Your nDCG Score May Be Correct. But the Conclusion? A Call for Stability-Aware Offline Recommender Evaluation

Joeran Beel[1,2]

[1]University of Siegen, Siegen, Germany
[2]Recommender-Systems.com, Düsseldorf, Germany

Abstract

Offline recommender-system evaluations often end with one selected score per algorithm, such as nDCG. That score may be correct for the reported condition, yet another defensible condition might have produced a different score and favored a different algorithm. Similar scores can also hide different outputs: two Top-10 recommendation lists can share no items and still receive exactly the same nDCG@10. I propose stability-aware offline evaluation to expose this condition dependence. I distinguish score, output, and comparative stability and explain how researchers can define a condition space, construct a condition panel, and match claims to the conditions actually evaluated. Stability is relevant to both reproducibility and the offline–online gap because reproduction and deployment can change some of the same dimensions that vary across offline conditions. I hypothesize that results and algorithm-selection decisions that are stable across defensible offline conditions are more likely to persist under later reproduction or deployment. This hypothesis requires empirical testing. Benchmark studies should quantify score variation, A-versus-B reversals, output changes despite similar scores, and changes in the selected algorithm. Studies with later or online outcomes should then test whether stability improves algorithm selection beyond one selected offline score. The community should compare stability measures and condition-panel designs, while evaluation libraries should retain condition-level scores and outputs and support stability analyses directly. Authors can already report the condition behind each claim, limit claims accordingly, and retain evidence their workflows already produce. nDCG is the running example, but the proposal applies to scalar offline metrics generally.

Keywords

offline evaluation, stability, robustness, context sensitivity, nDCG

CEUR-WS.org/Vol-4341/paper3.pdf

1. Introduction

For a moment, forget recommender systems. Imagine that you want to buy new car tires and that braking distance is your most important criterion. Two models are available. The sales label reports a braking distance of 38 meters for Tire A and 49.5 meters for Tire B. The choice seems easy: Tire A stops more than ten meters earlier; you should buy it. Then you read the fine print. The manufacturer obtained both values with the same standardized test: an emergency stop from 100 km/h, with ABS, on dry asphalt. You search for more tests and find results for wet roads, other surfaces and speeds, and braking without ABS. Tire B has the shorter braking distance in every one of these other tests. The 38-meter result for Tire A remains correct and based on the standard test, but the generalized claim "Tire A brakes better than Tire B" no longer follows from the evidence. Choosing Tire A from that single test may be reasonable, but Tire B may be an equally or even more reasonable choice given that it performed best under all the other test conditions. So, the single number on the sales label describes one result under one condition; it does not show whether the conclusion, and therefore the resulting choice, persists when other test conditions change.

I argue that offline recommender-system evaluation faces the same problem. Researchers often summarize each algorithm by one score such as nDCG. Every reported score is the endpoint of an experimental process: researchers select hyperparameters, filter and split the data, choose a candidate protocol, and fix preprocessing rules before reporting the final nDCG value. Prior work documents

FRAME'26: Methodology First - Rethinking Research Assessment in RecSys Workshop, September 28, 2026, Minneapolis, Minnesota, USA

joeran.beel@uni-siegen.de (J. Beel)
0000-0002-4537-5573 (J. Beel)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-26

cases in which such evaluation choices alter scores and algorithm orderings [1]. In Cañamares et al.'s MovieLens experiment, changing how user-level scores were aggregated reversed the numerical kNN-versus-MF nDCG@10 ordering [2].

A reported nDCG score can describe the evaluated condition accurately while supporting a much narrower claim than "A outperformed B" suggests. Such comparative claims guide decisions about which algorithm to test further or deploy. If the ordering changes across defensible conditions, that choice can hinge on a condition that the final paper leaves in the background. When researchers report only the selected score, they omit score variation that the experimental workflow already produced. If they do not retain and compare condition-specific recommendation lists, they also lose the corresponding output variation. Hyperparameter searches, alternative splits, and methodological comparisons can already generate condition-level scores and recommendation outputs before researchers reduce them to a selected result.

I propose that offline evaluations explicitly examine *stability*: the degree to which an evaluation result persists across defensible experimental conditions. I use *result* to refer to a metric score, such as nDCG=.72; a recommendation list produced for a user; or the comparative conclusion that A performs better than B. The corresponding views are score stability, output stability, and comparative stability. Stability complements effectiveness: an algorithm can be consistently weak or consistently strong across different conditions. Stability qualifies how broadly an effectiveness result persists across the declared panel; it does not determine whether that level of effectiveness is desirable.

Offline recommender-system evaluation has been criticized on several grounds, including its limited correspondence with online outcomes, sensitivity to methodological choices, and mismatch with user or organizational objectives [3, 4, 5, 6, 1, 7, 8, 9]. Yet it remains consequential because it is used to decide which algorithms proceed to more expensive evaluation. If the offline comparison is condition-dependent, the algorithm selected for later evaluation can change with the offline condition. Reproduction or deployment may also differ from the offline experiment along some of the same dimensions. I hypothesize that results and algorithm-selection decisions that are stable across defensible offline conditions are more likely to persist when those dimensions change later. If reproduction or deployment changes along similar dimensions, that evidence bears directly on whether the result persists. Persistence across the declared panel establishes only that the result is not unique to one condition in that panel; whether it predicts later persistence requires empirical validation. Validity, user value, and deployment success remain separate questions.

My contribution is a systematic, condition-based treatment of stability in offline recommender evaluation. I bring score, output, and comparative stability together, clarify which condition-panel relationships support which comparisons, and develop a reporting and research agenda tied to the claim scope and algorithm-selection decisions. This treatment makes the scope of an offline claim explicit and turns condition-level results that experiments already produce into evidence that authors can report and reviewers can inspect.

Earlier recommender-system work uses *stability* primarily for the persistence of predictions or rankings under changes such as newly arriving ratings [10, 11, 12, 13]. Here, stability concerns persistence of scores, recommendation outputs, and comparative conclusions across explicitly varied experimental conditions. In two adjacent position papers, I question treating offline results as intrinsic properties and relying on nDCG as a stand-alone summary [14, 15]. I develop one response here through condition panels and the three stability types. nDCG remains the running example; the proposal applies to scalar offline metrics generally, including Precision@k, Recall@k, F-measure, mean average precision (MAP), mean reciprocal rank (MRR), hit rate, and area under the ROC curve (AUC).

2. Three Types of Stability

The experimental condition gives a stability claim its scope. A *condition* is one assignment of values to the dimensions declared in the analysis, for example a split, filtering rule, candidate protocol, or set of hyperparameters. A *variation dimension* is one aspect that changes between conditions: neighborhood

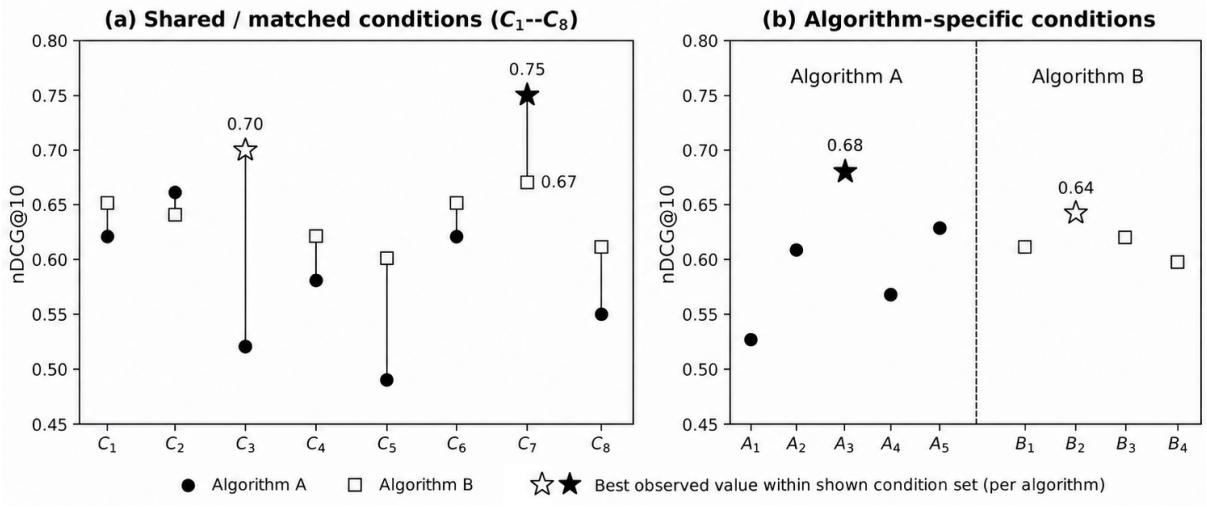

Figure 1: Hypothetical examples of matched and algorithm-specific condition panels. In both panels, B's displayed scores vary less than A's. Panel (a) permits matched within-C_i comparisons and also shows each algorithm's best observed value in the displayed panel. Panel (b) shows within-algorithm variation under algorithm-specific conditions; a direct A-versus-B stability comparison depends on whether the two panels represent comparable variation.

size k is a dimension, while $k = 20$ and $k = 50$ are two values. A *condition panel* is the declared set of conditions over which a study examines stability; every stability statement is conditional on that panel. I call a condition *defensible* for a claim when it is a plausible alternative way of evaluating the same intended question. For example, 5-core or 10-core pruning may be defensible alternatives. Changing what counts as positive feedback — treating clicks as positive feedback in one condition and purchases in another — may instead change the evaluation target itself. Below, *score* refers to the result of an offline metric such as nDCG.

2.1. Score Stability

Score stability is the degree to which one algorithm's metric score persists across the evaluated conditions. Within a declared condition panel, an algorithm is more score-stable when its scores vary less across defensible conditions. Peak effectiveness and stability answer different questions: an algorithm can attain the highest score in a panel while its scores vary strongly across the remaining conditions.

Figure 1 gives a hypothetical example of this distinction and illustrates two ways researchers can structure condition panels. Panel (a) uses eight matched condition pairs C_1–C_8. Two conditions are *matched* when the dimensions that define the comparison have the same meaning for both algorithms. Matching can still allow algorithm-specific components, such as separately tuned hyperparameters, to differ when the evaluation design calls for separate selection. Panel (b) instead uses algorithm-specific conditions A_i and B_i; no A_i–B_j pairing follows from the labels alone.

Panel (a) makes the difference in score stability visible. A's nDCG ranges from .49 to .75, a range of .26, while B ranges from .60 to .70, a range of .10. B is therefore more score-stable by this range-based view. The highest observed value is A=.75 at C_7. A one-score report that selects the maximum would report A=.75; under the matched reading, the comparison at C_7 is A=.75 against B=.67.

If the C_i instead represent model configurations that each algorithm may select separately, B may be reported under its own selected configuration; in the illustration, C_3 with B=.70 is B's best observed candidate. The stars mark best observed values in the displayed panel, not the study's selection rule. Either way, a one-score summary shows notably lower performance for B than for A. It omits the fact that B varies much less across the evaluated conditions than A.

Panel (b) shows the pattern under algorithm-specific variation, for example when each algorithm is evaluated across its own set of hyperparameter configurations. A ranges from about .53 to .68, while

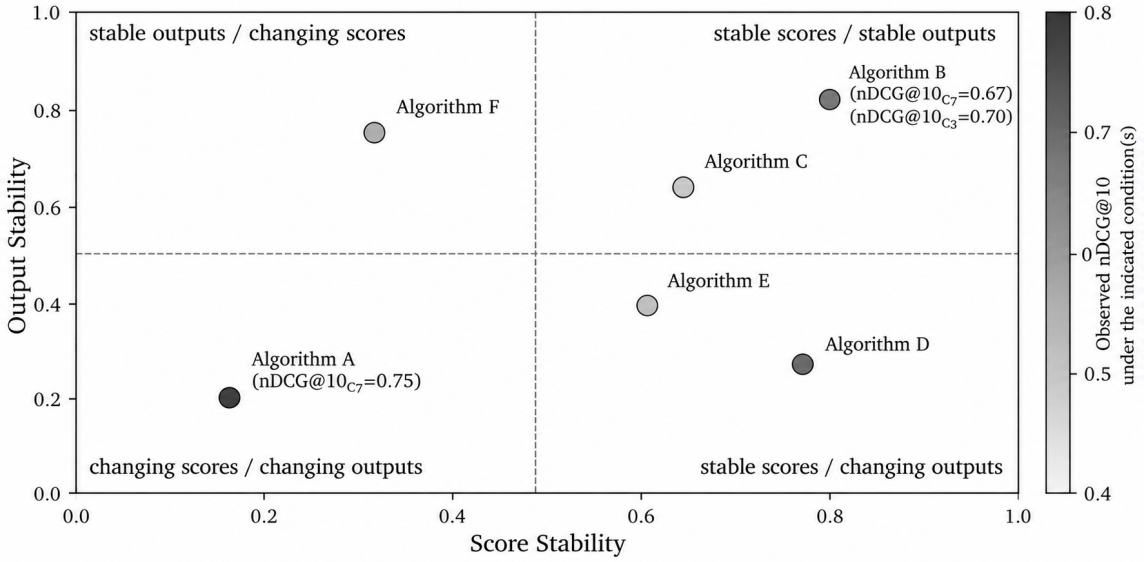

Figure 2: Illustrative score versus output stability. Coordinates are schematic and do not define normalized stability metrics. The condition-specific A/B annotations show that matched and separately selected scores can refer to different conditions; cross-algorithm point comparisons depend on comparable condition panels.

B ranges only from about .60 to .64, so B again shows greater within-panel score stability. Because the panels in (b) need not represent comparable variation, that visual pattern alone does not support a cross-algorithm stability comparison.

2.2. Output Stability

Output stability describes how similar the recommended item lists remain across experimental conditions. Within a declared condition panel, an algorithm is more output-stable when its lists retain more of the same items at similar ranks across conditions. Score stability and output stability capture different properties and can vary independently: similar scores can accompany different recommendation lists, while similar lists can receive different scores when the relevance evidence changes. For example, consider one user whose ideal Top-10 contains items 1–10 with equal relevance. Under C_x, algorithm A recommends relevant items 1, 3, 5, 7, 9 in ranks 1–5, followed by irrelevant items 101–105. Under C_y, it recommends relevant items 2, 4, 6, 8, 10 in the same ranks, followed by irrelevant items 201–205. The lists share no items but receive exactly the same nDCG@10, because relevance occupies the same ranks. The score is perfectly stable while item overlap is zero. By construction, nDCG depends on relevance at ranks rather than item identity [16].

I hypothesize that similar scores accompanied by similar recommendation lists across defensible condition changes provide stronger evidence of persistent behavior than similar scores produced by different lists. When population, catalog, or objective changes, adaptation can appropriately change the recommended items.

Figure 2 combines score and output stability in one schematic view. Because the two dimensions can vary independently, all four quadrants are possible. In the two diagonal cases, scores and recommendation lists either both persist or both change. The upper-left case can occur when lists remain similar but a changed split or temporal window supplies different relevance evidence and therefore different scores. The zero-overlap example above illustrates the lower-right case. The 0–1 axes are schematic. The A/B annotations reuse condition-specific values from Figure 1. B=.67 at matched C_7 and B=.70 at C_3 show how the score attached to an algorithm depends on whether the study reports a matched condition or a separately selected model configuration. The grayscale shows observed nDCG under the indicated condition or conditions.

2.3. Comparative Stability

Score and output stability describe how one algorithm's own results persist across conditions. *Comparative stability* describes how consistently the same A-versus-B conclusion holds across matched conditions. A comparison is more comparatively stable when the same algorithm remains ahead, or when the two remain equivalent under a declared rule, across the matched panel. It is less comparatively stable when the ordering changes with the condition. When offline evaluation determines which algorithm researchers take forward, a change in the preferred algorithm across defensible conditions makes the selection decision condition-dependent.

The matched panel in Figure 1(a) illustrates this directly. Across its eight matched conditions, A has the higher score twice and B six times. Treating only exact ties as equivalent, this gives A a 2/0/6 win/equivalence/loss profile. The comparative conclusion changes across the panel; a conclusion drawn from one condition does not describe how consistently that algorithm remains ahead. Studies should declare an equivalence rule, for example a pre-specified difference small enough to be practically irrelevant for the intended decision.

3. Designing a Stability Analysis

I have argued that stability can provide information that a single offline score does not capture. Turning that idea into a concrete analysis, however, is less straightforward. Researchers must decide which experimental alternatives belong to the analysis, which of them to evaluate, and how to summarize the resulting score, output, and comparative evidence. In this section, I outline a practical approach to these decisions. I do not claim that it is comprehensive, nor do I propose a canonical stability metric.

3.1. Defining the Condition Space

It is well established that experimental choices should be documented for reproducibility. A paper should state which choices produced the reported result, ideally together with a justification. Stability requires an additional layer of information. Reproducibility asks which choices produced the reported result; stability additionally asks which choices were varied, over what range, and according to what strategy.

I use *condition space* for the experimental alternatives that a stability analysis treats as candidates for variation. The analogy to hyperparameter optimization is useful: a hyperparameter study defines a search space before it selects configurations from that space. A stability analysis should likewise state which dimensions may vary and which values or ranges are considered. A practical way to organize these dimensions is by whether variation enters through the model, the data, the evaluation procedure, or the operating context. Examples include pruning rules, splitting strategies, candidate protocols, preprocessing choices, model configurations, temporal windows, populations, and catalogs.

The condition space should contain defensible alternatives for the same intended evaluation question. Authors need not vary every plausible dimension, but they should state which dimensions are included and which are deliberately held fixed. A change that alters the evaluation target itself belongs to a different question rather than serving as evidence of instability. The next step is to select the conditions that will actually be evaluated from this space.

3.2. Constructing the Condition Panel

Evaluating every combination in the condition space will often be infeasible. Researchers should therefore specify a condition-selection strategy, analogous to the search strategy in hyperparameter optimization. A small space may permit a complete grid; a larger one may require random sampling or a targeted set of conditions chosen to probe particular sensitivities. The strategy should be fixed before the stability results are inspected so that the observed results do not determine which conditions enter the analysis. The selected conditions constitute the *condition panel*.

Stochastic repetitions should normally be treated separately from condition variation. Each condition should ideally be evaluated with multiple random seeds or repeated runs when the procedure is stochastic. These repetitions quantify variation within a condition, whereas stability concerns variation between conditions. For example, several random seeds can be repeated realizations under one random-splitting protocol, while random versus chronological splitting represents two different conditions. If sensitivity to the seed itself is the object of the analysis, the seed can instead be treated as a condition dimension.

Comparative stability requires matched conditions whenever the same condition can meaningfully be applied to both algorithms. Algorithm-specific condition spaces may be necessary, for example when algorithms have different hyperparameters. A direct claim that one algorithm is more stable than another then requires a justification that both algorithms were exposed to comparable amounts and types of variation. Without that justification, the panels can still support within-algorithm stability claims.

The panel design should be visible to the reader. For a small panel, the paper can show a compact table with one row per condition and columns for the dimensions that vary. For a larger analysis, the manuscript should report the dimensions, value ranges, and condition-selection strategy, while the complete panel and run-level results can be provided in an appendix or repository. A stability claim is conditional on this design: stability across the evaluated panel does not establish stability across defensible conditions that were not examined.

3.3. Analyzing and Reporting Stability

Once the panel has been evaluated, authors should retain the condition-level results and report the three stability types separately. Score stability can be shown through score distributions, ranges, or condition-level plots. Output stability should compare recommendations for the same eligible users, for example with Overlap@k or Rank-Biased Overlap [17]. Semantic or exposure-based comparisons may be useful when item identity alone does not capture the relevant output change [18, 19]. Comparative stability should be evaluated on matched conditions and can be summarized through win/equivalence/loss patterns under a declared equivalence rule.

Repeated runs within each condition should remain visible in this analysis. Statistical uncertainty describes how precisely a result is estimated within a condition; stability describes how that result changes between conditions. For example, A can have a precisely estimated advantage over B under 5-core pruning while B has a precisely estimated advantage over A under 10-core pruning. Both within-condition estimates can be precise even though the comparative conclusion is unstable.

The condition-level evidence should not disappear into one additional scalar. A stability number may eventually be useful for a particular purpose, but it would not show which conditions changed the score, which changed the recommendations, or which reversed the comparative conclusion. Authors should therefore report the panel together with the score, output, and comparative evidence from which any stability summary is derived.

4. From Proposal to Practice

My outlined proposal raises two practical questions: whether instability across defensible conditions is common enough to matter, and whether stability provides useful information about later reproduction or deployment. Both questions require empirical validation. At the same time, some changes can be adopted immediately without waiting for that evidence. This section therefore outlines a research and practice agenda: quantifying the size of the problem, testing the predictive value of stability, aligning claims with the conditions actually evaluated, making better use of existing condition-level evidence, developing suitable metrics and tooling, and creating incentives for more transparent stability reporting.

Problem size. The first priority is empirical: the community needs to establish whether the stability problem described in this paper is large enough to justify changing evaluation practice. My proposal currently rests on conceptual arguments and on studies that document individual cases of condition

dependence [2, 20, 21, 22]. Larger benchmark studies should quantify how much algorithm scores vary across defensible conditions, how often an A-versus-B conclusion changes, how often recommendation outputs change despite similar scores, and how often a different condition changes the selected algorithm. These estimates would show whether instability is a rare exception or a recurring problem in offline recommender-system evaluation.

Predictive validation. My stronger hypothesis also needs testing: results and algorithm-selection decisions that are stable across defensible offline conditions are more likely to persist under later reproduction or deployment. Testing this hypothesis requires studies that first evaluate several algorithms across a declared condition panel and then observe what happens in a later period, population, site, or online experiment. Such studies should compare algorithm selection based on (i) one selected offline score, (ii) the score plus comparative stability, and (iii) the score plus comparative and output stability. The outcome is whether these selection rules agree with the later winner or ranking. This would directly test whether offline stability adds predictive information beyond one selected score. Prior work already shows that offline and later outcomes can disagree [3, 4, 5, 6] and that richer offline evidence can improve correspondence with later model selection [23, 24]. Stability should therefore be tested for additional predictive value, not assumed to provide it.

Claim scope. Some improvements do not need to wait for further empirical evidence. Authors should state which condition supports each main comparison and keep the claim within the scope of the evaluated conditions. If only one condition was evaluated, "Under the reported condition, A achieved a higher nDCG score than B" matches the evidence more closely than the unqualified "A outperformed B." If a claim is based on a condition panel, authors should report enough about that panel for readers to understand the scope over which the conclusion was tested.

Existing evidence and targeted tests. Researchers should first use condition-level evidence their workflow already produces. Hyperparameter searches, alternative splitting strategies, methodological comparisons, and temporal evaluations can provide scores and Top-k outputs for conditions that have already been evaluated. These results should be retained and analyzed instead of keeping only the selected result. Repeated random seeds should normally remain repetitions within each condition, so that within-condition variation can be distinguished from variation between conditions. When this existing evidence covers too little of the declared condition space, researchers should add targeted conditions, such as 5-core versus 10-core pruning or random versus chronological splitting when both are defensible for the same evaluation question. The aim is to examine plausible sources of condition dependence, not to expand the experiment across every possible choice.

Metrics and tooling. The community should determine which stability summaries are most useful. Candidate measures should be judged by whether readers can interpret them, whether they remain informative across different reasonable panels, and whether they help predict later results or algorithm selection. No single stability number should replace the underlying condition-level evidence. Software support can precede metric standardization: evaluation libraries such as LensKit or RecBole could retain records linking the algorithm, condition values, repetitions, eligible users, scores, and Top-k outputs, and generate score distributions, paired differences, win/equivalence/loss summaries, output similarities, and joint score–output plots. Libraries should also expose defaults that affect the experiment [25].

Incentives and infrastructure. Reviewers and venues should make stability visible in the evaluation process. Reviewers can ask which condition supports a claim, whether other defensible conditions were already evaluated or available, and whether the breadth of the claim matches the breadth of the evidence. Conferences and journals can require transparent condition reporting immediately and encourage authors to report stability when the necessary runs already exist. As empirical evidence accumulates, benchmark maintainers can provide reusable condition panels and venues can decide when additional stability analysis should become a routine part of recommender-system evaluation.

5. Related Work

Prior work establishes condition dependence along individual evaluation dimensions. Within recommender-system evaluation, implementation, baseline selection, and tuning can alter apparent progress [26, 27, 28, 29]. Splitting can reorder systems. Meng et al. generated 230 model configurations and compared rankings under three splitting strategies; Kendall's τ ranged from .53 to .76, where $\tau = 1$ would indicate identical rankings [20]. Candidate sampling can also change offline conclusions, while later work develops more reliable estimators and shows that no single sampling strategy is uniformly best across settings [30, 31, 32, 33].

Other studies document variation from metric implementation [34], split seeds [35], benchmark datasets [36], and time. Scheidt and Beel evaluated six algorithms on ten datasets over time; measured effectiveness changed in nine of the ten datasets and the algorithm ranking changed in six of ten [21]. On MovieLens, Cañamares et al. report that unweighted aggregation gives MF a slightly higher nDCG@10 than kNN (.455 vs. .452), whereas weighting users by their number of test ratings reverses the numerical ordering (.619 for kNN vs. .611 for MF, $p = .001$) [2]. These studies show that individual experimental choices can change offline results.

Evaluation surveys systematize evaluation choices and frameworks [37]. Earlier work I co-authored found variation with user characteristics [38], previous exposure [39], and presentation labels [40]. These studies illustrate population and context dependence; none examines stability in the sense used here. I organize the condition-dependent evidence into three stability views—score, output, and comparative—and treat algorithm selection as the downstream decision they can inform.

Reproducibility asks whether a result can be recovered under the declared setup; I use stability for the separate question of whether that result persists when defensible conditions change. In earlier work with Weiler et al., we treated result stability as a pre-condition for reproducibility and compared event-detection outputs across preprocessing and temporal windows [41]. Recommender-system research has also studied the persistence of predictions, rankings, and Top-k recommendations under changes to ratings or training data [10, 11, 12, 13]. Reproducibility work documents scenario sensitivity [1], surveys practice [42], revisits apparent progress [43], and calls for transparent tuning and pipelines [44]. The PRIMAD framework—Platform, Research Goal, Implementation, Method, Actor, and Data—describes sources of reproducibility variation [45]. I group variation by model, data, evaluation procedure, and operating context to locate where condition variation enters an offline recommender evaluation and which result it affects.

Output-stability work studies similarity and perturbations [46, 47, 48, 49, 22, 50]. Betello et al. provide a direct example of score and output stability diverging: two initialization seeds for GRU4Rec on Foursquare-NYC differ by only 0.1% in nDCG@20, while list similarity remains low (Finite Rank-Biased Overlap=.110; Jaccard=.083) [22]. Measures for comparing outputs include Rank-Biased Overlap and exposure comparisons [17, 18, 19]. Underspecification shows that models with similar selected scores can behave differently beyond that criterion [51], while multiverse and specification-curve methods examine conclusions across plausible analytical alternatives [52, 53]. Learning-theoretic stability uses the same word for a narrower concept: sensitivity to changes in training samples and its relation to generalization [54, 55]. Here, stability refers to the persistence of evaluation results across experimental conditions.

Stability and validity are distinct dimensions of measurement [56, 57]. A result can persist across conditions and still measure the wrong construct or fail to predict outcomes that matter to users or deployment. Recent work further shows that the validity of offline model rankings itself depends on the evaluation design, with no single design performing uniformly best across the studied datasets and dense evaluation targets [58]. Industry studies and user-facing evaluations document gaps between offline objectives, organizational goals, and user-perceived outcomes [7, 8, 9]. The stability analysis proposed here keeps score, output, and comparative variation visible over declared condition panels.

6. Discussion and Conclusion

Offline recommender-system evaluation has made a difficult decision look deceptively simple: compare one score per algorithm and select the better one. Yet every score is tied to an experimental condition. Different defensible choices can change the score, the recommended items, and the A-versus-B conclusion. A paper that reports only the selected score therefore hides how dependent its conclusion is on those choices. The score itself may be correct, while the implied claim about the algorithm is much broader than the evidence supports. Because offline results are used to decide which algorithms proceed to further evaluation or deployment, this dependence can also change the actual algorithm-selection decision.

I propose treating that dependence explicitly through score, output, and comparative stability. Score stability describes whether measured effectiveness persists across the declared condition panel. Output stability describes whether the recommendations persist. Comparative stability describes whether the same algorithm remains preferable across matched conditions. These three views expose different failures of a one-score summary. They can distinguish an algorithm that performs well across many defensible conditions from one whose advantage is confined to a narrow part of the evaluation setup. They can also reveal cases in which almost identical scores correspond to very different recommendation outputs. nDCG has been the running example, but the argument applies to any offline metric whose value or resulting algorithm choice depends on the experimental condition.

I see stability as one possible factor in the reproducibility problem and the offline–online gap. Reproduction and deployment can change some of the same dimensions that differ across offline conditions. I therefore hypothesize that results and algorithm-selection decisions that remain stable across defensible offline conditions are more likely to persist later. If that hypothesis holds, stability analyses could improve candidate selection before expensive online testing and help identify offline conclusions that are especially fragile. Stability does not establish that an offline metric measures user or business value, nor does it replace online evaluation. It provides evidence about whether an offline conclusion already survives smaller changes within the offline evaluation.

The first task is to find out how large this problem actually is. Benchmark studies should quantify score variation, A-versus-B reversals, output changes despite similar scores, and changes in the selected algorithm across defensible conditions. Larger studies should then test the predictive hypothesis directly: does offline stability improve agreement with later reproduction or online outcomes beyond the selected score alone? The figures in this paper are constructed illustrations; they show that these forms of instability are possible, not how often they occur or how large they are in real evaluations. Empirical results should determine which variation dimensions deserve routine attention and which stability summaries are useful.

Several improvements can start before that evidence is complete. Authors should state the condition behind a claim and keep the claim within the scope of the evaluated conditions. They should retain and analyze condition-level evidence that their workflow already produces and add targeted conditions when important alternatives remain untested. Evaluation libraries can automate much of this work, and reviewers and venues can check whether claims match the evidence behind them. Better metrics, panels, and tooling will take time, but better claims and transparency do not have to wait.

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT (OpenAI) and Claude (Anthropic) for Peer review simulation, Content enhancement, Paraphrase and reword, Improve writing style, and Grammar and spelling check. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] J. Beel, C. Breitinger, S. Langer, A. Lommatzsch, B. Gipp, Towards reproducibility in recommender-systems research, User Modeling and User-Adapted Interaction 26 (2016) 69–101. doi:10.1007/s11257-016-9174-x.

[2] R. Cañamares, P. Castells, A. Moffat, Offline evaluation options for recommender systems, Information Retrieval Journal 23 (2020) 387–410. doi:10.1007/s10791-020-09371-3.

[3] J. Beel, S. Langer, A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems, in: S. Kapidakis, C. Mazurek, M. Werla (Eds.), Research and Advanced Technology for Digital Libraries, volume 9316 of *Lecture Notes in Computer Science*, Springer, Cham, 2015, pp. 153–168. doi:10.1007/978-3-319-24592-8_12.

[4] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, A. Huber, Offline and online evaluation of news recommender systems at swissinfo.ch, in: Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 169–176. doi:10.1145/2645710.2645745.

[5] M. Rossetti, F. Stella, M. Zanker, Contrasting offline and online results when evaluating recommendation algorithms, in: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 31–34. doi:10.1145/2959100.2959176.

[6] K. Krauth, S. Dean, A. Zhao, W. Guo, M. Curmei, B. Recht, M. I. Jordan, Do offline metrics predict online performance in recommender systems?, CoRR abs/2011.07931 (2020). URL: https://arxiv.org/abs/2011.07931. doi:10.48550/arXiv.2011.07931. arXiv:2011.07931.

[7] D. Jannach, L. Chen, Recommender systems in industry: Applications, challenges, and the academia–industry gap, ACM Transactions on Recommender Systems (2026) 1–17. doi:10.1145/3827915.

[8] H. Vandenbroucke, L. Michiels, A. Smets, Welcome to the metrics jungle: Organizational stakeholder perspectives on evaluation of news recommender systems in industry, ACM Transactions on Recommender Systems 4 (2026) 54:1–54:42. doi:10.1145/3778173.

[9] A. Zanon, L. Rocha, M. Manzato, Can offline metrics measure explanation goals? a comparative survey analysis of offline explanation metrics in recommender systems, ACM Transactions on Recommender Systems (2025). doi:10.1145/3779420.

[10] G. Adomavicius, J. Zhang, On the stability of recommendation algorithms, in: Proceedings of the 4th ACM Conference on Recommender Systems, RecSys '10, Association for Computing Machinery, 2010, pp. 47–54. doi:10.1145/1864708.1864722.

[11] G. Adomavicius, J. Zhang, Stability of recommendation algorithms, ACM Transactions on Information Systems 30 (2012) 23:1–23:31. doi:10.1145/2382438.2382442.

[12] G. Adomavicius, J. Zhang, Improving stability of recommender systems: A meta-algorithmic approach, IEEE Transactions on Knowledge and Data Engineering 27 (2015) 1573–1587. doi:10.1109/TKDE.2014.2384502.

[13] G. Adomavicius, J. Zhang, Classification, ranking, and Top-K stability of recommendation algorithms, INFORMS Journal on Computing 28 (2016) 129–147. doi:10.1287/ijoc.2015.0662.

[14] J. Beel, The offline evaluation crisis in recommender systems: Have we learned nothing in 20+ years of recommender systems evaluation?, ResearchGate preprint, 2026. URL: https://doi.org/10.13140/RG.2.2.10779.63526. doi:10.13140/RG.2.2.10779.63526, survey and position paper, work in progress, not peer reviewed.

[15] J. Beel, Two decades are enough: nDCG should retire as the default metric for recommender-system evaluation, ResearchGate preprint, 2026. doi:10.13140/RG.2.2.23002.71362, preprint.

[16] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of IR techniques, ACM Transactions on Information Systems 20 (2002) 422–446. doi:10.1145/582415.582418.

[17] W. Webber, A. Moffat, J. Zobel, A similarity measure for indefinite rankings, ACM Transactions on Information Systems 28 (2010) 20:1–20:38. doi:10.1145/1852102.1852106.

[18] F. Diaz, B. Mitra, M. D. Ekstrand, A. J. Biega, B. Carterette, Evaluating stochastic rankings with

expected exposure, in: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, CIKM '20, Association for Computing Machinery, 2020, pp. 275–284. doi:10.1145/3340531.3411962.

[19] A. Singh, T. Joachims, Fairness of exposure in rankings, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18, Association for Computing Machinery, 2018, pp. 2219–2228. doi:10.1145/3219819.3220088.

[20] Z. Meng, R. McCreadie, C. Macdonald, I. Ounis, Exploring data splitting strategies for the evaluation of recommendation models, in: Proceedings of the 14th ACM Conference on Recommender Systems, RecSys '20, Association for Computing Machinery, 2020, pp. 681–686. doi:10.1145/3383313.3418479.

[21] T. Scheidt, J. Beel, Time-dependent evaluation of recommender systems, in: Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2021, volume 2955 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 1–9. URL: https://ceur-ws.org/Vol-2955/paper10.pdf, paper 10.

[22] F. Betello, F. Siciliano, P. Mishra, F. Silvestri, Investigating the robustness of sequential recommender systems against training data perturbations, in: Advances in Information Retrieval, volume 14609 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 205–220. doi:10.1007/978-3-031-56060-6_14.

[23] A. Maksai, F. Garcin, B. Faltings, Predicting online performance of news recommender systems through richer evaluation metrics, in: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 179–186. doi:10.1145/2792838.2800184.

[24] P. Kasalický, R. Alves, P. Kordík, Bridging offline-online evaluation with a time-dependent and popularity bias-free offline metric for recommenders, in: Proceedings of EvalRS: A Rounded Evaluation of Recommender Systems 2023, volume 3450 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3450/paper3.pdf, paper 3.

[25] H. Berling, R. Svahn, A. Said, The hidden cost of defaults in recommender system evaluation, in: Proceedings of the Nineteenth ACM Conference on Recommender Systems, RecSys '25, Association for Computing Machinery, 2025, pp. 1311–1316. doi:10.1145/3705328.3759321.

[26] M. Ferrari Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? a worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19, Association for Computing Machinery, 2019, pp. 101–109. doi:10.1145/3298689.3347058.

[27] M. Ferrari Dacrema, S. Boglio, P. Cremonesi, D. Jannach, A troubling analysis of reproducibility and progress in recommender systems research, ACM Transactions on Information Systems 39 (2021) 20:1–20:49. doi:10.1145/3434185.

[28] S. Rendle, L. Zhang, Y. Koren, On the difficulty of evaluating baselines: A study on recommender systems, CoRR abs/1905.01395 (2019). URL: https://arxiv.org/abs/1905.01395. doi:10.48550/arXiv.1905.01395. arXiv:1905.01395.

[29] F. Shehzad, D. Jannach, Everyone's a winner! on hyperparameter tuning of recommendation models, in: Proceedings of the 17th ACM Conference on Recommender Systems, RecSys '23, Association for Computing Machinery, 2023, pp. 652–657. doi:10.1145/3604915.3609488.

[30] W. Krichene, S. Rendle, On sampled metrics for item recommendation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '20, Association for Computing Machinery, 2020, pp. 1748–1757. doi:10.1145/3394486.3403226.

[31] Y. Liu, A. Medlar, D. Głowacka, On the consistency, discriminative power and robustness of sampled metrics in offline top-n recommender system evaluation, in: Proceedings of the 17th ACM Conference on Recommender Systems, RecSys '23, Association for Computing Machinery, 2023, pp. 1152–1157. doi:10.1145/3604915.3610651.

[32] D. Li, R. Jin, Z. Liu, B. Ren, J. Gao, Z. Liu, Towards reliable item sampling for recommendation evaluation, Proceedings of the AAAI Conference on Artificial Intelligence 37 (2023) 4409–4416. doi:10.1609/aaai.v37i4.25561.

[33] B. L. Pereira, A. Said, R. L. T. Santos, On the reliability of sampling strategies in offline recommender evaluation, in: Proceedings of the Nineteenth ACM Conference on Recommender Systems, RecSys '25, Association for Computing Machinery, 2025, pp. 360–369. doi:10.1145/3705328.3748086.

[34] Y.-M. Tamm, R. Damdinov, A. Vasilev, Quality metrics in recommender systems: Do we calculate metrics consistently?, in: Proceedings of the 15th ACM Conference on Recommender Systems, RecSys '21, Association for Computing Machinery, 2021, pp. 708–713. doi:10.1145/3460231.3478848.

[35] L. Wegmeth, T. Vente, L. Purucker, J. Beel, The effect of random seeds for data splitting on recommendation accuracy, in: Proceedings of the 3rd Workshop on Perspectives on the Evaluation of Recommender Systems, volume 3476 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 1–13. URL: https://ceur-ws.org/Vol-3476/paper4.pdf, paper 4.

[36] V. Shevchenko, N. Belousov, A. Vasilev, V. Zholobov, A. Sosedka, N. Semenova, A. Volodkevich, A. V. Savchenko, A. Zaytsev, From variability to stability: Advancing recsys benchmarking practices, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24, Association for Computing Machinery, 2024, pp. 5701–5712. doi:10.1145/3637528.3671655.

[37] E. Zangerle, C. Bauer, Evaluating recommender systems: Survey and framework, ACM Computing Surveys 55 (2022) 170:1–170:38. doi:10.1145/3556536.

[38] J. Beel, S. Langer, A. Nürnberger, M. Genzmehr, The impact of demographics (age and gender) and other user-characteristics on evaluating recommender systems, in: T. Aalberg, C. Papatheodorou, M. Dobreva, G. Tsakonas, C. J. Farrugia (Eds.), Research and Advanced Technology for Digital Libraries, volume 8092 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2013, pp. 396–400. doi:10.1007/978-3-642-40501-3_45.

[39] J. Beel, S. Langer, M. Genzmehr, A. Nürnberger, Persistence in recommender systems: Giving the same recommendations to the same users multiple times, in: T. Aalberg, C. Papatheodorou, M. Dobreva, G. Tsakonas, C. J. Farrugia (Eds.), Research and Advanced Technology for Digital Libraries, volume 8092 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2013, pp. 386–390. doi:10.1007/978-3-642-40501-3_43.

[40] J. Beel, S. Langer, M. Genzmehr, Sponsored vs. organic (research paper) recommendations and the impact of labeling, in: T. Aalberg, C. Papatheodorou, M. Dobreva, G. Tsakonas, C. J. Farrugia (Eds.), Research and Advanced Technology for Digital Libraries, volume 8092 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2013, pp. 391–395. doi:10.1007/978-3-642-40501-3_44.

[41] A. Weiler, J. Beel, B. Gipp, M. Grossniklaus, Stability evaluation of event detection techniques for Twitter, in: H. Boström, A. Knobbe, C. Soares, P. Papapetrou (Eds.), Advances in Intelligent Data Analysis XV, volume 9897 of *Lecture Notes in Computer Science*, Springer, Cham, 2016, pp. 368–380. doi:10.1007/978-3-319-46349-0_32.

[42] A. Said, A. Bellogín, Reproducibility in recommender systems: A survey, ACM Transactions on Recommender Systems 5 (2026) 16:1–16:23. doi:10.1145/3831686, published online in 2026; issue dated January 2027.

[43] M. Benigni, M. F. Dacrema, D. Jannach, Diffusion recommender models and the illusion of progress: A concerning study of reproducibility and a conceptual mismatch, ACM Transactions on Recommender Systems 4 (2026) 46:1–46:69. doi:10.1145/3795792.

[44] D. Jannach, L. Chen, Improving methodological standards in recommender systems offline evaluation, ACM Transactions on Recommender Systems 4 (2026) 34e:1–34e:15. doi:10.1145/3800587, editorial.

[45] N. Ferro, N. Fuhr, K. Järvelin, N. Kando, M. Lippold, J. Zobel, Increasing reproducibility in IR: Findings from the dagstuhl seminar on "Reproducibility of Data-Oriented Experiments in e-Science", ACM SIGIR Forum 50 (2016) 68–82. doi:10.1145/2964797.2964808.

[46] J.-G. Liu, L. Hou, X. Pan, Q. Guo, T. Zhou, Stability of similarity measurements for bipartite networks, Scientific Reports 6 (2016). doi:10.1038/srep18653.

[47] L. Hou, K. Liu, J. Liu, R. Zhang, Solving the stability–accuracy–diversity dilemma of recommender systems, Physica A: Statistical Mechanics and its Applications 468 (2017) 415–424. doi:10.1016/j.physa.2016.10.083.

[48] S. Oh, B. Ustun, J. J. McAuley, S. Kumar, Rank list sensitivity of recommender systems to interaction perturbations, in: Proceedings of the 31st ACM International Conference on Information and Knowledge Management, CIKM '22, Association for Computing Machinery, 2022, pp. 1584–1594. doi:10.1145/3511808.3557425.

[49] S. Oh, B. Ustun, J. J. McAuley, S. Kumar, FINEST: Stabilizing recommendations by rank-preserving fine-tuning, ACM Transactions on Knowledge Discovery from Data 18 (2024) 237:1–237:22. doi:10.1145/3695256.

[50] B. M. Francomano, F. Siciliano, F. Silvestri, Robust solutions for ranking variability in recommender systems, in: V. Guarrasi, F. Siciliano, F. Silvestri (Eds.), Proceedings of the Workshop on Design, Evaluation, and Deployment of Robust Recommender Systems (RobustRecSys 2024), volume 3924 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 35–40. URL: https://ceur-ws.org/Vol-3924/short7.pdf.

[51] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, F. Hormozdiari, N. Houlsby, S. Hou, G. Jerfel, A. Karthikesalingam, M. Lucic, Y. Ma, C. McLean, D. Mincu, A. Mitani, A. Montanari, Z. Nado, V. Natarajan, C. Nielson, T. F. Osborne, R. Raman, K. Ramasamy, R. Sayres, J. Schrouff, M. Seneviratne, S. Sequeira, H. Suresh, V. Veitch, M. Vladymyrov, X. Wang, K. Webster, S. Yadlowsky, T. Yun, X. Zhai, D. Sculley, Underspecification presents challenges for credibility in modern machine learning, Journal of Machine Learning Research 23 (2022) 1–61. URL: https://jmlr.org/papers/v23/20-1335.html.

[52] S. Steegen, F. Tuerlinckx, A. Gelman, W. Vanpaemel, Increasing transparency through a multiverse analysis, Perspectives on Psychological Science 11 (2016) 702–712. doi:10.1177/1745691616658637.

[53] U. Simonsohn, J. P. Simmons, L. D. Nelson, Specification curve analysis, Nature Human Behaviour 4 (2020) 1208–1214. doi:10.1038/s41562-020-0912-z.

[54] O. Bousquet, A. Elisseeff, Stability and generalization, Journal of Machine Learning Research 2 (2002) 499–526. URL: https://www.jmlr.org/papers/v2/bousquet02a.html.

[55] A. Elisseeff, T. Evgeniou, M. Pontil, Stability of randomized learning algorithms, Journal of Machine Learning Research 6 (2005) 55–79. URL: https://www.jmlr.org/papers/v6/elisseeff05a.html.

[56] N. Bogduk, On understanding reliability for diagnostic tests, Interventional Pain Medicine 1 (2022) 100124. doi:10.1016/j.inpm.2022.100124.

[57] N. Bogduk, On understanding the validity of diagnostic tests, Interventional Pain Medicine 1 (2022) 100127. doi:10.1016/j.inpm.2022.100127.

[58] S. Parajuli, S. V. Barenji, M. D. Ekstrand, On the convergent validity of offline evaluation designs for recommender systems, in: Proceedings of the 20th ACM Conference on Recommender Systems, 2026. doi:10.1145/3773078.3831818. arXiv:2607.25097, to appear.