The Epistemology Is the Compiler

Build-gate enforcement of evidential standards in an evidence-graded historical knowledge graph

Adala, Y. — DeepTunisia Project, deeptunisia.org

Release V0.1

Contents

1. Introduction . . . 2
2. Background and requirements . . . 4
3. The dataset in brief . . . 5
4. Related work . . . 6
5. System design . . . 8
6. Enforcement: the validator layer . . . 20
7. The hardening audit: seven ways the promise was bypassable . . . 21
8. Evaluation . . . 23
9. Discussion . . . 26
10. Future work . . . 30
11. Conclusion . . . 32
Data and code availability . . . 32
References . . . 33
Appendix A — Validator table (V1–V24) . . . 35

Release V0.1 — the first numbered release of this paper. The paper is henceforth versioned like the dataset it describes: each release is a stable, citable artifact built from the then-current graph, and every number in it is re-derivable from that graph at that date. The next release (V0.2) is planned to carry the empirical results this version scopes but does not yet report — the inter-annotator study (§10.1) and the sensitivity analysis of the influence-index constants (§10.2 item 1).

Keywords: knowledge graphs · evidence grading · provenance · build-time validation · epistemic basis · fuzzy temporal intervals · computational social science · verifiability

---

Abstract. Knowledge bases that document contested political history face a problem that policy cannot solve: their evidentiary standards are enforced socially — as editorial norms, review conventions, or post-hoc fact-checking — and social enforcement fails under volume, inattention, and adversarial pressure, and fails catastrophically in the era of language-model-generated fabrication. We describe DeepTunisia, a jurisdictional implementation of an open, source-backed framework for evidence-graded historical knowledge graphs: a knowledge graph of political, military, security and economic power in Tunisia (1956–2026) in which the evidentiary policy is *compiled*, not followed. Every claim record carries a nine-field evidence envelope — source, confidence grade, derived epistemic basis, attribution, reasoning, falsifier, disputes, review, verification status — and the build fails on any violation: an inferred claim without a stated falsifier, a low-confidence claim without a named claimant, a "fact" without a source, an impossible date, an unanchored influence assertion. Uncertain dates are represented as four-field fuzzy intervals whose clamping is published, never silent. An adversarial audit of the shipped system found seven distinct ways the enforcement could be bypassed — silently stripped fields, calendar-date rollover, inverted intervals, substance-free review records — all seven closed with validators and regression tests that fail on the pre-fix dataset, and

a mutation-testing campaign measures the strength of the enforcement rather than assuming it: 25 of 26 deliberate breakages detected, the survivor documented as by-design. The compiler is one half of the architecture; the other half is a human verification loop — the Agora — whose discussion is anchored to the records themselves, whose corrections can reach the dataset only through the same build gate that guards every other edit, and whose honesty rules are fixed before its features exist. The two halves answer questions neither can answer for the other: the compiler checks whether a claim is admissible under the rules it declares; the community checks whether the evidence actually warrants the status the record declares. Together with recompilation — accepted corrections re-entering the gate as changed records — they form a closed epistemic feedback loop that is this paper's research object. We report the architecture, the audit, and the boundaries the system does not claim: it makes the *status* of every claim structurally visible and structurally non-upgradable; it does not authenticate documents, and it does not judge whether an epistemic judgment is correct — only whether the record honours the judgment it declares. The build prevents declared epistemic constraints from being silently violated — the dataset cannot be quietly wrong — and because the enforcement layer is deliberately jurisdiction-agnostic, the framework can be re-instantiated for any other jurisdiction without weakening any of these guarantees. The contribution is the framework, not the claim that it is unprecedented; Tunisia is where it is first exercised.

---

1. Introduction

The research question behind DeepTunisia is deliberately narrow, because its purpose is to be answerable: which institutions, individuals and economic groups retain influence when the formal regime in Tunisia changes, and which relationships are genuinely continuous rather than reconstructed after each rupture — 1956, 1987, 2011, 2021. The project is frequently described as an exposé of a "deep state." It is not. That framing presupposes its conclusion, and a project that begins from a conclusion cannot be trusted to test it; the design question here is narrower and harder: **how does a public dataset make it structurally impossible for an unsourced inference to pass as a documented fact?**

It is worth stating the project's identity precisely, because it determines what this paper is about. **DeepTunisia is a jurisdictional implementation of an open, source-backed framework for evidence-graded historical knowledge graphs.** The Tunisia deployment is the reference implementation — the dataset on which the architecture was designed, built and evaluated. But the enforcement layer, the evidence envelope, the temporal model, the validators and the build gate are all deliberately jurisdiction-agnostic; the framework can be re-instantiated for any other state, province or community without weakening any of the guarantees this paper describes. What Tunisia is *about* is its subject matter; what the framework *is* is the answer to a general question: can the discipline of evidence-graded knowledge be compiled rather than merely recommended? The word "first" is deliberately avoided here as a claim about the world; it appears only as a statement about this project's own history (Tunisia is the first deployment this project has built). Whether the *framework* is unprecedented is a separate claim, graded C in §4 and subject to external review, exactly like the Table 3 novelty claim. §9 makes the portability of the architecture explicit and defines what a new deployment keeps and what it replaces.

The difficulty is not that evidence standards are unknown. Epistemologies of evidence are among the oldest artefacts of organised knowledge — the levels-of-evidence ladder in medicine [9,10], the verifiability doctrine of Wikipedia [24], the proof and trust layers of the original semantic-web vision [1,18]. The difficulty is that in nearly every existing system the standard is *policy*: it is written down, appealed to in disputes, and violated routinely. Wikipedia's verifiability rule is enforced by argument among editors, not by the software. A GRADE recommendation is a guideline a reviewer applies, not a constraint a database enforces. Fact-checking intervenes after publication, one claim at a time [11].

Two events make the gap between *policy* and *enforcement* no longer tolerable for public-interest data. First, the generation of plausible false documents has become effectively free: language models produce fabricated case law that has been cited in court filings [23], and undisclosed automated posting has demonstrated persuasiveness greater than human argument [20,21]. A dataset whose integrity rests on editors remembering to follow a policy cannot be the evidentiary base for claims about living people and operating institutions. Second, the volume of relevant public material — gazette decrees, registries, press — exceeds what careful manual curation can police by vigilance alone.

This paper reports an architecture that treats **the evidence policy as the compiler**. In DeepTunisia, YAML files in

version control are the canonical dataset; the build — a deterministic pipeline of validation, resolution, derivation and emission — is the enforcement mechanism; and the build *fails* on any breach. An inference without a stated falsifier does not enter the dataset. A grade-C claim that does not name its claimant does not enter the dataset. A date that cannot exist does not enter the dataset. The claim of this paper is not that DeepTunisia is unbiased or complete — it is neither, and it says so in its own published statistics — but that **the build prevents declared epistemic constraints from being silently violated**: a weaker claim cannot be upgraded to a stronger one by accident, inattention, or shortcut, because the machine refuses to compile it. We will use the shorter phrase "the dataset cannot be quietly wrong" as shorthand for precisely this narrower property — the reader should hear "cannot silently violate its declared epistemic contract", never "cannot be factually wrong". The precise definition, the residuals the gate does not cover, and the formal statement are each given once — below, in §8.3, and in §5.10 — rather than repeated as running caveats.

The phrase "quietly wrong" is used here with a precise, deliberately narrow meaning, and we state it now because the whole paper turns on it. **"Quietly wrong" means an epistemic-status degradation or schema-level violation that passes into the published dataset undetected by the build** — a claim rendered as more authoritative than the evidence warrants, a field silently dropped, an impossible date shipped as a different date, a contradiction resolved by arithmetic instead of recorded as a dispute. It does **not** mean factual falsity of the underlying source material: a forged gazette page, a misread source, a maliciously misgraded record, a validator bug, an ontology that encodes an inappropriate assumption, or a true claim that was never entered are all ways the dataset can be wrong that the gate does not and cannot detect. Those residuals are enumerated and owned in §8.3 and §9, and the formal property is stated in §5.10. The reason for the narrow definition is that the strong version of the claim — "the dataset cannot be wrong" — is undeliverable by any system, and a paper that implied it would be committing exactly the overclaim the architecture exists to prevent.

That narrowness is not a limitation the design tolerates; it is the architectural axiom the design is built on. We state it once, because it resolves most of the objections a reader might raise against a "compiled epistemology":

> **Axiom A1.** The compiler verifies that a claim conforms to its declared epistemic contract. It does not determine whether the underlying epistemic judgment is correct.

The distinction is the whole architecture. *"Is this claim documented, reported, inferred or unsubstantiated?"* is a judgment about the world, made by an editor from sources; the compiler cannot and must not make it. *"Given that an editor declared this claim* `inferred`*, does the record carry the reasoning and the falsifier the contract requires?"* is a judgment about the record, and the compiler enforces it perfectly. The system therefore never automates epistemology — it automates the *enforcement of an epistemic constitution*, while leaving epistemic judgment contestable by humans (§9). It does not judge truth; it refuses to compile a record that violates its own declared contract.

The division of labour A1 implies is the paper's central architectural claim, and it governs everything that follows. The system answers two questions, with different organs. The compiler answers the *formal* one — **is this claim admissible under the rules it declares?** — checking sources, basis-consistency, falsifiers, calendar dates and graph anchors mechanically and uniformly, on every record, at every build. The community answers the *substantive* one — **does the evidence actually warrant the status this record declares?** — judging whether a document supports a claim, whether a grade is more generous than the evidence deserves, whether three apparently independent newspapers all copied the same original report. Neither can do the other's job: the compiler cannot decide whether a source means what the record says it means, and no community can be trusted to remember every falsifier rule on every record, forever, under pressure. A community without the compiler can require sources and still miss the one record among thousands whose source was silently stripped; a compiler without the community can certify that every claim carries a source and ship one that is irrelevant, misunderstood, or circular. The two layers close each other's blind spots, and the loop that joins them — claim → compiler → publish → scrutiny → correction → recompilation — is the research object this paper reports. The machine half of that loop is experimentally validated in §7–§8; the human half is architecturally specified in §5.9, and its empirical validation is a defined study (§10.1) rather than an aspiration.

The paper's contributions are four, and each is stated at the level of what the system demonstrably does:

1. **An evidence envelope as an executable contract** (§5.2–5.4): nine fields composed uniformly across every claim-bearing record kind, with epistemic *basis* derived from authoring shorthand by a published, pinned mapping; four-field temporal intervals whose contradictions fail the build and whose every clamp is published; and derivation that is computed, labelled, and never silently unprovenanced. The envelope is not a semantic claim about identity

— it is the minimum contract every claim must honour.

2. **A verified enforcement layer** (§5.10, §7, §8.1): the build gate expressed as a formal contract over four constraint families; an adversarial audit that found seven real bypasses in the shipped implementation and closed each with a validator and a regression test; and a mutation-testing campaign that measured the pipeline's ability to detect a broken validator — from three of twelve deliberate breakages detected in the first pass to twenty-five of twenty-six in the extended campaign, with the latent class closed by synthetic invalid fixtures and a fixture-pipeline runner, and the permanent closes the campaign earned (the deriveBasis truth table, the interval-trims freshness check, the V23 calendar round-trip, the rule-2 ratchet). The theorem is stated *with its condition*: the validators are trusted code, and the strength of that trust is now a measured number, not an assumption.
3. **Honest self-measurement as a design property** (§8): published statistics that the build itself rewrites from the graph, risk-bucketed review coverage, per-tier translation coverage, and coverage audits that surface what the map does *not* contain — so the project's prose cannot drift from its data, and its shortfalls are published rather than hidden.
4. **A portable epistemic infrastructure, not a Tunisia database** (§5.9, §9): the enforcement layer is jurisdiction-agnostic — a new deployment (DeepMorocco, DeepCalifornia, DeepLagos) forks the repository, replaces the canonical dataset and jurisdiction-specific ontology, and retains the evidence envelope, validators, build gate, audit machinery and community layer, independently operated and independently responsible for its evidentiary judgments. The Agora completes the loop: the graph creates the discussion, the discussion proposes changes, and only the compiler changes the graph — with discussion anchored to records, identity designed so a server compromise impersonates nobody, votes that rank but never grade, and petitions whose honesty rules are fixed before the feature exists. The loop is the architecture: the compiler admits, the community warrants, and only recompilation changes the record.

The system is source-available and runnable (`npm run data && npm run test`), and the full dataset ships as downloadable JSON and CSV beside the site. Licensing status is stated precisely in the availability section — this paper deliberately does not use the term "open source" until a license is in force, because a paper whose thesis is that an unenforced claim must not pass as an enforced one should not make an unenforced licensing claim itself. At the time of writing, the graph holds 874 sources, 175 institutions, 116 roles, 422 people, 416 positions, 290 relationships and 84 events, plus first records in the corporate and documentary layers [4].

2. Background and requirements

2.1 The evidentiary problem, stated as a requirement

A dataset about influence in a single country under successive regimes has properties that make social enforcement of evidence standards uniquely fragile:

- **The subjects are alive.** Named individuals may be heads of state or ministers at the time a reader consults the record. The cost of a false claim is not abstract.
- **The sources are asymmetric.** Official gazette decrees (tier 1) give exact dates; the press (tiers 3–4) reports relationships; allegations circulate (tier 5). A system that cannot distinguish a decree from a rumour is worse than none, because it launders the rumour into the same visual language as the decree.
- **The incentive to fabricate is real and increasing.** Fabricated documents are cheap to produce and hard to detect at a glance; a single forged gazette page entering the dataset and being cited by a journalist destroys the project's entire value and hands ammunition to the very institutions it documents [6].
- **The dataset is a public good, not a private opinion.** Its integrity must be checkable by readers who have no reason to trust its authors. "Trust us" is not an architecture.

The requirement that follows: **the distinction between evidence grades must be enforced by the system, not requested of the editor**; and the system must make its own enforcement auditable — including by adversarial audit of the enforcement itself.

2.2 Design principles

The architecture rests on eight principles, each mapped to a mechanism (§5–§7):

#	Principle	Mechanism
P1	The evidence envelope belongs to every claim-bearing record	shared schema composition (§5.2)
P2	YAML in git is canonical; the build is the validator; git is the audit log	`build-data.ts` (§6)
P3	The graph is the product; every layer ships as downloadable data	JSON/CSV/GeoJSON exports (§6)
P4	Machines propose; humans dispose	agent outbox + proposal state machine (§5.8)
P5	No inference renders as a fact; every computed measure is labelled computed	basis rendering, `derived` flags (§5.5, §5.7)
P6	Migration is additive; existing data stays valid	schema extension, no rewrites
P7	Numbers are claims	every number carries a date and provenance (§5.3)
P8	Direction and time are explicit on every typed edge	direction table + intervals (§5.4, §6)

Table 1 — design principles, each tied to the mechanism that enforces it.

3. The dataset in brief

The graph models positions as the join between a person, a role and a span of time: `Position = (role, holder, interval)`. Time-travel queries — "who held power in 1994" — are answered from intervals, never from lists. Around the position join sit people, institutions, roles, a ~30-type relationship taxonomy with explicit direction semantics, first-class events with causal edges, agreements, and — since v0.0.2 — companies, contracts, licences, declarations and education records, a region/place gazetteer, and per-entity derived timelines.

The ontology grows additively: new record kinds compose the same claim envelope as the original ones, so a new kind inherits the entire epistemic contract on the day it is introduced. A record kind with zero records is valid — the schema is the feature gate, and no invented data enters the graph to fill a column.

Record kind	Count	Claim-bearing	Notes
sources	874 (836 cited)	—	tier 1–5, archive URLs
institutions	175	yes	incl. corporate-like
roles	116	—	canonical offices with authority weights
people	422	yes	trilingual, aliases
positions	416	yes	the time-join
relationships	290	yes	~30 types, directed
events	84	yes	causal edges, rupture flag
agreements	7	yes	treaties/accessions
questions / hypotheses	25 / 6	—	the open research agenda
companies / contracts / licences / declarations / education	8 / 12 / 11 / 8 / 21	yes	v0.0.2 kinds, first records
regions / places	84 / 12	yes	1 country + 6 regions + 24 governorates + 53 delegations; 12 point-places

Table 2 — dataset scale. Build snapshot 2026-08-08; historical data cutoff 2026-07-26 (the two dates are distinct by design — the build can run on any date without moving the research cutoff).

Counts are the build state quoted in the canonical specification [4] and drift with every commit; the build publishes the live numbers.

4. Related work

The systems closest to DeepTunisia fall into five families. We position against each in turn; the argument is summarised in Table 3. The comparisons are made in the same evidentiary spirit as the data: a prior-art claim is itself a claim, and the strongest form we can defend is that we did not identify a directly comparable system combining these properties — not that no such system exists. The paper's architecture does not depend on being first; a nearer neighbour discovered tomorrow would sharpen the positioning, not invalidate the contribution.

Knowledge graphs and statement models. Wikidata [2,3] is conceptually the nearest neighbour: a statement carries references attached to the claim rather than the page, and a rank system (preferred / normal / deprecated) retains statements known to be wrong rather than deleting them — an instinct shared with this project's rule that unsubstantiated claims are kept, never deleted. What Wikidata does not have is an *epistemic basis*: a deprecated statement is marked wrong, but nothing records whether a statement is documented, reported, inferred or circulating, and nothing requires an inferred statement to state its reasoning and its falsifier. The general knowledge-graph literature [5] concerns construction, storage and querying, not evidence grading.

The semantic-web trust vision. The original semantic-web programme explicitly included proof and trust layers above the data [1,18]. Twenty-five years on, those layers remain a vision: mainstream RDF/SPARQL practice carries no confidence contract, and "trust" was never operationalised as a machine-checkable gate. DeepTunisia can be read as one implementation of that vision, built twenty-five years late and on a stricter schedule: the trust layer is the build.

The compiler half of the metaphor. The title's claim — the epistemology is the compiler — has a second, independent grounding in programming-language research that the epistemology literature alone does not supply. Design-by-contract [25] makes the obligations of a software component machine-checkable: a routine that violates its precondition or postcondition fails at runtime, by construction, rather than being "expected to behave." Proof-carrying code [26] goes further, moving the proof obligation to the producer of a binary and having the consumer verify it locally before trusting the artefact. Both share the structural insight this paper builds on: **a standard that is stated but not enforced is not a standard, it is a hope** — the enforcement mechanism is the contribution, not the rule itself. DeepTunisia differs in what is being enforced and how: design-by-contract polices *program behaviour*, proof-carrying code polices *code*, and DeepTunisia polices *the epistemic status of claims in a dataset*. The enforcement substrate is also different — a deterministic build pipeline rather than runtime assertions or a proof checker — but the family resemblance is real, and it is the compiler half of the title's metaphor. A formal continuation of this line is future-work item 3 (§10.2): a machine-checkable specification of the validator layer, in the spirit of refinement types or a Datalog/SMT encoding, which would let the enforcement itself be checked by a checker.

Evidence grading in medicine and law. The GRADE system [9] and the Oxford levels of evidence [10] are the mature, decades-old statement that not all evidence is equal and that the grade must be visible to the consumer. This is exactly the position of the present work — with the crucial difference that GRADE is a guideline applied by human reviewers, while DeepTunisia encodes the ladder as a compile-time contract with automated consequences. The medical community separated "grading the evidence" from "strength of recommendation"; DeepTunisia separates "how strong is this claim" (confidence) from "what kind of claim is this" (basis), and the two axes are deliberately not collapsed (§5.3).

Provenance. The PROV family [7] records *lineage* — which process produced which entity, and who was responsible. Lineage is orthogonal to epistemic status: a perfectly recorded lineage can attach to a rumour. DeepTunisia keeps provenance (sources, review records, dispute ownership) and adds the axis PROV lacks: a graded, machine-checked statement of what kind of claim the record makes. Temporal conflict resolution in RDF knowledge bases [8] addresses inconsistent dates algorithmically; this project takes the position that a contradiction is an editorial fact to be recorded with its parties — never resolved by arithmetic (§5.4, V22).

Verification and fact-checking. Fact-checking is a post-hoc, per-claim human intervention after publication [11,12];

claim-verification research aims to automate it [12]. DeepTunisia is structurally different: its gate is *pre-publication* (nothing ships unvalidated), *uniform* (every record, not a sample), and *cheap* (deterministic schema and graph checks, not classifiers). The fabrication threat that motivates current verification research [20,21,23] is precisely the threat this architecture prices in at design time.

Three evidence-adjacent lines that must be distinguished from this work. A reviewer's own literature check will land on three families that are *adjacent* to evidence grading but enforce something different, and it is worth stating the difference precisely rather than letting it be inferred.

Probabilistic knowledge bases. The never-ending-learning tradition [28] extracts a knowledge base and attaches a **confidence score** to each extracted fact, learned from the extraction pipeline's own consistency signals. This is genuinely a graded claim — but the grade is *probabilistic and machine-assigned*, reflecting how confident the extractor is, not what kind of claim it is. A fact scored 0.74 in NELL is not marked "documented, reported, inferred or unsubstantiated"; it is marked "74% likely to be a true instance of this relation." DeepTunisia's basis is *epistemic* — a category about the relationship between a claim and its sources, assigned by a human under a rubric and enforced by the build — not a probability of truth. The two are complementary (a probabilistic extractor could propose candidates into the Agora outbox), but they answer different questions.

Trust management. The trust literature, surveyed by Artz and Gil [29], is about **who to believe**: reputation metrics over sources, trust propagation through social networks, policy-based trust negotiation. It computes a trust value for an *information source or agent* and uses it to filter or weight what that source says. DeepTunisia does not compute trust in sources; it records the *status of claims* — and it deliberately declines to derive one from the other, because a low-trust source can still state a documented fact and a high-trust source can be wrong. The threat-model table of §9 assigns "bad interpretation" to the community, not to a trust score.

Claim structures. schema.org's ClaimReview [30] is a **data model for publishing fact-check verdicts**: it marks up, in a machine-readable form, that a fact-checking organisation reviewed a specific claim and rated it (e.g. "True", "Pants on Fire"). It standardises the *output* of fact-checking — the verdict and the claim it attaches to — for search-engine consumption. It does not enforce anything about how the verdict was reached, what evidence supports it, or whether the rating was consistent with the claim's own epistemic status. DeepTunisia could emit ClaimReview markup as an export (a natural extension of the RDF-export item, §10.2), but the *machinery of grading* is the contribution; the markup standard is a channel for it.

The distinction that runs through all three: **each neighbouring line attaches a number or a category to a claim without making the claim's epistemic obligations mechanically enforceable.** NELL grades confidence; trust management grades sources; ClaimReview publishes verdicts. DeepTunisia grades *claims* and compiles the obligations that follow from the grade. The relationship is stated here precisely so the novelty claim of Table 3 is not read as sweeping these lines aside — it is read as identifying the axis they do not occupy.

System / family	What it has	What it lacks	DeepTunisia
Wikidata [2,3]	statement ranks, references on claims	no epistemic basis; no falsifier requirement	basis ladder + build gate
Semantic-web proof/trust [1,18]	the vision	never operationalised	the trust layer *is* the build
GRADE / CEBM [9,10]	evidence ladders	human-applied guidelines	the ladder is a compile-time contract
PROV [7]	lineage	no epistemic status of the claim	lineage + graded status
Temporal RDF [8]	conflict resolution by algorithm	silent arithmetic on disagreement	disputes recorded, never resolved by arithmetic
Fact-checking [11,12]	post-hoc verdicts	after publication, one claim at a time	pre-publication, uniform, deterministic
Wikipedia verifiability [24]	the doctrine	social enforcement	machine enforcement

The evidence for Table 3 is narrower than the table, and the paper says so. The claim rests substantially on an internal prior-art study [15] that examined *collaboration platforms* — Discourse, NodeBB, Flarum, Wikibase, PR workflows — and concluded the graph-object-as-root model with a CI merge gate "has no prior art to adopt." That study was scoped to collaboration infrastructure, not to the epistemic and temporal literature engaged above; and the study, the specification [4], the authenticity note [6] and the roadmap [16] are all internal project documents. By the paper's own grading rubric, a claim backed by a single unreviewed source is **grade C** — and that is exactly how the novelty claim of Table 3 should be read until an external literature review (§10.2 item 2) either corroborates it or finds the nearer neighbour this paper says it did not. The table is the strongest form the current evidence supports, not a settled result; the contribution does not depend on the claim surviving that review unchanged.

Table 3 — the novelty position. We did not identify a prior system that combines all of the following in a single enforcement layer: claim-level epistemic basis, mandatory evidence-dependent obligations (a falsifier requirement, an attribution requirement), temporal uncertainty represented as four-field ranges with published clamping, and build-time enforcement across a heterogeneous historical knowledge graph. The claim is deliberately narrow and stated as such: it is what the search found, not what exists everywhere — and a nearer neighbour discovered tomorrow would change the positioning, not the architecture.

5. System design

5.1 The pipeline at a glance

Vector source: `output/paper/fig1-pipeline.svg` *(editable diagram source kept in the repository).*

The pipeline is deterministic in the sense that matters: the same `data/` directory produces the same *graph* — the same records, the same derived values, the same validator outcomes — every time, byte for byte in the records themselves. (The emitted bundle additionally carries a build timestamp and the git changelog, which are metadata about the build, not the graph.) That determinism is what makes "the build is the validator" meaningful — there is no separate runtime path in which an unvalidated record could leak into publication, and no randomness in which validation outcomes depend on.

5.2 The claim envelope

Every claim-bearing record — people, positions, events, institutions, relationships, agreements, companies, contracts, licences, declarations, education, places — composes the same nine-field envelope via one schema function (`withClaimEnvelope`). A new record kind cannot forget a field, because the envelope is a composition, not a convention.

The envelope is a **minimum common contract, not a claim to semantic identity**. A person, an event and an influence assertion are not epistemically identical objects, and the architecture does not pretend they are: the envelope guarantees that every claim-bearing record carries the same *epistemic obligations* (a source, a graded status, a derivable basis, attribution where weak, a falsifier where inferred), while each kind adds its own semantics above the contract — a relationship carries direction and equity, a contract carries an award, a licence carries a term. The conceptual shape is thus `ClaimEnvelope` as a base with `PersonClaim`, `EventClaim`, `RelationshipClaim`, `InfluenceClaim`, and so on as extensions — not one universal object into which all knowledge is squeezed. The risk the universal envelope creates — that "carries the same minimum" silently becomes "has the same meaning" — is held in check by the requirement that every kind-specific field is itself declared, validated and published (§3, V19); a field that matters only to one kind is never smuggled into the shared contract.

Field	Type	Rule
`confidence`	`A–D`	A: primary/official record · B: several credible secondaries · C: single source or reasoned estimate · D: circulating, no reliable evidence
`basis`	derived, may be overridden	documented / reported / inferred / unsubstantiated (§5.3)

Field	Type	Rule
`verification`	`verified` / `needs-primary-source` / `disputed`	drives derivation and the open-questions page
`attributed_to`	string	**mandatory for grades C and D** — who is making this claim (V20)
`reasoning`	string	**mandatory when basis is `inferred`** (V18)
`falsifiable_by`	string	**mandatory when basis is `inferred`** — what evidence would refute it (V18)
`disputes`	list	competing versions of the same fact, recorded with holders — never resolved silently (V22)
`review`	`{by, date, method, note}`	provenance of the human check, method is an enum, date calendar-valid (V23)
`sources`	slug list	**`≥ 1` on every claim** — the build fails otherwise (V12)

Table 4 — the evidence envelope.

None of the nine fields is individually new: confidence grades exist in medicine [9,10], references-on-claims in Wikidata [2,3], lineage in PROV [7], attribution and review in journalism practice [11]. That observation is correct and misses the point. The novelty is not any single field but **four properties of the envelope as a whole**, each of which the neighbouring systems lack in combination: (1) *composition* — the nine fields travel together as one schema block applied uniformly to every claim-bearing kind, so no record kind can silently carry less metadata than another and no editor has to remember which fields a given kind supports; (2) *derivation* — `basis` is computed from `confidence` and `verification` by the build according to a published, deterministic mapping, rather than asserted by the editor; the machine does not decide whether a claim *is* documented in the philosophical sense — the editorial rubric does — but it does determine the *operational* epistemic class from the author's declared confidence and verification state, and an editor cannot quietly upgrade a claim by omitting a field (H1, V18); (3) *consequence* — the derived class triggers automated obligations (a falsifier requirement, an attribution requirement), so a weak claim carries the cost of its weakness at entry rather than at review; and (4) *failure semantics* — a record that violates any field does not merely fail a lint check: the build emits nothing, and the change cannot reach publication by any path. Each neighbour in §4 has one or two of these properties in some form; we did not identify one that composes all four into a single enforcement layer (§5.10).

Two design decisions in the envelope deserve emphasis because they are the paper's core.

First: `basis` describes the claim, not its dates. A position record bundles two separable claims — "X held this post" and "between these dates" — and they routinely have different standing. A security chief whose office is well reported but whose start date nobody has published is a *reported* officeholding with an *inferred* span. Collapsing them would make every imprecisely-dated fact look like speculation; the envelope keeps them apart, with the span carrying its own uncertainty through the interval machinery (§5.4).

Second: basis is derived, not authored. Authors write `confidence` and `verification`; the build derives `basis` unless it is explicitly overridden:

```
deriveBasis(confidence, verification):
  explicit basis present        → explicit
  confidence = A                → documented
  confidence = D                → unsubstantiated
  confidence = C, verification = needs-primary-source → inferred
  otherwise                     → reported
```

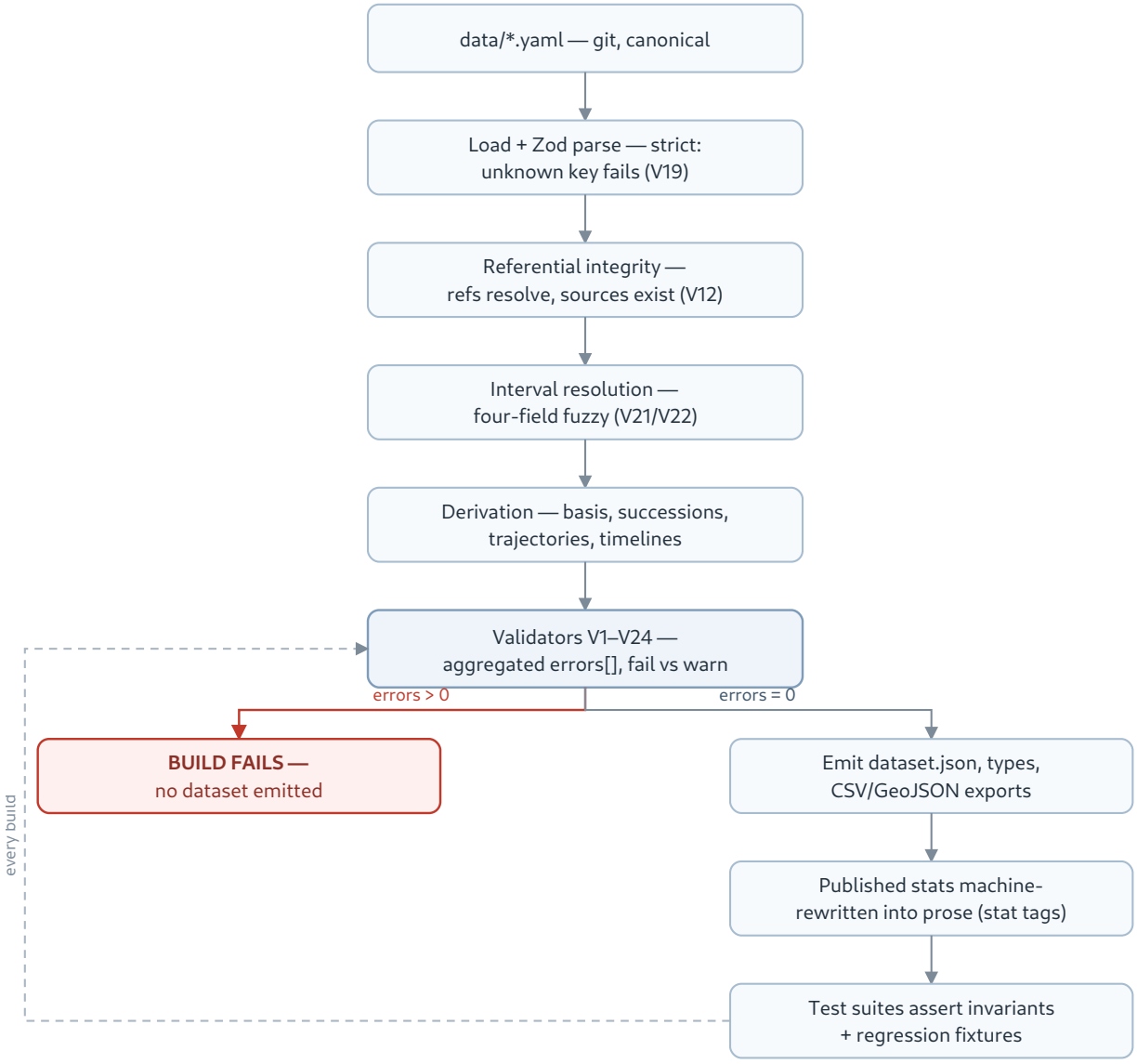

Figure 1: The build pipeline. The gate sits between validation and emission: any error means no dataset is produced.

The important derivation is the third row: a grade-C claim without a primary source is *not* a weak report — it is a *reasoned estimate*, and it therefore triggers the falsifier requirement. A system that collapsed "thinly sourced report" and "reasoned inference" would let an editor's guess wear the clothes of journalism. Deriving rather than authoring basis matters for another reason: the check runs on the *derived* value, so an author who writes `basis: documented` on a D-grade claim is overridden, and an inference that inherits its status from a `needs-primary-source` verification cannot escape the falsifier requirement by omitting the field (H1/V18, §7).

The derivation is a compiler primitive, not a convenience. `deriveBasis` is the single highest-leverage function in the architecture: if its mapping is wrong, every downstream obligation can be wrong while the build stays green. It is therefore specified completely, by its full truth table — every combination of its inputs, with no implicit cases:

`confidence`	`verification`	derived `basis`
A	verified / needs-primary-source / disputed	`documented`
B	any	`reported`
C	verified	`reported`
C	needs-primary-source	`inferred`
C	disputed	`reported`
D	any	`unsubstantiated`
(explicit `basis` authored)	—	the authored value, overriding all rows

The table is complete by construction — six outcome rows plus the explicit-override row, no `otherwise` clause hiding a case. Two of the six rows collapse inputs deliberately: A maps to `documented` under *all three* verification states, and D maps to `unsubstantiated` under all three, so the table shows each as one row. The regression test (§8.1) therefore enumerates the full **fifteen input combinations** — the twelve confidence×verification pairs plus three explicit-override cases, each chosen to prove the authored value beats the derivation: an authored `reported` over a C/needs-primary claim (which would otherwise derive `inferred`), an authored `inferred` over an A claim (which would otherwise derive `documented`), and an authored `documented` over a D claim (which would otherwise derive `unsubstantiated`). The override row in the table is one row; the test pins three instances of it, so the collapse is a display choice, not a hidden case, and a reader can verify the pinned test against the table row for row. No other place in the build may derive a basis; the function is the single epistemic transformation, and the four-field interval resolver is its only sibling at the same level of trust. Elevating these two functions to "compiler primitives" — specified, exhaustively tested, published — is what the formal-account future-work item (§10.2) would turn into a machine-checkable guarantee.

Vector source: `output/paper/fig2-epistemic.svg` (editable diagram source kept in the repository).

5.3 Numbers are claims

Consistent with P7, a number never floats free. A company capital figure is stored as `{ tnd, date }` — "a primary document shows capital X at this date", not "the capital is X". A contract award value is a fuzzy token that must either be backed by a tier-1/2 source or carry `attributed_to` naming who reports the figure (V4/V5). Equity percentages are bounded (V1), beneficial ownership requires its own attribution (V13). The same discipline that governs a date governs a number: the value and its provenance travel together.

5.4 Fuzzy time: four-field intervals with published arithmetic

Historical personnel records are rarely precise. A source may say a general was "appointed 1 June 2018", or "in post by at least December 2025", or "around 2017". Collapsing these into a single timestamp forces the dataset to invent precision it does not have — precisely the failure mode the project exists to avoid. Every date token therefore resolves to a *range* with four fields plus a status. The representation is a deliberate working out of the interval-based temporal reasoning that Allen introduced [19] and that the semantic-web time ontology later formalised for machine-readable content [17], applied to the specific problem of source-graded historical records: where those traditions represent

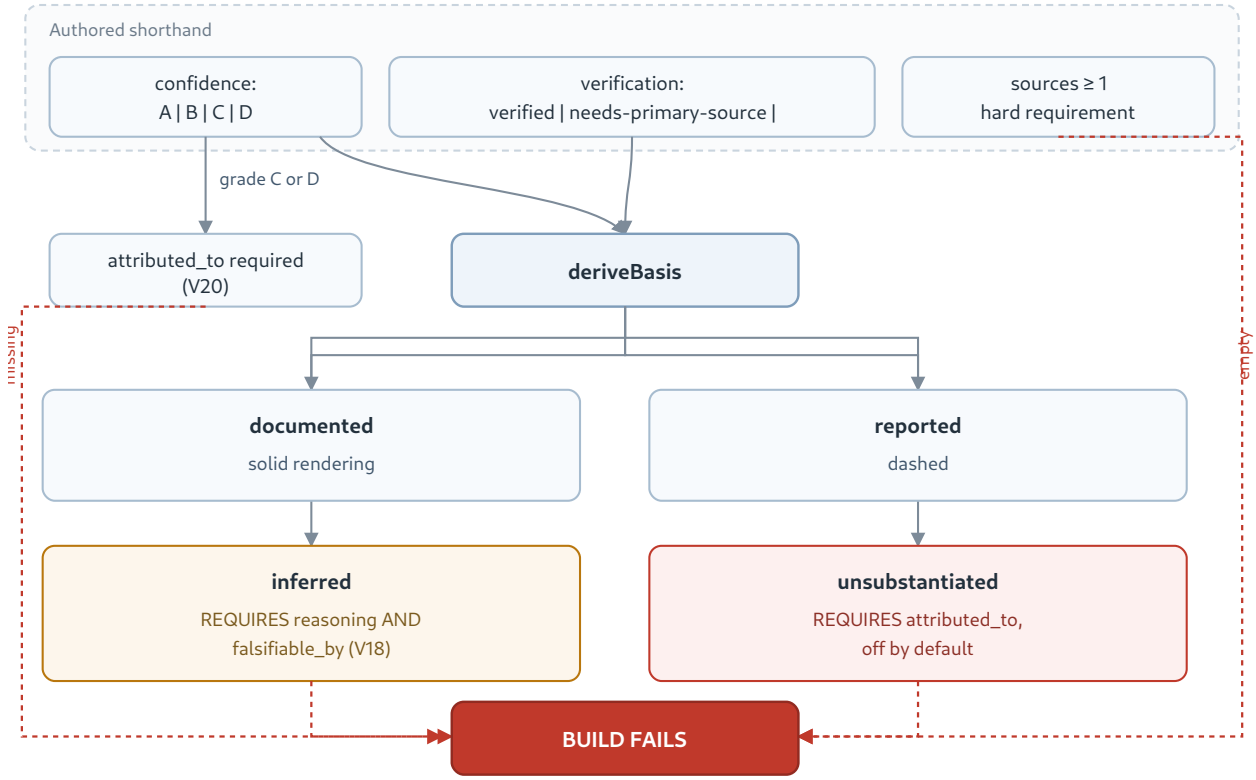

Figure 2: The epistemic flow. The four bases never render identically anywhere in the product (solid / dashed / dotted / dot-dash).

uncertainty as an ontology, DeepTunisia represents it as a four-field range with a published resolution procedure (§5.4's grammar, V21/V22).

```
ResolvedInterval = {
  startEarliest, startLatest,     // the window in which the interval may have begun
  endEarliest, endLatest | null,   // the window in which it may have ended; null = open
  startPrecision, endPrecision,    // day | month | year | approx | before | after | unknow
   status,                        // ended | ongoing | last-verified | unknown
   raw,                             // the original tokens, kept for display
}
```

Token	Meaning	Resolution
`2018-06-01`	exact day	single instant (calendar-validated, V21)
`2018-06` / `2018`	within the month / year	month/year bounds
`~2017`	approximately	widened ±1 year
`<=2018-06`	in post by then, start unknown	bounded backwards to a stated 8-year assumption, not to 1956
`>=1984`	no earlier than	bounded forwards similarly
`verified:2025-12`	documented in place at this date, may have continued	endEarliest = verified date, endLatest = cutoff, status `last-verified`
`ongoing`	positively confirmed at the cutoff	open end
`?`	unknown	full floor-to-cutoff envelope, status `unknown`

Table 5 — the date grammar. Two modelling assumptions are explicit and published as data, not hidden in code: the 8-year backwards window for <= tokens, and the dataset floor (Tunisian independence, 1956) and cutoff (2026-07-26).

The query semantics are specified, not left to the reader's guess. A four-field interval is a representation; what a *query* does with it is a separate decision, and the paper states it because "who held power in 1994" has a precise answer only if the inclusion rule is published. Two predicates define it, and they are the same two the implementation exposes:

- **`certainlyActive(interval, t)`** — true only inside the *confident core*: `t ≥ startLatest` and (`t ≤ endEarliest`, if the end is known). This is the conservative reading: the person *definitely* held the post at `t`, on every interpretation of the fuzzy bounds.
- **`possiblyActive(interval, t)`** — true anywhere in the *outer envelope*: `t ≥ startEarliest` and (`t ≤ endLatest`, if bounded). This is the inclusive reading: the person *may* have held the post at `t`, consistent with some interpretation of the sources.

The two are combined into a three-valued status (`certain` / `possible` / `not active`) that the views use deliberately: the Chronicle and the /now view render a post as held when the interval is *certain* at the displayed instant, and render the hatched ambiguity *around* it when only *possible*; the indices and the Network use `possiblyActive` as the inclusion predicate at the selected time, so a fuzzy-boundary officer is counted at the fringe but the ranking communicates the uncertainty through the interval's rendering rather than through a false-precision inclusion. The choice is published rather than buried in code: the conservative core answers "definitely held power"; the inclusive envelope answers "consistent with the record." A query that does not say which it wants is a query that has not asked a well-formed question.

Three properties of this machinery matter for the paper's argument:

1. **Impossible dates fail the build.** The grammar rejects non-date tokens at the schema boundary, and calendar validation round-trips every day-precision token: `2018-02-31` does not silently become `2018-03-03`, it *fails* (V21; §7, H4).

2. **Contradictions are recorded, never resolved by arithmetic.** A genuine contradiction (`endLatest < startEarliest`) fails the build unless the record carries a `disputes` entry documenting the disagreement. An over-wide *envelope* — a fuzzy start that stretches past a known end — is clamped by exactly one correct rule (a fuzzy start cannot be later than the known end), and **every clamp is published** in `interval-trims.json` with the record, the original span and the resolved interval (V22; §7, H5). Trimming is never silent.
3. **`last-verified` is not `ended`.** The four statuses render distinctly in every locale: "confirmed in place at this date, may have continued" is a different claim from "ended", and the product's `/now` view divides on exactly that distinction. The four-field model exists so that uncertainty is *shown* — hatched, dotted, labelled — rather than resolved behind the reader's back.

The modelling constants are stated with their rationale, and their sensitivity is an explicit deliverable. Three constants shape published behaviour and deserve the same evidentiary treatment as any other claim, because each changes query answers or rankings:

- **The 8-year backwards window** (for <= tokens, e.g. "in post by 2013, start unknown") exists because a genuinely open lower bound would silently place the officer in every earlier era, inflating survival metrics. The stated rationale is that eight years is a plausible ceiling for a senior Tunisian command tenure — the longest documented run in this dataset is about seven and a half years — and the bound is a *modelling assumption about plausible tenure*, not a fact about any person, which is why such edges still render hatched (§5.4 grammar).
- **The ±1 year slack** for ~ tokens (a quarter for month-precision guesses) encodes that an "approximately" date is uncertain to roughly the granularity it names.
- **The basis-discount weights** of the influence index (§5.7: documented 1.0, reported 0.55, inferred 0.2, unsubstantiated 0) encode that influence evidence is worth less as its basis weakens — and that unsubstantiated evidence is worth nothing in a published score.

Each of these is configuration, not settled value: the rationale is stated, the constant is emitted in the published metadata (so a reader can recompute), and the sensitivity of the *published rankings* to perturbations of the weights and the window is an explicit deliverable of future-work item 1 (§10.2). Until that analysis runs, the constants are honest choices with stated reasoning — not hidden assumptions.

5.5 Derivation, never authored — and never unprovenanced

Successions, contradictions, gaps, trajectories and per-entity timelines are computed at build time, never hand-listed. The point is structural: a gap or overlap in a command chain *surfaces* instead of being papered over, because the derivation notices it whether or not an editor would. Succession thresholds are published as data (`successionMeta`: gap/overlap years, the unknown-end exemption) so every published count is reproducible from the emitted constants (V24) — a threshold that lives only in code is treated as a bug. Derived values are labelled derived: a person whose trajectory was computed from position records carries `trajectoryDerived: true`, and derived trajectories are refused when they would merely restate a single post. The same rule governs the visual layer: no inference renders as a fact (P5), and a computed measure is always labelled computed.

A derived conclusion is a claim, and it lives under the same architecture. The deepest danger in any derivation system is the one the audit of §7 exists to prevent, in derived form: highly rigorous source records flowing through a computation and emerging as a *fact* that no source ever stated. The rule that closes this hole is the same rule that governs authored claims, applied to their outputs: **every emitted derived assertion is itself a claim-bearing object, carrying provenance to the exact input claims and the exact transformation that produced it** — not merely a `derived: true` flag, but a trace that answers "why does the system believe this?" by walking claim → input claims → sources. Concretely:

```
derived_from:
  - claim: pos-2011-001        # the input records
  - claim: pos-2011-002
  - claim: rel-2011-014
derivation:
  rule: succession_overlap_v2   # named, versioned transformation
  version: abc123               # git commit of the build that produced it
```

It matters to state precisely what exists today and what is the required architecture, so the paper is not read as claiming more than the implementation does. **Currently implemented:** derived values carry flags (`trajectoryDerived`, `datesInferred`); timelines carry refs back to their source records; `interval-trims.json` publishes the arithmetic the interval resolver performed; succession thresholds are emitted as named constants (V24). **Required architecture, not yet fully implemented:** the uniform `derived_from`/`derivation` contract applied to *every* derived assertion, with versioned rules and recursive provenance traversal — a reader walking any derived value back to its input claims and their sources. The first is the foundation; the second is the design commitment stated here and scoped as future-work item 4 (§10.2). Until then, the safeguard that already holds is the simpler one: a derived value is never *rendered* as a fact — the build labels it computed, and no derived output is allowed to feed a validator as though it were a source (P5).

5.6 The influence model: anchored or rejected

The dataset contains only claims, and the influence layer is where that discipline is most visible. An influence edge says "a documented or reported relationship of this type connects these two nodes, with a source saying so" — never "this person secretly runs X". Three rules enforce the boundary:

1. **Influence-family edges must be attached.** An `influence` or `advisory` edge whose endpoints have neither a position nor a documented edge of a formal kind (appointment, ownership, board, prosecution) is *unmoored* and fails the build (V9). A `reported-influence` edge — which asserts only that sources report the chain — is kept and flagged, because rule 6 requires the popular account to remain visible even when its anchors are not yet in the graph. The distinction is the whole argument: **the map can show a rumour as a rumour; it cannot show a rumour as a fact.**
2. **Influence strength is an editorial judgment that must state its reasoning.** A `strength` value without `reasoning` fails the build, under the same contract as the `authority` weights on roles.
3. **The influence index is basis-discounted.** A composite influence score sums influence edges weighted by authored strength and discounted by the evidence basis of each edge (documented 1.0, reported 0.55, inferred 0.2, unsubstantiated 0). The weights follow the basis ladder's own semantics, stated so the numbers are not arbitrary: a documented edge is a primary record and counts in full; a reported edge rests on one or more credible publications but not a primary record, so it carries roughly half the standing; an inferred edge is a reasoned estimate, carrying a fifth; and an unsubstantiated edge is worth zero because a score built from evidence that cannot be checked would contradict the index's own definition — "evidenced routes" requires evidence. The exact intermediate values (0.55, 0.2) are a stated editorial mapping awaiting the sensitivity analysis of §10.2 item 1, which will show how far the ranking moves when they do. An influence index computed from documented evidence is the only kind honest to publish — and the published score carries the label: *"reads: how many evidenced routes. Not a judgement of who truly runs the country."*

5.7 No single power score

Reflecting the same philosophy, the product computes *six* structural indices and shows them side by side with reader-adjustable weights — and deliberately refuses to emit a default composite:

Index	Definition	Derived from
Formal authority	highest `authority` weight of any post held	published, openly-editable weight table (an editorial judgment, stated)
Proximity	inverse hop distance to the presidency	BFS over documented edges only
Survival	years in senior post + bonus per regime rupture survived	position intervals — the most defensible index
Brokerage	normalised betweenness centrality [22]	graphology metrics
Reach	share of analytical layers touched	layer membership, denominator = enabled layers

Index	Definition	Derived from
Evidenced influence	weighted, basis-discounted influence edges	influence-family edges (§5.6)

Table 6 — the six indices. The composite is computed *by the reader* with their own weights; the design's claim is that "most powerful" is a function of what you decide to measure.

This is a deliberate response to a well-documented failure mode of quantitative social science [14]: many analysts, one dataset, divergent conclusions [13]. A single "power score" would be read as truth it cannot support. Six transparent indices, each labelled with what it actually computes, turn the ranking into an *instrument for the reader* rather than a verdict about the country. The dataset's own hypotheses page performs the same service at the level of the research questions: six hypotheses, each with a stated support verdict and a falsifier, two of them currently coming out badly for the project's own preferred narrative.

5.8 Machines propose; humans dispose

Automated research agents (procurement, registry reconciliation, contradiction detection, citation audit) exist and are active — but no agent writes canonical data. Agents write candidate records to a reviewable outbox (`data/contrib/`), and every path into the canonical dataset runs the same byte-exact emission and build gate as any human edit. A proposed change travels a state machine (pending → under-review → accepted/rejected → applied) that requires a named source, a human review, and a passing build. Merges are atomic across all referencing records and leave the merged duplicate in place, marked `merged_into`. Git is the audit log: every factual change to the dataset is a dated, attributable diff, surfaced on the site's corrections page. The gate does not ask *who* is proposing; it asks *what the source says* — which is why unlimited sybils cannot get past it (§9).

Candidate and canonical are separated, which is also the answer to the availability objection. A one-violation-fails-the-build semantics (§5.10) invites an operational objection: a single malformed record could stop the whole graph from publishing — a denial-of-service vector. The separation of *candidate* from *canonical* is the architectural answer, and it is already in place: an invalid candidate is quarantined in the outbox with the rejecting validator named, never merged; it does not touch the canonical dataset, so it cannot stop anything else from compiling. The all-or-nothing gate applies to the *canonical artifact* — and only to it. This is the correct boundary for a published record of record: quarantine is for contributions, compilation is for the dataset, and the two never share a failure mode. For the generalised framework (§9), the same separation scales: a fork receives candidates, quarantines the invalid ones, and compiles the valid remainder — epistemic guarantees intact, availability preserved.

5.9 The Agora: discussion that can only change the graph through the compiler

This section describes the second half of the paper's architecture, and its relationship to the thesis must be stated before anything else: **the graph creates the discussion; the discussion proposes changes; only the compiler changes the graph.** The system is not a compiler with a discussion forum attached; it is a **compiler plus a human verification loop**, and the two perform epistemic functions that neither can perform for the other. The compiler answers the formal question — *is this claim admissible under the rules it declares?* — checking that a source exists, that the basis is consistent with the declared confidence, that an inferred claim carries a falsifier, that dates exist, that influence edges are anchored; it does so mechanically and uniformly, on every record, at every build. It cannot answer the substantive question — *does the evidence actually warrant the status this record declares?* — because that is a judgment about the world: whether a document supports a claim, whether a technically-B-grade claim deserves B, whether three apparently independent newspapers all copied the same wire report. That second question is the community's, and it is not an add-on: it is the reason the layer exists. The community layer is therefore the second half of the division of labour of §9 — the half that exercises epistemic judgment — but it is deliberately built so that its judgment can never reach the dataset except through the same build gate that guards every other edit. Everything below is subordinate to that sentence: the identity machinery, the budgets, the pauses and the petitions exist to make discussion *deliberate and answerable*, and to make its influence on the dataset *auditable and non-unilateral* — never to make the community a second, ungoverned source of truth.

The name is taken from the Greek *agora* — the public square where citizens gathered to speak, to be heard, and to act — and the analogy is architectural rather than decorative. Its defining feature, for this paper's purposes, is that **speech in the agora was made deliberate**: the assembly ordered turns; the *graphe paranomon* — the Athenian prosecution for proposing an unlawful decree — made a citizen answerable for the proposal itself; and Thucydides has Pericles praise the Athenians for thinking before they act. These are used as architectural analogies, not historical claims. Four mechanisms in the Agora realise the pattern: speech costs something (a posting budget), it waits (a cooling pause), it is paced (a floor on how fast anyone may speak), and it is answerable (a persistent identity, an anchored record, and a proposal path that fails the build rather than the proposer). The ordering, pricing, pausing and accountability of speech are the four ways the Agora makes everything said a meditated thought.

Threads hang off records, not off the air. The root object of the community layer is not the thread; it is the record. Every thread carries a `target_type` and `target_id` — a person, an institution, a role, a position, a relationship, an event, a source — so a discussion cannot float free of the thing it is about. A conversation about a minister is anchored to that minister's record; a debate about an ownership claim is anchored to that relationship; a dispute about a date is anchored to that position, where its interval and its `disputes` block sit within reach. The composer's `@`-mention and the "attach this thread to a record" picker search the *same* index as the site-wide (cmd)K palette, so a name ranks identically wherever it is typed. The consequence is that discussion is never abstract: to talk about a claim in the Agora is to talk about it where its sources, its basis and its disputes already live. This is the graph-object-as-root model the prior-art study concluded no existing platform offers [15] — and it is why the community layer cannot silently become a parallel truth-bearing surface: a thread is *about* a record; it is never a record.

Identity: the server holds nothing worth taking — and the name is a claim, never a fact. People posting here write about named Tunisian officials under a decree-law that the project's internal legal brief [27] — written to be handed to Tunisian counsel, not a statement of established law — analyses as carrying severe criminal exposure, escalated for public officials. Identity is therefore designed so a compromise of the server compromises nobody: Ed25519 keypairs generated in the browser, the private key never leaving it, and the server storing only the public key, creation time, trust level and counters — **no columns for IP address, user agent, device fingerprint or last-seen**; rate limiting runs on a rotating salted hash that expires and cannot be correlated across days. A leak exposes public keys, which verify signatures but cannot produce them. Handles are derived from the key (`anon-` + 96 bits of its digest), so they cannot be claimed or collided. Recovery uses a generated twelve-word BIP-39 phrase (128 bits, never human-chosen — a chosen passphrase's ~20–30 bits of entropy falls in hours against the offline attack that public keys invite) derived offline in the browser, English-only because "every additional wordlist is another chance for a normalisation difference to silently derive a DIFFERENT key" — the identity-layer form of a date that silently changes — and opt-in because a recovered identity is a longer-lived one that leaks more through its own corpus. The single most important rule is the one the project's code states directly: **a display name never replaces the derived handle, and both are shown, always** — the handle is a fact, the name is a self-assertion, exactly the paper's axiom applied to people. Names cannot borrow the project's authority (the reserved-word list refuses "DeepTunisia Moderator" in Arabic, French and English), are normalised against invisible and bidi-reordering characters, and may be cleared at no cost: "a person must be able to stop being anybody." Standing is earned, shown, never asserted: trust levels gate what an identity may do (a new identity posts slowly, cannot open threads or propose changes; promotion requires demonstrated behaviour, never mere existence; ban evasion is made *worthless* — a returning banned user starts at level 0 — and level 3 moderation is never reached automatically, because "an automatic path to moderation privileges is a path a patient adversary walks"). A self-description — "journalist", "worked in the ministry until 2014" — is stored as a *claim* that is never checked, because a source's standing changes how their account should be weighed, and the alternative is people asserting it in prose where it carries no marking at all.

The budget: a citation costs half. Standing gates what an identity may do, but it does not govern how loudly — and the volume mechanism is where the project's epistemology is literally priced in. Every contribution draws from a rolling-weekly budget measured in units; a bare assertion costs two, **a contribution carrying a citation costs one**. The design comment states the reasoning without hedging: "the budget falls on assertion, not on evidence: a contribution carrying a citation costs half. An uncited assertion is a bill sent to the reader; a cited one comes with its own receipt. That is the project's epistemology, in the rate limiter." A thread costs roughly double a reply — it claims everyone's attention, not just the room's — and a post shorter than 280 characters needs a citation to be worth that attention at all. The window is a rolling week, deliberately not a calendar one, so there is no midnight rush of unconsidered posting; and standing compresses volume rather than multiplying it (the top of the ladder is five times the bottom, not twelve),

because "standing should unlock WHAT you may do, not how loudly you may do it." The budget is derived from the content tables, never from a counter — a `posts_this_week` column with timestamps "would be a last-seen field wearing a hat," and the anonymity audit forbids one. The prices are "configuration, not architecture," as the design says — but the *shape* of the pricing is the architecture: assertion is charged, evidence is discounted, attention is budgeted.

Posts are frozen claims, and downvotes move position, never existence. Two details complete the epistemic discipline of the community layer. First, a post records its author's name and self-description *as they stood when it was written* — denormalised on purpose, so that changing your description to "lawyer" does not silently re-label every post you ever wrote as a lawyer's, retroactively re-weighting arguments people have already read. The schema comment is the identity-layer version of this paper's central rule: that is "the identity-layer version of letting an inference become a fact," and it is the same reason positions are intervals rather than a current job title. Second, ranking is designed so that a coordinated group cannot weaponise it: downvotes affect a post's *position* in a list, never its existence — the effective score is floored so a brigade can push something off the first page but cannot bury a sourced investigation beyond reach — and there is no threshold at which a post disappears. Removal is a moderator action with a stated reason, reversible, and logged in an append-only moderation audit that existed before there was anything to log. Reports weight *distinct identities*, so one person filing fourteen reports is one opinion, not fourteen.

The cooling pause and the pace floor: meditation enforced as mechanics. Two more mechanisms make speech deliberate in the physical sense of the agora — and neither is moderation, which is what makes them important. First, a post waits **ten minutes between submission and publication**. This is not a review; nothing is inspected in it. It is a pause, and the design states its purpose without adornment: "it separates the decision to write from the decision to publish. Most posts a person regrets are regretted within ten minutes." Visibility is *derived* from the post's `created_at`, so there is no scheduled job to fail and no `published_at` column that could disagree with the pause: the pause is a property of the clock, not a process that can be skipped or that can silently not run. Second, no one may speak faster than a person reads: a floor of four seconds between consecutive actions exists because "a human composing a sentence about a decree does not submit twice in the same second" — posting faster than that is the mechanical signature of automation, and the floor catches it without a CAPTCHA, without a fingerprint, and without punishing the person on an old phone through Tor, who is exactly the audience this site exists to serve. The ten-minute pause and the four-second floor are the Agora's answer to the reflex: the first makes sure the impulse has cooled before it becomes a published claim, the second makes sure the room is not spoken over. In the terms of the opening of this section: the pause is the *meditated*, and the floor is the *thought* — between them, nothing said in the Agora has to be a reflex.

The petition: discussion that becomes action — with the honesty rules fixed before the feature exists. The loop "graph creates discussion, discussion creates action" completes in the petition: a community that has debated an individual, an entity or a relationship should be able to do more than talk about it. A petition in the Agora is anchored to a graph object like a thread; it can be signed pseudonymously (with a key) or — accepting the correlation risk, and told so — with identity; and its target can be a person, an institution, a relationship, or the correction of a record itself. But the design posture, fixed now and before the feature ships, refuses the one thing that makes most petitions dangerous: a bare headcount. An unverifiable count of pseudonymous signatures is a number an adversary can set to anything, and the first credible accusation of inflation destroys not just the petition but the atlas attached to it [6]. The petition therefore publishes its *arguments* — a threaded, sourced case anyone can read and check — treats the count with the same risk-bucketed honesty as the graph's own statistics: named signatures are the strong ones, anonymous signatures are labelled exactly as what they are, and a count that cannot be backed is published as an absence rather than a number. This is the paper's rule 6 applied to collective action: the petition shows the case, the evidence, and what is and is not measurable — it does not pretend that unverified signatures constitute a mandate.

The community layer is not project data, and the boundary is CI-asserted. The Agora's database holds no graph data at all; `scripts/test-community.ts` asserts in CI that the graph build never reads the community schema and the community server never reads `data/`. The separation is the load-bearing one for the whole architecture: discussion is where people *think*; the canonical dataset is where claims *are*. The bridge between the two is the proposal path of §5.8 — discussion asks what people think, a proposal asks whether the record should be different, and only the second needs sources and review — and every accepted proposal enters the graph through the same emitter and build gate as any other edit, so the community's influence on the dataset is real, auditable, and never unilateral. The proposal is also the Agora's *graphe paranomon*, used here as a conceptual analogy: in Athens, a citizen who proposed an unlawful decree was answerable for the proposal itself — the proposer bore the risk, not the city. The architectural analogue is gentler and stricter at once: a proposal that would break the epistemic constitution is not prosecuted, it fails

the build — but it fails *named*, with the proposer's claim, the rejected operation and the reason attached, so proposing a change to the record is an act of accountability, not a suggestion. This is the loop the paper's division of labour describes (§9), made concrete.

The loop this section describes has a third stage that is easy to miss because it is the least glamorous: **recompilation.** The community's epistemic judgment becomes epistemically effective only when it re-enters the compiler as a changed record — a re-grading, a re-attribution, a dispute entry, a source split. There is no path by which a community conclusion changes what the graph shows without a record changing and the build re-running; that is what makes the community's influence *canonical* rather than conversational. The architecture is therefore three-layered, and the layers are not interchangeable: **L1, the epistemic compiler**, prevents structurally inadmissible knowledge from becoming canonical; **L2, the deliberative community**, determines whether admissible evidence deserves the epistemic status declared for it; and **L3, recompilation**, turns accepted human correction back into machine-enforced canonical state. A community without L3 is a discussion forum; with L3 it is half of a closed epistemic feedback loop — and the loop, not the compiler alone, is the epistemic system.

5.10 What the build gate formally guarantees

The preceding subsections describe mechanisms; this one states the property they jointly deliver, in the narrowest form that is true. Let D be the canonical dataset — the YAML files under version control — and let the build be the deterministic transformation of D into the published graph. Define four families of constraints:

- $E(D)$ — **evidence-envelope constraints**: every claim-bearing record carries the nine-field envelope; an inferred claim has both `reasoning` and `falsifiable_by`; a grade C/D claim has `attributed_to`; an unsubstantiated claim names where it circulates; every claim has `sources ≥ 1`; `basis` is derived and consistent with `confidence` and `verification` (V12, V18–V20, V23).
- $T(D)$ — **temporal constraints**: every date token is grammar-valid and calendar-valid; no resolved interval is internally contradictory or inverted; every envelope clamp is published (V21, V22, V15).
- $G(D)$ — **graph constraints**: every reference resolves to a graph entity; directed edges respect their direction semantics; influence-family edges are anchored to documented adjacency; the event causal graph is acyclic (V1–V10, V13–V16, V24).
- $R(D)$ — **referential constraints**: every cited source exists in the source register and is itself cited; no duplicate record slips through (V3, V6, V12, V16, V19).

Then the build gate is:

```
Build(D) = Emit(D)    if E(D) AND T(D) AND G(D) AND R(D)
Build(D) = Fail       otherwise - no dataset is published
```

The four families deliberately cover only the **fail-level** validators. Warn-level findings (V7 board overlap, V11 orphaned events, V17 unlinked ruptures, the warn branches of V4/V5/V6/V10/V13/V22/V24) do not gate emission and are therefore *not* part of the formal guarantee — they are reviewer-facing signals, and a dataset can emit with warnings outstanding. This is a chosen property, not an omission: the guarantee is exactly as wide as the set of validators that can stop the build, no wider.

The theorem is conditional, the paper says so, and the condition is now measured, not assumed. "If the build emits a dataset, no record violates any implemented invariant" is logically correct *provided the implementation actually computes E, T, G and R correctly* — and §7 demonstrates that the implementation previously did not. The guarantee is therefore not "the architecture mathematically guarantees this"; it is "the current implementation is intended to realise this contract, and the audit of §7 plus the regression suite of §8 provide evidence that it does." The validators are trusted code. The strength of that evidence is itself an empirical question, and §8.1 reports the mutation-testing campaign that measured it: the first pass introduced twelve deliberate breakages of the hardening validators and detected three — one killed outright by an existing test, two first observed to survive and then killed by tests the campaign itself added, including the reclassification of 138 positions that no pre-campaign test had caught. The extended campaign then widened the surface to every validator, closed the *latent* class — the weakened checks invisible on a clean graph — with synthetic invalid fixtures and a fixture-pipeline runner, and reached twenty-five of twenty-six breakages detected, with the single survivor documented as by-design redundancy. The condition is being tightened, not assumed away. This is the load-bearing admission of the formal section, and it is why future-work item 3 (§10.2) — a machine-checkable

specification of the validator layer — is not a nice-to-have but the natural next recursion of the paper's own argument: an epistemic contract whose enforcement mechanism is itself mechanically checked.

The property the gate delivers, stated as a theorem about the pipeline rather than about the world:

> **Guarantee.** If the build emits a dataset, no record in it violates any implemented invariant. Equivalently, the set of published datasets is exactly the set of datasets satisfying $E \wedge T \wedge G \wedge R$.

And its complement, stated with equal force:

> **Non-guarantee.** The build does not imply that any emitted claim is true. Every residual failure mode in §8.3 and §9 — a false source, a misunderstood source, a maliciously misgraded record, a validator bug, an ontology assumption, a true claim omitted — is a way for the published dataset to be false while the build passes. The gate constrains the *relationship between a record and its declared epistemic status*; it does not constrain the relationship between a record and the world.

Two corollaries follow. First, the guarantee is *checkable by construction*: it is the tautology of the pipeline, not an empirical claim — which is exactly why the regression tests of §7 matter, because a validator that does not run is not an invariant, and the audit proved that claim false once already. Second, the guarantee is what "cannot be quietly wrong" means in §1: quiet wrongness is precisely an $E \wedge T \wedge G \wedge R$ violation that reaches publication undetected. The formal statement converts the slogan from rhetoric into a property a reader can test against the code, and it is the target of §10.2 item 3, which proposes re-encoding these constraints in a machine-checkable form (a Datalog fragment or SMT encoding) so the enforcement itself becomes subject to the same standard of proof it applies to the data.

A third boundary follows from the same care, and it is the most important open one: **record validity is not graph validity.** The guarantee above is a guarantee about records — every emitted edge is locally valid, sourced, temporally coherent and correctly typed. It is not a guarantee about what the composed graph says to a reader: five individually valid edges A→B→C→D→E can present as an influence chain even when the edges carry different types, dates, confidence levels and contexts. The paper therefore distinguishes three levels of validity — *record validity* (Level 1), *relationship validity* (Level 2), and *interpretation validity* (Level 3) — and states plainly that the compiler is strong at Levels 1–2 and that Level 3 remains open: guarded only by the rendering rule of P5 (no inference renders as a fact), owned in §8.3 item 6, and scoped as future-work item 5 (§10.2). A locally honest graph that composes into a narratively misleading one is a failure mode the gate does not close, and the paper says so in the same breath as its guarantee.

6. Enforcement: the validator layer

All validation runs in one deterministic build step, aggregated through a single `errors[]` mechanism: **warn** populates the report; **fail** breaks the build — no dataset is emitted, and the process exits non-zero. The validators fall into five groups:

- **Structure** — schema parsing in strict mode (an undocumented key is a build failure naming file, record and key; V19), duplicate ids, malformed tokens.
- **Referential integrity** — every ref resolves to a graph entity, every source exists (V12), every company id is a company-like institution, agreements name real parties.
- **Epistemic** — inferred completeness (V18), C/D attribution (V20), unsubstantiated attribution, review provenance (V23), the gazette-overclaim guard (a review note may not claim a JORT verification the record's sources cannot support).
- **Arithmetic and graph** — ownership sums by sweep-line over interval windows (V1), duplicate company detection by script-aware name normalisation and Levenshtein similarity (V6), duplicate relationship detection on overlapping intervals (V16), directionality semantics (V14), equity bounds (V13).
- **Time and causality** — calendar-date validity (V21), interval contradiction and inversion guards with published trims (V22), event causal ordering and cycle detection via Tarjan's strongly-connected-components algorithm (V8), unmoored influence (V9), succession sanity (V24), geographic hierarchy (V10).

The full validator table (V1–V24) is reproduced in Appendix A. Two properties of the layer are worth stating as design commitments: **fail-vs-warn is chosen, not accidental** — hard failures are reserved for structural and epistemic breaches, while reviewer-facing findings (orphaned events, unlinked ruptures, borderline duplicates) warn; and **no val-**

idator is prose — every row of the table is implemented, and every row that closes an audit finding has a regression test that fails on the pre-fix build (§7).

7. The hardening audit: seven ways the promise was bypassable

The most important evidence in this paper is not the design — it is the audit that found the design was, on first inspection, *not true of the shipped build*. In August 2026 the project audited its own enforcement promises against the code and found seven ways each could be bypassed. This section reports all seven, because the credibility of a "compiled epistemology" rests on demonstrating that the compiler actually rejects what it claims to reject — and because each defect is the kind that a reader could previously exploit to launder a claim.

#	The promise	The shipped defect (verified with file:line evidence)	Closed by
H1	An inference never ships without reasoning and a falsifier	The Zod refine checked the *authored* basis only, which the data almost never sets; the build's re-check covered positions only. A real relationship record shipped with `basis: inferred` and no reasoning and no falsifier.	V18 — one shared re-check over the **derived** basis, applied to every claim kind
H2	A claim field written in the source file reaches the reader	`attributed_to` and `reasoning` written on people, positions and institutions were **silently stripped**: Zod record schemas run in strip mode, and the fields were not declared, so 15 values vanished with no error, no warning, no trace.	V19 — strict schemas everywhere; an undocumented key is a build failure
H3	Grade C/D claims name who is making them	The attribution requirement was enforced for relationships and agreements only; C-grade positions, people, events and institutions could ship un-attributed.	V20 — one shared refine on the envelope, every kind
H4	Dates that cannot exist never enter the dataset	`Date.UTC` rolls over invalid calendar dates: `2018-02-31` silently became `2018-03-03`, `2018-00-01` became `2017-12-01`. In a project whose product is precise dates, a wrong-but-plausible date shipped as a *different* date.	V21 — regex grammar at the boundary + calendar round-trip validation

#	The promise	The shipped defect (verified with file:line evidence)	Closed by
H5	Contradictory intervals are flagged, not silently trimmed	The envelope trim clamped only the end side; for a real record (`start: 2010`, `end: 2010-01-14`) it produced an **inverted interval** (`endEarliest > endLatest`) and a synthetic "certainly active at 31 December 2010" core.	V22 — start-side clamp, inversion guard, every clamp published in `interval-trims.json`
H6	Human review is real	`review.date` was an unvalidated string, `method` was free text, and the published "reviewed" statistic counted any record bearing a review object regardless of substance — 27 of 719 records at audit time, all positions.	V23 — enum methods and the statistic redefined; full calendar round-trip (not just ISO format, which admitted `2026-02-31`) pinned by the §8.1 campaign
H7	Successions are derived, not guessed	Succession was derived by nearest-dated adjacency with an unpublished 1-year gap threshold, and records with unknown end dates were silently exempted from overlap detection.	V24 — thresholds published as data; unknown-end exemptions surface as open items

Table 7 — the hardening contract. Each row closes with a validator and a regression test that fails on the 2026-08-07 dataset.

Three things about this audit are worth the reader's attention. First, the defects were not hypothetical: each is cited with the concrete record that shipped the failure, and each regression fixture is a real record in the pre-fix tree. Second, the audit *changed the statistics*: the review-coverage number published on the about page was recomputed under the stricter definition. Third, the audit is the difference between a methodology described and a methodology enforced: a project that found its own enforcement broken in seven places, published the finding, and closed each hole is making a different kind of claim than one that asserts the same promises in prose. The paper's title — *the epistemology is the compiler* — is only defensible because the compiler was adversarially tested and, on first testing, failed. It now passes, and the tests that prove it are part of the repository. The audit is self-performed — it was the authors who attacked their own implementation, checked for the defect classes the hardening contract names (silently stripped fields, rollover dates, inverted intervals, substance-free review records, unpublished assumptions), and could in principle have missed a class they never thought to look for. Independent verification is the named follow-up: mutation testing (§8.1) removes the author-blind-spot for the *validators*, and the community's external adversarial pressure, once the project is public (§9), removes it for the *dataset*.

8. Evaluation

8.1 What the tests verify

The verification strategy is invariant-based rather than sample-based. `npm run test` asserts over the *built graph* that: no `inferred` record lacks reasoning or a falsifier; no C/D record lacks attribution; no interval has an inverted core; `interval-trims.json` is emitted; every review carries a valid ISO date and enum method; every resolved interval is calendar-valid (fuzz pass); published succession gap/overlap counts are reproducible from the emitted constants; and the seven audit fixtures of §7 fail against the pre-fix dataset and pass against the current one. The suite also enforces representational invariants — a relationship id derived from array position is impossible (§5.8 note), schema-audit exports diff declared-vs-seen keys — and, at the UI level, a browser smoke suite exercises every route in three languages and both themes. The epistemic rules, in other words, are not sampled; they are exhaustively checked at every build **against the records that exist** — a scope that mattered because the first mutation campaign (next paragraph) reported a *latent* class: a weakened ingestion or schema guard is invisible to a suite that only reads the clean graph, until a record exists that would exercise it. That class is now closed by construction rather than by patience: the pure-surface suite (`test-validators.ts`) and the fixture-pipeline runner (`test-pipeline.ts`) make the bad records exist, so the weakened guard fails regardless of what the current graph contains. Within its original scope — the graph as built — a change that violates an invariant cannot reach publication, and the two synthetic layers extend that coverage to the guards themselves.

The scale is worth stating in numbers, because "the tests exist" is not the claim — "the tests cover the claim" is. The graph test suite (`scripts/test-data.ts`) currently carries **68 assertions** over the built graph, including the seven audit fixtures, the calendar fuzz pass, the pinned deriveBasis truth table, the interval-trims content-freshness check, and the rule-2 ratchet; the pure-surface validator suite (`scripts/test-validators.ts`) carries **59 assertions** over the schema and date machinery with synthetic inputs — negative fixtures the validators must reject and positive ones they must produce; the fixture-pipeline suite (`scripts/test-pipeline.ts`) carries **8 assertions** that run the real build against a throwaway copy of the data carrying one injected record per defect class and assert the build fails with each message; and the community-layer suite (`scripts/test-community.ts`) carries **69 assertions** covering the identity, ranking, moderation and isolation invariants — including the assertion that the graph build never reads the community schema and the community server never reads `data/`, read from the actual build and schema files rather than from documentation. The 24 validators of Appendix A break into fail-level and warn-level checks as §5.10 describes, and every validator that closes an audit finding has its own regression fixture. These are the numbers the repository reports, and they are reproducible from a clean checkout by running the test suite itself; they are stated here so that the evaluation can be compared against the claims rather than taken on faith.

Operational facts. The build takes about fifteen seconds and the full test suite under a minute on the reference development machine (measured during this revision: `npm run data` ≈ 15s, `npm run test` ≈ 38s across the **eleven** suites; the earlier "3s / 54s across nine suites" figures were taken on an earlier reference machine under an earlier measurement method and are superseded — the 38s reflects a faster machine absorbing the fixture-pipeline runner's two synthetic builds). On a failed build, **nothing new is published**: the deterministic pipeline emits no dataset when validation fails, so the running site serves the last successfully-built artifact — which is guaranteed by the same gate, because it is by construction a dataset that passed every invariant at the time it was built. This is the fail-closure semantics of §5.10 made operational: a bad build cannot replace a good artifact. The dataset's own statistics at the time of writing, from the build's published metadata: the 790 claim-bearing records break down by basis as documented 158, reported 608, inferred 16, unsubstantiated 8; 215 records are flagged `needs-primary-source`; 41 succession gaps and 2 overlaps are derived and published; 26 source contradictions are recorded as disputes; 27 of 790 records carry a substantive human review; and 29 interval clamps are published in `interval-trims.json`. Every figure in this paragraph is emitted by the build from the data — none is hand-typed into the paper.

Mutation testing of the enforcement layer — measured, not proposed. The §10.2 item 3 experiment has been run, and this is its honest result. The kill criterion is stated first, because it determines what the numbers mean: **a mutation is counted as killed if the test suite fails on the mutated build — where "the test suite" is the suite as it stands after the campaign, including the tests the campaign itself added.** This matters because two of the three first-pass kills were not detected by the pre-campaign suite: they were first *observed* to survive, and the campaign then closed the gap by adding a test, converting the survivor into a kill. That is the standard mutation-testing loop — a survivor is a test gap to be closed, not a finding to be ignored — but it would be misleading to report the first-pass "three killed"

without saying that the detecting assertions for two of them did not exist when the campaign began. The campaign is runnable end-to-end (`npm run mutation`): one deliberate breakage is applied to a validator, the build and every test layer are run against it, the working tree is restored byte-exact, and the outcome is classified into one of four classes. **Killed** — the build crashes or a test fails. **Silent drift** — every assertion stays green but the emitted graph changed; the campaign's most novel category, defined here once because it recurs: the class of mutation that rewrites published data with nothing objecting, which only a before/after comparison of the emitted graph can see. **Latent** — nothing observable changed, because the weakened guard is invisible on a clean graph. **By-design** — the mutated constant is configurable by contract, so the "breakage" is a reconfiguration, not a validator defect. What follows is the full history across three passes, because the intermediate numbers are the point: the campaign's value is visible in the progression, not in the final figure.

First pass — the hardening validators (V18–V24). Twelve mutations were introduced — removing a validator, dropping a refine, weakening the basis mapping, relaxing a regex, removing a guard, changing a threshold, disabling a publication — and the build-plus-test pipeline killed three and let nine through. The three kills are real and load-bearing. Weakening `deriveBasis` so grade-C/needs-primary-source claims map to `reported` instead of `inferred` silently reclassified **138 positions** with no pre-campaign assertion noticing; the campaign's new truth-table test now kills it — this is the paper's H1 failure mode, found by mutation rather than by audit. Removing the `interval-trims.json` publication first survived because a stale file masked it (the test checked existence rather than freshness); the campaign's new content-freshness check now kills it. Removing the emitted succession constants was killed outright by the existing V24 reproducibility test, which crashed on the missing values. The nine survivors were not dismissed — they were *classified*. Eight were **latent**: the test suite validates the current clean graph, so a weakened ingestion- or schema-level guard is invisible unless bad data exists to expose it (a future record lacking a falsifier would be caught; the removal of the guard that would have rejected it is not, at the time). One survivor (changing the published succession threshold) was **by design**: the V24 contract pins *reproducibility* — emitted constant and recomputed count must agree — not the constant's value, which is explicitly configurable (§5.5). The honest reading of a 25% kill-rate was not "the tests are weak" but "the tests validate the graph, and the current graph is clean — the ingestion-layer guards are only exercised when bad data exists to exercise them"; and the two permanent tests the pass earned — the **deriveBasis truth table** pinning all 15 input combinations of the published mapping (so the 138-record reclassification can never recur silently) and the **interval-trims content-freshness check** (so a stale file can never mask a disabled publication) — are the first two closes of that class.

Second pass — the latent class, closed with synthetic fixtures. The response to the eight latent survivors was not to accept them but to make the bad data exist: `scripts/test-validators.ts` tests the pure schema and date surface with synthetic inputs — negative fixtures the validators must reject (a grade-C record with no claimant, an inferred claim with no falsifier, a review dated 2026-02-31, an interval that ends before it begins) and positive ones they must produce (the `~2017` widening window, the floor clamp of a `<=1956` bound, the cutoff clamp of a `verified:2026-07` bound). Against the pre-fixture suite, a 23-mutation pass killed **7 of 23 (30%)** and — more alarmingly — surfaced two **silent-drift** survivors, the class the graph tests cannot see by construction: a mutation removing the ~ widening silently tightened the published intervals of 21 positions, and a mutation breaking open-ended duration measurement silently rewrote every open-ended `years` field, both with every assertion green. The campaign closed both with positive fixtures and reached **20 of 23 (87%)** — thirteen schema-layer kills attributable to the new fixtures. The pass also found, while investigating the review-date mutation, that V23's "calendar-valid" was aspirational: the review date was format-checked only, so `2026-02-31` passed the validator and the graph test alike; the schema now applies the same calendar round-trip the date parser uses (V21), and the fixture pins it. And a mutation of the sources minimum exposed that the older claim kinds could ship zero-source records — 19 institutions (the presidency, the armed-forces branches, the parties, the judiciary bodies), 3 relationships, 1 person and 2 eras rendered `documented` citing nothing; a rule-2 ratchet now pins those counts so they can only fall. The V23 hardening and the ratchet are permanent closes this pass earned, and they are the paper's own rules being enforced on the paper's own data.

Third pass — the pipeline layer, closed with a fixture runner. The remaining latent survivors were the build's own invariants — the V8 causal-graph checks (cycle detection and temporal ordering), V9 edge anchoring, and the referential-integrity guards — which live in `build-data.ts`, a module that runs at import time against hardcoded directories and is unreachable from any pure-surface test. `scripts/test-pipeline.ts` makes the bad data exist at the pipeline level: it copies `data/` to a throwaway tree, injects one record per defect class — a two-event

causal cycle, a cause that begins after its consequence ends, a position citing a nonexistent source id, a relationship from a ghost entity, an influence edge between two institutions nothing anchors — and asserts the real build fails with each message. A clean-control build proves the fixture tree is byte-identical to the real graph, so the injected run is the only difference. The five pipeline mutations — the two carried over from the second pass plus three new ones targeting the temporal-ordering, anchoring and ghost-endpoint checks — are now all killed, and the final figure is **25 of 26 mutations killed (96%)**. The derivation of the denominator must be stated, because a paper whose thesis is that numbers are claims cannot leave its own headline number underivable: **the passes are not summed.** The first pass was a hand-run probe of V18–V24 whose twelve mutations predate the harness — its three kills survive only as permanent closes now enforced in the suite (the deriveBasis truth table, the interval-trims freshness check, the V24 reproducibility test), so they are not re-counted — and the harness catalogue of passes two and three holds the 26 distinct mutations (m01–m26) the figure counts. Of those, 25 are killed, none survive as silent drift, and one survives by design (m09, the review-date format regex), documented as by-design redundancy behind the calendar round-trip that rejects every non-calendar date anyway; threshold and weight mutations never enter the catalogue, because V24's thresholds and §5.4's weights are configurable by contract — mutating a stated constant is reconfiguration, not validator breakage — and it is those chosen invariants, not the implemented ones, that §10.2 items 1 and 3 target with the sensitivity analysis and the formal specification.

8.2 The project measures itself

A dataset whose entire argument is honesty must publish its own shortfalls, and the build is wired so that it cannot avoid doing so:

- **Published statistics are machine-written.** Prose in the project's own documents carries `<!--stat:-->` tags that the build rewrites from the graph at every build (§6). Hand-maintained counts drifted historically — the README advertised 14 succession gaps while the graph held 36 — so the build now owns the numbers. A hand-edit of a published stat fails the test suite.
- **Review coverage is risk-bucketed.** A single "reviewed" percentage is refused, because verifying a gazette-dated appointment is not interchangeable with verifying an unsubstantiated allegation about a named living person. Coverage is reported per risk bucket (unsubstantiated → attributed → inferred → reported → documented), and the buckets are ordered most-damaging first.
- **Translation coverage is per-tier, never summed.** A translation produced by a model and checked by a model is a different act from one a human has read, and the project refuses to file them under the same number — `human` is the only tier that means a person who reads the language has looked at it.
- **Absence is measured.** The build computes what the map does *not* contain: a coverage audit of presidential networks (of twelve family relationships in the dataset, eight are Ben Ali's; the incumbent president has none — an asymmetry the audit refuses to let pass unremarked, because "insufficient evidence for H1" must not be allowed to mean "nobody looked"), and a card-completeness worklist ranking senior figures by how little is on file.

8.3 What is not yet evaluated — stated plainly

Consistent with the project's own rules, this section names the evaluations that have *not* been performed:

1. **No inter-annotator agreement study.** The confidence/basis grading is applied by the project's editors under a documented rubric; the consistency of that grading has not been measured. This is the single gap that separates the paper's claim — "grades are applied under a documented rubric" — from a measurement — "grades are reproducible across independent raters." The study that closes it is scoped in detail in §10.1, with its design, sample, rater requirements and published deliverables, precisely so that it can be funded and executed as a defined piece of work rather than an aspiration.
2. **No external validation of the indices.** The six indices are computed from the graph by stated definitions; none has been validated against an independent measure of influence (there is no consensus external measure, which is precisely the point of §5.7 — but the absence of validation is a limitation, not an advantage).
3. **No sensitivity analysis of composite weights.** The composite is reader-weighted by design, so the project publishes no default; a systematic sensitivity analysis of how rankings change under weight perturbations is future work.

4. **The gate does not authenticate documents.** A forged gazette page would enter as a documented, A-grade claim; the build cannot and does not claim to catch forgery. The boundary is addressed in §9.
5. **No evidence-independence model: the count of sources is not the independence of evidence.** The confidence rubric reads "several credible secondaries" as grade B, and the build enforces `sources.length ≥ 1` on every claim-envelope kind — the legacy kinds that predate the envelope are pinned, not exempt, by the rule-2 ratchet (§8.1) — but neither mechanism distinguishes *independent* sources from *downstream* ones. Source A → source B → source C, each citing the previous, is one underlying evidentiary lineage wearing three URLs; a naive count can make it look stronger than a single primary document — and a B grade can rest on three newspapers that copied the same wire report. Detecting this is not a compiler problem — it is a *provenance* problem, which the community can surface (two sources citing the same original report are not two corroborations) and an independence model can represent (future-work item 4, §10.2). Until then, "sources ≥ 1" and "several secondaries" are statements about *records*, not about *independent evidence* — an honest limitation, stated.
6. **Claim validity is not graph-interpretation validity.** The compiler guarantees every edge is locally valid and sourced; it does not guarantee that the *graph they compose* is narratively honest to a reader. Five individually-sourced edges A→B→C→D→E can present as a meaningful network even when the edges carry different types, dates, confidence levels and contexts — a chain of weak connections reads as a path of influence. Guarding against that requires interpretation-level validators (temporal coherence of paths, relationship-type compatibility along chains, confidence-propagation floors, minimum-evidence thresholds before a *derived network* claim is rendered) rather than record-level ones — future-work item 5, §10.2. The paper's stance is that this is a real boundary: the compiler can keep the map from *lying*; teaching it not to *mislead* is the harder, next problem.
7. **Declaration gameability: the gate is only as strong as the declarations.** The obligations the compiler enforces — a falsifier for `inferred`, attribution for C/D — create an *incentive to declare away the obligation*: a claim declared `confidence: B` / `verification: verified` derives to `reported`, which requires neither. Axiom A1 says the compiler does not judge the epistemic judgment itself, and this is exactly where that boundary bites: a strategically-declared grade sails through, because the compiler cannot see the strategy. The mitigation is the architecture's own division of labour (§9): re-grading is a community action that travels the proposal path and must compile; the risk-bucketed review queue surfaces C/D and unsubstantiated records for human scrutiny; and the inter-annotator study of §10.1 measures — rather than assumes — how honestly the rubric is applied. The gameability path is real, it is not closed by the gate, and the paper says so rather than implying the declarations are trustworthy because the build passed. This item is not one limitation among eight: it is the central boundary of the architecture — the exact place where Axiom A1 divides what the machine can establish from what only humans can. It is also why the system is two-layer by construction rather than by enhancement: the compiler's guarantee is conditional on the declarations, the declarations are the community's jurisdiction, and a validator that tried to judge the declarations themselves would violate A1 by deciding the epistemic question it was built to leave open. The gameability path is therefore not a defect a stronger validator can fix; it is the specification of why Layer 2 exists.
8. **Single-country, mid-scale dataset.** Generalisability claims about the architecture rest on the schema being country-agnostic, not on evidence of deployment elsewhere; a comparative atlas is roadmap, not result.

9. Discussion

Verifiability, not truth. The project's doctrine, arrived at independently and shared with Wikipedia's policy [24] and the semantic-web trust vision [1,18], is that a dataset should make its claims *checkable*, not assert their truth. The build gate is that doctrine operationalised: every claim carries the means of its own checking (sources, archive URLs, reasoning, falsifier, disputes), and the claim's *status* is visible and non-upgradable. This is a different and weaker promise than "verified true" — and deliberately so, because the stronger promise is undeliverable by any system and every system that implies it is lying.

What the gate can and cannot do. The gate makes *status* structural: laziness, sloppiness, inattention and pressure cannot upgrade a claim's grade, because the machine refuses to compile the upgrade. The gate does not make *documents* authentic: a determined human can fabricate a decree and cite it, and no confidence system can detect that at the schema level. The project's defence against forgery is therefore not the gate alone but the gate *plus* the provenance practices that surround it: archive snapshots of every source URL (Tunisian media and gazette links rot fast), a review workflow whose notes must describe what was actually checked, and a legal-operational stance that treats the forged document

as the single highest-consequence failure mode [6]. The paper's claim is that the dataset cannot be *quietly* wrong — not that it cannot be *deliberately* wrong. Those are different properties, and the first is attainable while the second is not; a system that conflates them is overclaiming.

The compiler, the community and the audit trail: the division of labour. The boundaries above could read as a list of the gate's failures. They are not — they are the specification of the architecture's second half. DeepTunisia is not a compiler operating in isolation; it is a **compiler plus a community plus version-controlled data**, and each mechanism covers the ground the others cannot:

Problem	Best mechanism	Why
Missing required evidence metadata	Compiler	a record-level fact, checkable by schema
Impossible dates	Compiler	calendar arithmetic, checkable exactly
Unsupported structural relationships	Compiler	graph topology, checkable exactly
Silent epistemic upgrades	Compiler	the axiom of §1, checkable by construction
Bad source interpretation	Community	a judgment about the world, not the record
Wrong confidence grade	Community	a judgment about evidence, not the record
Missing source	Community	knowledge of where to look, not a rule
Source laundering	Community + provenance model	only humans (or a lineage model) know two URLs are one report
Political disagreement	Community + `disputes`	the dataset records the disagreement, it does not settle it
Validator bug	Developer audit + tests	the §7 method, recursively
Bad ontology	Community + maintainers	schema design is itself a contested claim
Malicious submission	Community + review + build gate	the gate makes the last mile safe, the review makes the first

The compiler enforces the *epistemic constitution* — the obligations every claim carries regardless of subject. The community exercises *epistemic judgment* — what the evidence means, which grade a claim deserves, whether two sources are one lineage. Neither can replace the other: the compiler cannot decide "this should be `reported`, not `documented`", and the community cannot be trusted to remember the falsifier rule on every record, forever, under pressure. What makes the combination sound is that the community's corrections *themselves* pass through the compiler — a re-grading, a re-attribution, a dispute entry, a source split, all travel the proposal path and all must compile. The loop is the architecture: claims → structured dataset → compiler → published graph → community scrutiny → corrections → recompilation, with git recording every step. The community half of this loop is not an abstraction: it is the Agora (§5.9), whose threads hang off the records, whose identity design holds nothing worth taking, whose votes rank but never grade, and whose petitions convert discussion into action — always under the same honesty rules as the graph itself. This is why Axiom A1 is the load-bearing wall: the system does not automate epistemology, it automates the enforcement of an epistemic constitution while leaving epistemic judgment contestable by humans — and the same division of labour is what makes the forkability of §9 safe, because a fork's community, not the framework, is its epistemic authority.

The three-layer structure makes the architecture's central empirical question precise. The machine-verification layer (L1) is experimentally validated: the seven bypasses of §7 and the mutation campaign of §8.1 measure, rather than assume, how far the implementation actually enforces its contract. The substantive human-evidentiary layer (L2) is architecturally specified — its role, its mechanisms and its place in the loop are defined in §5.9 — but it is not yet *empirically* validated: nothing yet measures whether humans using these mechanisms produce reliable, reproducible evidentiary judgments. That is not a gap in the architecture; it is the experiment §10.1 defines, and either outcome

is scientifically valuable. If three independent raters agree with the project's grades and with each other, the human layer has its evidence — the rubric is applied reliably, and the loop can be trusted at the volume the dataset demands. If they do not agree, the result is equally a result: it names the ceiling of what a two-axis rubric can enforce, and the disagreement taxonomy tells the field exactly where evidence grading is underspecified. The architecture is not embarrassed by the second outcome — it is designed to publish it, under the same source-backed discipline as the data.

The threat model, made explicit. The division of labour above is best understood as the answer to a stated adversary model — five classes of failure the architecture must withstand, and the mechanism responsible for each:

Threat	Gate	Community	Audit	Not solved
Careless editor — forgets required metadata	Y			
Malicious editor — deliberately misattributes (omits the required claimant)	Y	Y		
Malicious editor — deliberately misgrades (declares a weak claim strong)		Y		
Malicious contributor — injects invalid records	Y	Y		
Adversarial community participant — sybils, brigading, spam		Y		
Bad interpretation — misreads a source, misgrades evidence		Y		
Forged source — a fabricated document cited as evidence		Y	Y	Y
Validator bug — the enforcement itself is defective			Y	
Ontology error — the schema encodes a wrong assumption		Y	Y	Y
Source laundering — one lineage wearing many URLs		Y		Y

The table is the specification of what the architecture claims and what it does not. Rows where "Not solved" is checked are the honest boundaries the paper owns in §8.3 — the gate cannot authenticate a document, the community cannot see through a colluding pair of editor and source, and no mechanism catches an ontology that is internally consistent and wrong. A row with a check in "Gate" means the compiler makes the failure structurally impossible; a row with checks only in "Community" and "Audit" means the system can surface and correct the failure but cannot prevent it. The two malicious-editor rows are split deliberately, and the split is the point: *misattribution* — omitting the claimant

the contract requires — is a structural failure the gate rejects (a C/D claim without `attributed_to` fails the build), while *misgrading* — declaring a weak claim strong, so that no obligation attaches — is a judgment about the world that Axiom A1 puts beyond the compiler, caught only by community re-grading (§8.3 item 7). One more actor belongs in the model, and it is the most realistic one for a project like this: **the operator and maintainers themselves, under state or legal pressure.** A government that wants this dataset changed need not forge a document or corrupt an editor — it can pressure the people who run the site. The architecture's answer is structural rather than personal, and it is the same answer the portability section gives for jurisdictions: the canonical data is plain text under version control, the site is a window onto it, the exports are downloadable by anyone, the repository is forkable, and there is no central authority whose replacement would change the record. The data outlives the operator — and the operator's vulnerability is not a row the architecture can close, which is why it is stated as the boundary it is rather than hidden. The threat model is why the division of labour is not a philosophical preference: it is the minimal partition of responsibility that leaves no row unowned.

Keeping what cannot be proven. Rule 6 of the project — unsubstantiated claims are recorded, never deleted — is the most philosophically distinctive design decision, and the one most easily mistaken for recklessness. The defence is that a map containing only what it can prove cannot show *why the popular account is wrong*. The dataset carries a D-grade record of a circulating claim that the sitting president holds a doctorate, sitting beside the B-grade record, with its cited refutation, that the evidence says otherwise [4]; readers can see the popular account *and* the evidence against it. This is the same instinct as Wikibase's deprecated rank [2,3], applied to the claims themselves rather than to statements about the world.

The blast-radius argument. The project's community layer — the Agora (§5.9) — is where bots and sybils are a realistic threat, and it demonstrates the same philosophy applied to adversarial identity. Community content is never project data; votes rank and never grade; and a graph change requires a named source, a human reviewer and a passing CI build. The blast radius of a perfect bot attack on the community layer is a worse-ordered discussion page: nothing a thousand sybils can do changes a single record [6]. The gate asks what the source says, not who proposes — which is why identity verification, however sophisticated, is architecturally secondary here.

Portability: the framework versus the instance. The strongest statement of what this paper is about is the one made in the introduction: **DeepTunisia is a jurisdictional implementation of an open, source-backed framework for evidence-graded historical knowledge graphs.** The distinction between *framework* and *instance* is not rhetorical — it is architectural, and it is what makes the contribution portable rather than merely interesting about one country. Three layers must be kept apart:

Layer	What it contains	Portable?
1. The epistemic engine	the evidence envelope, the four-field temporal model, the validators V1–V24, the build gate, the derived-computation machinery, the dispute/provenance handling, the regression-test discipline, and the Agora community layer (§5.9) — threads-on-records, the identity scheme, the trust ladder, the vote/rank/report machinery, the proposal path	**yes — reusable unchanged**
2. The jurisdictional ontology	roles, institution types, administrative divisions, historical periods, local terminology, source-type taxonomies, jurisdiction-specific rules	**re-authored per deployment**
3. The canonical dataset	people, positions, relationships, events, sources, claims, dates	**entirely jurisdiction-specific**

Nothing in Layer 1 names Tunisia: the schema is written over a `role`/`holder`/`interval` join, not over the Tunisian Ministry of the Interior; the validators reason about evidence status, not about Tunisian offices; the interval grammar is about historical dates, not Tunisian dates. This separation is a design decision, not an accident — the ontology and the data were kept out of the enforcement layer so that the enforcement layer could be reused. A hypothetical *DeepMorocco* deployment would fork the repository, replace Layer 3 with Moroccan sources and records, re-author Layer 2 with Moroccan roles, institutions and governorates, and retain Layer 1 — the evidence envelope, the validators, the build gate, the temporal machinery — unchanged. That "unchanged" is asserted, not yet demonstrated: the Layer-1-retains-unchanged claim is untested until a real second deployment runs the build against non-Tunisian data, which is exactly what the comparative-atlas item (§10.2) scopes as the first test. The resulting deployment would be **independently operated and independently responsible for its evidentiary judgments**: the framework provides the standard, not the verdicts. This is a deliberate stance, and it matters politically as much as technically. DeepTunisia is not a central authority claiming to hold "the official truth about Morocco"; it is infrastructure that lets any community enforce the same epistemic contract on its own evidence, and no deployment inherits another's judgments by forking its code.

The research question this raises is larger than DeepTunisia: **can an epistemic infrastructure be standardised while the knowledge it governs remains locally controlled?** If ten independent groups fork the framework — a DeepMorocco, a DeepAlgeria, a DeepLebanon, a DeepCalifornia, a DeepLagos — they would all compile against the same contract while each remains sovereign over its own ontology and data. The standard lives in the compiler, not in a central editorial authority — which is exactly the position this paper has argued throughout, generalised from a single jurisdiction to the framework itself. The comparative-atlas roadmap item (§10.2) is the concrete first test of that claim.

Ethics and legal posture. The dataset names living people, including office-holders. Three properties of the design are the primary harm-reduction mechanisms, and they are the same properties this paper argues for on epistemic grounds: unsubstantiated claims are recorded but rendered distinctly and off by default, no inference renders as a fact, and every claim carries the means of its own checking. The project's retention-with-labelling stance deliberately diverges from Wikipedia's biographies-of-living-persons policy; the reasons are documented [6] together with a structured legal analysis of the exposure the dataset creates [27]. Two caveats are stated here because the review standard demands them: **the legal analysis [27] is an internal brief, written to be handed to Tunisian counsel — it is not public, not legal advice, and not a statement of established law — and the specific figures it discusses require verification against primary statutory text and external legal review before any of them is relied upon in publication.** A legal-verification appendix will accompany the paper at submission. "Never deleted" is not "anything anyone says gets immortalised": the retention of a D-grade claim is governed by explicit criteria — the claim must be *historically significant* (it circulated and shaped events or reputation), *materially relevant* to the graph, *documented* (the circulation is attested even if the content is not), and *assessed for harm* (a living person's claim carries a higher bar and stricter visibility than a historical figure's). D-grade claims render off by default, are individually reviewable, and the removal path exists. The evidentiary discipline is not incidental to the ethics — it *is* the ethics, in the sense that a system which cannot distinguish a rumour from a decree is the harm, and the build gate is the mechanism that refuses to produce it.

10. Future work

The roadmap [16] is explicit about what comes next, and this section adds the items the paper's own argument requires. Item 1 is scoped in enough detail to be executed and funded as a defined study; the rest are stated at the level of intent.

10.1 The inter-annotator agreement study — scoped

Why this study, and why it costs what it costs. The build gate proves that the grading rules are *enforced*; it cannot prove that the rules are *applied consistently by the people who do the grading*. That consistency — "would an independent, qualified rater give this record the same basis and confidence we did?" — is the one property the machine cannot check, because it is a property of the humans, not of the data. It is the difference between the assertion "we grade under a documented rubric" and the measurement "our grades are reproducible." It is also the experiment that closes the last asymmetry between the two halves of this paper's architecture: the compiler half is measured (§8.1), while the human half — until this study runs — is specified rather than measured. The gap is a fact about the project's age, not a defect in the design; the study below is the defined work that closes it. Measuring it requires raters with a specific, scarce skill set: fluent reading in **both Arabic and French** (the source register is majority-Arabic and majority-French),

working knowledge of Tunisian institutions (a Ministry of the Interior official is not a Ministry of Defence official, and the distinction is load-bearing), and enough training to apply a four-grade and four-basis rubric consistently. That skill profile is not a crowd-sourcing pool; it is a small professional population, which is why the study is a funded research item rather than a weekend exercise.

Design. We propose a two-round, multi-rater design over a stratified sample:

- **Sample.** ~300 records stratified by kind (positions, relationships, events, and the v0.0.2 kinds) and deliberately *oversampled* toward the difficult cases — grade C/D records, `inferred` bases, disputed intervals — because those are where disagreement lives. Disagreement on A-grade gazette appointments would be a rubric failure; disagreement on the murky middle is informative.
- **Raters.** Three independent raters, none of whom authored the records they grade, working blind to each other's grades and blind to the project's own published grades.
- **Rounds.** Round 1: all three grade all 300. The disagreement set is then analysed — not adjudicated — into a *taxonomy of disagreement* (is it rubric ambiguity, source ambiguity, or genuine editorial judgment?). The rubric is revised where it is ambiguous, and Round 2 regrades only the disagreement set. This is the standard design; the calibration round is where the real cost sits, and it is also where the real value sits, because its output — the disagreement taxonomy — is the artifact that tells the project (and the field) exactly where evidence grading is underspecified.
- **Measures.** Fleiss' kappa (three raters) per field (confidence, basis) and per record kind, with **pre-specified target thresholds** — $\kappa \geq 0.8$ for basis and $\kappa \geq 0.7$ for confidence — agreed before the study runs, rather than interpreted as universal acceptance standards; these are conventions the field commonly works with, not laws of nature. Because confidence and basis are categorical and likely imbalanced, Krippendorff's alpha will be reported alongside Fleiss' kappa as a complementary measure robust to uneven category use. The Round 2 deltas are reported whether they meet the target or not. A kappa that stays low after revision is a finding, not a failure — it names the ceiling of what a two-axis rubric can enforce.

Parallel arm: model-graded consistency. The same 300 records are graded by an independent language model given only the public rubric, run twice under varied prompts. This measures a different question — *is the rubric expressible enough that a machine applies it consistently?* — and the human-vs-model divergence is itself a publishable result: where models are systematically more lenient, or where they invent basis from confidence, is exactly the failure mode this architecture exists to police. The model arm is cheap and fast; the human arm is the expensive one, and the two together answer the reviewer's question and the field's question at once.

Published deliverables. The rubric as administered (v2 after calibration); the anonymised grading dataset (rater × record × grade); the disagreement taxonomy; kappa per field, per kind, and per round; and the human-vs-model comparison. All published with the paper, under the same source-backed discipline as the dataset itself — including the negative results.

Why the project cannot do this alone. The editorial capacity to grade exists; the *independent* capacity does not. Independence is the point: raters who are not the authors, paid for their time, with no stake in the grades coming out well. That is a funding requirement by construction, and it is the specific deliverable this section defines for funders.

10.2 Further items

1. **Sensitivity analysis and external validation of the indices.** The named constants of §5.4 and §5.7 are the explicit targets: the basis-discount weights (1.0/0.55/0.2/0), the 8-year backwards window, and the ±1-year ~ slack. The deliverable is the ranking delta under each perturbation, so a reader can see how far the published "most influential" ordering moves when a constant moves — turning the constants from stated choices into *measured* choices. The same corpus (press-citation frequency, gazette counts) serves external validation of the indices against an independent signal, with the method and the null results published.
2. **An external literature review** of the evidence-grading and verification space, conducted outside the project — the internal prior-art study [15] is the current basis of the Table 3 novelty claim, which §4 grades C under the paper's own rubric until an independent review corroborates or refutes it.
3. **A formal account of the validator layer.** The §5.10 contract is stated in prose; a specification of V1–V24 in a machine-checkable form (e.g., a Datalog fragment or SMT encoding) would let the enforcement itself be subject to the same standard of proof it applies to the data — the "who enforces the compiler?" recursion, moving the

validators out of the trusted computing base. The mutation-testing extension this item once motivated has been run and is reported in §8.1: the campaign now covers the full validator surface rather than only the hardening validators, and the *latent* class — mutations of ingestion- and schema-layer guards that no current record exercises — has been closed empirically by the synthetic-fixture suite (`scripts/test-validators.ts`) and the fixture-pipeline runner (`scripts/test-pipeline.ts`), reaching 25 of 26 mutations killed. What remains is the structural half: re-encoding the validators in a machine-checkable form so that the enforcement layer is checked by a checker rather than by the campaign's next pass.

4. **Evidence independence and full derivation provenance.** An independence-aware representation of sources (§8.3, item 5) — a graph recording which source is downstream of which, so corroboration counts independent lineages rather than URLs, and a grade B cannot rest on three copies of one wire story — paired with the uniform `derived_from`/`derivation` contract of §5.5, so "why does the system believe this?" is answered by traversing claim → inputs → sources rather than by a flag. Together they close the two largest remaining provenance gaps.
5. **Interpretation-level validators.** Extending the gate from record validity to graph-interpretation validity (§8.3, item 6): temporal coherence of paths, relationship-type compatibility along chains, confidence-propagation floors, and minimum-evidence thresholds before a *derived network* claim is rendered — the checks that keep a locally-honest graph from composing into a narratively misleading one.
6. **The comparative atlas.** The region layer is the interface; deploying the schema to a second state is the concrete first test of the portability claim of §9 and the untested "Layer 1 unchanged" assertion it makes. The design target is a real fork, not a paper exercise: a second jurisdiction running the framework independently, with its own ontology, data and editorial responsibility, so that the "standardised infrastructure, locally controlled knowledge" question is answered by demonstration rather than assertion. An RDF/triples export (the epistemic envelope mapped onto a PROV-adjacent vocabulary) is the natural companion, grounding the contribution in the semantic-web literature it responds to.

11. Conclusion

DeepTunisia is an attempt to make a specific promise machine-checkable: that a dataset about contested political history cannot be quietly wrong. The mechanism is simple in outline — evidence policy compiled as build-time invariants, uncertain time represented rather than resolved, influence anchored or rejected, the system's own shortfalls published by the same build that publishes the data — and the evidence that the promise is kept is the audit of §7: seven real bypasses, found by testing the enforcement, closed with validators and regression tests that fail on the pre-fix dataset. The design's limits are stated as loudly as its claims: the gate makes status structural and non-upgradable; it does not authenticate documents, and it does not decide whether an epistemic judgment is right — only whether the record honours the judgment it declares (Axiom A1). The architecture's answer to everything the gate cannot do is not a bigger gate; it is the division of labour of §9 — the compiler enforces the epistemic constitution, the community exercises epistemic judgment, git records the audit trail, and every correction re-enters the compiler. Within those limits, the architecture demonstrates that the epistemic discipline the semantic web imagined as a trust layer [1,18], medicine formalised as an evidence ladder [9,10], and Wikipedia maintains as a policy [24] can be *compiled* — and that when it is, a dataset of 790 claim-bearing records — more than a thousand separable claims, since each record carries at least one and a position carries two (the officeholding and its span, §5.2) — about living people and operating institutions can honestly offer itself for adversarial inspection by humans and machines alike. And because the enforcement layer is jurisdiction-agnostic (§9), the compiler is the contribution — but only half of it. The other half is the loop that completes the compiler: human epistemic judgment becomes an executable contract, machine enforcement publishes it, and public scrutiny, challenge and recompilation close the circle. Tunisia is where the standard was first compiled and first subjected to that loop — not the only place it can run.

Data and code availability

- Source code, dataset and build: the public repository of the DeepTunisia project. The canonical data is plain-text YAML under version control; every change is a dated, attributable diff, and the corrections page surfaces the resulting history. One qualification, stated because this paper is about exactly this: git authorship is *attributable* in the sense of a dated, named diff, but it is not *cryptographically authenticated* — the repository does not currently require signed commits (GPG/Sigstore). Signing is a named hardening item: without it, a commit's

author field is a claim like any other, and history could in principle be rewritten by someone with write access. The build's determinism and the published exports do not depend on signing; the *attribution* of each change does, and the paper says so rather than implying git authorship is proof of identity.

- **Forking** (see §9): the repository is the framework. A new jurisdiction forks it, replaces the canonical `data/` directory and the jurisdiction-specific ontology, and retains the enforcement layer — the evidence envelope, validators, build gate and audit machinery — without modification. The build runs identically against the new data, so a fork inherits the guarantees, not the judgments.
- Reproducibility: `npm run data && npm run test` rebuilds the graph and runs the invariant suite from a clean checkout. Every claim this paper makes about the implementation is verifiable against that checkout: the claim envelope is composed in `scripts/schema.ts`, the truth table of §5.2 is the `deriveBasis` function, the validators of Appendix A are the fail/warn checks of `scripts/build-data.ts`, the seven audit fixtures of §7 are regression tests in `scripts/test-data.ts` that fail on the pre-fix dataset, the validator invariants are additionally exercised with synthetic inputs (`scripts/test-validators.ts`) and by running the real build against injected fixture datasets (`scripts/test-pipeline.ts`), the community-layer invariants (including graph/community isolation) are asserted in `scripts/test-community.ts` from the actual schema and build files, and the published statistics are rewritten from the graph by the build itself. A reviewer who wants to attack the architecture can do so by breaking a validator and watching whether the suite notices — `npm run mutation` automates that attack, applies each breakage to a working tree it restores byte-exact, and reports the kill rate, which is how the §8.1 campaign figures were produced.
- The full dataset ships beside the website as downloadable JSON and CSV (`dataset.json`, `positions.csv`, `relationships.csv`, `sources.csv`, `geo.json`).
- An archived, versioned snapshot with a DOI will be deposited before submission. The repository's license decision is in progress — an open-source license is being added — and the paper's own use of the term is conditional on it (§1); the archived snapshot will carry whatever license is in force at deposit time.
- **Citation verification, applied to the paper itself.** This paper's references were verified against primary or registry sources on 2026-08-08: [1]–[3], [5], [8], [9], [11]–[14], [17], [19], [23], [25], [26], [28], [29] against Crossref, dblp, the W3C publication record, or the CourtListener docket (1:22-cv-01461, S.D.N.Y.) for [23]; [20] and [21] against the original articles; [30] against the schema.org vocabulary itself. The pass caught and corrected real errors — [8] was mis-dated, [3]'s title was incomplete, [7] was mis-dated. The remaining entries ([10], [18], [22], [24], [27]) are standard references or internal documents; [22] and [10] require the same primary check before publication, and the checking record is kept with the paper in the same spirit as the dataset's own sources register. A paper whose thesis is that evidence standards should be enforced rather than asserted cannot ship unverified bibliographic claims.

References

[1] Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5), 34–43.

[2] Vrandečić, D., Krötzsch, M. (2014). Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10), 78–85.

[3] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., Vrandečić, D. (2014). Introducing Wikidata to the Linked Data Web. *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, Lecture Notes in Computer Science 8797, 50–65. DOI: 10.1007/978-3-319-11964-9_4. (Verified against Crossref, 2026-08-08.)

[4] DeepTunisia Project (2026). From a Political Database to a National Influence Graph — Hardening Revision (v0.0.2 sprint specification, revision v2). *Canonical specification*, deeptunisia.org.

[5] Hogan, A., et al. (2021). Knowledge Graphs. *ACM Computing Surveys*, 54(4), Article 71.

[6] DeepTunisia Project (2026). Authenticity — bots, sybils, and what verification can actually deliver. *Research note*, deeptunisia.org. (Archived with the project.)

[7] Moreau, L., Missier, P., eds. (2013). PROV-DM: The PROV Data Model. *W3C Recommendation*, 30 April 2013. (Verified against the W3C publication record, 2026-08-08.)

[8] Dylla, M., Sozio, M., Theobald, M. (2011). Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases.

Datenbanksysteme für Business, Technologie und Web (BTW 2011), 474–493. (Verified against dblp, 2026-08-08.)

[9] Guyatt, G. H., Oxman, A. D., Vist, G. E., et al. (2008). GRADE: An Emerging Consensus on Rating Quality of Evidence and Strength of Recommendations. *BMJ*, 336(7650), 924–926.

[10] OCEBM Levels of Evidence Working Group (2011). The Oxford 2011 Levels of Evidence. *Oxford Centre for Evidence-Based Medicine*.

[11] Graves, L., Amazeen, M. A. (2019). Fact-Checking as Idea and Practice in Journalism. *Oxford Research Encyclopedia of Communication*.

[12] Vlachos, A., Riedel, S. (2014). Fact Checking: Task Definition and Dataset Construction. *Proceedings of the ACL Workshop on Language Technologies and Computational Social Science*, 18–22.

[13] Silberzahn, R., Uhlmann, E. L., et al. (2018). Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results. *Advances in Methods and Practices in Psychological Science*, 1(3), 337–356.

[14] Lazer, D., Pentland, A., Adamic, L., et al. (2009). Computational Social Science. *Science*, 323(5915), 721–723.

[15] DeepTunisia Project (2026). Prior-art study. *Project document*, deeptunisia.org.

[16] DeepTunisia Project (2026). Research roadmap. *Project document*, deeptunisia.org.

[17] Hobbs, J. R., Pan, F. (2004). An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Information Processing*, 3(1), 66–85. DOI: 10.1145/1017068.1017073. (Verified against Crossref, 2026-08-08.)

[18] Berners-Lee, T. (1999). *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. HarperSanFrancisco.

[19] Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11), 832–843.

[20] Willison, S. (2025). Unauthorized Experiment on CMV Involving AI-generated Comments. *SimonWillison.net*, 26 April 2025. (Verified against the original post, 2026-08-08.)

[21] Bell, K. (2025). Researchers secretly experimented on Reddit users with AI-generated comments. *Engadget*, 28 April 2025 (updated 29 April 2025). (Verified against the original article, 2026-08-08.)

[22] Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1), 35–41.

[23] *Mata v. Avianca, Inc.*, 678 F. Supp. 3d 443 (S.D.N.Y. 2023), No. 1:22-cv-01461 — sanctions for the submission of AI-fabricated case citations.

[24] Wikipedia contributors. Wikipedia:Verifiability — policy page. Wikimedia Foundation. (Accessed 2026.)

[25] Meyer, B. (1992). Applying "Design by Contract". *IEEE Computer*, 25(10), 40–51. DOI: 10.1109/2.161279. (Verified against Crossref, 2026-08-08.)

[26] Necula, G. C. (1997). Proof-Carrying Code. *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '97)*, 106–119. (Verified against Crossref, 2026-08-08.)

[27] DeepTunisia Project (2026). Brief for counsel — legal exposure of the community layer. *Internal project document*, deeptunisia.org. (Cited for the project's legal posture; not public at the time of writing.)

[28] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E., Mitchell, T. (2010). Toward an Architecture for Never-Ending Language Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1), 1306–1313. DOI: 10.1609/aaai.v24i1.7519. (Verified against Crossref, 2026-08-08.)

[29] Artz, D., Gil, Y. (2007). A survey of trust in computer science and the Semantic Web. *Journal of Web Semantics*, 5(2), 58–71. DOI: 10.1016/j.websem.2007.03.002. (Verified against Crossref, 2026-08-08.)

[30] schema.org (2026). ClaimReview — vocabulary entry. Schema.org, accessed 8 August 2026. https://schema.org/ClaimReview

Appendix A — Validator table (V1–V24)

ID	Validator	Scope	Behaviour
V1	ownership percentage sum ≤ 100	all `ownership` edges, per company, per `direct` group, per interval window (sweep-line)	**fail** > 100.001
V2	capital/registration date consistency	company `founded` vs `institution.active`, `capital.date`	**fail** on incoherent ordering
V3	CIN/registration validity	`registration.number`	**fail** on duplicates
V4	contract award → winner	contracts with award status	**fail** no winner; **warn** announced-but-not-awarded
V5	procurement losers sanity	contracts	**warn** no winner and value > 0
V6	duplicate company detection	name normalisation (Arabic/French/English), Levenshtein + token similarity	**warn** > 0.85; **fail** identical registration
V7	board overlap / conflict of interest	person in two overlapping `board` edges	**warn**
V8	event ordering + causal cycles	`causes`→`consequence` intervals; Tarjan SCC	**fail** on ordering violation; **fail** on cycle
V9	unmoored influence	influence-family edges with no documented adjacency	**fail** influence/advisory; **warn** reported-influence
V10	geographic hierarchy	places parent chain, coordinates, bounding box	**fail** no parent/coords for point classes; **warn** otherwise
V11	orphaned events	events with no participants/orgs/documents	**warn**
V12	missing citations	every claim-envelope kind; the legacy kinds that predate the envelope are pinned by the rule-2 ratchet (§8.1)	**fail**
V13	equity validity	`equity.pct` `[0,100]`, `beneficial` → attributed	**fail** invalid; **warn** untattributed beneficial
V14	directionality	directed edge semantics, endpoint kinds	**fail** reversed/incorrect endpoint
V15	interval sanity	all record kinds, end-before-start	**fail**
V16	duplicate relationships	same (from,to,type,subtype), overlapping intervals	**fail** with merge/conflict guidance
V17	rupture linked	`rupture: true` events in the causal graph	**warn**
V18	inferred completeness (H1)	derived basis, every claim kind	**fail** if inferred lacks reasoning or falsifier

ID	Validator	Scope	Behaviour
V19	schema-field audit (H2)	strict schemas, every YAML key declared	**fail** on undeclared key or missing envelope field
V20	attribution completeness (H3)	grades C/D on every claim kind	**fail** without `attributed_to`
V21	calendar-date validity (H4)	every date token, every precision	**fail** on rollover or out-of-range
V22	interval contradiction (H5)	every resolved interval	**fail** on contradiction without dispute; **fail** on inverted core; trims published
V23	review provenance (H6)	every `review` object	**fail** non-ISO date or non-enum method
V24	succession sanity (H7)	derived successions, gaps, overlaps	**fail** if stats not reproducible from emitted constants; **warn** unknown-end exemptions