all the other files
  3) Compute the absolute values of all the received values and sort them in increasing order.
  4) The first k files are the files which are the k-nearest neighbors of query file.

  The output of this task looks like



Files that are most similar to query with respect to topic --> 1

Topic-->1		Topic -->2	
Score	File number	Score	File number
575	3	625	15
684	9	568	18

Files that are most similar to query with respect to topic --> 2

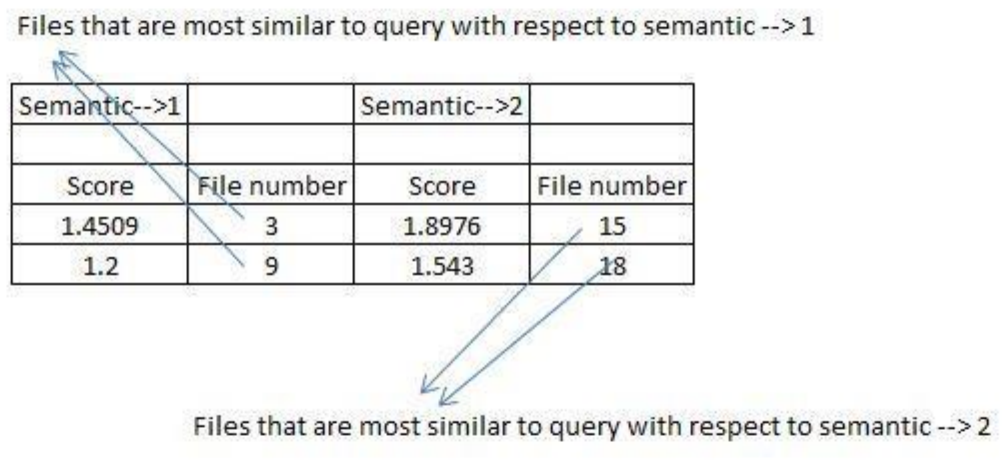
- **Task 3f**

  This task is similar to Task 3c except for the result depicting the k nearest neighbors of a query file the user gives. This is based on the scores of semantic values obtained after SVD is implemented on all the files and top r semantics are picked.

  The algorithm for this task is all same except for computing the k nearest neighbors of the query file provided. The steps for accomplishing the same are

  1) Get all the latent semantic scores of all the semantic files and the query file
  2) Take the score of query file and subtract all the scores from all the files
  3) Compute the absolute values of all the received values and sort them in increasing order.
  4) The first k files are the files which are the k-nearest neighbors of query file.

  The sample output looks like:



Files that are most similar to query with respect to semantic --> 1

Semantic-->1		Semantic-->2	
Score	File number	Score	File number
1.4509	3	1.8976	15
1.2	9	1.543	18

Files that are most similar to query with respect to semantic --> 2

Task 4:

Description of the Solution:

This task implements the FastMap algorithm proposed by Christos Faloutsos and King-Ip (David) Lin [2]. This task mainly consists of two parts. They are :

- **Task 4a:**

  Computation of Distances in Original Space:

  Fast map requires distances between objects in original space as input. The distances between the objects in the original space are computed based on the similarity metric choice given by the user. Depending on the similarity metric we construct an N x N matrix that contains the distances between objects in original space.

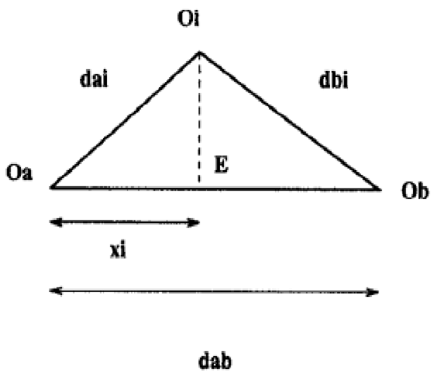
  Mapping the data Objects on to new space [2]:

The core part of this algorithm is to select r-dimensions depending on pivot objects for each dimension. This modification from the MDS made the FastMap algorithm efficient in terms of data objects mapping complexity and query processing complexity [2].

Step 1: Each dimension is a line joining the two pivot objects corresponding to that dimension. Pivot objects are selected as followed [2].

- Given set of objects and distances among the objects, select an object randomly from the given set of objects and assume it as Ob
- Select the farthest object from Ob and label it as pivot object Oa.
- Now again select the farthest object from Oa and mark it as pivot object Ob. Return the objects Oa and Ob as pivot objects.

Step 2: After getting the dimension along Oa and Ob, all the objects are projected on to this dimension, to get the representation of the objects along this dimension. The projection of objects on to the line(dimension) uses cosine law in a triangle. Given the distance between pivot objects (dab) and distance between an Object (Oi) and the pivot objects (say dai, dbi), we want the projection (xi)of the object Oi onto the line Oa-Ob [2].



In the triangle (OaOiOb), According to cosine law [2]

$$d_{b,i}^2 = d_{a,i}^2 + d_{a,b}^2 - 2x_i d_{a,b}$$

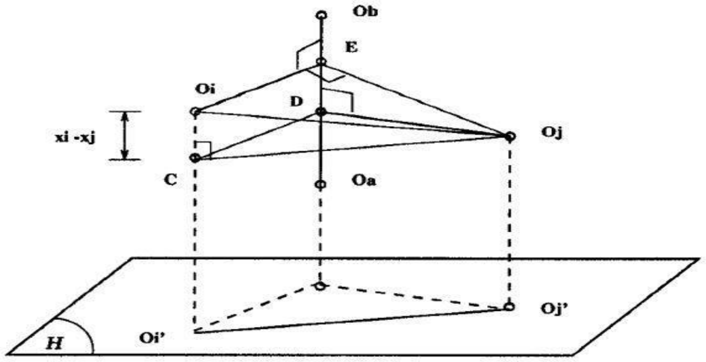
….. *Equation 1*

From the above equation Xi (projection of the object on line OaOb) can be written as

$$x_i = \frac{d_{a,i}^2 + d_{a,b}^2 - d_{b,i}^2}{2d_{a,b}}$$

….. *Equation 2*

Step 3: To extend the concept of projecting objects onto a line, a perpendicular hyper plane 'H' to the line OaOb is considered [2].



On the Hyperplane 'H' can be calculated using the formula

$$(\mathcal{D}'(O_i', O_j'))^2 = (\mathcal{D}(O_i, O_j))^2 - (x_i - x_j)^2 \quad i, j = 1, \ldots, N$$ ….. *Equation 3*

Where
D`(Oi, Oj) = distance between the objects Oi, Oj on the hyperplane
D(Oi, Oj) = original distance between the objects Oi, Oj
xi = projection of object Oi on to the line OaOb
xj = projection of object Oj on to the line OaOb

Step 4: For each of r - dimensions, the above process is continued and in each iteration pivot objects are recorded to facilitate the query processing [2].

- **Task 4b:**
  Query Processing [2]**:**
  To process the query we need to get a unique representation of both query object and data objects. So we need to map the query from unknown space into the new r-dimensional space. This can be done by repeating STEP 2 for the query object and pivot objects of each dimension. Now to execute the range query or nearest neighbor query we just need to compute the euclidean distance between the query and data objects and return the results according to user request.

  Pseudo Code [2]**:**

  Here is a pseudo code to reduce the data objects using FastMap Algorithm.

```
1) global array to store pivot objects for each dimension.
2) global array to store the objects representation in the new dimensions.
3) Read all the epidemic simulation files and compute the distances between
   each pair of the simulation files in original space.
4) FastMap (numberOfDimensions, Objects)
   if(numberOfDimensions<=0)
           return;
   else
```

```
            select pivot objects and record them in the global array;
            if distance between the pivot objects is 0,
                then current dimension value of all objects become 0;
            else
                compute the projection of the objects on the pivot objects
                line using cosine law and store them in the global array.
        endif;
    4) repeat the process for FastMap(numberOfDimensions-1, O) by
       considering the distances between the objects in the hyperplane.
```

Pseudo Code for Pivot Objects:

```
Choose a random simulation file from the set of simulation files
Compute the distance from random pivot to all the objects
First Pivot = Choose the object which is maximum distance from the random pivot
Second Pivot = Choose the object which is maximum distance from the first pivot
```

Pseudo Code for Query Computation:

```
Map the query to new space by projecting the object on the pivot dimensions
Compute the euclidean distance between the query and the objects represented in new
dimensions
Sort the distance values
Return top K similar objects
```

Relevance to the project:

FastMap is useful when we don't have a clear idea about the initial feature set in the data set of objects. Given the epidemic simulation files, if we are unable to identify the initial feature set then we can use FastMap algorithm to map the data objects on to the new r-dimensional space for efficient retrieval and visualization. Even though we don't know anything about the original space and original features, we are extracting dimensions from the data to help the efficient query processing.

Mapping Error:

Mapping error is defined as the relative change of distances in original space to distances in new space.

Mapping Error for a pair of objects in our task is computed using the below formula

- Error (obj_i, obj_j) = (distanceInOriginalSpace (obj_i, obj_j) - distanceInReducedSpace (obj_i, obj_j)) / distanceInOriginalSpace (obj_i, obj_j)
- Mapping Error = Sum all errors for objects

Other mapping error functions considered

- We have considered the below mapping error function.

    - Mapping Error = sum(Error (obj_i, obj_j)
    - Error (obj_i, obj_j) = (distanceInOriginalSpace (obj_i, obj_j) - distanceInReducedSpace (obj_i, obj_j))

However, we got better results when we considered the relative distance between objects.

Distance to Similarity Function
In fastmap computation we need to provide distances between objects in original space as input. Since the distances are inversely proportional to distances, we have considered the following functions to convert similarity to distances.

- Distance = 1 / (similarity + epsilon). Distances are inversely proportional to similarity, we have used this formula. However, we get 0 similarities when computing binary vector similarity for the word, average and difference files. To avoid infinity values, we have used as epsilon value of 0.000001.
- Distance = 1 / (similarity). This distance function is used in case of euclidean and DTW similarity conversion.

Other Distance Measures Considered
We have considered the following distance measures as well.

- Distance = 1 / $e^{Similarity}$. However we observed that for small changes in similarity there is exponential difference in distance.
- Distance = (1 / Similarity) - 1. We observed that this distance function generates negative values in case of binary vector similarity for word, average and difference files.

Observations:

- As the number of dimensions increases in fastmap computation, the mapping error has decreased.
- We have observed both underestimation and over estimation of distances.

Interface specifications

We used MATLAB for implementation of phase of project. As we are dealing with large data sets and Matrix constructions, Matlab provides inbuilt functions for matrix operations and hence will be a better solution to deal with.

- **Task 1a**
  getEuclideanSimilarity.m: This function takes computes the euclidean similarity which takes two epidemic simulation file path f1 and as input and outputs the similarity value.

  Input: Directory of the files, Epidemic Simulation files 1 and 2

Output: Euclidean similarity measure between input simulation files 1 and 2

- **Task 1b**
  getDTWSimilarity.m:_ This function computes the DTW similarity. It takes two simulation files as input and outputs the DTW similarity score. This file will aggregate all the state similarities received from getDTWDistance.
  getDTWDistance.m:_ This function computes the DTW distance between two state simulations. It takes two time series as input and outputs the DTW distance score.

  Input: Directory of the files, Epidemic Simulation files 1 and 2
  Output: DTW similarity measure between input simulation files 1 and 2

  Tasks 1c-1d uses the function **getSimilarityMeasureforWindows.m ;** this function computes the similarity word measure for given epidemic simulation files (internally using the word files, averages word files, difference word files)

- **Task 1c**
  Proj2_1c.m:_ This function reads the input directory, file1 and file2 and passes the full path name of the files to the function getSimilarityMeasureforWindows.m
  Input: Directory of the files, Epidemic Simulation files 1 and 2 *(uses the Epidemic word files corresponding to the Simulation files)*
  Output: Similarity word measure between input simulation files 1 and 2

- **Task 1d**

  Proj2_1d.m:_ This function reads the input directory, file1 and file2 and passes the full path name of the files to the function getSimilarityMeasureforWindows.m
  Input: Directory of the files, Epidemic Simulation files 1 and 2 *(uses the Epidemic average word files corresponding to the Simulation files)*
  Output: Similarity average word measure between input simulation files 1 and 2

- **Task 1e**
  Proj2_1e.m: This function reads the input directory, file1 and file2 and passes the full path name of the files to the function getSimilarityMeasureforWindows.m

  Input: Directory of the files, Epidemic Simulation files 1 and 2 *(uses the Epidemic difference word files corresponding to the Simulation files)*
  Output: Similarity difference word measure between input simulation files 1 and 2

- **Task 1f**
  getTermIDF.m: This function calculates the IDF values for each unique word in the database of epidemic files given as input.
  Input: Directory of the files containing the epidemic word files
  Output: idfArray containing the idf values of all the unique words in the database and uniqeWords containing the list of all unique words.

  Proj2_1: This function computes the similarity measure by constructing the A matrix by taking the epidemic word files, the connectivity graph and the output from the getTermIDF functions.
  Input: Epidemic word files file 1 and file 2 , the output from getTermIDF
  Output: similarity measure obtained.

- **Task 1g**
  getTermIDF.m: This function calculates the IDF values for each unique word in the database of epidemic average files given as input.
  Input: Directory of the files containing the epidemic average word files
  Output: idfArray containing the idf values of all the unique epidemic average words in the database and uniqeWords matrix containing the list of all unique words.

  Proj2_1f.m: This function computes the similarity measure by constructing the A matrix by taking the epidemic average word files, the connectivity graph and the output from the getTermIDF functions.
  Input: Epidemic average word files file 1 and file 2 , the output from getTermIDF
  Output: similarity measure obtained.

- **Task 1h**
  getTermIDF.m: This function calculates the IDF values for each unique word in the database of epidemic difference files given as input.
  Input: Directory of the files containing the epidemic difference word files
  Output: idfArray containing the idf values of all the unique words in the database and uniqeWords matrix containing the list of all unique words.

  Proj2_1f.m: This function computes the similarity measure by constructing the A matrix by taking the epidemic difference word files, the connectivity graph and the output from the getTermIDF functions.
  Input: Epidemic difference word files file 1 and file 2 , the output from getTermIDF
  Output: similarity measure obtained.

- **Task 2**
  getChoiceSimulationSimilarity.m: This function takes the query path, simulation file path and the similarity choice given as input by the user. This function will call the respective similarity function to compute the similarity between the query and the simulation file.
  getSimilarSimulationFiles.m: This function takes the input from the user and aggregates all the similarities with respect to query and the simulation files obtained from getChoiceSimulationSimilarity.m function.

  Input: Query file path (newSimFilePath), directory of simulation files (datasetDir), no of similar simulation files required (FileCount), similarity measure choice by the user (similarityMeasureChoice).
  Output: Top K similar files name with the respective similarity value.

- **Task 3a**
  Proj2_3a.m: This function takes the directory path of epidemic word files and number of latent semantics
  Input: Directory of the input Epidemic Simulation files *(uses the Epidemic word files corresponding to the Simulation files),* the latent semantics r, Object Feature Matrix computed from the Epidemic word files.

  Output: List of <simulation, score> for each semantic listed in the non-increasing order.

- **Task 3b**
  Task3B.m: This function takes the directory path of epidemic word files and number of latent semantics

  Input: Directory of the input Epidemic Simulation files *(uses the Epidemic word files corresponding to the Simulation files),* the latent semantics r, Object Feature Matrix computed from the Epidemic word files with each cell value being the frequency of a word in the epidemic word file. Uses the external dll files of GibbsSampling

  Output: List of <simulation, score> for each semantic listed in the non-increasing order

- **Task 3c**
  Proj2_3c.m: This function takes the directory path of epidemic word files, number of latent semantics and a similarity measure. It internally uses getEuclideanSimilarity.m, getDTWSimilarity.m, getDTWDistance.m, getSimilarityMeasureforWindows.m, Proj2_1f.m and getChoiceSimulationSimilarity.m

Input: Directory of the input Epidemic Simulation files *(uses the Epidemic word files corresponding to the Simulation files),* the latent semantics r, a similarity measure, and Simulation- Simulation similarity matrix based on given similarity measure.

Output: List of <simulation, score> for each semantic listed in the non-increasing order

- **Task 3d**
  Proj2_3d.m: This function takes the directory path of epidemic word files, number of latent semantics, number of nearest neighbor's k, directory path of the query word file.

  Input: Directory of the input Epidemic Simulation files *(uses the Epidemic word files corresponding to the Simulation files),* the latent semantics r, query file directory *(uses the query word file from this directory)* number of nearest neighbors k, Object Feature Matrix computed from the Epidemic word files.

  Output: List of k most similar simulations to the query file for each semantic listed in the non-increasing order with the format <simulation, score>.

- **Task 3e**
  Proj2_3e.m: This function takes the directory path of epidemic word files and number of latent semantics**.**

  Input: Directory of the input Epidemic Simulation files *(uses the Epidemic word files corresponding to the Simulation files),* the latent semantics r, query file directory *(uses the query word file from this directory)* number of nearest neighbors k, Object Feature Matrix computed from the Epidemic word files with each cell value being the frequency of a word in the epidemic word file.

  Output: List of k most similar simulations to the query file for each semantic listed in the non-increasing order with the format <simulation, score>.

- **Task 3f**
  Proj2_3f.m: This function takes the directory path of epidemic word files, number of latent semantics, similarity measure, number of nearest neighbor's k, directory path of the query word file. It internally uses getEuclideanSimilarity.m, getDTWSimilarity.m, getDTWDistance.m, getSimilarityMeasureforWindows.m, Proj2_1f.m and getChoiceSimulationSimilarity.m

  Input: Directory of the input Epidemic Simulation files *(uses the Epidemic word files corresponding to the Simulation files),* the latent semantics r,query file directory *(uses the query file simulation/word/average/difference from this directory)* number of nearest neighbors k, a similarity measure, Simulation- Simulation similarity matrix based on given similarity measure.

Output: List of k most similar simulations to the query file for each semantic listed in the non-increasing order with the format <simulation, score>.

- **Task 4a**
  Main.m: This script file is used to call all the functions related to fast map computation. It takes the inputs from user and performs the computation.
  getFastMapReducedSpace.m: This function is the master file for this task. It does the entire fast map computation and outputs the mapping error. This function internally has the logic to compute distances in original space, distances in reduced space and to compute the mapping error between objects.
  getDistanceFromSimilarity: This function calculates the distance from similarity.
  getPivotObjects.m: This function computes the pivot objects that form the required dimensions.
  getFastMapSimilarSimulations

  Input: Director for input files (datasetDir), number of dimension to reduce the query (reducedDimensions), similarity measure (similarityMeasureChoice)
  Output: return the mapping error

- **Task 4b**
  getChoiceSimulationSimilarity.m: This function takes the query path, simulation file path and the similarity choice inputed by the user. This function will call the respective similarity function to compute the similarity between the query and the simulation file.
  getFastMapSimilarSimulations.m: This function maps the query object to new space and gets the top K similar files with respect to query.

  Input: Path of the query (newSimulationFilePath), number of dimension to reduce the query (reducedDimensions) , number of similar files required (similarFileRequired)
  Output: Returns the top K similar files with respect to query.

System requirements

- This phase is developed in Matlab in Windows 7 Operating System and is compatible to run with MATLAB 2013b version.

Installation and execution instructions

- For the project to run, MATLAB should be installed and opened.

- Load all the Matlab scripts into the MATLAB folder of the system.

Find below the execution instructions for all the tasks:

- **Task 1a**
```
>> Proj2_1a
Input the folder for input data files: C:\Users\sgaripal\Documents\MATLAB\Input
Please enter a epidemic simulation file number:1
Please enter a epidemic simulation file number:2
The similarity between epidemic simulation files1 and2 -->1.9432e-05
```

- **Task 1b**
```
>> Proj2_1b
Input the folder for input data files: C:\Users\sgaripal\Documents\MATLAB\Input
Please enter a epidemic simulation file number:1
Please enter a epidemic simulation file number:2
Elapsed time is 5.636364 seconds.
The similarity between epidemic simulation files1 and2 -->2.2472e-09
```

- **Task 1c**
```
>> Proj2_1c
Input the folder for input data files: C:\Users\sgaripal\Documents\MATLAB\Input
Please enter a epidemic word file number:1
Please enter a epidemic word file number:2
The similarity between epidemic word files1 and2 -->11
```

- **Task 1d**
```
>> Proj2_1d
Input the folder for input data files: C:\Users\sgaripal\Documents\MATLAB\Input
Please enter a average file number:1
Please enter a average file number:2
The similarity between epidemic average files1 and2 -->5
```

- **Task 1e**
```
>> Proj2_1e
Input the folder for input data files: C:\Users\sgaripal\Documents\MATLAB\Input
Please enter a difference file number:1
Please enter a difference file number:2
The similarity between epidemic difference files1 and2 -->5
```

- **Task 1f**

```
>> Proj2_1f
Input the folder for input data files:
C:\Users\pdadi\Downloads\SampleData_P2\SampleData_P2\Set_of_Simulation_Files\wo
rd
Enter the input for File - 1 15.csv
Enter the input for File - 2 11.csv
Enter the connectivity Graph location
C:\Users\pdadi\Downloads\sampledata_P1_F14\sampledata_P1_F14\Graphs\LocationMa
trix.xlsx
The final similarity is 297.581493
```

- **Task 1g**

```
>> Proj2_1f
Input the folder for input data files:
C:\Users\pdadi\Downloads\SampleData_P2\SampleData_P2\Set_of_Simulation_Files\av
erage
Enter the input for File - 1 18.csv
Enter the input for File - 2 19.csv
Enter the connectivity Graph location
C:\Users\pdadi\Downloads\sampledata_P1_F14\sampledata_P1_F14\Graphs\LocationMa
trix.xlsx
The final similarity is 22551.61811
```

- **Task 1h**

```
>> Proj2_1f
Input the folder for input data files:
C:\Users\pdadi\Downloads\SampleData_P2\SampleData_P2\Set_of_Simulation_Files\dif
ference
Enter the input for File - 1 4.csv
Enter the input for File - 2 4.csv
Enter the connectivity Graph location
C:\Users\pdadi\Downloads\sampledata_P1_F14\sampledata_P1_F14\Graphs\LocationMa
trix.xlsx
The final similarity is 7981.763769
```

- **Task 2**

```
>> Proj2_2
Please enter a query file
path:C:\Users\pdadi\Desktop\TestExecution\Query_Simulation_File\query.csv
Please enter a root directory path: C:\Users\pdadi\Desktop\TestExecution\Input
Please enter the number of similar files: 3
Select the similarity measure
Enter a - Euclidean Distance
Enter b - Dynamic Time Warping Distance
Enter c - Word File Similarity without A Function
Enter d - Average File Similarity without A Function
Enter e - Difference File Similarity without A Function
Enter f - Word File Similarity with A Function
Enter g - Average File Similarity with A Function
Enter h - Difference File Similarity with A Function
Enter the choice of your similarity : a
4.31336005283700e-05    18
4.30620579459438e-05    16
4.30484819228808e-05    19
```

- **Task 3a**

```
 >> Proj2_3a
Please enter a directory:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Set_of_Simulation_Files\word
Please enter the value of r:4
```

- **Task 3b**

```
 >> Task3B
Input the folder for input data files:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Set_of_Simulation_Files\word
Input the value for r: 4
Importing word document counts from:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Set_of_Simulation_Files\word/i
nput.txt
Running LDA Gibbs Sampler Version 1.0
Arguments:
        Number of words       W = 37
        Number of docs        D = 20
        Number of topics      T = 4
        Number of iterations N = 500
        Hyperparameter   ALPHA = 12.5000
        Hyperparameter        BETA = 5.4054
        Seed number           = 3
        Number of tokens      = 42840
Determining random order update sequence
        Iteration 0 of 500
        Iteration 10 of 500
        ……..
        Iteration 490 of 500
```

- **Task 3c**

```
 >> Proj2_3c
Please enter the directory for epidemic simulation
files:C:\Users\pdadi\Desktop\TestExecution\Input
Please enter a value for number of latent semantics (r) :4
Select the similarity measure
Enter a - Euclidean Distance
Enter b - Dynamic Time Wrapping Distance
Enter c - Word File Similarity without A Function
Enter d - Average File Similarity without A Function
Enter e - Difference File Similarity without A Function
Enter f - Word File Similarity with A Function
Enter g - Average File Similarity with A Function
Enter h - Difference File Similarity with A Function
Enter the choice of your similarity : e
```

- **Task 3d**

```
 >> Proj2_3d
Please enter a directory:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Set_of_Simulation_Files\word
Please enter the value of r:4
Please enter a directory for query file:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Query_Simulation_File\word
Please enter the value of k(number of nearest neighbors):3
```

- **Task 3e**

```
>> Proj2_3e
Input the folder for input data files:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Set_of_Simulation_Files\word
Input the value for r: 4
Please enter a directory for query file:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Query_Simulation_File\word
Please enter the value of k(number of nearest neighbors):3

C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Query_Simulation_File\word\*.
csv
```

```
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Query_Simulation_File\word\qu
ery-word.csv
query-word.csv
Importing word document counts from:
C:\Users\sgaripal\Downloads\SampleData_P2_Sampath\Set_of_Simulation_Files\word/i
nput1.txt
Running LDA Gibbs Sampler Version 1.0
Arguments:
        Number of words      W = 51
        Number of docs       D = 21
        Number of topics     T = 4
        Number of iterations N = 500
        Hyperparameter   ALPHA = 12.5000
        Hyperparameter    BETA = 3.9216
        Seed number            = 3
        Number of tokens       = 44982
Internal Memory Allocation
        w,d,z,order indices combined = 719712 bytes
        wp (full) matrix = 816 bytes
        dp (full) matrix = 336 bytes
Starting Random initialization
Determining random order update sequence
        Iteration 0 of 500
        ……...
        Iteration 490 of 500
```

- **Task 3f**

```
> > Proj2_3f
Please enter the directory for epidemic simulation
files:C:\Users\pdadi\Desktop\TestExecution\Input
Please enter a value for number of latent semantics (r) :4
Please enter a directory for query file:
C:\Users\pdadi\Desktop\TestExecution\Query_Simulation_File
Please enter the value of k(number of nearest neighbors):2
Select the similarity measure
Enter a - Ecludean Distance
Enter b - Dynamic Time Wrapping Distance
Enter c - Word File Similarity without A Function
Enter d - Average File Similarity without A Function
Enter e - Difference File Similarity without A Function
Enter f - Word File Similarity with A Function
Enter g - Average File Similarity with A Function
Enter h - Difference File Similarity with A Function
Enter the choice of your similarity : a
```

- **Task 4a**

```
Enter the directory of datasets : C:\Users\pdadi\Desktop\TestExecution\Input\word
Select the similarity measure
Enter a - Ecludean Distance
Enter b - Dynamic Time Wrapping Distance
Enter c - Word File Similarity without A Function
Enter d - Average File Similarity without A Function
Enter e - Difference File Similarity without A Function
Enter f - Word File Similarity with A Function
Enter g - Average File Similarity with A Function
Enter h - Difference File Similarity with A Function
Enter the choice of your similarity : c
Enter the number of dimensions to reduce : 5

Mapping Error is :88.3847
```

- **Task 4b**

```
Enter the path of query file :
C:\Users\pdadi\Desktop\TestExecution\Query_Simulation_File\word\query-word.csv
Enter the number of dimensions to reduce : 5
Enter the number of similar files required : 3
2.2361e-06->11.csv
2.2361e-06->12.csv
2.2361e-06->13.csv
```

Related work

In the paper “Spatio-temporal Analysis on Enterovirus Cases through Integrated Surveillance in Taiwan”[9], an analysis is performed to understand the characteristics of mild and severe enterovirus(EV) cases through integrated surveillance data. They have analyzed the clusters of severe enterovirus outbreaks using various spatial temporal statistics. The results found out were that the mild EV cases are highly correlated to the severe EV cases occurring in the same week.

Conclusions

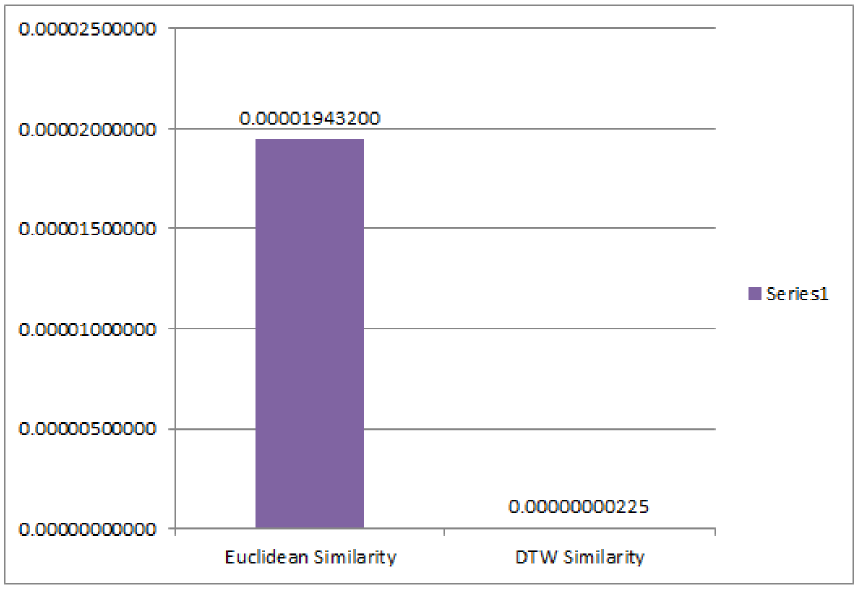
The following are some of the key observations from all tasks:

Key Observations:

- **Task 1**

  Comparing Euclidean and DTW similarity measures

  The Euclidean and DTW similarity are calculated using the input epidemic simulation files and these values are in the range [0-1]. For input simulation files 1 and 2 the comparison between the Euclidean and DTW similarity is as shown below.



  If there are only Euclidean and DTW similarity measures available for simulation files 1 and 2, then we can use DTW similarity as it is very good differentiation factor between the files*(without considering the application semantics).*

Comparing sim $_{\text{word}}$, sim $_{\text{avg\_word}}$, sim$_{\text{diff\_word}}$

It is observed that the values of sim $_{\text{avg\_word}}$, sim$_{\text{diff\_word}}$ are equal for all the pairs of input epidemic simulation files.

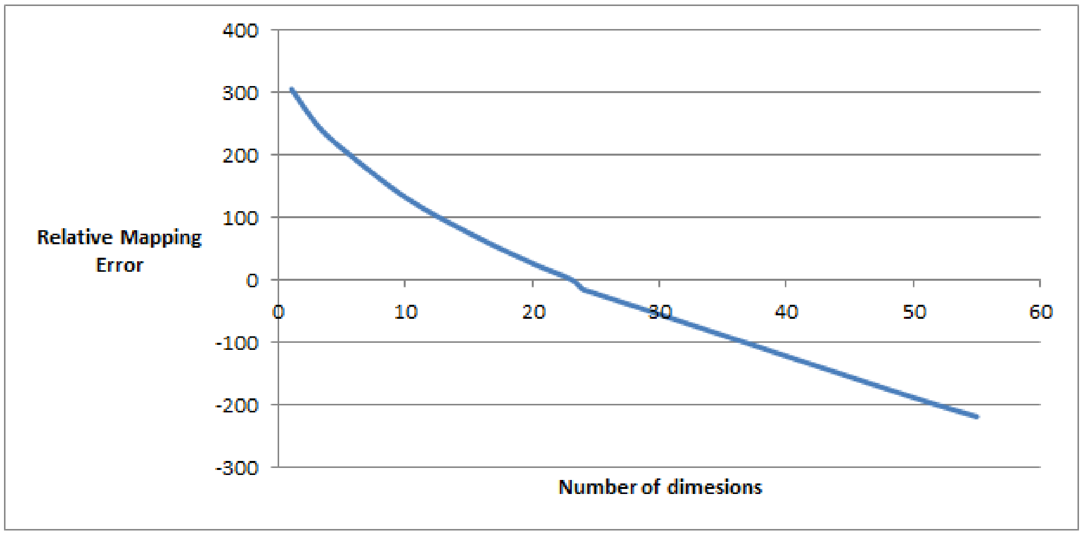
Comparing sim $_{\text{weighted\_word}}$, sim $_{\text{weighted\_avg\_word}}$, sim$_{\text{weighted\_diff\_word}}$

It is observed that the values of sim $_{\text{weighted\_avg\_word}}$, sim$_{\text{weighted\_diff\_word}}$ are equal for all the pairs of input epidemic simulation files.

- **Task 2**
  We observed that, the top K similar files differ with different similarity measure as different measures have different levels of compactness for the same set of database(i.e; set of input epidemic simulation files)

- **Task 3**
  We have implemented the SVD both by using binary vector concept and term frequency. In both the approaches, we have observed the same semantic score of all the files.

- **Task 4**
  The below graph shows the execution of fastmap algorithm on 20 simulation files for various dimensions by considering euclidean distance metric in both new space and original space.. The X axis corresponds to the number of dimensions and the Y axis corresponds to the relative mapping error.
  - When the number of dimensions less we get under estimation
  - As the number of dimensions increases we get low mapping error.
  - After certain number of dimensions we get over estimation.
  - The optimal number of dimensions for this graph is 23



Number of dimensions versus Relative Mapping error

Bibliography

[1] “Exact indexing of dynamic time warping” by Eamonn Keogh, Chotirat Ann Ratanamahatana. http://www.cs.ucr.edu/~eamonn/KAIS_2004_warping.pdf
[2] “FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets” by Christos Faloutsos and King-Ip (David) Lin. http://cs-people.bu.edu/evimaria/cs565/fastmap.pdf
[3] "Euclidean Distance." *Wikipedia*. Wikimedia Foundation.

[4] "Dynamic Time Warping." *Wikipedia*. Wikimedia Foundation.
[5] "Heat Map." *Wikipedia*. Wikimedia Foundation.
[6] "Latent Semantic Analysis." *Wikipedia*. Wikimedia Foundation.
[7] "Centrality." *Wikipedia*. Wikimedia Foundation.
[8] "MetaboNetworks." *File Exchange*. N.p., n.d. Web. 02 Nov. 2014.
[9] "Spatio-temporal Analysis on Enterovirus Cases through Integrated Surveillance in Taiwan." *BMC Public Health 2014,* ***14****:11 doi:10.1186/1471-2458-14-11.*
[10] "Data Management for Multimedia Retrieval", *K. Selçuk Candan, Maria Luisa Sapino Cambridge University Press, May 31, 2010.*
[11] "Matlab Topic Modeling Toolbox 1.4." *Topic Modeling Toolbox*. N.p., n.d. Web. 02 Nov. 2014.

Specific roles of the group members

- Pranay and Sampath worked on the DTW similarity calcualtion.
- Sampath and Anvesh worked on the Fastmap algorithm implementation.
- Mohan and Sravan worked on the getting similarity based on word files (word, average, difference) and Performing the dimensionality reduction and latent semantics discovery.
- Pranay worked on the A matrix construction and optimizations on the algorithm for its construction and getting similarity using the weighted word files (word, average, difference).
- Sampath implemented the task 2 of visualization of input epidemic simulations files for the k-nearest neighbor search using different similarities.
- Anvesh worked on getting outputs and testing for all the similarity measures calculation.
- All team members were involved in the group discussions.
- All team members were involved in the report preparation.