Paper-to-Forecast: An Automated Pipeline for Generating Calibrated Probabilistic Forecasts from Research Papers

Hauke Hillebrandt

April 14, 2026

Abstract

We present an automated pipeline that transforms research papers into calibrated binary forecasting questions with probabilistic estimates. The system implements a 13-stage process combining large language model (LLM) agents with structured quality gates: text extraction, chunked proto-question generation, quality and priority filtering (using an adapted Importance–Tractability–Neglectedness framework augmented with temporal urgency heuristics), cross-project deduplication, adversarial peer review, ensemble forecasting, question decomposition, probabilistic component analysis, and multi-method reconciliation. Built on the FutureSearch platform [Bosse et al., 2025], the pipeline processes multiple source papers in parallel and produces interactive HTML reports aggregating forecasts across projects. We describe the methodology in detail, report on initial deployments across six research domains (cybersecurity, AI R&D automation, biosecurity, governance), and discuss the design trade-offs between automation, cost, and forecast quality.

1 Introduction

Forecasting is a critical tool for evidence-based policy and decision-making [Tetlock and Gardner, 2015]. Yet the process of generating well-formed forecasting questions from domain research remains largely manual: analysts must read papers, identify falsifiable claims, formulate precise binary questions with unambiguous resolution criteria, verify feasibility, and finally produce probability estimates. This is slow, expensive, and difficult to scale across multiple research domains.

Recent advances in LLM-based agentic systems suggest that much of this process can be automated [Halawi et al., 2024, Schoenegger et al., 2024]. Bosse et al. [2025] demonstrated that LLM agents equipped with web search and structured prompting can generate forecasting questions and produce calibrated probability estimates that approach the accuracy of experienced human forecasters. Their FutureSearch system provides primitives—single-agent generation, parallel agent mapping, ranking, forecasting, and deduplication—that can be composed into complex workflows.

We build on this work to construct a fully automated *paper-to-forecast pipeline*: given a research paper (or any source text), the system extracts falsifiable claims, generates candidate forecasting questions, applies multi-stage quality filtering and adversarial review, produces probability estimates using ensemble methods, and decomposes high-priority questions into component sub-forecasts that are then reconciled into final calibrated estimates. The pipeline is designed to be:

1. **End-to-end**: from PDF input to interactive HTML report with no manual intervention.
2. **Quality-controlled**: multiple filtering stages with explicit gates eliminate poorly-formed, ambiguous, or trivial questions.
3. **Scalable**: adaptive effort allocation directs computational resources toward high-priority questions.

4. **Multi-project**: a global deduplication and aggregation layer prevents redundancy across independent research domains.

We deploy the pipeline across six source papers spanning cybersecurity, AI R&D automation, biosecurity, and AI governance, generating a total of 16 fully forecasted questions from an initial pool of over 300 proto-questions.

2 Related Work

Forecasting tournaments and platforms. Structured forecasting competitions [Tetlock and Gardner, 2015] and prediction markets [Arrow et al., 2008] have demonstrated that aggregated probabilistic judgments can outperform individual expert opinion. Platforms such as Metaculus, Polymarket, and Good Judgment Open provide infrastructure for question creation and crowd-sourced forecasting, but question generation remains a manual editorial process.

LLM-based forecasting. Halawi et al. [2024] showed that LLMs with retrieval-augmented generation can approach human forecaster accuracy on binary questions. Schoenegger et al. [2024] found that ensembles of LLM forecasters can match or exceed crowd forecasts. Bosse et al. [2025] introduced FutureSearch, a platform combining LLM agents with web search for automated question generation and forecasting, providing the SDK primitives on which our pipeline is built.

Automated question generation. Prior work on automated question generation has focused primarily on educational settings [Kurdi et al., 2020] or reading comprehension benchmarks. Our work differs in targeting *forecasting questions*—binary questions about future events with explicit resolution criteria, deadlines, and probability estimates.

Priority frameworks. The Importance–Tractability–Neglectedness (ITN) framework [Open Philanthropy, 2016] is widely used in effective altruism and policy analysis to prioritize research questions. We extend this framework with temporal urgency heuristics from Ord [2020]—Soon, Sudden, and Sharp—to produce a six-dimensional priority score that guides resource allocation within the pipeline.

3 System Overview

The pipeline consists of 13 stages organized into four phases: *generation* (Stages 01–02d), *refinement and verification* (Stages 03–05), *forecasting* (Stages 06–06d), and *reporting* (Stages 07–08). Figure 1 shows the complete data flow.

3.1 Design Principles

Progressive filtering. The pipeline generates many candidate questions cheaply, then applies increasingly expensive quality gates. This funnel architecture—inspired by information retrieval cascades [Wang et al., 2011]—ensures that expensive operations (adversarial review, ensemble forecasting, deep decomposition) are only applied to high-quality candidates.

Adaptive effort allocation. Computational effort scales with question priority. A composite ITNSSS score (defined in Section 4.4) determines whether each question receives HIGH, MEDIUM, or LOW effort at each stage, reducing cost without sacrificing quality on important questions.

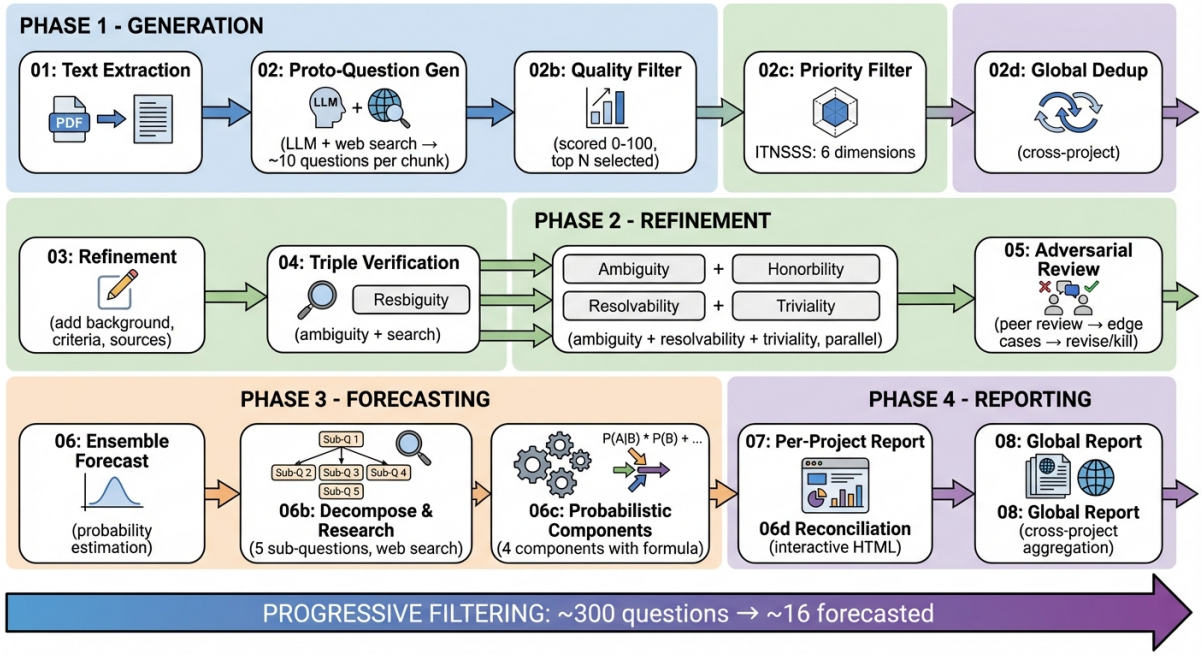

Figure 1: Pipeline architecture. The 13-stage system is organized into four phases: generation and filtering (blue), refinement and verification (green), forecasting and decomposition (orange), and reporting (purple). Questions are progressively filtered from ∼300 proto-questions down to ∼16 fully forecasted questions across six projects.

Multi-method forecasting. Rather than relying on a single probability estimate, the pipeline produces up to four independent estimates per question: (1) an ensemble forecast, (2) a decomposition-informed re-forecast, (3) a mechanically combined component forecast, and (4) a holistic reconciled estimate. Divergence between methods signals uncertainty.

4 Methodology

4.1 Stage 01: Text Extraction

Source documents (PDF or plain text) are processed using PyPDF2 with a sanitization pipeline that removes binary artifacts (embedded images, base64 data), drops lines with >20% non-printable characters, and preserves page boundaries. The output is a clean plain-text representation of the source paper.

4.2 Stage 02: Proto-Question Generation

Long papers are split into chunks of approximately 4,000 tokens (≈16,000 characters) using a section-aware chunking algorithm. The chunker identifies section boundaries via regex patterns matching markdown headers, numbered sections, and named sections (Abstract, Introduction, Methods, Results, Discussion, References). Large sections are recursively split at paragraph boundaries; small adjacent sections are merged. This ensures coverage of the full paper while respecting semantic boundaries.

For each chunk, a FutureSearch `single_agent` with web search access generates $k = 10$ candidate proto-questions (configurable via `QUESTIONS_PER_CHUNK`). The agent is instructed to extract falsifiable claims and formulate binary questions with explicit resolution deadlines. Each question includes a rationale grounding it in the source text and a paper reference to the specific section or claim.

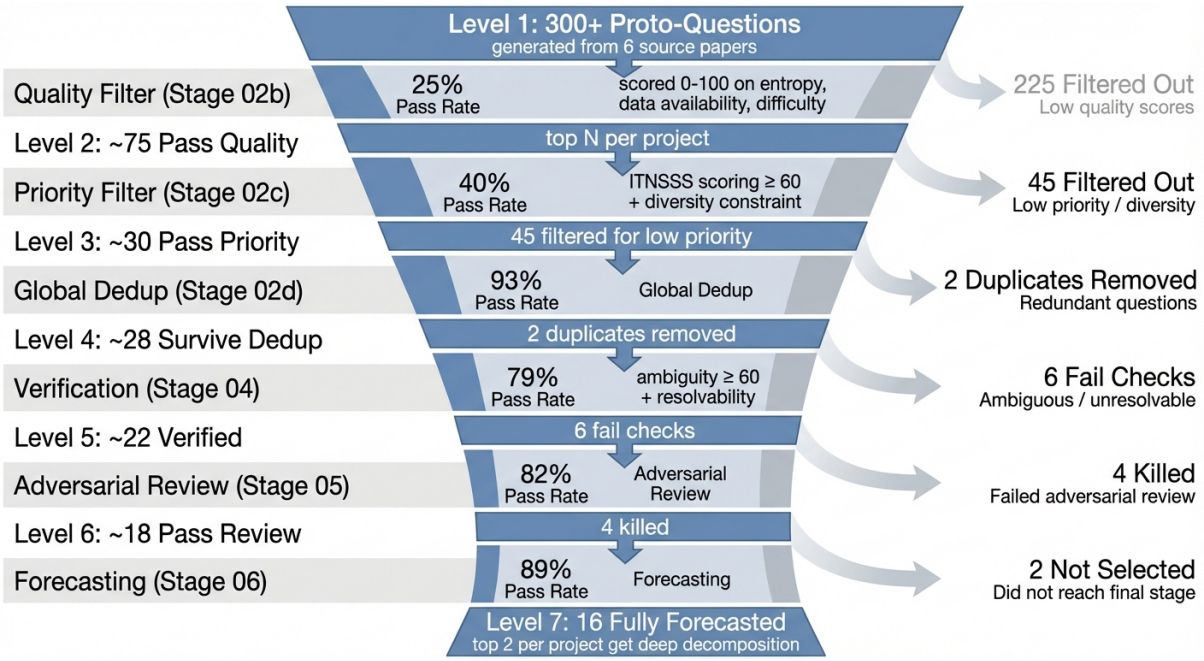

Figure 2: Filtering funnel. Progressive question selection from ∼300 proto-questions to 16 fully forecasted questions through successive quality gates: quality scoring, ITNSSS priority filtering, global deduplication, verification, and adversarial review.

An inline quality scoring step (using `rank()`) selects the top $m = 2$ questions per chunk (`TOP_PER_CHUNK`), yielding approximately $2 \times |\text{chunks}|$ proto-questions per paper.

4.3 Stage 02b: Quality Filtering

All proto-questions from Stage 02 are scored on a 0–100 scale using FutureSearch's `rank()` operation with three criteria:

1. **High entropy**: Is the question's outcome genuinely uncertain?
2. **Data availability**: Can resolution be determined from publicly observable sources?
3. **Difficulty**: Does the question require genuine forecasting effort (not trivially answerable)?

The top N questions (default $N = 5$, configurable via `QUALITY_FILTER_TOP_N`) advance. An interactive HTML review report is generated for human oversight.

4.4 Stage 02c: Priority Filtering (ITNSSS)

Surviving questions are scored on six dimensions using FutureSearch `agent_map()` with web search. Each dimension produces a 0–100 score:

- **Importance** ($w = 0.25$): Decision-relevance within the project scope.
- **Neglectedness** ($w = 0.25$): Marginal information value—how much existing research covers this question.
- **Tractability** ($w = 0.20$): Feasibility of producing a meaningful forecast given available data.
- **Soon** ($w = 0.10$): Temporal urgency—how quickly the question will resolve.
- **Sudden** ($w = 0.10$): Whether the outcome is a discrete event vs. gradual change.
- **Sharp** ($w = 0.10$): Absence of warning signals before resolution.

The composite priority score is:

$$\text{priority} = 0.25 \cdot I + 0.25 \cdot N + 0.20 \cdot T + 0.10 \cdot S_{\text{soon}} + 0.10 \cdot S_{\text{sudden}} + 0.10 \cdot S_{\text{sharp}} \quad (1)$$

The first three dimensions (Importance, Tractability, Neglectedness) follow the ITN framework commonly used in cause prioritization [Open Philanthropy, 2016]. The latter three operationalize the "Soon, Sudden, Sharp" heuristics from Ord [2020], which characterize risks that warrant urgent attention because they may materialize quickly, discretely, and without precursors.

A *topic diversity constraint* prevents cluster dominance: the agent assigns each question a topic cluster label, and at most $c = 3$ questions per cluster advance (configurable via `MAX_PER_CLUSTER`).

4.5 Stage 02d: Global Deduplication

Before investing in refinement, proto-questions are checked against a global repository of all questions from prior pipeline runs across all projects. Two modes are available:

- **Local** (default, free): Case-insensitive string similarity using Python's `difflib.SequenceMatcher` with threshold $\tau = 0.70$. The threshold is set higher than typical text similarity tasks because proto-questions share structural patterns ("Will X by December 31, 2027?") that inflate similarity scores.
- **Semantic** (optional, ∼$0.50): FutureSearch's `dedupe()` operation with a custom equivalence relation that identifies questions forecasting the same event even when differently worded.

4.6 Stage 03: Question Refinement

Each surviving proto-question is refined by a FutureSearch agent with web search into a full forecasting question comprising: (1) the question text, (2) contextual background, (3) explicit unambiguous resolution criteria, and (4) resolution sources (specific URLs or data sources where the outcome can be determined).

4.7 Stage 04: Triple Verification

Refined questions undergo three parallel verification checks:

Ambiguity scoring. A `rank()` operation scores each question on a 0–100 scale assessing whether 10 independent readers would agree on the outcome given the resolution criteria. Questions scoring below 60 are rejected.

Resolvability verification. An `agent_map()` operation checks a four-point checklist: (1) concrete observable event, (2) explicit resolution date, (3) publicly available evidence, (4) exhaustive YES/NO outcome space. Questions must receive "probably yes" or "very certainly yes" to advance.

Triviality screening. A separate agent produces a preliminary probability estimate. Questions with near-certain outcomes (<2% or >98%) are flagged to avoid wasting forecasting resources on trivial outcomes. This is a soft signal, not a hard gate.

4.8 Stage 05: Adversarial Peer Review

Verified questions undergo three-phase adversarial review, each using `agent_map()` with web search:

05a: Substantive review. An agent acts as an expert peer reviewer, identifying conceptual, methodological, or practical issues. Each question receives an assessment of PASS, NEEDS_REVISION, or FATAL_FLAW, along with detailed review notes and supporting evidence from web research.

05b: Edge case analysis. A separate agent identifies scenarios where the question breaks or becomes ambiguous—extreme but plausible outcomes, technical loopholes, resolution source failures—and rates overall risk as LOW, MEDIUM, or HIGH.

05c: Revision or kill. Questions flagged by either 05a (NEEDS_REVISION or FATAL_FLAW) or 05b (HIGH risk) are sent to a revision agent that either produces a corrected version addressing all concerns or recommends termination with a detailed rationale. Questions with verdict KILL are excluded from forecasting.

4.9 Stage 06: Ensemble Forecasting

Surviving questions (status PASS or REVISED) are forecasted using FutureSearch's `forecast()` operation, which implements an internally managed ensemble of multiple LLM forecasters. The operation returns a calibrated probability estimate (0–100) and a detailed rationale for each question. For REVISED questions, the revised text is used as the primary question.

4.10 Stage 06b: Decomposition and Re-Forecasting

The top $n = 2$ questions by priority score receive deeper analysis. For each question, the pipeline:

1. **Decomposes** the question into $k = 5$ independent research sub-questions using a `single_agent()` call.
2. **Researches** each sub-question in parallel via `agent_map()` with web search, producing evidence-based answers.
3. **Re-forecasts** by synthesizing all sub-question answers into an updated probability estimate, explaining how each piece of evidence affects the overall estimate.

This implements the "research and decompose" approach described in Bosse et al. [2025], where breaking a complex question into tractable sub-questions and researching each independently can improve calibration.

4.11 Stage 06c: Probabilistic Component Analysis

Building on the qualitative decomposition from Stage 06b, this stage identifies $c = 4$ probabilistic components—sub-events whose probabilities can be mechanically combined to estimate the parent question's probability. For each question, the system:

1. **Identifies components**: A `single_agent()` determines the key sub-events, their roles in the probability structure, expected dependency patterns, and an explicit combination formula (e.g., $P(\text{YES}) = P(C_1) \cdot P(C_2|C_1)$).
2. **Forecasts components**: Each component is independently forecasted via `agent_map()` with web search.

The combination type is classified as one of: *sequential chain* (events must happen in sequence), *disjunctive paths* (any path suffices), *weighted mixture*, or *hybrid*.

4.12 Stage 06d: Multi-Method Reconciliation

The final forecasting stage reconciles the multiple estimates produced in Stages 06–06c. An `agent_map()` operation processes each question through a five-step reconciliation:

1. **Bottom-up estimate**: Mechanical combination of component probabilities using the formula from Stage 06c.
2. **Dependency adjustment**: Correction for identified conditional dependencies between components.
3. **Structure adjustment**: Check for conceptual issues in the decomposition that might bias the estimate.

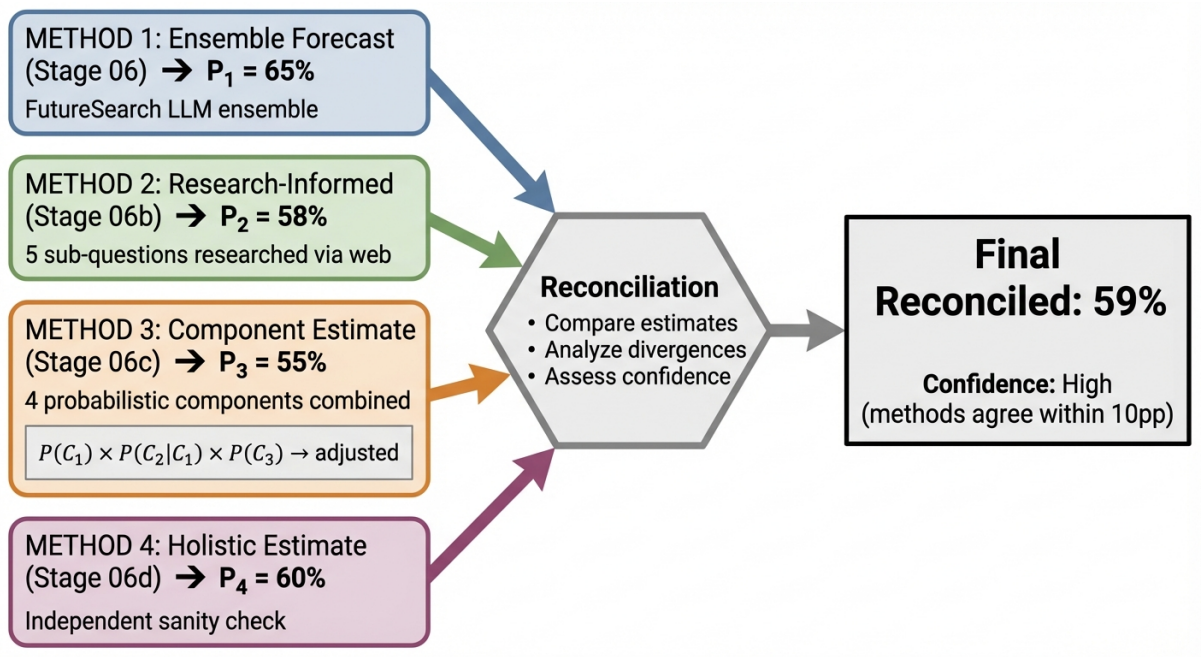

Figure 3: Multi-method forecast reconciliation. Four independent estimation methods—ensemble forecast, research-informed re-forecast, bottom-up component combination, and holistic estimate—converge through a structured reconciliation process into a final calibrated probability.

4. **Holistic estimate**: An independent probability estimate produced *without* reference to the decomposition.
5. **Reconciliation**: Divergence between bottom-up and holistic estimates is analyzed, and a final probability is produced with explicit rationale.

This multi-method approach—combining ensemble forecasting, research-informed re-forecasting, structured decomposition, and holistic estimation—provides robustness against any single method's systematic biases.

4.13 Stages 07–08: Report Generation

Per-project reports (Stage 07) and a global cross-project report (Stage 08) are generated as self-contained interactive HTML files. Reports include:

- Sortable/filterable question cards with probability estimates, priority breakdowns, and full provenance.
- Expandable sections showing decomposition trees, component forecasts, and reconciliation analysis.
- Cross-project duplicate detection (string similarity at $\tau = 0.78$ in the global report).
- A methodology summary with links to source papers.
- An "Explored Proto-Questions" section showing all candidate questions that were filtered at early stages, with their quality scores, enabling transparency about what the pipeline considered and rejected.

Reports are published to GitHub Pages for sharing and review. Each question card includes a stable anchor link (derived from the question text) that enables direct deep-linking to specific questions from external platforms.

4.14 Prediction Market Integration

The pipeline includes an optional final stage that publishes forecasted questions to *Manifold Markets*[1], a prediction market platform. An automated upload script:

1. Reads the top-ranked questions from the global report (sorted by ITNSSS priority score).
2. Uses an LLM to shorten the full question text into a concise market title (≤ 120 characters) suitable for prediction market listings, following the informal style conventions of the platform.
3. Checks for existing duplicate markets via the Manifold search API to avoid redundant listings.
4. Creates binary markets via `POST /v0/market` with the initial probability set to the pipeline's reconciled forecast, the close date extracted from the question's resolution timeline, and a rich description containing the full question text, background context, resolution criteria, decomposition sub-questions, probabilistic component breakdowns, and multi-method reconciliation estimates.
5. Saves the Manifold market URLs back to the question data, which are then displayed as "Bet" badges on the interactive HTML report—closing the loop between the forecast repository and the prediction market.

This integration enables crowd-sourced calibration: the prediction market provides a complementary signal from human forecasters that can be compared against the pipeline's automated probability estimates. As of publication, six markets have been created from the pipeline's top-priority questions across biosecurity, AI governance, and AI automation domains.

5 Adaptive Effort Allocation

A key design choice is *adaptive effort allocation*, which directs more computational resources (and cost) toward higher-priority questions. The system supports four effort modes:

- `max`: Highest-capability model (Claude Opus) at every stage. Most expensive (∼$20/question).
- `auto`: High effort with model selection delegated to FutureSearch (∼$15/question).
- `adaptive`: Two-dimensional scaling—per-stage baselines combined with per-question scaling by priority score.
- `low`: Minimal-cost model (Gemini Flash) everywhere (∼$1–2/question).

In adaptive mode, the effort function is:

$$\text{effort}(s, q) = \begin{cases} \text{baseline}(s) & \text{if priority}(q) \geq 75 \\ \min(\text{baseline}(s), \text{MEDIUM}) & \text{if } 55 \leq \text{priority}(q) < 75 \\ \text{LOW} & \text{if priority}(q) < 55 \end{cases} \tag{2}$$

where s is the pipeline stage and q is the question.

6 Implementation

The pipeline is implemented in Python and orchestrated by a CLI tool (`run_pipeline.py`) that accepts a project name, source document, and configuration parameters. Each stage is an independent script that reads its predecessor's output and writes structured JSON/CSV files, enabling selective re-execution of individual stages.

FutureSearch SDK. We use five core operations from the FutureSearch Python SDK [Bosse et al., 2025]: `single_agent()` for open-ended generation, `agent_map()` for parallel per-row

[1] `https://manifold.markets`

Table 1: Pipeline deployment across six projects. Proto-Qs: initial proto-questions generated. Final: questions surviving all filters and receiving full forecasts.

Project	Domain	Source	Proto-Qs	Final
cyber	Cybersecurity	Institute for AI Policy and Strategy [2025]	75	3
ai_rd_automation	AI R&D	Kwa et al. [2025]	7	2
biosecurity	Biosecurity	Moulange [2025]	8	2
80k_biosec_pod	Biosecurity	80,000 Hours Podcast [2025]	19	2
govai_fellowship	AI Governance	Centre for the Governance of AI [2026]	13	7
lab_leaks	Biosecurity	(unpublished)	–	–
Total			**>300**	**16**

agent execution with web search, `rank()` for scoring/ranking, `forecast()` for calibrated ensemble probability estimation, and `dedupe()` for semantic deduplication. All operations support Pydantic schema enforcement for structured output.

Prompt management. LLM prompts are maintained in Google Docs with tab-based versioning, loaded at runtime via the Google Docs API, and cached to avoid redundant fetches. Template variables (e.g., `{{resolution_deadline}}`) are rendered at invocation time.

Cost tracking. The pipeline queries the FutureSearch billing API before and after each stage, printing per-stage costs and running balance. A typical full pipeline run across six projects costs $15–30 depending on effort mode.

Project structure. Each project is defined by a `project.json` metadata file containing the project name, paper title, description, resolution deadline, source URL, and processing flags. Data flows through a standard directory structure (`projects/<name>/data/`) with numbered output files corresponding to pipeline stages.

7 Results

We deployed the pipeline across six source papers spanning four research domains. Table 1 summarizes the results.

The pipeline's filtering funnel reduces the initial pool by approximately 95%, with the largest reductions occurring at quality filtering (Stage 02b) and priority filtering (Stage 02c). The adversarial review (Stage 05) typically passes 60–80% of verified questions, with the remainder revised or killed.

7.1 Filtering Funnel

For the cyber project (the largest), the funnel proceeds as follows: 75 proto-questions $\rightarrow$ 15 after quality filter $\rightarrow$ 5 after priority filter $\rightarrow$ 3 after deduplication, verification, and adversarial review $\rightarrow$ 3 fully forecasted questions, of which 2 receive deep decomposition and probabilistic analysis.

7.2 Cross-Project Deduplication

The global report's cross-project deduplication (at threshold $\tau = 0.78$) identified 2 genuine duplicate pairs across the 16 forecasted questions. These were flagged in the report rather than removed, since questions from different source papers may have subtly different framings

that are independently valuable. An earlier threshold of $\tau = 0.50$ produced 82 pairs, the vast majority false positives caused by structural similarity in question formatting.

7.3 Multi-Method Forecast Comparison

For questions receiving all four estimation methods, we observe that:

- The ensemble forecast (Stage 06) and research-informed re-forecast (Stage 06b) typically agree within 10 percentage points.
- The bottom-up component estimate (Stage 06c) sometimes diverges more substantially, often due to conditional dependencies not fully captured by the component structure.
- The reconciliation stage (Stage 06d) identifies and corrects for these divergences, producing final estimates that are typically between the ensemble and component estimates.

8 Discussion

Automation vs. oversight. The pipeline is designed to run end-to-end without manual intervention, but produces HTML reports with full provenance at each stage to enable human review. The "Explored Proto-Questions" section—showing all candidates that were generated but filtered—provides transparency about the pipeline's coverage and decisions. We envision a workflow where the pipeline runs automatically but a human analyst reviews the final report before publication.

Calibration. We do not yet have resolution data to evaluate calibration empirically, as the questions have resolution deadlines in late 2027. Future work will compare pipeline forecasts against actual outcomes and against human forecaster baselines.

Cost–quality trade-offs. The adaptive effort mode provides a reasonable middle ground: high-priority questions receive full computational resources while lower-priority questions are processed more cheaply. A full pipeline run across all six projects costs approximately $15–30, making it feasible to process dozens of papers per month.

Limitations. The pipeline inherits the limitations of its underlying LLM agents: potential for hallucination in research summaries, sensitivity to prompt wording, and possible systematic biases in probability estimation. The adversarial review stage mitigates some of these issues but cannot eliminate them entirely. Additionally, the quality of proto-questions depends heavily on the richness of falsifiable claims in the source paper; papers with few concrete predictions yield fewer viable forecasting questions.

9 Conclusion

We have presented a 13-stage automated pipeline that transforms research papers into calibrated binary forecasting questions. The system combines LLM-based generation with structured quality gates, multi-dimensional priority scoring, adversarial review, and multi-method probabilistic forecasting. Initial deployments across six research domains demonstrate that the pipeline can generate meaningful forecasting questions at scale while maintaining quality through progressive filtering. The interactive reports and full provenance tracking enable human oversight without requiring manual question generation. Future work will focus on empirical calibration evaluation once questions resolve, expanding the pipeline to additional source types (podcasts, policy documents, technical reports), and investigating whether the multi-method reconciliation approach improves calibration relative to single-method estimation.

References

80,000 Hours Podcast. Richard moulange on ai, bioweapons, and biorisk, 2025. URL https://80000hours.org/podcast/episodes/richard-moulange-ai-bioweapons-biorisk/.

Kenneth J Arrow, Robert Forsythe, Michael Gorham, et al. The promise of prediction markets. *Science*, 320(5878):877–878, 2008.

Philipp Bosse et al. Futuresearch: Leveraging ai agents for automated forecasting. *arXiv preprint arXiv:2601.22444*, 2025. URL https://arxiv.org/abs/2601.22444.

Centre for the Governance of AI. Winter fellowship 2026, 2026. URL https://www.governance.ai/post/winter-fellowship-2026.

Danny Halawi, Fred Schneider, Tim Ye, et al. Approaching human-level forecasting with language models. *arXiv preprint arXiv:2402.18563*, 2024.

Institute for AI Policy and Strategy. Highly autonomous cyber-capable agents. Technical report, 2025. URL https://www.iaps.ai/research/highly-autonomous-cyber-capable-agents.

Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30:121–204, 2020.

Thomas Kwa et al. Measuring ai's ability to complete long tasks. *arXiv preprint arXiv:2603.03992*, 2025. URL https://arxiv.org/abs/2603.03992.

Richard Moulange. Reasons to be pessimistic and optimistic about ai bioweapons risk, 2025. URL https://www.owlposting.com/p/reasons-to-be-pessimistic-and-optimistic.

Open Philanthropy. Importance, tractability, and neglectedness, 2016. URL https://www.openphilanthropy.org/research/some-useful-heuristics/. Accessed 2026-04-14.

Toby Ord. *The Precipice: Existential Risk and the Future of Humanity*. Hachette Books, 2020.

Philipp Schoenegger, Peter S. Park, Ezra Karger, and Philip E. Tetlock. Wisdom of the silicon crowd: LLM ensemble prediction capabilities match human crowd accuracy. *Science Advances*, 2024.

Philip E Tetlock and Dan Gardner. *Superforecasting: The Art and Science of Prediction*. Crown, 2015.

Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of SIGIR*, pages 105–114, 2011.

A ITNSSS Priority Score Components

Table 2 details the six components of the priority scoring framework used in Stage 02c.

B Pipeline Stage Summary

Table 3 provides a concise summary of all pipeline stages with their FutureSearch operations and quality gates.

Table 2: ITNSSS priority score components. Each is scored 0–100 by an LLM agent with web search access.

Component	Source	Weight	Description
Importance	ITN	0.25	Decision-relevance: would knowing the answer change resource allocation or policy?
Neglectedness	ITN	0.25	Marginal information value: how much existing research, data, or forecasting covers this question?
Tractability	ITN	0.20	Feasibility: are there base rates, resolution sources, and analogues that make meaningful forecasting possible?
Soon	Ord	0.10	Temporal urgency: will this question resolve in the near term?
Sudden	Ord	0.10	Discreteness: is the outcome a sudden event rather than a gradual trend?
Sharp	Ord	0.10	Surprise potential: might the outcome occur without clear warning signals?

Table 3: Pipeline stage summary. Operations refer to FutureSearch SDK functions.

Stage	Name	Operation	Gate	Output
01	Text extraction	Local	–	Plain text
02	Proto-generation	single_agent	Top m/chunk	Proto-questions
02b	Quality filter	rank	Top N by score	Quality-scored Qs
02c	Priority filter	agent_map	Score ≥ 60, diversity	Priority-scored Qs
02d	Global dedup	difflib/dedupe	Similarity $< \tau$	Deduplicated Qs
03	Refinement	agent_map	–	Refined questions
04	Verification	rank + agent_map	Ambiguity ≥ 60, resolvable	Verified questions
05	Adversarial review	agent_map $\times 3$	Pass/revise/kill	Reviewed questions
06	Forecasting	forecast	–	Probability estimates
06b	Decomposition	single_agent + agent_map	–	Sub-question research
06c	Component analysis	single_agent + agent_map	–	Component forecasts
06d	Reconciliation	agent_map	–	Final estimates
07–08	Reporting	Local	–	Interactive HTML