ARXIDEA: AN AGENTIC SYSTEM FOR PERSONALIZED RESEARCH IDEATION AND EVIDENCE-BACKED VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model systems can propose research ideas at scale, but they leave two gaps for individual researchers. The proposals are rarely conditioned on the researcher's own publication record or on papers appearing that week, and the resulting ideas are typically handed over without verification. Existing systems often evaluate personalization using simulated users or model-based judges rather than the intended researcher. We present ARXIDEA, a personalized, human-in-the-loop agentic system that closes these gaps. Its four stages are Recommendation, Ideation, Desk Check, and Experiment Check. Recommendation ranks the week's relevant papers using a profile derived from the researcher's publications. Ideation proposes ideas from them. The two checks form a two-level idea checker that returns an evidence-backed feasibility verdict. We evaluate relevance directly with the profiled researcher. We construct a week-sized evaluation pool by sampling 2,574 new papers uniformly across twelve weeks of arXiv, and the researcher grades every paper for relevance. The resulting pipeline returns 100 papers of which 84 carry the researcher's top relevance grade, spread across subject areas. Adding a reranker over the retrieved candidates reduces recommendation quality according to the researcher's grades. A frontier agent (Claude Opus 5.5) that reads the week-sized sample in place of retrieval only matches it while spending 15 to 29 million tokens per week. Retrieval and ranking therefore operate without a generative model. In the first level, the Desk Check, every candidate idea undergoes premise, theory, and novelty checks, after which it is retained with supporting evidence or eliminated with a recorded reason. The idea the researcher selects then enters the second level, the Experiment Check, a bounded experiment–review loop whose persistent memory is a versioned research specification. The loop records its verdict and hands surviving ideas back to the researcher with findings and next steps. We report three runs—one that passes, one that fails, and one that needs the researcher's input—that expose the checker's verdicts and supporting evidence. We will release the code, prompts, and researcher-provided relevance grades.

1 INTRODUCTION

Deciding which idea to pursue takes a researcher's time. Language models summarize the literature, draft code and manuscripts, and propose research ideas. Systems chain these abilities from a topic or supplied idea to a research report or paper (Lu et al., 2024; 2026; Schmidgall et al., 2025). The researcher still has to decide which proposals fit their work and inspect the evidence behind them. Conditioning such pipelines on the individual scientist has been identified as an open problem (Ni et al., 2026), and where personalization has been evaluated, relevance has come mainly from simulated users or citation proxies (Wang et al., 2026b; Arustashvili & Balog, 2026) rather than from the researcher the output is intended for.

We present ARXIDEA, an agentic system that helps a researcher find and check ideas. Given the researcher's publication record and one week of arXiv postings, it recommends the week's papers and proposes research ideas from them. It checks those ideas first at the desk against the source papers and the literature, then by experiment for the idea the researcher chooses to pursue (Figure 1).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Surviving ideas return to the researcher with findings and next steps; the researcher decides whether to pursue them.

The later stages depend on papers this researcher wants to read. The researcher whose profile the system is built on graded every one of 2,574 new arXiv papers in a week-sized sample, including the papers no method retrieved. These grades let us compare designs on equal terms. The best list fuses three signals derived from the profile. Ranking the week calls no language model, and 84 of its first 100 papers received the top grade. A language model and a cross-encoder each made the list worse when used to rerank retrieved papers. In place of retrieval, a frontier agent (Claude Opus 5.5, allowed to start sub-agents) read the week-sized sample and only drew level, spending 15 to 29 million tokens per week (Appendix D).

The two-level idea checker aims to filter weak ideas and report the evidence behind its decisions. We compare the ideas the Desk Check eliminated and retained over four weeks (Section 5.1). For the Experiment Check, we show one review round from each of three runs (Section 5.2); the verdict rule is in Section 3.5.

We make three contributions. First, we build an agentic system that takes one researcher's record and one week of arXiv to research ideas with evidence from their checks. The researcher steers the process, and its code will be released. Second, we provide a week-sized evaluation pool graded in full by the profiled researcher; the grades will be released with the researcher's consent. On these grades we compare 1,799 retrieval configurations and test a language model as a reranker and in place of retrieval. Third, we evaluate the two-level idea checker on four weeks of candidate ideas and show one review round from each of three Experiment Check runs.

The checks do not certify that an idea will work. They remove the ideas whose premises fail, whose theory breaks, that the literature already covers, or that a reviewer rejects after the experiments, and hand the survivors to the researcher, who continues the rigorous verification.

2 Related work

Personalized paper recommendation. Researchers need to find relevant work as new papers appear. arXiv recommendation has been studied in living labs and inbox services (Kreutz & Schenkel, 2022; Gingstad et al., 2020; Flicke et al., 2025). In general retrieval, language models have been used to rank full candidate lists and smaller groups (Sun et al., 2023; Ma et al., 2023; Zhuang et al., 2024; Long et al., 2026; Chen et al., 2025). For scholarly recommendation, a model can write a natural-language profile from the user's papers (Arustashvili & Balog, 2026). PaperFlow updates a user profile as new papers and feedback arrive (Wang et al., 2026b). Evaluations use different feedback. The recommendation survey describes clicks and citation proxies (Kreutz & Schenkel, 2022), while Scholar Inbox uses ratings from real users (Flicke et al., 2025). SciNUP derives positives from references in authors' later papers (Arustashvili & Balog, 2026), and PaperFlow uses simulated relevance judgments alongside human evaluation (Wang et al., 2026b). In this paper the profiled researcher grades a week-sized sample in full, and every ranking in Section 4 is scored on those grades.

Autonomous research systems. Experiments let research systems test proposed ideas. The AI Scientist proposes ideas, runs experiments, and writes papers (Lu et al., 2024; 2026; Yamada et al., 2025). Co-Scientist generates and refines hypotheses for laboratory testing (Gottweis et al., 2026). Other systems keep structured records of experiments (Tang et al., 2026; Meng et al., 2026; Liu et al., 2026a; Wang et al., 2026a). AutoResearch draws on a continuing stream of papers and other research signals (Liu et al., 2026b). In SoundnessBench, model judges rated weak proposals too favorably (Ho et al., 2026). In SciIntegrity-Bench, agents generated synthetic data in missing-data scenarios (Yang et al., 2026). An audit of reproduction runs finds agents substituting components (Yu et al., 2026). Scientists judged 58% of the interpretive statements they assessed in Kosmos reports correct (Mitchener et al., 2025). Systems also vary in researcher participation. Some run after setup without researcher intervention (Tang et al., 2026; Mitchener et al., 2025), while others allow researcher feedback (Gottweis et al., 2026; Schmidgall et al., 2025; Liu et al., 2026a).

ARXIDEA uses the researcher's publication record to recommend papers and propose ideas. The Desk Check screens every candidate, and the Experiment Check tests the idea the researcher

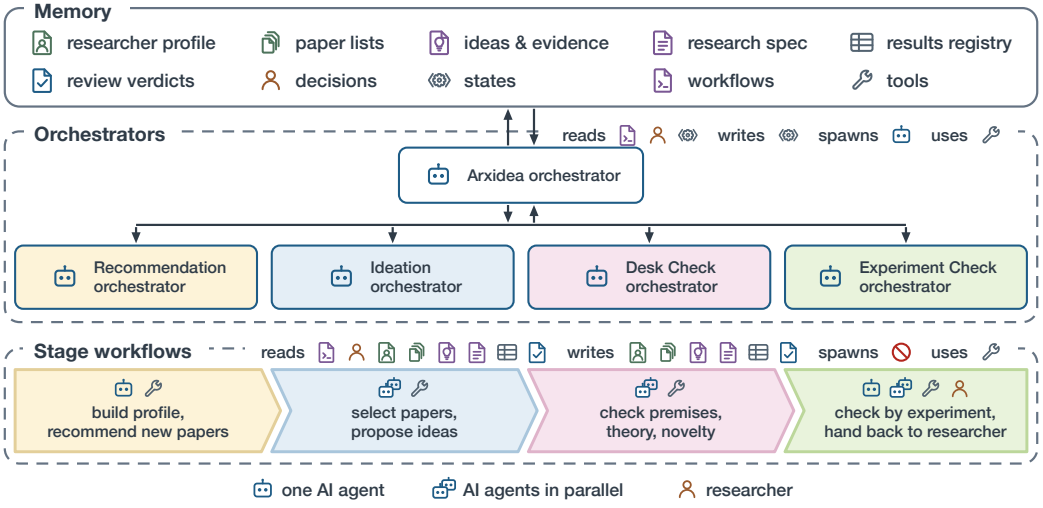


Figure 1: The three layers of ARXIDEA. *Memory*: the files the system keeps on disk; its icons key the strips on the two frames below, which list what each layer may read, write, spawn, and use. *Orchestrators*: a main orchestrator and one orchestrator per stage; vertical arrows are control. *Stage workflows*: the four stages in running order; the wrench marks a program.

chooses. Each surviving idea reaches the researcher with the record of what was checked and what was found.

3 SYSTEM

3.1 OVERVIEW

ARXIDEA carries one researcher's week from new papers to checked ideas. The researcher chooses which idea enters the Experiment Check and resolves points that require their judgment. The system has three layers (Figure 1) so that no agent approves its own work and an interrupted run resumes from what is recorded. The bottom layer holds the four stage workflows, which run in order. Recommendation builds a profile of the researcher and ranks the week's papers against it, and Ideation selects papers from the list and proposes research ideas from them. The Desk Check tests each idea's premises, theory, and novelty. The Experiment Check runs experiments on the idea the researcher picks. If it survives, the researcher receives findings and next steps. The Desk Check and the Experiment Check form the two levels of the idea checker.

Above the workflows sit the orchestrators, each an agent that follows a workflow, code-like text that fixes the steps, their inputs and outputs, and the conditions for moving on. The main orchestrator's workflow fixes the order of the stages and the points where the researcher decides, and each stage orchestrator starts the agents its stage needs. An orchestrator reads the state on disk to see where the run stands, carries out the next step, and writes the state back. The stage orchestrators do not call each other, and everything one stage hands to the next passes through the top layer, a memory of files on disk. In Recommendation, an agent builds the profile; programs fetch and rank the week's papers without a model call. Ideation and the Desk Check are done by agents working in parallel and by programs, and the Experiment Check by an agent, a panel of agents, programs, and the researcher. The agents inside a stage propose, check, run, and review, but they do not decide what happens next and may not start further agents. Figure 6 in Appendix A draws the workflow of each stage.

3.2 RECOMMENDATION

The first stage picks, from the week's new papers, the ones this researcher would want to read, and the later stages build on these papers. It ranks the week against a profile, built once per researcher,

offline, from the researcher's publications. Programs extract key phrases and count arXiv categories from these papers. An agent reads them, writes a note on each, and groups them into research threads, each a line of work the researcher has pursued across several papers. It also summarizes the researcher's recent work. The profile supplies the ranking's only researcher-specific input; ranking a week calls no language model.

Each week the system fetches new arXiv papers in the categories the researcher publishes in. The threads signal ranks candidates by SPECTER2 similarity (Singh et al., 2023) to a thread's papers and their shared references. The BM25 signal (Robertson & Zaragoza, 2009) matches profile phrases in candidate titles and abstracts. The category signal weights a candidate's categories by the researcher's publication counts (Appendix B.1). Each signal yields one or more rankings of the week. Let $r_\ell(p)$ be the rank assigned to paper p in ranking ℓ, and $\mathcal{L}(p)$ the rankings containing p. A weighted form of reciprocal rank fusion (RRF) (Cormack et al., 2009) merges these ranks into the score

$$s(p) = \sum_{\ell \in \mathcal{L}(p)} \frac{w_\ell}{k + r_\ell(p)}, \qquad k = 60, \tag{1}$$

where k is the smoothing constant, fixed before any grade was seen, and w_ℓ is the weight of ranking ℓ. The threads signal gives each research thread its own ranking. Together these rankings carry one vote, with $w_\ell = 1/T$ for T threads. The BM25 and category signals each have weight $w_\ell = 1$. The reading list is the first 100 papers of the fused ranking, a week's reading budget.

This configuration was selected on the researcher's grades (Section 4.2) from seven signals (Table 1). Of the four signals not deployed, the profile signal is a dense query from the whole profile, and the PRF signal (pseudo-relevance feedback) is a time-weighted centroid of the researcher's own papers. The citation signal propagates over the citation graph, and the ref signal measures closeness to the references the researcher's papers share. We also evaluated rerankers after the fusion and deploy none (Section 4.3).

3.3 IDEATION

Relevance alone does not say whether a paper can support new work. For each research thread, the seed is the best-ranked paper that passes three checks on its full text. Its claims must match their evidence, its released code must be reproducible, and it must leave a problem that can be extended. A seed per thread spreads the proposals across the researcher's threads. For each seed an agent, given the profile, proposes up to three candidate ideas. Each is recorded as a falsifiable claim with its premises. The record also specifies a test for each premise, a threshold at which the claim counts as refuted, and the data and code the work would need. Both levels of the idea checker use this record.

3.4 DESK CHECK

The Desk Check eliminates failures that can be found before experiments consume resources. It runs on every candidate idea, checking premises, theory, and novelty in that order and eliminating at the first failure. The premises are tested by a verifier that did not write the idea, using the seed paper's own numbers, symbolic checks, or data the paper released. A premise that would need an experiment to settle is marked as deferred and handed to the next stage. A derivation is written as numbered steps, each checked by a computer-algebra system. Novelty is checked against a scholarly index with replayable queries and a web search for negative results and concurrent work. An idea already covered by prior work, or whose result would add too little, is eliminated.

A program with no model in it re-runs completed tests and replays searches; a mismatch with the record eliminates the idea. An idea leaves the stage either retained, with its evidence and a plan, or eliminated, with the reason recorded. Nothing eliminated comes back.

3.5 EXPERIMENT CHECK

Claims left open by the Desk Check, including deferred premises, require experimental evidence. The Experiment Check takes the idea the researcher picks and tries to refute it by doing the work. The idea becomes a versioned research specification that states its hypothesis and failure criteria.

Its plan first tests any premise deferred by the Desk Check. A loop of experiment and review then runs for at most three rounds. In each round an agent carries out the plan, and a program logs every measured number in a registry. Auditors re-derive the numbers from the raw outputs. A panel of reviewers who saw none of the work votes to keep or to reject the idea. After each round a fixed rule continues the loop or gives one of three verdicts. A reject vote from any reviewer fails the idea, which is eliminated with the reasons recorded, and the researcher is not asked to decide anything. Otherwise, in an earlier round, a point that only the researcher can settle flags the idea, and the loop waits for the researcher's answer before going on. If neither occurs in an earlier round, the loop continues. In the last round the loop allows, an idea with no reject vote passes and is handed back to the researcher with findings and next steps. The researcher answers any point raised in that round. Throughout, the agent that did the work never judges it, and no number enters the specification except from the registry.

4 Evaluating the recommendation

The later stages depend on the 100 papers the Recommendation stage returns each week. We test that foundation by scoring every ranking against the researcher's grades on a week-sized sample of arXiv.

4.1 Ground truth: a week-sized sample

Recall counts missed relevant papers, so every paper in the pool carries the researcher's grade. We took the ten arXiv categories in which the researcher has published and collected every paper first submitted to any of them over twelve consecutive weeks, from 15 June to 6 September 2026, 30,695 papers in all. From these we drew a uniform random sample of 2,574 papers, about the number those categories receive in an average week. A paper belongs to the week of its first version, so revisions of older papers do not enter, and the researcher's own papers are removed.

Every paper in the sample carries the researcher's grade on a scale from 0 to 4 for relevance to their own work (Appendix B). The top grade marks a paper squarely on one of the researcher's research threads, one they would certainly read in full. The 218 papers that received it, 8.5% of the sample, are the relevant papers. Ninety papers appeared twice in the grading sheets, forty within the first of the two batches and fifty across them, and each received the same grade both times. The profile uses only the researcher's papers dated before the pool begins.

We quantify how well a ranked list serves the researcher with three metrics. Write $\mathcal{R}$ for the relevant papers, L_K for the first K papers of the list, and $r(p)$ for the rank of paper p in it. Papers fall into subject groups by their primary arXiv category, and $\mathcal{R}_g$ holds the relevant papers of group g. The set G keeps the groups with at least five relevant papers, leaving out a residual group of papers whose primary category fits no other group. These five groups hold 210 of the 218 relevant papers. Then

$$\begin{aligned} \mathrm{R@}K &= \frac{|L_K \cap \mathcal{R}|}{|\mathcal{R}|}, & \mathrm{GAR@}K &= \frac{1}{|G|}\sum_{g\in G}\frac{|L_K \cap \mathcal{R}_g|}{|\mathcal{R}_g|}, \\ \mathrm{nDCG@}K &= \frac{1}{Z_K}\sum_{p\in L_K\cap\mathcal{R}}\frac{1}{\log_2(r(p)+1)}, & Z_K &= \sum_{i=1}^{K}\frac{1}{\log_2(i+1)}. \end{aligned} \tag{2}$$

R@K is the share of the top-grade papers that the first K places hold. GAR@K, the group-averaged recall, computes the same share within each group and averages it over groups. A list concentrated in one group can score well on R@K while scoring poorly on GAR@K. nDCG@K measures whether the relevant papers sit near the top. Its normalizer Z_K is the standard ideal DCG, the value of the ideal list. Since the sample holds 218 relevant papers and we report nDCG only for $K \le 100$, the ideal list has a relevant paper in all K places, so nDCG@K runs from 0 to 1.

Intervals are 95% bootstrap intervals over papers. The sample is resampled 2,000 times, and every list and selection rule is rerun on each resample. The interval of a score, or of a paired difference between two lists, spans the 2.5th to 97.5th percentiles over the resamples. Part of the profile is written by a language model, so we build it five times independently and report the mean.

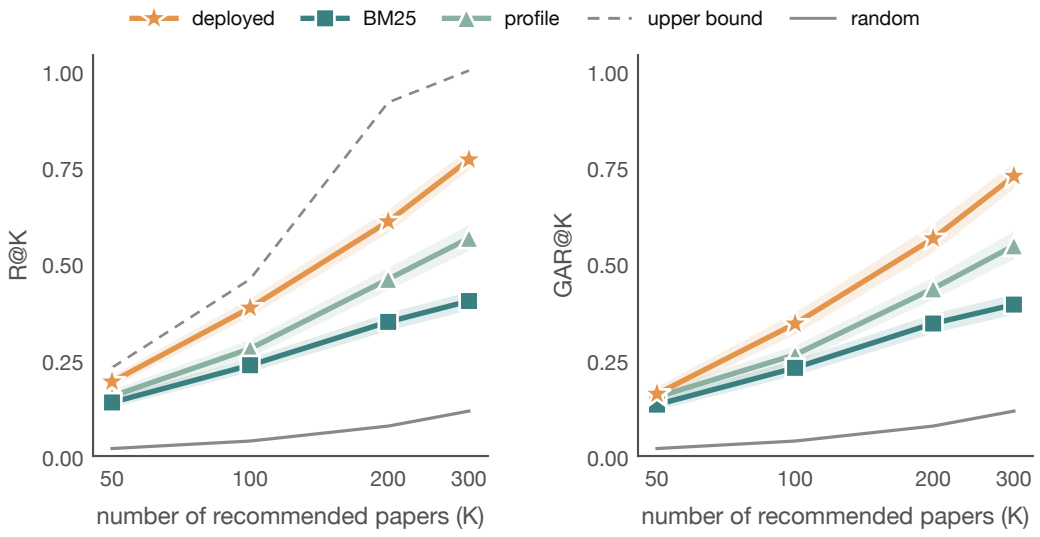


Figure 2: R@K (left) and GAR@K (right) against the number of recommended papers K (log scale, up to 300). Curves show the deployed configuration (in-sample), the BM25 signal alone, the profile signal alone, and a random order (gray). The single-signal curves average their variants. The dashed line on the left is the most any list of length K can hold. Shading is one bootstrap standard deviation over papers.

4.2 Retrieval

Fusing the seven signals of Section 3.2 determines which papers reach the researcher. Since every paper carries the researcher's grade, every fusion can be scored. Each signal is off or set to one of its variants; for the threads signal, these include four ways to split its vote across threads. This gives 1,799 configurations.

We choose by GAR@100 because a researcher works in several subject groups and a list can reach a high recall by filling up on the largest group while missing the smaller ones. GAR@100 rewards a list that serves every group. R@100 also matters, and it breaks the ties that GAR@100 cannot separate. A fixed rule selects the deployed configuration. Among the configurations that include the threads signal, it keeps those whose GAR@100 trails the best by at most one standard error of their paired difference from it. From these it takes the one with the highest R@100. Sixty-four configurations meet this rule; 92% include the category signal and 66% include the BM25 signal. For the selection we report cross-fitted numbers. We split the pool into odd and even weeks, select on each half, score on the other, and pool the scores.

Cross-fitted, the selection reached R@100 of 0.382 [0.295, 0.412] and GAR@100 of 0.331 [0.262, 0.381]. In-sample (Figure 2), the BM25 signal reached R@100/GAR@100 of 0.236/0.229, and the profile signal reached 0.278/0.263. A random order reached 0.039 on both metrics. The paired-difference intervals for the two single-signal baselines against the in-sample deployed configuration excluded zero. A list of 100 can hold at most 100 of the 218 relevant papers, a recall of 0.459. In-sample, 84 of the reading list's 100 papers were relevant, and the five profile builds' top-grade counts differed by at most two papers. The first 300 papers of the fused order held 77% of the relevant papers and formed the reranker input (Section 4.3).

Because the configurations form a full Cartesian product, every configuration that lacks a signal has a twin that adds it. Figure 3 reports the in-sample mean difference over these pairs. The threads signal added 0.027 to R@100 and 0.020 to GAR@100, and the category signal added 0.026 and 0.018, respectively. The BM25 signal added 0.003 and 0.005. The profile signal, the PRF signal, the citation signal, and the ref signal changed either metric by less than 0.01. The best two-signal configuration used the threads signal and the category signal and reached an in-sample GAR@100

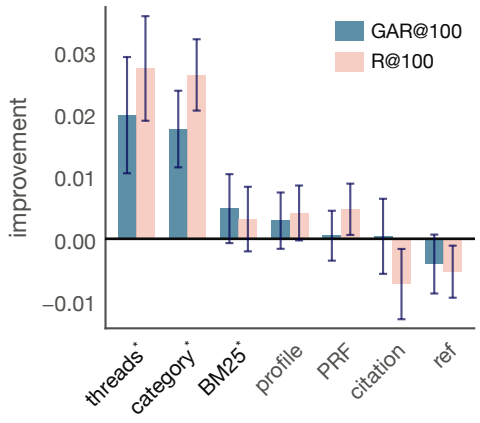


Figure 3: Mean change in GAR@100 (blue) and R@100 (pink) when a signal is added to a configuration that lacks it. The mean is over all such pairs and the signal's variants. Error bars are one bootstrap standard deviation over papers. Stars mark the three signals the system deploys.

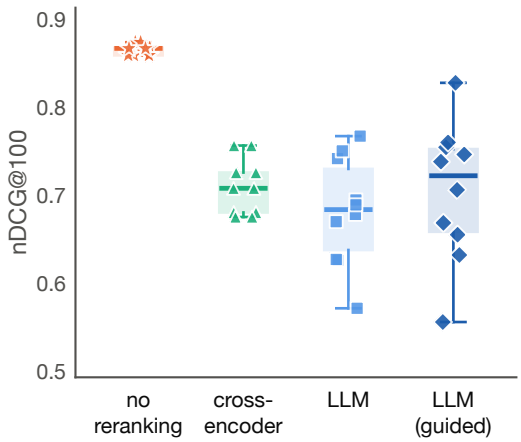


Figure 4: nDCG@100 of the 100 papers each arm returns from the first 300 of the fused order. Each marker is one run on one profile build, two runs per build for every arm. Boxes show the median and quartiles, and whiskers span all runs.

of 0.324. The best three-signal configuration, which is deployed, reached 0.343. No configuration with four or more signals exceeded 0.344, and the best with all seven reached 0.317.

4.3 Reranking after retrieval

Each deployed signal compares a candidate with just one piece of the profile (a thread, a phrase list or a category count), and the fusion sees only their ranks. A model that reads each abstract against the whole profile could choose the reading list better, and the fused order leaves plenty to choose from. In-sample, its first 300 papers held 165 to 169 relevant papers, against 84 in its first 100.

We gave the first 300 papers of the fused order to a reranker and compared the 100 it returned with *no reranking*, which simply keeps the first 100. We tried three rerankers. The *cross-encoder*, `BAAI/bge-reranker-v2-m3` (built on BGE-M3; Chen et al., 2024), scores each candidate against one query per research thread and keeps its best score. *LLM* and *LLM (guided)* each make a single call to a Claude Sonnet model, which reads the profile and the 300 candidates in fused order and returns 100 ids. LLM states the task in one sentence, while LLM (guided) adds guidance on judging relevance and on treating the input order as a prior (Appendix C). We ran every arm twice on each of the five profile builds, with new random paper ids the second time. For no reranking and the cross-encoder, which are deterministic, the two runs coincided. Because a reranker changes the order of the list as well as its contents, we scored it by nDCG@100 alongside GAR@100.

All three rerankers made the reading list worse. On nDCG@100, not a single reranker run beat no reranking on the same profile build (Figure 4). No reranking averaged 0.862 in-sample, where a random order of the same 300 papers scored 0.556 and the best order 1.000. The cross-encoder fell to 0.706, LLM to 0.678 and LLM (guided) to 0.701. Their paired differences from no reranking were -0.156 [-0.248, -0.067], -0.184 [-0.260, -0.099] and -0.161 [-0.230, -0.074], and none of the intervals contained zero. Nor did those of the GAR@100 differences, -0.059, -0.067 and -0.057 (Table 3). Each reranker replaced about half of the reading list, and the trade went the wrong way. About 40 of the 50 papers it removed were relevant, but only 21 to 24 of the 50 it brought in, so the reranked list held 15 to 20 fewer relevant papers. The LLM arms also changed their picks from one run to the next. On the same profile build, two runs shared 66 of their 100 papers on average for LLM and 70 for LLM (guided).

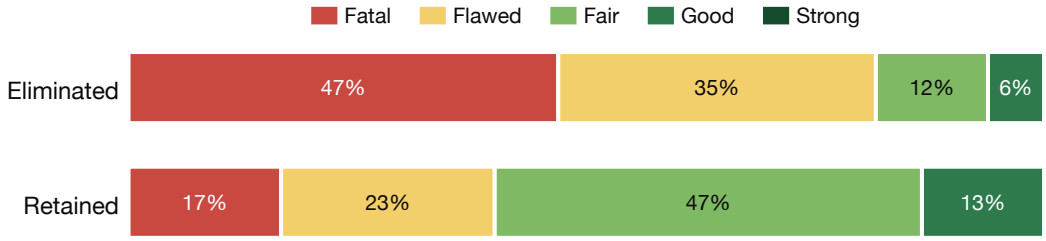


Figure 5: Grades an external grader gave to the candidate ideas the Desk Check eliminated (top bar) and retained (bottom bar), as a share of each group. The scale runs from *Fatal* (lowest) to *Strong* (highest).

The grades come from one researcher. Appendix D compares the pipeline's lists for two further researchers in applied mathematics with those of a language-model agent reading the same week-sized sample.

5 Evaluating the idea checker

The idea checker filters candidate ideas and returns evidence the researcher can act on. We assess its two levels by comparing what the Desk Check eliminates with what it retains (Section 5.1) and tracing Experiment Check decisions through recorded review rounds (Section 5.2).

5.1 What the Desk Check removes

The Desk Check filters candidate ideas before the researcher chooses one for experiment. We compared the ideas it eliminated with those it retained, using an external language model that read the evidence without being given the verdict. We ran the Desk Check on four consecutive weeks of arXiv papers, 10 August to 6 September 2026, with the profile of the researcher whose grades Section 4 uses. It gave a verdict on 79 candidate ideas drawn from 37 seed papers, retaining 30 and eliminating 49. The model graded the candidate ideas on a five-point scale from one that cannot work as stated to one worth writing up (Appendix E).

Of the 49 candidate ideas the Desk Check eliminated, 82% were graded *Fatal* or *Flawed*, the two lowest grades, against 40% of the 30 it retained (Figure 5). The mean grade on the 0–4 scale was 0.78 for the eliminated ideas and 1.57 for the retained ones, a difference of 0.79 ($p = 0.0003$, one-sided permutation test). Keeping only the retained ideas raised the mean from 1.08 over all 79 to 1.57. The Desk Check thus removed, for the most part, the ideas the grader found fatally or seriously flawed. The typical retained idea was graded *Fair*, one that could work but whose contribution or premises remain open. The researcher chooses which retained idea enters the Experiment Check, where those premises are tested. These grades come from a model outside the system and may differ from a researcher's assessment.

5.2 Three runs of the Experiment Check: pass, fail, and flag

Each review round of the Experiment Check ends in a verdict given by a fixed rule (Section 3.5), and the verdict decides what reaches the researcher. One review round from each of three real runs, one run for each verdict, shows how the rule treats rounds in which the reviewers disagree or keep an idea with reservations (Figure 8 in Appendix F).

Pass. The idea is to put error bars on the molecular energies an AI model predicts. Of four claims tested in the run, one is met. The error bar holds for 90% of molecules, as targeted. The other three are not met. The weighting the idea was built on does not make the bar hold; the bar fails on unfamiliar molecules and does not carry over to a second model. In the third round, the last the loop allows, all three reviewers vote *keep*. One notes that the numbers check out but one cost explanation does not. Another notes that the write-up states its limits but skips a cheap ensemble baseline. The

third requests a clearer comparison with standard calibration. With no reject, the rule passes the idea and hands it back to the researcher with the findings and three next steps (a harder benchmark, recalibration for unfamiliar molecules, and a comparison with a model ensemble). The researcher answers the round's one point, the missing ensemble baseline, by naming it in the write-up and ruling it out.

Fail. The idea is to make survival predictions with reliable error bars in trials where patients drop out. Of two claims fixed before the experiments, one is met (the method beats the standard one when each patient's dropout chance is known). The other is not (the win does not hold when that chance must be estimated from little data). In the first round, two of the three reviewers vote *reject*. One says the new method is the standard one with different weights. The other says real trials drop patients in ways the experiments never test. The third votes *keep*, noting that the win follows from an existing theorem. The two rejects fail the idea, which is eliminated with the reasons recorded. The researcher is not asked to decide anything.

Flag. The idea is to put error bars on an AI model's score from a short adaptive test. Two of the three claims fixed before the experiments are met. On the one benchmark admitted by a screening test fixed with the claims, the error bar is right 90% of the time. An early-stopping rule keeps it right. The third is not (the bar does not hold on the unluckiest of ten data splits). In the second round, all four reviewers vote *keep* but raise two points that only the researcher can settle. The early-stopping result rests on only the 45% of models that ever reach a stop, and the one benchmark kept passes the screening test by a margin within noise. The rule flags the idea and puts both points to the researcher. The default is the reviewers' recommendation, to keep the claims and state both limits. The researcher accepts that recommendation and declines the further experiments offered as an option. The claims, reworded to state both limits, go back into the loop.

6 Conclusion

Language models can generate research ideas in quantity. For an individual researcher, two things are missing: the ideas are not conditioned on that researcher's own record and on the papers appearing that week, and they arrive unchecked. ARXIDEA supplies both. From the researcher's publication record it builds a profile and ranks one week of arXiv against it. It proposes ideas from the papers it selects and checks each at the desk against the source papers and prior work. The Experiment Check then tests the idea the researcher chooses. Every stage reads and writes a shared memory, and no agent approves its own work.

We evaluated the recommendation on the researcher's grades and the checker on recorded evidence. In a week-sized sample in which every paper carries the researcher's grade, 84 of the first 100 papers of the fused ranking carried the top grade. Rerankers made the list worse, and a frontier agent reading the sample in place of retrieval only drew level while spending 15 to 29 million tokens per week. Ranking the week therefore calls no language model. Over four weeks, a language model outside the system found most of the ideas the Desk Check eliminated fatally or seriously flawed, and the retained ideas scored higher on average. One recorded review round from each of three Experiment Check runs shows how the rule treats reviewer objections and the points that only the researcher can settle.

An idea that fails a check is eliminated, with the reason recorded. An idea that passes reaches the researcher with the evidence, the findings, and the next steps. The decision to pursue it stays with the researcher, who can trace it from its seed paper through every check. Seen this way, the system is a front end for research rather than a research system. Today its output goes to a person. If autonomous research becomes reliable enough to trust, the same output could go to such a system instead.

REPRODUCIBILITY STATEMENT

On acceptance we will release the code of the system and of the evaluation, every agent's prompt templates, and the pool of 2,574 arXiv identifiers with the researcher's grades. We will also release the definitions of the 1,799 retrieval configurations of Section 4.2 and the scripts that rank the pool with them, so that every ranking in Section 4 can be rescored on the same grades. Appendix C gives the reranker instructions. Appendix D gives the whole-week agent's instructions and the judge's rubric, and Appendix E gives the grader's rubric; omissions are marked. Appendix G lists the fixed settings, the seeds of the bootstrap and permutation tests, and the version string of every model by stage. The profile is built from the researcher's public publication record with the code and prompt templates we will release. The names of the three researchers, the mapping from their codes to people, and the blinding keys of the graded lists are not released.

ETHICS STATEMENT

One researcher graded public arXiv papers for relevance to their own work; the grades record what that researcher would read, not the quality of the papers, and we will release them with the researcher's consent. The two further researchers of Appendix D appear only under codes; the system read only their public publication records, and their profiles are not released. The system reads public arXiv metadata and full texts, a scholarly index, and the web, and the experiments of the Experiment Check ran on public datasets or simulated data. The researcher also confirmed the publication list for the profile and answered review points in the Experiment Check. The system proposes ideas but does not act on them: surviving ideas reach the researcher with findings and next steps, and the researcher decides what to pursue.

AI USE STATEMENT

Language models run the system this paper studies. Claude Sonnet and Opus models run every agent in Figure 1: the agent that writes the profile, the agents that check candidate seed papers and propose ideas, the agents of the Desk Check's premise, theory, and novelty checks, and the experiment, audit, and review agents of the Experiment Check. Ranking the week calls no language model. In the evaluation, the language-model rerankers of Section 4.3 use a Claude Sonnet model, and the whole-week agent of Appendix D is Claude Opus 5.5. The grader of Section 5.1 and the judge of Appendix D are GPT-6 Sol, from a different model family than the models whose output they grade. Version strings are listed in Appendix G.

In this work, we used generative AI tools (Claude and Codex) to implement methods, namely the system and the evaluation code, and to clean and reformat datasets. The authors formulated the hypotheses, designed the evaluation, and interpreted the results; generative AI tools were used to draft and critique evaluation plans (design or provide feedback on research methodology or experiments), to draft candidate hypotheses that the authors kept, rewrote, or dropped (propose or refine hypotheses), and to draft analyses of results that the authors checked against the data (interpret results). We have not used generative AI tools to generate synthetic data sets or to assist with translation, and the development of theoretical models, the formulation and proof of mathematical claims, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, to create or modify scientific figures, to draft the text of this paper in English from the authors' outlines and results, to edit it for readability, to identify and summarize existing literature, and to format references. We have reviewed all AI-assisted work. The authors designed the study and ran the experiments; the release package recomputes the reported results from the graded pool and the recorded runs and compares them with the packaged values; and the authors checked every number in the text against its source. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Mariam Arustashvili and Krisztian Balog. SciNUP: Natural language user interest profiles for scientific literature recommendation. In *Advances in Information Retrieval: 48th European Conference on Information Retrieval, ECIR 2026, Proceedings, Part IV*, Lecture Notes in Computer Science, pp. 432–446. Springer, 2026. doi: 10.1007/978-3-032-21321-1_51. URL https://arxiv.org/abs/2510.21352.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Yiqun Chen, Qi Liu, Yi Zhang, Weiwei Sun, Xinyu Ma, Wei Yang, Daiting Shi, Jiaxin Mao, and Dawei Yin. TourRank: Utilizing large language models for documents ranking with a tournament-inspired strategy. In *Proceedings of the ACM on Web Conference 2025*, pp. 1638–1652, 2025. doi: 10.1145/3696410.3714863. URL https://arxiv.org/abs/2406.11678.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. doi: 10.1145/1571941.1572114.

Markus Flicke, Glenn Angrabeit, Madhav Iyengar, Vitalii Protsenko, Illia Shakun, Jovan Cicvaric, Bora Kargi, Haoyu He, Lukas Schuler, Lewin Scholz, Kavyanjali Agnihotri, Yong Cao, and Andreas Geiger. Scholar Inbox: Personalized paper recommendations for scientists. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 307–317. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-demo.30. URL https://aclanthology.org/2025.acl-demo.30/.

Kristian Gingstad, Øyvind Jekteberg, and Krisztian Balog. ArXivDigest: A living lab for personalized scientific literature recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 3393–3396, 2020. doi: 10.1145/3340531.3417417. URL https://arxiv.org/abs/2009.11576.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R. D. Costa, José R. Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Accelerating scientific discovery with Co-Scientist. *Nature*, 655(8122):487–496, 2026. doi: 10.1038/s41586-026-10644-y. URL https://arxiv.org/abs/2502.18864.

Sy-Tuyen Ho, Minghui Liu, Huy Nghiem, and Furong Huang. SoundnessBench: Can your AI scientist really tell good research ideas from bad ones?, 2026. URL https://arxiv.org/abs/2605.30329.

Christin Katharina Kreutz and Ralf Schenkel. Scientific paper recommendation systems: a literature review of recent publications. *International Journal on Digital Libraries*, 23(4):335–369, 2022. doi: 10.1007/s00799-022-00339-w. URL https://arxiv.org/abs/2201.00682.

Jiaqi Liu, Shi Qiu, Mairui Li, Bingzhou Li, Haonian Ji, Siwei Han, Xinyu Ye, Peng Xia, et al. AutoResearchClaw: Self-reinforcing autonomous research with human-AI collaboration, 2026a. URL https://arxiv.org/abs/2605.20025.

Xiang Liu, Qumeng Sun, Bruno, Xiao Zhang, Jiahao Li, and Haoyang Zhang. AutoResearch: Insight in, hallucination out, 2026b. URL https://arxiv.org/abs/2608.17906.

Meixiu Long, Duolin Sun, Dan Yang, Yihan Jiao, Lei Liu, Jiahai Wang, Binbin Hu, Yue Shen, Jie Feng, Zhehao Tan, Junjie Wang, Lianzhen Zhong, Jian Wang, Peng Wei, and Jinjie Gu. GroupRank: A groupwise paradigm for effective and efficient passage reranking with LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25165–25186. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1260. URL https://aclanthology.org/2026.findings-acl.1260/.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651(8107):914–919, 2026. doi: 10.1038/s41586-026-10265-5.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. Zero-shot listwise document reranking with a large language model, 2023. URL https://arxiv.org/abs/2305.02156.

Rui Meng, Bhavana Dalvi Mishra, Jiefeng Chen, Chun-Liang Li, Palash Goyal, Mihir Parmar, Yiwen Song, Yale Song, Rajarishi Sinha, Parthasarathy Ranganathan, Burak Gokturk, Jinsung Yoon, and Tomas Pfister. ScientistOne: Towards human-level autonomous research via chain-of-evidence, 2026. URL https://arxiv.org/abs/2605.26340.

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C. Landsness, Daniel L. Barabasi, Siddharth Narayanan, Nicky Evans, Shriya Reddy, Martha Foiani, Aizad Kamal, Leah P. Shriver, Fang Cao, Asmamaw T. Wassie, Jon M. Laurent, Edwin Melville-Green, Mayk Caldas, Albert Bou, Kaleigh F. Roberts, Sladjana Zagorac, Timothy C. Orr, Miranda E. Orr, Kevin J. Zwezdaryk, Ali E. Ghareeb, Laurie McCoy, Bruna Gomes, Euan A. Ashley, Karen E. Duff, Tonio Buonassisi, Tom Rainforth, Randall J. Bateman, Michael Skarlinski, Samuel G. Rodriques, Michaela M. Hinks, and Andrew D. White. Kosmos: An AI scientist for autonomous discovery, 2025. URL https://arxiv.org/abs/2511.02824.

Bo Ni, Franck Dernoncourt, Hongjie Chen, Yu Wang, Nesreen K. Ahmed, Zhengzhong Tu, Tyler Derr, and Ryan A. Rossi. Personalized auto-research: Towards a true AI co-scientist, 2026. URL https://arxiv.org/abs/2608.14881.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 68772–68802, 2024. URL https://arxiv.org/abs/2404.13076.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent Laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.320/.

Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. SciRepEval: A multi-format benchmark for scientific document representations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5548–5566, 2023. doi: 10.18653/v1/2023.emnlp-main.338.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? Investigating large language models as reranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.923/.

Qiong Tang, Tianxiang Sun, Xiangkun Hu, Xiangyang Liu, Yiran Chen, Yunfan Shao, et al. FARS: A fully automated research system deployed at scale, 2026. URL https://arxiv.org/abs/2606.31651.

Chengcheng Wang, Qinhua Xie, Wei He, Jianyuan Guo, Shiqi Wang, and Chang Xu. Sibyl-AutoResearch: Autonomous research needs self-evolving trial-and-error harnesses, not paper generators, 2026a. URL https://arxiv.org/abs/2605.22343.

Fuqiang Wang, Song Tan, Zheng Guo, Jiaohao Fu, Xinglong Xu, Bihui Yu, Jie Dong, Zheng Sun, Siyuan Li, Jingxuan Wei, and Cheng Tan. PaperFlow: Profiling, recommending, and adapting across daily paper streams, 2026b. URL https://arxiv.org/abs/2606.07454.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066.

Zonglin Yang, Xingtong Liu, and Xinyan Xu. SciIntegrity-Bench: A benchmark for evaluating academic integrity in AI scientist systems, 2026. URL https://arxiv.org/abs/2605.10246.

Lezhi Yu, Xiaogang Xu, Yuhua Zhou, Shuibing He, and Aimin Pan. Beyond execution: Auditing experimental fidelity in LLM-driven scientific research, 2026. URL https://arxiv.org/abs/2608.26753.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 38–47, 2024. doi: 10.1145/3626772.3657813. URL https://arxiv.org/abs/2310.09497.

A System details

The stages pass recorded outputs from one to the next. Figure 6 shows the steps behind those outputs, extending Section 3.

Recommendation. One agent and several programs prepare the inputs of Section 3.2. The agent builds the profile from the researcher's publication list, gathered from Google Scholar and arXiv and confirmed by the researcher. Programs supply each paper's abstract and arXiv categories. The agent's note on each paper names the paper's field, its core method, and up to five key phrases that appear verbatim in its title or abstract. Each research thread the agent forms has a name, a summary, and its member papers. A program fetches the week's new papers from arXiv.

Ideation. The three checks of Section 3.3 run in parallel on every paper of the reading list, each by its own agent. A program then assigns each paper to the research thread under which it ranks highest and picks the seeds. A thread whose papers all fail the checks has no seed that week.

Desk Check. The three checks of Section 3.4 run in order, each by its own agent. The premise check runs the tests registered in Ideation that can be completed without experiments. The novelty check also merges near-duplicate candidates. A program replays completed tests and searches before the verdict is written.

Experiment Check. One agent writes the research specification described in Section 3.5; another runs the experiments. A program logs every number into the registry. When a round is flagged, the workflow resumes at the authoring step after the researcher's answer.

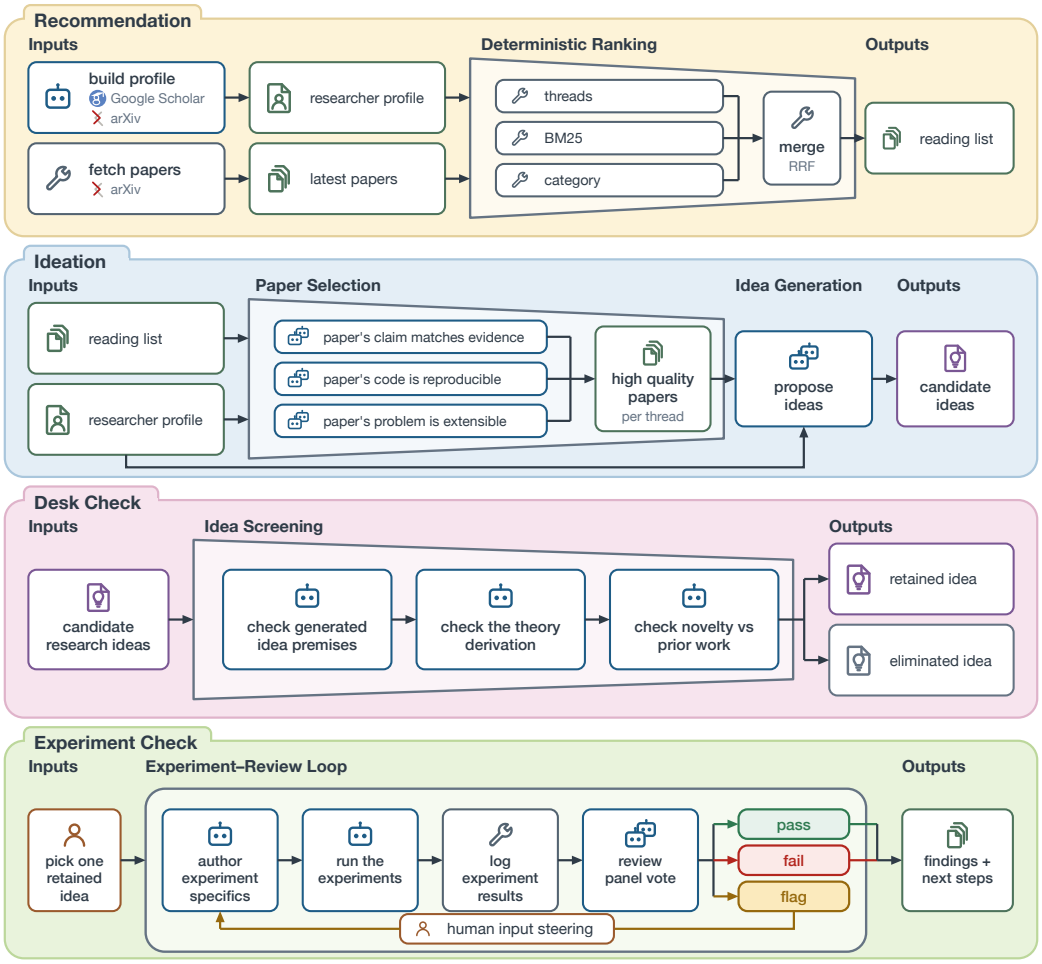


Figure 6: The four stage workflows of Figure 1, one band per stage, each read from its inputs on the left to its outputs on the right. Icons mark one agent, several agents in parallel, a program (the wrench), and the researcher; small labels beneath steps give source names or step details.

B RETRIEVAL: GRADES, SIGNAL INVENTORY AND BEST CONFIGURATIONS

The selection of Section 4.2 depends on how the grades were collected and on which signals were available to combine. The system deploys three of the seven signals of Table 1; the other four change GAR@100 and R@100 by less than 0.01 on average (Figure 3).

The grades. The papers were presented in two batches under blinded ids, with their titles and abstracts but not their authors or arXiv ids. Every paper carries the researcher's grade. The researcher gave 2,194 of the grades one by one. The other 380 come from a rule the researcher approved before grading began: a paper receives grade 0 when its primary category is on a list of categories outside the researcher's research and none of its other categories is on a list of fields the researcher keeps. These grades are the researcher's as well, since the rule states the researcher's judgment of those categories once rather than once per paper.

B.1 SIGNAL DEFINITIONS

Reproducing the ranking requires the score functions of Section 3.2. Let C be the candidate pool, and let $e(p)$ denote the SPECTER2 embedding (Singh et al., 2023) of the title and abstract of a paper p. The profile groups the researcher's papers into T research threads. Write M_t for the set of

Table 1: The seven retrieval signals: what each reads from the profile, its variants with the short names used in Table 2, and which variant, if any, the system deploys. Every configuration of Section 4.2 sets each signal to off or to one of its variants. Excluding the configuration with every signal off, this gives $5 \cdot 5 \cdot 2 \cdot 3 \cdot 2 \cdot 3 \cdot 2 - 1 = 1{,}799$ configurations.

Signal	Reads	Variants	Deployed
threads	the member papers of each research thread and the shared references they cite, embedded with SPECTER2	how the thread rankings share the vote in Equation (1): one vote each (*each*), $1/T$ each ($1/T$), in proportion to the square root of the thread's size ($\sqrt{\ }$), or one ranking by the best thread rank (*best*)	$1/T$
BM25	the profile's key phrases, matched in a paper's title and abstract	the phrase list: written by the agent in the paper notes (*b1*); those of them that appear in at least two of the researcher's papers (*b2*); the agent's and the tools' phrases together (*b3*); the tools' phrases alone (*b4*)	b3
category	the arXiv categories of the researcher's papers, weighted by how many carry each	one	yes
profile	an embedding of the profile's text	the thread summaries and the recent-work note as one query (*d0*); the mean of one "field. method" line per paper note (*dN*)	no
PRF	a time-weighted centroid of the researcher's own papers, each weighted by a factor that halves every 24 months	one	no
citation	the citation graph built from the researcher's papers	personalized PageRank from the researcher's papers (*graph*); the number of references a candidate shares with them (*coupling*)	no
ref	the references cited by at least two of the researcher's papers, with their abstracts	one	no

member papers of thread t, and F for the set of references cited by at least two of the researcher's papers.

The threads signal. It produces one ranking per thread. For thread t let $Q_t = M_t \cup \{\, f \in F : f \text{ is cited by a paper in } M_t \,\}$. Every candidate $p \in C$ receives the score

$$s_t(p) = \max_{q \in Q_t} \cos\big(e(p), e(q)\big), \tag{3}$$

and the ranking sorts C by s_t in decreasing order, so it contains every paper in C. The variants scored in Section 4.2 differ in the weight the T rankings receive in Equation (1). Each ranking receives one vote (*each*), $1/T$ of a vote ($1/T$, deployed), or a weight proportional to the square root of its thread's size ($\sqrt{\ }$). The fourth variant (*best*) uses a single ranking that orders papers by their best thread rank.

The BM25 signal. The query terms are the profile's key phrases, matched as whole phrases in a paper's title and abstract, with BM25 at its default parameters ($k_1 = 1.5$, $b = 0.75$). The ranking contains the papers with a positive score, in decreasing order. Variants differ in the phrase list. Variant *b1* uses the phrases the agent wrote in the paper notes, and *b2* keeps those among them that appear in at least two of the researcher's papers. Variant *b3* (deployed) uses the union of the agent's and the tools' phrases, and *b4* the phrases the tools extracted.

The category signal. Each arXiv category c receives the weight w_c, the number of the researcher's papers that carry c. A paper p with categories $\mathrm{cat}(p)$ scores $\sum_{c \in \mathrm{cat}(p)} w_c$. The papers with a positive score are ranked in decreasing order, and tied papers share the mean of the ranks their tie spans.

Table 2: The ten configurations with the highest GAR@100 among the 1,799, scored in-sample on the researcher's grades as the mean over the five profile builds. Signal columns: the variant of Table 1 that the configuration uses, ✓ for a single-variant signal that is on, and – for a signal that is off. The star marks the deployed configuration. R@300: recall of the first 300 papers of the fused order, the reranker input of Section 4.3.

	threads	BM25	category	profile	PRF	citation	ref	GAR@100	R@100	nDCG@100	R@300
1	$1/T$	b3	✓	–	–	–	✓	0.344	0.368	0.833	0.732
2*	$1/T$	b3	✓	–	–	–	–	0.343	0.384	0.862	0.769
3	$1/T$	b4	✓	–	–	–	✓	0.336	0.366	0.832	0.739
4	$1/T$	b4	✓	–	–	–	–	0.335	0.381	0.857	0.763
5	best	b4	✓	–	–	–	–	0.333	0.376	0.849	0.729
6	$1/T$	b3	✓	d0	–	–	–	0.332	0.366	0.828	0.771
7	$1/T$	b4	✓	d0	–	graph	–	0.330	0.350	0.799	0.730
8	$1/T$	b2	✓	–	✓	graph	–	0.327	0.332	0.776	0.672
9	$1/T$	b4	✓	–	–	graph	–	0.327	0.361	0.821	0.761
10	$1/T$	b4	✓	–	–	graph	✓	0.327	0.343	0.788	0.716

The other four signals. The profile signal scores a paper by cosine similarity to a profile-derived embedding. That embedding comes from either a single passage of thread summaries and the recent-work note (*d0*) or the mean of one "field. method" embedding per paper note (*dN*). The PRF (pseudo-relevance feedback) signal scores a paper by cosine similarity to a time-weighted centroid of the embeddings of the researcher's own papers, each weighted by a factor that halves every 24 months. The citation signal scores a paper either by personalized PageRank from the researcher's papers over the citation graph (*graph*) or by the number of references it shares with them (*coupling*). The ref signal scores a paper by its highest cosine similarity to the references in F.

B.2 The best configurations

The top ten configurations show which signal choices recur at high GAR@100 (Table 2). The first row's GAR@100 has a 95% bootstrap interval of [0.287, 0.394]. All ten fuse the threads, BM25, and category signals. Nine give each thread $1/T$ of the vote and one ranks by the best thread rank, and the BM25 phrase list is one of b2, b3 and b4. Seven of the ten add one or two of the other four signals. The deployed configuration, selected by the rule of Section 4.2, is second by GAR@100 and first by R@100 and nDCG@100.

C Reranking after retrieval: results and prompts

In the evaluation, the fused ranking's first 300 papers include top-grade papers beyond the reading list. Reranking tests whether those papers enter the first 100.

Reranking changes list membership as well as order. Table 3 reports relevant-paper counts and GAR@100 alongside the nDCG@100 in Figure 4 for the four arms of Section 4.3. The 95% interval of every difference from no reranking excludes zero. On nDCG@100, no single reranker run beats no reranking on the same profile build. The instructions below document the two language-model arms.

Cross-encoder. The cross-encoder, `BAAI/bge-reranker-v2-m3`, uses BGE-M3 as its base model (Chen et al., 2024). Its queries are one line per research thread, the thread's name followed by its summary and its key phrases. The cross-encoder scores each candidate's title and abstract against every query and keeps its highest score across threads. The 100 highest-scoring candidates are returned in that order.

Language-model rerankers. Each run of a language-model arm is one call to Claude Sonnet (`claude-sonnet-5`), which reads the whole prompt from one file and writes the 100 ids to another. A prompt has four parts: the instruction, the profile, the 300 candidates, and the output rule. The instruction of *LLM* reads, in full:

Table 3: The four arms of Section 4.3, each returning 100 papers from the first 300 of the fused ranking. Mean: over the ten runs of an arm (five profile builds, two runs each; the two runs of no reranking and of the cross-encoder coincide). Δ: paired difference from no reranking, with a 95% bootstrap interval over papers. *Relevant papers*: the top-grade papers among the 100 returned, the count behind R@100.

	nDCG@100		Relevant papers		GAR@100	
Arm	Mean	Δ [95% interval]	Mean	Δ [95% interval]	Mean	Δ [95% interval]
no reranking	0.862		83.8		0.343	
cross-encoder	0.706	-0.156 [-0.248, -0.067]	68.4	-15.4 [-25.4, -4.6]	0.284	-0.059 [-0.114, -0.008]
LLM	0.678	-0.184 [-0.260, -0.099]	63.6	-20.2 [-28.6, -9.1]	0.277	-0.067 [-0.109, -0.018]
LLM (guided)	0.701	-0.161 [-0.230, -0.074]	66.2	-17.6 [-25.4, -6.6]	0.286	-0.057 [-0.097, -0.008]

```
Read the researcher profile and the candidate papers (id,
title, abstract, categories).  Rank the candidates by how
relevant each is to this researcher's research directions,
most relevant first.  Output the ids of the 100 most
relevant papers in order, one per line.
```

The instruction of *LLM (guided)* reads, in full:

```
Read the researcher profile and the candidate papers (id,
title, abstract, categories).  Select and rank the 100
papers this researcher would most want to read for their
own work.

How to judge relevance:
- A paper is relevant when it advances, applies, or directly
challenges one of the researcher's threads, or when it
brings the researcher's methods and framing (as described
in the profile text, not only in the thread keywords) to a
neighbouring problem.  Judge each paper against the whole
profile, not against keyword overlap.
- Sharing an arXiv category or a broad topic word with
a thread is not enough.  Within a large research area,
separate the papers that touch the researcher's specific
questions from those that merely belong to the same area.
- The candidates are listed in the order produced by a
first-stage retrieval system, best first.  Treat that order
as a reasonable prior:  keep it where you have no specific
reason to disagree, and move a paper up or down only when
its abstract gives you one.

Output the ids of the 100 most relevant papers in order, one
per line.
```

The profile part is the same for both arms and sits under the heading `Researcher profile`. It opens with a paragraph on the researcher's work and gives one subsection per research thread with its summary and the titles of its member papers. The note on recent work comes next, followed by the key phrases of each thread. The candidate part lists the 300 papers in the fused order, each under a blinded id with its title, its arXiv categories, and its abstract. The two runs of an arm on the same profile build use different blinded ids. The output rule reads:

```
Write your answer to [the run's result file] as plain text, the ids
of the 100 most relevant papers in order, one per line.
Nothing else in the file:  no headings, no commentary, no
code fences, no blank lines.  (...)
```

Every run returned 100 distinct valid ids. The prompt templates and blinded outputs of every run will be released with the code.

D RETRIEVAL AGAINST AN AGENT READING A WEEK-SIZED SAMPLE

Retrieval ranks a week-sized sample from paper titles, abstracts and arXiv categories without a language-model call. A frontier agent reading the same sample tests whether direct reading improves the list. We ran that comparison for three researchers in applied mathematics: R1, whose grades define the evaluation set of Section 4, and two further researchers, R2 and R3 (Table 4). All profiles are built from papers dated before the pool, all three researchers are evaluated on the same 2,574 papers, and Figure 7 holds the result.

Table 4: The three researchers. Papers: the number of papers in the publication record the profile is built from, all dated before the pool. Years: the span of that record.

	Papers	Years
R1	40	2019–2026
R2	45	2008–2026
R3	61	2013–2026

R3 has no paper and no co-author in common with R1 or R2, and none of R3's papers is in the sampled pool.

D.1 THE AGENT

The comparison tests reading against matching, so the agent has to choose every paper by reading it. The agent is Claude Opus 5.5 running in Claude Code. Each run is a fresh session started in a folder that holds only the files below. `papers.md` lists the 2,574 papers of the pool, each with its title and abstract, in a random order under blinded ids (`Q0001`, ...). The ids are the same in a researcher's two inputs, described below. The agent decides for itself how to read them; it may start up to three helper sub-agents, which can only read the files in the folder. The folder's rule file reads:

```
Do not operate the computer.  No computer use / screen
control, no screenshots, no opening applications, browsers
or anything outside this folder.  Do not use the web.  Do
not try to find out who the researcher is, who wrote a
paper, or which real paper an id refers to.  Read only
prompt.md, researcher.md, papers.md and task.md.  Write only
under out/.  Every choice of paper must come from a model
reading the papers:  no code, keyword search or script that
scores, filters or ranks papers.  Subagents:  at most 3 in
total, and only of type helper (...).
```

The task file (profile input; the one-sentence version omits `researcher.md` and the categories):

```
A researcher typed the request in prompt.md and provided
researcher.md.  papers.md holds all 2574 new arXiv papers
of this week (title, arXiv categories and abstract each,
ids like Q0001).  Do what the researcher asked:  return the
100 papers from papers.md they would be most interested in,
ranked best first.  How you go about it is up to you.  You
may spawn a very small number of subagents (at most 3 in
total, type helper) if you find it necessary.  (...)
```

A session ends by writing 100 distinct ids, whose format a validator checks.

Two inputs. The *one-sentence* input is what a researcher would type, and it names no thread, method or keyword taken from the profile. It opens with "I work on ...", naming three broad areas of the researcher's work in the words a researcher would use. It closes with "From these papers, return the 100 I'd be most interested in, ranked best first." The *profile* input gives the agent the information the retrieval stage reads and nothing else. Its request reads: "`researcher.md` describes my research: my own papers in groups, references that my papers share, key phrases, and the arXiv categories I publish in. From these papers, return the 100 I'd be most interested in,

ranked best first." The file `researcher.md` is rendered from the same profile that the deployed signals read. For each research thread it lists the member papers as "title. abstract" and the ids of the references that at least two of the researcher's papers cite. It then lists those references once each, the key phrases the BM25 signal searches for, and the number of the researcher's papers in each arXiv category. Because the category signal reads a candidate's categories, the profile input also adds them to `papers.md`. Thread names, the profile's prose summary and its note on recent work are left out, because retrieval does not read them. Where an abstract of the researcher's own paper names the researcher, the name is replaced by "[author]". The file runs to between 158 and 323 KB, and the session logs show that every run read all thread groups, the key phrases and the categories, and a part of the shared references. One run takes 14.7 to 29.4 million tokens per run on the week-sized sample (Table 5).

Table 5: Tokens per run and sub-agents started. Tokens are in millions and count input, cache writes, cache reads and output from the session logs, with sub-agents included and a response logged more than once counted once.

	One-sentence input		Profile input	
	Tokens	Sub-agents	Tokens	Sub-agents
R1	28.1	0	29.2	0
R2	14.7	3	29.4	3
R3	22.7	3	27.9	3

D.2 THE JUDGE

R2 and R3 have not graded the pool, so a language model grades the lists of all three researchers. The judge is GPT-6 Sol in a Codex session, one session per researcher. Each session starts in a folder that holds only the rubric below, the researcher's publication list, the papers to grade (title and abstract, blinded ids, shuffled) and the task. The publication list holds every paper dated before 15 June 2026, newest first, with its title and abstract. For R1, four papers are omitted from it at R1's request. A researcher's three lists are merged, de-duplicated and shuffled, and the judge grades the union in one session without knowing which list a paper came from. A paper that appears in two lists therefore carries one grade, and a researcher's three lists are graded on one scale. The judge grades every paper itself, with no sub-agent and no code that assigns or adjusts grades, and writes integer grades only. The session logs show that each session read the whole publication list and every paper. The judge comes from a different model family than the Claude systems behind the lists; prior work reports self-preference in model-based evaluation (Panickssery et al., 2024). The agent is Claude Opus 5.5, and the profile that retrieval reads is written by Claude models. The rubric reads, leaving out the original-language wording of each grade, which the rubric declares authoritative:

> (...) *Description.* The researcher is the author of every publication in the publication list. Each week a list of new arXiv papers reaches them; for each paper they see the title and abstract only. *Narrative.* A paper is relevant to the extent this researcher would want to read it, given their own research. Grade relevance to the researcher only—not the paper's quality, not the authors' fame, not how interesting it is in general. Look nothing up; use the title and abstract only. *Scale.* (...) 4: squarely on one of the researcher's lines of work; would certainly read it closely. 3: bears directly on one of the researcher's lines; would read it. 2: same broad direction; would click on it. 1: somewhat adjacent; might glance at it. 0: nothing to do with the researcher, including papers in the same broad field but far from them. When torn between two grades, take the lower one. *Aspects.* (1) Consider the underlying intent: what does this researcher work on, and what would they be looking for in a weekly listing? (2) Measure how well the paper matches that intent. (3) Decide the final grade, an integer 0–4.

Scale and agreement with the researcher's grades. The judge used the same 0–4 grades as R1 but graded more strictly; their grades were positively associated. On 250 papers R1 had graded, fifty at each of R1's grades, the judge's mean grade is 0.02, 0.10, 0.08, 0.30 and 1.00 for R1's grades 0 to 4 (Spearman correlation 0.55). Both R1 and the judge graded the 176 papers of R1's three lists. On them, R1's mean grade rises with the judge's grade and is 1.84, 3.64, 3.94, 4.00 and 4.00 for the judge's grades 0 to 4. Every one of the 22 papers the judge graded 3 or 4 carries R1's top grade.

Of the 72 papers it graded 2 or above, 69 carry the top grade and the other three the grade below it. The judge gives most of R1's top-grade papers a 1 or a 2. The judge's mean grade also moves between sessions, by 0.1 to 0.2. The same papers graded in two independent sessions under the same rubric received the same grade in 65 to 74% of cases and a grade within one step in every case. The difference between two lists moved by at most 0.05. Grades are therefore compared only between lists graded in one session, and a paper counts as *worth reading* at grade 3 or 4.

D.3 Results

A researcher reads the top of a list first, so the comparison counts papers worth reading in the first 20 and the first 50 (Figure 7). In the first 20, retrieval holds 11, 7 and 12 papers worth reading for R1, R2 and R3; the one-sentence agent holds 8, 6 and 8, and the profile-input agent 11, 9 and 13. In the first 50, retrieval holds 14, 11 and 16, the one-sentence agent 10, 12 and 13, and the profile-input agent 17, 13 and 17. Retrieval is thus at or above the one-sentence agent everywhere except R2's first 50, where it has one paper fewer, and at most three papers below the profile-input agent.

Over the full list of 100 (Table 6), retrieval holds 19, 13 and 17 papers worth reading, the one-sentence agent 13, 13 and 17, and the profile-input agent 20, 13 and 17. For R2 and R3, each list contains every paper worth reading that any of the three found. On the mean grade of all 100 papers, retrieval minus the one-sentence agent is -0.01 [-0.15, $+0.13$], -0.44 [-0.61, -0.26] and $+0.02$ [-0.15, $+0.19$] for R1, R2 and R3, with 95% paired bootstrap intervals over papers. The same difference against the profile-input agent is $+0.16$ [-0.04, $+0.36$], -0.45 [-0.62, -0.27] and -0.16 [-0.31, -0.01]. For R2 the difference lies in what fills the list after the papers worth reading. Retrieval reaches its last one at position 77, against 54 and 28 for the two agent lists, and 44 of its 100 papers are graded 0, against 12 in each agent list.

The two agent inputs change which papers are selected. The profile input raises R3's papers worth reading in the first 20 from 8 to 13. For R1 it adds 18 papers from a field in which R1's record holds a small group of four papers. R1 grades these 18 papers 1.17 on average, and the judge, whose copy of R1's publication list omits those four papers at R1's request, grades all of them 0. Retrieval, reading the same profile, returns 4 papers from that field.

Table 6: Papers worth reading (the judge's grade 3 or 4) by position in each list of 100, and the judge's mean grade of all 100. Δ: retrieval minus the list in that row, with a 95% paired bootstrap interval over papers.

		Papers worth reading at positions						
	List	1–10	11–20	21–50	51–100	all	Mean	Δ [95% interval]
R1	Retrieval (deployed)	6	5	3	5	19	1.66	
	Agent, one-sentence input	5	3	2	3	13	1.67	-0.01 [-0.15, $+0.13$]
	Agent, profile input	8	3	6	3	20	1.50	$+0.16$ [-0.04, $+0.36$]
R2	Retrieval (deployed)	6	1	4	2	13	0.98	
	Agent, one-sentence input	5	1	6	1	13	1.42	-0.44 [-0.61, -0.26]
	Agent, profile input	7	2	4	0	13	1.43	-0.45 [-0.62, -0.27]
R3	Retrieval (deployed)	9	3	4	1	17	1.44	
	Agent, one-sentence input	6	2	5	4	17	1.42	$+0.02$ [-0.15, $+0.19$]
	Agent, profile input	10	3	4	0	17	1.60	-0.16 [-0.31, -0.01]

Check against R1's own grades. Every paper in the pool carries R1's grade, so R1's three lists can also be scored on R1's own grades (Table 7). By mean grade, R1 and the judge order the three lists the same way, with the one-sentence agent's list level with retrieval and the profile-input agent's list below it. The judge compresses the differences, with retrieval minus the profile-input agent at $+0.54$ on R1's grades and $+0.16$ on the judge's. Counting a paper as worth reading at grade 3 or 4 from either grader, the judge finds 22 among the 176 and R1 finds 151. Among the first 50 papers, the profile-input list holds the most by the judge's grades, 17 against 14 for retrieval, and the fewest by R1's, 41 against 49. One further difference is visible only in R1's grades. On the top grade alone, the one-sentence list holds 91 papers and retrieval 84.

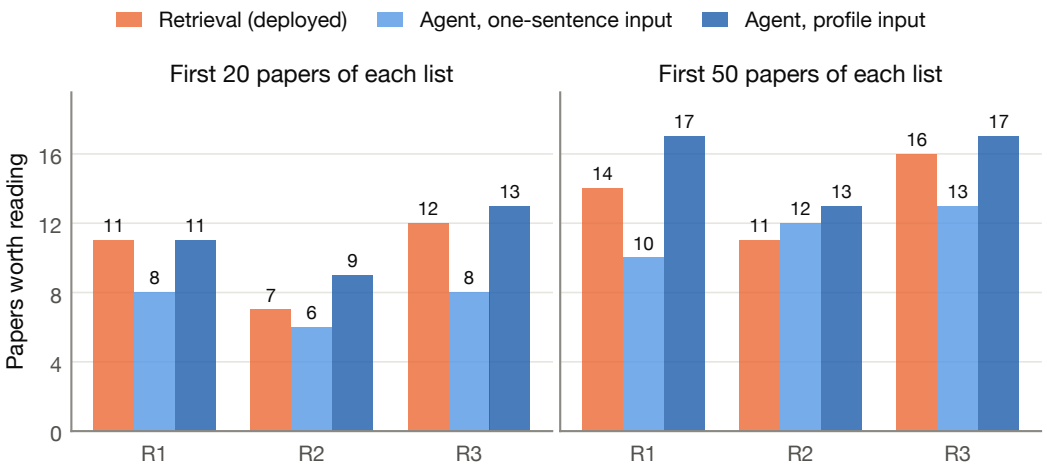


Figure 7: Papers worth reading (the judge's grade 3 or 4) among the first 20 papers (left) and the first 50 papers (right) of each list, for R1, R2 and R3. Orange shows retrieval, light blue the one-sentence agent, and dark blue the profile-input agent. The number above each bar is its count. Counts over the full list and mean grades are in Table 6.

Table 7: R1's three lists graded by R1 and by the judge, as the mean grade of the 100 papers on the 0–4 scale. Δ: retrieval minus the list in that row, with a 95% paired bootstrap interval over papers.

	R1's grades		Judge's grades	
List	Mean	Δ [95% interval]	Mean	Δ [95% interval]
Retrieval (deployed)	3.78		1.66	
Agent, one-sentence input	3.89	-0.11 [-0.24, $+0.01$]	1.67	-0.01 [-0.15, $+0.13$]
Agent, profile input	3.24	$+0.54$ [$+0.30$, $+0.80$]	1.50	$+0.16$ [-0.04, $+0.36$]

E THE DESK CHECK GRADER

The comparison in Section 5.1 rests on an external grader's assessment of the ideas. Its input and scoring protocol establish what that assessment measures.

What was graded. The 80 candidate ideas that the Desk Check processed over the four weeks of Section 5.1 were written to one file each, under a blinded id in a random order. Each file holds the abstract of the seed paper, the idea in the words of the agent that proposed it, and selected records from the initial checks. Those records include registered premise checks, executed test outputs, a written derivation, and the closest existing work found by literature search. The assembled file omits explicit verdict and reason fields, as well as the week and arXiv identifier. The Desk Check rescoped one of the 80 ideas rather than retaining or eliminating it, because its premise check found that the idea could hold but that the test proposed for it could not be run. This idea was graded with the rest and is left out of Section 5.1, which compares the remaining 79.

The grader. The grader is GPT-6 Sol (`gpt-6-sol`) in Codex sessions. It comes from a different model family than the Claude models that proposed the ideas and ran the Desk Check. Prior work documents self-preference in model-based evaluation (Panickssery et al., 2024). A coordinator session, which grades nothing and opens no case file, starts sixteen fresh sub-agent sessions of the same model, each grading a fixed batch of five ideas. The coordinator gives each grader only the one-line instruction to do the task below for its five ids, without example grades, advice on how strict or generous to be, or a target distribution. A grader may read only the rubric and its own five case files, without web access. The workspace rule requires it to grade every idea by reading it: "Grade every direction yourself, by reading it. Never write or run code that decides, filters or suggests grades." The task file reads:

```
You were given ids <FIRST>-<LAST> and a label <LABEL>.
1.  Read rubric.md in full.
2.  For each id in your range, read cases/<id>.md completely
before grading it.  Read it with sed -n in pieces of at
most 100 lines (...)  until you reach the end of the file.
If any output says it was truncated, re-read that range in
smaller pieces.  Do not open any other case, and nothing in
out/ except the file you write.
3.  Grade each direction with exactly one of:  strong,
promising, marginal, weak, unsound, following rubric.md.
Give the one or two facts that decided the grade, in at most
50 words.
4.  Write out/grades-<LABEL>.json, exactly in this form,
with every id in your range exactly once:  [the layout of the file: the batch label, the grader's exact model id, and one grade and reason per id]
5.  Reply with one line:  <LABEL> done:  <how many
directions you graded>.
```

The rubric reads, in full:

> **How to grade a research direction.** Grade the direction *as it was proposed*: is it worth a researcher's next few months, as a paper? Use everything the case shows—the seed paper's abstract, the direction's own text, and the evidence from the vetting step (premises and their registered checks, probe outputs, the written derivation, the closest existing work). Where the evidence bears on the direction, weigh it; where it is silent, judge from the direction itself and standard results you can name.
>
> Pick exactly one grade: *strong*—sound as far as the material shows, and a clear contribution: a paper worth writing as proposed. *promising*—sound as far as the material shows and worth pursuing, but the contribution is moderate or a real open risk remains. *marginal*—could work, but the contribution is thin or incremental next to the closest work, or an important premise is unsupported either way. *weak*—the material shows a serious problem (a key premise looks doubtful, the contribution is largely already done, or the plan cannot test the claim), though it is not clearly impossible. *unsound*—the material, or a standard result you name, shows the direction cannot work as stated.
>
> Judge each direction on its own, not against the other directions you were given, and do not let the length or polish of the write-up move the grade. You are not told what happened to any direction afterwards; do not guess.

Grade names and the test. The five grades are reported in Section 5.1 and Figure 5 under the names *Fatal*, *Flawed*, *Fair*, *Good* and *Strong*, which stand for the rubric's *unsound*, *weak*, *marginal*, *promising* and *strong*. They are scored 0 to 4 for the mean grade. The test of Section 5.1 was fixed before any of these 80 grades was read. It is a one-sided permutation test over 100,000 permutations, at the 0.05 level, of whether the retained ideas score higher on average than the eliminated ones.

F The Experiment Check on three runs

The three runs of Section 5.2 show how experimental claims and reviewer votes reach a verdict. Figure 8 records one round for each verdict under the rule of Section 3.5.

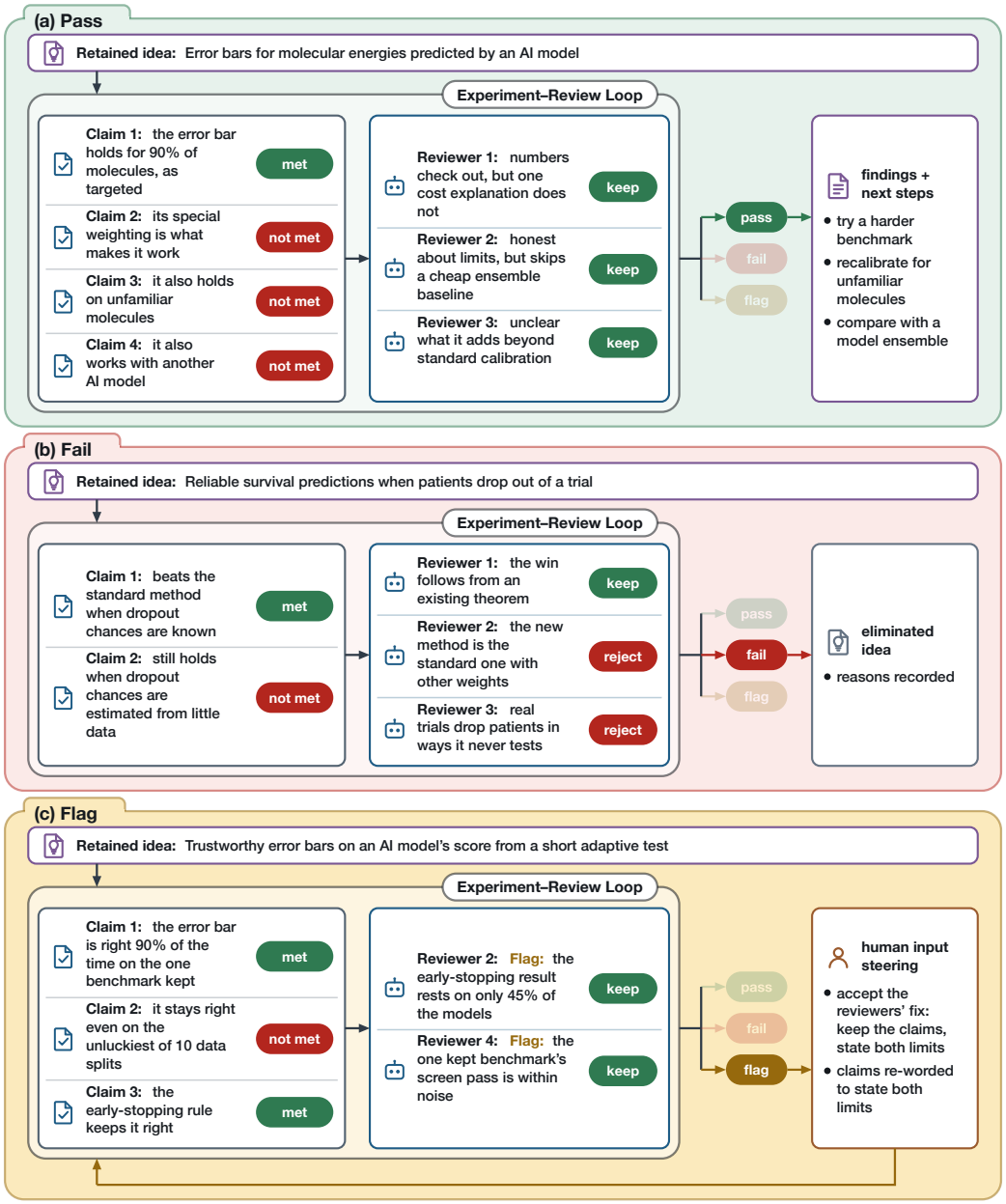


Figure 8: One review round from each of three Experiment Check runs, (a) pass, (b) fail, and (c) flag, each laid out like the Experiment Check band of Figure 6. Inside the loop, the left column lists the claims fixed before the experiments, each marked met or not met, and the right column the reviewers' reasons and votes. The highlighted verdict leads to the box on the right, and in (c) from there back into the loop. Panel (c) shows two of the four reviewers, one for each point raised to the researcher; all four vote *keep*.

G FIXED SETTINGS, MODELS, AND COST

The reported runs depend on fixed ranking, sampling, and test settings. Table 8 records these settings and seeds, Table 9 the model versions, and Table 10 the measured or estimated language-model use.

The largest recorded totals come from Ideation with the Desk Check and from the Experiment Check. The profile is built once offline, and ranking candidates calls no language model. A weekly

Table 8: Fixed settings and seeds, with the part of the paper where each enters.

Setting	Value	Where it enters
Reciprocal rank fusion constant k	60	Equation (1)
Reading list	first 100 papers of the fused ranking	Section 3.2
Reranker input	first 300 papers of the fused order	Section 4.3
Research threads per profile	2 to 10	Section 3.2
Candidate ideas per seed paper	up to 3, one round	Section 3.3
Experiment–review rounds	at most 3	Section 3.5
Profile cutoff	papers dated before 15 June 2026	Section 4.1
Embedding model	`allenai/specter2_base`	Appendix B.1
BM25 parameters	$k_1 = 1.5$, $b = 0.75$	Appendix B.1
PRF half-life	24 months	Table 1
Cross-encoder	`BAAI/bge-reranker-v2-m3`	Appendix C
Pool sample (2,574 of 30,695 papers)	seed 20260911	Section 4.1
Bootstrap over papers, retrieval	2,000 resamples, seed base 20260916	Section 4.2, Appendix B
Bootstrap over papers, reranking	2,000 resamples, seed 20260916	Section 4.3, Appendix C
Bootstrap over papers, full-read agent	10,000 Poisson-weighted resamples, seed 0	Appendix D
Permutation test, Desk Check	100,000 permutations, seed 20260925	Section 5.1

Table 9: Models recorded for agents in the reported runs. *Tier*: the setting in the system's configuration. *Version string*: the model identifier recorded in the run logs.

Stage	Agents	Tier	Version string
Recommendation	profile builder (the five profile builds of Section 4)	Opus	`claude-opus-5`
Ideation	stage orchestrator; the three seed-paper checks	Sonnet	`claude-sonnet-5`
	idea proposer	Opus	`claude-opus-4-8`
Desk Check	premise verifier, theory check, novelty and counterexample search, recheck, plan	Sonnet	`claude-sonnet-5`
Experiment Check	specification author, experiment worker, review synthesis	Opus	`claude-opus-4-8`
	auditors, reviewers, record keeping	Sonnet	`claude-sonnet-5`
Evaluation	rerankers (Section 4.3)	Sonnet	`claude-sonnet-5`
	full-read agent (Appendix D)		`claude-opus-5-5`
	Desk Check grader (Appendix E) and full-read judge (Appendix D)		`gpt-6-sol`

run of ideation with the Desk Check on every candidate consumed 163 to 255 million tokens, about nine tenths of them cache reads, in each of the four weeks of Section 5.1. The fail and flag runs of Appendix F consumed 203 and 748 million tokens, with cache reads at 95%. A LLM reranker reads an estimated 0.09 million input tokens in one call, and the full-read agent of Appendix D used 14.7 to 29.4 million tokens for one researcher on the week-sized sample.

Table 10: Language-model tokens in millions: input, cache writes, cache reads and output. Logged totals include every agent started and deduplicate responses by id; the profile has no recorded total, and the reranker input is estimated. Parentheses give the cache-read share of logged totals.

Stage	Unit	LM tokens (millions)
Profile	one researcher, built once	–
Ranking candidates	one ranking	0
Ideation and Desk Check	one weekly run; the four weeks of Section 5.1	190 (92%), 255 (94%), 163 (89%), 191 (93%)
Experiment Check	one run; the fail and flag runs of Appendix F	203 (95%), 748 (95%)
LLM reranker	one call	0.09 input, estimated
Full-read agent	one run on week-sized sample	14.7 to 29.4 (Table 5)