CSE 515 Multimedia and Web Databases

Phase #2

(Due November 2nd 2014, midnight)

Description: In this project, you will experiment with

- dimensionality reduction,
- unsupervised learning, and
- time series.

This project will build on the deliverables of the previous phase. Consider the data that was provided to you in the previous phase.

- **Task 1: Time series similarity**: For this task, let us assume that $epidemic\_word\_file$, $epidemic\_word\_file\_avg$, and $epidemic\_word\_file\_diff$ are already created for a given directory and parameter settings.
  - **Task 1a:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as
    $$sim_{Euc}(f_1, f_2) = \frac{1}{1 + AVG_{s_i \in states}\, \Delta_{Euc}(f_1.s_i, f_2.s_i)}$$
  - **Task 1b:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as
    $$sim_{DTW}(f_1, f_2) = \frac{1}{1 + AVG_{s_i \in states}\, \Delta_{DTW}(f_1.s_i, f_2.s_i)}$$
    For Dynamic Time Warping (DTW), see "*E. Keogh, C. A. Ratanamahatana, Exact indexing of dynamic time warping, Knowledge and Information Systems 7 (3) (2005) 358386.*"
  - **Task 1c:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as
    $$sim_{word}(f_1, f_2) = \vec{w_1}\vec{w_2},$$
    where $\vec{w_i}$ is a binary vector consisting of the words extracted from f_i.
  - **Task 1d:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as
    $$sim_{avg\_word}(f_1, f_2) = \vec{w_{avg,1}}\vec{w_{avg,2}},$$
    where $\vec{w_{avg,i}}$ is a binary vector consisting of the *average words* extracted from f_i
  - **Task 1e:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as
    $$sim_{diff\_word}(f_1, f_2) = \vec{w_{diff,1}}\vec{w_{diff,2}},$$
    where $\vec{w_{diff,i}}$ is a binary vector consisting of the *difference words* extracted from f_i

– **Task 1f:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as

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

where $\vec{w_i}$ is a binary vector consisting of the words extracted from f_i and A is a matrix where each entry,

$$A[idx_{1,i}, idx_{2,j}] = A[\langle\langle f_1, s_i, t_i\rangle, \vec{win}_i\rangle, \langle\langle f_j, s_j, t_j\rangle, \vec{win}_j\rangle],$$

measures

* how close the state time pair, s_i and t_i, is to the state time pair, s_j and t_j and
* how discriminating the windows, $\vec{win}_i$ and $\vec{win}_j$, are in the database.

– **Task 1g:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as

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

where $\vec{w_{avg,i}}$ is a binary vector consisting of the *average words* extracted from f_i and A_{avg} is a matrix as described above, but using the averaged windows.

– **Task 1h:** Implement a program which, given two epidemic simulations files, f_1 and f_2, computes the similarity between them as

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

where $\vec{w_{diff,i}}$ is a binary vector consisting of the *difference words* extracted from f_i and A_{diff} is a matrix as described above, but using the difference windows.

- **Task 2: Time series search**: For this task, let us assume that $epidemic\_word\_file$, $epidemic\_word\_file\_avg$, and $epidemic\_word\_file\_diff$ are already created.

  – **Task 2a-h:** Implement a program which, given a new epidemic simulation file, f_q, an integer k, and one of the similarity measures listed in Task 1, returns the k most similar simulations to f_q and visualizes the query and results as heatmaps.

- **Task 3: Latent Epidemic Analysis and Search Tasks** $\#1$**:**

  – **Task 3a:** Implement a program which, given a set of epidemic simulation files and an integer r, identifies and reports the top-r latent semantics, using SVD.

  – **Task 3b:** Implement a program which, given a set of epidemic simulation files and an integer r, identifies and reports the top-r latent semantics, using LDA.

  – **Task 3c:** Implement a program which, given a set of epidemic simulation files and an integer r and a similarity measure,

  1. creates a *simulation-simulation* similarity matrix,
  2. performs SVD on this *simulation-simulation* similarity matrix, and
  3. reports the top-r latent semantics.

  – **Task 3d-f:** Implement a program which, given a new epidemic simulation file, f_q, an integer k, and one of the options 3a through 3c, returns the k most similar simulations to f_q relying on the corresponding top-r latent semantics.

  You can use Matlab packages for SVD and LDA. The top-r latent semantics should be reported in the form of $\langle simulation, score\rangle$ pairs sorted in non-increasing order of scores.

- **Task 4: Latent Epidemic Analysis Task** $\#2$**:**

- **Task 4a:** Implement a program which, given a set of epidemic simulation files, a similarity measure, and an integer r, applies FastMap to map the simulation files into a r dimensional space and returns the mapping error. For FastMap, see "*Christos Faloutsos and King-Ip Lin. 1995. FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. SIGMOD Rec. 24, 2 (May 1995), 163-174.*"
- **Task 4b:** Implement a program which, given a new epidemic simulation file, f_q, an integer r, returns the k most similar simulations to f_q in this reduces space.

Deliverables:

- Your code (properly commented) and a README file.
- Your outputs for the provided sample inputs.
- A short report describing your work and the results.

Please place your code in a directory titled "Code", the outputs to a directory called "Outputs", and your report in a directory called "Report"; zip or tar all off them together and submit it through the digital dropbox.