# Improving ML models in Search Everywhere

#JetBrains

Since our goal is not to build a single loss (or reward if it is an RL agent) function for the ranking model, we can construct a number of metrics that measure the performance of the ranking model in different aspects and not bother with trying to combine them for single numerical output.

## Key Properties of Ranking Metrics

Before moving on to metrics themselves, we should first investigate the properties to look at when considering different approaches. For this purpose, I have studied this paper. Hereby I list the properties I find to be the most important in an IDE setting - in the order of their relevance.

- **Sensitivity**
  - Focuses on the top items of the ranking, which is critical for users in an IDE who care most about the first few search results.
- **Stability**
  - Evaluates whether adding more items to the ranking changes the metric consistently
  - This is very relevant to an IDE setting, since the project environment constantly changes, with new files being added and removed.
- **Robustness**
  - Type I - a single swap in one of the rankings compared implies small changes in score
  - Type II - the same swap in both of the rankings implies no changes in score
  - Measures the metric's ability to reflect small changes proportionally, crucial for an IDE where slight ranking changes (like moving a frequently used file up or down) should have consistent effects.
- **Identity of Indiscernibles** (lol)
  - As long as there exists even slight difference in rankings, they are given different score.
- **Symmetry**
  - The order during comparison of rankings does not matter.

Now provided with a list of properties to look at during metric choice, we can finally list the metrics most applicable to the task in question.

> **Note**
>
> **Ground Truth**
>
> - In all of the metrics below, one way or the other, we mention a notion of **ground truth**
> - This suggests that we are testing ranking models on a set of items that have their relevance scores, and other statistics **predefined**, meaning there has to be substantial research done beforehand
>   - For example, a set of people could be tasked to provide explicit feedback about item relevance to different queries on a scale from 0 to 1
>   - From that we can derive items' actual relevance scores and use them for evaluation as ground truth

## Normalized Discounted Cumulative Gain

$$DCG = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)}, nDCG = \frac{DCG}{IDCG}$$

Where:

- $rel_i$ - the relevance score of the result at rank $i$
  - Numeric attribute obtained from ground truth
- $IDCG$ - the ideal DCG that represents the best possible ranking
  - The maximum obtainable $DCG$
  - To calculate, construct another ranking that sorts the results of a search by their relevance score and calculate $DCG$ of it

Properties:

- Sensitive
- Stable
- Type I robust
- lol

Likely to be the most important metric on the list, judging from its property set. It gives the most priority to relevant results appearing at the top of the ranking, while still considering all of the items.

## Mean Reciprocal Rank

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

Where

- $rank_i$ - the position of the **first** relevant result for query $i$
  - Requires ranking items being classified as relevant or not. We can derive this feature from the relevance score discussed in $nDCG$ (say, $rel_i > 0.5$ implies an item is relevant)

Properties:

- Sensitive - as $rank_i$ focuses only on the first relevant result of the query, $MRR$ is highly sensitive

This metric, despite its simplicity, can still be very beneficial for an IDE setting, since users expect relevant items appear as close to the top of the search as possible. It is thus crucial to evaluate how well the model ranks the first relevant result.

## Precision @K

$$P@K = \frac{\#\text{ of relevant results in top K}}{K}$$

Properties:

- Sensitive - focuses on the top $K$ items of the ranking
- Partially type I robust - changes in the set of top $K$ items will result in different scores, but changing the ordering within them will not

In an IDE setting users will typically focus on the first $K$ search results (say, the ones that appear on the first page), so it is vital to evaluate how many relevant results are present in this list.

## Kendall's Tau

$$\tau = \frac{\#\text{ of concordant pairs} - \#\text{ of discordant pairs}}{C_n^2}$$

Where

- A pair of items $(i, j)$ is called *concordant* if the relative ordering between two different rankings is the same, and called *discordant* otherwise.

Properties:

- Type-II robust

Requires ground truth to feature a full true ranking of the items. For this metric we then compare the ranking provided by the model with the true one.

Kendall's Tau is useful for evaluating overall ranking quality, but in an IDE, where top results matter most, it's less sensitive to relevant changes at the top of the list.

## Recall

$$Recall = \frac{\#\text{ of relevant items retrieved}}{\text{Total }\#\text{ of relevant items}}$$

Recall is helpful in ensuring that the search retrieves all relevant items, but it doesn't account for how highly they are ranked. This might be more important in scenarios where users expect to see all relevant results.

## Summary

While there exist dozens of ranking evaluation metrics out there, it is vital to only consider the ones that are most applicable for the task at hand. In this submission I have picked 5 metrics I have found to be the most relevant to the Search-Everywhere function in an IDE.

### Literature


- Balestra, C., Mayr, A. and Müller, E. (2024) *Ranking evaluation metrics from a group-theoretic perspective*, *arXiv.org*. Available at: https://arxiv.org/abs/2408.16009 (Accessed: 12 October 2024).