Mapping CORDIS Horizon Europe projects to SDG targets

Description document, registration §6 item 8. The registration is `docs/cordis-sdg-registration-v10.md` in this repository; section and item numbers throughout refer to it.

registration	**`cordis-sdg-registration-v10.md`, sha256 `c5f2a26ce6049e801a7e757efc2cbe4253689987fad761f1d97cfb5b0ad68fb2`**
ratified	v10 on 2026-09-12; its predecessor **was ratified by hash on 2026-09-11, before any pipeline code was written**, and sections 1 to 6, 9 and 10 are byte-identical between them. The commit history shows it: the first pipeline code, `eb3ec4a`, was committed at 23:01Z on 2026-09-11, seventeen minutes after the ratification recorded in the working record
licence	**CC BY 4.0** for the documentation, the crosswalk and the synonym list; **MIT** for the code (§9)
certificate	one name, **Attila Angyan**

Every number in this document is reproducible from an artefact whose hash is below. Nothing here is typed from memory.

---

1. What was built, and the one rule everything else follows from

The mapping decision is made by rules a reader can run by hand. A language model writes the explanation of a decision already made; it never makes one. Nothing in the scoring path calls a model, and that is structural rather than a matter of discipline: the decision and the explanation are separate programs, and the explanation reads only the evidence rows of a decision already fixed.

Two load-bearing artefacts, **both built before any project text was scored**:

Crosswalk	**235 rows, EuroSciVoc category → SDG target, each with a one-line justification citing the target's own text, reviewed before use**
Vocabulary	key terms extracted from target labels by a committed written rule, plus a curated synonym list, a stop-list and a negation rule

Neither was derived from project text. The crosswalk was built from the category vocabulary on one side and the taxonomy's target labels on the other; the only project data consulted was the list of distinct category paths, which is the table's own left-hand side. **That is what makes the**

held-out evaluation worth anything, and it is checkable rather than asserted — `verify_crosswalk.py` reads the same two sources and nothing else.

A project reaches a target through six independent channels: five text fields and the reviewed crosswalk. Every match is recorded as **the exact phrase found with its source field**, which is what makes the result explainable at row level rather than in general.

---

2. The registration came first, and it was ratified by hash

This entry was registered before it was built, so **results could not move the method**. The registration fixes the scope, the sources, the scoring rule, the validation design, the roles, the deliverables and the timeline; it went through ten drafts, every change attributed and logged, and **each successive draft was read against itself for the fault of amending one section and leaving a dependent section standing** — a fault caught five times and logged each time.

Validation was designed before it was run. One seeded draw of 150 projects, split 50 development and 100 evaluation. Parameters are tuned on the 50 and frozen by the owner's word before the 100 is scored. The evaluation labels are sealed **and unread** until the frozen pipeline has run once on the 100 and its output hash is posted.

The freeze is enforced by the code, not by good intentions. Four conditions, all required, each refusing by name: the freeze is recorded; the parameters' values digest equals the recorded one; the caller states which freeze the run is under; and every rule file matches the digest recorded for it, with the run hashing the files it actually opens, found by an audit hook rather than declared beside the loaders. **Each of the four has been shown refusing on purpose.**

---

3. The numbers (§4)

Evaluation 100, held out, scored once under the freeze

	tp	fp	fn	precision	recall	F1
target level	21	26	74	**0.447**	**0.221**	0.296
goal level, secondary	29	42	50	0.408	0.367	0.387

The unit is one (project, target-URI) pair, matched exactly. The reference holds 95 pairs over 100 projects, 31 of which carry no target at all.

Development 50, reported and not the headline

It is what the parameters were tuned against, so it cannot measure the method. Two labellers worked it, **each set uploaded and hashed before either was revealed**:

Cohen's kappa, target level	**0.677**
Cohen's kappa, goal level	**0.806**
agreement rate	0.9942 over 8,450 (project, target) items

That figure is the scale for reading the pipeline's 0.447 precision. Two careful labellers do not fully agree about what a project is about, so the reference is not a ground truth.

Kappa's item universe is every (project, target) pair over the 169 targets, each classified yes or no. The alternative — the union of pairs somebody assigned — has no negative class and leaves kappa undefined. **The choice is stated because it changes the number**: the negative class is enormous, expected agreement is very high, and kappa is accordingly harsh.

The whole snapshot

projects scored	**23,451**
rows in the mapping table	28,833
with at least one target	9,394 (40.1%)
goal level only	6,882 (29.3%)
nothing at all	**7,175 (30.6%)**
goals reached	**17 of 17**
targets never assigned anywhere	**33 of 169**

---

4. The finding: recall is bounded by the method, and the bound is measured

A recall number invites the reading that tuning would fix it. **Here it would not.** For each pair the reference contains, ask whether the pipeline found *any* evidence for that target in that project, at any score:

reference pairs	**95**
with some evidence	42
with no evidence at any threshold	**53**
recall ceiling	**0.442**
recall achieved	0.221, **exactly half the ceiling**

53 of 95 pairs are not recoverable by lowering a threshold, reweighting, or any other tuning move. A project whose objective describes its science without naming the outcome an SDG target names is invisible to a rule a reader can run by hand. **That is the cost of §3's requirement**, and it is the honest centre of this result rather than a footnote to it.

The held-out set behaves like the set the method was tuned on: the development ceiling is 0.446 (45 of 101 pairs, counting a pair once if either of the two development-set labellers assigned it) and the evaluation ceiling is 0.442 (42 of 95, over one labeller). Those are not like for like, and the closeness is evidence about the method's reach rather than a controlled comparison.

The failure is silence, not noise. Of the 64 projects where the pipeline and the reference disagree: **46 are recall-only** — the pipeline found nothing the labeller found — **4 are precision-only**, and 14 disagree in both directions. A method that mostly says nothing is a different object from one that mostly says the wrong thing, and it fails in the safer direction.

5. The independence limitation of §4.4, in its own words

The registration requires this section to quote rather than summarise. Two names in the quotation are seats, not people: **Hermes** and **counsel** are two of the automated assistants that worked this entry, named as its working records name them. Hermes reviewed the crosswalk and was one of the two development-set labellers; counsel is the reviewing seat that reads every build from its own copy of the code, which is why the clause bars it from labelling the evaluation set.

> **4. What "blind" cuts.** Labellers do not see the pipeline, its output or each other's labels before their own are sealed. **Labellers may have seen the reviewed crosswalk**; the registration says so rather than pretending otherwise, and Hermes's crosswalk pass is scoped in §4.6 to leak as little as possible. **And counsel cannot be an evaluation labeller at all:** its standing duty is to read every build hash from its own worktree, which IS the pipeline, so a counsel label on the evaluation set would come from the one seat guaranteed to have seen the method and its code. v2 caught this conflict for Hermes and missed it for counsel. **Counsel therefore labels the development 50 only**, where labels are used for tuning and contamination costs nothing.

What that costs in practice. The evaluation set has one labeller, because the seat that could have been a second one is disqualified by the clause above. The registration compensated with a human anchor, and the anchor was not made — see §6 below.

---

6. Limitations

This section is a pointer, not a summary. The limitations, the uncertain cases with worked examples, and the possible improvements are registration section 6 item 5's own deliverable, and they live in `docs/limitations.md`.

They were in both places until this commit. **A limitation written twice is a limitation that drifts**, and this registration has paid for a rule living in two places four times in its own sections. What stood here is there now, and more of it: the disagreement shape is carried down to named projects with the evidence cells that produced them.

`docs/limitations.md` covers the measured recall ceiling; that no human labelled anything in this study; the single blind automated reference and the human anchor that was never made; the third of the corpus that receives nothing; the band distribution and what a band does not mean; the deliberately unmapped four fifths of the category vocabulary; the parked explanation pass; and scope. **Its section 2 works through four failure mechanisms on real projects**, and its section 3 gives five improvements, each with what it would cost and what it would not fix.

Section 4's numbers stay here, because the bounded recall is this study's result rather than an apology for it, and **section 5 keeps the independence limitation of §4.4 in its own words** because registration section 6 item 8 asks this document for that by name.

---

7. Provenance

7.1 The snapshot

One dataset in four distributions plus the taxonomy, each hashed with its download moment and its upstream `Last-Modified` in `data/manifest.json`, whose own digest `bf558b0c04db70acc9a83b5badaeae959f7f85ad8ccff472ab68d22a03dce2f3` appears on **every row** of the mapping table.

The snapshot is not one upstream moment and does not claim to be. The projects extract was last modified upstream eight days after the other three, and the four were retrieved at three local moments. Anything computed across them carries that gap, and the manifest states it rather than leaving it to be inferred from four hashes side by side.

A standard CSV reader gets `project.csv` wrong. CORDIS under-escapes any field whose content *begins* with a quotation mark. Measured across every field: **994 rows (4.24%) carry such a field; a standard reader breaks 193 of them outright and silently alters 801 more.** Every read goes through one reader that splits on the literal separator. **This was diagnosed in someone else's data and then reproduced in ours the same day** — four committed CSVs were written with unquoted free text, truncating 29 crosswalk justifications where the pipeline read them — because "their export is broken" had been treated as a fact about them rather than a property of the format.

7.2 The labelling chain, with its corrections

sealed 150-project hand-off	`fd506beb26684df464127df73d351b26f3cbdd6d8db6a8dacc73581e1d231d11`
evaluation-only hand-off	`1603a0d7eb351ac92cf17bc488e7a1180cf40305748120076888ac30383a43cc`
evaluation label set	`c9fa780aae00c2c0927577cbbcb3bedbaf0f0f5cbd9b191b1bfe4355a1a25532`

The label artefact records `1603a0d7…` as the artefact it was labelled from; the prose was read from `fd506beb…`, the sealed 150 held since earlier in the build. The labeller corrected this himself, unprompted, once the discrepancy was noticed.

Nothing turns on it, and the reason is measured rather than asserted: all 100 evaluation records are byte-identical between the two artefacts, 100 of 100. The labels therefore rest on exactly the text the hand-off carries, and both sides of every comparison read the same input. **The correction is recorded because a provenance field that is accurate only under a gloss is exactly the kind of thing this project exists to catch.**

Labellers and the pipeline read the same text, gaps included. Every field is present on every record; an absent field carries the literal `(absent)` and is listed in that record's `absent_fields`. A labeller who cannot tell "this project has no editorial description" from "this field was not included" reads the same gap two ways on two different projects.

7.3 The labelling-time question, named here rather than left to be found

114 seconds elapsed between the labeller receiving the hand-off and uploading 100 labels made from full prose.

The labeller's own account: the full prose of all 100 was read in the turns *before* the acknowledgement, batched ten projects at a time against the taxonomy already held from the crosswalk review, each judgement formed while reading, with nothing written to disk until the validation, assembly and upload pass. **The 114 seconds measured the upload path, not the reading.**

This is stated because the evaluation precision and recall rest on these labels as the sole reference, and because **a reader with the timestamps can compute what we computed. Better that the submission names it than that a judge finds it.** It is recorded as an account of a seat's own work, not as a measurement.

7.4 The ordering at the freeze

The ordering that carries the guarantee — **the freeze word first, and only then any evaluation label set uploaded** — held. A second ordering in the same word did not run as written: the review of the hand-off was to precede any label upload, and the upload preceded it by 96 seconds. **The review passed, so nothing turns on it, but it held rather than was guaranteed**, and the record says so.

7.5 One taxonomy, not two

`data/terms/sdg-targets.csv`	`a0c61a4f5b089863d6c8a60a83671495fb530969fa1a0e04228e3a01cc072b94`
targets / goals	169 / 17

Checked rather than asserted: the labels name 35 distinct targets and the pipeline named 25; **every one of both sets is in that taxonomy and none is outside it.** It is also the file the freeze record pins, so the evaluation guard would have refused the frozen run had the taxonomy moved — **the claim rests on a control, not on this document saying so.**

7.6 The artefacts, by hash

frozen pipeline commit	`0246412b44f722567feaa2e6c1b6e695de86c923`
parameters, values digest	`3808845b553dfda751bf4a3e5c87bd17f32b6ca8c195a0c54321ed25e03e09e1`
freeze recorded at	entry 207 of the project's working record, asserted by the caller on the command line as `--freeze-seq 207`
evaluation-100 output	`0938fe34e41cd5d8ddccfba69238481fdf6470e54046104ad7a3535da6743709`
§4.7 metrics	`babd37415581ce82ecb09c4501052ce811e834e57dad7c6735389ce321dd8a2e`
development-50 output	`45a514cf0c7a0984c419f086f11025077e9a4122e7bf2879263176b2b5430bbf`
mapping table, 23,451 projects	`181d378a3078014193f3ae4f684b00b375ed62f3d24b2b34c060f6457b4a00d3`

frozen pipeline commit	`0246412b44f722567feaa2e6c1b6e695de86c923`
figure counts	`03d79580081ca40bf0d1d25b5279a57880ddf09d750490553593387126d92f6f`

The parameters record hashes the parameter VALUES, not the file, with every documentation key stripped, so the recorded digest moves when a weight moves and stays still when someone explains one.

Every artefact above records the predecessor registration's hash, and that is correct rather than stale. A frozen output names the registration that governed the run that produced it. v10 added a second builder seat and pinned the timeline; **it moved nothing in sections 1 to 6, 9 or 10**, measured section by section, so nothing these artefacts depend on changed. Rewriting sealed bytes so a document reads tidily is the opposite of provenance.

7.7 How to reproduce

The repository builds itself from a clone. That is measured, not promised.

At commit `33e38de` the README's fifteen-step block was run in a fresh clone with nothing from the author's working tree except the snapshot payload under `data/raw/`, which is not committed by design and whose hashes are in the manifest. **Every step exited 0**, and every artefact the block regenerates matched its published hash:

taxonomy	`a0c61a4f…`
hand-off, 150	`fd506beb…`
hand-off, evaluation 100	`1603a0d7…`
development output	`45a514cf…`
frozen evaluation run	`0938fe34…`
§4.7 metrics	`babd3741…`

`git status` **after the run was empty**, so every committed generated file equals what a clone produces — not what it produces after a normalisation.

It was run twice, by two seats, and the second was not the author. The first run found a defect the author's own line-by-line check had missed: one input the metrics step reads was excluded from the repository, so the block passed in the tree that already had the file and failed everywhere else.

From a clone, not an archive, for the mapping table, because every row carries the commit that scored it and that is read from git rather than guessed; the build refuses rather than inventing it. Everything else reproduces from an archive or a clone.

The figures reproduce; the notebook does not. Matplotlib's SVG writer embeds a timestamp and randomised element ids, both now pinned, so a second execution gives six byte-identical figures. The notebook's own bytes carry per-cell execution timestamps, which are **kept deliberately** as the machine-checkable evidence that it was executed rather than written.

7.8 The frozen pipeline was sped up once, and the speed-up changed no byte

The frozen pipeline commit is not the commit that was frozen. The freeze fixed the pipeline at `c8a610477bc170817c20e0432bd197dec62c52c8`. Scoring the whole snapshot at that commit took about ten hours, because every one of 1,222 vocabulary phrases ran a regular expression over every project. So one line went in afterwards: skip a phrase's expression when the phrase is not a plain substring of the text, since an escaped literal between two zero-width assertions cannot match text that does not contain it. That is commit `0246412b44f722567feaa2e6c1b6e695de86c923`, and it scored the whole snapshot in minutes.

The owner allowed it on one condition, recorded as entry 244 of the project's working record: the output must equal the slow run's. An argument that the skip cannot change an answer is not the same as a measurement that it did not. So the slow pipeline was run again over all 23,451 projects, in a separate worktree at `c8a6104`, and its table was compared with the committed one.

control table, pipeline at `c8a6104`	**`a9f8f66f17810d48094c738a887285f732f1b0bf26ab101870bcadd7d7b6c130`**
committed mapping table, pipeline at `0246412`	`181d378a3078014193f3ae4f684b00b375ed62f3d24b2b34c060f6457b4a00d3`
rows on each side	28,833 over 23,451 projects
projects missing from the control	0
rows present on one side only	0
rows whose score or band differs	**0**
cells differing in any column but `pipeline_commit`	**0**
control with its commit hash swapped for `0246412`'s	**`181d378a3078014193f3ae4f684b00b375ed62f3d24b2b34c060f6457b4a00d3`**

The last row is the result. The two tables differ in `pipeline_commit` by construction, because each row records the code that scored it. Both hashes are 40 characters, so swapping one for the other moves no byte. **With that one column swapped, the control hashes to the committed table's own sha256: the two runs are byte-identical in everything else.** The swap cannot hide a difference. Each file carries its own commit exactly once per row and the other's never, and a single changed byte in the control breaks the match.

Two instruments, by two seats. The byte comparison is one. The other is a differential on every column, paired on project, target and assignment level. **It is scoped by the order the build runs in, not by what the control happens to contain**, and that correction was paid for. Its first version compared only the projects the control held, so a project the control had lost disappeared from both sides. Dropping a single-row project from a copy passed. Counsel then wrote an independent comparison, took a separate copy of the finished file, and reached the same zeros and the same substituted hash.

To re-run it. The control table is not committed, being a 24 MB second copy of a result the repository already holds. It is kept beside its build worktree as the artefact the hash above names, so **its hash cannot be checked from a clone**, and this section does not pretend otherwise. To rebuild it and compare:

```
git worktree add --detach ../cordis-sdg-slow c8a6104
cd ../cordis-sdg-slow
git checkout 0246412 -- . && git checkout HEAD -- scripts/pipeline.py
python scripts/build_mapping_table.py --freeze-seq 207 --records ../cordis-
sdg/data/corpus/records.jsonl --out data/corpus/mapping-table.csv
cd ../cordis-sdg
python scripts/compare_control_table.py --control ../cordis-sdg-slow/data/corpus/mapping-
table.csv
```

The third line restores every later file except the pipeline. Without it, a resumed build would score with the skip in place and stamp the fast commit, and the comparison would pass for the wrong reason. `compare_control_table.py` refuses to report until both instruments have caught a moved score, a moved band, a dropped single-row project and a changed byte. It also refuses if the two tables do not carry exactly the two expected commits.

---

8. Why we think this is good work, against the criteria

(1) Correct use of the taxonomy. Targets and goals are the SKOS concepts of the published EU Vocabularies snapshot, cited by URI, counted from the downloaded file rather than from a web page. Two defects in that snapshot were found and are recorded rather than silently repaired: a truncated English label on target 12.3 that was emitting the bare year `2030` as a key term across three targets, and a footnote marker glommed onto target 14.6's label.

(2) Quality of the mapping. Every crosswalk row carries a justification citing the target's own text and was reviewed before use. The rows most likely to be wrong are **named in the repository rather than left to be found**. Precision is reported against a reference whose own labellers agree at kappa 0.677.

(3) Explainability. Every assignment carries the exact phrases that produced it with their source fields and weights, and the crosswalk justification where one fired. **A reader can recompute any row by hand.**

(4) Reusability. The crosswalk is useful on its own, without this pipeline. The decision path is pure standard library — no dependency resolution stands between a reader and a reproduction. CC BY 4.0 and MIT.

(5) Data handling. Every input hashed, no payload committed, no personal data read, no credential in the repository or in any of the working records.

(6) Visualisation and communication. Six figures, committed as SVG, each reproducing byte-identically, with **the counts behind every chart published separately** so a figure can be checked without being run.

(7) Awareness of limitations. §6 above. The ceiling, the missing human anchor, the single reference, the 30.6% unassigned and the parked explanation pass are all stated with their numbers. **Several of them were found by looking for them rather than by being caught.**

(8) Prototype. §6 item 7 of the registration; built from the published artefacts by hash.

(9) This document.

---

9. What went wrong, and what the repository does about it

One failure recurred throughout this build: a control reported as applying to a tree it was not run against. It happened more than ten times and was usually found by a reviewer rather than by its author. Examples: a check that could not fail cited as evidence a bug was fixed; a verifier that parsed a file the same wrong way as the code it verified; a guard that tested one spelling of the thing it forbade and was bypassed by the flag one word over; a reproduction hash checked before the last edit and quoted after; a recall ceiling computed against rows that carried no project text, reporting 0.0 on a run with 21 true positives.

The repository's working rules are the scar tissue, and they are committed: a check must be able to fail and be made to fail on purpose; it must run last, against the committed state; it must not share the assumption it is checking; it must not mutate what it checks; and a hash is read back from the commit, never from the working tree that produced it.

This document lists those failures because criterion 7 rewards knowing where the work is weak, and because a submission that describes only its successes is asking to be checked rather than read.