TNO innovation for life

Application of Large Language Models to semantic data sharing: State of the Art and opportunities

TNO Publiek › TNO 2023 M12681
28 maart 2024

ICT, Strategy & Policy

www.tno.nl
+31 88 866 15 00
info@tno.nl

TNO 2023 M12681 – 28 maart 2024

Application of Large Language Models to semantic data sharing: State of the Art and opportunities

Authors	Cornelis Bouter, Eliza Hobo, Wout Hofman
Rubrication report	TNO Publiek

In collaboration with the CEF EU FEDeRATED project

Report text	TNO Publiek
Appendices	TNO Publiek
	1
Number of pages	27
Number of appendices	1
	Stichting Connekt

ChatGPT voor semantic adapter

060.56982

Summary

Large Language Models (LLMs) are expected to have a high impact on our society and economy. One of the many applications of LLMs is to support decision making, for instance in risk assessment and planning. Another area is that of productivity for software development. The first example of applying LLMs is to be explored by each individual organization implementing the FEDeRATED capabilities; the second example, namely to include LLMs in these capabilities, is explored further.

Data sharing capabilities as developed and specified by FEDeRATED can foster with LLMs. They can potentially be applied for transforming - or accessing data of various sources. These are two examples that require technical expertise and development time. By applying LLMs, seamless interoperability for sharing and accessing data can be achieved. Without development effort, data can be shared and accessed by organizations (of course it requires identification and authentication).

The potential applications of LLMs with respect to the capabilities are:

- Extending functionality – extending the semantic model with new functionality by aligning with existing models or ontologies. This is called ontology alignment.
- Data set specifications – this is about specifying events, business document data structures, message structures, etc. as tree structures of concepts, associations, and properties with the semantic model for configuring a node.
- Data transformation – integration of data sharing capabilities with IT Backend Systems. It is about transformation between the data structure of an IT system and the semantic model for sharing events and access to data.
- Data assessment – formulating a semantic (SPARQL) query on data stored by one or more organizations (e.g. container track is an example of such a query).

The applicability for LLMs in extending the functionality, data transformation, and data assessment is further explored. Regarding functionality extension, especially ontology generation and ontology matching with LLMs have been investigated.

The findings are summarized as follows:

- Ontology generation – this is still fundamental research. In the past, approaches without applying LLMs but based on Natural Language Processing (NLP) have shown a low accuracy, without practical applications. Since there is already a semantic model and alignment maybe supported by matching, ontology generation does not require more attention.
- Ontology matching – this is applicable to extending the functionality and data transformation. Many efforts based on statistical algorithms (without LLMs) have been developed since 2000. These solutions are not practical applicable. The application of an LLM to improve this aspect is identified as promising in vision papers. The issue of these LLMs is: learning. They cannot learn by feeding ontologies, they need to be fed by (human approved) matches. Ontology matching with LLMs is worthwhile exploring.

Our proposal is to explore matching of tree structures representing data set specifications. As part of matching, an LLM chatbot for constructing tree structures can be explored to support users in operating the FEDeRATED capability 'Service Registry'.

- Data assessment - commercial (triple store) vendors have implemented prototypes of SPARQL query generation with LLMs. They need further improvements for operational application and still require technical knowledge. A small experiment also showed that these types of tools can be applied for training purposes in technology.

  Data assessment over multiple sources still requires human intervention for decomposing a single querying into multiple ones.

We recommend the following for FEDeRATED and similar data sharing infrastructures based on semantic technology:

- Data transformation - investigate LLM functionality for ontology alignment and matching for data transformation fed by human validated matches and alignments. As indicated, LLM based data transformation should start from matching tree structures. A chatbot may be developed for specifying these tree structures as data set specifications.
- Data assessment chatbot - develop or improve an existing LLM-based assistant (chatbot) for data assessment and data set specification with a natural language. Two types of chatbots can potentially be developed: one to support technical developers and another to support domain (logistics) experts, both in formulating queries and data set specifications. Such chatbots must hide technical complexity of the capabilities.

At this moment, technical experts are still needed to configure and operate the capabilities. The potential impact of LLMs is that domain experts can configure and operate them as part of their operational use.

Table of content

1 Introduction

1.1 Challenge

The challenge is assessment of the State Of The Art (SOTA) and opportunities of Large Language Models (LLMs like chatGPT) in a data sharing infrastructure that is based on semantic standards and – technology, links to data are shared, and data is kept at the source.

1.2 Background

The CEF funded FEDeRATED Action and the EC DG REFORM funded Datapipe project both apply semantic web standards and – technology for data sharing between various stakeholders. A so-called node has been developed, can be configured for various use cases, and is applied in logistics and circularity (so-called Digital Product Passports).

Many stakeholders have (and will have) IT systems with other technology and proprietary data structures. Interoperability between different stakeholders is achieved by providing a semantic model for data transformation between heterogeneous IT systems and business processes.

Lately, LLMs like chatGPT are providing lots of possibilities by acting as chatbot for humans. Questions of various types can be rapidly answered, based on the training – and data set of an LLM. LLMs can be configured to fulfil a certain function that can become available as an app on a smart device.

These types of LLM applications are expected to boost productivity of software development. Other examples of applying LLMs are to support decision making and planning. These utilize data shared according to the FEDeRATED capabilities.

This study assesses the applicability of LLMs to boost productivity for data sharing and – assessment, with the potential aim of seamless data sharing (i.e. with limited human intervention for (design time) configuration.

Based on an assessment of data sharing capabilities, this main question is broken down into sub-questions:

- Extending functionality - LLMs for (semi-)automatic ontology generation and -alignment,
- Data transformation and extending functionality - LLMs for (semi-)automatic ontology matching and alignment, and

- Data assessment - LLMs as a chatbot that generates SPARQL queries with natural language input.

1.3 Limitations of LLMs

Aside from its potential, LLMs also come with risks and limitations. A key limitation is the inherent unreliability of the output of an LLM chatbot, which stems from three main factors: hallucinations, the long tail, and data freshness. Depending on its application, this is a barrier for its implementation. Where these limitations are not relevant when an LLM is used at design time (a user of an LLM can always correct the output generated by the LLM), they are a barrier to runtime use. For instance, applying generated SPARQL queries at runtime, means they have to be complete and correct.

1.3.1 Hallucinations

When an LLM produces a response that initially seems accurate and plausible but is factually incorrect, this is called a hallucination. At runtime, precision is crucial, and hallucinations can be detrimental. Even a single typo, incorrect concept, or misinterpretation can potentially invalidate an entire query, with all ensuing repercussions. If the LLM, for instance, generates a concept that does not exist in the ontology being used, a generated query would not return anything. Therefore, it would be preferable for the LLM to refrain from providing an answer in such cases. This would require an LLM to recognize its own hallucinations, which is a difficult fundamental problem that is being investigated in academia. The tendency to hallucinate is less problematic and potentially useful in other contexts, such as in assistance with writing letters and emails, where the LLM's capability to generate text flexibly and creatively is considered a valuable feature.

1.3.2 Short tail

A frequent problem of LLMs is its short tail. This means that an LLM performs good on problems with an abundance of data, but its performance quickly degrades on problems where the data quantity or quality is lacking. Examples on programming languages, such as Java and C#, are readily available, but data is scarcer in the context of BDI.

1.3.3 Data freshness

LLMs also have an issue with data freshness, perceiving the world as static or 'frozen in time.' Trained on data up to a specific date, these models lack awareness of developments after their training date such as news events, newly released ontologies, or technologies. For instance, the training data of ChatGPT-3 extends up to September 2021, making it unaware of information or advancements beyond that period. More recent files can be added via the context or, starting from GPT-4, be looked up at real-time.

1.3.4 Chatbots and LLMs

A distinction needs to be made between chatbots and LLMS. A chatbot is an interface that allows a user to interact with an LLM. The mechanism underneath this chatbot can fall into a wide range of complexity, from simple rules to complex models, such as LLMs. LLMs can be used in other applications as well, where they are not directly interacting with users.

1.4 Structure of this report

The content and the structure of this report is as follows:

- Section 2 – identifying the capabilities where LLMs can play a role.
- Section 3 – ontology generation with LLMs for extending functionality.
- Section 4 – ontology alignment and matching with LLMs for data transformation and extending functionality.
- Section 5 –an LLM query (SPARQL) chatbot for data assessment.
- Section 6 – conclusions and recommendations for future work.
- Annex – references to background material.

2 The context – semantic data sharing

Data sharing between organizations with semantic web standards and – technology is the context. The FEDeRATED architecture specifies this context. In this section, LLM opportunities in this type of data sharing infrastructure are identified. The sections hereafter explore three potential LLM applications as indicated in the introduction of this document.

2.1 Data sharing capabilities

There are different reasons for sharing data, for instance:

- Business transactions – data is shared for service – or product delivery. One organization provides a product or service to another. In logistics, these are services like transport of cargo.
- Supervision – authorities supervise the movement of goods (and persons) from various perspectives, always with a legal basis. eFTI (electronic Freight Transport Information) Regulation will lead to a query of authorities for accessing goods details onboard of for instance a truck; a digital battery passport is another example of a query in the context of the Battery Regulation.
- Decision making – organizations access external data to improve their decisions, for instance route information to avoid traffic congestions or creditworthiness for doing business with an organization.

The focus of FEDeRATED is on supervision of - and business transactions between enterprises. For this purpose, capabilities have been defined for implementation by stakeholders. They are all based on semantic web standards and -technology.

The following figure shows the various capabilities from the perspective of a single organization that shares data with others.

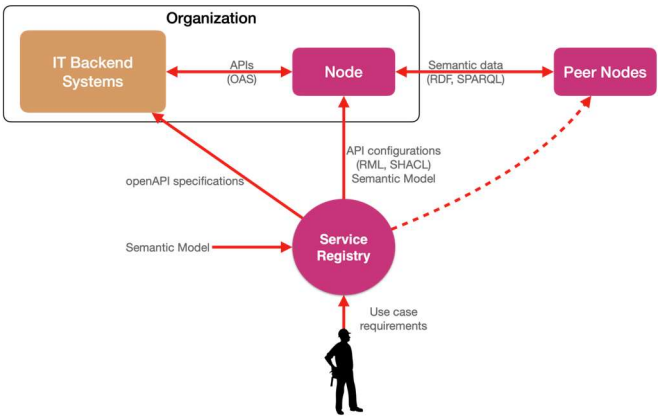


Figure 1 – data sharing from the perspective of an organization.

The following capabilities are visualized in the previous figure:

- Semantic Model – an ontology (turtle file) specifying concepts and properties for supply and logistics.
- Service Registry – a tool for data set specifications ('messages'. 'events', 'interactions' 'profile', etc.) to configure a node and generate openAPI specifications (OAS) for IT backend systems. The configurations are the semantic model, RML (currently applied for data transformation), and SHACL (data validation).
- Node – the capability for data sharing with peers using semantic standards and integrating with IT Backend Systems of an organization via openAPIs. These openAPIs are generated by the Service Registry. A node stores events with links to data shared (send and received) by an organization with its peers.

The current prototype of the Service Registry is shown in the figure. This version can be used by a community of organizations to formulate their data sharing requirements and is therefore not visualized in the domain of an organization. Future versions will also have to provide support to organizations in specifying and publishing their business services.

The capability for Identification and Authentication (IA) is not visualized in the figure. There is an intermediate proposal for IA based on authorization tokens and a long-term proposal with Verifiable Credentials. The latter require inclusion of 'profile' as is explained in the FEDeRATED Architecture. This implies that the Service Registry supporting profile development by an organization provides input for a VC of that organization.

These capabilities form an infrastructure (see following figure).

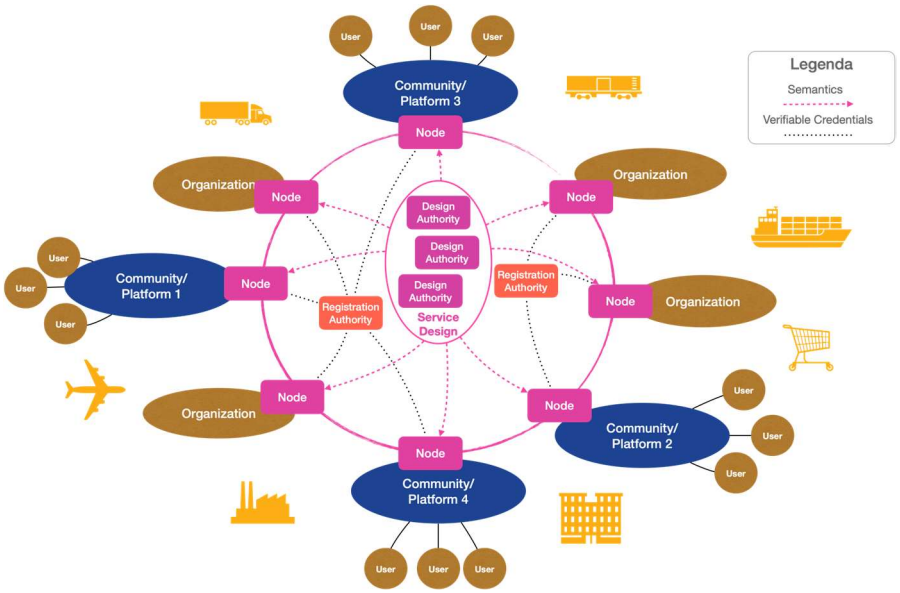


Figure 2 – interconnected nodes and different roles constituting an (open and neutral) infrastructure

The previous figure has the roles 'Registration Authority' as a (trusted) issuer of VCs and 'Design Authority' for specifying services with their interactions ('tree structures') using a Service Registry.

In this architecture, the following opportunities for applying LLMs can be identified:

- Extending functionality – extending the semantic model with new functionality by aligning with existing models or ontologies. This is called ontology alignment.
- Data set specifications – this is about specifying events, business document data structures, message structures, etc. as tree structures of concepts, associations, and properties with the semantic model for configuring a node. These tree structures are typically developed by a 'Design Authority' at this moment.
- Data sharing and - transformation – integration of data sharing capabilities with IT Backend Systems. It is about transformation between the data structure of an IT system and the semantic model for sharing events and access to data.
- Data assessment – formulating a semantic (SPARQL) query on data stored by one or more organizations (e.g. container track is an example of such a query).

These opportunities are briefly discussed hereafter.

2.2 Extending functionality

This is about including new functionality in the semantic model by alignment. Existing ontologies with new functionality are aligned with an (upper) ontology. Ontology alignment has already been done manually: the FEDeRATED ontology designers have investigated and

implemented an alignment between the FEDeRATED ontology and several external ontologies, such as the ERA Vocabulary[1], SAREF4AUTO[2], and several geography models[3].

Within the context of the FEDeRATED ontology, an additional challenge in alignment lies in the upper ontology. This is a high-level layer that provides a common framework via the inclusion of generic concepts, such as events, transport means, locations, and legal persons. The broadness of these concepts allows diverse specific ontologies to be aligned with FEDeRATED. Projecting an ontology onto an upper ontology can be extra challenging, as it requires additional common-sense reasoning to map concepts onto more general categories.

LLMs provide opportunities, for instance, to infer that a truck is a vehicle or to detect synonyms. This type of functionality is also required for ontology matching.

2.3 Data set specification

A node is configured for sharing data, where the semantics of that data is represented by a tree structure on the semantic model. Integration of a node with an IT backend system requires an API provided by that backend system. The tree structures required for configuring a node are also the specifications of these IT backend APIs.

A tree structure is a hierarchy of concepts, associations, data properties and value constraints, required to support functional requirements.

Currently, a tree editor (the one of Semantic Treehouse) supports a user in specifying tree structures. This is a structure process. The idea is to have a chatbot where a user can formulate a tree structure (and its relationship with another one) in natural language or some other type of (less structured) input (like an Excell sheet).

Most often, these types of specifications start from Excell sheets, that already are the result of a collaboration process of stakeholders. These processes currently take a lot of time and are not always easy to understandable for users. For instance, the specification of the eFTI data set already takes more than a year and its presentation to end-users is quite bad. Of course, this all depends on tools, but improvements are required.

Thus, a chatbot is expected to make several steps redundant and rapidly generate the required output. Users must be able to validate the output, which may require some type of tree viewer and/or document (on top of a SHACL file for configuring a node). Having such a chatbot can help adoption of the capabilities, since users will not be aware of any semantic technology and don't have to learn new tools.

---

[1] ERA vocabulary (europa.eu)

[2] https://saref.etsi.org/saref4auto/

[3] For example, Geonames, the EU Knowledge Graph, and the Kadaster KG.

Functionally, different (related) tree structures are identified (see also the FEDeRATED architecture); they are of importance to data transformation and are therefore listed here:

- Business activity – a tree structure specification of all data that can be shared for a business activity. Transport and transhipment are examples of business activities. This is basically the end state of a business activity, i.e. all data that can be shared for successful completion of an activity.
- Service – a tree structure specification of all data that can be shared in the context of a service for a business activity. A multimodal supply chain visibility service is an example of a service. This tree structure represents the end state of a service.
- Event or interaction – a tree structure of data that can be shared according to the specification of a service. Examples are an order – and an ETA event.
- State – a concept of 'service' specifying all data that must be shared according to the service specification. State data can also be shared, for instance representing a business document like an eCMR or eB/L.
- Profile – a selection of data capabilities of a single organization for a business activity. A profile contains only those concepts, associations, data properties, and value constraints that support a single organization.

The current version of the Service Registry supports the specification of 'event' or 'interaction', but the tool can also be used to specify the other structure. However, the current version of the tool is not able to validate consistency between the various tree structures. For example, the tree structure a service for a business activity must be contained in the tree structure of that business activity. A service tree structure is a subset or the complete business activity tree structure. These rules are defined by the architecture.

2.4 Data transformation

Data transformation is about matching two models, constructing the intersection of those models, and support the transformation of data sets according to the intersection.

The Semantic Adapter is a capability for data transformation. The current version of the Semantic Adapter is configured by Semantic Treehouse. The Semantic Adapter handles the data transformation between a company-specific internal format and triples in terms of the shared ontology. Semantic Treehouse provides a process that guides a user to construct such a (simple) mapping, serialized as RML[4], that can be used to appropriately configure the Semantic Adapter. An LLM based ontology matching module could potentially assist a user in constructing these mappings (e.g. based on natural language processing). This may result in a specific matching module of Semantic Treehouse

[4] RDF Mapping Language (RML)

potentially supporting more complex matches. Whether RML will still support these complex matches requires further study as will be discussed hereafter.

Data transformation is the core problem for interoperability amongst heterogeneous IT systems of different organizations. The FEDeRATED semantic model can assist as intermediate in constructing matches for data transformation. Solving this problem with LLMs makes data sharing easy. It currently takes a lot of time and effort, also for testing settings with peers. Especially large organizations have addressed the issue by constructing an intermediatory file, the in-house file, that supports all types of implementation guides of standards. LLMs can help in constructing this file structure, which is reflected by 'profile', and matching it with internal IT systems.

When these models are ontologies, this is called ontology matching. Applying matching requires that any two models have the same technical representation, for instance OWL or Turtle. Thus, whenever models have a different technical representation, the first step is to apply the same representation. Since the FEDeRATED semantic model is technically represented by OWL/Turtle, this representation is applied for matching.

Transformation of data sets requires to process input data with its syntax and transform it to output with potentially another syntax using the FEDeRATED semantic model as an intermediate structure (see next figure).

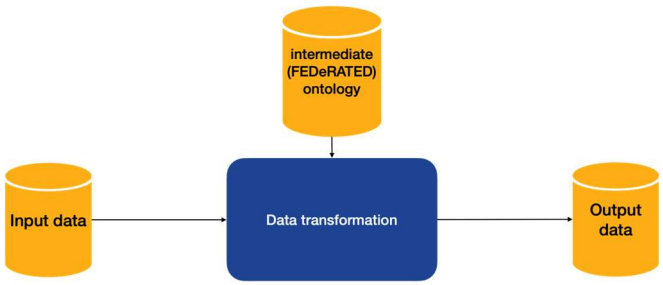


Figure 3 – data transformation in its context.

These types of data transformations can be complex and lead to data loss. Simple matches are a one-to-one match where an input element is directly matched to an output element. Complex operations are splitting a single element into multiple ones. Combining, the reverse operation, is simple. Other matches may require processing. Simple processing is hash calculation or summation to provide totals. These types of processing operations are not always reversible, meaning that receiving a sum does not allow a recipient to see the individual parts that compose the sum if these parts are not shared.

Data loss is where the output data does not have a concept or data property for specific input data. Not all input data can be transformed into output data.

Therefore, we can distinguish different strategies for ontology matching, namely:

- Model matching – two semantic models (an IT backend systems model and the FEDeRATED semantic model) are matched with each other. Where the FEDeRATED semantic model covers (logistics) data sharing, any internal model may have

additional functionality, e.g. to support humans in their daily work according to internal processes and procedures). This matching may not necessarily lead to success, due to these different functions of both models.

- Tree structure matching – a subset of the FEDeRATED semantic model is matched with the model of an IT Backend System. The tree structure represents a data set specification. It contains relevant concepts, associations, data properties and value constraints for those properties. It is represented by SHACL.

  The functionality of a tree can differ, depending on the way it has been specified as explained before. The following options are feasible:

  - Profile matching – the assumption is that matching a profile with an internal data model is feasible, since a profile contains concepts, associations, and properties (with value constraints) that reflect a business activity (like transport of containers) of an organization. This can result in a single API or so-called inhouse file between a node and an IT Backend System.
  - State matching – certain states can probably be matched with an internal data model, where the assumption is that this matching must cover the mandatory elements (concepts, associations, and data properties) of a state. State matching is considered relevant for matching for instance an eCMR or order data set with internal IT systems.
  - Event matching – like state matching this is also considered to be feasible, but only for those events that are supported by an internal IT system. Thus, if an IT system only supports ordering and no visibility, visibility events cannot be matched. When matching an event, it can be associated to an API or so-called inhouse structure.

Business activity – and service matching can also be considered but are like profile matching with the exception mandatory elements of a business activity – or service tree structure must be supported.

In practice not everyone will have a node. In practice, which implies that a single node must be able to support semantic web – and other standards (like XML and JSON) for sharing data with peers. This implies that any matches between a data model of an IT back-end system may have to be matched with a data sharing standard implemented by a peer via the FEDeRATED semantic model. Thus, two matches must be combined into one: between a data model and a standard.

Second aspect for consideration is the implementation of these matches. Currently, RML is used to configure the so-called semantic adapter. Most probably, RML does not have all functionality required for complex matches and transformations. Different options can be considered, for instance the creation of data transformation software, extensions of RML, or new types of configuration files.

A third aspect of data transformation is the current approach: design time matching results in runtime configurations or transformation software. In the longer term, LLMs may support runtime matching and transformation, like language translation software.

2.5 Data assessment

This is about formulating a query to the infrastructure in natural language and transforming that query into the applicable semantic web standard, namely SPARQL. SPARQL is about data creation, update, and retrieval. At a SPARQL endpoint, new data can be included, or data can be retrieved (and updated). In case of data creation or update, this is already represented by tree structure specification of events or interactions (see before).

A SPARQL can be simple, like retrieval of all data properties of a concept based on querying the identification of that concept, or complex. A complex query requires the specification of the required output, i.e. the tree structure of the output (see before). This tree structure can be specified and published as some type of standard, e.g. the data requirements for the eFTI Regulation can be represented by a tree structure when querying an identification, for instance a document (eCMR) identification or the identification of a truck to retrieve its safety status. It can also be an ad hoc query, where the tree structure of the required output can be formulated for once only use (of course these queries can be stored for future use).

Another example of query complexity are conditions related to a query. For instance, a query like ‘container track’ must have conditions like the movement from a location up to another location. In this type of query for a container movement from China to a European port, all relevant locations in China are given but only those that are relevant in the movement to a European port. Thus, not all locations are given in China that relate to the container entering China and moving to a certain location for stripping and later stuffing at another location.

Both simple and complex query formulation like the examples illustrate, may possibly supported by LLM based chatbots. This will ease the adoption of capabilities and semantic technology, since a user may not have be familiar with technology and/or is limited to domain knowledge (e.g. knowledge of logistics operations).

2.6 Assessing multiple data sources with a query

A query may have to be decomposed into queries as formulated in the previous section, where each of these individual queries provides input to the results of the main query. The individual queries may also be executed in a particular sequence. This is about queries where links are not stored by a node of a data user, but additional links may have to be retrieved.

An example of a query is the validation of a truck driver for transporting and handling dangerous cargo. Various checks can be performed in parallel: is the driver an employee of the carrier he/she represents, does the driver have a valid driver's license, does the carrier have permits for transporting dangerous cargo, is the truck safe to transport the dangerous cargo (state of health of the truck), and has the driver a license for transporting and handling dangerous cargo. These queries are partly in parallel (employee and driver's license can be validated in parallel) and have dependencies (knowing the carrier is required to ask for a permit and validating this with the issuer of the permit). The result of this example is 'true' or 'false'. As soon as one of the conditions is 'false', the result is 'false'. However, it may still be worthwhile to execute all individual queries, for instance to detect if a permit is still valid. These types of complex queries may be replaced by VCs at a later stage, thus simplifying them.

Another example is where data of a data holder is required for retrieving data of another holder. For instance, if a supervising authority wants to know the content of a container, a carrier is not able to provide this content. However, a carrier may provide its customer and the location where the container was loaded for transport. The customer might know the content or provide a link to another stakeholder, e.g. a forwarder providing the identification of a shipper. We don't consider the existence of a regulation by which a supervising authority is able to retrieve the necessary data, we only consider the technical capabilities.

There are several ongoing projects, such as Comunica by Ghent University and FedX by Ontotext that attempt to solve this federation problem. These are not LLM based chatbots, but they explore and implement a query strategy. LLMs as such are not expected to automatically answer these complex queries, since they are not (yet) familiar with data holders.

In case these data holders are known to the infrastructure, including the data they can provide and the conditions for accessing the data, such LLM based algorithms might be applicable for answering complex queries. In the meantime, an LLM chatbot may support a user by including the potential data holder to which a query can be posed, whereas, in the previous section, the query was always posed to the data holder that provided the link to a data user.

3 Ontology generation

Although FEDeRATED already has a semantic model, other application areas may still require one. Therefore, in a broader context, the first potential application area of LLMs is in the generation of ontologies from natural language texts.

Ontology generation is about development of ontologies from any textual sources. It can potentially be applied for extending the functionality of the upper ontology. It aims to extract concepts and associations within a domain from a set of (textual) sources and combines these into a comprehensive ontology. Manual ontology construction is a labour-intensive process involving lots of discussions and compromise between domain experts, conceptual modellers, and IT specialists. The goal of ontology generation is to improve this process in terms of speed and accuracy.

Research into automatic ontology generation has a (relative) long history, pre-dating the advent of LLMs with its inception in the early 2000s, so this section can be generalized to ontology generation in general. We identify a significant challenge in the field: the technical accuracy of the generated ontology models. Via desk research, we did not find any ontology generation method that to has successfully extended beyond the TRL 1-2 of fundamental or applied research. The literature survey lacks an indication that this will significantly change soon.

We additionally want to remark that the use case for fully automatic ontology generation has not yet been clearly identified. For example, it is unclear what the consequences of numerous small ontologies would be, since a single qualitative ontology may be desired.

While completely automatic ontology generation may be unfeasible, technological methods can still assist the people that craft the ontology. There is ongoing research on several subtasks that can contribute to ontology generation. These tasks include, but are not limited to, knowledge extraction, keyword extraction, relation extraction, triple construction, and schema construction. These are tasks that can potentially be executed by an LLM, as described in the LLM and KG white paper[5] on opportunities and challenges, authored by representatives of several universities and research institutes.

Summarising, we find that the research on automatic ontology generation is still in the stage of fundamental academic research, regardless of whether the research applies an LLM. So, we recommend that I&W should monitor ongoing research and discuss which problem automatic ontology generation would solve, if any. Now we recommend that

[5] https://arxiv.org/pdf/2308.06374.pdf

manual ontology generation should be employed, so it is and was a valid decision to manually construct the FEDeRATED ontology as well as developing other ontology models.

4 Ontology alignment & matching

A potential application of LLMs is in its application for ontology alignment and matching, where matching is relevant for **extending the functionality** and **data transformation** (see section 2). In ontology alignment and - matching, the goal is to identify similarities and difference between ontologies or data models. The functionality of a solution results in matches and differences and can thus be applied as follows:

1. **Ontology alignment** aims to combine the functionality of two complementary ontologies into one, by identifying common concepts and data properties in both ontologies. The goal is to create a formal specification how the two ontologies relate to each other, such that both can be used in conjunction with one another. The result is a join of these two ontologies. This is about **extending the functionality** of the upper ontology with new functionality.
2. **Ontology matching** is the specification of the intersection of two ontologies, where preferably the intersection covers both ontologies if they have the same functionality. The intersection can be simple (one-to-one matching) or complex (based on a formula). The goal is to identify all identical concepts between two ontologies. Ontology matching is applied for **data transformation**.

As explained in section 2, the process of ontology alignment and matching is usually performed manually. It is known to be complex and labour-intensive because it requires the user to have in-depth knowledge of the source and target ontologies, and of linked data in general. The task is therefore usually performed by the ontology engineer, whereas it should eventually be taken up by a user without expert ontology knowledge.

Ontology matching is necessary for data transformation as explained in section 2.

4.1 Automatic ontology alignment & matching

The demanding nature of ontology matching and - alignment has led to numerous endeavours to aimed at reducing it, such as manual tools to increase user-friendliness of the process and statistical methods to automatically generate mappings. Research into automatic mappings goes back well into the early 2000's (Hu, 2006)[6], and thus predates the advent of LLMs by a large margin. A recent example is the thesis by Majeed Mohammadi [1][7], which focuses on the use of neural networks and text mining (NLP-Natural Language Processing) for ontology alignment. The general conclusion is that these

[6] https://link.springer.com/chapter/10.1007/11926078_22

[7] https://repository.tudelft.nl/islandora/object/uuid%3A7d8ac519-f3f7-425f-82ce-1df481bc1c34

technologies were not yet satisfactory in terms of accuracy, user-friendliness, and potential time saved. LLMs represent a new technique that could potentially score better on alignment and matching benchmarks than previously used algorithms and techniques.

These ontology matching and - alignment technologies never made a difference between model – and tree matching. The Ontology Alignment Evaluation Initiative (OAEI) always addressed the issue by generating a new ontology from a subset of an existing ontology and aligning that new ontology with the existing one. Whether this generated ontology is a new model or a tree structure is implicit.

Since the rise of LLMs, the potential for ontology alignment and matching has been recognised, but the amount of (peer-reviewed) work is still small. We identify two types of LLMs that can be used for this goal: general purpose LLMs and fine-tuned LLMs dedicated to e.g., ontological alignment and matching.

4.1.1 General purpose LLMs

General purpose LLMs are off-the-shelve available and can be used for a range of applications. All large language models in principle have the potential to generate alignment and mappings. The other deliverable lists some of the most well-known LLMs. The quality of the outcomes is, however, highly dependent on the model, the prompt engineering employed, and on the use case it is applied to. In He et al. (2023)[8] the authors conclude that application of LLMs for ontology alignment is a promising direction but point out that this is highly dependent on the prompt design. They managed to place the code[9] of a preliminary investigation online.

Dean Allemang, semantic web expert and co-author of Semantic Web for the Working Ontologist, documented several tests with ChatGPT in his blog[10]. Allemang shows ChatGPT's capability to produce an initial mapping, but his expert knowledge was required to assess and correct the model's output. The prevalence of this blog-post shows that potential research may not have finished the extensive peer-review process.

4.1.2 Fine-tuned LLMs

General-purpose LLMs can thus already provide some starting points for an automatic mapping, but an LLM's performance can improve after it is fine-tuned for a specific task. Therefore, we can expect the quality of general purpose LLMs to be surpassed by the LLMs that are specifically tailored to RDF or SPARQL. Further research is required to fine-tune an LLM (by training) in ontology alignment and matching in general or to do the same for the FEDeRATED ontology. The latter will give a trained LLM that can be applied (and improved

---

[8] https://arxiv.org/pdf/2309.07172.pdf

[9] https://github.com/KRR-Oxford/LLMap-Prelim

[10] https://medium.com/@dallemang/llms-closing-the-kg-gap-29feee9fa52c

by training) in the context of capabilities for supply and logistics. The latter is preferred in the context of FEDeRATED, thus also becoming a base for a more generic approach.

An example of a model that is tailored to ontology alignment, is BERTMap[11]. It uses transformer-based language model BERT to predict aligned classes based on the textual knowledge of ontologies. It is part of DeepOnto[12], a python package for ontology engineering with deep learning. However, this mapping tool is based on an older generation of LLMs, again showing how much time it takes to create an LLM-based application of Linked Data technology, indicating that we may expect significant improved contributions in the future. The applied science perspective and technical knowledge of TNO can explicitly contribute to these further improvements.

4.1.3 Additional work

Piest et al. claim to have developed an architecture for creating schema mappings similar to ontology matching ("Smarter interoperability based on automatic schema matching and intelligence amplification", *Jean Paul Sebastian Piest, Lucas Onno Meertens, Johan Buis, Maria Eugenia Iacob and Marten van Sinderen*, *https://ceur-ws.org/Vol-2900/WS2Paper4.pdf).* They call their approach intelligence amplification: a "symbiotic" collaboration between humans and AI to facilitate the decision-making process, rather than a manual or a fully automatic process. Therefore, the paper is not about an ontology matching technology, but rather about setting up a process to do it semi-automatically in an efficient way.

The prototype described in section 4 of the paper uses an easy-to-implement machine learning tool from Microsoft. Perhaps this component could be replaced by an LLM in the future. Thus, in the context of the *technical* question of ontology alignment/mapping based on LLMs, the proposed solution is not (yet) relevant. Regarding the cooperation between humans and machines, as also addressed in our experiment, this paper does endorse the important of symbiosis.

4.2 Conclusion

Summarizing, we find that that automatic ontology alignment and matching is at TRL 3-4. Several prototypes have been developed and tested, but a clear success story is lacking although the potential has long been recognised. The advent of LLMs is expected in vision papers to further improve the effectiveness of automatic alignment and matching, but empirical peer-reviewed reproducible papers to support this idea are lacking as of December 2023, which may be caused by the extensive procedure of publishing academical work.

---

[11] https://ojs.aaai.org/index.php/AAAI/article/view/20510

[12] https://github.com/KRR-Oxford/DeepOnto

Furthermore, the issue of LLM based alignment and matching has never been approached from the perspective of data sharing. A generic approach seems to be made of matching models, whereas in data sharing, data sets always have a tree structures (e.g. XML and JSON based structures).

Whilst LLM based ontology matching and alignment tools are not available (yet), some tool suppliers state to have included a chatGTP-4 API for matching with other structures. For instance, Altova states to include chatGTP-4 in their MapForce product (AI-powered Data Integration | Altova). How this is done and what it contributes is yet uncertain, but it is promising to see efforts are being made.

4.3 Next steps

One of the issues with applying LLMs for ontology matching (& alignment) is their training set. LLMs must be familiar with ontologies (and SHACL and potentially SKOS, since these standards are applied to represent format constraints) and with matches. Highly advanced (NLP) algorithms will be applied in those LLMs. These algorithms must be trained with data in such a way they improve. Where ontologies will remain stable, these cannot improve an LLM for matching. Matches need to be provided as part of the training. A matching chatbot must be able to instruct an LLM and support a human in validating matches. The latter could be like tooling developed for XSLT (XML Style Language Transformation).

Our proposed approach would be to explore potential according to the following steps:

1. Input data to semantic data – transformation of input data with a tree structure (e.g. XML or JSON data) into semantic data.
    a. Tree structure generation via alignment – try to match a tree structure representing for instance an XML data set (e.g. an XSD) and align it with the FEDeRATED semantic model. This generates a representation of the tree structure in the semantic model.
    b. Tree structure matching – generate an ontology of the tree structure of input data, e.g. an ontological/SHACL representation of an XSD, and match this one with the generated tree structure of the previous step. This results in a matching.
    c. Data transformation – generate and test a data transformation environment for the previous match(es). Test data sets should be available for the input data of which the tree structure was generated.
2. Semantic data to output data – the same as the previous, but the reverse transformation based on a matching.
3. Input – to output data via the semantic model – combine the previous two steps into one and generate data transformations between input and output data. This provides the capability to interface with existing (guides of) standards.

4. From design – to runtime – explore opportunities for implementing the previous steps at runtime. This will require proper training since it requires no human intervention. It must work.

The recommendation is to start with simple data structures like events specified by XSDs and/or JSON and repeat them with gradually more complex structures like transport orders. This requires not only data structures for input, but also data sets.

The steps 1 to 3 must be supported by a chatbot interfacing with an LLM. The chatbot may have to be developed, also to assist a user in validating the output of each step at design time, before deploying a solution in runtime.

In the alignment step, new concepts or properties may be identified that are not yet part of the FEDeRATED semantic model. These should only be included in the FEDeRATED semantic model (via alignment) if they cannot be expressed otherwise by the model. For instance, some concepts may seem new, but may also be specified by queries on the model. This is the case of for instance ‘itinerary’ of a ‘transport means’ or ‘equipment-container’, which can be called ‘voyage’ (of a vessel), ‘trip’ (of a truck), ‘track’ (of a container).

5 Data assessment – LLM query chatbot

This is about assessing the state of the art of LLMs for the generation of SPARQL queries. As with ontology alignment and matching, engineering SPARQL queries requires a certain level of expertise. A user must know the structure of the ontology, be familiar with its concepts and the domain, and be proficient in the SPARQL query language. All to be able to formulate a SPARQL query.

The promise of LLMs in this context lies in lowering the level of required expertise and improving the efficiency of query writing. OpenAI has highlighted SQL generation as a prime example of applying GPT, of which can see SPARQL generation as a variation (Sequeda, 2023). If an LLM can translate a natural language question to a SPARQL query, a user does not need to be fluent. Additionally, the LLM may function as an improved coding assistant. Several triple store providers, such as Open Link Software[13] and Stardog[14], and researchers have trained LLM chatbots for SPARQL generation. Therefore, this application of LLMs in the semantic web has TRL 7.

We have additionally conducted an experiment to investigate the level of expertise required for operating such a SPARQL chatbot. Both novice as well as expert users have been requested to apply LLMs for SPARQL query generation. Both the quality of the chatbot generated queries, as well as the trust users put in the responses of the LLM are evaluated.

5.1 State of the art

LLMs are already widely in use by software developers to generate software code. These developers use an LLM as a chatbot and validate and paste the generated code into their software. This greatly improves their productive. Since SPARQL is a type of software code, the expectation is that this type of LLM application is or will become available. The state of the art depends on the use of an LLM for this type of code generation.

There is one type of application of LLM chatbot technology for SPARQL generation. This is the LangChain python package[15]: a framework for developing applications powered by language models that we also use in our experiment described in the next section.

---

[13] OpenLink Software: Home (openlinksw.com)

[14] https://www.stardog.com/

[15] https://python.langchain.com/docs/get_started/introduction

Particularly relevant in this context, is the pipeline for translation of natural language questions to SPARQL queries, addressed in a dedicated graph query class[16].

One of the initiatives for SPARQL generation with an LLM chatbot is the Voicebox tutor[17] by company Stardog. It makes use of the LangChain python package. It is an ontology assistant that can generate SPARQL and apply it on an underlying knowledge graph, based on the natural language input of a user. As of November 2023, it has been released to an initial set of users as a prototype.[18]

Unfortunately, the prototype release of Voicebox was not yet available for an experiment, so we were not able to assess the quality compared to the open source LangChain itself.

The company Ontotext[19], which develops the GraphDB triple store, has also provided a ChatGPT integration into their software[20]. However, their integration does not cover SPARQL generation. Instead, they enable a user to add a natural language question in their SPARQL, which is then sent to ChatGPT. The ChatGPT response because of the query on GraphDB is returned to the user together with any additional query results. Employing this integration already requires SPARQL expertise, so we cannot consider this a case of LLM-based SPARQL generation.

5.2 Experiment

We conducted an experiment to investigate limitations and required knowledge for applying LLM chatbots for SPARQL generation. We created a survey, with the aim to identify the trust in the LLM, the experience with using it as a tool, and potential improvements. The experiment was performed with seven TNO employees, three of whom have no knowledge of SPARQL and four have extensive knowledge.

The underlying research question of the experiment was “which level of expertise does a user already need to have to effectively leverage the SPARQL generation LLM in their current state with the FEDeRATED ontology?”. The level of expertise can be divided into different fields of expertise: FEDeRATED ontology -, semantic web technology -, and logistics expertise.

The hypothesis at the start of the experiment was: users still need a high level of SPARQL expertise to effectively leverage the LLM for a faster or more accurate workflow. The LLM generated SPARQL queries where hardly ever without mistakes, so the user needs the expertise to know the results they aim for.

---

[16] https://api.python.langchain.com/en/latest/chains/langchain.chains.graph_qa.sparql.GraphSparqlQAChain.html

[17] https://www.stardog.com/blog/stardog-voicebox-makes-building-data-models-dead-simple/

[18] https://info.stardog.com/demo-day/llm-powered-stardog-voicebox-and-knowledge-graphs/2023-08-16/watch

[19] https://www.ontotext.com/

[20] https://www.ontotext.com/company/news/graphdb-10-3-harness-the-power-of-chatgpt/

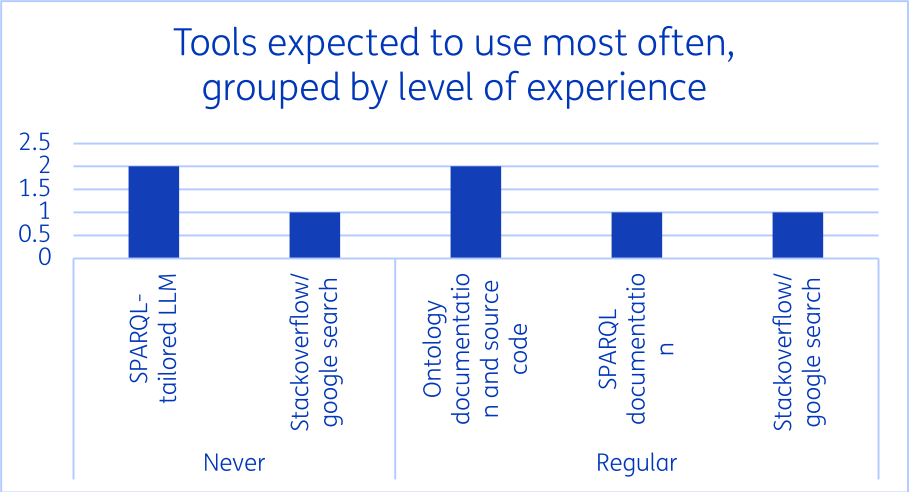
The experiment starts with an assessment of base knowledge via a survey. This assessment and the experiment are detailed hereafter.

5.2.1 Expectations

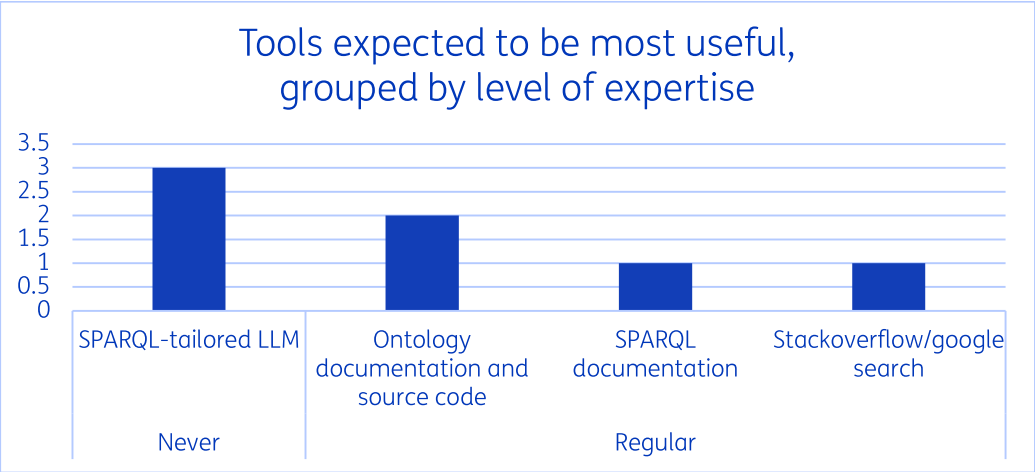
The survey contained a section about the respondents' background and their expectations and gives room to reflections about required expertise. This provides input as to the level of expertise of the participants and is relevant to assess trustworthiness and quality of the results. For instance, more experienced users require a different functionality to improve their workflow compared to novice users.

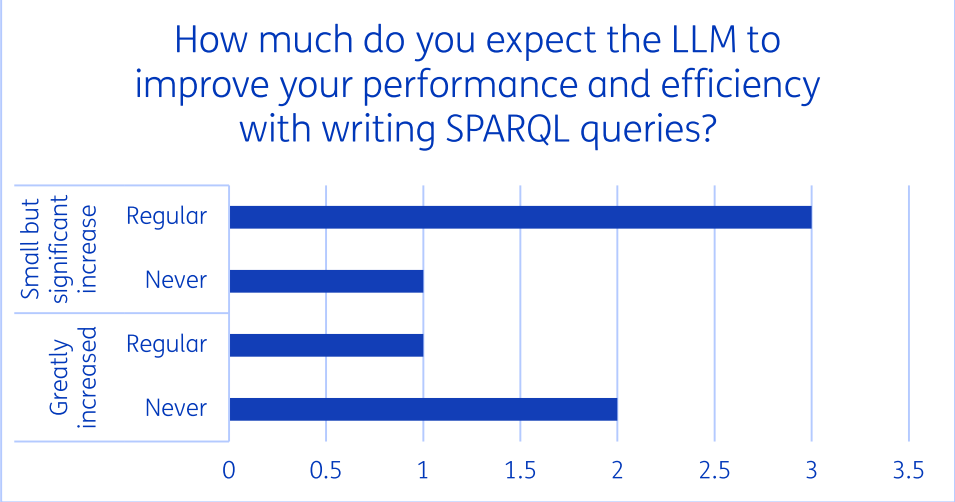
The respondents were requested to describe their expectations of the tools, in terms of expected use and expected helpfulness. Inexperienced users expect to use the SPARQL LLM and Stackoverflow the most, whereas experienced users expect to use the ontology documentation, SPARQL documentation, and Stackoverflow. The inexperienced users have higher expectations of the improvement the LLM can bring than experienced users.

Rank the following tools in order of how often you expect to use it (1=most often used during experiment, 5= least used during experiment)



Rank the following tools in order of much you expect these tools to help you during the process. (1=most useful input, 5= least useful input)





5.2.2 The experiment

The participants in the experiment were asked to write four SPARQL queries and were free to use any tool at their disposal to help them in the process. They were also provided with an implementation of an LLM (see figure 1), where they could write a query in natural language, and receive a translation in a SPARQL query in the FEDeRATED ontology. This would function similarly to VoiceBox (see before)

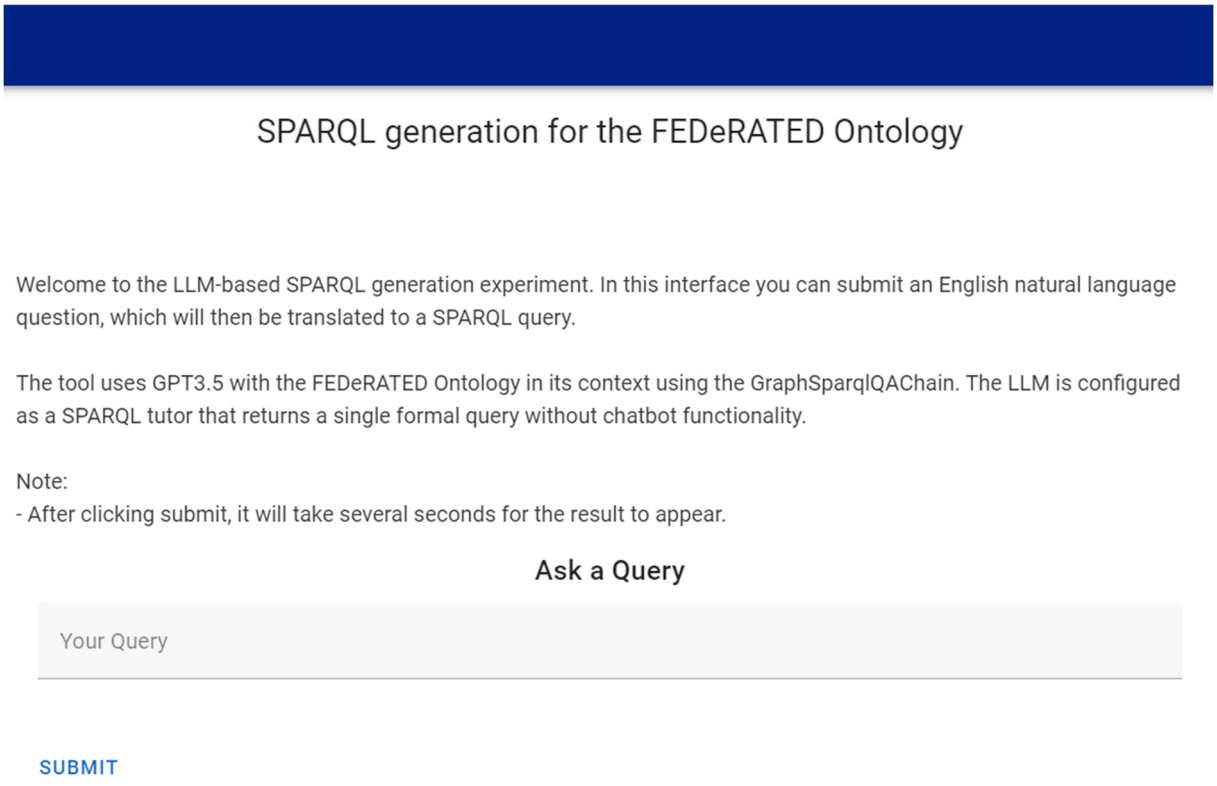


Figure 4 - SPARQL generator.

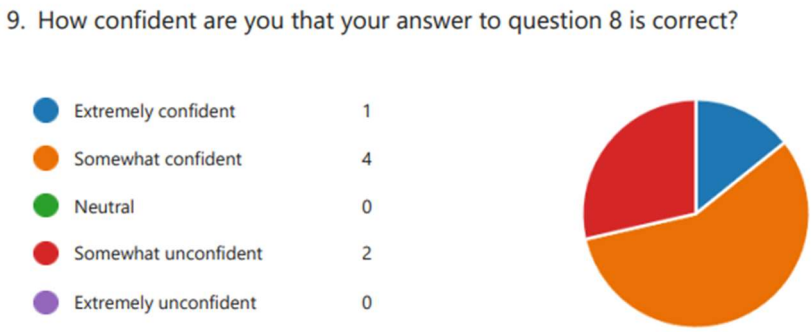
The open source LangChain project was used for our SPARQL generation experiment. The SparqlQAChain[16] is prepared to generate a SPARQL query and to run it on an underlying knowledge graph. The LLM is to function as a SPARQL generator that purely returns the generated SPARQL query as a instructed result. GPT3.5 was deployed with a context of 16k in the backend, because a smaller context was unable to fit the FEDeRATED ontology. GPT3.5 has been trained on data until 2021 and thus did not already "know" the ontology. We had to prune off some highly detailed classes to fit the context.

The experiment sketched a scenario in which the participant takes the role of a customs officer who monitors vessels arriving at the port of a European port. Some background on the FEDeRATED ontology is included by highlighting the use of events and giving some examples of their application. The participants were asked to formulate four SPARQL queries, ranging from easy to increasingly complicated. These queries are given hereafter, including a feedback by the participants.

Q1 First of all, you are interested in all vessels that are planned to arrive to get a general picture of the situation at the port. Write a query that selects those vessels, including their weight and identification number.

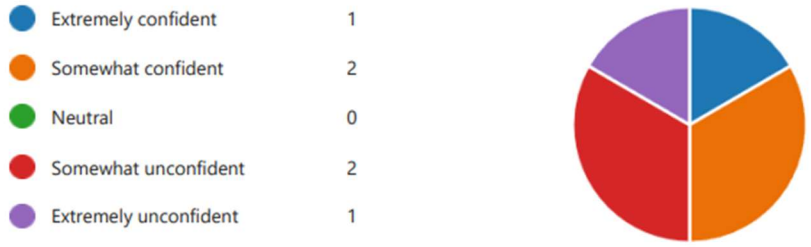
This first question was intended to be simple for both the LLM and the human to generate or construct. Most participants were (somewhat) confident in the result of their query because the various results of the LLM looked credible. In some cases, the results indeed were credible, but the participants lacked the expert knowledge to differentiate between query structure 1 and query structure 2 as shown in the following table. The former query structure lacks the differentiation between an event and the vessel involved in the event. One participant, who answered to be expert at SPARQL, was able to identify the problem with query structure one.

Query structure 1	Query structure 2
?vessel rdf:type event:ArrivalEvent . ?vessel event:hasDateTimeType event:planned . ?vessel event:weight ?weight .	?event rdf:type fed:ETAEvent . ?event fed:involvesDigitalTwin ?vessel . ?vessel rdf:type fed:Vessel .



Q2 Receiving a stream of planned arrivals, you remember that your database contains events from around the globe instead of just those for a European port. Since you are employed by a port authority, you are only interested in the events taking place in that port. Adapt the query accordingly. The following figure shows the confidence in the generated query.

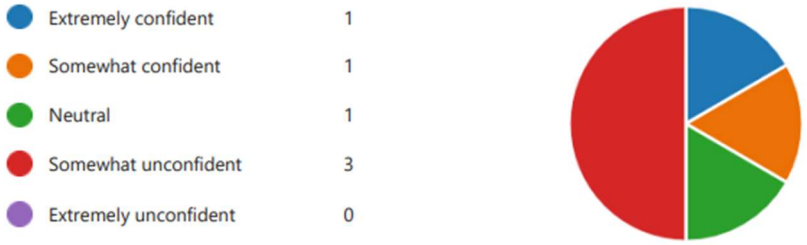
11. How confident are you that your answer to question 10 is correct?



Q3 By now you have a nice picture of the arrivals of vessels at a port. You want to narrow it down by selecting only the vessels that arrive during the coming week.

The common factor of the previous query and this one is that the participants were to apply additional filters to the first query. The filters concern the port location and the time window. The correctness of the answers was very dependent on the correctness of the participant's answer to the first query. The quality of the filters added to the query was high, indicating that the LLM properly generates a syntactically correct query addition, but seems to lack a proper grasp of the ontology structure.

13. How confident are you that your answer to question 12 is correct?



Q4 Having a clear picture of what will happen during the coming week, you can pinpoint some vessels that may be suspicious. You are particularly interested in vessels that have visited South America and will depart for Germany. Adapt the query to select vessels that have previously departed from South America, will arrive during the coming week in a European port, and will depart at a later moment to Germany.

Query 4 was intended as a question that could be asked by an operator in a real use case, whereas queries 1-3 were instead intended as test cases for the participants to test their usage of the LLM. The only participant that was extremely confident with their answer didn't use the LLM, but opted to use their expert knowledge of the ontology to write the

query themselves. The different responses produced by the LLM vary wildly in their accuracy.

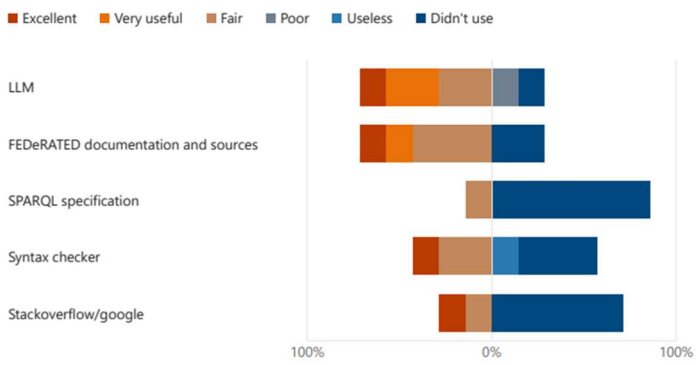
15. How confident are you that your answer to question 14 is correct?

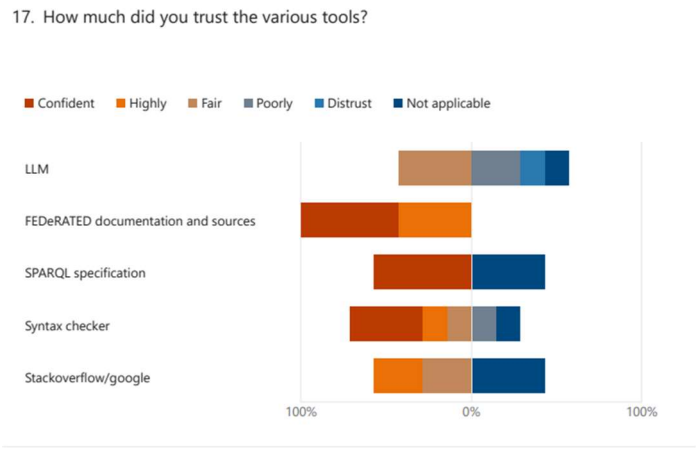
It is very clearly noticeable that as the complexity of the queries progresses, the respondents' confidence with their responses decreases. In the final query, even the daily SPARQL users felt extremely unconfident about the correctness of the query. The only confident respondent is one that not only is familiar with SPARQL, but also has in-depth knowledge of the FEDeRATED ontology.

5.2.3 Reflections

In the last part of the experiment, the respondents were asked to reflect on their experience in comparison with their expectations expressed at the beginning of the experiment. The LLM and the ontology documentation are considered the most useful, whereas the ontology and syntax checker are deemed most trustworthy. Novice users indicated they found the LLM especially useful, whereas experienced users particularly used the FEDeRATED documentation.

16. How useful did you find the various tools at your disposal?





5.2.4 Evaluation of the experiment

A result of the experiment is that novices, despite their limited prior knowledge of SPARQL, demonstrated a notable improvement in query formulation with the assistance of the LLM. Their feedback about their trust in the model revealed scepticism about its performance, as well as inability to adequately assess the outcome. Multiple respondents namely indicated that they looked for the names of concepts in the ontology documentation, to check or alter the model's output.

Experts were able to provide more detailed feedback about the model's performance. For example, someone indicated that the model's performance was not consistent: giving incredibly good outcomes for one query and nonsensical ones for others. Some of the complex queries posed challenges beyond the experts' expertise, leading to an inability to generate answers even with LLM assistance, indicating the difference between ontology expertise and FEDeRATED ontology experience. Feedback from experts indicated that while the LLM provided valuable suggestions, certain complex or nuanced queries remained difficult for both the model and the experts to address adequately.

TNO employees participating in the experiment are all familiar with LLMs and the underlying technology, understand the potential hallucinations, and know not to simply trust its outcomes. They indicate their trust in the LLM as negative, but according to their responses in usefulness, they were able to work around the lack of trustworthiness. The scepticism these respondents possess, was adequate and necessary at this stage, because the performance of the LLM did not merit a higher level of trust.

The quality of the answers participants provided, as well as the SPARQL queries that were generated by the SPARQL-LLM, were on average incorrect, especially the more complicated were incorrect. So, a human expert must remain in the loop to assure the correctness of the generated queries. Although the leverage of an LLM can soften the learning curve for a novice user, the usage of an LLM does not decrease the level of expertise required.

5.3 Conclusions

Summarising there are special LLM based chatbots for SPARQL available as open source. Since these are chatbots, they still require human validation and can thus only be applied at design time. However, after validation queries can immediately be deployed at runtime.

Since human validation is still required, a certain expertise level of semantic standards and ontologies is required. The experiment illustrated that an LLM chatbot for SPARQL generation can also be applied for learning. Users with limited knowledge of SPARQL were improving when using an LLM.

The following research is recommended:

- Market assessment. Monitor and validate the progress of LLM chatbots for SPARQL generation, for instance by integrating them with the triplestore of the node.
- Training chatbot. Develop a training chatbot by which a user gradually learns to formulate and validate queries and gets acquainted with the FEDeRATED ontology. This can assist person with technical – and domain (logistics) skills.
- Logistics Chatbot. Develop an LLM chatbot for query formulation that can be applied by domain experts to rapidly retrieve data for decision making. These are experts without technical knowledge but have knowledge of logistics planning or risk assessment. There are potentially different chatbots for different domain experts.

  At a later stage, such a logistics chatbot can be embedded in runtime applications. This requires more validation of output by users.

A chatbot for domain experts, either applied at design – or runtime, is expected to hide any details of ontologies and technology, which results in a challenge to validate the output. How is a domain expert able to trust the output of such a chatbot? It requires parallel use of existing technology and a chatbot.

6 Summary and next steps

6.1 Summary

This report identifies the potential of LLM in a data sharing capabilities with semantic web standards and – technology, as developed by FEDeRATED. When applying LLMs, trust with respect to the generated output is important. LLMs are trained with historic data sets, that are not always up to date. Furthermore, the concept of 'chatbot' is relevant: a means for interaction by a user and validating the output of an LLM. LLMs like chatGPT have a single line interface, although there is (recently) the capability to rapidly configure chatGPT for new chatbots. This latter feature has not be exploited; it became available at the end of this study.

The following opportunities are identified for applying LLMs (chatbots) in a semantic data sharing infrastructure:

- Extending functionality – joining two ontologies via common concepts and properties. This is like ontology matching.
- Data set specification – specifying all types of tree structures for data sets with the semantic model.
- Data transformation – specifying the intersection of two ontologies or tree structures for ontology matching as basis for data transformation.
- Data assessment – the ability to formulate queries without prior knowledge of semantic web standards, in this case SPARQL.

Besides generating ontologies from textual descriptions and other sources, which was not considered useful for an application like FEDeRATED (but can be in other application areas), the findings were as follows:

- Ontology alignment and matching – R&D for applying LLMs is yet to be started in this area. Past developments are based on other types of algorithms like NLP/text mining. This addresses the requirements for ontology alignment and data transformation identified above.
- Query generation – chatbots are becoming available, but still require technical expertise and knowledge of the semantic model. Query generation addresses simple and complex queries (complex queries require a filter in a SPARQL query). A query federation strategy is not yet supported by an LLM chatbot.

As part of query generation, the requested output is generated by the query according to its filter. This cannot be specified by for instance a tree structure. Tree structure generation

by an LLM chatbot is not yet investigated but might be required for LLM based ontology matching.

Another LLM or a custom-tailored LLM (chatbot) would have to be developed for a more recent ontology, such as the FEDeRATED ontology, to be included in the training process.

6.2 Next steps

Future improvements in an LLM-based ontology alignment and matching chatbot (or runtime algorithm) lie in its training and gradually exploration of capabilities from a simple start. LLMs must have knowledge of both ontologies (and SHACL and SKOS) representing domain knowledge of data and matches. Initial matches must be made for training an LLM. A chatbot needs to have the capabilities to support a user in formulating their requirements and validating the LLM output, where validated output (i.e. validated matches) can be used for training and improving an LLM.

Tree structure generation can be developed as component of an LLM ontology matching chatbot to explore capabilities for matching ontologies or tree structures (‘named graphs’) of two ontologies.

With respect to SPARQL generation, additional functionality to the one offered by existing LLM SPARQL chatbots can be explored, for instance a training chatbot. Such a chatbot would assist a human in getting acquainted with semantic standards and – technology, but could also assist users in getting familiar with domain ontologies like the one developed by FEDeRATED.

Another proposal is to develop a domain specific chatbot, e.g. one to support a logistics user. Such a specific chatbot would hide technical complexity and assist domain experts in formulating their queries. This could lead to various user-centric chatbots, all providing different perspectives on the same domain. For instance, targeting officers require another chatbot than logistics planners.

User validation is of the uttermost importance for all chatbots, since LLMs may still hallucinate and give inconsistencies. These chatbots must also support ‘explainability’, i.e. assisting users in validating the LLM generated output. Chatbots will keep the ‘human in the loop’.

Finally, LLM, or more generic AI, can of course have many other additional functions in a data sharing infrastructure at business process level. These generic applications all require data semantics. A common approach developed by FEDeRATED and supported by a data sharing infrastructure with LLM-based chatbots for querying, ontology alignment, and – matching contributes to implementation of these AI solutions.

Annex - References

Papers

He, Y., Chen, J., Antonyrajah, D., & Horrocks, I. (2022, June). BERTMap: a BERT-based ontology alignment system. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 5, pp. 5684-5691). https://arxiv.org/pdf/2112.02682.pdf

High-Level Expert Group on Artificial Intelligence Set Up by the European Commission, Ethics Guidelines for Trustworthy AI, 2019._Error! Hyperlink reference not valid.

Hu, W., & Qu, Y. (2006, November). *Block matching for ontologies*. In International Semantic Web Conference (pp. 300-313). Berlin, Heidelberg: Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/11926078_22

Ahlem Chérifa Khadir, Hassina Aliane, Ahmed Guessoum, (2021) Ontology learning: Grand tour and challenges, Computer Science https://www.sciencedirect.com/science/article/pii/S1574013720304391

MD Rashad Al Hasan, R. et al. (2022). *SGPT: A Generative Approach for SPARQL Query Generation From Natural Language Questions.* https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9815253, https://github.com/rashad101/SGPT-SPARQL-query-generation

Mohammadi, M. (2020). Ontology Alignment - Simulating Annealing-based System, Statistical Evaluation, and Application to Logistics Interoperability. Delft.

Pan, J.Z., (2023) Large Language Models and Knowledge Graphs: Opportunities and Challenges *https://arxiv.org/pdf/2308.06374.pdf*

Reyd, S., Zouaq, A., & Diallo, P. A. K. K. (2023). *A Comprehensive Evaluation of the Copy Mechanism for Natural Language to SPARQL Query Generation.* arXiv preprint arXiv:2304.07772. https://arxiv.org/abs/2304.07772

Sequeda, Juan, Dean Allemang, Bryon Jacob, (2023) "A Benchmark to Understand the Role of Knowledge Graphs and Large Language Model's Accuracy for Question Answering on Enterprise SQL Databases", arXiv preprint arXiv:2311.07509, https://arxiv.org/abs/2311.07509.

Blogposts

Blogs by Dean Allemang, author of Semantic Web for the Working Ontologist. All containing at least partially successful experiments to redo & gain experience.

- LLM's Closing the KG Gap
- Avoiding Hallucinations
- LLM, explain yourself!

Open source software

- Langchain SPARQL : GraphSparql QA Chain
    - GraphSparqlQAChain
    - https://python.langchain.com/docs/use_cases/graph/graph_sparql_qa
- SGPT code GitHub : SGPT
- BERTMap

Linked Data solution and service providers

Stardog

- LLM will Accelerate Knowledge Graph Adoption | Stardog
- Stardog | Webinar -LLM plus Knowledge Graph equals Stardog Voicebox

Companies working on their own SPARQL LLM:

- Release Notes — GraphDB 10.3.0 documentation (ontotext.com) Integrate ChatGPT to "explain SPARQL queries and results". "Pose generic questions to ChatGPT directly from SPARQL".
- Stardog Voicebox FAQ: How LLM, Generative AI, and Knowledge Graphs are the Future of Data Management | Stardog Created their own LLM, which they've finetuned on "thousands of example queries and data models". Uses the above langchain internally.
    - https://info.stardog.com/demo-day/llm-powered-stardog-voicebox-and-knowledge-graphs/2023-08-16/watch?submissionGuid=ece0fcf5-8177-4a18-9042-d4ed0c62d303
- Virtuoso Frequently Asked Questions (FAQ) (openlinksw.com) Seemingly less integration than the two above options.
- Introducing the OpenLink Personal Assistant | LinkedIn A personal assistant using knowledge-graph driven retrieval augmented generation to retrieve data from datastores via SQL, SPARQL or SPASQL.

Other companies (Topquadrant, Semantic Web Company, Franz Inc, Anzo) don't seem to have something significant in this area.