**Implementation Details -**

We start by calculating an Artist X Categories matrix. This is a precomputed matrix and is stored in a file named mappedPCA5.csv. In our code this is stored in matrix name 'S' -:

Next we input tag information in a list named tags from a file named tag1.dat.

Next we need to convert our implicit data (present in user\_artists.dat) to explicit ratings (in matrix freq). For this we use the formula as explained in the lastfm paper. The data present in 'user\_artists.dat' is spilt into 4 groups with testing data constituting of 20%, 40%, 60% and 80% respectively in each group. This training data is stored in userMatrix and remaining 80%, 60%, 40%, and 20% respectively data is used as testing data and is stored in matrix userMatrixTest.

Next we define 4 matrixes P, Q, C, G (User X LatentFactor, Artist X LatentFactor, Category X LatentFactor, Tag X LatentFactor respectively) randomly. To converge C and G to their respective values we used traditional matrix factorization method with C and G as random matrixes and matrix CXT as the resultant matrix. CXT is Category X Tag matrix which can be calculated by multiplying Artist X Category (S) and Artist X Tag (T).

Next we compute Artists X Tags (c) matrix from data available in file user\_tag\_data.dat. This file contains 6 rows(userID, artistID, tagID, day, month, year) out which we only take 1st, 2nd and 6th rows as input and compute our Artist X Tag matrix.

Next we try to converge P and Q to their values using our modified matrix factorization technique (matrix factorization + implicit\_semantic\_tag\_time). We also compared this to previously used methods i.e. implicit + semantic and implicit + tag\_time which I will not be discussing in this section.

**implicit\_semantic\_tag\_time (Implementation details):**

We first scaled down freq from a scale of 0-4 to 0-1.

Here we will make some precomputation in order to reduce runtime complexity. We will precompute Artist X Tag and Artist X Category matrixes. Now, as both of these matrixes depend on Artist matrixes for their computation which will keep on changing over the period of time, so we will keep on updating both of these matrixes after each and every 50 iterations.

Rest of the function consists of implementation of binning and updation of P and Q matrix

Finally we will test this on our testing data (userMatrixTest). For this we will compute the dot product b/w P and Q and will compare them against available ratings in test data. Error is calculated is RMSE.