Pinterest Project

1) Data Set:

1. pintree.csv -- edge list, it describes how pin propagates from whom to whom
[pintree-id / meta number / (sender) pin id / (received) pin id]

Note: [meta number, pin id] is an identifier for each pin/repin. -> You can link to "pin.csv" data via this identifier

2. pin.csv
[meta number / pin id / user id / category / source / # of likes / birth date]

Note: You can link to user data via "user id"

3. user.csv
[user id / # pins / # followers / # followings / # likes / gender / locale / country / # of followers (twitter) / # of followings (twitter) / # of tweets]

Note: empty field -> unavailable.
Note: Also twitter info is included for future purposes.

4. pintree_pinner.csv
[pintree-id, pinner id]

Note: pinner id is just subset of user id. pinner id are the user id of the users who start the pin tree (who initially pinned the image in our dataset).

5. pinner.csv -- pinner's detailed info
[pinner id, Number of Pins, Number of Followers, Number of Followings, Number of Likes, gender (if available), location (if available), number of categories, number of boards]

6. pin-tree-properties.csv
[pintree-id, size (# nodes in a tree), width (max width in a tree), depth (max depth in a tree), virality (wiender index), category (e.g., food, health, ...), source (e.g., imdb.com, yahoo.com)]

2) Active Followers:

To get this quantity (taking user id 88888 for example):
1. First get all (meta number, pin id) pairs with user id 88888 from the pin.csv file.
2. For each (meta number, pin id), treat it as (meta number, sender pin id) in the pintree.csv and get the corresponding (meta number, received pin id).
3. After having all (meta number, receieved pin id) pairs, go to the pin.csv file again and retrieve all corresponding user ids and treat them as the active followers for user id 88888.

Note: it is possible that a user can repin from his/her non-followers. For example, he/she can search the category menu, and if he/she finds a nice pin, he/she can fetch it. After that he/she can request to be the follower of the one who owns the nice pin. Thus, it's possible that some users' active followers are larger than their true follower

Note: in this project, we only deal with the image propagation probability distribution in active followers.

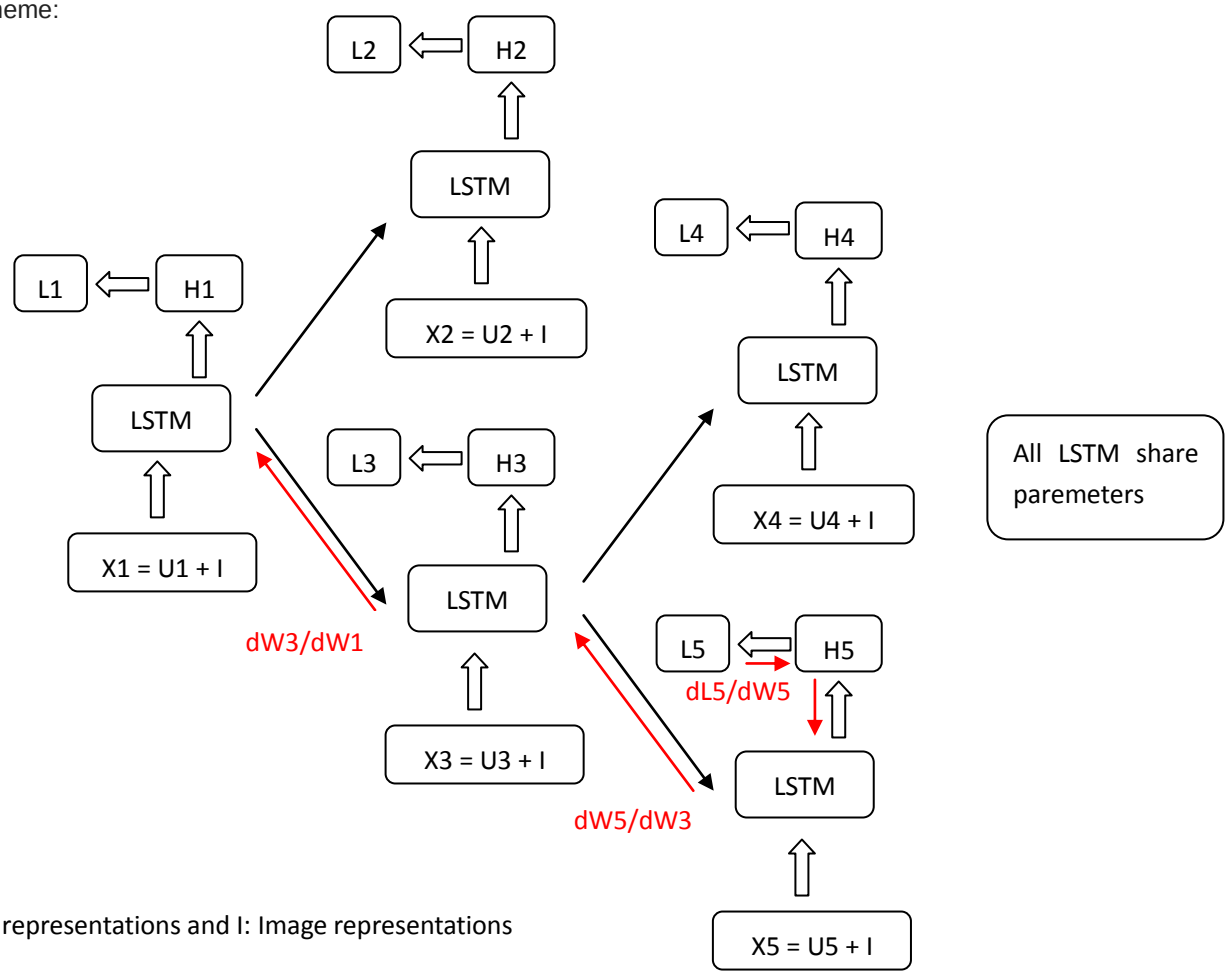
3) Methodology:

1. Image representations: use pre-trained VGG19 FC2 to extract 4096 features for each image.

2. User feature (10 features) manipulation:
(# pins, # followers, # followings, # likes, gender, locale, country, # of followers (twitter), # of followings (twitter), # of tweets)
pins: normalized to [0, 1]
followers: normalized to [0, 1]
followings: normalized to [0, 1]
likes: normalized to [0, 1]
gender: 0: male and 1: female
locale: (longitude, latitude), normalized by the square root of the largest Euclidean distance between two locales
country: (longitude, latitude) of the capital, normalized by the square root of the largest Euclidean distance between countries
# of followers(twitter) : normalized to [0, 1]
# of followings(twitter): normalized to [0, 1]
# of tweets: normalized to [0, 1]

3. User representations:
Cluster all existing users into 500(or other numbers) clusters using k-means according to user features. Then represent each user using 500 cluster centers. At last, each user is represented by a 500 dimensions vector.

4. Training scheme:



Note: Ui: User representations and I: Image representations

5. Softmax loss function:

Total loss: $$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

loss at user i: $$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Use Backpropagation Through Time (BPTT) algorithms to calculate the gradients and update hidden weights. Explicitly:

I. Take user 5 for example.
II. Suppose LSTM's weight at user 5 is W5. Loss at user 5 is L5, where L5 = softmax(V*H5) and V is a mapping matrix.
III. Calculate dL5/dW5, dW5/dW3, dW3/dW1
IV. Update W as: W += lambda*[(dL5/dW5) + (dL5/dW5 * dW5/dW3) + (dL5/dW5 * dW5/dW3 * dW3/dW1)]
V. I call this whole process as the weight update for user 5.

If the above descriptions are correct, I have another question:
We have users 1, 2, 3, 4, 5. Is there a particular order to update LSTM's weight ? Or am I correct to update the weight as follow?
I. Tree level 0: Input X1 and output H1. Update the weight for user 1.
II. Tree level 1: Input X2 and output H2. Update the weight for user 2. Input X3 and output H3. Update the weight for user 3.
III. Tree level 2: Input X4 and output H4. Update the weight for user 4. Input X5 and output H5. Update the weight for user 5.

Another question: Do we need the regularization loss during training?

6. Stop criterion:
Add a "stop" user id in the user list. If we hit this user id, we stop.

7. Cross-Validation Data Set:
Split the original data set into two parts (70% training : 30% CV) isolately.