Report On Tf-Idf and Low-Rank Matrix Factorisation

(1).tf_idf.pm is a package file which contains functions listed below:

1:To calculate Tf-score based on that

tf = No. of occurence of word / total no. of words in document ;

2:To calculate Idf-score of each word based on that

idf = log of (total no. of document / no. of document contain that word) ;

3:To calculate Tf-Idf-score of each word based on that

Tf-Idf = tf*idf ;

Here we give some stop words tf-idf value = 0

4:To find Similarity between Documents based on

sim(a,b) = (a.b)/(|a|.|b|)

where a , b are two document

tf-idf vector

This is Content Based

for this function to run we have other tf_idf_recommender file

in which we form a list of doc. from database by taking book detail

and by taking input from user we recommend item similar to user

query and past book transction which algo is based on tf-idf

(2): ## Low Rank Matrix Factorisation ##

In this we have a R = user -item rating matrix of |U| * |I| dimension in which user rate for each item

from this we assume that we would like to discover K latent features.

Our task, then, is to find two matrics matrices m1 (|U| *|K|) and m2 (|D| *|K|) such that their product approximates

A new R_a = user -item rating matrix of |U| * |I|

1: R =~ m1 * m2T = R_a ;

2: R_a [i][j] = sum(m1T[i][k] * m2 [k][j]) for all 0<=k <K

3: err ^ 2 = (R[i][j] - R_a[i][j]) ^ 2;

= (R[i][j] - sum(m1T[i][k] * m2 [k][j])for all 0<=k <K) ^ 2

To minimize error we use gradient slope method and differentiate with both i and j each

And obtained the gradient, we can now formulate the update rules

Then updated m1[i][j] and m2[i][j]

m1'[i][j] = m1[i][j] + 2*alpha*err*m2[i][j] ;

m2'[i][j] = m2[i][j] + 2*alpha*err*m1[i][j] ;

alpha is a constant whose value determines the rate of approaching the minimum.

A common extension to this algorithm is to introduce regularization to avoid overfitting.

This is done by adding a parameter beta squared error as follows:

err ^2 = (R[i][j] - sum(m1T[i][k] * m2 [k][j])for all 0<=k <K) ^ 2 + beta/2 * sum(||m1|| ^2 + ||m2||^2) for all 0<=k <K

Then updated m1[i][j] and m2[i][j]

m1'[i][j] = m1[i][j] + alpha*(2*err*m2[i][j] - beta *m1[i]

[j]) ;

m2'[i][j] = m2[i][j] + alpha*(2*err*m1[i][j] - beta *m2[i][j]) ;

for a number of steps or untill error get too small ;

This is Collaborative Based

for this function to run we have other recommender_low_rank file

in which R is formed by taking user-item rating from Database

Which Calculate random low_rank aproxmiation matrix

and call matrix factorisation function

and then approxmiate new user_item rating matrix

Then Similarity function method between 2 users which is based on

sim(ui , uj) = sum((R_a[i][k] - mean_r[i])*(R_a[j][k] - mean_r[j])) / |(sum(R_a[i][k] - mean_r[i]))| * |(sum(R_a[j][k] - mean_r[j]))||

for all 0<=k<|I|

**********##########**********
****

Above two methods were implemented in perl language and using a Database :

Database Name :Library

Table: (1)Student_information

(2)Book_information

(3)Book_Accession_information

Which is implemented in sql.

END