Item Based Collaborative Filtering Recommendation Algorithms

Badrul Sarvar, George Karypis, Joseph Konstan & John Riedl.

http://citeseer.nj.nec.com/sarwar01itembased.html

By Vered Kunik 025483819

Article Layout -

- Analyze different item-based recommendation generation algorithms.
- Techniques for computing item-item similarities (item-item correlation vs. cosine similarities between item vectors).

Article Layout – cont.

- Techniques for obtaining recommendations from the similarities (weighted sum vs. regression model)
- Evaluation of results and comparison to the k-nearest neighbor approach.

Introduction -

- Technologies that can help us sift through all the available information to find that which is most valuable for us.
- Recommender Systems – Apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services, usually during a live interaction.

Introduction – cont.

- Neighbors of x = users who have historically had a similar taste to that of x.

- Items that the neighbors like compose the recommendation.

Introduction – cont.

- Improve scalability of collaborative filtering algorithms .
- Improve the quality of recommendations for the users.
- Bottleneck is the search for neighbors – avoiding the bottleneck by first exploring the ,relatively static, relationships between the items rather than the users.

Introduction – cont.

- The problem – trying to predict the opinion the user will have on the different items and be able to recommend the “best” items to each user.

The Collaborative Filtering Process -

- trying to predict the opinion the user will have on the different items and be able to recommend the “best” items to each user based on the user’s previous likings and the opinions of other like minded users.

The CF Process – cont.

- List of m users and a list of n Items .
- Each user has a list of items he/she expressed their opinion about (can be a null set).
- Explicit opinion - a rating score (numerical scale).
- Implicitly – purchase records.
- Active user for whom the CF task is performed.

The CF Process – cont.

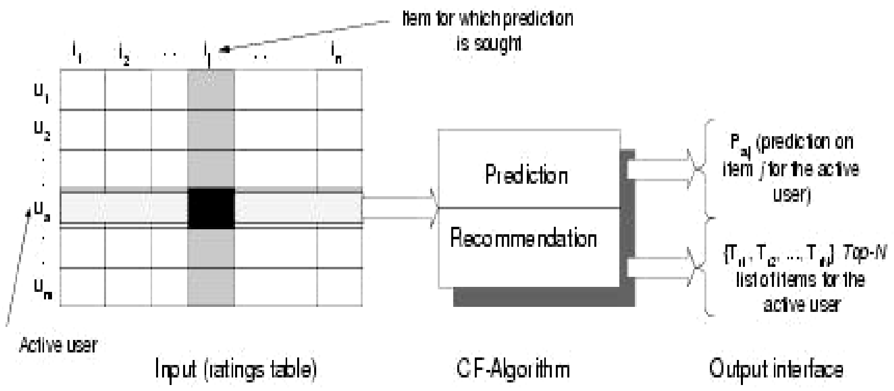
- The task of a CF algorithm is to find item likeliness of two forms :

  Prediction – a numerical value, expressing the predicted likeliness of an item the user hasn't expressed his/her opinion about.

  Recommendation – a list of N items the active user will like the most (Top-N recommendations).

The CF Process – cont.

- The CF process :


Item for which prediction is sought
i_1
i_2
i_j
i_n
u_1
u_2
u_a
u_m
Active user
Input (ratings table)
Prediction
Recommendation
CF-Algorithm
$P_{a,j}$ (prediction on item j for the active user)
$\{T_{i1}, T_{i2}, \ldots, T_{iN}\}$ *Top-N* list of items for the active user
Output interface

Memory Based CF Algorithms -

- Utilize the entire user-item database to generate a prediction.
- Usage of statistical techniques to find the neighbors – <u>nearest-neighbor</u>.

Model Based CF Algorithms -

- First developing a model of user ratings.
- Computing the expected value of a user prediction , given his/her ratings on other items.
- To build the model – Bayesian network (probabilistic), clustering (classification), rule-based approaches (association rules between co-purchased items).

Challenges Of User-based CF Algorithms -

- Sparsity – evaluation of large item sets, users purchases are under 1%.

- difficult to make predictions based on nearest neighbor algorithms.

=> Accuracy of recommendation may be poor.

Challenges Of User-based CF Algorithms –cont.

- Scalability - Nearest neighbor require computation that grows with both the number of users and the number of items.
- Semi-intelligent filtering agents using syntactic features -> poor relationship among like minded but sparse-rating users.
- Solution : usage of LSI to capture similarity between users & items in a reduced dimensional space. Analyze user-item matrix: user will be interested in items that are similar to the items he liked earlier -> doesn't require identifying the neighborhood of similar users.

Item Based CF Algorithm -

- Looks into the set of items the target user has rated & computes how similar they are to the target item and then selects k most similar items.
- Prediction is computed by taking a weighted average on the target user's ratings on the most similar items.

Item Similarity Computation -

- Similarity between items i & j is computed by isolating the users who have rated them and then applying a similarity computation technique.
- Cosine-based Similarity – items are vectors in the m dimensional user space (difference in rating scale between users is not taken into account).

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

Item Similarity Computation – cont.

- Correlation-based Similarity - using the Pearson-r correlation (used only in cases where the uses rated both item I & item j).

$$sim(i,j) = corr_{i,j} = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_j)^2}}.$$

- R(u,i) = rating of user u on item i.
- R(i) = average rating of the i-th item.

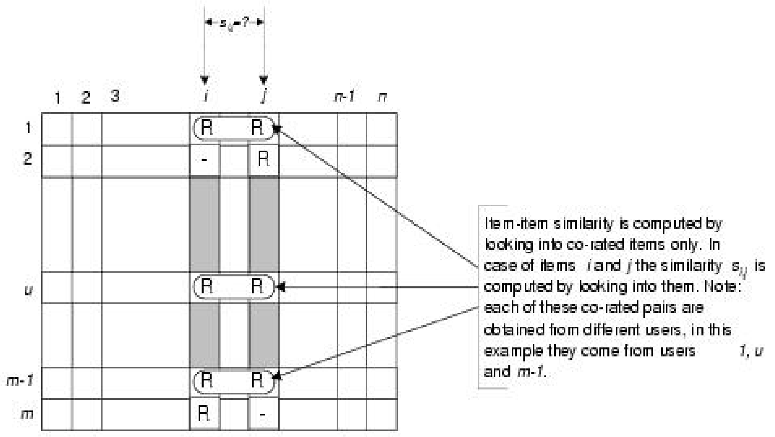
Item Similarity Computation – cont.

- <u>Adjusted Cosine Similarity</u> – each pair in the co-rated set corresponds to a different user. (takes care of difference in rating scale).

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}$$

  - R(u,i) = rating of user u on item i.
  - R(u) = average of the u-th user.

Item Similarity Computation – cont.



s_{ij}=?
1
2
3
i
j
n-1
n
1
2
u
m-1
m
R
-
Item-item similarity is computed by looking into co-rated items only. In case of items i and j the similarity s_{ij} is computed by looking into them. Note: each of these co-rated pairs are obtained from different users, in this example they come from users 1, u and m-1.

Isolation of the co-rated items and similarity computation

Prediction Computation -

- Generating the prediction – look into the target users ratings and use techniques to obtain predictions.
- Weighted Sum – how the active user rates the similar items.

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}} (s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}} (|s_{i,N}|)}$$

Prediction Computation –cont.

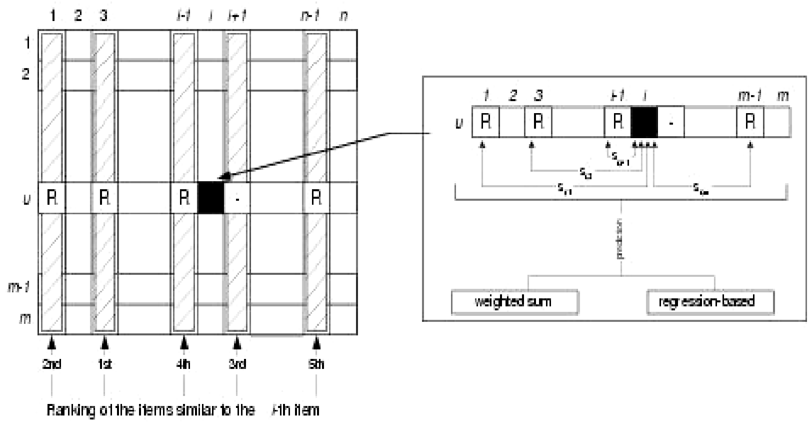
- Regression – an approximation of the ratings based on a regression model instead of using directly the ratings of similar items. (Euclidean distance between rating vectors).

$$\bar{R}'_N = \alpha \bar{R}_i + \beta + \epsilon$$

- R'(N) = ratings based on regression.
- Error.
- Regression model parameters.

Prediction Computation –cont.

- The prediction generation process -



Performance Implications -

- Bottleneck - Similarity computation.
- Time complexity - $\bar{O}(m^4)$, highly time consuming with millions of users and items in the database.
- Isolate the neighborhood generation and predication steps.
- “off-line component” / “model” – similarity computation, done earlier & stored in memory.
- “on-line component” – prediction generation process.

Performance Implications -

- User-based CF – similarity between users is dynamic, precomupting user neighborhood can lead to poor predictions.

- Item-based CF – similarity between items is static.

  enables precomputing of item-item similarity => prediction process involves only a table lookup for the similarity values & computation of the weighted sum.

Experiments : The Data Set -

- MovieLens – a web-based movies recommender system with 43,000 users & over 3500 movies.
- Used 100,000 ratings from the DB (only users who rated 20 or more movies).
- 80% of the data - training set.
- 20% 0f the data - test set.
- Data is in the form of user-item matrix. 943 rows (users), 1682 columns (items/movies – rated by at least one of the users).

Experiments : The Data Set – cont.

- Sparsity level of the data set –
  1- (nonzero entries/total entries)
  => 0.9369 for the movie data set.

Evaluation Metrics -

- Measures for evaluating the quality of a recommender system :
- Statistical accuracy metrics – comparing numerical recommendation scores against the actual user ratings for the user-item pairs in the test data set.
- Decision support accuracy metrics – how effective a prediction engine is at helping a user select high-quality items from the set of all items.

Evaluation Metrics – cont.

- MAE – Mean Absolute Error : deviation of recommendations from their true user-specified values.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$$

The lower the MAE, the more accurately the recommendation engine predicts user ratings.

MAE is the most commonly used and is the easiest to interpret.

Experimental Procedure -

- Experimental steps – division into train and test portion.
- Assessment of quality of recommendations - determining the sensitivity of the neighborhood size, train/test ratio & the effect of different similarity measures.
- Using only the training data & further subdivision of it into a train and test portion.
- 10-fold cross validation - randomly choosing different train & test sets, taking the average of the MAE values.

Experimental Procedure – cont.

- Comparison to user-based systems – training ratings were set into a user-based CF engine using Pearson nearest neighbor algorithm (optimizing the neighborhoods).
- Experimental Platform – Linux based PC with Intel Pentium III processor – 600 MHz, 2GB of RAM.

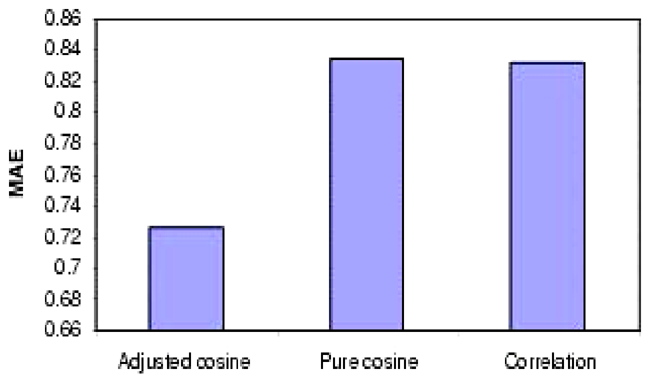
Experimental Results -

- Effect of similarity Algorithms -

- For each similarity algorithm the neighborhood was computed and the weighted sum algorithm was used to generate the prediction.
- Experiments were conducted on the train data.
- Test set was used to compute MAE.

Effect of similarity Algorithms -

- Impact of similarity computation measures on item-based CF algorithm.

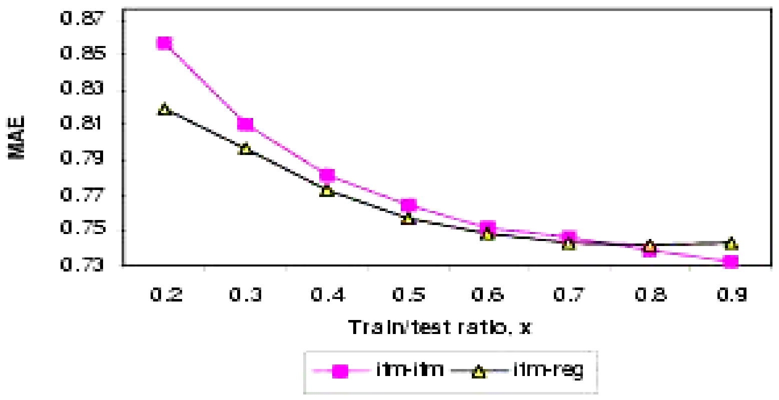


Experimental Results – cont.

- Sensitivity of Train/Test Ratio –
- Varied the value from 0.2 to 0.9 in an increment of 0.1
- Weighted sum & regression prediction generation techniques.
- Use adjusted cosine similarity algorithm.

Sensitivity of Train/Test Ratio -

- Train/Test ratio – the more we train the system the quality of prediction increases.



Sensitivity of Train/Test Ratio -

=>

- Regression based approach shows better results.

- Optimum value of train/test ratio is 0.8

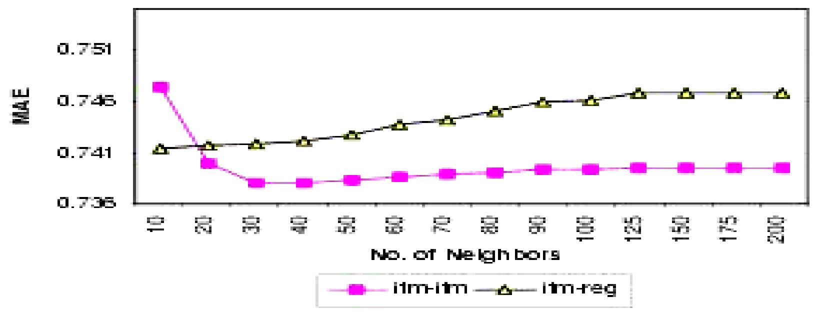
Experimental Results –cont.

- Experiments with neighborhood size -

- Significantly influences the prediction quality.

- Varied number of neighbors and computed MAE.

Experiments with neighborhood size -



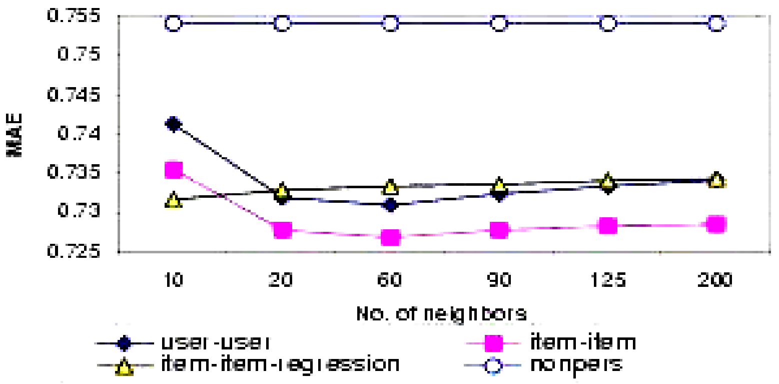
- Regression increase for values > 30.
- Weighted sum tends to be flat for values > 30.

=> Optimal neighborhood size = 30.

Quality Experiments -

- Item vs. user based at selected neighborhood sizes.

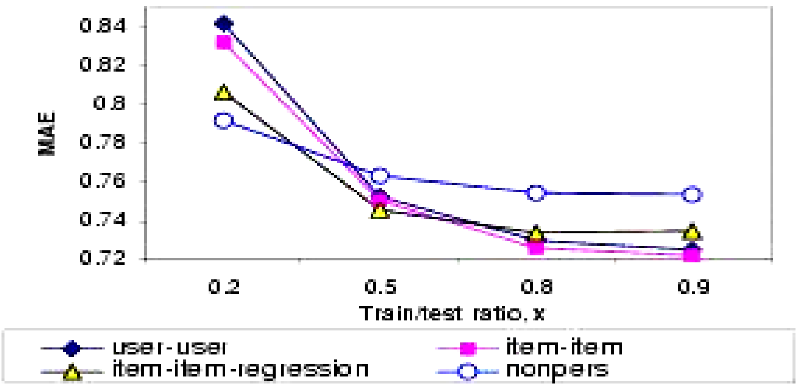
  (train/test ratio = 0.8)



Quality Experiments – cont.

- Item vs. user based at selected density levels.

  (number of neighbors = 30)



Quality Experiments – cont.

=>

- item-based provides better quality than user-based at all sparsity levels – we may focus on scalability.

- Regression algorithms perform better in sparse data (data overfiting at high density levels).

Scalability Challenges -

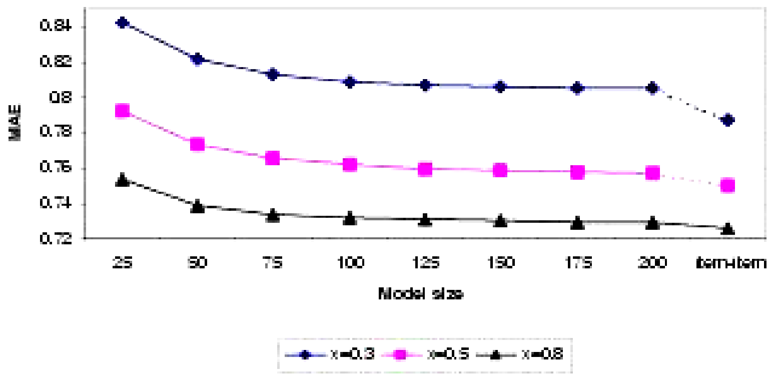
- Sensitivity of the model size – impact of number of items on the quality of the prediction.
- Model size of l = we consider only the l best similarity values for the model building and later on we use k<l of the values to generate the prediction.
- Varied the number of items to be used for similarity computation from 25 to 200.

Scalability Challenges – cont.

- Precompute items similarities on different model sizes using the weighted sum prediction.
- MAE is computed from the test data.
- Process repeated for 3 different train/test ratios.

Scalability Challaenges – cont.

- Sensitivity of model size on selected train/test ratio.



Scalability Challenges – cont.

⇒ MAE values get better as we increase the model size but gradually slows down.

⇒ for train/test ratio of 0.8 we are within 96% - 98.3% item-item scheme's accuracy using only 1.9% - 3% of items.

⇒ High accuracy can be achieved by using a fraction of items – precomputing the item similarity is useful.

Conclusions -

- item-item CF provides better quality of predictions than the user-user CF.
- Improvement is consistent over different neighborhood sizes and train/test ratio.
- Improvement is <u>not</u> significantly large.

- Item neighborhood is fairly static ,hence enables precompution which improves online performance.

Conclusion – cont.

=>

Item–based approach addresses the two most important challenges of recommender systems – quality of prediction & High performance.

THE END

☺