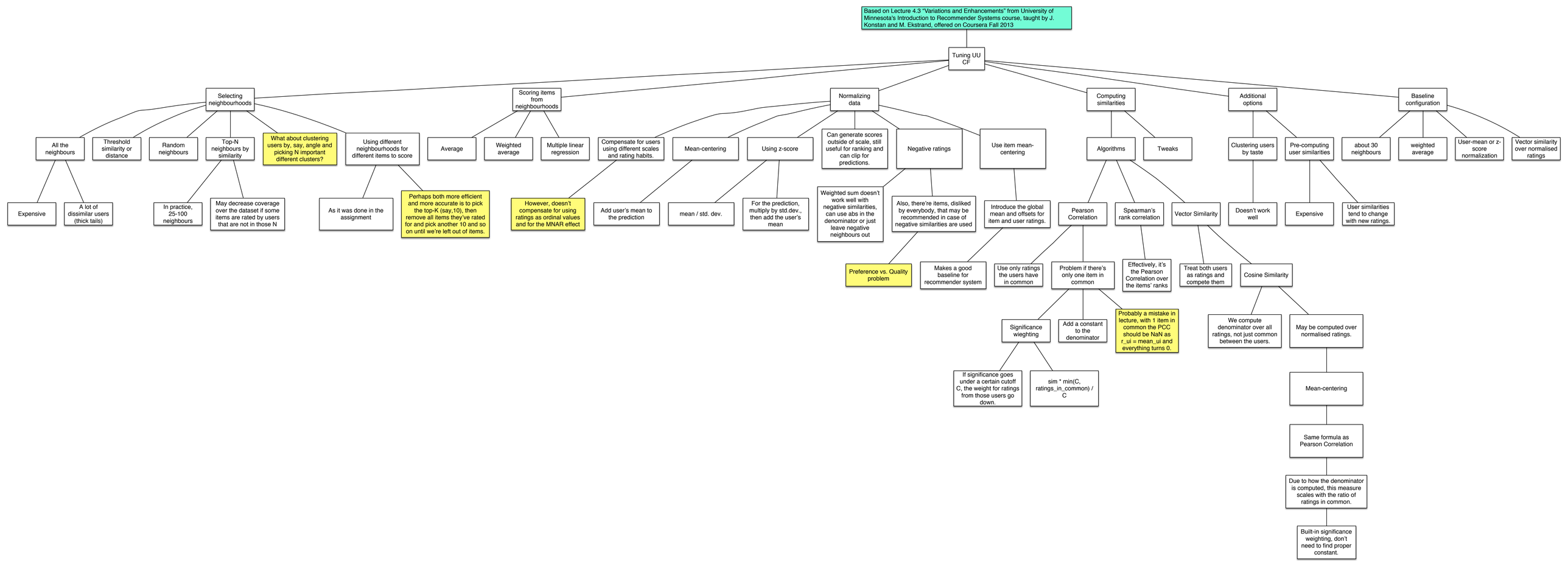
Based on Lecture 4.3 "Variations and Enhancements" from University of Minnesota's Introduction to Recommender Systems course, taught by J. Konstan and M. Ekstrand, offered on Coursera Fall 2013
Tuning UU CF
Selecting neighbourhoods
All the neighbours
Expensive
A lot of dissimilar users (thick tails)
Threshold similarity or distance
Random neighbours
Top-N neighbours by similarity
In practice, 25-100 neighbours
May decrease coverage over the dataset if some items are rated by users that are not in those N
What about clustering users by, say, angle and picking N important different clusters?
Using different neighbourhoods for different items to score
As it was done in the assignment
Perhaps both more efficient and more accurate is to pick the top-K (say,10), then remove all items they've rated for and pick another 10 and so on until we're left out of items.
Scoring items from neighbourhoods
Average
Weighted average
Multiple linear regression
Normalizing data
Compensate for users using different scales and rating habits.
However, doesn't compensate for using ratings as ordinal values and for the MNAR effect
Mean-centering
Add user's mean to the prediction
Using z-score
mean / std. dev.
For the prediction, multiply by std.dev., then add the user's mean
Can generate scores outside of scale, still useful for ranking and can clip for predictions.
Negative ratings
Weighted sum doesn't work well with negative similarities, can use abs in the denominator or just leave negative neighbours out
Also, there're items, disliked by everybody, that may be recommended in case of negative similarities are used
Preference vs. Quality problem
Use item mean-centering
Introduce the global mean and offsets for item and user ratings.
Makes a good baseline for recommender system
Computing similarities
Algorithms
Pearson Correlation
Use only ratings the users have in common
Problem if there's only one item in common
Significance wieghting
If significance goes under a certain cutoff C, the weight for ratings from those users go down.
sim * min(C, ratings_in_common) / C
Add a constant to the denominator
Probably a mistake in lecture, with 1 item in common the PCC should be NaN as r_ui = mean_ui and everything turns 0.
Spearman's rank correlation
Effectively, it's the Pearson Correlation over the items' ranks
Vector Similarity
Treat both users as ratings and compete them
Cosine Similarity
We compute denominator over all ratings, not just common between the users.
May be computed over normalised ratings.
Mean-centering
Same formula as Pearson Correlation
Due to how the denominator is computed, this measure scales with the ratio of ratings in common.
Built-in significance weighting, don't need to find proper constant.
Tweaks
Additional options
Clustering users by taste
Doesn't work well
Pre-computing user similarities
Expensive
User similarities tend to change with new ratings.
Baseline configuration
about 30 neighbours
weighted average
User-mean or z-score normalization
Vector similarity over normalised ratings