

Desearch 2.0: A Maintained Web Index for AI

An open contribution network on Bittensor

Version 0.1 | 21 September 2026 | Technical design proposal

Prepared for technical discussion. This draft develops the Desearch 2.0 proposal at commit `e5eda3282de813a32a3b34204f2a82911e3e606d`. It describes a proposed system and an evaluation program; it does not report a deployed 2.0 network or measured performance gains. Record structures and lifecycle requirements below are proposed elaborations for review, not released APIs. Operational parameters and reward rules remain to be specified before participation opens. [1]

Contents

Desearch 2.0 — a proposal for building and serving reusable search infrastructure.

Abstract	3
1. From search responses to maintained capability	3
2. Objectives and first-phase scope	4
3. Participants and trust boundaries	4
4. Architecture: building, assessing and serving	5
5. Source records and search versions	7
6. Task execution and contribution validation	8
7. Contribution accounting and Bittensor incentives	10
8. Retrieval and controlled product releases	12
9. Threat model and failure analysis	12
10. Evaluation and full-system economics	14
11. Related work and design alternatives	15
12. Openness, rights and implementation milestones	16
13. Limitations and future work	17
14. Conclusion	17
Appendix A. Worked contribution example	18
Appendix B. What must be fixed before participation opens	19
References	20

Abstract

AI applications need current sources and usable evidence within the time available for a task. A network that rewards individual search responses can supply useful answers while leaving the underlying collection and retrieval capabilities dependent on particular operators or external services. Desearch 2.0 proposes organizing network work around a maintained web collection and a versioned search index. Miners would fetch assigned sources and compute embeddings for approved text using a selected model. Validators would assess those outputs under published task rules, with assessed contributions informing miner scores and Bittensor reward allocation. Desearch would initially operate the task queue, stored collection, index and customer API. Customer queries would use an approved search version while collection and preparation continue in the background. The design separates task validity, contribution rewards and product deployment, allowing completed work to be assessed without making every customer request depend on its contributor. This paper describes the architecture, task contracts, validation boundaries, incentive requirements and trust assumptions. It proposes evaluation of relevance, freshness, latency and full production costs, including verification. The central hypothesis is that open contributions can build reusable search capability at a useful quality and cost; this draft does not claim that hypothesis has been demonstrated.

1. From search responses to maintained capability

The value of search for an AI application depends on what the application can do with the result. An agent may need to identify the current specifications of a product, compare a set of companies, or retrieve the passage supporting a factual claim. A relevant link is useful, but the application also needs accessible text, sufficient context and a way to judge whether the source is appropriate and current. These needs connect retrieval to the continuing work of collecting, preparing and maintaining information.

Desearch 1.0 organized outside contributions around search responses. Miners supplied results through their retrieval approaches, validators evaluated answers, and the Desearch team operated the customer API. The team's account identifies three recurring limitations: response time, dependence on capabilities outside its control, and insufficiently retained source material and retrieval components. It also acknowledges that 1.0 delivered useful results. These observations motivate the proposed change; the historical account does not provide a controlled benchmark establishing the size of each limitation. [2]

A high-quality response does not by itself establish what reusable capability produced it. An answer may depend on a miner's private index, a licensed provider, or a combination of services. Evaluating the answer can tell us whether that request was served well, while revealing little about whether the network can preserve the capability, inspect its coverage, refresh its contents or reproduce the result later. Desearch 2.0 addresses this gap by making retained information and compatible retrieval components explicit outputs of contribution work.

The proposed system prepares information before an ordinary query arrives. Contributors collect assigned pages; accepted text is cleaned and organized; selected models generate representations; and compatible components are assembled into a search version. A contribution can therefore support many subsequent queries. Continuing maintenance remains necessary: pages change, sources disappear, and a growing collection can accumulate stale or low-value material. Retention creates an opportunity for reuse, with additional obligations for refresh, deletion and quality control. [3]

The paper's central question is whether an open contribution network can build that maintained capability effectively. Crawling, indexing and hybrid retrieval are established techniques. The design question here is how to connect them to independent execution, credible assessment, contributor incentives and accountable product releases. The answer requires both a clear mechanism and evidence that useful output justifies the total work of producing, checking and integrating it.

2. Objectives and first-phase scope

Desearch is intended for developers building AI agents and research applications. The initial product path is search that returns ranked sources and useful supporting text. Deeper research can use a wider set of sources and a larger retrieval budget, but its requirements should be evaluated separately from a fast agent lookup. The initial collection would cover selected public-web sources such as company sites, documentation, news, articles and blogs. The first phase is bounded by chosen workloads and domains. [3], [4]

Four objectives guide the design. First, contribution work should leave inspectable, reusable information or retrieval components. Second, a contributor should understand the task and assessment rules before investing resources. Third, customer search should use a coherent, tested version while new work is evaluated. Fourth, improvements should be assessed through relevance, freshness, response time and full operating cost, with the scope of each measurement made explicit.

The first two miner task families are **CRAWL** and **EMBED**. CRAWL retrieves selected sources under a defined extraction and permitted-use contract. EMBED applies a selected model and configuration to approved text chunks. Running that model is execution work. Training a new model, changing the representation or entering a reranking competition is a separate contribution program with different evaluation needs. The first-phase scope does not grant miners the freedom to substitute arbitrary models for an assigned embedding task. [5]

The initial architecture places the task queue, retained collection, index and customer-serving system under Desearch's operation. Independent contributors build and maintain inputs to that system. This control boundary is part of the proposal: access to an open contribution network does not imply distributed ownership of every component or decentralized query serving. The paper evaluates openness through participation, assessment, attribution and opportunities to improve the system, while making operational control visible. [3], [6]

Later programs may include extraction improvements, better refresh methods, model training and reranking. A bounded reranking experiment is identified in the proposal as an early candidate for testing method improvement, subject to suitable data and evaluation. Additional data products would require their own customer need, preparation and usage rights. These opportunities do not define a fixed release sequence. [3]

3. Participants and trust boundaries

The system involves five operational roles and the Bittensor protocol. Some organizations may perform more than one role, so independence should be evaluated through actual control and disclosed conflicts rather than role names alone.

Role	Responsibility in the proposed first phase	Boundary of that responsibility
Desearch team	Select workloads and sources; publish task terms; operate queue, collection, index and API; approve releases	Accountable for allocation policy, integration, rights, security and product behavior
Miners	Execute assigned CRAWL or EMBED work and submit attributable outputs	A valid submission establishes compliance with its task contract, not universal source truth or search quality
Validators	Assess outputs, record findings and produce miner evaluations under published rules	Checks must disclose their scope; error rates require measurement. Agreement does not establish that every output was checked
Source operators	Publish or control the information accessed by crawling	Sources may change, be unavailable, or contain incorrect and adversarial material
Customers	Submit queries and use returned sources and text	Retrieved content remains external input to the customer's application
Bittensor protocol	Aggregate eligible validator weights and allocate protocol rewards	The protocol does not select useful pages or establish retrieval quality on Desearch's behalf

Desearch's authority over source selection and releases creates both accountability and a dependency. The team can reject a candidate index that fails product criteria, but it can also influence which work is available and which improvements reach customers. Published criteria, attributable decisions and a challenge process are therefore part of a credible participation model. Team participation in contribution programs should be disclosed, including any different eligibility or reward treatment. [6]

The design also depends on access to external sources and on correct identity binding between tasks, miners and submissions. A validator needs enough evidence to apply a task's rule; a consumer needs a result tied to the source and search version that produced it. Neither requirement can be satisfied by a participant's assertion alone. The following sections describe the records and checks needed to make those relationships inspectable.

4. Architecture: building, assessing and serving

Desearch 2.0 separates the path that builds search assets from the path that answers customer queries. It also separates contribution accounting from product adoption. These paths share records, but they have different decision criteria. [5], [7]

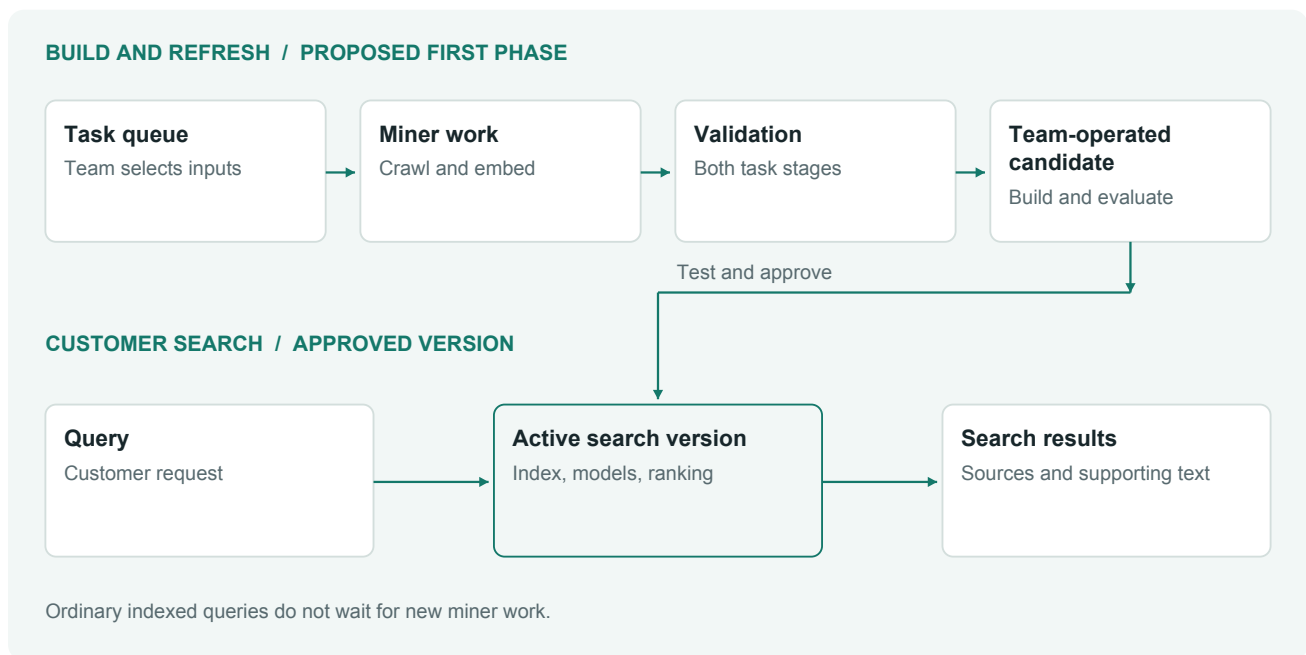


Figure 1. Proposed first-phase architecture. Miners execute CRAWL and EMBED tasks; validators assess outputs. Desearch initially operates the queue, collection, candidate index and search API. Assessed work can enter contribution accounting independently of a candidate's approval for customer use. Content preparation is under team oversight; the assignment of each preparation step to an operator remains to be specified.

The build path starts with source selection and task issuance. A crawl submission is assessed against its requested source and extraction requirements. Accepted content is prepared into usable text chunks, and embedding tasks produce representations under a specified model configuration. The team assembles compatible content, representations and retrieval settings into a candidate version. Evaluation then determines whether that version is suitable for customer traffic.

The query path begins with an already approved version. It retrieves candidate passages, ranks them and returns sources with supporting text. Ordinary indexed queries would not wait for a new crawl submission or a validator decision. Collection and model preparation can proceed independently of a specific query, although serving still depends on the availability and capacity of the active index, query model and API. Removing one dependency from a request does not establish an end-to-end latency guarantee.

The accounting path begins with attributable submissions and recorded assessments. Published rules convert eligible work into miner scores, which inform validator weights and protocol allocation. A task can be correctly completed without immediately improving the active search product. Keeping these paths distinct allows product releases to be selective without retroactively changing the terms offered for completed work.

4.1 A document's path through the system

Consider an assigned documentation page. The miner returns the observed page outcome, permitted content, extracted text and the identifiers needed to relate them to the task. After assessment, preparation removes unsuitable material, applies the chosen duplicate policy and splits accepted text into chunks. EMBED tasks then produce vectors for those exact chunks. The candidate index records which observation, preparation procedure and model generated the searchable entries.

When a customer later asks about the documented feature, retrieval can return the relevant passage and its source. If the page changes, a new observation follows the same assessment and release process. The old record must not silently acquire the new text's identity, and the new result must not inherit an earlier

observation timestamp. A deletion instruction must reach every derived representation that could continue returning the affected content.

This walkthrough specifies relationships the implementation should preserve. It does not select an object store, queue technology, vector database or index engine. Those choices should be made against the first workload and measured operational requirements.

5. Source records and search versions

The retained collection should preserve source information independently of the model used to retrieve it. A vector represents text for a particular retrieval procedure; it cannot replace the source text, its observation context or the permissions governing its use. Keeping these records separate also makes model changes possible without discarding the information from which representations were produced. [5]

The following record types are proposed logical abstractions for implementation review. They identify required relationships rather than prescribe an API schema.

Record	Information it should bind	Purpose
Source observation	Requested and resolved source, observation time, fetch outcome, permitted retained material and extraction version	Distinguish what was observed from what was requested
Prepared document or chunk	Parent observation, preparation version, stable identifier and text identity	Trace filtering, deduplication and chunking decisions
Embedding artifact	Ordered chunk identities, document model/version, configuration and vector metadata	Prevent representations being attached to different inputs or incompatible models
Task and submission	Task identity, contributor, required inputs/configuration, output references and timing	Support assessment, attribution and duplicate handling
Assessment	Assessor, rule version, evidence scope, outcome and reason	Make the basis and limits of a decision reviewable
Search version	Eligible collection snapshot, preparation, document/query models, indexes and ranking settings	Define a compatible unit of evaluation and release

A content digest can help detect accidental substitution and distinguish byte sequences. It does not establish that a page is factually correct, that a claimed source actually served it, or that its use is permitted. Identity and provenance therefore require more than a hash: the record needs an observation context and an evidence policy appropriate to the task.

A search version can be written descriptively as $V = (C, P, Md, Mq, I, R)$, where C identifies the eligible collection, P the preparation procedure, Md and Mq the document and query models, I the relevant indexes, and R the ranking configuration. This notation names the compatibility boundary; it is not a selected storage format. Equal vector dimensions are insufficient to make representations from unrelated models interchangeable.

Freshness requires more than a recent crawl timestamp. Observation time, source publication time, source modification time and the time an update becomes searchable describe different events. An unchanged page may have been checked recently; a newly indexed page may already contain old information. The implementation should preserve these distinctions and report freshness against a defined workload.

Deletion must constrain version selection. A rollback should restore a compatible earlier serving configuration while continuing to enforce current deletion restrictions. Otherwise, recovery from a bad release could reintroduce material removed for quality, rights or other valid reasons. The proposal requires rollback without restoration of deleted material; the implementation must define how this invariant applies to text, chunks, vectors and retained snapshots. [5]

6. Task execution and contribution validation

Task rules form the agreement between the party requesting work, the contributor executing it and the validator assessing it. Rules should identify the inputs, required configuration, expected output, assessment procedure and consequences of incomplete or unavailable evidence. Contributors should be able to determine these conditions before working. Exact task and submission formats remain a release requirement. [5], [7]

6.1 CRAWL tasks

A CRAWL task would specify selected URLs, relevant prior source information, extraction settings and permitted-use instructions. The expected result includes the fetch outcome, permitted page content and extracted text, the source URL, observation time and extraction version. A valid unsuccessful fetch can be meaningful output when the task allows it. Treating every inaccessible source as miner misconduct would confuse external conditions with execution failure.

Assessment should establish whether the submission corresponds to the requested source and whether the required procedure and fields were supplied. Dynamic content complicates reproduction: the page visible to a validator may differ from the miner's observation. The checking policy must therefore define acceptable evidence, timing and treatment of changed or inaccessible sources. Re-fetching can provide corroborating evidence, but an unexplained mismatch is not automatically proof of dishonest work.

Preparation follows crawl assessment under team oversight. It can remove low-value material, resolve duplicates and divide text into retrieval chunks. These operations affect downstream usefulness, and their versions should be recorded. The initial proposal does not assign every preparation operation to a miner or validator; the operator guide must make those assignments explicit.

6.2 EMBED tasks

An EMBED task would bind approved chunks to a selected model version and representation configuration. A submission must return the required number of vectors in the required order, with the identifiers needed to relate each output to its input. Structural checks can reject missing outputs, inconsistent dimensions and invalid numerical values. These checks are necessary but cannot alone establish that the selected model was actually applied to the assigned text.

Validators would use GPU compute to reproduce outputs under an announced procedure. That procedure must account for permitted variation across supported hardware and numerical configurations. Demanding byte-identical vectors without testing reproducibility could reject correct work. Accepting a

broad similarity tolerance without adversarial testing could admit incorrect work. Tolerances must be calibrated using both compliant outputs and deliberately corrupted submissions.

Embedding miners and validators checking embeddings would need appropriate GPU access. The proposal has not selected models, memory sizes, batch limits or supported hardware. Tested requirements should be published with the task configuration so operators can estimate feasible throughput and cost.

[8]

6.3 Assessment outcomes and evidence scope

The protocol should distinguish three substantive outcomes. **Valid** means the evidence satisfies the task's stated conditions. **Invalid** means it establishes a failure to satisfy those conditions. **Unverifiable** means the available evidence is insufficient for a reliable decision under the rules. These outcomes may require different retry, credit and dispute treatment. A record should preserve the reason and the applicable rule version.

Validation scope must be visible. A check of every item and a check of a sample support different conclusions. If sampling is used, the program must define how samples are selected, what claims can be made about the remainder, how adaptive attackers are considered and when additional checking is triggered. A passed sample should never be presented as an independent examination of every item. This draft selects neither a sampling fraction nor a detection guarantee.

The cost of checking belongs to the mechanism. Full reproduction of embedding work can duplicate substantial computation; sampling trades reduced checking effort against uncertainty and attack exposure. Credible validation requires measuring both incorrect acceptance and incorrect rejection, together with the resources needed to obtain those error rates. A method that cheaply approves more submissions is not useful if it cannot distinguish correct work from a profitable shortcut.

6.4 An illustrative lifecycle

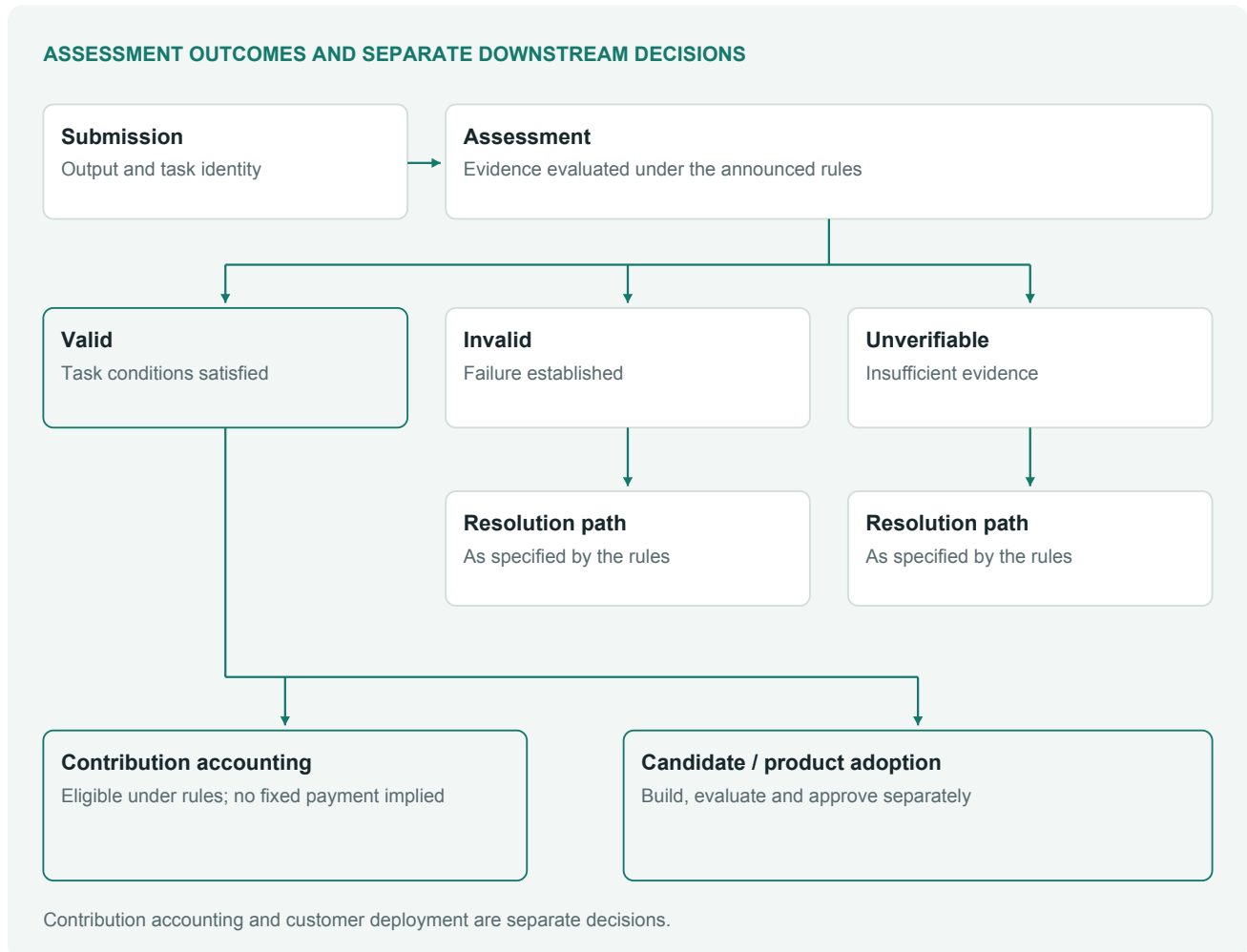


Figure 2. Proposed decision lifecycle. A submission leads to an evidence-based assessment. Valid work can qualify for contribution accounting under the published rules; invalid and unverifiable outcomes follow their specified resolution paths. Product adoption is a separate decision. This diagram defines distinctions for review, not a released state machine or payment schedule.

Suppose a miner receives a batch of approved chunks and the configured document model. If all required outputs meet the announced checks, the work can qualify under the program's scoring terms. If vector order is incorrect, the submission fails the corresponding contract. If a checking environment cannot reproduce a result reliably, the evidence may be insufficient until the stated resolution procedure runs. None of these cases determines a fixed token payment from the task count alone.

7. Contribution accounting and Bittensor incentives

Desearch's incentive design connects the work of maintaining a search collection to the participants who perform and assess that work. The first programs would recognize execution of assigned CRAWL and EMBED tasks. Their immediate question is whether a contribution satisfies an announced contract. Its effect on the deployed search product is assessed through a separate release process. This division lets the network account for useful preparation and maintenance before a particular contribution appears in a customer response. [3], [6]

7.1 Attribution and access to work

We recommend a versioned contribution record as the accounting unit between assessment and scoring. It should identify the governing contract, submission, assessment evidence and any later decision that changes the assessment. This record would let an operator trace a score back to particular work and challenge an error without reconstructing an entire scoring period. Access controls should protect restricted source content and evaluation material while preserving sufficient evidence for review. The record format and access policy are design recommendations for the implementation specification.

Allocation rules determine which participants have an opportunity to earn credit. We recommend documenting assignment eligibility, expiration, retries and access for new miners alongside the scoring contract. The same specification should distinguish repeated submission of one result from a separately authorized refresh or independent check. This distinction matters because repeated work can have maintenance or verification value, whereas repeated accounting for the same obligation can distort reward allocation. Contract versions should make these cases reviewable before work begins.

7.2 Scores, weights and protocol allocation

The proposal leaves the numerical score-to-weight mechanism open. [7] We recommend that its specification include an accounting window, comparable units or an explicit normalization rule, treatment of incomplete evidence, and worked examples covering mixed workloads and zero accepted work. These requirements make the combination of assessed work across miners and task families reproducible. An implementation should also state which properties it intends to encourage and test whether participants can profit by exploiting its aggregation rules.

Validator weights connect subnet assessment to Bittensor's reward allocation. Eligible validators express their relative assessments of miners through weights, which the protocol uses to determine miner reward shares. Validator dividends follow the protocol's own rules; they are not a Desearch payment for each check. A contribution score consequently cannot be read as a fixed token amount, and a larger task count does not establish a proportional reward increase. The final integration specification must identify the applicable protocol version and distinguish subnet scoring decisions from protocol allocation. Current documentation describes the relationship between validator assessments, Yuma consensus and emissions; it is a separate reference from the foundational whitepaper. [7], [9], [10]

Completion terms must remain stable when product decisions change. Correctly executed work can qualify under its announced terms even if a candidate index is held back or a later release follows a different direction. A deployment decision must not retroactively rewrite the terms promised for completed work. Programs for improved models or rerankers would need their own comparative criteria, baselines and reward conditions, because executing a selected method and demonstrating an improvement are different contributions. [3], [6]

7.3 Economic boundaries

Company revenue, company expenditure, participant costs and protocol transfers require separate accounts. API customers pay Desearch for services, while eligible participants receive protocol rewards under applicable rules. Neither product adoption nor customer revenue creates an automatic claim to company income, equity or ownership of contributed assets. Economic viability therefore depends on measured costs, useful outputs and participation conditions; it cannot be inferred from emissions or customer growth alone. [6]

8. Retrieval and controlled product releases

The working retrieval design combines keyword matching for exact terms with vector matching for semantic similarity. The two paths generate candidates that ranking combines and orders. Keyword retrieval can help preserve exact names and identifiers; semantic retrieval can find passages whose wording differs from the query. Their relative usefulness depends on the workload, and the proposal does not select a fusion formula, ranker or numerical weighting. [5]

A query must be processed with models and indexes compatible with the active search version. The query model must match the document representations, and returned passages must refer to eligible source records. A result should provide enough supporting text and provenance for the application to inspect why it was returned. Retrieval relevance is not equivalent to factual verification: a relevant page can contain a false statement, and a source citation does not make the claim true.

The initial evaluation should isolate retrieval from downstream answer generation. Query processing time includes the operations necessary to return ranked sources and supporting text. If an application subsequently generates a summary or conducts multi-step research, those stages should have separate timing and quality measurements. This separation makes comparisons meaningful and avoids attributing a generation bottleneck to index retrieval or vice versa.

Candidate versions should be evaluated before promotion. The review should cover usefulness, response time, source quality, freshness and operating cost under a declared workload. A candidate that improves one metric can still be unsuitable if it introduces incompatible representations, missing sources, excessive resource requirements or unreliable behavior. Promotion therefore uses a set of product criteria, not a single offline score.

Release records should identify the candidate, the evidence considered, the decision and the rollback target. Serving should present a coherent version rather than silently mix components that have not been evaluated together. Monitoring should detect quality and availability regressions after promotion, with a defined response to unavailable dependencies and incomplete results. These are proposed operational requirements; their exact implementation and acceptance thresholds remain to be specified.

The first-phase search path is distinct from existing customer routes. The supplied direction notice does not migrate AI-search, X or social endpoints, change weights or open a new operator program. A later release must identify affected routes and operators, the effective change, compatibility arrangements, support and rollback. [8]

9. Threat model and failure analysis

The analysis below develops recommended safeguards and tests for this architecture. It distinguishes compliance with a task contract, fidelity to observed source material, and the factual truth of that material. A reproducible embedding can represent an inaccurate page correctly. Agreement among validators can also reflect shared evidence or shared error. The security objective is to state what each check establishes and expose the assumptions that remain outside it. [5], [6]

9.1 Contribution and assessment integrity

Contribution accounting should resist replay, duplicate credit and substitution of work from a different task. Candidate controls include binding records to task inputs and model versions, defining retry behavior, and retaining the evidence used to award credit. These controls must also recognize legitimate repetition: identical outputs may result from deterministic execution or authorized rechecking. Output correctness alone does not prove who originally performed the computation. Tests should cover duplicate submissions, altered identifiers and splitting one contribution across identities, measuring both unjustified credit and rejection of legitimate work.

Attackers may exploit the evidence and tolerance boundaries described in Section 6. A fabricated observation can be designed to resemble ordinary source variation; a wrong vector can be engineered to pass a permissive numerical check. Tests should pair these attacks with legitimate failures under the same checking conditions, so that stronger detection is not mistaken for improvement when it merely rejects more honest work.

Assessment can fail through collusion, copied judgments or selective checking even when output formats are correct. Full reproduction and sampled checking impose different costs and residual risks. Before selecting a procedure, the implementation should evaluate how predictable checks can be exploited, what evidence distinguishes independent assessment, and how conflicting judgments are reviewed. Reported assurance should describe the work actually checked. The published proposal appropriately requires sampled checks to be identified as such; a sample cannot establish independent verification of every item.

[5]

9.2 Product control and retained data

Team control creates an additional trust boundary around allocation and adoption. A queue operator could exclude contributors, assign disproportionately favorable tasks or privilege an internal candidate. The proposal calls for disclosure of material team participation, conflicts and product control. We recommend reviewable decision records, stated change authority and a challenge route that identifies who can correct an allocation or assessment error. Such procedures make decisions inspectable; their effectiveness still depends on operator accountability and the evidence available to challengers. [6]

Retained content also creates security and lifecycle obligations beyond contribution scoring. Candidate controls include restricting crawl destinations, isolating untrusted page processing and separating fetched content from operational instructions. Version records should connect accepted text, chunks, representations and active indexes so that deletion and incident recovery can be traced through the collection. Release tests should exercise incomplete builds, incompatible model versions and rollback after deletion. These tests address failures that a valid miner submission alone cannot prevent, including restoration of material that should remain removed. [5]

The resulting assurance is conditional on the chosen checks, evidence and operational controls. Protocol reward agreement, source authenticity, permitted reuse and dependable serving each need their own support. A useful security report will publish observed failures, false acceptance and rejection rates, checking costs and unresolved cases alongside the deployment conditions under which they were measured.

10. Evaluation and full-system economics

Desearch 2.0 requires separate evidence for contribution validity, retrieval usefulness, and the value of distributing work. Task assessment evaluates compliance with the published contract. Miner contribution scores aggregate assessed work under the program rules; neither establishes customer retrieval quality. This document reports no experimental results. The following studies define a proposed evaluation program whose workloads, sample sizes, configurations, and acceptance thresholds must be fixed before the final test results are examined.

10.1 Retrieval quality and judgment quality

The retrieval study will compare lexical BM25, dense retrieval, and their hybrid on the same frozen collection. Model versions, text preparation, document boundaries, candidate budgets, and hardware will be recorded. Shared preprocessing will remain fixed; method-specific tuning will receive comparable budgets. A reranker will be included only when an implementation is available. Component removal and replacement experiments will isolate the effects of hybrid retrieval, preparation choices, and reranking where applicable. Models will be selected for the intended workload before testing, rather than chosen retrospectively from favorable results.

Evaluation queries will cover exact entities, semantic requests, and tasks requiring several sources, with domain-specific results reported alongside aggregates. Development, validation, and held-out test queries will remain separate. For queries with relevant documents, the primary ranking measure will be $nDCG@10$, complemented by $Recall@k$ against the declared relevance judgments and, where appropriate, MRR . Recall over judged documents will not be described as recall over the web. BEIR motivates heterogeneous evaluation and also shows why an average score can conceal substantial differences between tasks. [16]

Relevance judgments will pool candidates from every compared system and hide system identity from assessors. The protocol will specify the rubric, assessor count, disagreement resolution, pooling depth, and treatment of unjudged documents. Judgment coverage and $Hole@k$ - the proportion of top-k results without judgments - will accompany scores. Additional assessment will address unequal coverage before attributing score differences to retrieval quality, following the bias identified in BEIR. Queries intentionally unanswerable within the test collection will form a separate set: evaluation will measure incorrect positive returns and correct empty responses under a declared response contract. They will not enter ranking averages with an undefined relevance denominator. [16]

10.2 Serving, freshness and verification

Serving measurements will report retrieval-only p50, p95, and p99 latency, throughput, and timeout/error rates at specified load, with cold and warm caches distinguished. Any answer-generation time will be measured separately. Comparisons will use a matched serving baseline and disclose quality differences; a faster configuration that returns less useful evidence will not constitute an unqualified improvement. A comparison with Desearch 1.0 will be included only if its configuration and workload can be reproduced sufficiently to support the stated conclusion.

Freshness will be tested through controlled sources with known publication, update, and deletion times. Measurements will include change-to-searchable delay, observation-age distributions, refresh success, and unique usable coverage within a declared source set. A static retrieval benchmark cannot establish these properties. Release tests will introduce partial builds, incompatible model/index versions, miner

unavailability, serving dependency failures, and failed promotions. Recovery measurements will include stale or inconsistent responses and whether rollback preserves deletions.

Verification tests will distinguish malicious submissions from legitimate variation and unavailable evidence. Cases will include incorrect models, missing or reordered vectors, nonfinite values, replayed results, changed sources, source outages, and benign numerical differences across hardware. Full and sampled checking will be compared by false acceptance, false rejection, unverifiable outcomes, attack detection, and computation cost. Scoring simulations will examine duplicate submissions, cheap-task selection, identity splitting, and validator collusion once allocation and reward rules are specified. Reported attack rewards will remain conditional on those rules and the assumed adversary.

10.3 Resource economics and reproducibility

The first economic study will hold assigned inputs and output-quality requirements constant while comparing team-operated execution with network execution, verification, and integration. It will measure time and cost per valid, unique usable output, including rejected work and retries. The accounting boundary will include miner resources, validator checks, provider fees, preparation, indexing, storage, serving, and operations. Company cash expenditure, participant costs, and protocol transfers will be reported separately. Token emissions will neither be treated as free computation nor added twice to the same resource ledger. Amortized query costs will disclose collection lifetime, refresh frequency, query volume, and sensitivity to these assumptions.

A separate future study would test whether independent contributors expand useful coverage under equal total budgets and a common discovery objective. It would measure incremental usable sources, held-out query answerability, and integration cost. This experiment is applicable only if contributors can discover or propose sources; executing identical team-selected inputs cannot establish a discovery benefit. Both studies may find that coordination and checking cost more than the benefits they provide.

Before testing, the study protocol will select sample sizes from pilot variability and required precision, and specify paired comparisons and confidence-interval methods. Reproduction materials will include code and configuration versions, permitted corpus/query manifests, hardware, seeds where relevant, logs, and raw measurements. Independent relevance evaluation will remain separate from miner reward assessment throughout. Publication will report failure cases and uncertainty as well as favorable results; empirical advantages remain unestablished until these studies are completed.

11. Related work and design alternatives

Bittensor provides the network and incentive setting in which Desearch proposes to organize contributions. Its foundational whitepaper frames a peer-to-peer market for intelligence and develops an incentive model under stated assumptions. Desearch uses that framing as motivation, while its initial work concerns assigned collection and representation tasks. It must define those tasks' utility and assessment criteria explicitly. Bittensor chain consensus, reward aggregation and Desearch's evidence checks are distinct mechanisms; the historical whitepaper does not specify the proposed CRAWL/EMBED scoring contract. [9], [10], [11], [17]

Search-system architecture provides the foundation for the maintained-index approach. Brin and Page describe a complete path from crawling and document representation to indexing, ranking, and query execution, explaining implementation choices through storage and processing constraints. Their paper also distinguishes measured system behavior from an incomplete evaluation of result quality. Desearch

adopts the need to explain both collection construction and query serving, while its proposal introduces a separate question: how independently operated contributors should be assessed and rewarded for work incorporated into that collection. Neither maintaining an index nor combining retrieval signals is claimed as a new primitive here. [12]

Decentralized protocols clarify the importance of narrowly stated verification claims. Bitcoin presents a transaction-history mechanism with an explicit computational-majority assumption and a corresponding attacker analysis. IPFS describes content-addressed objects and a modular peer-to-peer architecture; identifying bytes by a hash does not establish the truth or relevance of their contents. Filecoin separately defines storage-network properties, proof mechanisms, participant interactions, and markets, while identifying unfinished analyses and implementation work. These distinctions inform Desearch's separation of accepted work, data integrity, retrieval quality, and reward allocation. They do not supply a proof of Desearch's correctness: transaction ordering and storage verification address different objects and adversaries from crawling, embedding, and source selection. [13] [14] [15]

Information-retrieval evaluation supplies a complementary empirical foundation. BEIR compares retrieval approaches across diverse tasks and domains, making performance variation and computational trade-offs visible. Its analysis of incomplete relevance judgments shows that a system can retrieve useful documents and still be disadvantaged when those documents were absent from the annotation pool. Desearch's proposed evaluation therefore combines shared-corpus comparisons with pooled, blinded assessment and judgment-coverage reporting. Live-source freshness, contribution fraud, release consistency, and total operating cost require additional studies because fixed retrieval collections do not measure those properties. This paper's contribution is a design proposal for connecting these concerns; it does not claim a demonstrated improvement over existing retrieval systems. [16]

12. Openness, rights and implementation milestones

Open participation should be expressed through inspectable opportunities and rules. A program should state who is eligible, what resources are required, how submissions are evaluated, how credit is assigned, how decisions can be challenged and how changes take effect. Publishing these terms makes the network's opportunity understandable without implying that every dataset, source page or operating component is freely redistributable. [6], [8]

Source material, code and model weights can carry different usage conditions. Search access, retention, redistribution and use for training are distinct activities. A contribution should identify its origin and the rights needed for its intended use; Desearch remains responsible for integrating it under appropriate terms. A publicly reachable source is not by itself an unrestricted data license. The release process should preserve attribution and support applicable removal requirements.

The first reviewable checkpoint should demonstrate a bounded collection and representative queries with a clear reason for their selection. It should include inspectable content and source-record samples, a reproducible baseline and an evaluation specification. Results should report useful retrieval, latency, coverage, freshness, data quality and the full cost of building, checking and serving the collection. The checkpoint should also assess whether the data and evaluator are ready for a bounded reranking experiment, or identify the gap preventing it. [3]

A usable operator release must then provide task formats, installation and configuration, the selected model, measured hardware requirements, validation procedures and the scoring contract. Worked examples should cover successful, incomplete, duplicated and unverifiable work. Transition instructions

should identify what carries over and how operators recover from failures. These requirements belong before requests for new work or equipment commitments. [8]

Progress reporting should distinguish design decisions, implemented components, measured results and live releases. A commit can establish that code changed; a benchmark can support a particular measurement; a deployment record and runtime verification can establish that a version is serving. Each answers a different question. A larger corpus or more participants is insufficient evidence of a better product without corresponding quality and use measurements.

13. Limitations and future work

This draft leaves task allocation, eligibility, verification coverage and tolerances, score normalization, disputes and score-to-weight conversion open. It selects no numerical reward formula, search configuration or hardware recommendation. A bounded implementation and reproducible evaluation are needed to test its performance and economic hypotheses.

Desearch initially retains control of source selection, integration and serving. Independent participation must demonstrate value against competent team-operated execution, including verification and coordination costs. Coverage and freshness remain limited by source access, permitted uses, refresh budgets and the selected workload. A well-performing bounded collection can still omit important sources, and traceable records do not prove their contents true.

Future programs could target missing coverage, better refresh methods, extraction, representations or ranking. Each needs an appropriate baseline, including a declared parent model where relevant, and tests for index compatibility and practical quality, latency and cost. A strong isolated score would not authorize deployment. Additional storage or serving roles should be considered when they improve outcomes; no predetermined topology is required. Structured datasets, information feeds and data for other AI systems similarly require separate evidence of usefulness and suitable rights.

14. Conclusion

Desearch 2.0 proposes making reusable search capability the organizing object of network work. Assigned crawling and embedding tasks would produce source-linked information and compatible representations; validators would assess contributions; and Desearch would assemble and serve tested search versions. The architecture separates the evaluation of completed work from both protocol reward allocation and customer deployment.

The design is intended to let contributions remain useful beyond a single response. Its success depends on credible checking, fair and understandable rules, coherent data maintenance and useful search at a sustainable full cost. The next step is a bounded implementation and evaluation that tests those conditions directly. That evidence should determine which tasks, methods and participation models deserve expansion.

Appendix A. Worked contribution example

This example illustrates the proposed relationships. The identifiers are explanatory labels, not an API specification or reward quotation.

1. **Issue work.** Desearch selects a documentation URL and publishes task `crawl-A`, including required extraction settings, permitted retention and assessment rules. An eligible miner receives it under the program's allocation policy.
2. **Record the observation.** The miner submits the fetch outcome and required material linked to `crawl-A`, its contributor identity and observation `obs-A`. The record identifies the observation time and extraction version.
3. **Assess the crawl.** A validator applies the published checks and records the evidence scope. Valid completion can enter contribution accounting. An inaccessible or changed page follows the task's stated outcome rules.
4. **Prepare text.** The preparation process creates chunks `chunk-A1` and `chunk-A2`, linking both to `obs-A` and the preparation version. Duplicate handling preserves the provenance relationships relevant to the collection.
5. **Compute representations.** Task `embed-A` identifies the ordered chunks, selected document model and configuration. Its miner submits corresponding vectors. Embedding assessment records structural checks and the scope of reproduction.
6. **Build a candidate.** The team includes eligible material in `version-A`, which binds its collection, preparation, document/query models, indexes and ranking settings. Correct task execution has its own accounting outcome, regardless of this version's release decision.
7. **Evaluate and serve.** The team compares `version-A` against the declared baseline. If approved, a customer query can retrieve supporting text linked to `obs-A` without waiting for the original contributor to complete new work.
8. **Maintain the result.** A subsequent observation, correction or removal is propagated through the affected records. A rollback continues to respect removal restrictions.

Appendix B. What must be fixed before participation opens

Decision	Required published artifact
Initial scope	Source cohort, intended query families and explicit exclusions
Allocation and eligibility	Task-access rules, timing, reassignment, duplicate policy and team participation disclosures
Task contract	Versioned inputs, outputs, identity binding and failure semantics
Embedding configuration	Model/version, license, configuration and tested resource requirements
Validation	Evidence policy, full or sampled scope, tolerances, error handling and disputes
Contribution accounting	Units, aggregation windows, cross-task treatment, score-to-weight mapping and worked examples
Search release	Compatible version manifest, evaluation criteria, promotion and rollback procedure
Rights and maintenance	Attribution, permitted uses, deletion propagation and relevant retention rules
Transition	Affected operators/routes, effective change, carry-over, support and rollback

These decisions are required for an actionable program specification. The proposal can be reviewed before they are all settled, provided their status remains explicit.

References

1. Desearch. Desearch 2.0: the proposal. Pinned source package, commit e5eda3282de813a32a3b34204f2a82911e3e606d. [Source](#)
2. Desearch. Why Desearch 2.0. Pinned source package. [Source](#)
3. Desearch. Direction and first phase. Pinned source package. [Source](#)
4. Desearch. Company foundation. Pinned source package. [Source](#)
5. Desearch. First-phase technical overview. Pinned source package. [Source](#)
6. Desearch. Open network and Bittensor. Pinned source package. [Source](#)
7. Desearch. Incentives and rewards. Pinned source package. [Source](#)
8. Desearch. Participation and progress. Pinned source package. [Source](#)
9. Bittensor. Yuma Consensus. Official documentation, consulted 21 September 2026. [Source](#)
10. Bittensor. Emissions. Official documentation, consulted 21 September 2026. [Source](#)
11. Yuma Rao. Bittensor: A Peer-to-Peer Intelligence Market. Official whitepaper; web presentation consulted 21 September 2026. [Source](#)
12. Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks, 30, 107-117, 1998. [Source](#)
13. Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. [Source](#)
14. Juan Benet. IPFS - Content Addressed, Versioned, P2P File System. Draft 3, arXiv:1407.3561v1, 2014. [Source](#)
15. Protocol Labs. Filecoin: A Decentralized Storage Network. 19 July 2017. [Source](#)
16. Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava and Iryna Gurevych. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. NeurIPS Datasets and Benchmarks, 2021. [Source](#)
17. Bittensor. Chain consensus. Official documentation, consulted 21 September 2026. [Source](#)