CASE STUDY · PRIVACY AI · LIVE DEMO

TermsGuard AI

Cosine similarity alone marks "we do not sell your data" as a violation, so the classifier proposes and a model rules. 102 hand-labelled reference clauses, a 4,527-chunk retrieval index, and up to 15 flagged clauses sent to Claude for verification per analysis.

> What this evidences: a purely semantic classifier gets negation exactly backwards, and what you do about it. The interesting problem is not detection. It is the class of bug where the verifier silently never runs.

Taiwo Alabi · MSc Artificial Intelligence, National College of Ireland

Feb 2026 to Apr 2026 · Python · Flask · sentence-transformers (MiniLM) · FAISS · Anthropic API · Docker

01 · The problem

244 hours of unread text

Reading every privacy policy an average internet user meets would take roughly 244 hours a year. Nobody does it.

So people agree to data selling and tracking without knowing they have.

The problem is not that the documents are secret. They are published, free, and legally binding. They are simply unreadable at the length and density they are written, and the one clause that matters is buried among forty that do not.

02 · The bug worth surfacing first

The verifier that never ran

The previous build reported that the model had verified each clause, but the verifier had never actually run on the deployed service. The Anthropic key had no credit. The except block swallowed the 400 error. The app returned cosine scores while still reporting `llm_verification_enabled: true`.

A clause was labelled "Checked by the model" when no model had looked at it.

> **Why the silent fallback existed.** The original design chose to degrade to matching-only mode rather than refuse to work. The intent was correct (the demo stays live even when the paid verifier is unavailable) but the implementation reported success without running anything. The two failure modes have to be reported separately.

The fix added three routes for the verifier, tried in order with retry and a fallback model. The response now carries `verification_source` and `verification_error` so a silent fallback is impossible. A regression suite of twelve inputs runs against the live build with the real model ruling.

Two real bugs were found only because the verifier started actually running:

- A benign "refund policy" clause was being flagged as an Article 82 violation. Fixed by stating in the verifier prompt that generic commercial statements are not GDPR clauses.
- A politely worded third-party analytics clause was being read as compliant because asking permission was reasoning as lawful. Fixed by stating that politeness does not change what a clause does.

> **The principle.** "The verifier ran" is a different claim from "the verifier ran successfully." A pipeline that conflates the two is one credit balance away from shipping a false claim about what it did.

---

03 · How it works

Similarity proposes, rules kill the obvious, the verifier rules on what is left

Text is split into individual clauses and de-duplicated. Each clause is turned into a row of numbers by `all-MiniLM-L6-v2`, then compared by cosine similarity against labelled examples of fair and unfair clauses. Anything that looks unfair has GDPR context pulled for it from a FAISS index, and Claude Haiku double-checks the flag before it is shown.

STAGE	WHAT IT DOES	WHAT IT DECIDES
1. Cosine similarity	Compares each clause to 102 labelled examples (52 fair, 50 unfair)	Which clauses look like they might be unfair
2. Rule guards	Negation, protective context, children's data, keyword boosts	Which of those are obvious mistakes to kill
3. Verifier (Claude Haiku)	Re-reads what survived, with GDPR context pulled from the 4,527-chunk index	Which flags actually go to the reader

---

04 · The two reference sets

What the system compares against

The system carries two reference structures, and they do different jobs.

REFERENCE	SIZE	PURPOSE
Labelled clauses	102	52 fair + 50 unfair. Decides whether something is a problem.
Retrieval chunks	4,527	GDPR + EU consumer law + DPC guidance. Supplies why, when a clause is flagged.

That split matters. The labelled clauses decide whether something is a problem. The GDPR corpus supplies why, and it is the second half that makes a flag worth acting on. The page shows the passage a flag was drawn from rather than an article it claims to quote.

---

05 · Why similarity alone is not enough

This is the part I would defend hardest in an interview

Consider two sentences:

> *"We sell your personal data to third parties."*
>
> *"We do not sell your personal data to third parties."*

To a computer that only measures word patterns, these two sentences look almost identical. They share nearly every word, the same grammar, the same subject. So a system that judges purely by how similar the words look will flag the second sentence, the reassurance, as a violation, because it never actually checked which way the sentence points.

So the pipeline carries a layer of explicit rule guards on top of the similarity score:

- **Negation detection.** For exactly the case above.
- **Protective-context detection.** For clauses that describe a right being granted rather than removed.
- **Children's-data check.** Because clauses about minors carry obligations that generic similarity scoring does not weight correctly.
- **Keyword boosting.** For terms whose presence changes the reading regardless of how the sentence embeds.

None of these are clever. They exist because I ran the similarity-only version, read what it got wrong, and found the errors clustered around cases where meaning is inverted by a single word.

---

06 · The verification layer

Three routing paths so silent fallback is impossible

When an API key is configured, Claude Haiku re-reads each flagged clause and removes false positives before anything is shown to the user.

The design decision worth noting is the fallback: with no key set, the tool runs in matching-only mode rather than failing. It degrades to a less precise version of itself instead of refusing to work, which means the demo is always live and the tool is not hostage to a paid dependency.

PATH	TRIGGER	REPORTED AS
1. Claude Haiku with API key	ANTHROPIC_API_KEY configured and credit available	`verification_source: "claude-haiku"`
2. Fallback model	Path 1 fails, fallback configured	`verification_source: "fallback-model"`
3. Matching only	No key or no credit	`verification_source: "matching-only"`, `verification_error` populated

> **The contract.** A reader of the API response can tell, by reading `verification_source`, which of the three paths produced this result. No path is silent.

---

07 · Saying how sure it is

Tiered by confidence, not flat verdicts

Flags are tiered by confidence rather than presented as flat verdicts, with thresholds at 0.55 for high and 0.45 for medium. A document-level risk assessment then combines the proportion of clauses flagged with the average confidence across them, so a policy with two borderline flags is not reported the same way as one with fifteen strong ones.

This is the same instinct that runs through my other projects: state uncertainty rather than hide it behind a single number.

---

08 · Results

MEASURE	RESULT
End-to-end latency	28 seconds to under 10 for short policies
Reference clauses	102 hand-labelled (52 fair, 50 unfair)
Retrieval index size	4,527 chunks of GDPR, EU consumer law, DPC guidance
Clauses sent to the verifier per analysis	up to 15
Live regression suite	12 inputs, verified against the deployed service on 2026-10-25
Two real bugs found when the verifier started running	refund-policy false positive; politeness-as-consent false negative

---

09 · Honest limitations

What this is not, and what it does not prove

- **It is not legal advice.** The page says so. It surfaces clauses worth reading properly; it does not tell you your rights.
- **Most of the retrieval corpus is not regulation.** Of the 4,527 chunks, only about two percent cite a GDPR article at all. The retrieved passage is shown as context, not as the article text.
- **Politeness can read as consent.** The prompt now states that the tone of a clause does not change what it does, but this is an ongoing risk for any LLM verifier trained on human text.
- **This was a contributed project, not a solo build.** My work was the retrieval pipeline, the grounding, the rule guards, and the latency reduction.

10 · Run it

Easiest is the live demo: newtazer-terms-guard-ai.hf.space

Locally:

```
pip install -r requirements.txt
python server.py            # http://127.0.0.1:7860

# Optional, for the verification layer:
export ANTHROPIC_API_KEY="sk-ant-..."
```

Stack: Python · Flask · sentence-transformers (MiniLM) · FAISS · Anthropic API · Docker

11 · Terms used in this report

Cosine similarity
a way of measuring how close two pieces of text are in meaning, not just how many words they share.

Embedding
turning a piece of text into a row of numbers, so a computer can compare meaning mathematically instead of just matching words.

FAISS index
a fast lookup table, built for searching millions of embeddings for the closest match almost instantly.

Negation
the word or phrase that flips a sentence's meaning, like turning "we sell your data" into "we do not sell your data".

Rule guard
an explicit check that runs after similarity scoring to kill obvious mistakes (negation, protective context, children's data, keyword boosts).

Verifier
a separate model (Claude Haiku) that re-reads what survived the rule guards, with GDPR context pulled from the retrieval index.

Links

Live demo: newtazer-terms-guard-ai.hf.space

Project page: iamtamilore.github.io/p/termsguard

Source: github.com/iamtamilore/terms-guard-ai

Architecture: architecture diagram

Weekly corpus (14 EU and Big Tech policies): iamtamilore.github.io/corpus

Home: iamtamilore.github.io