Rank-Aware Top-N Metrics

Intro

Seen so far:

- how accurate are predictions?
- how good is recommender at finding things?

Now:

- *where* does the recommender list the items it suggests?
- alternatively: how good is the recommender at modeling *relative* preference?

Requirements

Two families of metrics:

Binary relevance metrics need to know if an item is ‘good’ or not (like decision support)

Utility metrics need a measurement of absolute or relative ‘goodness’ of items (e.g. ratings)

Mean Reciprocal Rank

Very simple rank metric:

Where is the first relevant item?

Needs binary relevance judgements.

Mean Reciprocal Rank

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:

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

X A

u B

C

D

$k_B = 2$

$RR = \frac{1}{2}$

$1 \quad 1/2 \quad 1/3 \quad 1/4$

MRR

Benefits

- Very simple
- Clearly models *targeted* search or recommendation tasks (user wants a thing)

Drawbacks

- Less clearly appropriate for general recommendation scenarios

Average Precision

Precision: what fraction of n recs are 'good'?

- Requires fixed n
- Treats all errors equally
    - But accuracy of first few items is more important

Average Precision

For each user

- For each relevant item
  - Compute precision of list *through* that item
- Average sub-list precisions

Result:

- relevance of 1st item counts in many measures
- relevance of 2nd counts one less
- etc

✓ A

✗ B

✓ C

✓ D

✗ E

$P = 1$

$P = 2/3$

$P = 3/4$

$$AP = \frac{1 + \frac{2}{3} + \frac{3}{4}}{3}$$

Mean Average Precision (MAP)

- Take mean of all users' average precision (AP) values

$$\mathrm{MAP}(O, U) = \frac{1}{|U|} \sum_{u \in U} \mathrm{AP}(O(u))$$

Rank Correlation

If we can *rank* items for a user

- Absolute judgements (ratings)
- Relative judgements (e.g. pairwise preferences)

Then we can compare system order O to the user's preference order O_u.

Spearman correlation

Pearson correlation over item ranks

- Assign item rank values
- Ties get average of ranks (e.g. 3,4,5,6 becomes 4.5)

$$\frac{\sum_i \left(k_O(i) - \overline{k_O}\right)\left(k_{O_u}(i) - \overline{k_{O_u}}\right)}{\sqrt{\sum_i \left(k_O(i) - \overline{k_O}\right)^2} \sqrt{\sum_i \left(k_{O_u}(i) - \overline{k_{O_u}}\right)^2}}$$

Problems with Spearman

- Punishes all misplacement equally
- However: we don't care as much low-down
  - swapping 1 and 3: bad
  - swapping 11 and 13: not nearly so bad
- Goal: weight things at the top of the list more heavily

Discounted Cumulative Gain

- Measure *utility* of item at each position in the list
  - Rating r_{ui}
  - For unary data, 1/0 ratings
- Discount by position, so things at front are more important
- Normalize by total achievable utility
- Result is Normalized Discounted Cumulative Gain (nDCG)

Discounted Cumulative Gain

$$\mathrm{DCG}(O, u) = \sum_i \frac{r_{ui}}{\mathrm{disc}(i)}$$

$$\mathrm{disc}(i) = \begin{cases} 1, & i \leq 2 \\ \log_2 i\,, & x > 2 \end{cases}$$

Other discounts possible

A	4
B	3
C	0
D	5

$$DCG = \frac{4}{1} + \frac{3}{1} + \frac{0}{1.58} + \frac{5}{2}$$

$$= 9.5$$

$$DCG_{perfect} = \frac{5}{1} + \frac{4}{1} + \frac{3}{1.58} + 0$$

$$= 10.9$$

$$NDCG = \frac{9.5}{10.9} = 0.872$$

Discounting

- Log discounting is very common
  - For base b (usually 2), no discount for items 1 … b
- Half-life discount has good theoretical basis

$$2^{-\frac{k(i)-1}{\alpha-1}}$$

  - Exponential decay
  - Users exponentially less likely to click each item
  - Half-life α is rank with 50% probability of click
  - Measures *expected utility*

Normalized DCG (nDCG)

- Different users have different ratings, different possible gains
- Normalize gain by *best possible gain*

$$\mathrm{nDCG}(O, u) = \frac{\mathrm{DCG}(O, u)}{\mathrm{DCG}(O_u, u)}$$

- 1 is perfect

Fraction of Concordant Pairs

- What fraction of pairs are in the correct relative order?
- Tests pairwise accuracy

Rank Effectiveness

If we have a user order O_u, we can measure *rank effectiveness*

- Ask recommender to order user's rated items, not pick them from the haystack
- Compare recommender order to user order
- Avoids certain problems with missing data

Conclusion

- Several metrics to measure recommender's ability to order items
- nDCG and MAP increasingly common; MRR also used, particularly in information retrieval

Rank-Aware Top-N Metrics