Data to Ontology Mapping Tools

Release 1.0

AIOTI WG Standardisation
Focus Group on Semantic Interoperability

30 September 2025

Executive Summary

A key challenge people face when using ontologies is how to map existing data to the concepts and properties defined by the ontologies. Mapping tools support this process, but there are quite a number of such tools available that differ in aspects such as expressiveness, required skills, and scalability. The purpose of this document is to give an overview of the different mapping tools available, classifying them into graphical user interface-based tools, configuration languages and programming frameworks. The working of each tool is summarised, with relevant links being provided for further details. Furthermore, each tool is scored on a scale of 1 to 5 according to important aspects: expressivity, maintainability, required programming expertise, required RDF expertise, user-friendliness, GUI-support, availability, and scalability, including an explanation for each score. As LLM-based approaches are becoming more and more interesting, an outlook on several example LLM-based approaches is given. Finally, a summary of the mapping tools is provided, sketching the inherent trade-off between expressiveness and (non-expert) user friendliness.

Table of Content

Table of Figures

1. Objectives

During 2023 the Alliance for AI, IoT and Edge Continuum Innovation (**AIOTI**) has performed an ontology gap analysis among ontology users in standardisation, (applied) research, and industry. The survey, among others, asked the participants to rank what they thought would be the main point to be improved among 1) data to ontology mapping, 2) example data sets, 3) visualization, 4) browsing, and 5) success stories.

The outcome was clear: improvements in the information on ontology mapping was primarily requested. This document takes a major step in providing that information by indicating to the reader where to find additional information on various mapping tools, by evaluating and categorising each tool based on our expertise in ontology mapping, and by making a recommendation on the tool a user would need based on their situation.

The main goal of this document is to direct users to the mapping technology that best fits their use case and their expertise. We order and categorise the various mapping tools based on the ontology expertise a user has, their programming expertise, and the expressiveness that the data transformation requires. Therefore, instead of simply listing available tools, this document provides structured guidelines to help users select the most suitable tool for their needs. This document presents an expert assessment based on practical experience and tool evaluation, rather than a theoretical literature review.

2. Why do we do mappings?

The focus of this paper is on mapping data to ontologies. Many definitions exist for what an ontology is. For the purposes of this document, the following is most instructive:

> *"An ontology defines (specifies) the concepts, relationships, and other distinctions that are relevant for modelling a domain." (Tom Gruber, in the Encyclopedia of Database Systems, Springer, 2009)*

Ontologies are typically intended to be application-independent, whereas most data models are application-specific, designed to function in the back-end of a particular application. As such, because the model would only be used in one particular situation for one application, there are less strict requirements on the modelling efforts in developing data models. Spending additional time on refining a data model structure may not always be efficient, depending on the specific use case.

The field of ontologies claims that additional effort put into creating an intuitive, reusable and modular data model will pay back its effort, because the ontology model can be reused among several applications, use cases, and, potentially, domains. However, if participants want to use these benefits, they have to adopt the ontology by expressing their data in terms of the ontology, if they want to share the data with other participants in the same ecosystem.

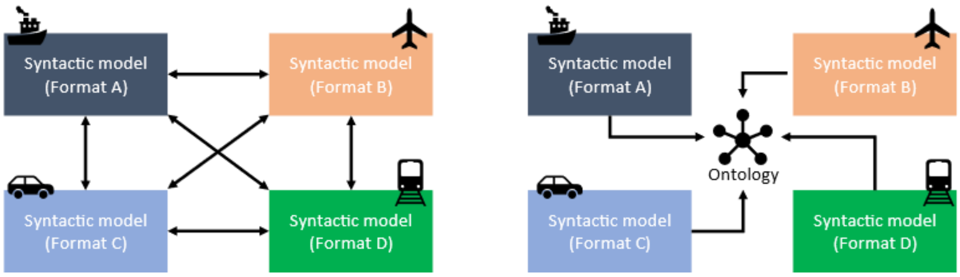


Figure 1 Without a shared semantic model compared to the situation with a shared semantic model

Figure 1 shows the difference in complexity between using a shared ontology versus not using a shared ontology. Without an ontology all syntactical models potentially have to be mapped to each other, resulting in a potential n*(n-1) mappings if all models have to be mapped to each other. Each owner of a model should potentially map it to all other models, each time starting over the process. When a shared model is available, all data owners only have to implement a single mapping: to the shared ontology model. The total number of mappings will be n, with each partner implementing a single mapping.

2.1 Why is it a bottleneck?

Implementing the arrows of Figure 1 is performed via the technical task of mapping a proprietary data format to the shared ontology format. This task is perceived as a bottleneck in the adoption of ontologies for various reasons:

- The task requires a range of experiences and expertise in various areas: the proprietary format, the ontology, and the mapping technique. Also, we may some experience and expertise related to the application domain in which the ontology will be applied.

However:

1. The documentation of some ontologies may be incomplete or difficult to access, making it harder for users to implement mappings correctly.

2. The proprietary data format may be different from the shared ontology, resulting in extensive conceptual modelling-based harmonisation effort.

- Users often struggle to identify the most suitable ontology tools for their needs, even though several repositories exist, such as OntoHub, BioPortal, LoV, Ontobee, and the OLS.

3. Some mapping tools may lack intuitive user interfaces, comprehensive documentation, or automation features found in professional-grade software.

4. The users don't know which available tool has the best fit to their expertise and use case.

This document tries to guide the user towards clarity, additional information, and suggestions on this. So, currently, it primarily tackles 1, 4, and 6 directly, and 3 and 5 indirectly.

A related problem is the matching of multiple extant ontologies to each other. In various domains there have grown a few ontologies or taxonomies to the level of semi-standard, requiring a harmonization approach between those. Although a highly relevant topic, it's outside of the scope of this white paper. Instead, we consider a hierarchical difference between the source data and the target model. The source usually is only used by one or at most a handful of users, whereas the target usually is a standardised (semantic) model prescribed by some authority for a group of subjects collaborating in a particular environment. These subjects may be organisations operating in the same domain, but may also be a set departments constituting a company, in which case we often talk about an Enterprise Knowledge Graph (EKG).

3. What is a mapping?

Ontology mapping is a task during which a custom data source format and a shared ontology are linked to each other. It requires the following:

- Ontology specification
- Proprietary data model specification
- Tool or technology to implement the mapping
- (Optional) Example data of ontology and proprietary data model

It usually requires at least the following roles, which may be performed by the same person, but often is not:

- Ontology specialist
- Proprietary data model specialist
- Mapping tool specialist
- Software development specialist

It should result in the following deliverables:

- Implementation of the mapping in chosen tool or technology
- Documentation describing the mapping

A system which configures how a database relates to a common ontology. Mapping requires making several key decisions:

- One-way versus two-way: does the mapping implement the direction of data model to ontology only, or should it also implement the reverse direction?
- Virtualization versus materialization: is the mapping performed at query time or is the ontology data constantly available?
- Should it be semantics-preserving or is an approximation acceptable?
- Does the mapping rely on an intermediate layer, or does it apply direct transformations to the data?1
- Does it operate under open- or closed-world semantics, or does it aim to link closed-world semantics of data stores to open-world semantics of ontologies?

There are some related tasks that we don't consider. For example, ontology alignment usually is the alignment of two ontologies that cover (partly) overlapping domains. We don't consider that task in this document. It's solely focused on the integration of a proprietary database in a shared ecosystem by mapping it to a (set of) ontologies that are expected to be used in the ecosystem.

[1] Two principal approaches have been defined. The mapping-based approach was first described comprehensively in the "Linking data to ontologies" paper by Poggi et al. (2008), in Journal on Data Semantics. The principal approaches, a transformation-based approach, and a comparison was first described in "KnowID: An architecture for efficient Knowledge-driven Information and Data access" by Fillottrani and Keet (2020) in Data Intelligence.

3.1 Categorisation

We identify three main categories of mapping tools/technologies:

1. Graphical user Interface based tools. These constitute the tools that require the least of both programming and ontology experience from the user, so are the best fit for domain experts. These are probably also the easiest to maintain when the responsibility for a mapping is shifted. On the other hand, this tool has less expressivity and flexibility than the others, so not all sophisticated data transformation may be possible.

2. Configuration languages: these tools are a middle way between programming experience and flexibility. They do not require full software development skills, but users need to write simple scripts instead of relying on a graphical user interface.

3. Programming libraries: These tools require a user to have software development skills on top of other expertise required. This gives a high threshold for usability and maintainability. However, on the other hand, it gives unlimited flexibility and expressivity to the user.

We additionally include a section on LLM based Generative AI (GenAI) technologies to support the data transformation. These technologies can be part of the workflow in any of the three above mentioned categories with several options of human involvement. An LLM component can, for example, be used to automate some core steps that can be performed manually by configuration languages or programming libraries, such as generating mapping rules and creating knowledge graphs. An additional example is to use an LLM component as a recommendation engine to suggest the probably ontology properties to map the data to.

A key challenge of AI-based ontology mapping is reproducibility, because prompting an LLM for the same mapping would return a different mapping for most instance. The same input may yield different mappings depending on prompt variations or model updates. This drawback is inherent to the technology underlying LLMs. The level of accuracy required by a particular use case should therefore be in line with the consistency provided by the LLM component.

There are additional criteria on which we can compare these tools, including but not limited to the following.

Web services. Sometimes web services are provided that spare the user from having to set up any software, and do not really require complex development skills, which is less time-consuming. The best cases are when web services can be executed by reading an ontology file or from an ontology URI and can provide an answer in the GUI (e.g., a website with an HTML GUI)

There are several aspects along which we can compare the mapping tools and technologies. In this document, we use the aspects described below. We assign for each feature a score on a 5-point Likert scale (i.e., between 1 and 5 (inclusive)) to all mapping tools. This exercise inevitably has a subjective component, so we give a short explanation for each scored aspect per technology.

Aspect	Description
Expressivity	The level of expressivity indicates the complexity of mappings you are able to construct with a particular technology. Simple mappings may require only the translation from csv-header to ontology property without changes to the structure, whereas complex mappings may require restructuring the data. A score of 1 indicates that you can do little more than aligning properties, whereas a score of 5 indicates that any complex heterogeneous mapping or structural transformation is feasible.
Maintainability	The mapping pipeline performance is dependent on both the input data as well as the ontology model the output is expected to follow. Changes in either the input data or ontology model often require adjustments in the mapping pipeline. A score of 1 indicates that it is very difficult for a user to maintain the mapping someone else has constructed, whereas a score of 5 indicates that it should be easy to continue another user's work.
Programming expertise required	The programming expertise required can vary from none to requiring professional software development expertise. A score of 1 indicates that usage of the technology would not require software development expertise, whereas a score of 5 indicates that professional software development expertise is requires to successfully apply it.
Semantic technology expertise required	The mapping exercise inevitable requires familiarity with the ontology to map to. This element describes the extent to which a user would additionally need expertise in semantic technology, such as RDF, OWL, SPARQL and related languages. A score of 1 indicates no specific RDF knowledge, since the tool would hide its complexities. A score of 5 indicates that a user would need to have expert RDF knowledge to leverage it properly.
User-friendliness & documentation	The ease for users to get onboard with a tool varies a lot depending on how much effort the developers have (been able to) put into the documentation. A score of 1 on this feature indicates that it's research-ware that still requires a lot of guidance from the creators. A score of 5 indicates that it's software that can smoothly run on its own with extensive documentation.
Editor support	Some tools provide extensive editors to work with. Clearly, a GUI is an editor, but also some of the mapping languages have tools to facilitate their workings. A score of 1 would indicate that there's no editor to go along with the tool, in that it may remain a specification or command line facility. A score of 5 would indicate that there is a mature editor to aid in development.
Availability and user support	Some tools are developed by companies that in parallel offer user support to the tool, whereas other tools are maintained by a community. A score of 1 indicates that there's very little in the way of a community or group where user support could come from. A score of 5 indicates a lively community group or a company that is easily reachable, although maybe for a fee.
Scalability	Another challenge is the scalability of a specific tool or technology. This concerns the question of whether the pipeline is sufficiently efficient for the transformation of large amounts of data. An additional concern is whether the pipeline can be further optimised independent of the transformation specification.

3.2 Other approaches

A more direct integration of the ontology model within the data framework is the usage of an Object-Relational Mapping (ORM) framework or library. We consider this a variant of the programming way, as it helps marshal or convert data transfer object (DTO) models between two representations: data as persisted in a database or file on disk, and data as objects in computer memory for processing. They are often tied to the programming language in use. We consider this an alternative approach that could complement existing ontology mapping techniques, when custom processing logic rather than a mapping language is preferred.

The following are some of these object-triple-mapping (OTM) projects that appear to have some level of public use to date:

- TrinityRDF (dotNET): GitHub - semiodesk/trinity-rdf: TrinityRDF
- KOMMA (Java): KOMMA - RDF Object Mapper for Java
- Spira (Ruby): Github - ruby-rdf/spira: Spira
- Owlready2 (Python): Owlready2 0.47 documentation

Of these, Python probably has the least mature ecosystem for RDF-object mapping, as Owlready2 focuses on loading an ontology into objects, and not necessarily the alignment of DTO classes with RDF triple statements (which is usually with an annotation mechanism on class attributes).

Some additional promising but possibly out-of-date alternative exist:

- Empire (JPA-Hibernate-ish interface to RDF databases using SPARQL): GitHub - mhgrove/Empire: JPA implementation for RDF
- ORM-ish: Ontologies as python objects, for example Object Description Mapper — Open Knowledge Foundation RDF v0.35 documentation
- RDFAlchemy: GitHub - openvest/RDFAlchemy: RDFAlchemy is an abstraction layer, allowing Python developers to use familiar dot notation to access and update an RDF triplestore.
- Never Mind the Semantic Gap: Modular, Lazy and Safe Loading of RDF Data (paper_149_Kamburjan_et_al.pdf (eswc-conferences.org))

There are also alternative techniques which supplant the need for such a framework, given that most programming languages have the least friction in transforming to and from JSON:

- JSON-LD 1.1 Framing, which allows results of SPARQL CONSTRUCT to be fitted to a desired model: JSON-LD 1.1 Framing
- Alternative RDF- and SPARQL-compatible graph databases that treat JSON as a native data format, e.g. Fluree

3.3 Scientific community

We expect the state of the art to be published in various scientific workshops and journals, such as the SemTab challenge on extracting semantic knowledge from tabular data and the Ontology Alignment Evaluation Initiative, that investigates ontology to ontology mappings. This work is also indebted to many discussions the contributors had at various instances of the SEMANTiCS conference, ISWC, and ESWC.

4. Tool overview

4.1 Graphical User Interface based tools

In this section, we describe graphical user interface based tools. These tools provide a user interface that facilitates the user to effectively construct and, in some cases, validate a mapping. These tools generally score high on user-friendliness and ease of use, but the GUI often means that it's impossible to create highly complicated mappings in the tool. Many tools supply an export functionality of the created mapping to an existing language, which may be used to tweak the mapping or add a more complicated mapping if the source data requires so.

4.1.1 Open Refine with RDF Extension

OpenRefine with RDF Extension	
Contact information:	
Organization:	*Open Refine community*
Contact person(s)	*Antonin Delpeuch, pintoch@mamot.fr (OpenRefine)* *Fadi Maali, Richard Cyganiak (RDF Extension)*
Website	*https://openrefine.org/*
Community	*https://openrefine.org/community*
Sources:	
Formal	*https://github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Developers*
Git	Open Refine: https://github.com/OpenRefine/OpenRefine/graphs/contributors RDF extension: GitHub - stkenny/grefine-rdf-extension: An extension to OpenRefine that enables graphical mapping of OpenRefine project data to an RDF skeleton and then exporting it in RDF format
Tutorials:	*https://openrefine.org/docs*
Papers:	*N/A*
Content:	
Abstract	"OpenRefine is a powerful free, open source tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data." *RDF Extension for OpenRefine* "This project adds a graphical user interface (GUI) for exporting data of OpenRefine projects as interlinked RDF data. Data can be reconciled against any SPARQL endpoint or RDF dump. The reconciled data can then be exported as RDF based on a template graph. These all are supported by a friendly graphical interface. Features: • Reconciliation and interlinking • Reconcile against any SPARQL endpoint or RDF dump file • Search the web for related RDF data using Sindice • RDF Export • Graphical interface for defining the mapping • Use of arbitrary vocabularies/ Ontologies • Import existing vocabularies/ Ontologies • Autocomplete for property and class names • RDF/XML and Turtle export

	• Namespace suggestions from prefix.cc
Related tools	"The following input formats are supported: TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents. Support for other formats can be added with OpenRefine extensions." "The following export formats are supported (with RDF extension) TSV, CSV, HTML Table, Excel (.xls and .xlsx), ODF spreadsheet, RDF as turtle, RDF as XML, and Google Sheets. [i.e. the result of the conversion can be exported, not the mapping/translation itself]."
Categorisation	
Category	*GUI*
Scoring	
Expressivity	*3 – many input and output formats are supported. There is limited functionality beyond simple mappings, e.g. the id can be derived from a single element.*
Maintainability	*2 – there is no export for the mapping as such into an independent format, however the project can be saved. Only the translated result can be exported in a number of formats.*
Programming expertise	2 – no real programming, but installation and configuration require know-how
RDF expertise	2 – representation as table, user needs to know about ontology concepts and put the respective mapping
User-friendliness	3 – the GUI provides simple, but straightforward support for creating the mapping.
GUI-support	3 – simple, bit straightforward visualization and editing support
Availability	3 – open source, available with documentation
Scalability	4 – The exported mapping definition can separately be used as a configuration language.

4.1.2 Linked Data Wizard

Linked Data Wizard	
Contact information:	
Organization:	Netwerk Digital Erfgoed (Dutch Network Digital Heritage)
Contact person(s)	GitHub - pldn/LDWizard: 🧙 LDWizard
Website	LDWizard (netwerkdigitaalerfgoed.nl)
Community	Involvement of various Dutch Linked Data communities, namely NDE, Kadaster, ISSG, Triply.
Sources	
Formal	N/A
Git	GitHub - pldn/LDWizard: 🧙 LDWizard
Tutorials:	Home · pldn/LDWizard Wiki · GitHub
Papers:	N/A
Content:	
Abstract	Enthusiasts have been able to publish linked data. At the same time, there is a much wider group of domain experts and data owners that want to experiment with linked data for the first time. This group of users is currently inhibited by the many technical hurdles that are imposed by traditional linked data publication approaches. This is where **LD Wizard** comes in. LD Wizard is a framework for creating end-user focused Graphical User Interfaces (GUIs) that simplify the creation and publication of linked data. LD Wizard allows domain experts and data owners

	to publish standards-compliant linked datasets, without having to worry about technical details and linked data-specific intricacies. The purpose of LD Wizard is to transform simple tabular data files into standards-compliant linked data. LD Wizard specifically focuses on the beginning user: it does not presuppose any prior knowledge of existing vocabularies or linked data technologies. At the same time, LD Wizard allows the standardized source data and the transformation configuration to be exported in commonly used formats. These exports can subsequently be used in more advanced tools in order to achieve more complex linked data transformations." – source, Github wiki
Related tools	Import functionality for csv and tabular data Export functionality for RML
Categorization	
Category	*GUI*
Scoring	
Expressivity	*2 – only csv and tabular data are supported as input, RML can be generated as output. Additionally, the complexity of the mappings is constrained by the options the interface gives.*
Maintainability	*4 – The resulting mapping can be reloaded and updated. Additionally, the configuration as specified in the tool can be saved and refined further.*
Programming expertise	*2 – There is no real software development expertise required, but installation and configuration may require know-how since the installation instructions may not be fully complete or accurate for each environment.*
RDF expertise	*2 – The user needs to know basic information about ontology concepts to decide which mapping to implement.*
User-friendliness	*2 – The tool was originally intended for the Dutch heritage sector so it may still have some artefacts from the initial audience, but has since been extended in language as well as domain.*
GUI-support	*2 – There is a simple online webpage where the user can implement the mapping. The GUI supports basic but fundamental tooling that is required.*
Availability	*3 – available with examples and configuration description (not fully complete/accurate). The tool is additionally available as an open-source tool. Incidental development happens via a bounty procedure.*
Scalability	*5 – The defined mapping can be exported as RML, so the scalability of the LD Wizard should be the same as the scalability of RML.*

4.1.3 Semantic Treehouse Wizard

Semantic Treehouse	
Contact information:	
Organization:	TNO
Contact person(s)	linda.oosterheert@tno.nl, michiel.stornebrink@tno.nl
Website	www.semantic-treehouse.nl
Community	~1000 unique business users per month of the application as a whole. Discord: https://discord.com/invite/kdrbm9RUu8
Sources:	
Formal	N/A
Git	https://gitlab.com/semantic-treehouse
Tutorials:	https://www.semantic-treehouse.nl/docs/wizard/
Papers:	Paper: https://www.semantic-treehouse.nl/assets/files/04_TNO-april-2022-The-Vocabulary-Hub-to-configure-data-space-connectors-b0c0ba64f3f915894003c331de9cca50.pdf Context: https://www.semantic-treehouse.nl/blog/vocabulary-hub-for-dataspaces/
Content:	
Abstract	The wizard is part of the Vocabulary Hub and provides users with an easier way to create schemas or even API specifications that are aligned to standardized ontologies. It's a type of wizard because it provides the user with a sequence of steps in their web browser where they: • identify and select any appropriate ontology for their use case, e.g., SAREF or OEO; (It can be more than one.) • create a data model of their dataset in the Vocabulary Hub by selecting all the classes/properties from the ontology selected at step 1; • let the Vocabulary Hub generate a schema or API specification for the data model they created in step 2. At the time of this writing, the Energy Vocabulary Hub supports XML Schema, JSON Schema, RDF/SHACL and Open API Specification (OAS). Next to a schema or API specification, the wizard also generates in step 3 an RDF Mapping Language (RML) specification that anyone could use to transform any data that conforms to this data model into a knowledge graph with links to the original ontology selected in step 1.
Related tools	**Import functionality for:** JSON schema, SHACL, RDFS/OWL **Export functionality for:** RML, XML schema, OAS, SHACL, JSON schema
Categorisation	
Category	*GUI*
Scoring	
Expressivity	*3 – The expressivity of the mapping is limited to what the Semantic Treehouse allows the user to configure. This has a sufficient basis of mappings, but lacks some complicated transformations that may be needed.*
Maintainability	*4 – The mappings are saved in the tool and can be further reconfigured, before an update is exported. So, mappings are easily maintainable.*

Programming expertise	*2 – The tool provides an extensive user interface, so there is very little software development expertise needed to successfully utilise it.*
RDF expertise	*3 – There is little RDF expertise needed, because the semantic models can be imported into Semantic Treehouse, which makes the contents available to the user. This means they have no need to read the RDF serialisation.*
User-friendliness	*4 – There is at the moment a TNO team that can provide support, although at a cost. Additionally, there is extensive documentation and tutorials available.*
GUI-support	*5 – The GUI of Semantic Treehouse has recently been upgraded and refined for newly identified user requirements. I*
Availability	4 – The STH project is available as open source and actively being maintained in a number of TNO projects.
Scalability	5 – The STH tool can export the mapping to RML, which is highly scalable, so, by extension, the STH mappings also are highly scalable.

4.1.4 Mapping WorkBench (MWB)

Mapping WorkBench	
Contact information:	
Organization:	*Meaningfy*
Contact person(s)	*Eugeniu Costetchi (hi@meaningfy.com)*
Website	[Meaningfy]()
Community	*Commercial company*
Sources:	
Formal	*N/A*
Git	[Meaningfy · GitHub]()
Tutorials:	[Mapping Workbench New – Meaningfy]()
Papers:	*Mapping Workbench: A collaborative platform for mapping complex XML data to RDF*
Content:	
Abstract	Mapping Workbench (MWB) provides a user-friendly graphical user interface (GUI) and powerful functionalities for developing and testing RML mapping rules, managing complexity of large data structures, and collaborative editing and validating with domain experts, in the manner of an integrated development environment (IDE). By adopting a test-driven approach for knowledge engineering and incorporating the notion of Conceptual Mapping (CM), MWB allows for evaluating the correctness of both the mapping rules and the transformed data against OWL ontologies, SHACL data shapes and semi-automatically generated validation rules derived from the CM. Features: 1. Project Management: create and manage projects, providing a structured environment for mapping tasks. 2. Data Selection: select specific XML data files to be mapped within their projects. 3. Ontology Integration: choose the ontology in which the data will be mapped, ensuring alignment with existing semantic models. 4. Conceptual Mapping: create conceptual mappings for each package, enabling validation with business stakeholders. 5. RML Mapping Rules: MWB supports the specification of RML mapping rules, which define how XML data elements are mapped to ontology fragments. 6. Translation of Data: MWB automatically translates selected XML data into fragments of the chosen ontology based on the specified RML mapping rules.

	7. Quality assessment through automatically generated SPARQL queries, SHACL Validation, and mapping coverage assessment with XPath or JSONPath. 8. Controlled Lists (Resources): Users can define controlled lists for each package to assist RML mapping rules in accurate data mapping.
Related tools	*Input data format: XML, JSON* *Output: [RML](RML) (mapping package with transformation rules, value mapping in CSV/JSON), tests (SPARQL, SHACL), sample output data (RDF) and quality assessment reports (JSON, HTML, CSV))*
Categorisation	
Category	*GUI*
Scoring	
Expressivity	*3 - The expressivity of the mapping is limited to what the Mapping Workbench allows the user to configure. This has a sufficient basis of mappings, but lacks some complicated transformations that may be needed. The MWB provides full support of RML (with XQuery 3.1 and XPath 3.1 and JSONPath), so the expressivity is limited to the expressivity of RML instead of additional limitations because of the tool design.*
Maintainability	*4 – The mappings are saved in the tool and can be further reconfigured, before an update is exported. So, mappings are easily maintainable.*
Programming expertise	*2 – The tool provides an extensive user interface, so there is very little software development expertise needed to successfully utilise it.*
RDF expertise	*4 – There is little RDF expertise needed, because the semantic models can be imported into the Mapping Workbench, which makes the contents available to the user. This means they have no need to read the RDF serialisation. Additionally, there are various services to validate correctness of the mappings.*
User-friendliness	*4 – There is extensive documentation and tutorials available.*
GUI-support	*5 – The Mapping Workbench contains an extensive GUI to support the data mapper in their tasks.*
Availability	2 – The Mapping Workbench is available as a commercial tool provided by Meaningfy, including extensive support. A version is publicly available as open source.
Scalability	5 – The MWB tool can export the mapping to RML, which is highly scalable, so, by extension, the MWB mappings also are highly scalable.

4.1.5 Ontotext Refine

Ontotext Refine	
Contact information:	
Organization:	Ontotext
Contact person(s)	https://www.ontotext.com/contact/
Website	https://www.ontotext.com/products/ontotext-refine/
Community	Commercial Company
Sources:	
Formal	*N/A*
Git	*N/A, closed source*
Tutorials:	https://platform.ontotext.com/ontorefine/
Papers:	*N/A*
Content:	
Abstract	*"Ontotext Refine (OntoRefine) is a free application for automating the conversion of messy string data into a knowledge graph.* *The tool supports analyzing the input data, applies various data cleaning and transformation algorithms, maps the string values into knowledge graph concepts and imports the generated model into GraphDB. The application is based on the popular OpenRefine tool and includes a visual RDF mapping interface, command line tool to automate all steps and integration with secured GraphDB endpoints."* *Source:* Ontotext Refine
Related tools	*Input data format: TSV, CSV, *SV, XLS, XLSX, JSON, XML*
Categorisation	
Category	*GUI*
Scoring	
Expressivity	*3 – The expressivity of the mapping is limited to what OntoRefine allows the user to configure. This has a sufficient basis of mappings but lacks some complicated transformations that may be needed.*
Maintainability	*4 – The mappings are saved in the tool and can be further reconfigured, before an update is exported. So, mappings are easily maintainable.*
Programming expertise	*2 – The tool provides an extensive user interface, so there is very little software development expertise needed to successfully utilise it.*
RDF expertise	*3 – There is some RDF expertise needed, because the user needs to decide on which RDF terms to map the data to. For this end, they need to be experienced about the underlying data model.*
User-friendliness	*4 – There is extensive documentation and tutorials provided by the Ontotext company.*
GUI-support	*5 – There is a GUI available with several useful functionalities in the GraphDB framework, including the option to upload the ontology you want to map to.*
Availability	4 – The OntoRefine project is open source available in a limited GraphDB implementation. OntoText provides commercial support.
Scalability	4 – The OntoRefine mappings can separately be exported, which may then be used as input for one of the configuration mapping language interpreters, which can separately be optimised.

4.1.6 Expressive RDF Mapper (XRM)

Expressive RDF Mapper	
Contact information:	
Organization:	Zazuko GmBH
Contact person(s)	Adrian Gschwend, info@zazuko.com
Website	https://zazuko.com/
Community	*Commercial support offered*
Sources:	
Formal	*N/A*
Git	[GitHub - zazuko/expressive-rdf-mapper: A friendly language for mappings to RDF]()
Tutorials:	
Papers:	*N/A*
Content:	
Abstract	*"Expressive RDF Mapper (XRM) is an RDF mapping language for humans. It greatly facilitates the mapping of non-RDF resources to RDF.* *XRM is available as an Eclipse and Visual Studio Code Plugin.* *The software is available under the MIT license, source code can be found here (opens new window). We offer commercial support for it, get in contact with us.* *Mapping existing data into graph-based structures is an essential part of building a knowledge graph. Knowledge engineers take highly specialized domain knowledge and map it to better understandable and re-usable terms and structures that can be used outside the context of the initial application. The Data-Centric Manifesto (opens new window)explains why this is useful for any organization.* *A simplified process might look like this:* • *Identify and analyze interesting data sources* • *Identify the data format, like RDBs, XML, JSON, CSV, etc.* • *Find a tool that supports that particular input data format and RDF output.* • *Map to the RDF domain model (schema, ontology, or taxonomy)* *In the domain of RDF knowledge graphs, there are multiple standards available, like:* • *R2RML (opens new window)for relational databases* • *CSV on the Web (opens new window)and RML (opens new window)for tabular data like CSV or Excel* • *RML (opens new window)for XML and JSON* *We know that using any of these mapping standards in real-world projects with large and diverse data sources is a challenge, and that is why we have created XRM.* *Creating mappings in any of these standards is a tedious and error-prone task, especially for more complex data sources. Due to different output formats, it is also not possible to switch between these standards."* *Source:* [Expressive RDF Mapper (XRM) | Zazuko]()
Related tools	*Output of the tool: RML, CSV on the web, R2RML* *These mapping tools can be used in a.o., Zazuko pipeline, Carml, Stardog, Ontop, R2RML parser*

Categorisation	
Category	Configuration language
Scoring	
Expressivity	*3 – The expressivity of the mapping is limited to what the Semantic Treehouse allows the user to configure. This has a sufficient basis of mappings, but lacks some complicated transformations that may be needed.*
Maintainability	*4 – The mappings are saved in the tool and can be further reconfigured, before an update is exported. So, mappings are easily maintainable.*
Programming expertise	*2 – The tool provides an extensive user interface, so there is very little software development expertise needed to successfully utilise it.*
RDF expertise	*3 – There is little RDF expertise needed, because the semantic models can be imported into Semantic Treehouse, which makes the contents available to the user. This means they have no need to read the RDF serialisation.*
User-friendliness	*4 – There is at the moment a TNO team that can provide support, although at a cost. Additionally, there is extensive documentation and tutorials available.*
GUI-support	*5 – The GUI of Semantic Treehouse has recently been upgraded and refined for newly identified user requirements. I*
Availability	4 – The STH project is available as open source and actively being maintained in a number of TNO projects.
Scalability	5 – The XRM tool can export the mapping to RML, which is highly scalable, so, by extension, the STH mappings also are highly scalable.

5. Configuration languages

In this section, we describe configuration languages, in particular RML and its flavours, R2RML and YARRRML, SPARQL Anything/ Façade-X and the OnTop / OBDA virtualization approach. A user can use these languages to describe mapping to be executed. The mapping and the source data are then fed into an interpreter, producing the required result. The expressivity is usually higher than with GUIs, but there may still be complicated mappings that the configuration language doesn't facilitate. The configuration language generally separate the mapping definition from the mapping execution, which facilitates optimizing the computational complexity, resulting in a potentially high scalability.

5.1 R2RML

R2RML	
Contact information:	
Organization:	*W3C*
Contact person(s)	*W3C*
Website	*https://www.w3.org/2001/sw/rdb2rdf/r2rml/FPWD-rdb2rdf-r2rml/*
Community	*Due to it being a specification, the number is unknown.*
Sources:	
Formal	*https://www.w3.org/2001/sw/rdb2rdf/r2rml/FPWD-rdb2rdf-r2rml/*
Git	*N/A*
Tutorials:	*https://github.com/chrdebru/r2rml-tutorial*
Papers:	*https://www.w3.org/TR/r2rml/*
Content:	
Abstract	*The RDF Mapping language (RML) is a mapping language defined to express customized mapping rules from heterogeneous data structures and serializations to the RDF data model. RML is defined as a superset of the W3C-standardized mapping language (R2RML), aiming to extend its applicability and broaden its scope, adding support for data in other structured formats.* *R2RML is the W3C standard to express customized mappings from relational databases to RDF. RML follows exactly the same syntax as R2RML; therefore, RML mappings are themselves RDF graphs.* *On top of that, YARRRML is a human readable text-based representation for declarative Linked Data generation rules. It is a subset of YAML, a widely used data serialization language designed to be human-friendly. It can already be used to represent R2RML and RML rules.* *NB. Same abstract for R2RML, RML, and YARRRML.* In the case of the R2RML specification, the latest release is from 2012. RML and R2RML are specific tools and specifications to transform heterogeneous formats into a common semantic model easily. The available tools are rather in the infancy but they are effective in the transformations.
Related tools	*RML*
Categorisation	
Category	*Configuration language*
Scoring	
Expressivity	*4 – RML is primarily designed to be able to express a multitude of mappings. Some edge cases may not be configurable. RML can be inefficient for mapping nested data and heterogeneous data sources due to the challenge in parsing hierarchical structured data.*

Maintainability	*2 – RML is difficult to maintain, since the syntax is quite idiosyncratic and hard to read or understand from the syntax which mapping is being described.*
Programming expertise	*2 – Although not necessarily simple, RML is a configuration language instead of a programming language. The RML interpreter & mapper has already been implemented and is available as open source program.*
RDF expertise	*5 – The concrete syntax has little resemblence to RDF, but the question of which mapping to implement does require knowledge about the underlying model that is being reused.*
User-friendliness	*1 – Along with the maintainability, RML is difficult to read without education.*
GUI-support	2 – There is little GUI support for RML. Essentially, YARRRML is intended as a syntactic sugar for RML, so the GUI support for YARRRML is a proxy for RML GUI support.
Availability	4 – Developments of RML are open source available. It's not (yet) standardized, unlike R2RML, which it extends upon.
Scalability	5 – The execution of the R2RML mapping is separate from the specification of the transformation, so the pipeline can be optimized without touching the transformation.

5.2 RML

RML	
Contact information:	
Organization:	*Ghent University*
Contact person(s)	*Ghent University*
Website	*https://rml.io/*
Community	*15-20 contributors by each of the 50 repositories.*
Sources:	
Formal	*https://rml.io/specs/rml/*
Git	*https://github.com/RMLio*
Tutorials:	*https://rml.io/docs/rml/tutorials/target/*
Papers:	*Dimou, Anastasia et al. "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data." LDOW (2014)*
Content:	
Abstract	*The RDF Mapping language (RML) is a mapping language defined to express customized mapping rules from heterogeneous data structures and serializations to the RDF data model. RML is defined as a superset of the W3C-standardized mapping language (R2RML), aiming to extend its applicability and broaden its scope, adding support for data in other structured formats.* *R2RML is the W3C standard to express customized mappings from relational databases to RDF. RML follows exactly the same syntax as R2RML; therefore, RML mappings are themselves RDF graphs.* *On top of that, YARRRML is a human readable text-based representation for declarative Linked Data generation rules. It is a subset of YAML, a widely used data serialization language designed to be human-friendly. It can already be used to represent R2RML and RML rules.* *NB. Same abstract for R2RML, RML, and YARRRML* RML reference documentation is rather new (2022) and the github repository is promoting tools related to RML up to date (most of the latest releases are from 2023). RML is still under active development, for example an extension to cover

	RDF-star is currently being developed as RML-star [https://ceur-ws.org/Vol-2980/paper374.pdf]
Related tools	*R2RML is closely to related to the other tools in the RML family. RML is a superset extending R2RML beyond just SQL mapping. CARML is a programming implementation of (R2)RML.*
Categorisation	
Category	*Configuration language*
Scoring	
Expressivity	*4 – RML is primarily designed to be able to express a multitude of mappings. Some edge cases may not be configurable. RML can be inefficient for mapping nested data and heterogeneous data sources due to the challenge in parsing hierarchical structured data.*
Maintainability	*2 – RML is difficult to maintain, since the syntax is quite idiosyncratic and hard to read or understand from the syntax which mapping is being described.*
Programming expertise	*2 – Although not necessarily simple, RML is a configuration language instead of a programming language. The RML interpreter & mapper has already been implemented and is available as open source program.*
RDF expertise	*5 – The concrete syntax has little resemblence to RDF, but the question of which mapping to implement does require knowledge about the underlying model that is being reused.*
User-friendliness	*1 – Along with the maintainability, RML is difficult to read without education.*
GUI-support	2 – There is little GUI support for RML. Essentially, YARRRML is intended as a syntactic sugar for RML, so the GUI support for YARRRML is a proxy for RML GUI support.
Availability	4 – Developments of RML are open source available. It's not (yet) standardized, unlike R2RML, which it extends upon.
Scalability	

5.3 YARRRML

YARRRML	
Contact information:	
Organization:	*Ghent University*
Contact person(s)	*Ghent University*
Website	*https://rml.io/yarrrml/*
Community	*8 contributors and 37 stars on GitHub*
Sources:	
Formal	*https://rml.io/yarrrml/spec/*
Git	*https://github.com/RMLio/yarrrml-parser*
Tutorials:	*https://rml.io/yarrrml/tutorial/getting-started/*
Papers:	*https://rml.io/yarrrml/*
Content:	
Abstract	*The RDF Mapping language (RML) is a mapping language defined to express customized mapping rules from heterogeneous data structures and serializations to the RDF data model. RML is defined as a superset of the W3C-standardized mapping language (R2RML), aiming to extend its applicability and broaden its scope, adding support for data in other structured formats.*

	R2RML is the W3C standard to express customized mappings from relational databases to RDF. RML follows exactly the same syntax as R2RML; therefore, RML mappings are themselves RDF graphs. *On top of that, YARRRML is a human readable text-based representation for declarative Linked Data generation rules. It is a subset of YAML, a widely used data serialization language designed to be human-friendly. It can already be used to represent R2RML and RML rules.* *NB. Same abstract for R2RML, RML, and YARRRML.* YARRML tools latest release is from 2023, so the updates are continuous.
Related tools	*YARRRML is highly related to the other languages of the RML family. It is intended as a language to make RML more user friendly. As such, the YARRRML engine translates YARRRML to RML, which only then can be used to transform data to RDF.*
Categorisation	
Category	*Configuration language*
Scoring	
Expressivity	*3 – the YARRRML spec is broad, but is limited by RML and limited by its translation to RML.*
Maintainability	*3 – the person updating the ontology can usually also update the YARRRML.*
Programming expertise	*2 – there is no programming expertise required to write the mapping in YARRRML, since the script is executed by a stand-alone pipeline.*
RDF expertise	*4 – the user has to be familiar with the ontology to correctly write the mapping script, as well as being familiar with browsing the ontology to identify the correct terms.*
User-friendliness	*3 – YARRRML is specifically intended as a user-friendly version of RML, so it is in a relatively readable YAML structure*
GUI-support	*3 – There is only GUI support for a web browser to get familiar with the language.*
Availability	*5 – the project is open source and welcomes contributors.*
Scalability	

5.4 Ontop/OBDA

Ontop	
Contact information:	
Organization:	
Contact person(s)	https://ontop-vkg.org/community/#active-core-team-members.
Website	https://ontop-vkg.org/
Community	https://github.com/ontop/ontop
Sources:	
Formal	*N/A*
Git	https://github.com/ontop/ontop
Tutorials:	*N/A*
Papers:	Guohui Xiao, Davide Lanti, Roman Kontchakov, Sarah Komla-Ebri, Elem Güzel-Kalayci, Linfang Ding, Julien Corman, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. The Virtual Knowledge Graph System Ontop (opens new window). In: International Semantic Web Conference (Resource Track), 2020.

	Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL Queries over Relational Databases (opens new window). In: Semantic Web Journal 8.3 (2017), pp. 471–487.
Content:	
Abstract	Ontop is a Virtual Knowledge Graph system. It exposes the content of arbitrary relational databases as knowledge graphs. These graphs are virtual, which means that data remains in the data sources instead of being moved to another database. Ontop translates SPARQL queries expressed over the knowledge graphs into SQL queries executed by the relational data sources. It relies on R2RML mappings and can take advantage of lightweight ontologies. A plugin for Protégé provides a graphical interface for mapping, editing and executing SPARQL queries. Ontop is also available as a Java library which implements both OWL API and Sesame API interfaces, and it constructs a SPARQL end-point through Sesame's Workbench that could be directly queried.
Related tools	1. It is based on R2RML 2. Supports the main database systems: PostgreSQL, MySQL, MariaDB (since 5.0.0), SQL Server, Oracle, DB2, Snowflake (since 5.0.0), Databricks (since 5.0.0), Google BigQuery (since 5.0.2), AWS Redshift (since 5.0.2), DuckDB (since 5.0.2), and AWS DynamoDB (since 5.1.0) 3. Supports database federators such as Denodo, Dremio (since 4.1.0), Teiid (since 4.1.1), Apache Spark (since 4.2.0) and Trino / PrestoDB / AWS Athena (since 5.0.2) 4. Provides a plugin for editing and testing mappings in the Protégé ontology editor
Categorisation	
Category	*Configuration mapping language* **Expertise required in semantic web/programming** **1.** Semantic expert **2.** Developer **Maintainability** **3.** Ontop framework is continuously updated by the core team. **Compliant with standards?** The tool is related to the R2RML specification from the W3C. Ontop is a widely used framework for easily enabling OBDA (ontology-based data access), in other words, to facilitate accessing data stored in legacy sources using Semantic Web technologies. It relies on virtual approach, which avoids materializing triples retrieved through mappings (duplicating information) and answers the SPARQL queries by translating them into SQL queries over the original data sources. It provides both, a more GUI oriented use, through the Protégé editor, that minimizes the required development skills, as well as a Java API for developers, in both cases supporting all W3C standards related to OBDA.
Scoring	
Expressivity	*4 – OnTop is based on R2RML and also provides an additional mapping language. These have a rich expressivity, but there may be some edge cases which are impossible to express.*
Maintainability	*3 – The R2RML and the OBDA language are difficult to maintain.*
Programming expertise	*4 – There is no programming expertise required to work with OnTop.*
RDF expertise	*4 – There is little semantic technology expertise required beyond familiarity with the ontology.*

User-friendliness	*4 – OnTop provides additional support for R2RML and OBDA users.*
GUI-support	*3 – There is no GUI support, but users can employ and IDE.*
Availability	*4 – The Ontop material is available in Git, with additionally the Ontopic commercial company that could provide (paid) support.*
Scalability	*4 – The scalability of R2RML is high, but OnTop additionally supplies the feature of virtualisation which may impact the scalability aspect.*

5.5 SPARQL Anything

SPARQL Anything / Facade-X	
Contact information:	
Organization:	SPARQL Anything community
Contact person(s)	Enrico Daga, Luigi Asprino
Website	SPARQL Anything
Community	2 main contributors, 4 additional contributors, 223 stars on github
Sources:	
Formal	Introduction - SPARQL Anything
Git	SPARQL-Anything/sparql.anything: SPARQL Anything is a system for Semantic Web re-engineering that allows users to ... query anything with SPARQL.
Tutorials:	Quick Start - SPARQL Anything
Papers:	Asprino, Luigi, Enrico Daga, Aldo Gangemi, and Paul Mulholland. **Knowledge graph construction with a façade: a unified method to access heterogeneous data sources on the web.** ACM Transactions on Internet Technology 23, no. 1 (2023): 1-31.
Content:	
Abstract	Features according to SPARQL Anything-website: • Query files in plain SPARQL 1.1, via the SERVICE <x-sparql-anything:> (see options) • Build knowledge graphs with CONSTRUCT queries, in plain SPARQL 1.1 • Supported formats: XML, JSON, CSV, HTML, Excel, Text, Binary, EXIF, File System, Zip/Tar, Markdown, YAML, Bibtex, DOCx (see format-specific options) • Transforms files, inline content, or the output of an external command • Full fledged HTTP client to query Web APIs (headers, authentication, all methods supported) • Functions library for RDF sequences, strings, hashes, easy entity building, ... • Combine multiple SERVICE clauses into complex data integration queries (thanks to SPARQL) • Query templates (using BASIL variables) • Save and reuse SPARQL Results Sets files as input for parametric queries • Slice large CSV files with an iterator-like execution style (soon JSON and XML) • Supports an on-disk option (with Apache Jena TDB2) • Command line and server (via Jena Fuseki)

Related tools	N/A
Categorisation	
Category	Configuration language
Scoring	
Expressivity	*5 – SPARQL Anything extends SPARQL, so in the construct part of the query there is more freedom to describe the resulting triples.*
Maintainability	*4 – SPARQL Anything separates on the one hand the extraction of source data information from on the other hand the definition of the resulting triples. This improves the maintainability, because both can be changed separately.*
Programming expertise	*2 – Very little programming expertise is needed, because the SPARQL Anything statements are used by an interpreter which has already been provided.*
RDF expertise	*4 – There is RDF expertise needed to successfully perform the data transformation task, because the user still needs to appropriately choose the RDF properties and classes to map the data to.*
User-friendliness	*3 – SPARQL Anything has among its features various additions to easily work with it, such as a functions library, a Command Line Interface, and an integration with Apache Jena.*
GUI-support	*3 – SPARQL Anything is more of a research project than a commercial tool, so there is no user interface and little additions specifically geared towards user-friendliness. Still a user's background knowledge of SPARQL may be beneficial.*
Availability	*5 – SPARQL Anything is open source available as a CLI, web server and software library in Java and Python.*
Scalability	*4 – Since SPARQL Anything separates the data transformation definition from the execution, there is potential to improve the scalability separately from working on the mapping. However, it is unclear which priority the optimization has and which role the inheritance of SPARQL plays.*

5.6 SPARQL-Generate

SPARQL-Generate	
Contact information:	
Organization:	Mines Saint-Étienne
Contact person(s)	Maxime Lefrançois
Website	https://ci.mines-stetienne.fr/sparql-generate/
Community	
Sources:	
Formal	
Git	https://github.com/sparql-generate/sparql-generate
Tutorials:	https://ci.mines-stetienne.fr/sparql-generate/playground.html
Papers:	Lefrançois, M., Zimmermann, A., & Bakerally, N. (2017). A SPARQL extension for generating RDF from heterogeneous formats. In The Semantic Web: 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part I 14 (pp. 35-50). Springer International Publishing.
Content:	
Abstract	SPARQL-Generate is an expressive template-based language to generate RDF streams or text streams from RDF datasets and document streams in arbitrary formats. It presents the following advantages: * It extends SPARQL 1.1. Anyone familiar with SPARQL can easily learn SPARQL-Generate; * SPARQL-Generate leverages the expressivity of SPARQL 1.1: Aggregates, Solution Sequences and Modifiers, SPARQL functions and their extension mechanism; * It can be extended to support new data sources and formats; * It integrates seamlessly with existing standards for consuming Semantic Web data, such as SPARQL or Semantic Web programming frameworks. SPARQL-Generate can generate RDF streams or text streams from RDF, SQL, XML, JSON, CSV, GeoJSON, HTML, CBOR, plain text with regular expressions, large CSV documents, MQTT or WebSocket streams, repeated HTTP GET operations.
Related tools	SPARQL-Anything
Categorisation	
Category	Configuration language
Scoring	
Expressivity	4 – The language is highly expressive with extensive inbuilt constructs. Additionally, it can perform the mapping in two ways, which many mapping languages cannot.
Maintainability	2 – The complications in the language means that familiarity is a specialised skill.
Programming expertise	4 – Since this is a configuration there's no software development expertise required
RDF expertise	5 – The language is an extension of SPARQL, so knowledge of SPARQL is required. Additionally, the high expressivity means the language is more complicated than other
User friendliness	2 – The language is quite complicated, so beyond the knowledge of SPARQL there are several highly specific constructs in the language.
GUI support	3 – There is a playground that has been developed to support the usage of the language and getting familiar with it.

Availability	5 – The project is open source available with extensive documentation: the web playground, inside Sublime Text, as an executable JAR, and as an open source Java library;
Scalability	4 – The definition of the mapping is done separately from the execution of the mapping, so the architecture facilitates the separation of the mapping and the optimization of the algorithm. As such, the scalability can be improved independently from the mapping.

5.6 Stardog Mapping Syntax 2 (SMS2)

Stardog Mapping Syntax 2	
Contact information:	
Organization:	Stardog
Contact person(s)	Contact Us | Stardog
Website	The Enterprise Knowledge Graph Platform | Stardog
Community	Commercial support and community forum at Stardog Community
Sources:	
Formal	*N/A*
Git	*N/A, closed source*
Tutorials:	Virtual Graph Mappings | Stardog Documentation Latest Mapping Data Sources | Stardog Documentation Latest stardog-tutorials/mappings at master · stardog-union/stardog-tutorials · GitHub
Papers:	*N/A*
Content:	
Abstract	*"Stardog supports a set of techniques for unifying structured enterprise data – chiefly, Virtual Graphs, which let you declaratively map data into a Stardog knowledge graph and query it via Stardog in situ."* *Source:* Virtual Graphs | Stardog Documentation Latest
Related tools	*Input functionality from: Cassandra, CosmosDB, MongoDB, Elasticsearch, JSON, CSV* *Output functionality to: SMS, SMS2,* R2RML
Categorisation	
Category	*Configuration language*
Scoring	
Expressivity	*3 – SMS2 supports the configuration of many type of data mappings and transformations, but highly complicated ones may not be possible or unfeasible to implement.*
Maintainability	*4 – SMS2 separates the definition of the mapping from the execution of the mapping, which improves the maintainability. Additionally, SMS2 supports the creation of a set of mappings all for a different part of the data transformation, which facilitates maintaining parts of the mapping while leaving the rest untouched.*
Programming expertise	*2 – There is no software development expertise required, because the configurations are fed into the SMS2 engine which interprets and executes the mapping.*
RDF expertise	*4 – Some RDF expertise is required, because a user needs to choose which classes or properties of the ontology to reuse.*
User-friendliness	*4 – The SMS2 configuration language has explicitly been kept concise and can be seen as a human-friendly version of R2RML.*

GUI-support	*3 – There is at the moment no dedicated GUI support for writing SMS2 mappings, but SPARQL-based syntax highlighting in an IDE is available.*
Availability	*4 – The SMS2 mappings can be exported to R2RML, so they are by extension as scalable as R2RML is. However, Stardog gives the user the option of virtualising instead of materialising the graph. The materialisation approach may be more scalable than the virtualisation approach.*

6. Programming framework

In this section, we describe software development frameworks that can be used to construct a pipeline application, in particular the Apache Jena library, RDFLib (Pythin), RATT (Typescrips), Carml (Java implementation of RML), and Neo4J.

6.1 Apache Jena Library

Apache Jena	
Contact information:	
Organization:	Apache Software Foundation
Contact person(s)	https://jena.apache.org/about_jena/team.html
Website	https://jena.apache.org/
Community	The community (Github repository have 100 active members) and the project has 1000 stars.
Sources:	
Formal	https://jena.apache.org/
Git	https://github.com/apache/jena/tree/main
Tutorials:	https://jena.apache.org/tutorials/index.html
Papers:	N/A
Content:	
Abstract	Apache Jena is a free and open-source Java framework for building Semantic Web and Linked Data applications. The framework is composed of different APIs interacting together to process RDF data. The framework is composed of the following components: - **RDF API**. Interact with the core API to create and read Resource Description Framework (RDF) graphs. Serialise your triples using popular formats such as RDF/XML or Turtle. - **ARQ (SPARQL)**. Query your RDF data using ARQ, a SPARQL 1.1 compliant engine. ARQ supports remote federated queries and free text search. - **TDB**. Persist your data using TDB, a native high performance triple store. TDB supports the full range of Jena APIs. - **Fuseki**. Expose your triples as a SPARQL end-point accessible over HTTP. Fuseki provides REST-style interaction with your RDF data. - **Ontology API**. Work with models, RDFS and the Web Ontology Language (OWL) to add extra semantics to your RDF data. - **Inference API**. Reason over your data to expand and check the content of your triple store. Configure your own inference rules or use the built-in OWL and RDFS reasoners.
Related tools	N/A
Categorisation	
Category	*Software development kit* Apache Jena is one of the most used frameworks in the elaboration of semantic tools. The library is widely documented and also, the community is huge to solve any problem. For the elaboration of semantic tools in Java, Jena is the most used library. **Maintainability** **1.** Apache Jena is continuously updated by the Apache foundation.

	Compliant with standards? 2. Apache Jena is compliant with RDF/RDFS/OWL semantic standards. Also is compliant with SPARQL.
Scoring	
Expressivity	*5 – Implementing a pipeline within Apache Jena gives full expressivity that is not limited by any choices made in the configuration language.*
Maintainability	*2 – Maintaining a Apache Jena pipeline means that the person also needs software development expertise. Additionally, the implemented mapping is integrated into the code instead. As such, it may be difficult to have changes in the data or ontology be reflected in the pipeline.*
Programming expertise	*2 – there is software development expertise required to succesfully implement a pipeline. Jena framework can be used with (Java) programming expertise for more advanced functionalities*
RDF expertise	*4 – the user has to be familiar with semantic web technologies (RDF, RDFS, OWL, SPARQL).*
User-friendliness	*5 – there is a GUI when using Jena Fuseki.*
GUI-support	*4 – there is a GUI when using Jena Fuseki, however this is a GUI for inspecting the triples instead of a GUI to perform the mapping. However, there is extensive support in terms of IDEs.*
Availability	*5 – Jena framework is open source and has been maintained since 2011 (https://jena.apache.org/). It is licensed under the Apache License, Version 2.0Apache License, Version 2.0.*
Scalability	*2 – The mapping definition and the mapping execution are defined in the same Java project, so it's difficult to separately work on the optimization.*

6.2 RDFLib

RDFLib	
Contact information:	
Organization:	Open source community
Contact person(s)	RDFLib/rdflib · Discussions · GitHub and other channels: rdflib 7.1.0 — rdflib 7.1.0 documentation
Website	https://rdflib.readthedocs.io/
Community	*Open source*
Sources:	
Formal	rdflib — rdflib 7.1.0 documentation
Git	GitHub - RDFLib/rdflib
Tutorials:	Getting started with RDFLib — rdflib 7.1.0 documentation
Papers:	
Content:	
Abstract	"RDFLib is a pure Python package for working with RDF. RDFLib contains most things you need to work with RDF, including: • parsers and serializers for RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, Trig and JSON-LD • a Graph interface which can be backed by any one of a number of Store implementations • store implementations for in-memory, persistent on disk (Berkeley DB) and remote SPARQL endpoints

	• a SPARQL 1.1 implementation - supporting SPARQL 1.1 Queries and Update statements • SPARQL function extension mechanisms" ~source: GitHub - RDFLib/rdflib: RDFLib is a Python library for working with RDF, a simple yet powerful language for representing information.
Related tools	**Import functionality for:** csv, tabular data, json, etc **Export functionality for: basically all semantic web serialisation (**RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, JSON-LD, RDFa and Microdata.)
Categorisation	
Category	*Programming framework* High software development experience required and significant semantic web experience required. Python may require a lower level of software development expertise compared to Java. The Maintainability is low, because the developed program is a full typescript program. The benefit of a typescript pipeline is the full flexibility of an ETL program, but it comes at the cost that it's only maintainable by a software developer.
Scoring	
Expressivity	*5 – as a fully scripted pipeline RDFlib can express any changes in structure from a source format to a target RDF format*
Maintainability	*4 – the python code means that a maintainer should also be fluent in python. Also, a person with RDF expertise who updates the ontology does not necessarily have the Python expertise to update the pipeline.*
Programming expertise	*5 – software development expertise is required to write an RDFlib data to ontology mapper.*
RDF expertise	*4 – the mapping script in RDFlib is different from its formalisation in SSSOM or from its description in Excel, so a proper RDF basis is needed to create the correct program.*
User-friendliness	*3 – the RDFlib has proper documentation, but its usage is not widespread enough for an abundance of issues answers on the web.*
GUI-support	*3 – there is enough support for Python, but no specific support for RDFlib*
Availability	*5 – the code is open source available.*
Scalability	*2 – The mapping definition and the mapping execution are defined in the same Java project, so it's difficult to separately work on the optimization.*

6.3 RDF all the things (RATT) and TriplyETL

TriplyETL / RATT	
Contact information:	
Organization:	Triply
Contact person(s)	Wouter Beek (Triply)
Website	Overview - Triply Documentation
Community	
Sources:	
Formal	
Git	
Tutorials:	RATT - Triply Documentation
Papers:	
Content:	
Abstract	RATT transformations are a core set of functions that are commonly used to change the content of TriplyETL Records. RATT transformations started out as TypeScript transformations that turned out to be useful in a wide variety of TriplyETL pipelines. Triply maintains this core set of transformation functions to allow new ETLs to make use of off-the-shelf functionality that has proven useful in the past.
Related tools	**Import functionality for:** csv, tabular data, json, etc **Export functionality for: semantic web serialisations**
Categorisation	
Category	*Programming framework* High software development experience required and significant semantic web experience required The Maintainability is low, because the developed program is a full typescript program. The benefit of a typescript pipeline is the full flexibility of an ETL program, but it comes at the cost that it's only maintainable by a software developer.
Scoring	
Expressivity	*5 – As a software development framework the user has full expressivity to describe any mapping, not constrained by the limitations of a programming language.*
Maintainability	*3 – Typescript is a simple language compared to, for example, Java, so the maintainability can be considered a little higher.*
Programming expertise	*5 – To succesfully implement a data transformation pipeline, the user still needs to have software development expertise.*
RDF expertise	*4 – A user still needs familiarity with RDF and the domain ontology to properly identify which concepts to map to each other.*
User-friendliness	*4 – The library is well-documented to support usage by software developers.*
GUI-support	*3 – There is no dedicated GUI-support, but as a programming library there is abundant support in the form of IDEs.*
Availability	*4 – The package is available on the common package repository NPM.*
Scalability	*2 – The mapping definition and the mapping execution are defined in the same Java project, so it's difficult to separately work on the optimization.*

6.4 CARML

CARML	
Contact information:	
Organization:	CARML was first developed by Taxonic in cooperation with Kadaster. And is now being maintained and developed further by Skemu.
Contact person(s)	Pano Maria
Website	github repository
Community	Open Source Community in Github
Sources:	
Formal	N/A
Git	github repository
Tutorials:	github readme
Papers:	N/A
Content:	
Abstract	CARML is a java library that transforms structured sources to RDF as declared in an RML mapping, in accordance with the RML spec.
Related tools	N/A
Categorisation	
Category	*Programming framework*
Scoring	
Expressivity	*5 – The CARML library is an implementation of RML, which in itself is already quite expressive, but the Java implementation provides options to express any transformation.*
Maintainability	*2 – It will be difficult for a user to easily reuse another developer's mapping.*
Programming expertise	*5 – Experience in software developing is required to appropriately use this package.*
RDF expertise	*4 – A user still needs RDF expertise to appropriately decide which parts of the programming framework to reuse.*
User-friendliness	*3 – The software package is well-documented.*
GUI-support	*2 – There is no dedicated GUI support, but there are enough IDEs for Java software development.*
Availability	*3 – The library is readily available through Git.*
Scalability	*3 – The mapping definition and the mapping execution are defined in the same Java project, so it's difficult to separately work on the optimization.*

6.5 Neo4J

Neo4J	
Contact information:	
Organization:	Neo4j, Inc
Contact person(s)	Jessica Cannella (jessica.cannella@neo4j.com)
Website	https://neo4j.com/
Community	Neo4j has an active support community.
Sources:	
Formal	N/A
Git	Neo4j GitHub https://neo4j.com/developer
Tutorials:	https://neo4j.com/docs/getting-started/appendix/tutorials/tutorials-overview/ Mapping graph models - Neosemantics (neo4j.com)
Papers:	N/A
Content:	
Abstract	"Mappings can be used for applying transformations to the RDF as it's imported into Neo4j or to transform the vocabulary used in a Neo4j graph as it's exported through the different RDF export methods described in **Exporting RDF data**. The following are two examples of situations where mappings can be useful: • We have a graph in Neo4j that we want to publish as JSON-LD through a REST api, but we want to map the elements in our graph (labels, property names, relationship names) to a public vocabulary so our API 'speaks' that public vocabulary and is therefore easily consumable by applications that 'speak' the same vocabulary. • We want to import a public taxonomy specified using **the SKOS model** but we want to persist it in Neo4j using our own schema terms. An example of this is that we want skos:narrower relationships in the taxonomy to be called :CHILD_CATEGORY in our Neo4j graph. Both types of transformations can be described using mappings and neosemantics will action on them both on import and on export as described." From: Mapping graph models - Neosemantics (neo4j.com)
Related tools	N/A
Categorisation	
Category	*Configuration language.*
Scoring	
Expressivity	*3 – various input and output formats are supported. Supports .csv, yaml, ttl, rdf, owl formats, among others.*
Maintainability	*3 – there is no export of the mapping to an independent format, but the project can be saved. the mapping can be exported in png or the queries can be saved in cypher.*
Programming expertise	*2 – no real programming, but installation and configuration require know-how*
RDF expertise	*3 – neo4j has a plugin (neosemantics) that allows you to import ontology languages, such as owl, rdf, sparql and others. The user needs to know the concepts of the ontology and map them accordingly*
User-friendliness	*4 – the GUI provides simple but far-reaching support for creating the mapping. In addition to the desktop version of Neo4J, there is also Neo4J Aura, which works in the cloud.*

GUI-support	*3 – simple, bit straightforward visualization and editing support*
Availability	*4 – available with documentation*
Scalability	*2 – The mapping definition and the mapping execution are defined in the same Java project, so it's difficult to separately work on the optimization.*

7. LLM based technologies

In this section, we describe LLM-based technologies, in particular existing research works that utilise different large language models, such as OntoGenix for generating RML statements and another tool that investigates the usage of LLMs throughout the KG production process.

7.1 OntoGenix

OntoGenix	
Contact information:	
Organization:	Leipzig University, Germany
Contact person(s)	Marvin Hofer1, Johannes Frey and Erhard Rahm
Website	*https://github.com/tecnomod-um/OntoGenix/blob/main/README.md*
Community	https://github.com/tecnomod-um/OntoGenix
Sources:	
Formal	N/A
Git	https://github.com/tecnomod-um/OntoGenix
Tutorials:	*https://github.com/tecnomod-um/OntoGenix/blob/main/docs/OntoGenix.mp4*
Papers:	"Towards self-configuring Knowledge Graph Construction Pipelines using LLMs - A Case Study with RML"
Content:	
Abstract	"This paper explores using large language models (LLMs) to generate resource mapping language (RML) files in the RDF turtle format as a key step towards self-configuring RDF knowledge graph construction pipelines. Our case study involves mapping a subset of the Internet Movie Database (IMDB) in JSON format given a target Movie ontology (selection of DBpedia Ontology OWL statements). We define and compute several scores to assess both the generated mapping files and the resulting graph using a manually created reference. Our findings demonstrate the promising potential of the state-of-the-art commercial LLMs in a zero-shot scenario."
Related tools	Import functionality for: CSV, JSON, TTL. Export functionality for: TTL, N-Triples, RML, Ontology diagram PNG. Requires: API access to OpenAI GPT-4 model. Uses: Morph-KGC (https://github.com/morph-kgc/morph-kgc) to construct RDF knowledge graphs.
Categorisation	
Category	*LLM-based*
Scoring	
Expressivity	*4 – OntoGenix is able to extract semantics from raw data and generate ontologies, RML rules and knowledge graphs using LLMs models.*
Maintainability	*3 – It is structured in a modular way using standard libraries and interfaces (e.g., RDFLib, Flask). The code is openly available and relatively easy to maintain. However, it depends on the OpenAI GPT-4 API access.*
Programming expertise	4 – Minimal programming expertise is needed to run OntoGenix via CLI or GUI.
RDF expertise	5 – It is designed to reduce the need for in-depth RDF expertise. It might need some background knowledge to confirm the correctness of the output.
User-friendliness	5 – It is easy to use and user-friendly. GPT-generated instructions and outputs lower the barrier to using the tools.

GUI-support	5 – It provides a GUI that allows users to upload data, provide prompts, and visualize results without needing to interact with command line.
Availability	4 – It is open source, available on GitHub with documentation.
Scalability	2 – It is still an open question how scalable this approach is, but we can already remark that for the scalability we need to take the potential cost into account for continuously prompting an LLM for a data transformation. Additionally, LLMs take a long time to generate their answer.

7.2 LLM assisted KG construction

LLM assisted KG construction	
Contact information:	
Organization:	*University Jena, Jena, Germany*
Contact person(s)	*vamsi.krishna.kommineni@uni-jena.de* *birgitta.koenig-ries@uni-jena.de* *sheeba.samuel@uni-jena.de*
Website	*https://github.com/fusion-jena/automatic-KG-creation-with-LLM/blob/master/README.md*
Community	-
Sources:	
Formal	*N/A*
Git	*https://github.com/fusion-jena/automatic-KG-creation-with-LLM*
Tutorials:	*N/A*
Papers:	*"From human experts to machines: An LLM supported approach to ontology and knowledge graph construction"*
Content:	
Abstract	*"The conventional process of building Ontologies and Knowledge Graphs (KGs) heavily relies on human domain experts to define entities and relationship types, establish hierarchies, maintain relevance to the domain, fill the ABox (or populate with instances), and ensure data quality (including amongst others accuracy and completeness). On the other hand, Large Language Models (LLMs) have recently gained popularity for their ability to understand and generate human-like natural language, offering promising ways to automate aspects of this process. This work explores the (semi-)automatic construction of KGs facilitated by open-source LLMs. Our pipeline involves formulating competency questions (CQs), developing an ontology (TBox) based on these CQs, constructing KGs using the developed ontology, and evaluating the resultant KG with minimal to no involvement of human experts. We showcase the feasibility of our semi-automated pipeline by creating a KG on deep learning methodologies by exploiting scholarly publications. To evaluate the answers generated via Retrieval-Augmented-Generation (RAG) as well as the KG concepts automatically extracted using LLMs, we design a judge LLM, which rates the generated content based on ground truth. Our findings suggest that employing LLMs could potentially reduce the human effort involved in the construction of KGs, although a human-in-the-loop approach is recommended to evaluate automatically generated KGs."*
Related tools	*CUDA enabled GPU machine is required*
Categorisation	
Category	*LLM-based*
Scoring	
Expressivity	*3 - Various input and output are supported.*

Maintainability	*3 - It is not difficult to maintain since it gives the possibility to change the configuration related to the LLMs.*
Programming expertise	3 - No detail programming, but user needs to know how to setup environments, configure access tokens and customize it.
RDF expertise	4 - It doesn't require in-depth knowledge of RDF; however, the correctness of the results from the automation needs to be validated.
User-friendliness	2 - No GUI provided but CLI is available.
GUI-support	2 - Only command line interface is available.
Availability	4 - Open source, available with documentation.
Scalability	3 – It's difficult to discuss the scalability of this approach, because it has so many facets. For the scalability of LLM-based tools we need to take into account the potential costs for continuously prompting an LLM and need to consider the long time it takes to answer a question.

8. Which tool is best for you to develop Semantic Web Applications?

8.1 A one-dimensional trade-off

The main trade-off we identify is between the extremes of on the one hand GUI-based tools and on the other hand programming or configuration languages. The GUI tools provide a high user-friendliness but this usually reduces the expressivity of the mappings that can be constructed. The high user-friendliness shows itself by requiring no software development expertise as well as little technical expertise with semantic language standards. The configuration or programming languages make up for that mainly in expressivity. Complicated transformations may only be possible in a JDK or in the RML core language.

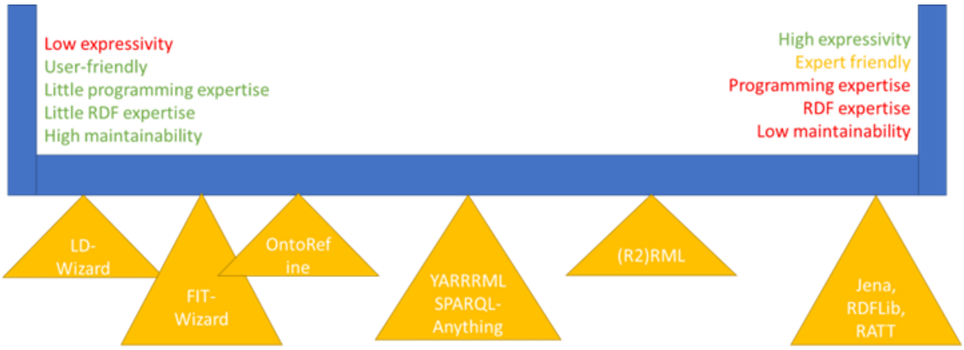
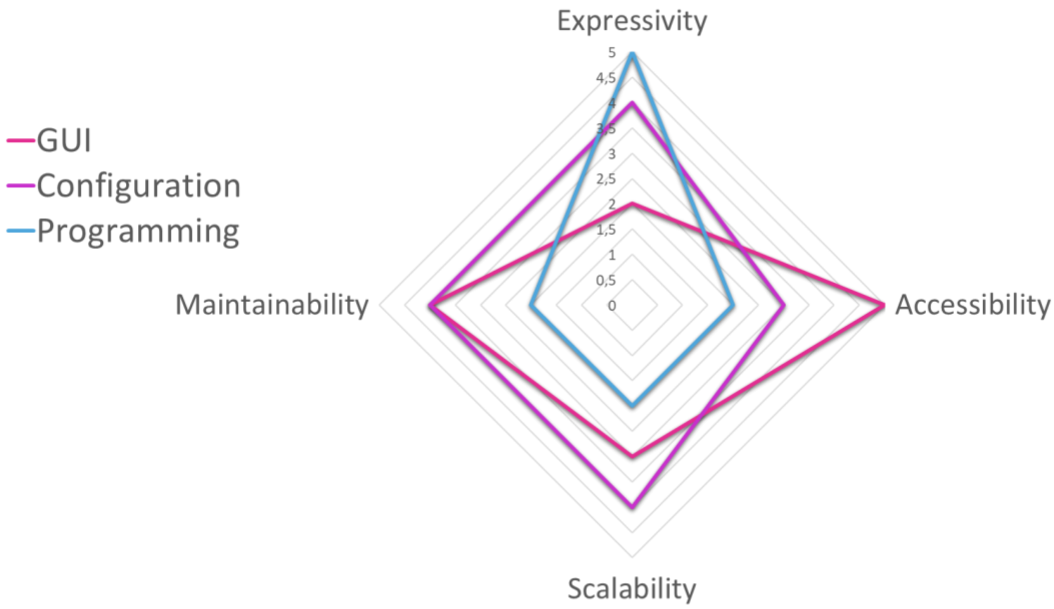


Figure 2 Sketch & hypothesis of outcome: an overview of where all tools would be situated based on their user-vs expert-friendliness

8.2 Multi-dimensional trade-off

We can extend the one-dimensional scale of the previous section to a multidimensional approach to identify four macro-characteristics: expressivity, maintainability, scalability, and accessibility.



We find that the programming approach excels in the expressivity aspect, but it lags quite a bit behind in the other aspects. Software development experience will in any case be required to maintain or construct existing mappings. Additionally, the exact structure of the triples being produced can be obscured by the programming code, dependent on the maturity of the library being used.

The GUI approach excels at accessibility for novice users, as we may expect of their key selling point. All mature GUI tools also provide an export to one of the configuration languages mentioned. Therefore, we can consider the maintainability and the scalability of the GUI tool to be dependent on the scalability of the configuration language. The GUI approach mainly lacks in the expressivity, because the tool may be more suited to a transformation of a single CSV-column to a single datatype property instead of restructuring a CSV file to the ontology structure.

We consider the configuration language a mixed approach that combines some of the strengths of both worlds. Its expressivity can usually cover the majority of use cases encountered. Additionally, it doesn't require software development expertise, but only requires a familiarity with the configuration language, which may be YAML-based. Additionally, it has the most potential for flexible scalability, because the conceptual mapping is defined separately from the technical mapping. There will still be some work to do on the developer support of the various configuration tools.

Informative references

Ontology Definition: Tom Gruber, in the Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009

R2RML specification: R2RML: RDB to RDF Mapping Language (w3.org)

RML specification: RDF Mapping Language (RML)

YARRRML specification: YARRRML

Carml: GitHub - carml/carml: A pretty sweet RML engine, for RDF.

(YARR)RML resources: RML.io

LD Wizard: LDWizard (netwerkdigitaalerfgoed.nl)

FIT Wizard: FIT wizard | Semantic Treehouse (semantic-treehouse.nl)

OntoRefine: Ontotext Refine

Apache Jena: Apache Jena - Home

RATT: 2. Transform: RATT - Docs - Triply

RDFLib: rdflib 7.0.0 — rdflib 7.0.0 documentation

C#/.NET library for RDF: dotNetRDF

Another .NET library: GitHub - mdesalvo/RDFSharp: Lightweight and friendly .NET library for realizing Semantic Web applications

SPARQL Anything: rdflib 7.0.0 — rdflib 7.0.0 documentation

Facade-X: [2106.02361] Facade-X: an opinionated approach to SPARQL anything (arxiv.org)

Scientific literature review: Dylan Van Assche | Publications

OWL Classes in Java: paper_149_Kamburjan_et_al.pdf (eswc-conferences.org)

GraphDB documentation: General — GraphDB 10.0.0 documentation (ontotext.com)

Oracle RDF: https://docs.oracle.com/en/database/oracle/oracle-database/19/rdfrm/rdf-views.html

Stardog mappings: Mapping Data Sources | Stardog Documentation Latest

Virtuoso mappings: Generate Relational Tables to RDF Entity Relationship Graphs mappings—using Virtuoso's RDF Views Wizard | by Kingsley Uyi Idehen | OpenLink Virtuoso Weblog | Medium

Appendix A: scores overview

Score all tools on these aspects with a (short) explanation of why you gave it that score.

Name	Expres-sivity	Maintain-ability	Program-ming exp.	RDF exper-tise	GUI-support	Availa-bility	Scala-bility
LD Wizard	2	4	2	2	2	2	3
STH Wizard	3	4	2	4	4	5	3
Meaningfy	3	4	2	4	4	5	2
GraphDB/OntotextRefine	3	2	2	3	3	3	3
R2RML	4	2	2	5	1	2	4
RML	5	2	2	5	1	2	4
YARRRML	4	3	2	4	3	2	4
Ontop/OBDA	4	3	4	4	4	3	4
SPARQL Anything / Façade-X	5	4	2	4	3	3	5
Jena	3	5	3	4	5	4	5
RDFlib	5	2	5	4	3	3	5
RDF all the things (RATT) and TriplyETL	5	3	5	4	4	2	3
Carml	5	2	5	4	3	2	3
Neo4J	3	3	2	3	4	3	4
Expressive RDF Mapper (XRM)	3	4	2	3	4	5	4
Stardog Mapping Syntax 2 (SMS2)	5	3	5	4	4	2	3

Contributors

Editor:

Cornelis Bouter, TNO

Reviewer:

Damir Filipovic, AIOTI Secretary General

Contributors:

Aitor Corchero, EURECAT
Amelie Gyrard, Trialog
Cornelis Bouter, TNO
Davide Conzon, Fondazione LINKS
George Suciu, BEIA
Henon Lamboro, Trialog
Igor Koropiecki, IHP
Izaskun Fernández, Fundacion Tekniker
Krzysztof Piotrowski, IHP
Laura Daniele, TNO
Maria Keet, Meaningfy
Martin Bauer, NEC Labs
Maxime Léfrancois, Institue Mines Telecom
Rashif Rahman, Meaningfy
Ronald Siebes, VU University Amsterdam
Svetlana Segarceanu, BEIA
Tiago Pereira, CCG/ZGDV

Acknowledgements

This paper has been written by the Semantic Interoperability Focus Group of the Alliance for AI, IoT and Edge Continuum Innovation (AIOTI) and is available under the Creative Commons Attribution 4.0 International License. The content of this document is provided 'as-is' and for general information purposes only; it does not constitute strategic or any other professional advice. The content or parts thereof may not be complete, accurate or up to date. Notwithstanding anything contained in this document, AIOTI disclaims responsibility (including where AIOTI or any of its officers, members or contractors have been negligent) for any direct or indirect loss, damage, claim, or liability any person, company, organisation or other entity or body may incur as a result, this to the maximum extent permitted by law.

<Tool or technology>	
Contact information:	
Organization:	
Contact person(s)	
Website	
Community	
Sources:	
Formal	
Git	
Tutorials:	
Papers:	
Content:	
Abstract	
Related tools	
Categorisation	
Category	
Scoring	
Expressivity	
Maintainability	
Programming expertise	
RDF expertise	
User friendliness	
GUI support	
Availability	
Scalability	

About AIOTI

AIOTI is the multi-stakeholder platform for stimulating AI, IoT and Edge Continuum Innovation in Europe, bringing together small and large companies, academia, researchers, policy makers, end-users and representatives of society in an end-to-end approach. We strive to leverage, share and promote best practices in the AI, IoT and Edge Continuum ecosystems, be a one-stop point of information to our members while proactively addressing key issues and roadblocks for economic growth, acceptance and adoption of the AI, IoT and Edge Continuum Innovation in society. AIOTI contributions goes beyond technology and addresses horizontal elements across application domains, such as matchmaking and stimulating cooperation by creating joint research roadmaps, defining policies and driving convergence of standards and interoperability.