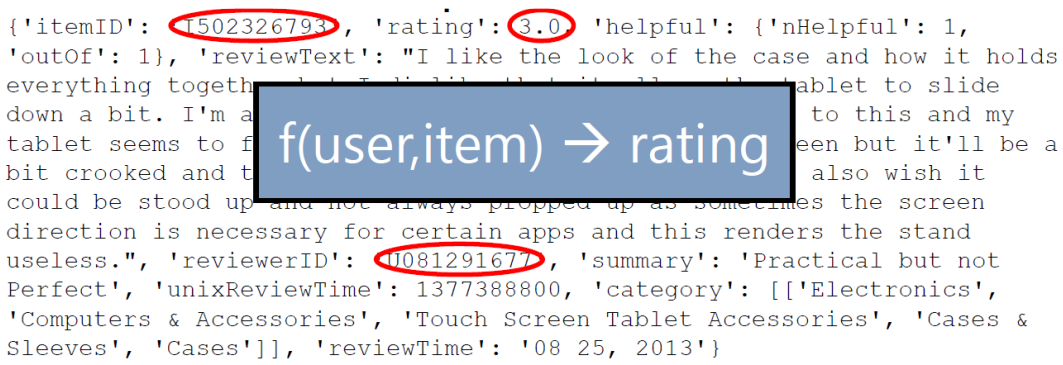
Matrix Factorization for Recommender System

Description of the Problem

In this assignment, you'll be building a recommendation system to make predictions related to reviews of Electronics products on Amazon. You'll be given data that comprises Users, Items, and Ratings. We'll focus on the User-Item utility matrix and build an alternating least square (ALS) based recommendation systems.

To begin, download the files for this assignment from:
http://jmcauley.ucsd.edu/cse255/data/assignment2.tar.gz
train.json.gz: 1,000,000 reviews to be used for training. It is not necessary to use all reviews for training, for example if doing so proves too computationally intensive. These files are one-json-per-line. You will need the following three fields,
itemID: The ID of the item. This is a hashed product identifier from Amazon.
reviewerID: The ID of the reviewer. This is a hashed user identifier from Amazon.
rating: Rating given by the reviewer. The range is 0-5.
Rating.txt: Pairs (userIDs and itemIDs) on which you are to predict ratings (see the tasks below).
In the following snapshot you are shown the relevant fields from training file.



Use the following guidelines.

1. Taking ratings data from the given data set, build an ALS model with a small number of latent factors, between 10-50 factors.
2. We strongly recommend that you first try your code on a smaller dataset.
3. Split the data set into 60-20-20 train-validate-test partitions. That is, the first 60% of the data is the training set. The next 20% is for validation and the remaining 20% is for test. You'll use the training set to learn your ALS model and use the validation set to choose the regularization parameter and number of latent factors. Your splits cannot have any overlapping users but can have overlapping products.
4. You'll evaluate these systems via RMSE (root mean square error) metrics on Validation and Test sets. Make sure you try different regularization parameters $\lambda \in$

$(0.01, 0.1, 1.0, 10.0)$ and several latent factor dimensions and select the model that gives you the best RMSE on the validation set.

5. Once you have finished choosing your model using the validation set, you'll test it on the test set and report that error as your final error metric.
6. Finally, write a simple recommendation engine that will take the ALS model and a ratings file that contains a few ratings from one user and then comes back with a recommendation of products for that user.
7. You can use any data structure library for sparse matrix representation and any linear algebra library for matrix inversion.