Term Harvester: A FAIR Data Tool to Map or Reuse Trusted Online Vocabularies in Agriculture and Health Science Research Project Data Schemas

Damion Dooley[1,*], William Hsiao[1]

[1]Centre for Infectious Disease Genomics and One Health, Simon Fraser University, 8888 University Drive, Burnaby, Canada

Abstract

A unified encyclopedia of terminology in natural science, agriculture and other research domains is slowly emerging via various ontology development projects, along with the tools to integrate this vocabulary into the data schemas used to collect, catalogue and distribute research data. However, popular FAIR data management objectives, especially the findable and reusable principles which aspire to fully machine-readable and comparable research data, have not been achieved yet due to a variety of factors, ranging from the lack of documented observational or experimental metadata used to determine relevance, to a difference in variable scales employed in particular studies, to a lack of standardised, semantically formalized terminology. Standards for how to measure variables or describe experiments are still emerging, as are equivalence mappings between different standards. Term Harvester is a command line python script that obtains a standardised representation of configured source vocabularies (or parts thereof) such as Plant Ontology and AGRO ontologies in the Open Biological and Biomedical Ontology Foundry family of life science ontologies, FAO AGROVOC thesaurus terms, Library of Congress subject categories, Statistics Canada survey variables, Government of Canada Soil Database terms, USDA NRCS Soil Field Book, and ISO country subdivisions. The script can also call on Claude AI to extract picklists of terms from designated sections of research papers that are not already registered sources. Together this provides a library of controlled vocabulary terms which can be enhanced with ontology cross references as they become available, and which can be mapped to project metadata in order to fulfil the FAIR vision of data harmonisation.

Keywords

data sharing, FAIR principles, ontology, agriculture, research, standards

CEUR-WS.org/Vol-4342/paper12.pdf

1. Introduction

A unified encyclopedia of terminology in natural science, agriculture and other research domains is slowly emerging via various ontology development projects, along with the tools to integrate this vocabulary into the data schemas used to collect or distribute research data. However, popular FAIR data management objectives, especially the findable and reusable principles [1], which aspire to fully machine-readable and comparable research data, have not been achieved yet due to a full spectrum of factors, ranging from the lack of documented observational or experimental metadata to determine relevance, to a difference in variable scales employed in particular studies. Standards for how to measure variables or describe experiments are still emerging, as are equivalence mappings between different standards. For project data managers, a provisional response to this situation is to at least reference existing or emerging vocabularies as they appear on websites, pdf documents, and (multi)agency-curated thesauri or ontologies, so that the semantics of their use can be documented for other researchers to consider dataset applicability. This approach enables the adequate FAIR data semantic ontology coverage and mapping of research domains to be incorporated as they arise.

This paper describes our main contribution to Term Harvester (TH), an under-development open-source software project, namely the capability to harvest controlled vocabulary picklists of choices, also called enumerations, from varied online sources as illustrated by a growing library of source-specific

Proceedings of FOIS 2026 Satellite events co-located with the 16th International Conference on Formal Ontology in Information Systems (FOIS 2026), September 21-25, 2026, Vitória, Brazil.

*Corresponding author.

damion_dooley@sfu.ca (D. Dooley)

0000-0002-8844-9165 (D. Dooley); 0000-0002-1342-4043 (W. Hsiao)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-27

parsers, as well as the unification of all the different formats into one LinkML standard structured file called schema.yaml. As these vocabularies often do not have ontology-grade semantic identifiers, we also note how manual mappings to preferred semantically equivalent ontology terms can be included. Finally, we briefly examine the case where an agency or project already internally uses a favoured vocabulary library in some format such as a json file. The schema.yaml can be an input to a conversion.py script which generates that existing json file, thus facilitating the transition to use both standardised external and internal vocabularies for data management.



Figure 1: the extraction of controlled vocabulary from multiple sources via API requests or by parsing downloaded pdf, html or other document types.

2. Term Harvester

A specific organization or project can use the **term_harvester.py** command line python script to obtain and manage a standardised representation of configured source vocabularies or portions thereof (Fig. 1) in a single consolidated **schema.yaml** file. The schema.yaml file is coded in the LinkML specification format for easy comparison, searchability and reuse [2]. Initial development of Term Harvester has focused on selecting particular flat or hierarchic menus of terms from a growing library of vocabulary sources (currently as shown in Table 1) such as the Open Biological and Biomedical Ontology Foundry family of life science ontologies, FAO AGROVOC thesaurus, Library of Congress subject categories, Statistics Canada survey variables, Government of Canada Soil Database terms, USDA NRCS Soil Field Book terms, and ISO country subdivisions [3–5]. An up-to-date list of sources is provided in the GitHub repository's README.md documentation file. Note that it is incumbent on administrative and technical staff who are utilizing the Term Harvester software to ensure that their organization is fulfilling any licensing, copyright or attribution requirements that reference vocabularies may require as a result of extraction.

Term Harvester distinguishes itself from other vocabulary management tools insofar as it has no constraints about what document format extracted vocabularies originate in - it is up to each module to define the source document type and how to optimally extract vocabulary. TH is designed to enable more sophisticated selection of vocabulary, but is not intended to allow editing or modification of extracted content - such operations may be carried on further downstream of the schema.yaml vocabulary library file. Note that while a TH configuration file may specify a suite of picklists (enumerations) to populate a schema.yaml file, that file can also maintain manually created picklist content that isn't affected by Term Harvester updates. In other words, an accidental or intentional removal of a configuration source doesn't automatically trigger removal of related schema.yaml vocabulary (an extra TH option is available for doing that manually).

The Term Harvester vocabulary extraction tool can be used by technical data managers insofar as the three-step add/compile/build workflow generates concrete data products at each step, and yields a key piece of agency or project infrastructure, a vocabulary library composed of disparate but reference standard vocabularies. Ontology specialist knowledge (the SSSOM methodology in particular, discussed below) is important however for advancing the semantic interoperability mapping from non-ontological reference vocabularies into the more formal world of ontology vocabularies.

Table 1
A list of current vocabulary sources.

Source	Description
Agricultural	
AgriFoodCA	AgriFoodData Canada bilingual (EN/FR) agricultural picklists; add its entire GitHub directory of picklists or individual ones. ~400 terms
AGROVOC (API)	FAO multilingual agricultural thesaurus covering farming, food, fisheries, and natural resources. ~40,000 terms (subtree on request)
CANSIS_GLOSSARY	Agriculture and Agri-Food Canada glossary of soil science terms with English definitions and French translations.~830 terms
NAPCSCanada	North American Product Classification System Canada with a hierarchical product and service code taxonomy. 900+ terms
NASIS	USDA NRCS National Soil Information System domain tables for all categorical soil survey fields. 10,000+ terms (~450 enums, 900 pages)
NSDB	Combined Canadian National Soil DataBase soil name and layer classification tables. ~400 terms
NSDBSLC	Canadian National Soil DataBase Soil Landscapes of Canada polygon-level classification data. ~120 terms
NSDBSLT	Canadian National Soil DataBase Soil Layer Table with horizon-level soil property classifications. ~370 terms
NSDBSNT	Canadian National Soil DataBase Soil Name Table with soil series classification and profile data. ~370 terms
Human and animal health	
CODEX	FAO/WHO Codex General Standard for Food Additives listing all internationally permitted additives by food category. ~640 terms
E_NUMBER	EU/UK food additive E numbers with functional-use annotations, organised by numeric range category, sourced from Wikidata.~580 terms
LOINC	HL7 LOINC listing page covering all available clinical observation, lab test, and measurement code systems and value sets. (Term count varies by ValueSet)
LOINCCodeSystem	A single HL7 LOINC clinical terminology code system in FHIR JSON format. (Term count varies by CodeSystem)
LOINCValueSet	A curated HL7 LOINC subset of clinical codes for a specific use case, in FHIR JSON format. (~1,400 typical)
SNOMED CT (API)	Comprehensive clinical health terminology covering diseases, procedures, findings, and anatomy. 350,000+ terms (subtree on request)
General	
CRediT	14-role contributor roles taxonomy for scholarly output attribution, published on Zenodo as a PDF.
FreeText	Claude API extraction of picklist enumerations from any human-readable source: URL, inline text, plain-text file, or PDF; useful when no structured vocabulary file exists.
ISO_COUNTRY	ISO 3166-2 first-level country subdivisions (provinces, states, territories) fetched via Wikidata SPARQL. ~14 terms per country (5,400+ all countries)
LinkML	LinkML YAML schema containing one or more named enumerations with permissible values. (Term count varies by source)
LOC_CLASSIFICATION	Library of Congress subject heading hierarchy spanning all academic disciplines, useful for general topic or domain picklists. ~6,600 terms

Source	Description
OBO terms (API)	Open Biological and Biomedical Ontologies covering environments, gene functions, anatomical structures, and more. Term count varies by ontology.
OWL	Web Ontology Language file defining a class hierarchy with properties and formal logical relationships. Term count varies by ontology.
STATSCAN	Statistics Canada classification variable with hierarchical codes from the IMDB concepts portal. (~20 to 7,000+ per classification)
STATSCAN_TABLE	Statistics Canada Census Dictionary reference table such as province/territory abbreviations. Typically <100 terms.

Term Harvester uses a configuration file to fetch vocabulary on demand on an ongoing basis, allowing for 3rd party vocabulary to be refined, extended, or mapped to normative values. For example, the term_harvester.py command can be used to set up a specific NRCS Soil Field Book source configuration file entry which takes a pdf document URI as input, and returns picklists documented in LinkML format. Source document files are generally downloaded and stored in a zip archive to facilitate provenance and rerunning modified parsing algorithms on a local copy of a source document which would have to be explicitly re-fetched, thus safeguarding against unexpected upstream document changes. (The exception to this behaviour is with API fetching of vocabulary where a configuration lists only root terms of a vocabulary source to retrieve, and a later stage of the build process determines what underlying content is fetched or if it is only fetched during application use).

The general cycle involved in adding a source is:

1. Add sources (auto-detects type, downloads, adds to harvester_config.yaml):
> python3 term_harvester.py -a https://example.org/some-valueset.json
2. Process sources into sources/*.yaml, in accordance with configuration and current parsers:
> python3 term_harvester.py -c
3. Build schema.yaml (syncs enums from all sources):
> python3 term_harvester.py -b
4. Optionally, add SSSOM mappings into the schema.yaml file
> python3 term_harvester.py -s [sssom file name]

2.1. Multilingual content

To the extent that a particular TH source parser has multilingual content, that can be carried through to the schema.yaml file, which is important for jurisdictions with multiple official languages or international outreach. The TH configuration file has a “languages” attribute in which multiple languages can be specified; in the scenario where a source has more than one language to offer content in, such as AGROVOC, it will provide a configuration-specified language if available. As a rule, Term Harvester does not currently provide any Google or other service for translating vocabulary source content as they are imported since this would go beyond the standardised language a source has authored. However, if a Term Harvester administrator needs translations layered into schema.yaml file this can be done by way of the SSSOM inclusion feature described below. Two special functions involving dynamic translation do exist in TH, one for matching the CANSIS_GLOSSARY source French terms to their English counterparts via Google translate, and the other involved in FreeText source work described below.

2.2. Claude AI

Regarding this paper, Claude AI Code was used with some guidance to generate the raw table 1 summary text which has been revised. Claude AI Code was heavily used in programming the Term

Harvester feature set including each term normalisation source module, ushering in an easier ability to extract flat or hierarchic vocabulary from source vocabulary APIs as well as single or multiple pages of possibly multi-lingual web content. A parser for each source - whether by API, pdf, ontology owl file, HTML page, etc. was separated out into its own code module, with shared functions in term_harvester.py and a source_utils.py script. After organizing a variety of extraction approaches for pdf, html, and API vocabulary content, and having created parsers that organize terms hierarchically as reflected in a source, Claude AI can build new parsers that anticipate some issues a given source content type can have. We found the greatest challenge is in parsing pdf documents since the apparently linear (top-down/left-right readable lines and indented bullet list) text can actually be organized non-linearly in pdf source code with coordinates associated with pieces of text ultimately placing them in order visually. Consequently parsing tables of text or page boundaries is challenging and extra logic is required on top of the python pdf parsing module. While Claude AI applies some degree of self-checking, especially with changes in the algorithm such that results of previous and current parses can be compared for discrepancies, ultimately human review is required to determine if vocabulary capture is complete and with correct hierarchy where present on any pdf or html extraction. Most modules have been manually reviewed to ensure they work as intended, but two sources stand out as not being comprehensively reviewed: the STATSCAN source since it has a large list of enumerations which are time-consuming to review with respect to the html pages they originate from; and the NASIS pdf document which has a number irregularly formatted tables (as documented in the TH README.md file) each of which would require special parsing development (Claude is good at this). API-based extraction is reliable to the extent that API sources are online. Note that AI non-determinism isn't an issue for all but the Free Text parser since the other Term Harvester parsing modules, while AI built, don't need/have any connection to Claude or other LLMs to work.

2.3. Free Text parser

The FreeText source merits description separately since it is the one module that depends on Claude AI (requiring an Anthropic API key setup) to extract picklists of terms from designated sections of research papers that are not already registered sources. For longer documents, an anchor piece of text that enables the Term Harvester to extract a 10,000 character window of the document for analysis by Claude is required. As well a fragment of the description or a more complete description of a variable - which can be used as the "description" attribute of the configuration entry for the extracted variable - is needed. If the processed text contains a few variables, it may take some finessing to exclude undesired ones. Users are prompted in the case where multiple variables are found, whether to consider them one hierarchic picklist menu, or break them up into separate selection lists, or consider them a flat list of choices. A FreeText function is provided to supply Claude translator language variants for recognized terms found in a source document, both of which require manual review and approval by a TH data manager on setup of a given FreeText source.

While non-determinism from AI-assisted extraction is possible with the Free Text parser, this extraction is typically just done once per Term Harvester installation in the case of academic papers or other documents which only have one published version (i.e. url) since the source document only needs to be added or fetched once.

An estimation of the percentage of leaf rust damage, as an average of damage to the two upper leaves, was made 10–12 days after heading, visually using a scale similar to the 'USDA' scale discussed by Peterson et al. [24]. The percentage of leaf area occupied by *Septoria* disease was also estimated visually on the two upper leaves without an external scale at milky ripeness. A visual estimation of lodging before harvest was done using a traditional scale of 1–9, similar to that described by Berry et al. [25]. On that scale, 9 means "no lodging", 7 – plants are slightly lodged or stand upright after lodging, 5 – medium degree of lodging, 3 – plants lodged significantly making machine harvesting difficult, 1 – severe lodging making harvesting impossible. Other values denote intermediate states of lodging. The visual estimation of grain is done on a 1–9 scale, with 1 being an unsatisfactory grain appearance and 9 being an excellent grain appearance.

Figure 2: An extract from a pdf document which mentions a few crop phenotypic characteristics [6]. The free-text parsing of a picklist set of choice terms is accomplished by providing a pdf url, and a free_text description of the variable being sought, or at least a textual anchor (exact string match, in this case "A visual estimation of lodging before harvest" or pdf page number) when needed for longer documents. A command line example (in a subfolder of the term harvester code folder) of setting up a new default-named **harvester_config.yaml** file with the lodging text example above:

> **python3 ../term_harvester.py -a "https://bmcplantbiol.biomedcentral.com/counter/pdf/10.1186/s1287**
025-07322-y.pdf#text=A visual estimation of lodging before harvest"

This yields an interactive dialog with progress log showing a newly created harvester_config.yaml, and a prompt to name the picklist or enumeration name (key) to put found options in:

Created **harvester_config.yaml**

Fetching **https://bmcplantbiol.biomedcentral.com/counter/pdf/10.1186/s12870-025- 07322-y.pdf** ...

Anchored to text 'A visual estimation of lodging before harvest' at position 10,999

Text truncated to 10 000 chars (source: 85,619 chars).

Consulting Claude for source key and title ...

Claude extracted 2 enum(s): LodgingScale, GrainAppearanceScale

Suggested source key: 'WheatPhenotypingScales'

Source key [WheatPhenotypingScales]: _

The last line above shows a user being prompted to name the extracted Wheat Phenotyping scale variable. Accepting the default variable name, the interface moves on to showing what terms it has found, and possible hierarchic or multi-picklist form:

Saved source document to sources/WheatPhenotypingScales.zip

Added source 'WheatPhenotypingScales' to harvester_config.yaml

Using downloaded file sources/WheatPhenotypingScales.zip ...

Anchored to text 'A visual estimation of lodging before harvest' at position 10,999

Text truncated to 10 000 chars (source: 85,619 chars).

Extracting 'WheatPhenotypingScales' via Claude ...

Generating labels for intermediate scale values ...

LodgingScale - Lodging Severity Scale (5 stated, 4 intermediate * = 9 total):

1: Severe lodging

2: Between severe and significant lodging *

3: Significant lodging

4: Between significant and medium lodging *

5: Medium lodging

6: Between medium and slight lodging *

7: Slight lodging

8: Between slight and no lodging *

9: No lodging

GrainAppearanceScale - Grain Appearance Scale (2 stated, 7 intermediate * = 9 total):

1: Unsatisfactory

2: Poor *

3: Fair *

4: Between fair and acceptable *

5: Acceptable *

6: Good *

7: Very good *

8: Near excellent *

9: Excellent

Written **temp.md** and **temp.tsv** — review extracted content before confirming.

Recognized potentially 2 enumerations, 18 choices total. Select which enumerations/-choices should appear in output schema:

a) 'Lodging Severity Scale' (9 values)

b) 'Grain Appearance Scale' (9 values)

Include [all by default]: _

It turns out that while the extracted text mentions an input range for lodging of 1 - 9, the names of intermediate values were not specified, which is problematic for user interfaces. Term Harvester uses Claude AI to interpolate names for intermediary values (indicated with * suffix), and the choice of these names is influenced by various types of generic Likert style scales of terms (in TH's sources/synonymy_scale.yaml). Two reports, **temp.md** and **temp.tsv**, are written to illustrate the labels of found picklists and their choices. These can be used to inform the following decisions. First, if document content seems to indicate a hierarchy or separate lists of choices, then a user is prompted about whether to keep each possible top-level picklist (options: "all" or by letter index). Then they are asked how they would like the choices organized, as separate choice lists, as a flat list of choices, or as a single hierarchy of choices.

Now, for the selected enumerations, choose whether to:

[1] Merge - merge all 18 choices into 1 'WheatFieldEvaluationScales' enum and ignore old enum names.

[2] Separate - keep enumerations separate (default)

[3] Hierarchy - one enum 'WheatFieldEvaluationScales' with section headers as terms

(To preview each option, view temp.md or temp.tsv)

Choice [2]: _

Having selected “2”, the user will see end of log:

Saved 2 enum(s) to sources/WheatFieldEvaluationScales.yaml

Search index: 20 terms added for 'WheatFieldEvaluationScales'

At this point a **harvester_config.yaml** (named by default) file has been created, containing an umbrella entry for Wheat Field Evaluation Scales, and with the following entries:

locales:

- en

sources:

WheatPhenotypingScales:

title: Wheat Agronomical Phenotyping Scales

name: WheatPhenotypingScales

version: null

content_type: FreeText

file_format: zip

reachable_from:

source_ontology: https://bmcplantbiol.biomedcentral.com/counter/pdf/10.1186/s12870-025-07322-y.pdf#text=A visual estimation of lodging before harvest

download_date: '2026-07-31'

description: 'Lodging Scale: Nine-point visual scale rating lodging severity from

severe lodging preventing harvest to no lodging. Grain Appearance Scale: Nine-point

visual scale rating grain appearance from unsatisfactory to excellent.'

section_format: 2

Additionally, a **WheatPhenotypingScales.zip** file contains the original pdf document, as well as the extracted 10,000 character section. The top of **sources/WheatFieldEvaluationScales.yaml** (slightly abbreviated) now looks like:

id: https://bmcplantbiol.biomedcentral.com/counter/pdf/10.1186/s12870-025-07322-y.pdf#text=A visual estimation of lodging before harvest

name: WheatFieldEvaluationScales

title: Wheat Agronomical Trait Visual Scoring Scales

in_language: en

imports:

- linkml:types

enums:

LodgingScale:

name: LodgingScale

title: Lodging Severity Scale

description: Nine-point visual scale rating pre-harvest lodging severity from

severe lodging preventing harvest to no lodging.

see_also: https://bmcplantbiol.biomedcentral.com/counter/pdf/10.1186/s12870-025-07322-y.pdf#text=A visual estimation of lodging before harvest

permissible_values:

'1':

title: Severe lodging

description: Severe lodging making harvesting impossible.

'2':

title: Between severe and significant lodging

'3':

title: Significant lodging

description: Plants lodged significantly making machine harvesting difficult.

…

'8':

title: Between slight and no lodging

'9':

title: No lodging

description: No lodging.

GrainAppearanceScale:

…

Lastly, invoking Term Harvester with a -b or –build parameter (in other words "python3 ../term_harvester.py -b") merges the picklist content in **sources/WheatFieldEvaluationScales.yaml** into **schema.yaml** (and creates that file if it doesn't exist). When the build step occurs, a report is returned of the basic action (added, updated, review flag) done for each source, if any.

2.4. Versioning

Term Harvester's approach is somewhat vulnerable to parsing errors or inconsistency / non-determinism over time given that source APIs, ontologies and document content can change, and we don't always know if a version change has essentially occurred. Stability of source vocabulary is assumed in a low-tech way to the extent that some source vocabulary URLs have version numbers in them, which implies that new versions will have new URLs which likely require Term Harvester source python script modification and testing due to format changes. Where version information is not explicit in a source, one can manually set it in the configuration file for a particular source after an –add or –fetch operation. Nine parsers attempt to detect the current version of a source document or uri according to various strategies: by parsing document URL path for a version Id, by presence of version in title of document, or a particular cover page text string (e.g. for CODEX, "CODEX STAN NNN-YYYY" or "CXS NNN-YYYY"), owl:versionInfo or config.versionIri or config.version for OWL ontology files / API access, and in the case of FreeText source, Claude AI seeks a URI path or document title page indication of version. This is experimental territory insofar as any kind of detected or undetected version change opens up the possibility of changes that will break a given parser, necessitating a parser revision. For now TH mainly reflects what reference vocabularies are in current form (the

latest fetch date of each particular source is recorded in the config file); and one usually must resort to a GitHub or backup system to find past versions. The exception to this currently is in the NSDB sources (e.g. "NSDBSNTv2") and NAPCSCanada and CODEX which have version year appended to their configuration keys. However a design solution is required to meet the desire to have picklists have constant names across source version changes such that downstream applications and databases that reference them do not have to have their references changed. As it stands technical data managers must make deliberate choices about updating particular sources in their schema.yaml files.

We aim to have Term Harvester engender an open source module development community that can contribute or refine source vocabulary parsers, and in this way keep pace with new versions of vocabularies as they are published. The speed and ease with which Claude AI-analyzed patches can be developed, as well as a differential view script available for comparing Term Harvester's output schema.yaml across Git commits, should help catch inconsistencies (see README_diff_schemas.md and diff_schemas.py in https://github.com/cidgoh/DataHarmonizer/tree/master/script).

2.5. SSSOM mapping

To complete the FAIR principle harmonisation of terms in a Term Harvester schema.yaml file requires adding a semantic identifier to each picklist or choice (in LinkML terms, an **enum_uri** or **meaning** URI attribute for each enumeration and each of its permissible values). Some LinkML source files already have ontology term identifiers associated with non-ontology controlled vocabulary terms, but for many other Term Harvester source types the semantic identifiers need to added, and we accomplish this with a Term Harvester **-s** or **–sssom** parameter which uses the Simple Standard for Sharing Ontological Mappings (SSSOM) file standard for adding human (or machine) curated mappings to schema.yaml file terms [7]. It accepts an input file with the standard SSSOM SKOS exact, close, broad, narrow and related matches since LinkML has corresponding annotations (e.g. "exact_mappings") for these attributes. Rather than having the schema.yaml file be the focus of semantic annotation, our recommendation is to provide one preferred semantic mapping per utilized reference vocabulary term, and then focus semantic annotation effort on the targeted ontology terms. TH doesn't provide any intelligence here in terms of automating such a SSSOM mapping. Future work could involve ontologically modelling domains of interest, and then applying AI to map from reference vocabularies in schema.yaml onto such model parts.

2.6. Demonstration implementation

To illustrate how source vocabularies can be used by down-stream applications, we illustrate this use-case with our ongoing Climate Smart Data Collaboration Centre (CS-DCC) project, a Genome Canada (GC) funded effort to provide research grade cataloguing and data exploration tools for nine GC funded agricultural research projects [8]. The Term Harvester script has a configuration file specific to the CS-DCC project which generates the schema.yaml file. A second **conversion.py** python script then provides a blueprint for converting such vocabulary to a project or agency specific variable library (say for example my_menus.json in Fig. 3) which often requires localized variable names, titles, descriptions, and picklist choices to be isolated and preserved for existing data description (Fig. 3). Two other inputs to conversion.py are a SSSOM table which handles addition of language translated text in my_menus.json, and a differential rule file (Fig. 3 "mappings.yaml") which handles adaptation of schema.yaml content to match my_menus.json ultimate picklists. We have found that with some guidance about SSSOM and the functionality for mappings.yaml, Claude AI was able to build conversion.py from scratch.

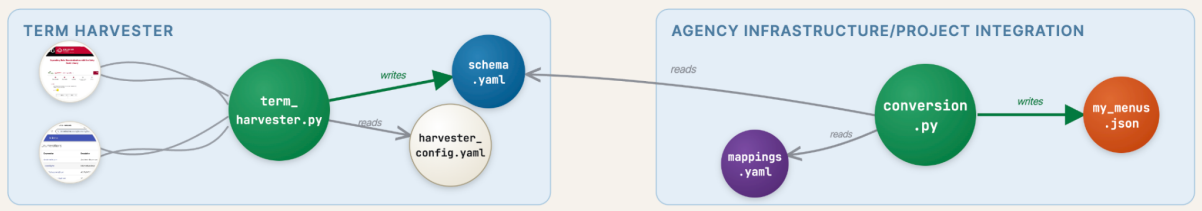

Figure 3: A two step process to convert standardised vocabulary to a local variable schema

Existing tools for tabular data management of datasets can readily make use of Term Harvester products: schema.yaml is in a format that DataHarmonizer, a browser-based tabular editor can use directly; and the Semantic Engine offers users a library of AgrifoodDataCanada picklists which Term Harvester will manage [9,10].

3. Conclusion

Managing digital documentation about research data variables often involves a provisional approach to locating sources of study design, protocol and sample metadata terms - but this is often an inconsistent manual curation step depending on personal knowledge rather than a process of selecting terms from a trusted library of vocabulary sources. Term Harvester acts as a research platform infrastructure bridge between existing and emerging scientific vocabularies of various formats and sizes, and local data schemas which agencies or research projects have often developed as stand-alone resources. Researchers still have control over preferred vocabulary sources and selected terms, but Term Harvester's standardised output simplifies downstream data processing, sharing, and term management.

Availability: Term Harvester is available as open-source software on GitHub at https://github.com/ClimateSmartAgCollab/term-harvester. It runs independently of Claude AI except for some vocabulary text-mining functionality used in the FreeText source parser which depends on an anthropic API key, and additionally a search of configured vocabularies can be enhanced by Claude generated synonymy.

Acknowledgments

The authors would like to acknowledge support from Genome Canada's Climate-Smart Data Collaboration Centre project 322DCC under Genome Canada's Climate-Smart Agriculture and Food Systems initiative.

Declaration on Generative AI

During the preparation of this work, the author(s) used AI (Claude on Table 1) for: Formatting Assistance, Drafting Content. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] FAIR principles, GO FAIR (2017). https://www.go-fair.org/fair-principles/ (accessed November 3, 2022).

[2] C. Mungall, LinkML - Linked data Modeling Language - LinkML - Linked data Modeling Language, (n.d.). https://linkml.io/ (accessed April 5, 2023).

[3] R.C. Jackson, N. Matentzoglu, J.A. Overton, R. Vita, J.P. Balhoff, P.L. Buttigieg, S. Carbon, M. Courtot, A.D. Diehl, D. Dooley, W. Duncan, N.L. Harris, M.A. Haendel, S.E. Lewis, D.A. Natale, D. Osumi-Sutherland, A. Ruttenberg, L.M. Schriml, B. Smith, C.J. Stoeckert, N.A.

Vasilevsky, R.L. Walls, J. Zheng, C.J. Mungall, B. Peters, OBO Foundry in 2021: Operationalizing Open Data Principles to Evaluate Ontologies, bioRxiv (2021) 2021.06.01.446587. https://doi.org/10.1101/2021.06.01.446587.

[4] AGROVOC, (n.d.). https://agrovoc.fao.org/ (accessed June 23, 2026).

[5] NRCS Soil Field Book, 2024. https://www.nrcs.usda.gov/resources/guides-and-instructions/field-book-for-describing-and-sampling-soils (accessed June 23, 2026).

[6] M. Bazhenov, L. Bespalova, M. Samarina, N. Polevikova, E. Agaeva, A. Debeliy, A. Beliakova, A. Ponisko, L. Nazarova, A. Chernook, G. Karlov, M. Divashuk, Genome-wide association study of agronomic traits in winter wheat (Triticum aestivum L.) using a custom SNP marker set, BMC Plant Biol. 25 (2025) 1279.

[7] N. Matentzoglu, J.P. Balhoff, S.M. Bello, C. Bizon, M. Brush, T.J. Callahan, C.G. Chute, W.D. Duncan, C.T. Evelo, D. Gabriel, J. Graybeal, A. Gray, B.M. Gyori, M. Haendel, H. Harmse, N.L. Harris, I. Harrow, H. Hegde, A.L. Hoyt, C.T. Hoyt, D. Jiao, E. Jiménez-Ruiz, S. Jupp, H. Kim, S. Koehler, T. Liener, Q. Long, J. Malone, J.A. McLaughlin, J.A. McMurry, S. Moxon, M.C. Munoz-Torres, D. Osumi-Sutherland, J.A. Overton, B. Peters, T. Putman, N. Queralt-Rosinach, K. Shefchek, H. Solbrig, A. Thessen, T. Tudorache, N. Vasilevsky, A.H. Wagner, C.J. Mungall, A Simple Standard for Sharing Ontological Mappings (SSSOM), arXiv [cs.DB] (2021). http://arxiv.org/abs/2112.07051.

[8] CS-DCC, Climate-Smart Agri-Food Network (2025). https://climatesmartagrifood.ca/cs-dcc-climate-smart-data-collaboration-centre/ (accessed June 23, 2026).

[9] I.S. Gill, E.J. Griffiths, D. Dooley, R. Cameron, S. Savić Kallesøe, N.S. John, A. Sehar, G. Gosal, D. Alexander, M. Chapel, M.A. Croxen, B. Delisle, R. Di Tullio, D. Gaston, A. Duggan, J.L. Guthrie, M. Horsman, E. Joshi, L. Kearny, N. Knox, L. Lau, J.J. LeBlanc, V. Li, P. Lyons, K. MacKenzie, A.G. McArthur, E.M. Panousis, J. Palmer, N. Prystajecky, K.N. Smith, J. Tanner, C. Townend, A. Tyler, G. Van Domselaar, W.W.L. Hsiao, The DataHarmonizer: a tool for faster data harmonization, validation, aggregation and analysis of pathogen genomics contextual information, Microb. Genom. 9 (2023). https://doi.org/10.1099/mgen.0.000908.

[10] AgriFoodDataCA picklists, (2026). https://github.com/agrifooddatacanada/picklists_for_schemas/tree/main/picklists (accessed June 23, 2026).