Ranking Is Not the Bottleneck: A Leak-Free Evaluation of Precedent Retrieval over Merged Pull-Request History

Vladimir Ivanov
Independent Researcher
Georgia
vladimir@ivanovresearch.dev

Abstract

Precedent retrieval serves an AI coding agent the merged pull-request history of its own repository, each item a problem–solution pair. We ask what its evaluation actually measures. On a backport-deduplicated corpus of 575 merged PRs (`dotnet/efcore`), known-item retrieval is strong (recall@5 = 0.90 over 218 held-out queries) — yet three further instruments show that ranking quality is not the binding constraint. (1) In a leak-free temporal holdout, a metadata-linked relevance label finds only 5 of 278 new tasks answerable, while blind human judgment finds 29 of 30: **the standard label misses 93–97% of true precedents**. (2) Under TREC-style pooling, **29% of judged-relevant precedents surface only outside the primary embedding retriever** — single-system evaluation is structurally biased. (3) An LLM judge fails its pre-registered calibration gate (Cohen's $\kappa = 0.39 < 0.6$; recall 0.35 against human labels), missing precisely the convention-transfer cases pooling exists to capture; it is retained advisory-only. (4) A 12-task paired deployment probe shows no outcome uplift on a shallow 3.5-month corpus but also zero harm; the only two value events were codebase-convention transfers from the deepest history available. Together these results locate the bottleneck of precedent retrieval in *relevance definition and corpus depth*, not ranking — and imply that retrieval evaluations which stop at known-item metrics measure the wrong thing.

Keywords

mining software repositories, retrieval evaluation, AI coding agents, temporal holdout, TREC pooling, LLM-as-judge calibration

1 Introduction

AI coding agents fail most where codebases are oldest: the conventions, workarounds, and edge-case lore of a mature repository live not in its current source — which context tools already embed — but in years of merged pull requests that no base model has seen. *Precedent retrieval* targets exactly that gap: index every merged PR as a problem–solution pair (the issue text that states what was wrong, and the diff the team actually accepted), and let an agent working on a new task retrieve how similar problems were actually solved *in this codebase* (Figure 1).

The uncomfortable question is how to know whether such a system works. The standard answer — known-item retrieval, "does the PR known to fix this issue rank in the top-k?" — is cheap, reproducible, and, we will argue, systematically misleading when read as a statement about deployment utility. It measures ranking quality. It says nothing about whether a *genuinely new* task can find a relevant precedent that predates it, which is the question that decides whether precedent retrieval is useful at all.

We built the instruments to answer that harder question over the `dotnet/efcore` history, holding a strict discipline: gates pre-registered before results existed, relevance labeled blind, leakage controls checked rather than assumed. The resulting measurements — including two negative results we report at full strength — compose into one finding:

> **Ranking is not the bottleneck of precedent retrieval. The relevance definition and the depth of the indexed history are.**

Specifically:

- **F1.** Ranking quality is high and stable: recall@5 = 0.90 on a backport-deduplicated known-item benchmark (§4). Yet the same system, evaluated temporally with the field's default relevance label (explicit issue links), appears almost useless: 5 of 278 new tasks answerable. Blind human judgment on a 30-task probe reverses the verdict — 29 of 30 tasks have a genuine prior precedent — exposing that **the metadata label misses 93–97% of true precedents** (§6).
- **F2.** Relevance, once judged rather than inferred, is *plural*: 29% of judged-relevant precedents surface only through retrieval legs other than the primary embedding system (§7). Evaluating a retriever against candidates it alone surfaces is structurally self-confirming.
- **F3.** The obvious scaling move — an LLM judge — fails its pre-registered calibration gate ($\kappa \leq 0.39 < 0.6$), and its misses concentrate on exactly the convention-transfer relevance that pooling exists to capture (§8). We keep it advisory-only.
- **F4.** In a 12-task paired deployment probe with live agents, a shallow corpus (3.5 months of history) produced no outcome uplift — and no harm: agents respected the system's below-threshold signal in 8 of 12 tasks, zero runs were misled, and the only two value events were convention transfers from the deepest history available (§9). Utility tracks corpus depth, not ranking headroom.

Our contributions: (i) the first leak-free temporal-holdout protocol for retrieval over PR history, with leakage controls that are checked, not assumed; (ii) empirical evidence that metadata-linked relevance labels underpower temporal instruments by an order of magnitude, diagnosed by a blind labeled probe; (iii) a TREC-pooled, human-calibrated relevance definition for PR-history retrieval, validated per retrieval leg; and (iv) a measured negative result on LLM-as-judge labeling for this task, with its failure mode characterized. The system, harness, and hash-pinned manifests are open source.

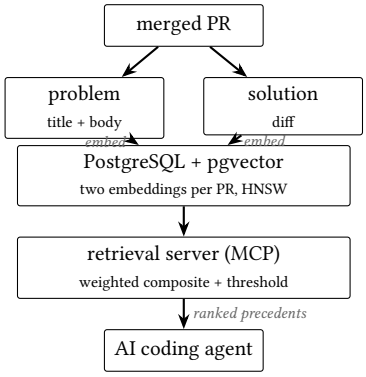

Figure 1: Precedent retrieval. Each merged PR is indexed as a problem–solution pair; an agent queries in natural language and receives ranked precedents with a confidence label, or an explicit below-threshold signal.

2 Background and Related Work

Known-item evaluation. Known-item retrieval — find the one document known to answer the query — is among the oldest evaluation designs in IR [11]. It requires no relevance judging beyond the query–target link, which makes it cheap and reproducible, but it measures only whether a target the system is *told* exists can be ranked highly. Voorhees's work on judgment variation [12] and Zobel's analysis of pooling reliability [14] establish the point we build on: what one measures is determined by how relevance is defined, and narrow definitions bias conclusions.

Temporal splits and leakage. Time-respecting evaluation is standard advice in mining software repositories, yet leakage enters through subtle channels: retrospectively edited artifacts, re-ingested corpora that silently move the split boundary, and labels computed over post-cutoff state. Kalliamvakou et al. [7] catalogue the perils of mining GitHub data at face value; our design (§5) treats their warnings as checklist items — e.g., PR descriptions are mutable, so the split must freeze on merge timestamps and original ingest state, not on current body content. Benchmarks for repository-level agent tasks such as SWE-bench [6] face analogous contamination questions from model training data; our instrument addresses the retrieval-side analogue.

Pooling. Judging every (task, candidate) pair is infeasible at corpus scale. TREC pooling [12, 14] unions the top candidates of multiple independent systems into a judgment pool; relevance is then judged only inside the pool. Pool composition is a known source of bias — a pool built from one system family under-represents what only other families retrieve. Our pool (§7) unions an embedding retriever, a lexical BM25 leg [10], and a second-family embedder (bge-m3 [2]), and we validate each leg's contribution against blind human labels.

LLM-as-judge. Scaling relevance labeling with an LLM judge is attractive and increasingly common [13]. The known risk is that judge agreement with humans is task-dependent and must be measured, not assumed. We pre-registered a Cohen's κ [3] authority gate and report a failure against it (§8); the failure mode — misses concentrated on convention-transfer relevance — is, to our knowledge, underdocumented and directly actionable for others building retrieval evaluations over software history.

Context for coding agents. Contemporary context tools embed *current* code (semantic code search [5], repository indexing in commercial assistants). Precedent retrieval is complementary: it embeds how the code has historically been *changed* — problem–diff pairs from merged PRs [4] — which is precisely the knowledge that is absent from current source and from base-model training in private, legacy codebases.

3 System Under Evaluation

SENRAH is an open-source, self-hosted retrieval server exposed to agents over the Model Context Protocol [1]. We describe the pipeline in the detail needed to interpret the experiments.

What is indexed. Ingestion pulls every merged PR (resumable traversal; bots and automation filtered by stop-list and title patterns; oversized PRs excluded). Two texts are constructed per PR: the *problem* text — PR title concatenated with the PR body — and the *solution* text — the raw unified diff. Each is truncated by *model tokens*, never characters, with independent budgets (1,500 tokens for problem, 6,000 for diff), so titles always survive truncation.

Embedding and storage. Both texts are embedded independently with `text-embedding-3-small` [9] (1,536 dimensions) and stored in PostgreSQL with pgvector; each embedding column carries an HNSW index [8] (cosine distance), so incremental ingest requires no index rebuild. The embedding model and version are recorded per row, giving a re-embedding migration path.

Scoring. A query is embedded once. Approximate nearest-neighbor search oversamples candidates on *both* columns, and candidates are re-ranked by a weighted composite of the two cosine similarities: $score = w_p \cdot sim_{problem} + w_s \cdot sim_{solution}$. The shipped default is $w_p/w_s = 0.6/0.4$; the frozen evaluation protocol uses $0.7/0.3$ with score threshold 0.45, fixed when the protocol was registered. (A systematic weight ablation is future work; the deployment probe in §9 suggests weight tuning is second-order relative to corpus depth.) Results below the threshold are withheld and replaced by an explicit below-threshold signal — a design choice that turns out to matter for deployment safety (§9).

4 Instrument 1: Known-Item Retrieval

Definition. For a known issue whose fixing PR is known, does that PR appear in the top-k results when the issue text is used as the query?

Corpus and protocol. `dotnet/efcore`, deduplicated: 575 merged PRs (merged 2024-04-06 through 2026-06-12), 218 held-out queries. The manifest is hash-pinned and the run is deterministic.

Deduplication is load-bearing. A naive corpus double-counts backports and cherry-picks: the "same" fix appears several times, so *a* hit is easy and recall inflates. We detect backport clusters from stored diffs using union-find over *corroborated* edges — diff

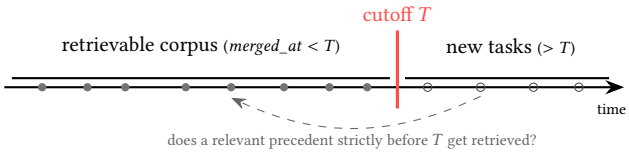

Figure 2: Temporal holdout. The split freezes on merge timestamps and original ingest state; labels freeze together with the corpus boundary; the scorer applies the product's own score threshold.

similarity alone is never allowed to merge two PRs — and score **per-cluster**: a hit on any cluster member counts once, and distractors are likewise counted per-cluster.

Table 1: Known-item retrieval on deduplicated `dotnet/efcore` (575 merged PRs, 218 held-out queries, per-cluster scoring).

Metric	Value
recall@1	0.71
recall@5	0.90
recall@10	0.93
MRR@10	0.79

Why this is not the whole story. Known-item recall answers "can the system rank a target it is told exists." It says nothing about coverage for genuinely new work. Worse, on a deeper corpus known-item recall can *fall*, because more PRs means more near-duplicate distractors competing for top-k slots — so optimizing this number alone penalizes exactly the corpus growth that §9 will show is where utility lives.

5 Instrument 2: Leak-Free Temporal Holdout

Design. Pick a cutoff time T. Everything merged strictly before T is the retrievable corpus; everything merged strictly after T is the incoming task stream (Figure 2). For each post-T task: does a relevant precedent exist strictly before T, and does the system retrieve it? This simulates the deployed setting — an agent working today, retrieving from history.

Leakage controls (where temporal evals die).

- The corpus filter is strictly *merged_at* < T; queries strictly *merged_at* > T. The split freezes on *merged_at* and the *original* ingest timestamps — never on current PR-body state, which is editable after the fact [7].
- Relevance labels and the corpus boundary freeze together, so a later re-ingest cannot silently move the boundary under the labels.
- The scorer applies the product's own below-threshold cutoff, so the instrument measures what a user actually receives, not an idealized ranking.

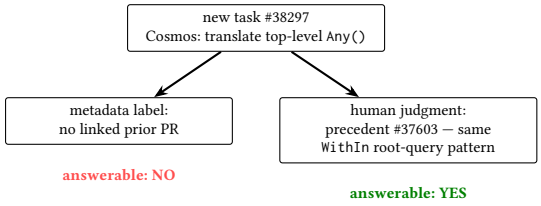

Figure 3: Why metadata labels undercount (real case from the deployment probe): the fix for #38297 reused the root-query detection pattern introduced by #37603 — a genuine precedent transferred by convention, invisible to issue-link metadata.

Materialization. The full multi-year history was ingested for this instrument (`efcore`: 487 → 8,449 PRs; one ingest, one index, every depth rung materializable from it), and a deep cluster map built (9,594 PRs, 397 multi-member backport clusters, hash-pinned).

6 The Power Failure and Its Diagnosis

With relevance defined the way the field usually defaults to — a post-T task is *answerable* iff a metadata-linked prior PR exists strictly before T — the harness came back underpowered (Table 2).

Table 2: Answerable post-T tasks under the metadata-linked relevance label, against a pre-registered power floor of ≥ 80.

Cutoff T (days back)	Post-T tasks w/ issue	Answerable
365	278	**5**
455	306	4
545	345	4

Five is not a measurement; it is noise. The intellectually cheap move is to pivot the evaluation to the known-item number that already looks good and never mention the failure. We did not, because n=5 has two *opposite* explanations with opposite consequences: either (**A**) the label is too strict — metadata-linked relevance misses precedents transferred by convention rather than by explicit issue link — or (**B**) precedents are genuinely rare and the honest output is a negative coverage result.

One cannot tell A from B by staring at the number. We drew 30 random post-T(365) tasks and hand-checked, independently of metadata links, whether a genuine relevant precedent existed strictly before T. A genuine pre-T precedent existed for at least 14 of 30 under a strict reading, and up to roughly 29 of 30 under a lenient one: **the metadata label undercounted true relevance by 14–29× — equivalently, it missed 93–97% of true precedents** (Figure 3 shows a representative real case). Verdict: **A**. The first number was not the finding — it was a symptom, and the probe identified the disease.

7 Rebuilding Relevance: TREC Pooling with Human Calibration

If metadata links undercount relevance, relevance must be judged, not inferred. Judging every pair is infeasible, so we pool: for each

task, the top candidates of three independent retrieval legs are unioned into a judgment pool — `eval` (the evaluated production embedding retriever), `bm25_unique` (candidates only lexical BM25 [10] surfaces), and `bge_unique` (candidates only a second-family embedder, bge-m3 [2], surfaces). Each leg is included because each contributes relevant precedents the others miss; a BM25-only pool recalled just 72% of judged-relevant candidates.

Blind gold set. We hand-labeled 270 (task, candidate) pairs — 30 tasks × 9 candidates, three per leg — *before* looking at any depth result. Density: 65 yes / 205 no (24% positive). Coverage resolves the power failure: **29 of 30** tasks are answerable under judged relevance (versus 5 of 278 under the metadata label).

Table 3: Per-leg precision on the blind gold set.

Leg	relevant / total	precision
`eval`	46 / 90	0.51
`bge_unique`	11 / 90	0.12
`bm25_unique`	8 / 90	0.09

The production leg is precise — every second candidate is relevant. The unique legs are low-precision but decisively non-zero: together they contribute **19 of 65 relevant precedents (29%) that the main leg missed** — the classical pooling trade of coverage bought at the cost of precision [14]. Neither the lexical nor the second-family leg can be dropped without biasing the pool: an evaluation that judges only what the evaluated system retrieves is structurally self-confirming.

8 A Negative Result: The LLM Judge Fails Its Gate

Hand-labeling 30 tasks rescued statistical power, but the instrument needs 80+ tasks to clear its pre-registered floor, and hand-labeling roughly 720 pairs per corpus does not scale. The plan was an LLM judge to label the full pool — *conditional* on passing a pre-registered authority gate: Cohen's κ (judge vs. human) ≥ 0.6 [3].

It did not pass (Table 4).

Table 4: LLM-judge calibration against human gold labels. The pre-registered authority gate was $\kappa \geq 0.6$.

Judge (framing)	κ vs. human	Verdict
Sonnet 4.6 (phase-9 gold)	0.27	below floor
Opus 4.8 (phase-9 gold)	0.39	below floor
Sonnet 4.6 (temporal gold)	**0.39**	**below floor**

On the temporal gold set the confusion matrix (human × judge) is TP = 23, FN = 42, FP = 7, TN = 198 — precision 0.77, recall **0.35**. The judge misses 65% of human-relevant precedents, and the misses concentrate exactly where it hurts: 10 of 11 `bge_unique` and 6 of 8 `bm25_unique` relevant precedents — the semantic and convention-transfer cases the pool was widened to capture. The judge's blind spot is aligned with the pool's reason for existing.

Decision. The judge is retained **advisory-only**: its yes-labels are reasonably trustworthy (precision 0.77); its no-labels are not (recall 0.35), and it can never override the recall guardrail. We prefer reporting a slower, human-anchored result over laundering a coverage number through a labeler we have measured to be unreliable on the cases that matter. Consequently, the full-power forward-coverage number remains *open* and human-labeling-bound; we track it as such rather than substituting a pretend value.

9 Deployment Probe: A Paired A/B on Live Tasks

Retrieval metrics, however honest, are proxies. As a reality check we ran a paired A/B on 12 real `efcore` tasks (well-specified bug reports, selected by a deterministic filter): the same agent model solved each task twice — once with precedent retrieval available (treatment), once without (control) — under a protocol frozen before the last pairs landed. Outcome metrics: file-level and symbol-level recall of the agent's patch against the actual merged fix. The corpus available to the treatment arm was shallow: 264 PRs, roughly 3.5 months of history.

Outcome. A dead heat: 6 control / 4 treatment / 2 ties (two-sided sign test on the 10 non-ties, $p \approx 0.75$); mean file recall 0.55 vs. 0.54. On these outcome metrics the system neither helped nor hurt.

What the outcome metrics do not show. Categorizing all 12 treatment runs:

- *Exact solution shape retrieved:* 0.
- *Codebase-convention transfer:* 2 clear (+2 weak). The fix for #38297 reused the root-query detection pattern of #37603; the fix for #38271 adopted #37652's pattern of excluding full-text/vector/JSON indexes from a DDL transform. Both transfers came from the deepest history present in the shallow corpus, and both pairs also scored at or near the top of the outcome metrics.
- *Found but useless:* 8 — scores clustered 0.31–0.49, mostly below the 0.45 threshold; agents read the below-threshold signal and ignored the results.
- *Misled by a precedent:* **0.** The explicit below-threshold signal did its protective job.

Reading. On a 3.5-month corpus, genuine precedents for novel bugs are rare — the correct answer to most queries is "nothing relevant," which the system reports honestly and agents respect. The two value events that did occur were convention transfers from the oldest history available, and 9 of 12 pairs converged on essentially the same fix regardless of arm — consistent with the task profile (well-specified bug reports that already pin the fix location). This locates the improvement axis in *corpus depth and task profile*, not in re-ranking the same shallow corpus — converging, from the deployment side, with what the temporal instrument showed from the evaluation side.

10 Discussion

The composite finding. Four instruments, one picture. Known-item says ranking is strong (0.90). The temporal instrument says the field's default relevance label hides 93–97% of true precedents.

Pooling says 29% of true relevance is invisible to any single-system evaluation. The deployment probe says utility on a shallow corpus is bounded by content, not ranking — and that an honest below-threshold signal converts "nothing relevant" from a failure mode into safe behavior. None of these facts is visible from the known-item number alone; three of the four would be invisible even in principle.

Why merged-PR history? Among the candidate substrates for precedent retrieval, merged PRs are the only one that bundles a stated problem, a reviewed and *accepted* solution, and a timestamp of acceptance. Commits are too granular and their messages rarely state the problem; issues state problems without solutions; Stack Overflow answers lack codebase specificity and licensing clarity; current-code search [5] shows what the code *is*, not how the team changes it. The merged PR is where a team's intent, review standards, and actual practice coincide [4].

Design guidance. Four practices did disproportionate work. (1) *Pre-register the gates*: the $\kappa \geq 0.6$ authority floor and the ≥ 80-task power floor were fixed before any result existed, so a disappointing result could not be redefined into a passing one. (2) *Blind-label before looking.* (3) *Treat the first number as a symptom*: n=5 was not the answer; the probe that explained it was. (4) *Report negative results as results*: a judge that fails calibration and a coverage question that remains open are stated plainly, with the numbers.

Threats to validity. Everything here is validated on a single corpus (`dotnet/efcore`); efcore's disciplined issue-linking culture likely makes the metadata-label finding *conservative* for messier repositories. The production retriever uses a single embedding family; the pool mitigates but cannot eliminate family bias. The deployment probe is small (N=12), uses one agent model, and over-represents well-specified bug reports — the task profile least likely to benefit from precedents. Judge prompting was varied across two framings but not exhaustively engineered; a materially better judge may yet pass the gate, which is why the gate is pre-registered rather than abandoned. A systematic ablation of the problem/solution weights, and an error taxonomy over the known-item misses, are planned follow-ups. The corpus is English-language.

11 Conclusion

Precedent retrieval over merged-PR history is easy to demonstrate and hard to measure. Measured honestly, its ranking is strong (recall@5 = 0.90), its default relevance labels are an order of magnitude too narrow (93–97% of true precedents missed), 29% of true relevance is invisible to single-system evaluation, the obvious scalable labeler fails its calibration gate ($\kappa = 0.39 < 0.6$) precisely on the cases that matter, and deployed utility tracks corpus depth rather than ranking headroom. The instruments stand; the shortcuts do not. We release the system, the harness, and the hash-pinned manifests, and we report the open questions as open. We hope future benchmarks for retrieval in software engineering adopt leak-free temporal evaluation with judged relevance as the default — and treat known-item retrieval as what it is: a necessary smoke test, not a verdict.

Data Availability

The system, evaluation harness, gold-set labels, and hash-pinned manifests are available in the open-source repository: https://github.com/ivanovresearch/senrah (evaluation account in `docs/EVAL.md`; deployment-probe protocol and per-run scores in `eval/ab/`).

References

[1] Anthropic. 2024. Model Context Protocol. https://modelcontextprotocol.io.

[2] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv preprint arXiv:2402.03216* (2024).

[3] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46.

[4] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-Based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. 345–355.

[5] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search. *arXiv preprint arXiv:1909.09436* (2019).

[6] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. In *The Twelfth International Conference on Learning Representations (ICLR)*.

[7] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The Promises and Perils of Mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*. 92–101.

[8] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836.

[9] OpenAI. 2024. New Embedding Models and API Updates. https://openai.com/index/new-embedding-models-and-api-updates/.

[10] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[11] Karen Spärck Jones and C. J. van Rijsbergen. 1975. *Report on the Need for and Provision of an "Ideal" Information Retrieval Test Collection.* Technical Report British Library Research and Development Report 5266. Computer Laboratory, University of Cambridge.

[12] Ellen M. Voorhees. 2000. Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. *Information Processing & Management* 36, 5 (2000), 697–716.

[13] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*.

[14] Justin Zobel. 1998. How Reliable Are the Results of Large-Scale Information Retrieval Experiments?. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 307–314.