Precision @ n

Calculates the fraction of n recommendations that are good. P@N considers the whole list as a set of items, and treats all the errors in the recommended list equally.

$$Precision@k = \frac{(\text{\#of recommended items k that are relevant})}{(\text{\# of recommended items k})}$$

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Recall @ n

$$Recall@k = \frac{(\text{\# of recommended items @k that are relevant})}{(\text{total \# of relevant items})}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

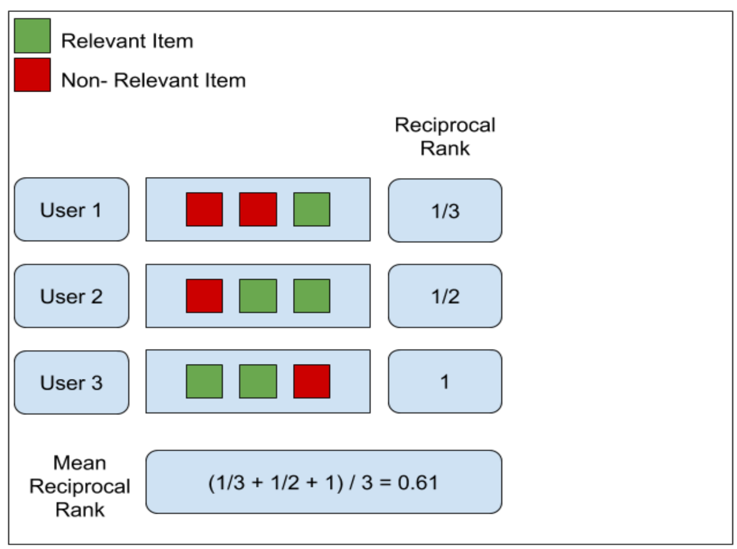
MRR

For each user u:

- Generate list of recommendations
- Find rank k_u of its first relevant recommendation (the first rec has rank 1)
- Compute reciprocal rank $\frac{1}{k_u}$

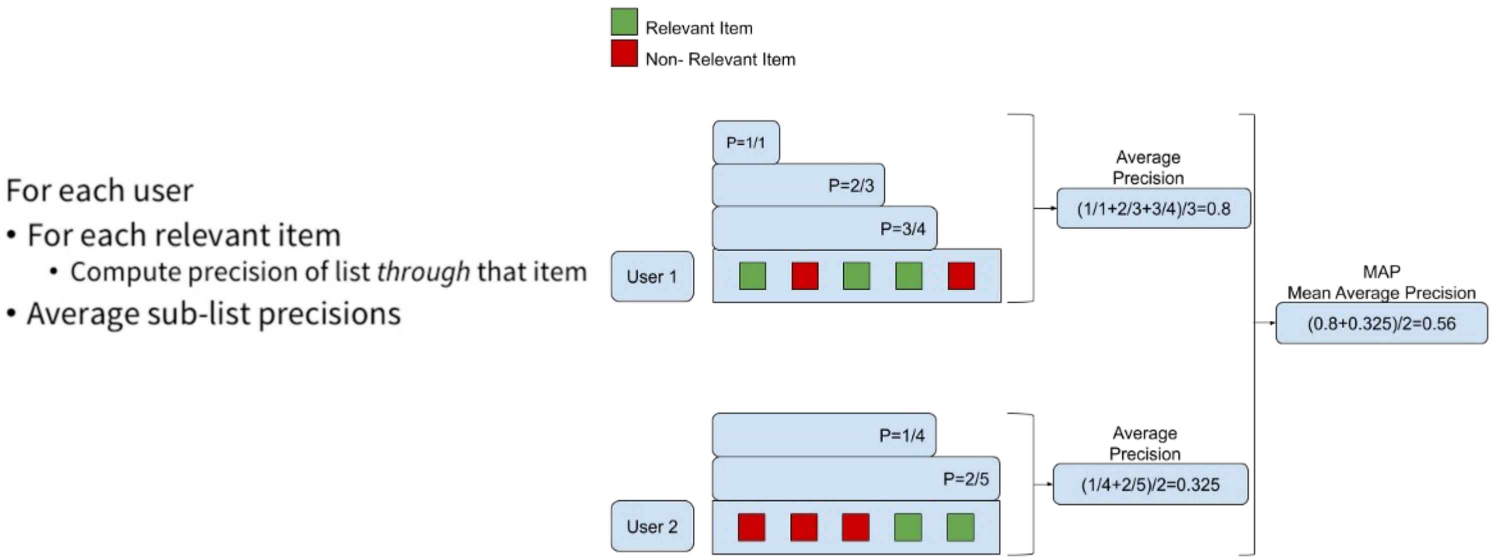
Overall algorithm performance is mean recip. rank:

$$\text{MRR}(O, U) = \frac{1}{|U|} \sum_{u \in U} \frac{1}{k_u}$$



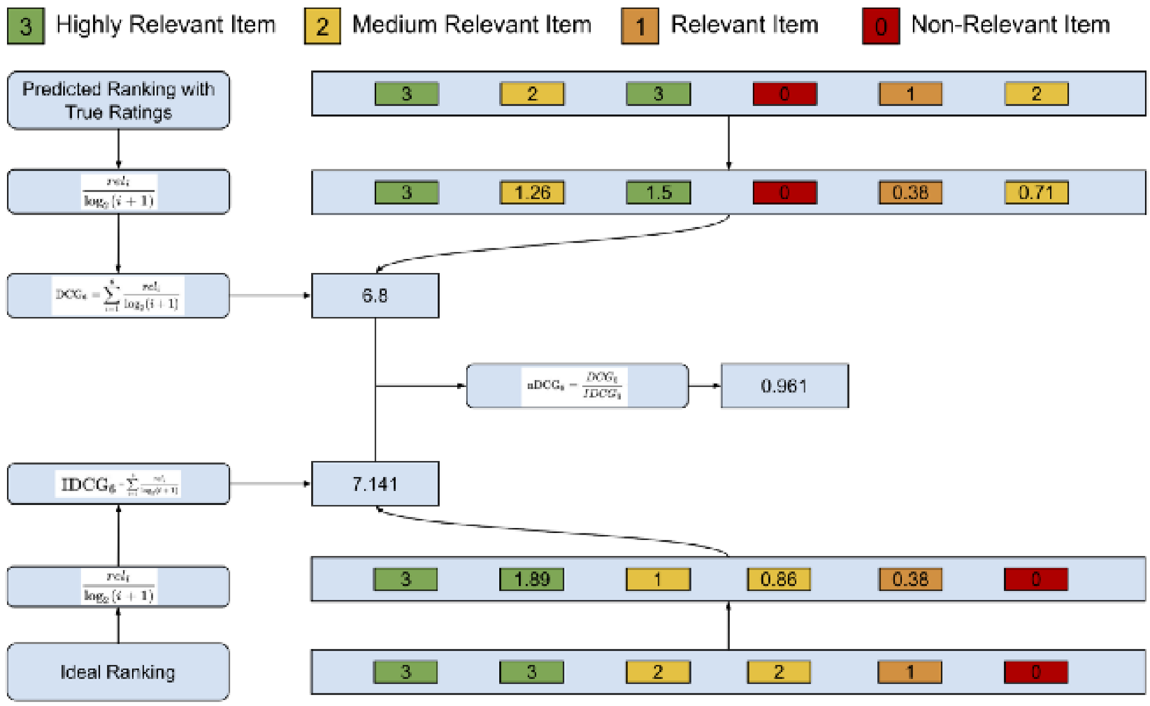
MAP

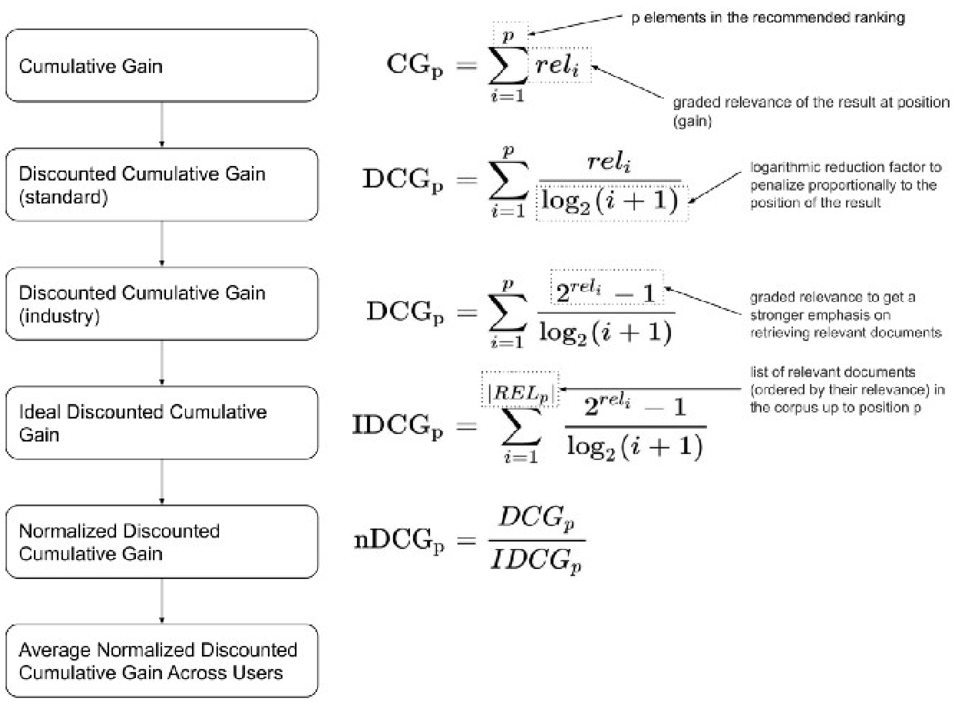
The goal is to weight heavily the errors at the top of the list. Then gradually decrease the significance of the errors as we go down the lower items in a list. It uses a combination of the precision at successive sub-lists, combined with the change in recall in these sub-lists.



NDCG

However, the NDCG further tunes the recommended lists evaluation. It is able to use the fact that some documents are “more” relevant than others.





Serendipity @ n