Enabling the Digital Transformation in Materials Science and Engineering: Leveraging Ontologies for Knowledge Representation, Provenance, and Text Mining

Von der Fakultät für Georessourcen und Materialtechnik der
Rheinisch-Westfälischen Technischen Hochschule Aachen

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

genehmigte Dissertation

vorgelegt von

Ahmad Zainul Ihsan, M. Sc.

Berichtende: Univ.-Prof. Dr. Stefan Sandfeld

apl. Prof. Dr. rer. nat. Ulrich Kerzel

Tag der mündlichen Prüfung: 02.09.2025

To Bapak and Ibu

Acknowledgements

This thesis is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 759419)

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Dr. Stefan Sandfeld, for his invaluable support and guidance throughout my PhD endeavor. I would like to thank him for allowing me to pursue the breakthrough research presented in this work and always being there whenever I had questions or doubts.

Second, I would like to express my gratitude to Dr. Said Fathalla, my PhD mentor and discussion partner. Your unwavering support and guidance were instrumental in advancing my research, refining my manuscript, and encouraging me to push my limits. I would also like to thank Dr. Volker Hofmann, the leader in the department of Concepts and Tools for Research Data Management. Your leadership and dedication in the department are qualities I truly admire.

Next, I am equally grateful to all collaborators: Prof. Harald Sack, Dr. Danilo Dessí, Dr. Mehwish Alam, Dr. Rossella Aversa, Dr. Mehrdad Jalali, Dr. Mirco Panighel, Fabian Kirchner, and Catriona Eschke, for helping me shaping and enriching my research experience. I had an excellent experience working with you and would love to collaborate again in the future.

I would like to acknowledge my colleagues from the Institute of Materials Data Science and Informatics (IAS-9), FZ Jülich; this journey would not be this colorful without you all—special shout-out to Chen Zhang and Kishan Govind, who always become my discussion partner about everything. Your support for this work is immeasurable.

Also, I would like to acknowledge my wife, Eva Afifah. Your constant help, understanding, and caring have been the foundation of myself to keep striving. Even when I have doubts and discouragement in this journey, your faith in me has been a source of comfort and strength. It has given me renewed spirit and determination to work towards my goals.

Finally, I am grateful to all my family and friends: Ibu Ike Kurniati, Bapak Dr. Odik Sodikin, Ayah Rodja, Mama Tati, Dr. Wildan Abdussalam, Dr. Hasbuna

Kamila, Grace Dina, Reza S. Thalfah, Zahwa, Ziyad, Zaid, Ahsan, Mas Faried, Mba Ulil, Dr. Anto, and Tsani, who have provided me moral and emotional support in my life. *Alhamdulillah...*

Abstract

The digital transformation of Materials Science and Engineering (MSE) is essential for accelerating the development of novel materials and enhancing understanding of the materials life cycle, encompassing raw resources, functional materials, engineered components, and beyond. This transformation entails integrating computational methods, data science, and artificial intelligence (AI) to advance the field of MSE. However, the heterogeneity of data formats, unstructured information, and the reproducibility crisis in MSE pose challenges to the effective management, reuse, and analysis of data. This thesis addresses these challenges by leveraging ontologies as the foundation for semantic data enrichment, facilitating knowledge representation, provenance documentation, and text mining.

The first contribution of this work is the development of the Dislocation Ontology (DISO), an ontology that represents the domain knowledge of linear defects in crystalline materials. The development of DISO was driven by the objective of facilitating data interoperability with other MSE-related data. DISO was aligned with the Elementary Multiperspective Material Ontology (EMMO) and Materials Design Ontology (MDO) to ensure interoperability. The ontology alignment efficiently represents the dislocation simulation data. Moreover, we present a real-world use case of representing discrete dislocation dynamics data as a knowledge graph (DisLocKG), which can depict the relationships between them. Additionally, DisLocKG is accessible, as we developed a SPARQL endpoint that offers considerable flexibility when querying DisLocKG.

Another contribution of this work is the PRovenance Information for MAterials Science (PRIMA) ontology, which was designed to document provenance information in MSE research, promoting data reliability, trustworthiness, and reproducibility. PRIMA was aligned with the Provenance Ontology (PROV-O) and the Platform Material Digital core ontology (PMDco) and was evaluated through use cases involving metallic biomaterial fabrication and microscopy data. Furthermore, this thesis presents a framework integrating Large Language Models (LLMs) and Semantic Web technologies to extract structured data from unstructured materials synthesis text. Free-text data is transformed into machine-readable formats such as JSON and further enriched semantically using an ontology.

The overarching objective of this work is to demonstrate an interdisciplinary approach that integrates MSE knowledge, Semantic Web technologies, and Natural Language Processing (NLP) to facilitate the digital transformation of MSE. The developed ontologies and knowledge graphs are pivotal in data enrichment and interoperability, ensuring that materials data adhere to the FAIR (Findable, Accessible, Interoperable, Reusable) data principles. The LLMs-based text mining framework provides a strategy for handling unstructured data in materials synthesis-related text, enabling the generation of linked data and knowledge extraction. Despite these advances, challenges persist in expanding DISO and DisLocKG to new use cases, refining data models, and improving long-context processing in LLMs.

Future works will entail the development of Application Programming Interfaces (APIs) for DisLocKG, extending PRIMA with computational modules, and exploring more efficient LLMs architectures for a more extensive range of text-mining applications. We envision a future where Semantic Web technologies and AI converge to enable machines to extract, process, and understand scientific data, ultimately driving the digital transformation in MSE.

Zusammenfassung

Die digitale Transformation der Materialwissenschaft und Werkstofftechnik (MSE) ist unerlässlich, um die Entwicklung neuartiger Materialien zu beschleunigen und das Verständnis des Lebenszyklus von Materialien zu verbessern, der Rohstoffe, Funktionsmaterialien, technische Komponenten und vieles mehr umfasst. Diese Transformation beinhaltet die Integration von Berechnungsmethoden, Datenwissenschaft und künstlicher Intelligenz (KI), um den Bereich der MSE voranzutreiben. Die Heterogenität der Datenformate, unstrukturierte Informationen und die Reproduzierbarkeitskrise in der MSE stellen jedoch eine Herausforderung für die effektive Verwaltung, Wiederverwendung und Analyse von Daten dar. Diese These befasst sich mit diesen Herausforderungen, indem sie Ontologien als Grundlage für die semantische Datenanreicherung nutzt und die Wissensrepräsentation, die Herkunftsdokumentation und das Text-Mining erleichtert.

Der erste Beitrag dieser Arbeit ist die Entwicklung der Dislocation Ontology (DISO), einer Ontologie, die das Fachwissen über lineare Defekte in kristallinen Materialien darstellt. Die Entwicklung von DISO wurde von dem Ziel angetrieben, die Dateninteroperabilität mit anderen MSE-bezogenen Daten zu erleichtern. DISO wurde mit der Elementary Multiperspective Material Ontology (EMMO) und der Materials Design Ontology (MDO) abgestimmt, um die Interoperabilität zu gewährleisten. Die Ontologie-Anpassung stellt die Versetzungssimulationsdaten effizient dar. Darüber hinaus stellen wir einen realen Anwendungsfall vor, bei dem diskrete Versetzungsdynamikdaten als Wissensgraph (DisLocKG) dargestellt werden, der die Beziehungen zwischen ihnen abbilden kann. Außerdem ist DisLocKG zugänglich, da wir einen SPARQL-Endpunkt entwickelt haben, der eine beträchtliche Flexibilität bei der Abfrage von DisLocKG bietet.

Ein weiterer Beitrag dieser Arbeit ist die PRovenance Information for MAterials Science (PRIMA)-Ontologie, die zur Dokumentation von Herkunftsinformationen in der MSE-Forschung entwickelt wurde und die Zuverlässigkeit, Vertrauenswürdigkeit und Reproduzierbarkeit von Daten fördert. PRIMA wurde mit der Provenance Ontology (PROV-O) und der Platform Material Digital core ontology (PMDco) abgestimmt und anhand von Anwendungsfällen im Zusammenhang mit der Herstellung von metallischen Biomaterialien und Mikroskopiedaten

evaluiert. Darüber hinaus wird in dieser Arbeit ein Rahmenwerk vorgestellt, das Large Language Models (LLMs) und Semantic-Web-Technologien integriert, um strukturierte Daten aus unstrukturierten Texten zur Materialsynthese zu extrahieren. Freitextdaten werden in maschinenlesbare Formate wie JSON umgewandelt und mithilfe einer Ontologie semantisch weiter angereichert.

Das übergeordnete Ziel dieser Arbeit besteht darin, einen interdisziplinären Ansatz zu demonstrieren, der MSE-Wissen, Semantic-Web-Technologien und Natural Language Processing (NLP) integriert, um die digitale Transformation von MSE zu erleichtern. Die entwickelten Ontologien und Wissensgraphen sind für die Datenanreicherung und Interoperabilität von entscheidender Bedeutung und stellen sicher, dass die Materialdaten den FAIR-Datenprinzipien (Findable, Accessible, Interoperable, Reusable) entsprechen. Das LLMs-basierte Text-Mining-Framework bietet eine Strategie für den Umgang mit unstrukturierten Daten in Texten, die mit der Materialsynthese in Zusammenhang stehen, und ermöglicht die Generierung verknüpfter Daten und die Extraktion von Wissen. Trotz dieser Fortschritte bestehen weiterhin Herausforderungen bei der Erweiterung von DISO und DisLocKG auf neue Anwendungsfälle, der Verfeinerung von Datenmodellen und der Verbesserung der Verarbeitung langer Kontexte in LLMs.

Zukünftige Arbeiten werden die Entwicklung von Programmierschnittstellen (APIs) für DisLocKG, die Erweiterung von PRIMA um Rechenmodule und die Erforschung effizienterer LLM-Architekturen für ein breiteres Spektrum von Text-Mining-Anwendungen umfassen. Wir stellen uns eine Zukunft vor, in der Semantic-Web-Technologien und KI zusammenkommen, um Maschinen in die Lage zu versetzen, wissenschaftliche Daten zu extrahieren, zu verarbeiten und zu verstehen, und so letztlich die digitale Transformation in der MSE voranzutreiben.

Acronyms

ABox Assertion Box
AI Artificial Intelligence
BFO Basic Formal Ontology
BIO Beginnning-Inside-Outside
BILOU Beginnning-Inside-Last-Outside-Unit
Bi-LSTM Bidirectional Long Short-Term Memory
CDO Crystalline Defect Ontology
CIF Crystallographic Information File
ChEBI Chemical Entitiy of Biological Interest
CPU Central Processing Unit
CSO Crystal Structure Ontology
CQ Competency Question
DCMI Dublin Core Metadata Initiative
DDD Discrete Dislocation Dynamics
DFT Density Functional Theory
DISO Dislocation Ontology
DisLocKG Dislocation Knowledge Graph
DL Deep Learning
DOLCE Descriptive Ontology for Linguistic and Cognitive Engineering
DOI Digital Object Identfier
ELN Electronic Lab Notebook
EMMC European Materials Modelling Council
EMMO Elementary Multi-perspective Material Ontology
EXPO The Scientific Experiment Ontology
FAIR Findable, Accessible, Interoperable, Reusable
FCC Face-centered Cubic
GPU Graphics Processing Unit
HTML HyperText Markup Language
IE Information Extraction
IRI Internationalized Resource Identifier
IUCR International Union of Crystallography
JSON Java Script Object Notation

JSON-LD Java Script Object Notation for Linked Data
LDA Latent Dirichlet Allocation
LLM Large Language Model
LoRA Low-Rank Adaptation
LSTM Long Short-Term Memory
MatKG Materials Knowledge Graph
MD Molecular Dynamics
MDO Materials Design Ontology
ML Machine Learning
MSE Materials Science and Engineering
MWO MatWerk Ontology
NLP Natural Language Processing
NER Named Entity Recognition
NERRE Named Entity Recognition and Relationship Extraction
NPLM Neural Probabilistic Language Model
ODP Ontology Design Pattern
OPM Open Provenance Model
OWL Web Ontology Language
PAV Provenance, Authoring and Versioning Ontology
PEFT Parameter-efficient fine-tuning
PMD Platform Material Digital
PMDco PMD Core Ontology
PRIMA PRovenance Information for MAterials science
PROV-DM PROV Data Model
PROV-JSON PROV Java Script Object Notation
PROV-O PROV Ontology
QLoRA Quantized Low-Rank Adaptation
QUDT Quantities, Units, Dimensions and Data Types Ontologies
RDF Resource Description Framework
RDF/XML RDF/eXtensible Markup Language
RDFS Resource Description Framework Schema
RE Relationship Extraction
RF Random Forest
RNN Recurrent Neural Network
SPARQL SPARQL Protocol and the RDF Query Language
SPO Subject, Predicate, and Object
SQL Structured Query Language
STM Scanning Tunneling Microscopy
SWAN Semantic Web Applications in Neuromedicine Ontology
TBox Terminology Box
TEM Transmission Electron Microscopy
TPU Tensor Processing Unit
Turtle Terse RDF Triple Language
W3C World Wide Web Consortium
XML Extensible Markup Language

List of Figures

List of Tables

Contents

Chapter 1
Introduction and State of the Art

Materials Science and Engineering (MSE) is undergoing a pivotal transformation mainly driven by advances in digitalization [1, 2]. This shift involves transforming MSE-related businesses and processes into the digital era by utilizing digitization, which converts physical information into digital data. Additionally, it employs digital technologies to handle, analyze, and turn the data into knowledge. Various national initiatives[1,2,3] envisage the digital transformation as a means to understand the materials life cycle, spanning from raw resources to operational components and beyond.

Digitalization in MSE refers to integrating digital technologies into all aspects of materials research and development [2, 3], including simulations, experiment design, materials preparation, materials characterization, materials fabrication, data analysis, and data management. A key component of this transformation is the combination of computational methods, experimental techniques, data science, and informatics.

Numerous studies highlight the effectiveness of digital integration in advancing materials research. The combination of computational methods, experimental techniques, and artificial intelligence has enabled breakthroughs such as "high-throughput" in materials design [4–6], high-throughput crystal plasticity simulation [7], and multiscale approaches, including concurrent frameworks [8].

Recent advancements in data science and computer science have further contributed to the progress of digital transformation by making Machine Learning (ML) and Deep Learning (DL) accessible through mature software libraries. The "democratization" of these tools, supported by comprehensive documentation and examples, has made them accessible to not only computer scientists, but also researchers from other fields, such as MSE. Coupled with significant improvements in hardware performance and the decreasing costs of Central Processing Units (CPUs) and Graphics Processing Units (GPUs), this has marked a turning point for the

[1] `https://nfdi-matwerk.de`

[2] `https://www.mgi.gov`

[3] `https://dice.nims.go.jp`

widespread adoption of not only advanced computational methods in MSE, but also advanced data analysis techniques leveraging ML and DL.

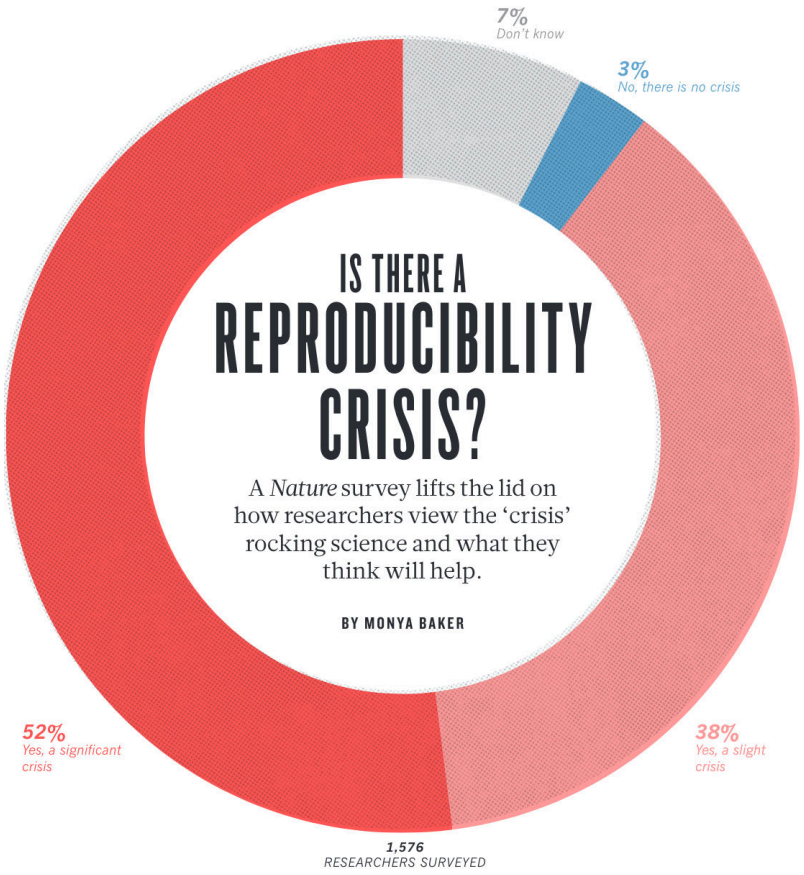


Fig. 1.1: The Nature survey in 2016 showed that the majority of researcher believed that there is a reproducibility crisis. The figure is taken from Baker [9].

However, with the rapidly growing volume of data being produced, challenges have emerged that highlight a disparity between the rate of data production and the ability to understand the data effectively [10]. One notable issue is the "reproducibility crisis," where many scientific studies cannot be reliably reproduced [11, 12]. This challenge was brought to attention by a survey conducted by Nature in 2016 with total respondents of 1576 researchers. The survey revealed that most respondents believed there was a reproducibility crisis (cf. Fig. 1.1) [9]. Furthermore, 70% of researchers reported having attempted and failed to reproduce other researchers' work, and more than half were unable to reproduce their experiments.

The complexity and interdisciplinary nature of MSE further add to these challenges, especially in its digital transformation. Drawing from disciplines such as chemistry, physics, and engineering, MSE poses a significant challenge to its digital transformation. Each discipline contributes its unique viewpoint, specialized terminology, and distinct methodologies, producing data in various formats and structures. For example, much of the data generated from these disciplines is stored in unformatted and unstructured natural language text [13], such as MSE scientific papers, books, and lab notebooks. Consequently, it is difficult for machines to read and understand, while also hindering humans from efficiently finding, understanding, and reusing the information.

As MSE progresses toward digital transformation, it is essential to develop new approaches for effectively handling the ever-expanding volume of data. While Artificial Intelligence (AI) and ML have significantly advanced in data analysis and materials design, there are still gaps in handling and mining the MSE data. These challenges limit the ability to integrate, analyze, and reuse the vast amounts of everyday data generated.

1.1 Towards the Digital Transformation in the Dislocations Domain

One area within MSE that benefits from digital transformation is the study of plastic deformation in metals and crystalline materials, primarily attributed to dislocations. The concept of the dislocation was brought to the surface to reconcile the difference between the theoretical and the experimental study of the applied shear stress required to plastically deform a single crystal. In 1926, the Frenkel model [14] describing the shear strength in crystal lattice computed in a theoretical study revealed a significant magnitude order difference than the experiment observation. Not until 1934, three scientists, E. Orowan [15], M. Polanyi [16], and G. I. Taylor [17], who worked independently, came up with the answer causing the discrepancy; they postulated that line defects, i.e., dislocations, can exist within crystal lattice and the movement of these defects at low-stress levels leads to deformation.

Understanding dislocations is crucial as they determine mechanical properties of crystalline materials (e.g., metals or semiconductors) such as strength, hardness, and ductility. For instance, materials engineers have discovered the strengthening mechanism of metals by studying the relationship between dislocation motion and the mechanical behavior [18]. By controlling the motion of dislocations in crystalline materials, materials engineers can build, for example, an airplane turbine blade that can withstand an operation temperature of $\sim 1000^{\circ}C$ and creep deformation caused by centrifugal forces while the turbine is rotating [19, 20].

Many efforts have been made to understand dislocation systems using several techniques and methods, including microscopy techniques [21, 22] and simulation methods [23, 24]. Recently, data-driven approaches have introduced new

methods and tools for analyzing and understanding dislocation systems significantly [25–30]. This transformation combines simulations, data mining, and experiments, making the digital transformation in MSE possible. However, without appropriate data infrastructure, this digital transformation often results in isolated and inaccessible data repositories, known as "data silos." In this regard, materials informatics is critical for addressing data silos by merging MSE and information technologies, e.g., the knowledge representation. This combination helps develop intelligent systems for exploring materials, discovering new material properties, and studying material phenomena. Specifically, in understanding dislocations, it requires consideration of various length scales data, making the representation of dislocation systems challenging.

In the past decades, several researchers have been involved in the knowledge representation of MSE through developing domain ontologies. One of the earliest materials ontology is developed in 1994, and it is the Plinius ontology. It is an ontology developed for describing ceramic materials covering the conceptualization of chemical compositions ranging from the single atom to complex chemical substances. Ashino [32] have developed the "Materials Ontology", an ontology describing substances, processes, environments, and properties. This ontology also has been used to exchange data between three different thermal property databases.

Another ongoing effort to establish semantic standards that apply at the highest possible level of abstraction is the Elementary Multi-perspective Material Ontology (EMMO)[4]. EMMO has been developed in the context of the European Materials Modelling Council (EMMC)[5] and provides a common semantic framework for describing material models, characterization, and data with the possibility of extension and adaptation to other domains. E.g., the application of EMMO in the mechanical testing domain was demonstrated in [33], and challenges of ontology alignment of level-domain ontologies and EMMO were addressed in [34]. However, EMMO currently contains only a small number of sub-domains. Furthermore, it does not include the domain of crystalline defects, particularly dislocation which is of great importance for materials scientists in the context of mechanical deformation of nano- and micometer sized specimens.

Apart from work related to EMMO, efforts also have been made to represent the domain knowledge of crystal structures. The authors of MatONTO [35] have developed a Crystalline Structure Ontology as a sub-module of MatONTO by means of mapping and re-engineering the terms from the Crystallographic Information File (CIF) dictionary [36]. Since CIF and the alternative CIF2 standard are published and distributed under the umbrella of the International Union of Crystallography (IUCR)[6], it serves as a *de facto* standard for this community. The authors of CIF in [37] also have further developed the STAR/CIF Ontology that is written in the mathematical symbolic script language called dREL [38].

[4] `https://emmo-repo.github.io`

[5] `https://emmc.eu/`

[6] `https://www.iucr.org/`

In the solid-state physics domain, Li et al. [39] have developed the Materials Design Ontology (MDO) which is an ontology covering knowledge in the field of materials design, e.g., with regards to ab-initio methods. MDO is developed to represent materials' data related to ab-initio calculations over disparate materials data repositories as Resource Description Framework (RDF) triples. While the work is related to the representation of crystalline material by means of the crystal structure, MDO does not represent data related to crystalline defects.

Related to the scientific data, many researchers paid a particular attention on developing ontologies to increase the semantic value of their data in different fields of science, such as physics [40], agriculture [41], and pharmaceutical science [42]. Specifically in the MSE field, several efforts have been carried out in creating ontologies representing materials-related domain data or semantically presenting actual materials data as knowledge graphs. Two examples of knowledge graphs existing in the domain science: Knowledge graphs with and without semantic web technology, including RDF, Web Ontology Language (OWL), and SPARQL Protocol and the RDF Query Language (SPARQL). Two examples from the latter are the *Propnet* Knowledge Graph [43] and the Materials Knowledge Graph (MatKG) [44].

Propnet is a knowledge graph enhancing materials properties data from the Materials Project [45] Repository[7]. It augments base properties data (e.g., lattice, basis, chemical formula, band gap, and total energy) resulting from the ab-initio calculation into derived properties, e.g., Debye temperature, bulk modulus, and shear modulus. The workflow and input for generating the augmented data are subsequently stored in the knowledge graph.

On the other hand, MatKG stores metadata from over 2.9 million materials science articles. This metadata includes abstracts, titles, keywords, and author data (e.g., name, email, affiliation, and ORCID). By accessing the MatKG, we can retrieve information such as the research progress and contributions to a novel material by multiple scientists and investigators

Another effort in the experimental materials science community that uses semantic web technologies is the NanoMine Knowledge Graph[8] [46]. It is a knowledge graph for polymer nanocomposite materials integrating diverse data from more than 1,700 polymer nanocomposite experiments. Moreover, the authors of the NanoMine knowledge graph have developed the NanoMine ontology[9], which is a backbone ontology to describe polymer nanocomposite experiments.

Despite significant efforts in ontology design across various scientific domains, a domain ontology specifically for dislocations in crystalline materials is still lacking. While there have been multiple efforts to represent the MSE domain, few researchers have tackled the challenge of semantically representing dislocations in crystalline materials. This gap is clearly noticeable in dislocation data, where, despite significant progress in applying semantic web technologies in MSE, the work for semantically representing this data is still missing. Addressing this gap is cru-

[7] `https://next-gen.materialsproject.org`

[8] `http://nanomine.org/`

[9] `https://github.com/tetherless-world/nanomine-ontology`

cial for increasing interoperability, and enabling advanced semantic querying and inference of implicit knowledge in materials science.

1.2 Preserving Provenance Information in Materials Science and Engineering Domain

Various measures addressing the reproducibility crisis have been studied in several fields, e.g., psychology [47], life science [48–50], and artificial intelligence [51–53]. Furthermore, the Findable, Accessible, Interoperable, Reusable (FAIR) data principles [54, 55] describe essential points to mitigate the reproducibility crisis. For example, the reproducibility can be achieved by first promoting data reusability by providing detailed provenance information [56, 57].

Provenance information, detailing a resource or data's origin, history, and derivation, is important for ensuring the reliability and trustworthiness of data [58]. In MSE, provenance information can include details about the materials used, the processes employed, and the identities of the researchers involved. Documenting this information can help researchers assess data quality and provide valuable context and transparency. For instance, in the correlative characterization method, a method combining multiple complementary measurement techniques, researchers can analyze materials at different length scales and gain a more comprehensive understanding of their structure, properties, and performance [59]. It integrates data from different measurement methods, linking features and properties across scales. In this regard, provenance information from each measurement method should be documented and stored in an interoperable way.

Documenting provenance with the help of formal knowledge representation through ontology enhances interoperability in materials experiments and simulations. Furthermore, it enables machines to read and interpret domain knowledge efficiently, thus facilitating the data integration from different sources and methods. In addition to that, ontologies ensure that provenance information is interoperable, allowing for seamless data exchange and interoperability between different machines and instruments. This approach not only boosts data management and analysis but also supports the principles of FAIR data by making provenance information more accessible and reusable across various platforms.

Several ontologies preserving provenance information have been developed. The Scientific Experiment Ontology (EXPO) [60] is an ontology that creates a proper understanding of generic knowledge regarding scientific experimentation, including design, methodology, and results representation. This formalization is comprehensive enough to capture the fundamental concepts for formalizing experiments in various domains. In the biomedical domain, the Semantic Web Applications in Neuromedicine Ontology (SWAN) [61] is an ontology that facilitates the exchange and integration of information related to biological researches. Furthermore, it supports the development of new applications in bioinformatics that rely on accurate and trustworthy information, e.g., alzSWAN [62].

Due to the numerous provenance ontologies within the semantic web community, the need to establish a standardized provenance data model arises. This standardized effort, which is also influenced by existing provenance data models (e.g., the Open Provenance Model (OPM) [63] and Dublin Core Metadata Terms[10]) and scientific workflows, resulted in PROV [58], a World Wide Web Consortium (W3C)[11] recommendation for a provenance information data model.

PROV provides several recommendations for capturing the provenance information of web resources, two of which are the PROV Data Model (PROV-DM) [64] and the PROV Ontology (PROV-O) [65]. PROV-DM is a provenance data model comprising six modular components, including entities and activities, derivations of entities, agents, responsibilities, bundles, properties linking entities, and collections. PROV-O formally represents PROV-DM encoded in OWL [66]. It allows for the representation, exchange, and integration of provenance information across various fields. Apart from OWL serialization, PROV-DM also has the Java Script Object Notation (JSON) called PROV Java Script Object Notation (PROV-JSON) serialization[12].

Extending PROV-O is viable, particularly to meet specific science domain needs. The Provenance, Authoring and Versioning Ontology (PAV) [67] is based on SWAN and PROV-O. PAV provides essential information for tracking web resources' provenance, authorship, and versioning. Another development is the REPRODUCE-ME Ontology [56] which represents provenance information in scientific experiments. It extends PROV-O and P-Plan [68] to describe scientific workflows in life sciences, biology, and microscopy domains.

A recent effort to establish the interoperable workflow by means of ontology is the PMD Core Ontology (PMDco) [69], a mid-level framework bridging the gap between top-level ontologies such as the Basic Formal Ontology (BFO) [70] or the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [71], and materials science applications. PMDco provides essential terms within an intermediate semantic layer, making it both easily understandable and usable. Furthermore, PMDco extends PROV-O to facilitate the representation and exchange provenance information across different systems and contexts.

However, as a mid-level ontology, PMDco lacks specific terms and relationships related to provenance information at the application level, such as experimental or computational workflows and data analysis lifecycles. Therefore, the development of a provenance-based ontology is necessary to fill this gap, and it is a crucial step for advancing MSE at the application level, particularly in areas such as materials synthesis, preparation, and measurement.

[10] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

[11] https://www.w3.org

[12] https://www.w3.org/submissions/prov-json/

1.3 Extracting Provenance Information in Materials Science and Engineering Texts

In the materials design, provenance-related (meta)data can offer insights into the origin of materials (e.g., materials used and materials generated by a materials synthesis), the equipment involved, methods employed in its development, and the identities of its researchers. Preserving provenance information is valuable for ensuring the quality and reliability of the materials' design. But, most of the metadata related to materials design is stored in unformatted and unstructured natural language text [13], such as materials science journal articles, books, and lab notebooks, making it difficult for machines to read and understand.

Fig. 1.2 illustrates an example of a typical MSE journal article, highlighting several key components, including text, tables (blue-colored boxes), and figures (yellow-colored boxes). In this work, we focus on extracting information from articles, specifically related to how materials are synthesized (as shown in the red-colored box). Parsing this information is often non-trivial [72, 73], posing challenges for researchers attempting to utilize this knowledge in experiment design or when understanding previous work. To address this, unstructured materials science text must be transformed into structured data formats, such as JSON, Extensible Markup Language (XML), or RDF.

Leveraging the Natural Language Processing (NLP) to structure scientific data in the materials science domain is still in its early stages compared to other domains, such as the biomedical [75, 76] and the chemistry [77–79] domain. The initial step in transforming the unstructured natural language text into linked data is by combining two NLP tasks: Named Entity Recognition (NER) and Relationship Extraction (RE). The former task involves identifying all instances of a specific type of thing within a text collection, e.g., materials entity in materials science journal articles. The latter is an NLP method extracting the relation between words in the natural language text.

The initial work of text mining in the material synthesis can be tracked by the work of Kim et al., 2017. The relationship between synthesized materials and reaction conditions is systematically studied in that work. In order to do that, the text mining process need to retrieve, classify, and parse materials science articles. Particularly in the latter stage, where authors parse the relevant synthesis paragraphs, the dependency parse tree extracts the relation between synthesis actions (e.g., dissolve, mill, and sinter). Furthermore, it extracts operating parameters to perform an action, such as the sintering temperature, the stirring speed, and the calcination temperature.

Several works have shown that transforming the material synthesis text into an action graph, a directed graph containing synthesis actions and synthesis parameters, is possible. Mysore et al. [80] presented a system for automatically extracting structured representations of synthesis procedures. These procedures describe experimental syntheses of inorganic materials. To achieve this, they leverage machine learning (e.g., Conditional Random Field) and neural network-based model

generators depends on the figure of merit of the constituting materials given by $zT = S^2 \sigma/\kappa T$. Here S, σ, and κ are Seebeck coefficient, electrical conductivity, and thermal conductivity, respectively. Mg_2Si and its solid solutions are very attractive due to their good thermoelectric performance in the mid-temperature region 500 – 800 K, environment compatibility, non-toxicity, abundance and low cost of materials [3-5].

Development of Mg_2Si based thermoelectric generators requires both good p- and n-type materials. For n-type, good thermoelectric properties have been reported for binary Mg_2Si with zT~1 [6, 7], and particularly for solid solutions with compositions around $Mg_2Si_{0.4}Sn_{0.6}$ reaching $1 \leq zT_{max} \leq 1.5$ [8, 9]. However, p-type $Mg_2(Si,Sn)$ has inferior properties in comparison to n-type, partially due to a less favourable band structure, lower dominant carrier type mobility and insufficient dopant activation [10]. Recently, there has been progress on fabricating complementary p-type materials and improved $zT > 0.5$ has been observed [9, 11-13]. Zhang *et al.* reported Li doped $Mg_2Si_{0.3}Sn_{0.7}$ synthesized by solid state reaction. This composition was selected due to double valence band convergence which is good for TE properties with $zT \sim 0.5$ [14]. Moreover, Gao *et al.*, prepared Li doped $Mg_2Si_{0.4}Sn_{0.6}$ by B_2O_3 encapsulation with $zT \sim 0.7$. However, the formation of MgO seems to be the cause of increasing thermal conductivity [12]. High energy ball milling has been employed to synthesize p-type $Mg_2(Si,Sn)$ reaching $zT > 0.5$ [11, 13]. This synthesis method is promising due to little or no oxidation or impurities, reduced Mg loss, and good dopant incorporation [15]. The obtained powders need to be compacted because the compaction process is important to obtain samples with good density. Its parameters such as sintering temperature and holding time affect the TE properties.

In this study, we prepared p-type Li doped $Mg_2Si_{1-x}Sn_x$ with x = 0, 0.2, 0.4, 0.6, 0.8, 1 by mechanical alloying followed by current assisted sintering. The synthesis parameters such as milling time, sintering temperature and holding time were varied to investigate the effects on TE properties. The optimal synthesis conditions were then applied to compositions with different Si:Sn ratio. Local surface mapping of the Seebeck coefficient proves that the synthesis of p-type materials for the whole compositional series is possible and furthermore allows us to correlate sample homogeneity with synthesis parameters and thermoelectric properties.

2. Materials and methods

The Li doped $Mg_2Si_{1-x}Sn_x$ with x = 0-1 solid solutions were synthesized by mechanical alloying, employing high energy ball milling (SPEX 8000D). The precursors (Mg turnings (Merck), Si (< 6mm, Chempure), Sn (< 71 µm, Merck) and Li granules with purity > 99.5%) were weighed according to nominal composition. No excess Mg was added as excess Mg can occupy the interstitial position and serve as an n-type donor [11, 16]. The desired elements were transferred into a stainless steel jar with a ball to powder ratio 1.6:1. All the procedures were conducted inside a glove box under Ar atmosphere to prevent oxidation and contamination. The elements were milled with constant speed for 4 - 20 h until fine and homogeneous powders were obtained. The obtained fine powders were transferred to a graphite die (Ø 15 mm) and sintered at 873 - 1073 K for 600 s by utilizing DSP 510 SE from Dr. Fritsch GmbH under vacuum condition (~10^{-5} bar), under sintering pressure 66 MPa with a heating rate 1 K/ s. The density of the obtained pellets was calculated using Archimedes' method. The detailed synthesis conditions are summarized in Table 1. The obtained pellets were characterized by XRD Siemens D5000 Bragg – Brentano diffractometer with a secondary monochromator, Cu-K_α radiations (1.5406 Å) in the range (2θ: 20° - 80°) with a step size of 0.01°. The microstructure and phase purity of the samples were observed by Zeiss Ultra 55. Moreover, functional homogeneity was investigated by a spatial mapping of the Seebeck coefficient at room temperature using a PSM [17, 18].

The temperature dependent electrical conductivity and Seebeck coefficient data was obtained by a 4 probe technique [19, 20]. The thermal diffusivity (α) of the pellet was measured by using Netzsch LFA 427 apparatus. The thermal conductivity (κ) was calculated using the relation: $\kappa = \alpha \cdot \rho \cdot C_p$ where ρ and C_p are density of sample and heat capacity. The C_p value was calculated from the Dulong-Petit limit for $C_v^{DP} : C_P = C_v^{DP} + \frac{9E_t^2 T}{\beta_T \rho}$, where E_t~ 2 x 10^{-5} K^{-1} and β_T ~ 2.07 x 10^{-11} Pa^{-1} are the linear coefficient of thermal expansion and isothermal compressibility, respectively. In the relevant temperature regime C_p increases from 0.582 J/gK to 0.605 J/gK. The measurements were performed under Ar and He from 300 – 698 K. The uncertainties of measurement for S, σ, and κ are ±5%, ±5% and ±8%, respectively.

Table 1 Synthesis parameters of Li doped $Mg_2Si_{1-x}Sn_x$ with x = 0-1

Composition	Milling time (h)	T_{sinter} (K)	t_{sinter} (s)	Density (g cm^{-3})
$Mg_{1.98}Li_{0.02}Si$	6	1073	600	1.92
$Mg_{1.95}Li_{0.05}Si_{0.8}Sn_{0.2}$	20	1023	600	2.27
$Mg_{1.98}Li_{0.02}Si_{0.6}Sn_{0.4}$	20	1023	600	2.57
$Mg_{1.98}Li_{0.02}Si_{0.4}Sn_{0.6}$	20	1023	300	2.99
$Mg_{1.98}Li_{0.02}Si_{0.4}Sn_{0.6}$	20	973	300	2.95
$Mg_{1.98}Li_{0.02}Si_{0.4}Sn_{0.6}$	20	973	600	2.99
$Mg_{1.98}Li_{0.02}Si_{0.4}Sn_{0.6}$	20	973	1200	2.98
$Mg_{1.98}Li_{0.02}Si_{0.2}Sn_{0.8}$	4	923	600	3.24
$Mg_{1.98}Li_{0.02}Si_{0.2}Sn_{0.8}$	20	923	600	3.18
$Mg_{1.98}Li_{0.02}Sn$	4	873	600	3.50

3. Results and Discussions

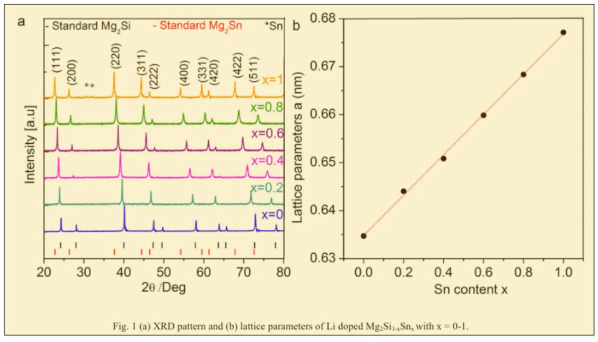


Fig. 1 (a) XRD pattern and (b) lattice parameters of Li doped $Mg_2Si_{1-x}Sn_x$ with x = 0-1.

The obtained pellets with relative density > 95 % except x = 0.4 (Table 1) were characterized by XRD. Fig. 1 (a) shows that all the peaks can be indexed according to the anti-fluorite structure Fm-3m space group of Mg_2Si and Mg_2Sn. The peaks gradually shift towards lower angles with increasing Sn content because Sn has larger ionic radius than Si. Within the detection limit of the XRD, all samples are single phase except Mg_2Sn, where minor peaks corresponding to elemental Sn can be found. The Sn peaks appear probably due to Mg loss. We do not observe the formation of MgO, which is often found in other synthesis routes [12, 21, 22] and detrimental for the thermoelectric properties [23]. The lattice parameter increases with increasing Sn content and rises approximately linearly, indicating that solid solutions are formed. The lattice parameter was calculated by Bragg's equation and approximately follows Vegard's rule. The results show good agreement with reference data [24].

Fig. 1.2: Components of a materials science journal article typically include text, tables (blue-colored box), and figures (yellow-colored box). The red-colored box highlights the "Materials and methods" section, where provenance information is documented. Furthermore, this section details the materials used for synthesizing a novel material, the equipment or instruments utilized in the experiment, and the specific actions taken, such as preparing and measuring the sample. The figure is adapted from Kamila et al. [74]

(e.g., Bidirectional Long Short-Term Memory (Bi-LSTM)) to extract the entity of words, i.e., NER task. Furthermore, dependency parsing is used to extract and relate events in the synthesis procedures resulting in a graph, i.e., RE task.

Another work generating material synthesis graph comes from Huo et al. [73]. It generates the material synthesis graph by classifying the text according to the method used, e.g., hydrothermal, solid-state, and sol-gel method. Latent Dirichlet Allocation (LDA) [81] and Random Forest (RF) [82] are used as the topic classifier. Moreover, to generate the graph, Markov chain construction is applied.

Even though several works related to graph construction structuring the natural language text have been done in the past five years, it is still left with several difficulties when applied to other downstream tasks. For instance, to set up a model learning to extract the entity of words, one should manually annotate the segment boundary encoding for tokens, such as Beginnning-Inside-Outside (BIO)1, BIO2, and Beginnning-Inside-Last-Outside-Unit (BILOU), which is not trivially done. Moreover, there is an overhead to process this segment boundary encoding before and after the training/fine-tuning process.

The recent development of Large Language Models (LLMs), the transformers-based language model (cf. Subsec. 3.3.4), has solved the difficulty of annotating the segment boundary encodings and combining two tasks of NER and RE named Named Entity Recognition and Relationship Extraction (NERRE). Recent work of Dunn et al., 2022 discovered that the NERRE task can be accomplished by employing a pre-trained large language model, GPT-3 [84], that is fine-tuned on about 500 examples of document completion to return structured data. By using LLMs on the NERRE task (LLM-NERRE), nearly any information extraction task can be done by specifying a new output schema or data model, and domain experts can easily construct their own models by simply reading passages and transcribing the type of output they want the model to produce. At the time of writing this thesis, the NERRE system developed by Dunn et al. [83] represents the state-of-the-art method for the information extraction.

However, there are two disadvantages that can be identified by using LLM-NERRE. First, the flexibility of using an arbitrary output schema or data model makes the difficulty to retrieve the information of the structured data. Thus, the data model needs to be designed properly following a standard, such that we can easily query the data that we are interested in. Second, GPT-3 is a closed source language model developed by OpenAI[13] and proliferating it to be used or fine-tuned to another downstream task requires additional cost. Nowadays, there are a lot of open source LLMs available that we can use for free, e.g. Falcon-7b, FLAN-T5, and FLAN-UL2. Furthermore, since the field of LLMs is rapidly evolving, with new models being developed within weeks or months, the LLMs mentioned, as of October 8th, 2024, are the most reliable open-source models for LLM-NERRE.

1.4 Summary of the Thesis

To summarize, we study four key areas to address the aforementioned challenges toward the digital transformation in MSE: The development of the Dislocation Ontology (DISO), Discrete Dislocation Dynamics (DDD) data enrichment, the development of PRovenance Information for MAterials science (PRIMA) ontology, and text mining on materials synthesis provenance.

[13] `https://openai.com`

1. **Development of the Dislocation Ontology.** DISO is designed to represent the concepts and relationships of linear defects in crystalline materials. DISO includes modules that describe scientific concepts, dislocation representations, and various simulation models within the dislocation domain. By providing a structured and standardized representation of dislocation data, DISO facilitates interoperability and enhances the ability to integrate dislocation data with other MSE-related fields.
2. **Discrete Dislocation Dynamics data enrichment.** DDD simulations are pivotal for understanding the behavior of dislocations in crystalline materials. These simulations generate extensive data that provide insights into plastic deformation mechanisms. However, managing and interpreting this data poses significant challenges due to its complexity of the (meta)data structure. In this thesis, we propose enriching DDD data using Semantic Web technologies. The enriched data, organized into a knowledge graph named Dislocation Knowledge Graph (DisLocKG), enables advanced semantic querying and inference of implicit knowledge and ensures data consistency.
3. **The development of PRovenance Information for MAterials science ontology**. PRIMA is developed to document the provenance information of research studies and their outputs in the MSE domain. Provenance information, including details about the materials, processes, and researchers involved, is essential for data reliability and trustworthiness. PRIMA is based on terms defined in the MDMC-NEP Glossary of Terms [85], and aligns with the PROV-O [65] and the Platform Material Digital (PMD) Core Ontology [69]. The broad applicability of PRIMA is demonstrated through use cases such as Scanning Tunneling Microscopy (STM) and metallic biomaterials fabrication.
4. **Materials synthesis provenance mining.** Another significant challenge in MSE is extracting and utilizing knowledge from unstructured data, such as materials science text, journal articles, and lab notebooks. This thesis presents a methodology for transforming natural text into structured or linked data using pre-trained LLMs fine-tuned on MSE synthesis text. The transformation involves NER and RE tasks to identify and structure relevant information. These procedures result in the semi-structured data, which is subsequently enriched by semantic web technologies and PRIMA. This approach enables the creation of linked data that can be queried semantically, facilitating machine interoperability and implicit knowledge inference.

1.5 Goal of the Thesis

The goal of this thesis is to introduce several key steps necessary for driving the digital transformation in MSE. By leveraging ontologies for knowledge representation, managing provenance information, and transforming unstructured text into structured data, this work promotes the integration of digital technologies into the MSE domain. These efforts subsequently support the development of a robust data

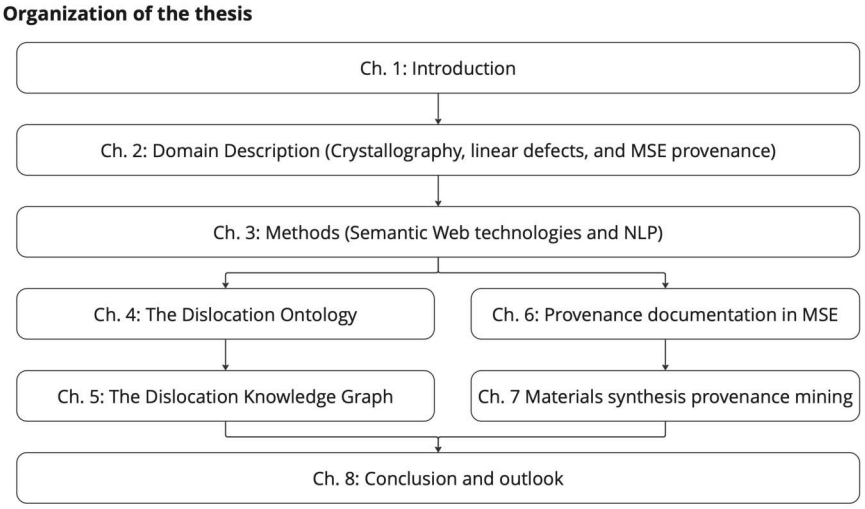


Fig. 1.3: The thesis begins with introductory and methodological chapters, followed by two parallel branches on knowledge representation: DISO (Ch. 4) and PRIMA (Ch. 6), each with a dedicated chapter demonstrating its ontology applications (Ch. 5 and Ch. 7, respectively). The thesis concludes with Ch. 8, summarizing findings and outlining future directions.

infrastructure, ensuring that materials data is FAIR. Ultimately, these contributions facilitate innovation and the discovery of novel materials, pushing the field toward digital transformation. Below we give an overview of the structure of the thesis.

In Ch. 2, we describe three scientific domains for which we want to represent their knowledge. These domains are the domain of crystallography, linear defects, and provenance information in an MSE study. Ch. 3 explains the methods used to facilitate the digital transformation in MSE: Semantic Web technologies and NLP. In the former, we utilize the Semantic Web technologies to develop DISO and PRIMA and serialize them in machine interoperable format. In the latter, we leverage NLP and the state-of-the-art LLM to extract the materials synthesis provenance information. In Ch. 4 and Ch. 5, we discuss the development of DISO and the application of DISO to generate the dislocation knowledge graph, respectively. Ch. 6 explains the development of provenance information ontology in MSE, known as PRIMA. The development of PRIMA involves collaboration between several researchers from various research centers, which fosters MSE community-based development. In Ch. 7, we demonstrate the combination of Semantic Web technologies and NLP to extract the materials synthesis provenance information for scientific texts found in the MSE domain. The final chapter, Ch. 8, discusses the conclusion and outlook of the thesis.

Fig. 1.3 illustrates the thesis structure along with the workflow of the thesis. It begins with Ch. 1, an introduction, and the subsequent chapters of Ch. 2 and Ch. 3, which are the domain description and methods, respectively. It then continues into two branches of knowledge representation, i.e., two parallel works on DISO and PRIMA in Ch. 4 and Ch. 6, respectively. Each branch has a chapter demonstrating the application of ontology. Ch. 5 illustrates the application of DISO in enriching the dislocation dynamics data, and Ch. 7 explains the application of PRIMA and Semantic Web technologies to extract the materials synthesis provenance information. Ch. 8 is the last chapter, where we conclude and envision the thesis and future work.

Chapter 2
Description of the Scientific Domain

This chapter briefly describes the relevant notions and concepts of line defects within the crystalline materials domain. Additionally, we discuss the relevant domain related to provenance information documentation in Materials Science and Engineering (MSE).

2.1 Representation of Crystalline Materials

Most of the metals and metallic materials have a crystalline structure, which implies that the atoms are arranged in a periodic structure with a high degree of symmetry. This periodic arrangement is at the basis of the *crystal structure* model, idealizing the physical concept of crystalline materials. For example, in Fig. 2.1 atoms are shown in an idealized manner as small spheres.

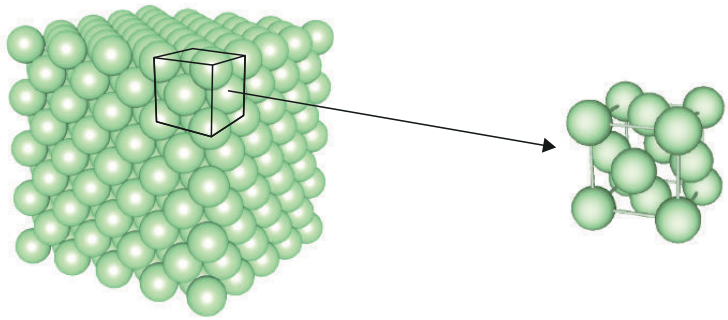


Fig. 2.1: Shown in the left panel, the crystal structure of face-centered cubic materials consists of a regular arrangement of atoms. On the right, a unit cell is shown.

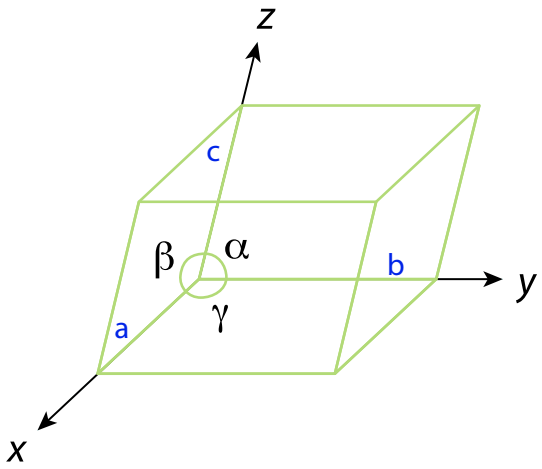


Fig. 2.2: The geometry of a unit cell is exactly defined through the three lengths, a, b, and c, and the three angles α, β, and γ.

The crystal structure is represented by the *lattice* together with a *motif*: the lattice is a mathematical concept of an infinite, repeating arrangement of points in space (3D), in a plane (2D), or on a line (1D), in which all points have the same surrounding and coincide with atom positions. The motif (or base) consists of an arrangement of chemical species, which can be atoms, ions, or molecules in crystalline materials. By putting a motif of one or more atoms at every lattice point, the crystal structure can be represented.

It is now possible to identify the smallest atom pattern that can be repeated along all spatial directions to cover the entire structure. This pattern is called a *unit cell*, shown as the black cube in Fig. 2.1. The lattice parameters of the unit cell denote the angles between the edges and the edge lengths.

Fig. 2.2 shows the six lattice parameters needed to characterize the unit cell: three lengths (a, b, c) and three angles (α, β, γ). These parameters also constitute the basis vectors in the crystal coordinate system; they are not necessarily mutually perpendicular. Unit cells are often classified based on the lattice parameters (cf. Fig. 2.3). For instance, the cubic system has $a = b = c,\ \alpha = \beta = \gamma = 90°$ and the orthorhombic system has $a \neq b \neq c,\ \alpha = \beta = \gamma = 90°$. Seven crystal systems are often ordered according to the increasing symmetry: cubic, tetragonal, orthorhombic, hexagonal, rhombohedral, monoclinic, and triclinic.

In the unit cell, we can also define *lattice points*, *lattice directions*, and *lattice planes*: A lattice consists of lattice points where the atoms, ions, or molecules are located (the leftmost cube in Fig. 2.4). The vector position of lattice points, $\overrightarrow{R}$, is described by the equation

$$\overrightarrow{R} = n_1\mathbf{a} + n_2\mathbf{b} + n_3\mathbf{c}\ , \tag{2.1}$$

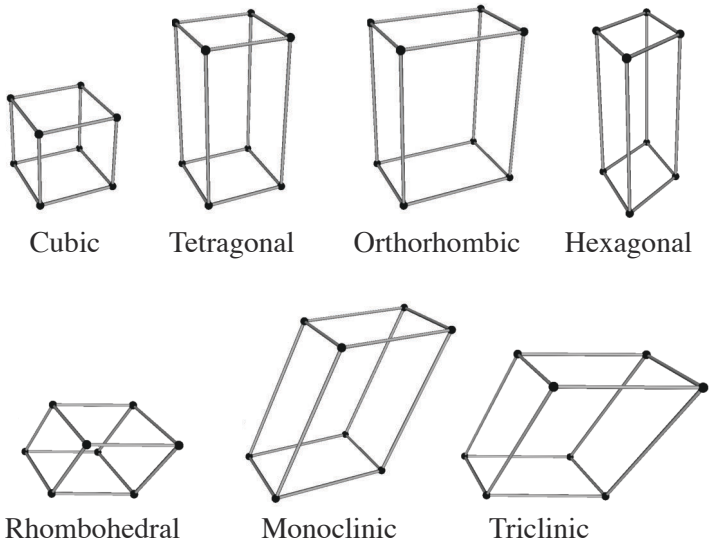


Fig. 2.3: The seven crystal systems. These seven crystal systems are also seven primitive Bravais lattices. Each of them only corresponds to a single lattice point.

where n_i are arbitrary integers and $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are basis vectors (pointing along the axes in Fig. 2.2) derived from the lattice parameters. As illustrated in Fig. 2.4, a lattice direction or lattice vector is a vector connecting two lattice points, whereas a lattice plane forms an infinitely stretched plane (characterized through a plane normal) that cuts through lattice points such that a regular arrangement of lattice points in the plane occurs.

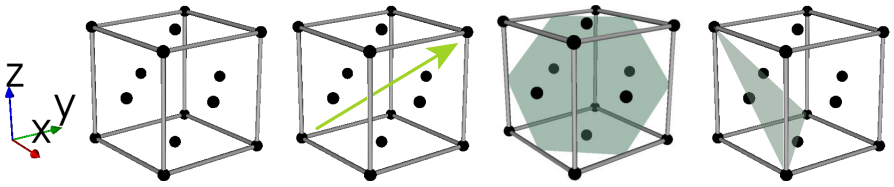


Fig. 2.4: On the very left, a unit cell is shown that corresponds to a "face-centered cubic" structure. The black points indicate the lattice points, i.e. the positions of the atoms. Second from left: a direction vector that connects to these lattice points. Lastly, two different lattice planes are shown.

The 14 *Bravais lattices* classify crystal structures based on their lattice arrangements and are divided into seven primitive and seven non-primitive lattices. A primitive lattice (denoted as **P**) is one in which each unit cell contains exactly one

lattice point, meaning there are no additional lattice points beyond those at the corners.

In contrast, non-primitive lattices contain additional lattice points at specific positions within the unit cell, referred to as centerings: *center*, *base*, or *faces* of the unit cell. There are three centering types: base-centered (**S**), body-centered (**B**), and face-centered (**F**). Among the crystal systems, the orthorhombic system is unique in having four Bravais lattice types (**P, S, B, F**), as illustrated in Fig. 2.5.

Fig. 2.5: Orthorhombic is the only crystal system with four centerings in its Bravais lattices (primitive, base centering, body centering, and face centering).

From the point of view of symmetry, all crystal structures are classified in 32 *point groups* and 230 *space groups*. A point group comprises symmetry operations, e.g., reflection, rotations, inversion, identity, and rotoinversion. A space group is then defined as a combination of Bravais lattice, point groups, and two other symmetry operations: screw rotation and glide reflection. Applying a symmetry operation makes the crystalline material structure indistinguishable/unchanged. While the point group describes only the symmetry of the macroscopic or finite object; thus, the Bravais lattice is negligible. The space group describes the complete spatial arrangement of a 3D periodic pattern.

2.2 Description of Linear Defects

In crystalline materials, atoms are not always arranged or positioned perfectly. Typically, different kinds of crystallographic defects lead to "disruption" of the local order in a material (in addition to thermal fluctuations affecting the atomic positions). A common type of such defect is the dislocation, which causes a strongly localized, tube-like region of disorder (highlighted by the dashed circle in the right panel of Fig. 2.6; this tube-like region stretches along the z-direction). This region contains the highly disordered dislocation core at the center. Further away from the dislocation core, the perfect lattice structure is restored, even though there is now a row of atoms shifted into the new position as indicated by the red spheres in Fig. 2.6.

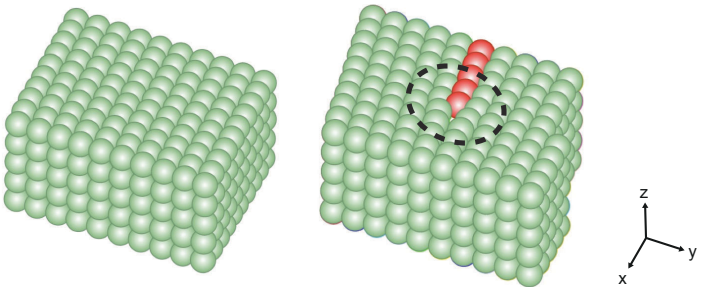


Fig. 2.6: The left figure shows the perfect order of atoms in a crystalline material. The right crystal contains a dislocation that destroys the local order of the crystal structure. Note that red-colored atoms only show the “irregularity”, and are not different atom types.

The “Burgers vector” of a dislocation can be defined through the “Burgers circuit”, as shown in 2.7. A Burgers circuit is an atom-to-atom path that is closed in a perfect crystal (left panel of the figure). The length of the path is given as multiple of the atomic distances in two directions. In the presence of a dislocation, a path of the same lengths would not be closed (right panel of the figure). The step-by-step procedure is as follows: We define a reference point C as the start point of the path. The line sense of the path is given by $\boldsymbol{\xi}$ assuming the "right-hand convention" (the thumb points along the vector $\boldsymbol{\xi}$ into the picture plane and the other fingers curl around that vector; their fingertips indicate the direction of the path). The symbol

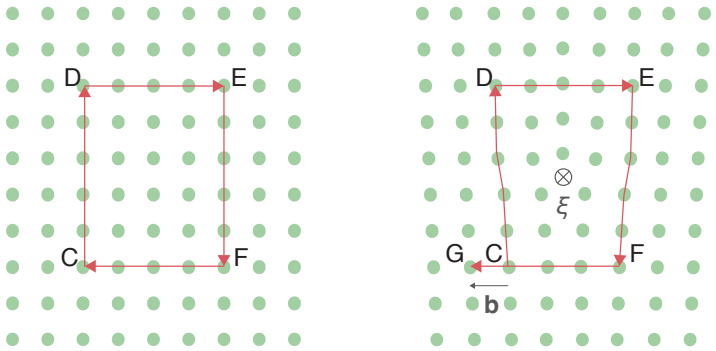


Fig. 2.7: The Burgers circuit in the crystalline materials. While the left panel has the Burgers circuit in the perfect crystal material. The right panel has the Burgers circuit around the dislocation in the crystalline materials. Due to the closure failure in the defective crystalline materials, we can define the Burgers vector, **b**.

$\otimes$ in the figure indicates that the vector $\boldsymbol{\xi}$ points into the picture plane. In the perfect crystal, this circuit goes up five atoms from the reference point, four atoms to

the left, five atoms down, and four atoms to the left to close a circuit at point C again. With the same reference point and the same number of atomic spacings as in the perfect crystal circuit, the same path can be traced in the crystal containing a dislocation. However, the circuit does not end at the same point as in the perfect crystal, but rather at point G as shown in the right panel of Fig. 2.7. The vector connecting the start point C with the finish point G is the Burgers vector $\mathbf{b}$.

There are two fundamentally different types of dislocations: "screw" and "edge" dislocation. A screw dislocation has a line sense parallel to its Burgers vector, $\boldsymbol{\xi} || \boldsymbol{b}$, whereas for an edge dislocation the line sense is perpendicular to its Burgers vector, $\boldsymbol{\xi} \perp \boldsymbol{b}$. Thus, Fig. 2.7 shows an edge dislocation.

In reality, screw and edge dislocations are extreme cases, while the most general case of a dislocation type in a crystalline material is a "mixed dislocation." This is a dislocation with the line sense, $\boldsymbol{\xi}$, neither parallel nor perpendicular to its Burgers vector, $\mathbf{b}$.

Since the atoms around dislocations are not positioned at the perfect lattice points, the lattice is distorted near a dislocation. This distortion results in a stress field in the crystalline material around the dislocation which is the reason why dislocations move: they try to minimize energy. In the context of plastic deformation, a dislocation is defined as the boundary of a slipped area within which atoms are displaced by the size of an elementary unit translation given by the Burgers vector.

In materials science, often the question arises on which "granularity level" a dislocation should be defined. Clearly, if we are interested in phenomena on the nanometer scale then we should resolve individual atoms (e.g., through high-resolution transmission electron microscopy or molecular dynamics simulations). When taking the *mesoscopic* perspective, typically the individual atoms can not be seen anymore and are not of interest (as, e.g., done through regular transmission electron microscopy or dislocations dynamics simulations). However, the dislocation line itself is still observable: the tube-like defect "region" is reduced to an idealized mathematical line, as demonstrated in the right panel of Fig. 2.8. Therefore, the transition from the atomic scale to the mesoscale requires a conceptual and mathematical idealization that significantly reduces the amount of information. These idealizations must be accompanied by further details and definitions from the atomic scale, including the crystal structure, the lattice, the lattice plane, and the lattice direction information, all of which have an impact on the dislocation's motion. For example, the motion of a dislocation line through a crystal is constrained to a specific crystallographic plane or lattice plane. Thus, it still requires crystallographic information, even though the "defect region" is now only represented as a mathematical line. These two different levels of information require particular attention when designing the dislocation ontology.

The particular crystallographic or lattice plane constraining the dislocation motion is called the *slip plane* (see the green plane in Fig. 2.9). There are specific *slip directions*, which are lattice directions along which plastic deformation occurs within the slip plane, given by the Burgers vector. A *slip system* is a set of slip planes with the same unit normal vector and the same slip direction. Thus, the

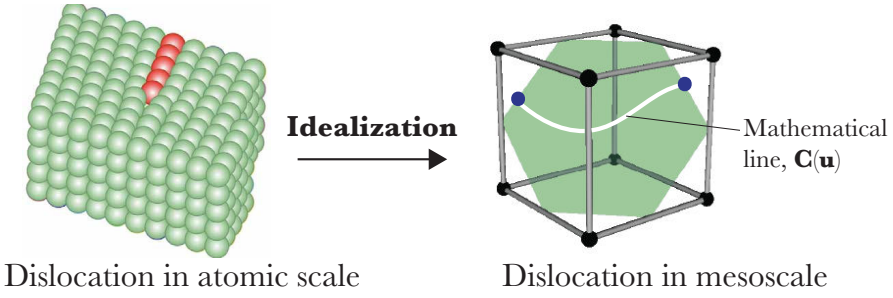


Fig. 2.8: The idealization represents the dislocation in the mesoscale. Here, the individual atoms are no longer visible. This idealization reduced the tube-like defect "region" to a mathematical line. Note that the line on the right does not correspond to the dislocation in the left (this would be a vertical, straight line).

unit normal vector and the slip direction or the Burgers vector (where the latter is not a unit vector) determine the slip system.

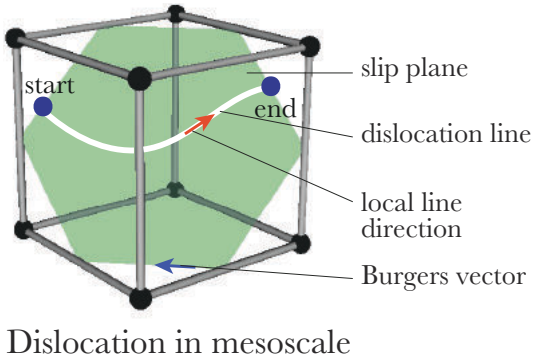


Fig. 2.9: Depiction of the mathematical dislocation line on the mesoscale as a mathematical object that has start and end points. The object is characterized by the Burgers vector and the line sense. Furthermore, the dislocation motion is constrained by the slip plane.

The mathematical representation of a mesoscale dislocation, as shown in Fig. 2.9, is an oriented curve with a start point and an endpoint. While the local line orientation may change along the curve, the Burgers vector remains constant at every point because it is not a property of individual points along the line. Since the dislocation is a directed curve, it has a line sense. Unlike the local line orientation, the line sense is a property of the whole line.

Various computational and experimental techniques are leveraged to predict and observe dislocations in crystalline materials, some of which were already men-

tioned above. For instance, high-resolution transmission electron microscopy (or field ion microscopy) is used on the atomic scale to image the arrangement of atoms. On the mesoscale, the focus is on examining the characteristics of individual dislocations and analyzing the distribution, arrangement, and density of dislocation in materials. Transmission Electron Microscopy (TEM) and Discrete Dislocation Dynamics (DDD) simulations are techniques for investigating these properties and simulating the dislocation behaviour, respectively.

TEM is a microscopy technique that generates a highly-magnified image of a material specimen. This technique involves an electron beam passing through the specimen and several lenses. In strongly simplified terms, if the electron beam hits an atom, then it is deflected. As a result of the deflection, the intensity of the transmitted beam is reduced, and the intensity of the diffracted beam is increased. The dislocation can be seen as a dark line in such a bright-field image.

Above, it was already mentioned that the displaced atoms around a dislocation result in stresses, because the atoms are no longer in their preferred equilibrium position. Dislocations move in such stress fields which are mainly described by the governing equations of (linear) elasticity theory. DDD simulations employ mathematical lines (polygons or splines) to represent dislocations, which are moved based on elastic interactions and further "local rules".

The numerical schemes used in DDD simulations require to numerically discretize the mathematical line, e.g., by a number of straight line segments. The discretization steps are illustrated in Fig. 2.10. Further details can be found in [86]. The

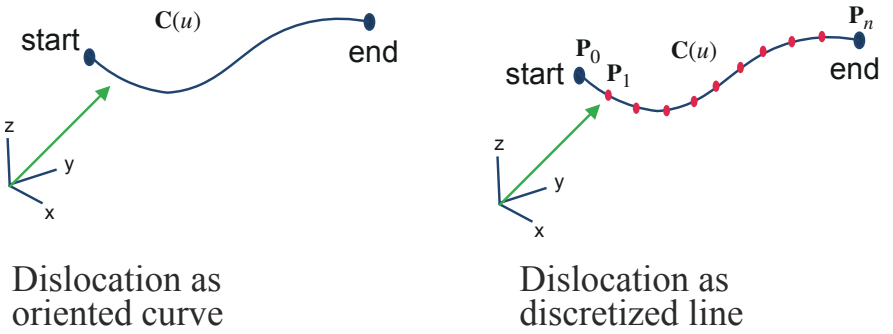


Fig. 2.10: The discretization of dislocation to a numerical representation. The oriented curve dislocation line shown in the left panel is discretized into a number of segments shown in the right panel.

discretization process can be easiest described based on the example of a polygonal chain. There, the smooth mathematical line is approximated by a polygonal chain, $\mathbf{C}$. $\mathbf{C}$ is a curve defined by a sequence of points ($\mathbf{P}_0$, $\mathbf{P}_1$,..., $\mathbf{P}_n$), and these points are called vertices. In addition, the curve consists of segments connecting consecutive pairs of vertices. In general, we can define the shape of a segment

through the *shape function* which allows to have not only straight line segments but also spline curves of different order.

2.3 Provenance Information in MSE Study

Provenance information is metadata that captures a resource or the origin of dataset, development, and transformation [58]. Provenance information in MSE comprises various details such as information about newly developed materials, the researchers involved in their creation, the sample fabrication, preparation, and measurement processes, the laboratories where these activities occur, their data analysis lifecycle, and the equipment or instruments utilized during experiments.

Provenance information plays an essential role in ensuring data reliability and integrity. Additionally, provenance information assists researchers in evaluating datasets by offering valuable context and enhancing data transparency. Proper documentation of provenance information also facilitates data reusability, making it easier to reuse the data in other research directions [56].

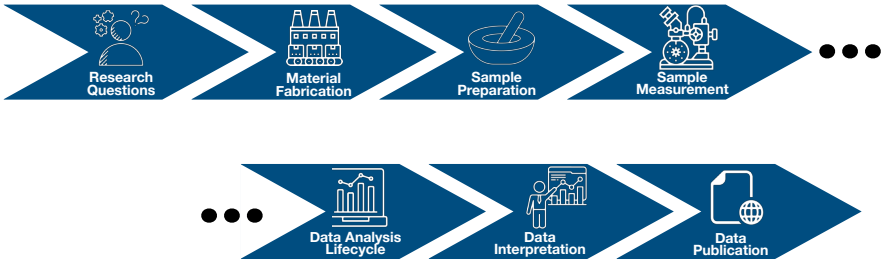


Fig. 2.11: A simplistic diagram of a MSE research project. Provenance information captures all information from the beginning of the study until the data or a study is published. By documenting provenance information on a project, it ensures the reliability and integrity of the data, facilitating data reusability.

In a typical MSE research project (cf. Fig. 2.11), research users conduct several studies to verify and validate scientific findings that support the answers to some specific research questions. They collect and analyze research data to obtain conclusions eventually published in one or more scientific publication(s). Every study generally comprises several processes: fabrication, preparation of samples, data acquisition, data analysis lifecycle, data interpretation, and publication [85].

Fabrication is a process to produce a material, the so-called "precursor". The process can be carried out by a commercial enterprise, researchers, or a third party. In the context of MSE, fabrication can refer to *materials synthesis*, where raw materials or chemicals are transformed into a desired precursor with specific properties. For example in an in-situ TEM experiment, ingots of CoCrFeMnNi equimolar alloy

(Cantor alloy) were fabricated from pure metals pellets or powders (purity always higher than 99.9% purity) [87].

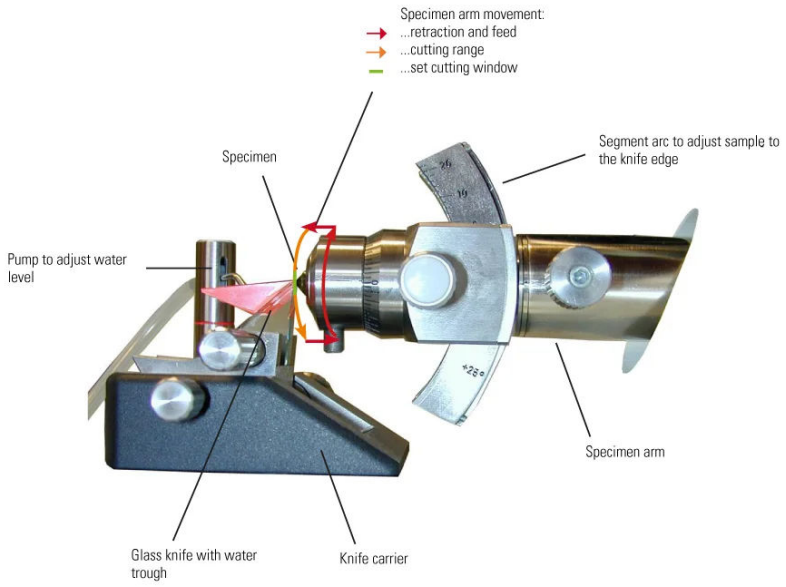
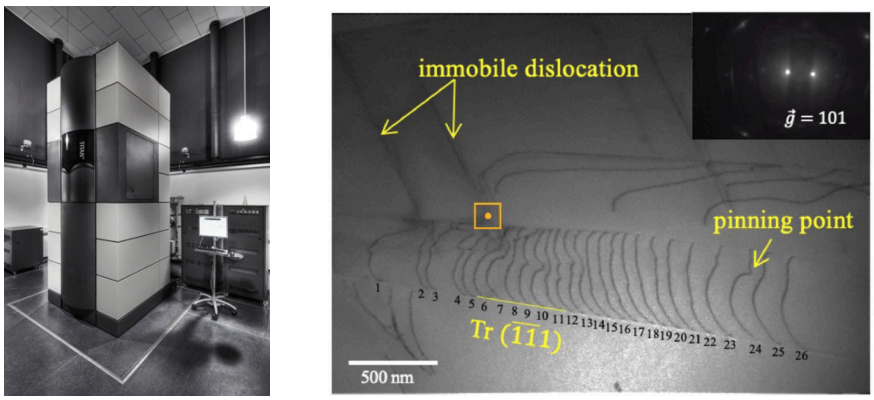


Fig. 2.12: Ultramicrotome is an instrument utilized to prepare extremely thin slices of a material, known as ultrathin sections, typically for observation under high-resolution microscopes such as TEM [88]. The image is adapted from [89].

The subsequent step is the *sample preparation*. It is a process of preparing a precursor into a sample that fits several requirements for a measurement. The sample preparation may involve using a single precursor or combining two or more precursors in a physical action, such as mixing and milling. Furthermore, the sample preparation may involve cutting out a sample to have a desirable thickness. For example, an ultramicrotome (cf. Fig. 2.12) cuts the ultrathin material used for a TEM experiment. The resulting *sample* from this preparation process is ready for the following process: *data acquisition* or *measurement*.

Measurement involves acquiring raw data using specific instruments and techniques tailored to the sample characteristics. The process of measuring a sample begins with mounting it on an appropriate *sample holder* and/or carrier to ensure proper alignment and stability (cf. Fig. 2.13). In many cases, the same sample or group undergoes multiple measurements with different instruments and methods, sometimes at various laboratories or institutions, to gather comprehensive data and support the validation of scientific hypotheses. As shown in the Fig. 2.14, a sample measurement is performed leveraging the *in-situ TEM experiment*. The in-situ TEM experiment is a real-time technique in which materials are observed un-

Fig. 2.13: Sample holder of a TEM is utilized to put and hold the sample during the data acquisition or measurement. The liquid helium flows to the sample holder to reach a lower temperature (down to 4-5 Kelvin). The image is adapted from [90].



(a) TEM instrument.

(b) TEM results in a figure.

Fig. 2.14: The advancement of electron microscope enables one to perform *an in-situ TEM* experiment. As shown on the right panel, the dislocation is captured during the in-situ TEM experiment where a sample is being pulled. These figures are adapted from [91] and Zhang et al. [27].

der an electron microscope while undergoing environmental changes such as heating, cooling, being pulled, or being pressed. Ultimately, as shown in the Fig. 2.14b, the in-situ TEM experiment results in a raw data of a figure that can be used for further analysis.

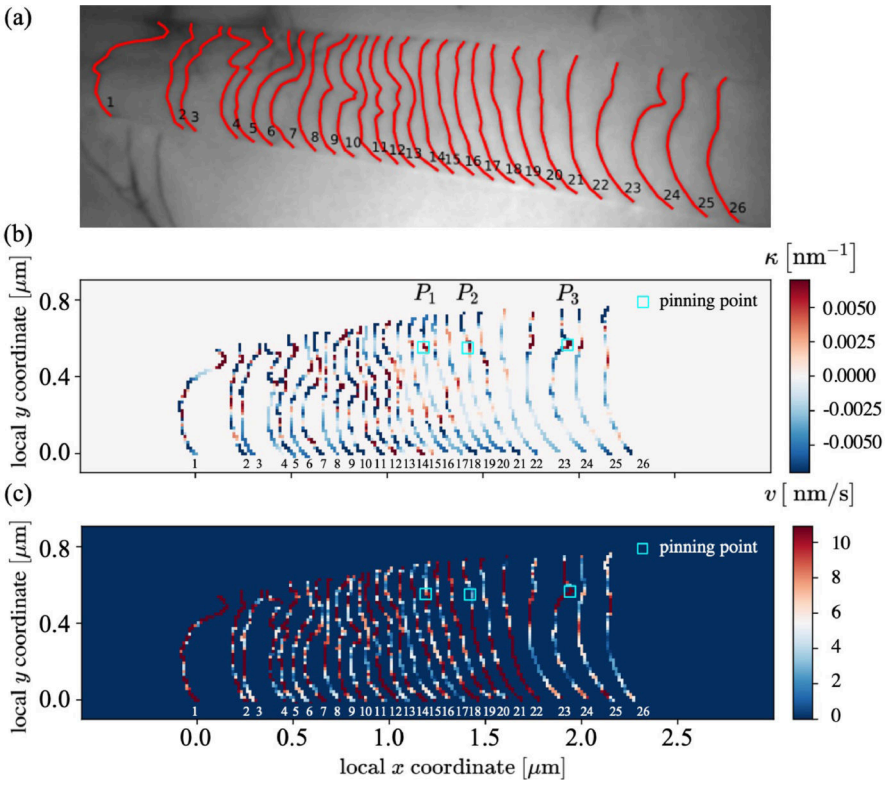


Fig. 2.15: The data analysis lifecycle is then performed given the raw data from the sample measurement. (a) Labeled dislocations in the original TEM image, (b) Further data analysis is needed to calculate a specific characteristic, e.g., the distribution of dislocation curvature, and (c) the velocity distribution of dislocation. These figures are adapted from Zhang et al. [27].

The scientific hypothesis is validated throughout the data analysis lifecycle. Raw data undergo data processing and analysis to extract insights supporting the answer to the scientific research question posed. This process might also be iterated. As shown in the Fig. 2.15, the data analysis lifecycle is performed on raw data generated from previous data acquisition or measurement. Furthermore, the image (a) shows the dislocation image taken from the TEM instrument, and image (b) and (c) are the analysis extracting insights of dislocation curvature and velocity.

Data interpretation is performed on analysed and reference data to draw conclusions and support decision-making about the following study processes. Research data that are scientifically related and underlie a specific scientific finding, along with all the relevant metadata describing their provenance, are collected into one or more datasets. These datasets represent the knowledge acquired. Furthermore, these datasets may be stored in a data collaboration platform or published in a data repository, e.g., Zenodo, to make them available to the scientific community.

Chapter 3
Methods

This chapter discusses technologies used to develop an ontology, knowledge graphs, and text-mining frameworks. The development of ontology and knowledge graph utilizes Semantic Web technologies, including Resource Description Framework (RDF), Resource Description Framework Schema (RDFS), Web Ontology Language (OWL), and SPARQL Protocol and the RDF Query Language (SPARQL). Subsequently, we elaborate on two ontology engineering methods: Ontology101 [92] and NeOn Methodology [93]. Ontology engineering involves developing, managing, and maintaining ontologies using methodologies, tools, and languages to guide the ontology development. In the last part of the chapter, we explain the Natural Language Processing (NLP) techniques focusing on the state-of-the-art transformer-based Large Language Model (LLM).

3.1 The Semantic Web Technologies

In 2001, a computer scientist named Tim Berners Lee presented the vision of the Semantic Web. The Semantic Web is an extension of the current web in which the information is given well-defined meaning, enabling computers and people to cooperate [94]. However, the current web is intended to be human-readable, where the information is available as text documents formatted in HyperText Markup Language (HTML). This makes it difficult for machines to understand human language and interpret the information on the web. Thus, the Semantic Web reconciles the understanding gap between humans and machines, allowing machines to exchange content without human intervention or guidance, i.e., machine interoperability.

Data is an essential subject in the Semantic Web. The Semantic Web derives information from data through semantic theory, e.g., vocabularies and ontologies [95]. With this approach, the semantic theory provides meaning to the data by expressing rich, linked, and machine-interoperable data. Ultimately, machines

can process data not only as a set of terms but also as interconnected concepts, generating the web of data.

A remaining question is how the data on the web should be organized to make machines interoperable, realizing the web of data. According to [96], the following points need to be considered in order to make machines interoperable:

- Use the canonical form of expressing factual claims, e.g., a simple sentence structure consisting of a Subject, Predicate, and Object (SPO) sequence.
- Use one name to refer to one thing, e.g., *Nickel*
- Use unambiguous names that refer to one thing, e.g., *Nickel (Crystalline Material)*.
- Writing queries analogous to the structure of data.
- Make the semantics of terms explicit, e.g., *Dislocation* is subsumed by *Crystallographic Defect*.
- Provide links to relevant sources.

In the following subsections, we will describe Semantic Web Technologies, a collection of technologies fulfilling requirements described in the points above. These technologies include the RDF for a framework describing *resources*, the RDFS and OWL as vocabularies for knowledge representation, as well as SPARQL for querying RDF data/graphs.

3.1.1 The Resource Description Framework

The RDF is a standard framework developed by the World Wide Web Consortium (W3C) for describing resources [97]. Resources refer to things with distinct identities that can be described in data. They include virtual entities like web pages, websites, and desktop files, as well as tangible entities like books, people, places, and stores. Additionally, resources can refer to abstract entities such as categories, animal species, locations, and ancestry relationships.

RDF Terms

Three types of RDF Terms are used to refer to resources: Internationalized Resource Identifier (IRI), literal, and blank node. Identifying resources using simple strings can be ambiguous. For example, the term “Metal” can refer to a musical genre, a type of material, or even a software[1]. To avoid confusion and ambiguity when identifying resources, RDF utilizes the native global naming scheme of the Web, called IRI.

[1] `https://developer.apple.com/metal/`

IRI

A typical IRI (cf. Fig. 3.1) consists of several parts. The `SCHEME` indicates the protocol used to locate information about the resource if it is available. The `HOST` provides the server's location as a physical IP address or a hierarchical domain name. The `PATH` refers to the file location on the server where information about the resource is stored. The `FRAGMENT` refers to something contained within a file. Other valid IRIs may have different schemes, such as `urn:isbn:0-7008-3259-9` or `mailto:a.ihsan@fz-juelich.de`.

Fig. 3.1: A typical IRI anatomy

Writing the full IRI for describing resources is a tedious task, thus many RDF syntaxes use prefixes as shorthand. For example, we can define the prefix `diso:` as `http://purls.helmholtz-metadaten.de/disos/diso#`, where `diso:Dislocation` refers to the full IRI of `http://purls.helmholtz-metadaten.de/disos/diso#Dislocation`.
In Table 3.1, we list core prefixes for the Semantic Web standards and an example prefix used throughout this section.

Table 3.1: Core prefixes for the Semantic Web standards

Prefix	**Value**
`rdf:`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`xsd:`	`http://www.w3.org/2001/XMLSchema#`
`rdfs:`	`http://www.w3.org/2000/01/rdf-schema#`
`owl:`	`http://www.w3.org/2002/07/owl#`
`ex:`	`http://example.org/`

Literals

The second term in RDF is Literal. Literal provides human-readable information to RDF descriptions, such as title, descriptions, numeric value, and dates. In RDF 1.1, a literal may consist of three parts [98]:

- **Lexical form**: a Unicode string;

- **Datatype IRI**: identifies the "type" of literal;
- **Language tag**: indicates the human language of the text.

The basic literal form in RDF consists only of a lexical form: a simple string such as a "`Crystal Structure`". Furthermore, a simple string can optionally be added to a *language tag*, e.g., "`Crystal Structure`"`@en`, specifying that the string is in English. The last ingredient of RDF literal is the datatype IRI, which denotes how a lexical form should be interpreted; for example, the lexical form "5" could be interpreted as a number or a character; in this regard, the datatype IRI indicates the appropriate value. Datatype IRIs are written using a double-caret symbol, where "5"`^^xsd:integer` is an example literal indicating an integer value of 5, while "5"`^^xsd:string` is a literal indicating a string of "5". The complete list of datatype IRIs utilized in RDF can be seen on the website[2].

Blank nodes

In some cases, identifying a resource with a globally unique IRI or a literal may be restrictive, impractical, or unnecessary. To address this, RDF offers an alternative: using blank nodes. Blank nodes represent some resource's existence rather than identifying a specific resource. Blank nodes are typically represented with an underscore prefix, such as `_:bnode001`.

Definition 3.1 RDF Terms [99]. Let $\mathbf{I}$ denote the set of IRIs, $\mathbf{L}$ denote the set of RDF literals, and $\mathbf{B}$ denote the set of RDF blank nodes. Furthermore, these three sets are pairwise disjoint: they share no elements. The set of RDF terms is defined as the union of these three sets: $\mathbf{I} \cup \mathbf{L} \cup \mathbf{B}$.

RDF Triples and Graphs

RDF triples describe resources by making statements about them. Their structure is based on a simple sentence structure found in natural language, SPO sentences, as illustrated in Fig. 3.2. Furthermore, each triple element is filled by a single RDF term.

Certain limitations are applied when positioning different types of RDF terms within triples. Particularly, the subject position can only have an IRI or a blank node, the predicate position can only have an IRI, and the object position can have an IRI, a blank node, or a literal. The object position can optionally include a datatype IRI or a language tag. Following on the previous discussion and using Definition 3.1 for RDF terms, we can now present a formal definition for an RDF triple:

Definition 3.2 RDF Triple [99]. Let $\mathbf{I}$, $\mathbf{B}$, and $\mathbf{L}$ be disjoint infinite sets of IRIs, blank nodes, and literals, respectively. A tuple $(s, p, o) \in (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times$

[2] `https://www.w3.org/TR/xmlschema11-2/`

ex:Nickel (SUBJECT) ex:hasCrystalStructure (PREDICATE) ex:FCC (OBJECT)

ex:FCC (SUBJECT) ex:hasLattice (PREDICATE) ex:FCC-Lattice (OBJECT)

ex:FCC-Lattice (SUBJECT) ex:hasCrystalSystem (PREDICATE) ex:Cubic (OBJECT)

ex:FCC-Lattice (SUBJECT) ex:centering (PREDICATE) "F"^^xsd:string (OBJECT)

Fig. 3.2: Three example of RDF triples. Each triple consists of a SPO sentence.

$(\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$ is denominated an RDF triple, where s is called subject, p the predicate, and o is the object.

The data unit in RDF is encoded as a triple, where a set of triples is called an RDF graph. The RDF graph is defined as follow:

Definition 3.3 RDF Graph [96]. An *RDF graph* G is a subset of $\mathbf{IB} \times \mathbf{I} \times \mathbf{IBL}$; i.e., an RDF graph is a set of triples.

Triples are not necessarily ordered in RDF graphs since they are based on sets. Furthermore, RDF triples are called graphs because a triple consisting of a subject, predicate, and object closely corresponds to a directed and labeled edge in a graph. In this representation, the predicate acts as the label for the edge between the subject and object nodes: $s \xrightarrow{p} o$. Thus, an RDF graph is a collection of labeled edges forming a graph.

Fig. 3.3: A corresponding RDF graph from the Fig. 3.2.

RDF Vocabulary

RDF provides the core built-in IRIs as for modelling the data, called RDF vocabulary. RDF vocabulary can be used to model data as RDF triples. The first features in the RDF vocabulary are `rdf:type` and `rdf:Property`. The former describes a property/predicate for relating an instance to its class, e.g., `ex:FCC-lattice rdf:type`

`ex:BravaisLattice`. Moreover, the latter is the class of all properties/predicates, e.g., `ex:hasCrystalSystem rdf:type rdf:Property`.

In Fig. 3.4, we can see a crystal structure data model. In that data model, a crystalline material has its corresponding crystal structure. A crystal structure relates to a Bravais lattice, which then relates to a crystal system. A data model can also be called the Terminology Box (TBox), which is a set of terminology components. Using RDF vocabulary, e.g., `rdf:type`, we can instantiate the data model resulting in an Assertion Box (ABox), as shown in Fig. 3.5.

Fig. 3.4: A crystal structure data model.

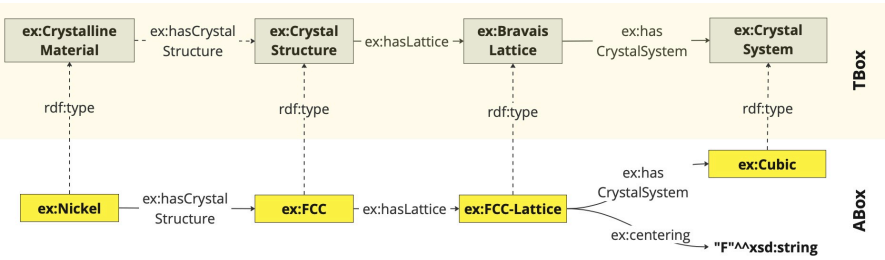


Fig. 3.5: Using RDF vocabulary to instantiate a data model, e.g., `rdf:type`. A data model is also called TBox which is a set of terminology components. And, the data instance from TBox is called ABox.

RDF is suitable for representing directional, named relationships between two resources in a binary relation. However, it may be less effective in representing relationships with higher *arity* between more than two resources. To model a *n-arity* relation, RDF introduces a special property, called `rdf:value`, which indicates the main value of the n-ary relation resource. E.g., in the following statement:

- `ex:FCC-Lattice ex:hasLatticeParameterAngle ex:AngleAlpha`
- `ex:AngleAlpha rdf:value "90.0"^^xsd:float`

Another feature in the RDF vocabulary is to model the container via `rdf:Bag`, `rdf:Seq`, and `rdf:Alt`. Furthermore, we can also model the collection via `rdf:List`, which is to represent a linked list.

RDF Serializations

RDF does not have a specific syntax to represent its data. However, it can be represented in different formats known as "Serializations." Many serializations are available to represent RDF documents, including RDF/XML, Turtle, N-Triples, JSON-LD, TriG, and others. The purpose of serializing RDF is to have a syntax that is machine-parseable. Below, we have listed some of the most popular serializations.

RDF/XML serialization

RDF/eXtensible Markup Language (RDF/XML) serialization. RDF/XML syntax is the oldest RDF syntax [100]. The RDF/XML syntax leverages the existing tools and languages for XML to create RDF tools and languages. How data models are represented makes sharing information easier for different applications on different operating systems. An RDF/XML document is structured like a tree, with the root node (Root Tag) being `rdf:RDF`, followed by a set of namespaces as attributes. Each resource in this syntax is identified by an `rdf:Description` element with the `rdf:about` attribute, which can contain several statements about the same subject. An example of RDF/XML serialization is shown in the Listing 3.1.

Listing 3.1: RDF graph in the Fig. 3.5 serialized using the RDF/XML serialization

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
   xmlns:ex="http://example.org/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://example.org/hasCrystalSystem">
    <rdf:type
    rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/Nickel">
    <rdf:type rdf:resource="http://example.org/CrystallineMaterial"/>
    <ex:hasCrystalStructure rdf:resource="http://example.org/FCC"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/FCC-Lattice">
    <rdf:type rdf:resource="http://example.org/BravaisLattice"/>
    <ex:hasCrystalSystem rdf:resource="http://example.org/Cubic"/>
    <ex:centering
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string">F</ex:centering>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/hasLattice">
    <rdf:type
    rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/CrystallineMaterial">
    <ex:hasCrystalStructure
    rdf:resource="http://example.org/CrystalStructure"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/CrystalStructure">
    <ex:hasLattice rdf:resource="http://example.org/BravaisLattice"/>
  </rdf:Description>
```

```
  <rdf:Description rdf:about="http://example.org/FCC">
    <rdf:type rdf:resource="http://example.org/CrystalStructure"/>
    <ex:hasLattice rdf:resource="http://example.org/FCC-Lattice"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/BravaisLattice">
    <ex:hasCrystalSystem rdf:resource="http://example.org/CrystalSystem"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/hasCrystalStructure">
    <rdf:type
    rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/Cubic">
    <rdf:type rdf:resource="http://example.org/CrystalSystem"/>
  </rdf:Description>
</rdf:RDF>
```

N-Triples serialization

N-Triples [101] is a syntax for writing an RDF graph in which each statement appears on a separate line. An N-Triples document contains a sequence of RDF triples in which every simple triple consists of a subject, predicate, and object separated by a white space and terminated by a period. The file extension used for an N-Triples document is ".nt." and it does not contain any parsing directives. An example of N-Triples serialization is shown in the Listing 3.2

Listing 3.2: RDF graph in the Fig. 3.5 serialized using the N-Triples serialization

```
<http://example.org/hasCrystalSystem>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .

<http://example.org/Nickel>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/CrystallineMaterial> .
<http://example.org/Nickel>
    <http://example.org/hasCrystalStructure>
    <http://example.org/FCC> .

<http://example.org/FCC-Lattice>
    <http://example.org/hasCrystalSystem>
    <http://example.org/Cubic> .

<http://example.org/hasLattice>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .

<http://example.org/CrystallineMaterial>
    <http://example.org/hasCrystalStructure>
    <http://example.org/CrystalStructure> .

<http://example.org/CrystalStructure>
    <http://example.org/hasLattice>
    <http://example.org/BravaisLattice> .

<http://example.org/FCC-Lattice>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
```

```
    <http://example.org/BravaisLattice> .

<http://example.org/FCC>
    <http://example.org/hasLattice>
    <http://example.org/FCC-Lattice> .

<http://example.org/FCC-Lattice>
    <http://example.org/centering>
    "F"^^<http://www.w3.org/2001/XMLSchema#string> .

<http://example.org/FCC>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/CrystalStructure> .

<http://example.org/BravaisLattice>
    <http://example.org/hasCrystalSystem>
    <http://example.org/CrystalSystem> .

<http://example.org/hasCrystalStructure>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .

<http://example.org/Cubic>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/CrystalSystem> .
```

Turtle serialization

Terse RDF Triple Language (Turtle) [102], is a format for representing RDF graphs that is easier for humans to read. Unlike other formats, Turtle groups multiple triples with the same subject under the same IRI, eliminating the need to repeat the IRI for each triple. Furthermore, predicates and objects are listed after the subject, separated by a semicolon, and terminated by a period. An example of Turtle serialization is shown in the Listing 3.3

Listing 3.3: RDF graph in the Fig. 3.5 serialized using the Turtle serialization

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:Nickel a ex:CrystallineMaterial ;
    ex:hasCrystalStructure ex:FCC .

ex:hasCrystalStructure a rdf:Property .

ex:hasCrystalSystem a rdf:Property .

ex:hasLattice a rdf:Property .

ex:CrystallineMaterial ex:hasCrystalStructure ex:CrystalStructure .

ex:Cubic a ex:CrystalSystem .

ex:FCC a ex:CrystalStructure ;
    ex:hasLattice ex:FCC-Lattice .

ex:FCC-Lattice a ex:BravaisLattice ;
    ex:centering "F"^^xsd:string ;
    ex:hasCrystalSystem ex:Cubic .
```

```
ex:BravaisLattice ex:hasCrystalSystem ex:CrystalSystem .

ex:CrystalStructure ex:hasLattice ex:BravaisLattice .
```

JSON-LD serialization

Java Script Object Notation for Linked Data (JSON-LD) [103], is a simplified way of representing the RDF graph. It is based on Java Script Object Notation (JSON), a data format that uses human-readable text for attribute-value pairs data object transmission. JSON-LD allows many JSON parsers to be reused to parse JSON-LD documents. The primary goal of JSON-LD is to make it easier to use RDF graphs in web-based applications and create interoperable web services. JSON-LD documents represent the JSON object as a sequence of attribute-value pairs surrounded by curly brackets and separated by a single comma. A colon separates the attribute and its value. An example of JSON-LD serialization is shown in the Listing 3.4.

Listing 3.4: RDF graph in the Fig. 3.5 serialized using the JSON-LD serialization

```
{
  "@context": {
    "ex": "http://example.org/",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  },
  "@graph": [
    {
      "@id": "ex:FCC-Lattice",
      "@type": "ex:BravaisLattice",
      "ex:centering": "F",
      "ex:hasCrystalSystem": {
        "@id": "ex:Cubic"
      }
    },
    {
      "@id": "ex:CrystalStructure",
      "ex:hasLattice": {
        "@id": "ex:BravaisLattice"
      }
    },
    {
      "@id": "ex:Nickel",
      "@type": "ex:CrystallineMaterial",
      "ex:hasCrystalStructure": {
        "@id": "ex:FCC"
      }
    },
    {
      "@id": "ex:BravaisLattice",
      "ex:hasCrystalSystem": {
        "@id": "ex:CrystalSystem"
      }
    },
    {
      "@id": "ex:CrystallineMaterial",
      "ex:hasCrystalStructure": {
        "@id": "ex:CrystalStructure"
      }
    },
    {
```

```
      "@id": "ex:hasCrystalSystem",
      "@type": "rdf:Property"
    },
    {
      "@id": "ex:hasCrystalStructure",
      "@type": "rdf:Property"
    },
    {
      "@id": "ex:Cubic",
      "@type": "ex:CrystalSystem"
    },
    {
      "@id": "ex:FCC",
      "@type": "ex:CrystalStructure",
      "ex:hasLattice": {
        "@id": "ex:FCC-Lattice"
      }
    },
    {
      "@id": "ex:hasLattice",
      "@type": "rdf:Property"
    }
  ]
}
```

3.1.2 Ontologies

The term "ontology" originates from philosophy, which is the study of the nature of things, their properties, and their relation across various domains [104]. In the computer science domain, ontology is defined as "*an explicit, formal specification of a shared conceptualization.*" In the 1980s, a group of researchers introduced the concept of "ontological analysis" [105] as a technique for knowledge engineering. which led to the adoption of ontologies in various scientific communities, e.g., bioinformatics [106], materials science [39, 107], and environmental science [108].

An ontology defines concepts and their relationships within a specific domain and can be used for knowledge representation by providing a structured vocabulary in a semantic format that explicitly specifies the meaning of these concepts and their relationships. When applied to data, an ontology facilitates the organization of the data by embedding semantic meaning and enabling the inference of new information. In addition, an ontology serves as a foundation for enabling machines or computers to work together, i.e., machine-interoperability.

RDF can create statements about resources, but it cannot describe groups of related resources and their relationships, such as classes, sub-classes, and properties. Therefore, another data modeling schema is needed to describe the formal meaning of RDF resources and their interlinking relationships so that machines/agents can process this information more efficiently. The following section will discuss the two most commonly used schema languages for creating ontologies: RDFS and OWL.

RDF Schema

Resource Description Framework Schema (RDFS) [109] is an extension of RDF vocabulary that allows for explicitly defining the types of semantic, e.g., taxonomic relations, domain and range of a property, and human-readable annotations. In RDFS, resources can be grouped into a group (category and type) called *class* which is encoded in RDF term of `rdfs:Class`. Furthermore, RDFS has ability to represent the taxonomic relation (subclass/subsume) describing the class hierarchy and the property of an ontology. By using RDFS, we can specialize a class *B* from a general class *A*, i.e., *B* is a subclass of *A*. By using RDF term property of `rdfs:subClassOf`, we can write `B rdfs:subClassOf A`. Apart from specializing the class, RDFS offers way to specialize the property via `rdfs:subPropertyOf`. E.g., `ex:hasBravaisLattice rdfs:subPropertyOf ex:hasLattice`.

Another extension in RDFS is we can define the *domain* and *range* of a property (`rdf:Property`) using the RDF term, `rdfs:domain` and `range`, respectively. When defining a domain property, RDFS asserts that any resource that has a given property is an instance of one or more classes On the other hand, a property's possible values –an object in the SPO sentence– can be restricted by specifying its range(s). The range can be one or more classes, indicating that the property's values are instances of those classes.

RDFS defines other auxiliary terms in addition to sub-property, sub-class, domain, and range. These are four terms provide human-readable information/annotation about resources: `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`, and `rdfs:isDefinedBy`. the `rdfs:label` is a property that relates a resource with a human-readable label giving its name. Multilingual labels can be given to the same resource using the language tag. For example, the English and German translations of the resource *Crystal Structure* can be represented as `ex:CrystalStructure rdfs:label "Crystal Structure"@en ; rdfs:label "Kristallstruktur"@de`.

The `rdfs:comment` is a property that relates a resource with a human-readable comment describing it in more detail. Then, the `rdfs:seeAlso` is a property that relates a resource with a location on the Web that contains information about it. Last, the `rdfs:isDefinedBy` is a property that relates a resource with a location on the Web that provides a definition of it. Please note that these four terms do not have specific semantics: they provide only a minimalist set of agreed-upon terms for basic annotations.

RDFS reasoning capabilities

Performing logical inference is a way to discover hidden knowledge that can be automatically inferred. RDFS can make deductions, such as inferring a class's membership based on the range or domain of one of its properties. For example, suppose we have the following triples:

```
ex:hasCrystalSystem rdfs:domain ex:Lattice .
```

```
ex:hasCrystalSystem rdfs:range ex:CrystalSystem .
ex:FCC-Lattice ex:hasCrystalSystem ex:Cubic .
```

It can be inferred that

```
ex:FCC-Lattice rdf:type ex:Lattice .
ex:Cubic rdf:type ex:CrystalSystem .
```

Another reasoning capability of RDFS is to deduce the entity-superclass membership from a class hierarchy. E.g., consider the following triples:

```
ex:BravaisLattice rdfs:subClassOf ex:Lattice .
ex:FCC-Lattice rdf:type ex:BravaisLattice .
```

It can be inferred that

```
ex:FCC-Lattice rdf:type ex:Lattice .
```

The last RDFS reasoning capability is to deduce an entity super property membership from a property hierarchy. E.g., consider following triples:

```
ex:hasBravaisLattice rdfs:subPropertyOf ex:hasLattice .
ex:FCC ex:hasBravaisLattice ex:FCC-Lattice .
```

It can be inferred that

```
ex:FCC ex:hasLattice ex:FCC-Lattice .
```

The Web Ontology Language

Web Ontology Language (OWL) is a language for representing ontologies based on the formal logic, a discipline that progressed from mathematics and philosophy. OWL formalizes ontologies by defining classes and relations between them for a particular domain. Furthermore, OWL is a rich vocabulary extending RDF vocabulary and RDFS by allowing the more semantic expressiveness in the representation of data and domain knowledge. As with two previous vocabularies categorizing resources into Class, Property, and Annotation, in OWL, resources are categorized into five entities: Class, Object property, Datatype property, Annotation property, and Individual. Furthermore, we can specify the (in)equality in OWL e.g., two different IRIs denoting the same or different resources, classes, and property can be specified by `owl:sameAs` or `owl:differentFrom`, `owl:equivalentClass`, and `owl:equivalentProperty`, respectively.

In OWL, the class entity is defined by `owl:Class`. Please note that `owl:Class` differs from `rdfs:Class` in terms of decidability in the reasoning process [96], i.e., `owl:Class` is decidable, and consequently for the rest of the works, we will refer a class to `owl:Class`. Furthermore, we can define a domain knowledge semantic expressively by allowing to describe a complex class via `owl:Restriction` Furthermore, to specify the cardinality restrictions of the above example, OWL defines the restrictions via, e.g., `owl:minCardinality`, `owl:maxCardinality`,

owl:someValuesFrom, owl:allValuesFrom, and owl:hasValue. To exemplify the use of OWL restrictions, particularly owl:allValuesFrom, the following statement: Bravais lattice is a specialization of lattice that has a relation to crystal system. It is modeled in Listing 3.5.

Listing 3.5: Modeling the Bravais Lattice class in OWL with owl:Restriction

```
ex:BravaisLattice rdfs:subClassOf
    [ a owl:Restriction ;
      owl:onProperty ex:hasCrystalSystem ;
      owl:allValuesFrom ex:CrystalSystem
   ],
   ex:Lattice.
```

Two other capabilities extending the class entity are defining the disjointness and combinations of classes. The former defines that two or more classes can be disjoint via owl:disjointWith, e.g., Lattice parameter of length and Lattice parameter of angle are disjoint classes. The latter defines two or more classes can be combined using set of operations, e.g., union (owl:unionOf), intersection (owl:intersectionOf, and complement (owl:complementOf).

As we have mentioned, the OWL property is divided into the data property (owl:DatatypeProperty) and object property (owl:ObjectProperty). While the data property is a class of all properties relating individuals, which are objects of a specific class, to datatype values, e.g., a string, a boolean, and an integer. The object property is a class of all properties relating individuals to other individuals. Furthermore, the object property has several characteristics, including inverse, functional, transitive, and symmetric.

OWL reasoning capabilities

OWL offers reasoning capabilities including RDFS reasoning and additional capabilities. Here we listed some of them by showing several generic examples in the following.

Inverse properties: if a property, P, is indicated as an inverse of another property, Q, then:

$$\forall x, y, P(x, y) \Longleftrightarrow Q(y, x)$$

For example, suppose a property, ex:parentOf owns an inverse property named childOf, then if ex:Ahmad ex:childOf ex:Ibu, we can infer that ex:Ibu parentOf ex:Ahmad.

Functional properties: if a property, P, is a functional property, then for any given individual, the property can have at most one value. Formally, it is described as follow:

$$\forall x, y, z, P(x, y) \wedge P(x, z) \rightarrow y = z$$

For example, if we say ex:hasBiologicalMother is a functional property, then if ex:Ahmad ex:hasBiologicalMother ex:Ibu and ex:Ahmad

`ex:hasBiologicalMother ex:Mama`, we can conclude that `ex:Ibu` and `Mama` are the same person.

Transitive properties: a property, P, is a transitive property, if P relates to individuals, $P(x, y)$ and $P(y, z)$, then $P(x, z)$. Formally, it is described as follow:

$$\forall x, y, z, P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

For example, if we say `ex:hasAncestor` is a transitive property, then if `ex:Ahmad ex:hasAncestor ex:Bapak` and `ex:Bapak ex:hasAncestor ex:Abah`, we can conclude that `ex:Ahmad ex:hasAncestor ex:Abah`.

Symmetric properties: a property, P, is a symmetric property, if $P(x, y)$, then $P(y, x)$. Formally, it is described as follow:

$$\forall x, y, P(x, y) \Longleftrightarrow P(y, x)$$

For example, if we say `ex:siblingOf` is a symmetric property, then if `ex:Ahmad ex:siblingOf ex:Mila`, we can infer that `ex:Mila ex:siblingOf ex:Ahmad`.

Class disjointness constraints: If two classes A and B are indicated as disjoint, and a class C is a subclass of B, then it can be inferred that C is also disjoint with A. Formally, it is described as follows:

$$\forall A, B, C, disjoint(A, B) \wedge subClassOf(C, B) \rightarrow disjoint(A, C)$$

3.1.3 SPARQL Query Language

The SPARQL Protocol and the RDF Query Language (SPARQL) [110] is a standard protocol and language for querying data and complies with the specification of the W3C. In contrast with the Structured Query Language (SQL) that is used to query the relational database, SPARQL is used to query the RDF data or RDF graphs. SPARQL queries can be written according to the query objective. Listing 3.6 lists the anatomy of a SPARQL query, which is inspired by Turtle syntax. A query anatomy in general is composed of six parts:

1. *Base and prefix declarations*: Similar to the Turtle syntax for RDF, base and prefix IRIs can be declared and later used to abbreviate IRIs.
2. *Dataset construction*: Rather than being defined on a single RDF graph, SPARQL is defined on multiple graphs, where a custom dataset can be composed for a particular query.
3. *Query type and solution modifiers*: SPARQL supports four query types:

   SELECT: It queries RDF graphs and returns the matching results in a tabular format after variable bindings.

ASK: It provides a Boolean result that indicates whether the query graph pattern matches.
CONSTRUCT: It combines the matched triples into a single RDF graph using the union operator and then constructs an RDF graph specified by a graph template.
DESCRIBE: It describes the resources found by their identification either by IRI or a blank node.

In the case of SELECT, solution modifiers allow for controlling whether or not duplicate solutions can (REDUCED) or must (DISTINCT) be removed.

4. *Where clause*: It specifies the patterns to be matched in the graph(s) being queried to generate solutions.
5. *Aggregation*: It indicates variables or other expressions by which results should be grouped, which allows aggregate functions (SUM, MAX, or MIN) to be applied within each group and further allows solutions to be filtered depending on the result of the aggregation function.
6. *Solution modifiers*: They state how the solutions should be ordered, how many solutions to return, and how many to skip.

Listing 3.6 has several symbols and notations used to specify queries. The pipe symbol (|) is used to separate alternatives, while square brackets with an appended ([]*) are used to indicate multiplicities. The default is precisely one, but a question mark (?) indicates zero or one, an asterisk (*) indicates zero or more, and a plus sign (+) indicates one or more. In addition, there are various variables used, including 'i', 'v,' 'x,' 'e,' 'B', 'B'', and 'n', which stand for IRIs, variables, RDF terms, expressions, basic graph patterns, basic graph patterns with property paths, and natural numbers, respectively. Capture groups are indicated using double colons (::), and white-space (space, tab, or newline) is used to separate multiple elements where allowed.

Listing 3.6: Anatomy of a SPARQL query [96].

```
###### (1) base and prefix declarations
[BASE i]?
[PREFIX i]*

###### (2) dataset construction
[FROM i]*
[FROM NAMED i]*

###### (3) query type and solution modifiers
[SELECT S ::
    [DISTINCT|REDUCED]? [v|(e AS v)]+|ASK|DESCRIBE v|CONSTRUCT {B}]

###### (4) where clause
WHERE {
    P::[
        B' |
        {P} [.|UNION|OPTIONAL|MINUS|FILTER EXISTS|FILTER NOT EXISTS] {P} |
        P FILTER (e) |
        BIND (e AS v) |
        VALUES ([v]+) {[([x|UNDEF]+)]*} |
        GRAPH [v|i] {P} |
```

```
        SERVICE [SILENT]? i {P} |
        { SELECT S }
    ]::P
}

###### (5) aggregation
[GROUP BY [v|e]+]?
[HAVING e]?

###### (6) solution modifiers
[ORDER BY [[v|e]|ASC([v|e])|DESC([v|e])]+]^?
[LIMIT n]?
[OFFSET n]?
:: S
```

Basic graph patterns

A basic graph pattern in SPARQL is formed by a set of triple patterns, which is similar to how an RDF graph is formed by a set of RDF tuples. A triple pattern is an RDF triple (SPO) permitting a variable to be queried in any position. Moreover, it consists of a tuple containing IRIs, literals, blank nodes, and query variables. IRIs and literals will only match themselves in the RDF graph being queried. In contrast, blank nodes and query variables can match any term in the RDF graph. Query variables can be returned as part of the query results, whereas blank nodes cannot. For instance, Listing 3.7 shows a basic graph pattern in a SPARQL query. This basic graph pattern has two triple patterns, each with variables, e.g., `?Crystal_Structure` and `?Bravais_Lattice`, in the object position of the triple. The query's graph pattern matches the RDF graph shown in Fig. 3.5. Ultimately, the resulting output, listed in the Table 3.2, is a solution sequence, which may be zero, one, or multiple solutions.

Listing 3.7: A SPARQL query of basic graph pattern for getting an information of what the bravais lattice of a crystal structure.

```
PREFIX ex: <http://example.org/> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

SELECT ?Crystal_Structure ?Bravais_Lattice
WHERE {
    ?Crystal_Structure rdf:type ex:CrystalStructure .
    ?Crystal_Structure ex:hasLattice ?Bravais_Lattice .
}
```

Table 3.2: The result of SPARQL query in Listing 3.7

?Crystal Structure	?Bravais Lattice
ex:FCC	ex:Lattice1

Matching RDF literals

RDF literals, which include strings and numeric types, are used in the value position of a triple pattern. Listing 3.8 shows how query results can be restricted using RDF literals. We can have a more expressive way to match RDF literals by utilizing FILTER. FILTER removes solutions for which the expression evaluates to false or an error. For instance, we want to further query the RDF graph in the Fig. 3.5 querying the Bravais lattice centering of a crystal structure that has a value of Face (F). The SPARQL query and query answer corresponding to the query objective can be seen in the Listing 3.8 and Table 3.3.

Listing 3.8: A SPARQL query of basic graph pattern for getting an information of a crystal structure that has face (F) of Bravais lattice centering.

```
PREFIX ex: <http://example.org/> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .

SELECT ?Crystal_Structure ?Bravais_Lattice_Centering
WHERE {
    ?Crystal_Structure rdf:type ex:CrystalStructure .
    ?Crystal_Structure ex:hasLattice ?bravais_lattice .
    ?bravais_lattice ex:centering ?Bravais_Lattice_Centering
    FILTER (xsd:str(str(?Bravais_Lattice_Centering)) = F)
}
```

Table 3.3: The result of SPARQL query in Listing 3.8

?Crystal Structure	?Bravais Lattice Centering
ex:FCC	F

3.2 Ontology engineering

Ontology engineering involves developing, managing, and maintaining ontologies using methodologies, tools, and languages to guide the ontology development [111]. This approach suggests several steps, tasks, and activities to follow. The importance of a particular activity within a specific ontology-related project depends on various factors. These factors include the complexity of the ontology to be developed, the availability of domain-relevant information sources, the experience of the ontology engineering team, and the characteristics of the environment in which the ontology will be used. Please note that no single correct ontology engineering method exists, as many viable options/approaches exist to develop an ontology. Furthermore, ontology development is necessarily an iterative process, which involves iteratively refining the ontology itself as well as continuously adapting it to new use case and extending it.

This section will describe two ontology engineering methodologies we utilize to develop ontologies: Ontology101 [92] and NeOn methodology [93]. Ontology101 is an ontology engineering method, classified into a micro-level methodology [112] and characterized by a single monolithic ontology developed by one or a few people working in one location. It focuses on guidelines to formalize the subject domain, i.e., guiding how to go from an informal representation to a logic-based one, which is a formal knowledge representation. We will also exemplify one case for developing an ontology leveraging Ontology101. This ontology will describe the concepts and relationships of crystal structure in the materials science domain. Contrary, NeOn methodology is classified as a macro-level methodology based on information system, enabling dynamics, context, collaborative, and distributed ontology development. Furthermore, given the rapid increase in the number of existing ontologies, this method emphasizes reusing the existing ontologies in an ontology development.

3.2.1 Ontology101

Ontology101 consists of several steps iteratively carried out to develop an ontology. Fig. 3.6 illustrates steps in Ontology101. Below we discuss several steps constituting the method and also exemplify a case of an ontology representing the domain knowledge of crystal structure, the Crystal Structure Ontology (CSO).

Fig. 3.6: Ontology101 consists of several steps that are iteratively carried out to develop an ontology.

Scope of the ontology

The initial step in this method is to determine the scope of the ontology. In order determine the scope of the ontology, we limit the scope of the CSO by fulfilling the following points:

- We want to have an ontology for describing the concepts and relationships of crystal structure.
- The ontology will be used to annotate a crystal structure that has several crystallographic properties, e.g., Bravais lattice, lattice parameters, crystal system, space group, and point group.

To further refine the scope we mentioned earlier, we have identified a set of questions (cf. Table 3.4) that a well-designed ontology should be able to answer. These questions are referred to as Competency Questions (CQs), which will later serve as a test to determine if the ontology contains sufficient knowledge to address them. They will also help identify areas that require more detail or specific representation. Even though these competency questions are not exhaustive, they serve as a rough outline of what needs to be addressed.

Table 3.4: Competency questions of CSO

No.	Question
1	Which crystal structure shares the same crystal system (e.g. cubic, hexagonal, and rhombohedral)?
2	What are the lattice parameters of length given a crystal structure?
3	What are the lattice parameters of angle given a crystal structure?
4	Given the space group of a crystal structure, what is the Bravais lattice centering?
5	Given the crystal structure, what are the corresponding space group and point group?

Ontology reuse

The subsequent step after determining the scope of an ontology is the ontology reuse. Reusing terms (classes) and relationships from existing ontologies will save development costs and increase interoperability between ontologies. For the crystal structure domain, ontologies are available, e.g., the Materials Design Ontology (MDO) and the Chemical Entitiy of Biological Interest (ChEBI).

MDO is an ontology developed to represent the domain knowledge of ab-initio calculation in solid-state physics. In solid-state physics, crystalline materials, which are materials with atoms arranged in a periodic structure with a high degree of symmetry (cf. Sec. 2.1), are studied. Thus, there are several MDO classes that we can reuse.

We reuse MDO classes describing:

- The motif (an arrangement of chemical species in the crystal structure) in a crystal structure defined by `MDO:Occupancy` and `MDO:Species`.
- The point group of a crystal structure is defined by `MDO:PointGroup`.
- The space group of a crystal structure is defined by `MDO:SpaceGroup`.

Furthermore, we reuse several data properties to describe point and space group data, e.g., `MDO:SpaceGroupID` and `MDO:PointGroupHMName`. Additionally, we reuse a ChEBI class defining the name of atomic element name, it is `ChEBI:Atom`.

Enumerate terms

After deciding which ontologies to reuse, we enumerate the terms in CSO. By following the scope of the ontology defined in CQs, we list the essential terms representing the domain of crystal structure.

The first term we include is the *crystal structure*. The crystal structure is represented by the *lattice* together with a *motif*. The former is a mathematical concept of infinite, repeating arrangements of points in space (3D), in a plane (2D), or on a line (1D), in which all points have the same surrounding and coincide with atom positions. The latter is the arrangement of chemical species, which can be atoms, ions, or molecules in crystalline materials. These two terms are included in CSO.

A crystal structure consists of a smallest repeating unit that can replicate to form the entire crystal structure. This smallest unit is known as the *unit cell* (cf. right panel of Fig. 2.1) and will be included as a term in CSO. The lattice parameters of the unit cell define its edge lengths and the angles between these edges. Both edge lengths and angles are also included into CSO. Furthermore, the three edge lengths of the lattice parameters are denoted as (a, b, c) while the three angles are represented as (α, β, γ).

Unit cells are categorized based on their lattice parameters (cf. Fig. 2.3). For example, in the cubic crystal system, all three edge lengths are equal ($a = b = c$) and the angles between them are 90° ($\alpha = \beta = \gamma = 90°$). In contrast, the orthorhombic system has unequal edge lengths ($a \neq b \neq c$) while still maintaining the 90° between edges ($\alpha = \beta = \gamma = 90°$). These two examples and other 5 systems (tetragonal, hexagonal, rhombohedral, monoclinic, and triclinic) they define the *crystal system*.

Crystal structure can also be classified based on their *Bravais lattices*, of which there are 14 different types. These are divided into 7 primitive lattices and 7 non-primitive lattices. The primitive lattice is a lattice where each unit cell corresponds to exactly one lattice point.

In non-primitive lattices, additional lattice points can be found at the *center*, *base*, or *faces* of the unit cell. These extra lattice points are referred to as *centerings*. There are three centering types: base-centered (**S**), body-centered (**B**), and face-centered (**F**). For example, a face-centered lattice has additional lattice points at the centers of all faces but not inside the unit cell, i.e., symbolized by **F**. Fig. 2.3 illustrates the 7 primitive Bravais lattices. All the terms are tabulated in Table 3.4.

Table 3.5: Terms included in CSO

Type	List of terms
Class	Crystal Structure, Lattice, Crystal System, Bravais Lattice, Unit Cell, Lattice Parameter Angle, and Lattice Parameter Length
Data Property	Centering, Lattice Parameter Length (a, b, c), and Lattice Parameter Angle (α, β, γ)

Define classes, properties, and constraints

Given the list of terms, we subsequently define the classes and the class hierarchy in CSO. In this case, we define the hierarchy between `Lattice` and `BravaisLattice`, in which the `BravaisLattice` is a specialization of the `Lattice` (cf. Fig. 3.7). `BravaisLattice` relates to its centering information via data property of `centering`, in which has a range of `xsd:string`. As a convention, please note that after we define the classes and class hierarchy, classes are now written in the so-called "upper camel case" format, e.g., `BravaisLattice`, `CrystalSystem`, and `CrystalStructure`. Additionally, object and data property in an ontology is written in the so-called "lower camel case" format, e.g., the object property of `hasLatticeParamterLength`.

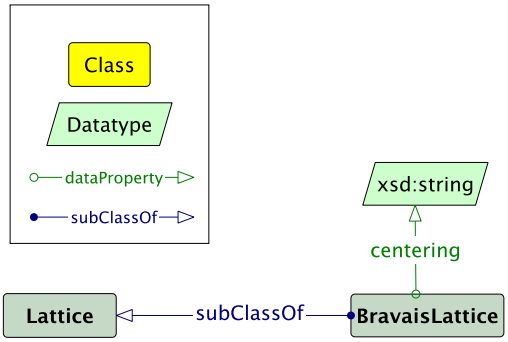


Fig. 3.7: As the `BravaisLattice` is a specialization term of `Lattice`, we relate these two classes with `subClassOf`. Furthermore, `BravaisLattice` relates to its centering information via a data property called `centering`, which has range of `xsd:string`.

As mentioned above, the `CrystalStructure` consists of the lattice and the motif, we model this class relates to `MDO:Occupancy` (as a motif) and `Lattice`, with object properties of `MDO:hasOccupancy` and `hasLattice`, respectively. To represent the atomic element name, the `MDO:Occupancy` relates to `MDO:Species` via an object property of `MDO:hasSpecies`. Ultimately, the `MDO:Species` relates to `ChEBI:Atom` via `MDO:hasElement`. The diagram of these classes relationships is shown in the Fig. 3.8.

The next step is to define the relationship between `CrystalStructure` and the space group (defined as `MDO:SpaceGroup`), as well as the point group (defined as `MDO:PointGroup`). `CrystalStructure` relates to `MDO:SpaceGroup` via the object property of `MDO:hasSpaceGroup`. This object proeperty has its own inverse property, `isSpaceGroupOf`. Consequently, `MDO:SpaceGroup` relates to `CrystalStructure` via `isSpaceGroupOf`. Furthermore, `MDO:PointGroup` relates to `MDO:SpaceGroup` via `isPointGroupOf`, which is also the inverse prop-

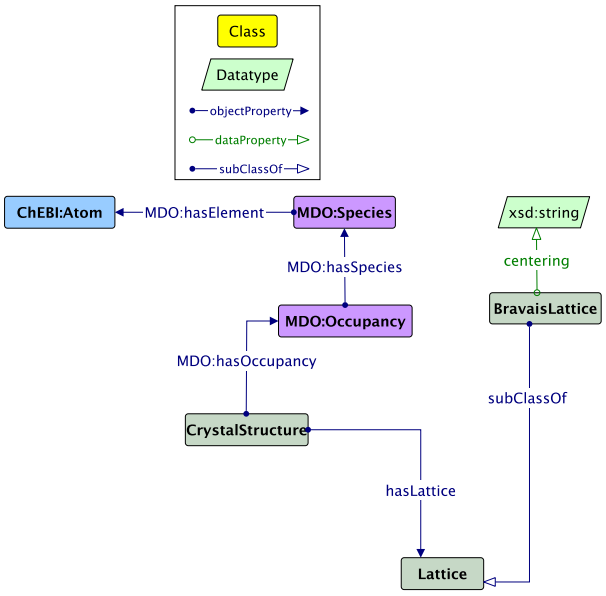


Fig. 3.8: Crystal structure relates to its motif and lattice.

erty of `MDO:hasPointGroupOf`. In `MDO:SpaceGroup`, the information related to space group ID and symbol are defined by two data properties, they are `MDO:spaceGroupSymbol` and `MDO:spaceGroupID`. Furthermore, they have a range of `xsd:string`. The diagram of these classes relationships is shown in the Fig. 3.9.

The last step is to define the relationship between `Lattice` and `UnitCell`, as well as with their lattice parameters. Moreover, the relationship between `Lattice` and `CrystalSystem` is defined. Firstly, `Lattice` relates to `UnitCell` via the object property of `hasUnitCell`. Secondly, `UnitCell` relates to two classes defining their lattice parameters: `LatticeParameterLength` and `LatticeParameterAngles`. Thirdly, `LatticeParameterLength` relates to the lattice parameter lengths information via data properties: `latticeParameterA`, `latticeParameterLengthB`, and `latticeParameterC`. Finally, `LatticeParameterAngle` relates lattice parameter angles information via data properties: `latticeParameterAngleAlpha`, `latticeParameterAngleBeta`, and `latticeParameterAngleGamma`. The diagram of these classes relationships is shown in the Fig. 3.10.

A restriction is put on the range of lattice parameters data properties, which is `xsd:double` because lattice parameters are real-valued measurements that require high precision to accurately represent crystal structures, typically expressed in angstroms (Å) or nanometers (nm) with decimal fractions (e.g., 5.431 Å for silicon). Using `xsd:double` ensures sufficient numerical precision, prevents rounding

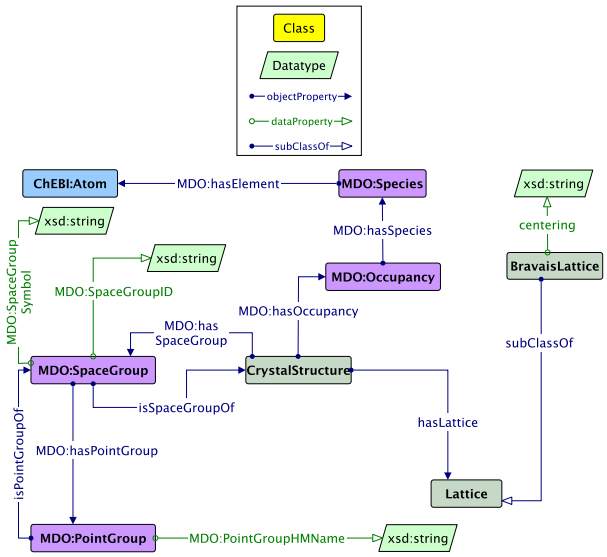


Fig. 3.9: Crystal structure relationship with the space group and point group.

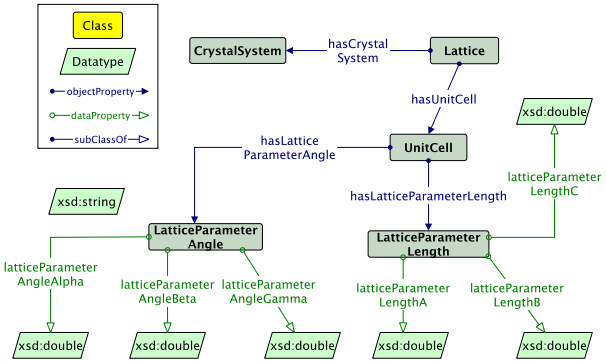


Fig. 3.10: Lattice relationship with the unit cell and crystal system.

errors, and maintains compatibility with crystallographic databases and computational models such as Density Functional Theory (DFT) simulations and Molecular Dynamics (MD). Ultimately, the complete diagram of CSO is shown in the Fig. 3.11.

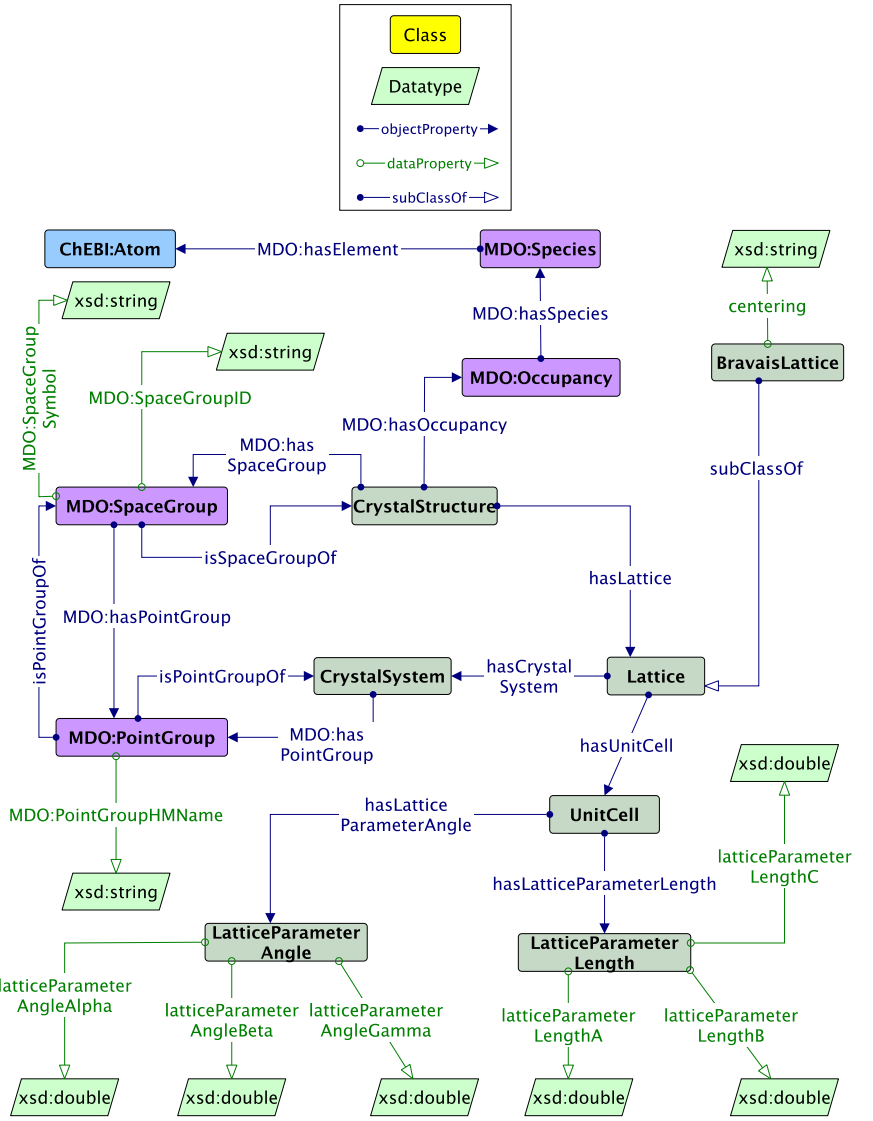


Fig. 3.11: The diagram of CSO developed by following Ontology101 method. Arrows with open arrow heads denote `rdfs:subClassOf` properties between classes. Regular arrows visualize *rdfs:domain* and *rdfs:range* restrictions on properties. Furthermore, colored boxes represent different ontologies, e.g., MDO and ChEBI.

3.2.2 Create instances

The last step in Ontology101 is to create instances or populate the ontology. Creating an instance of a class requires 1) choosing the class, 2) creating an instance of that class, and 3) filling in the property and literal value.

We can populate the ontology by giving the experiment data of a material sample. For example in the following statement, "According to the measurement of an experiment, a material specimen is a Face-centered Cubic (FCC) crystal structure with the lattice angle of 90° and length of 3.52Å". From the aforementioned information, we can conclude that:

1. There is material specimen that is an FCC crystal structure.
2. FCC has the Bravais lattice face-centering, **F**, and it is categorized as the cubic system.
3. The cubic system has unit cell which corresponds to lattice parameter angles ($\alpha = \beta = \gamma = 90°$) and lattice parameter lengths ($a = b = c = 3.52$Å).

Given the aforementioned points: number 1 and 2 above, we can populate CSO and it is illustrated in the Fig. 3.12. In the figure, the yellow points denote individuals of classes. Each individual is defined by an arrow having `rdf:type` relationship to the respective class and individuals are connected by object or data properties defined in CSO.

Given another point number 3, we can add instances in CSO as shown in the Fig. 3.13. Ultimately, we have annotated the crystal structure data using CSO, which is the process of ontology population. The complete diagram of the ontology population given the crystal structure data can be seen in Fig. 3.14.

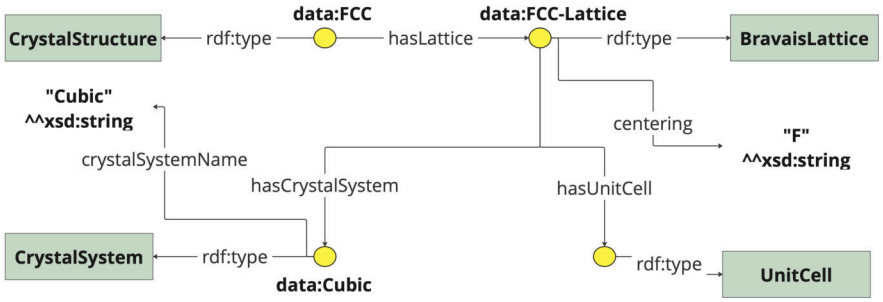


Fig. 3.12: The ontology population given that the material specimen is an FCC material and FCC has the Bravais-lattice face-centering and the cubic system.

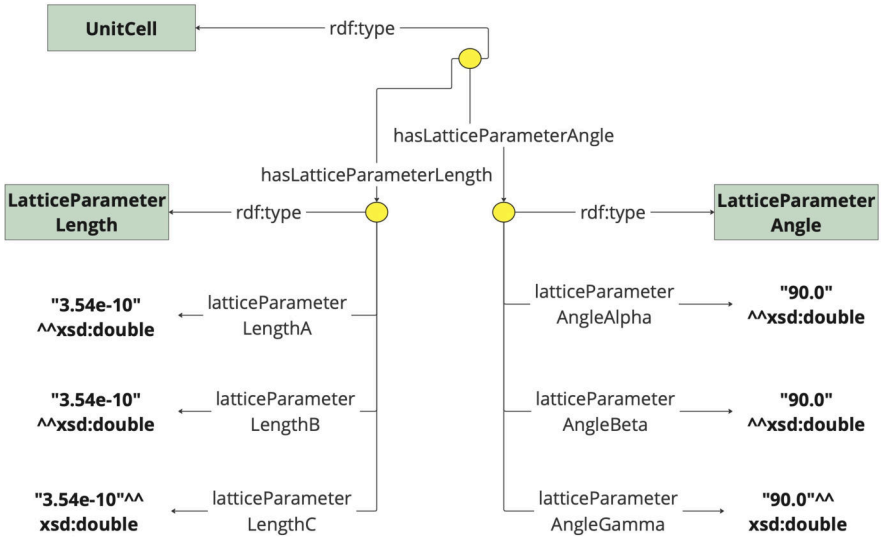


Fig. 3.13: The ontology population of the cubic system that has lattice parameters lengths and angles, $\alpha = \beta = \gamma = 90^\circ$ and $a = b = c = 3.52$Å, respectively.

3.2.3 NeOn Methodology

With the growing availability of ontologies, developing ontologies is now more focused on reusing existing ones [113]. This means that ontology development involves creating a network of ontologies where different people in different organizations may manage different resources. Therefore, given this approach to ontology engineering, it is crucial to provide strong support for the collaborative development of ontology networks.

The NeOn Methodology is a guide to creating ontologies and ontology networks. It is a scenario-based approach that facilitates various aspects of the ontology development process, including the reuse and dynamic evolution of networked ontologies in distributed environments [93]. The knowledge is subsequently contributed by different individuals, such as domain experts and ontology practitioners, at different stages of the ontology development process.

As can be seen in Fig. 3.15, there are nine scenarios introduced by the NeOn Methodology for collaboratively developing ontologies and ontology networks. These scenarios are flexible and do not need to follow a specific order. However, Scenario 1 should be included in any combination, as it covers the essential tasks required for ontology development. Each scenario is decomposed into different activities or processes and they are:

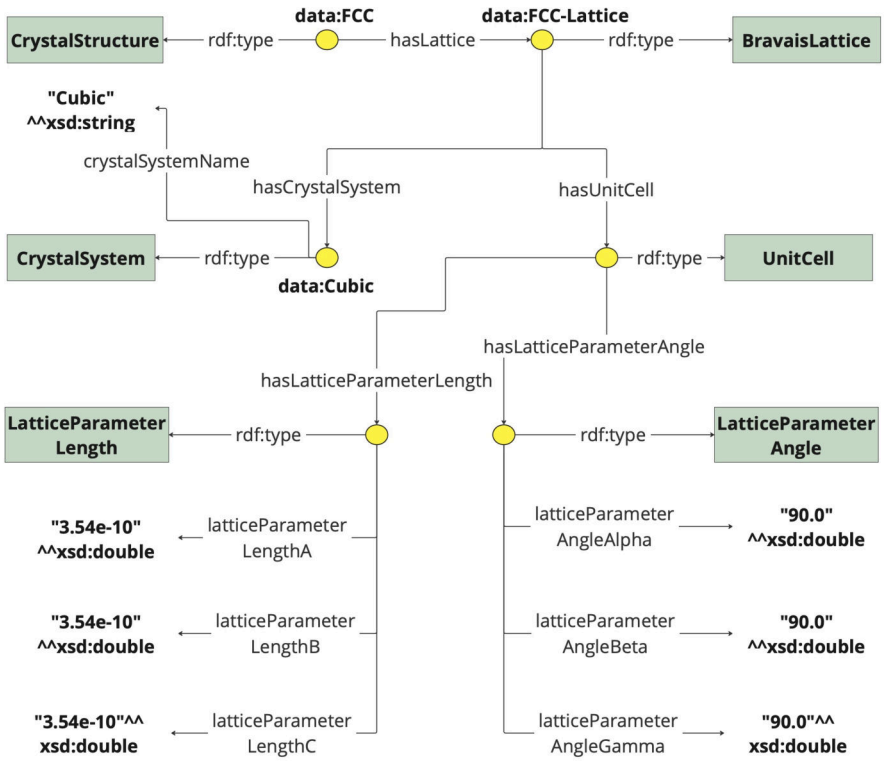


Fig. 3.14: The complete diagram of ontology population of CSO given a material specimen data.

- *Scenario 1: From specification to implementation.* In this scenario, the ontology network is developed from scratch, i.e., without reusing available knowledge resources.
- *Scenario 2: Reusing and re-engineering non-ontological resources.* In this scenario, the focus is on ontology development and how developers need to examine non-ontological resources, e.g., glossaries, dictionaries, lexicons, and thesauri, to determine which ones can be utilized to create the ontology network based on the needs and requirements. Additionally, the scenario involves transforming the chosen resources into ontologies through re-engineering.
- *Scenario 3: Reusing ontological resources.* In contrast to Scenario 2, here developers reuse ontological resources, e.g., ontologies as a whole, ontology modules, and/or ontology statements.
- *Scenario 4: Reusing and re-engineering ontological resources.* In this scenario, ontology developers both reuse and re-engineer ontological resources.

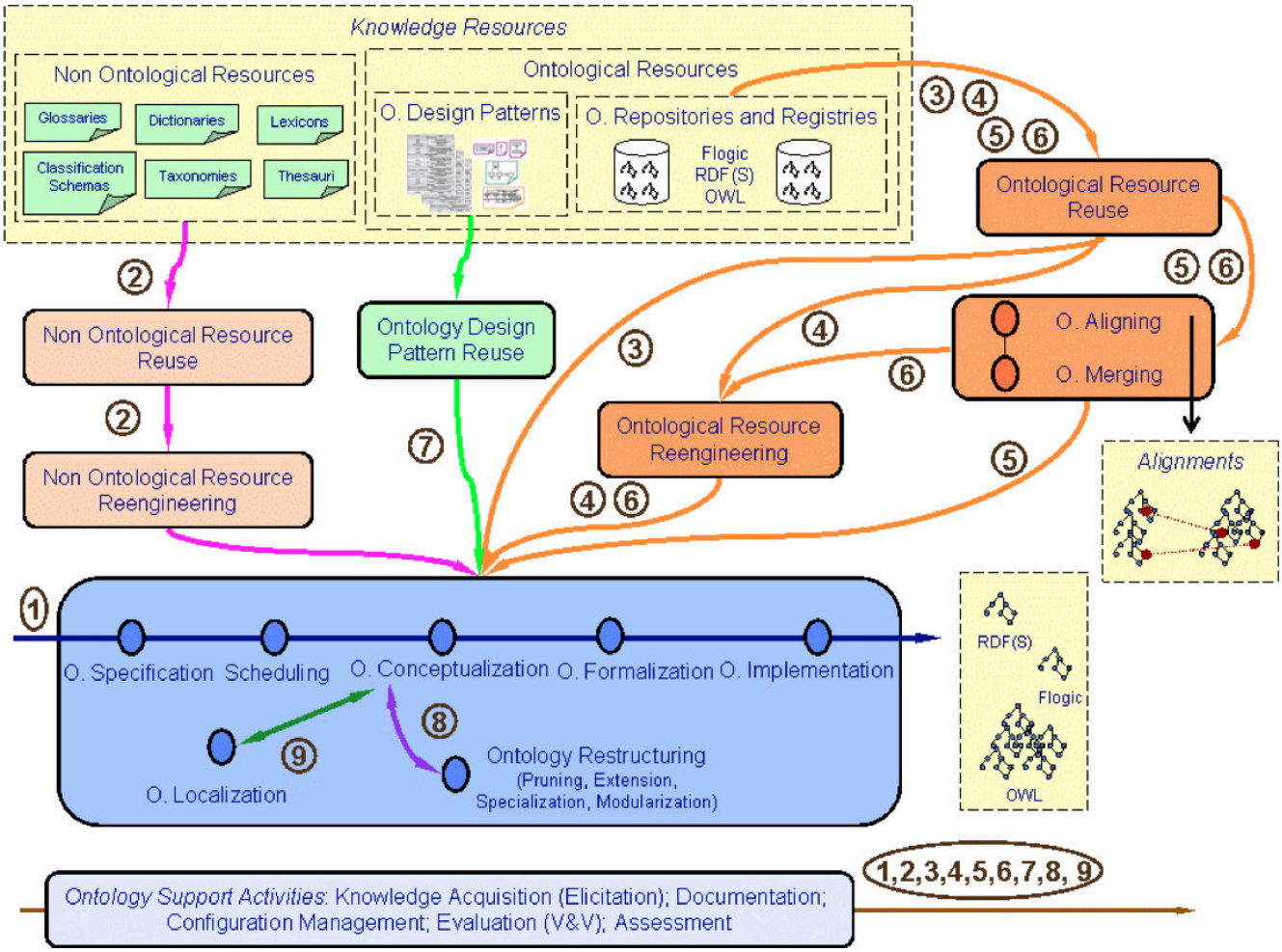


Fig. 3.15: The scenarios for building ontologies based on the NeOn Methodology. Each scenario is depicted by a circled number. The picture is copied from Suárez-Figueroa et al. [93].

- *Scenario 5: Reusing and merging ontological resources.* This scenario occurs only when ontology developers decide to reuse multiple ontological resources within the same domain and aim to develop a new ontological resource by combining these resources.
- *Scenario 6: Reusing, merging, and re-engineering ontological resources.* This scenario is similar to the previous scenario; however developers will re-engineer the set of merged resources.

- *Scenario 7: Reusing Ontology Design Patterns (ODPs) [114].* Ontology developers reuse ODPs from its repositories.
- *Scenario 8: Restructuring ontological resources.* Ontology developers modify ontological resources through modularization, pruning, extension, and specialization to be integrated into the ontology network.
- *Scenario 9: Localizing ontological resources.* Ontology developers adapt ontologies for other cultures or languages, producing multilingual ontologies.

3.3 Natural Language Processing

Natural Language Processing (NLP) is a research area in computer science and Artificial Intelligence (AI) that focuses on the communication between humans and computers using natural languages [115]. Its primary goal is to program computers to efficiently process large amounts of natural language text and extract useful information. The processing generally involves translating natural language into numbers. Subsequently, a machine/computer tries to understand numbers corresponding to natural language. Ultimately, this understanding can generate, e.g., a natural language text that reflects the understanding. Please note that we limit our scope to only focus on the natural language of English.

NLP has many real-world applications, including Information Extraction (IE). IE automatically attempts to extract information from unstructured data, e.g., natural language texts. By transforming the unstructured data, it will help humans and machines to process, link, and analyze the data. For instance, with the increase of ontologies being developed in the last couple of years, there is a need to populate ontologies given the existing data. In order to effectively populate the ontology, IE utilizing NLP can be used to extract the information from unstructured data such as text, and transform it into a knowledge graph. Thus, this section describes two NLP techniques that are used for IE, they are Named Entity Recognition (NER) and Relationship Extraction (RE) or semantic role labeling . The former is used to identify entities in the passage of text, and the latter generates the relationship between entities. In this section also, we describe the NLP system or pipeline, including tokenizer and language model. Furthermore, we briefly discuss *state-of-the-art* of large language models based on transformers architecture.

3.3.1 Named Entity Recognition and Relation Extraction

NER aims to identify a named entity of words in a text passage. A named entity is a real-world object that we can refer to by a proper name or a quantity of interest [116]. It includes a person, a place (e.g., country, city, and building), an organization, a company, a product, dates, time, and disease name. For instance, we sample the following sentence, "*B. J. Habibie was the third Indonesian Presi-*

dent, and he earned his Ph.D. from RWTH University Aachen." For the NER task, the Python module SpaCy [117] is utilized. SpaCy is a popular NLP library known for its efficiency, ease of use, and support for large-scale information extraction tasks. It offers pre-trained models that are optimized for performance, allowing for the rapid identification of various linguistic structures. Using SpaCy, the NER task is carried out by identifying several named entities (cf. Fig. 3.16). These named entities are person, ordinal, nationalities, work of art, and organization, symbolized by PERSON, ORDINAL, NORP, WORK_OF_ART, and ORG labels, respectively.

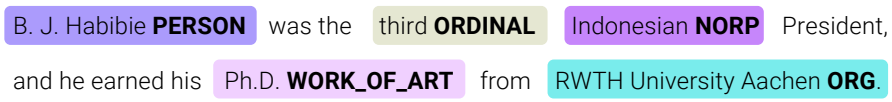


Fig. 3.16: The identification of named entities (e.g., person, organization, nationalities, and ordinal number) in a text passage using NER.

RE or semantic role labeling generates a relationship between named entities. It extracts the SPO sentences from the text passage. For example, this NLP technique can generate the relationship (predicate) between a person (subject) and an organization, school, or institution (object). Fig. 3.17 illustrates the relation extraction between a person and the university where he studied.

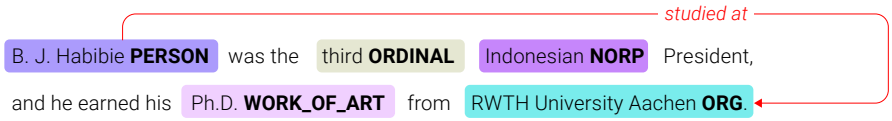


Fig. 3.17: Relation extraction given named entities in a text passage. The relationship is generated relating a person and a university where he studied.

3.3.2 Tokenization

Tokenization breaks down a string or sentence into atomic units called tokens, which are subsequently used in the model. Tokens can be words, numbers, punctuations, currencies, symbols, and any other symbols that form the building blocks of a sentence. Several tokenization strategies can be utilized. Below, we describe three strategies utilized to tokenize a passage of text.

Character Tokenization

The first tokenization strategy is character tokenization. This strategy splits the sentence or text on the character level. A Python script to tokenize at the character level is listed in Listing 3.9. This approach can handle misspellings and rare words more effectively, as it does not rely on predefined word boundaries, instead representing these as sequences of characters.

While this flexibility reduces issues related to out-of-vocabulary words, a fundamental limitation is that it neglects linguistic structures such as words and syntax [118]. Consequently, the model must infer these structures from the data, resulting in increased computational, memory, and data requirements.

Listing 3.9: Character tokenization. It tokenizes a sentence into characters.

```
text = "I love football."
tokenized_text = list(text)
print(tokenized_text)

['I', ' ', 'l', 'o', 'v', 'e', ' ', 'f', 'o', 'o',
    't', 'b', 'a', 'l', 'l', '.']
```

Word Tokenization

In this tokenization strategy, we can split the sentence into words instead of characters. One advantage of word tokenization is that it enables the model to skip the step of learning words from characters and, as a result, reduces the complexity of understanding text. In Python, a straightforward function to split the text into words is applying Python's `split()` function directly on the text (cf. Listing 3.10). This function splits the text according to a white space separator.

However, there is a problem using the word tokenization based on the white-space separator. As seen in the result of text splitting, the punctuation is not accounted for `'football.'`, i.e., the computer does not distinguish between the real "football" word and the one with a punctuation "football.". Some tokenizers have extra rules for punctuation, resolving the punctuation problem in the tokenized text. Furthermore, one can also apply stemming or lemmatization that normalizes words into their stem, e.g., "fast," "faster," and "faster" become "fast".

Listing 3.10: Word tokenization. In this tokenization strategy, the text is splitted into words.

```
text = "I love football."
tokenized_text = text.split()
print(tokenized_text)

['I', 'love', 'football.']
```

Subword Tokenization

Subword tokenization combines character and word tokenization to split rare words into smaller units, allowing the model to handle complex words and misspellings. At the same time, frequent words are kept as unique entities to keep input length manageable. A common subword tokenization algorithm used in NLP is WordPiece [119]. The WordPiece tokenizer is used by several language models, including *BERT* [120] and *DistilBERT* [121]. As listed in Listing 3.11, a sentence, "*In the football, I play in the playmaker position.*" is tokenized with WordPiece tokenizer from BERT model. We can observe that there are special `[CLS]` and `[SEP]` in the tokenized words, both mark the start and end of sequences, respectively. Furthermore, it shows that for a word "*playmaker*", the tokenizer splits it into two words: "*play*" and "*##maker*". The *##* prefix in "*##maker*" means that the preceding string is not whitespaces.

Listing 3.11: Subword tokenization. In this tokenization strategy, the strategy of charater and word tokenization are combined.

```
from transformers import AutoTokenizer

model_ckpt = 'bert-base-uncased'
tokenizer = AutoTokenizer.from_pretrained(model_ckpt)

text = 'In the football, I play in the playmaker position.'
encoded_text = tokenizer(text)
words = tokenizer.convert_ids_to_tokens(encoded_text.input_ids)
print(words)

['[CLS]', 'in', 'the', 'football', ',', 'i', 'play',
    'in', 'the', 'play', '##maker', 'position', '.', '[SEP]']
```

Tokenization Role in Language Model

Tokenization is the essential first step in any NLP system. It breaks down raw text into smaller units such as words, subwords, or characters. By transforming text into tokens, language models can analyze and learn from these numerical representations to perform tasks like text prediction, translation, and generation. performance [119]. Ultimately, tokenization lays the foundation for successfully operating language models and other NLP systems.

3.3.3 Language Model

The language model is a model predicting the probability of upcoming words or sequences of words given the history of previous words. For example, the language model can predict upcoming words given previous words of "I love " by giving the probability of the following words, e.g., the very likely *you*, or possibly *football*, but probably not the word *I* again. Additionally, the language model can assign the

probability to an entire sentence. For instance, we compare two sentences with the exact words but in different order: 1) "I love football." and 2) "Love football I.". The former will have a higher probability compared to the latter; by leveraging the language model, we can have a better sentence with correct grammar or spelling errors.

In this subsection, we will briefly discuss different types of language models and provide a formal definition of a language model. The first type we will cover is the one of the earliest statistical-based language model called *N-gram* language model. Subsequently, we will explain the neural probabilistic language model, which involves *feature vectors* and *neural networks*.

N-gram Language Model

Predicting the next word can be achieved by estimating the conditional probability of a word given its preceding words. Conditional probability (cf. Eq. 3.1) is the probability of event A occurring, given that event B has already occurred, $P(A|B)$. It is computed by dividing the probability of events A and B occur together, $P(A \cap B)$, and the probability of B, $P(B)$.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}. \tag{3.1}$$

Eq. 3.2 defines the conditional probability of predicting the upcoming word, w_n, given the history of previous words, $w_1, ..., w_{n-1}$,

$$P(w_n|w_{1:n-1}) = \frac{P(w_1, w_2, ..., w_{n-1}, w_n)}{P(w_1, w_2, ..., w_{n-1})}. \tag{3.2}$$

Here, $w_{1:n-1}$ is the abbreviation for $w_1, ..., w_{n-1}$. Both numerator and denominator can be computed using the chain rule of probability:

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, w_2, w_3, ..., w_{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_{1:k-1}). \end{aligned} \tag{3.3}$$

Eq. 3.3 links also the joint probability of a sequence and the conditional probability of a word given in previous words. This equation also suggest that we can estimate the joint probability of an entire sequence of words by multiplying a number of conditional probabilities.

In order to reduce the complexity of a language model, the *N-gram language model* is introduced. The N-gram language model computes the probability of a word given its history of entire previous words, by approximating the history by just the last few words, e.g., the last one word (bigrams), two words (trigrams), and n words (n-grams).

We exemplify the N-gram language model by choosing $N = 2$, i.e., bigrams. The bigram model approximates the word probability given all the previous words, $P(w_n|w_{1:n-1})$, by using only a preceding word, $P(w_n|w_{n-1})$. For instance, instead of calculating the probability

$$P(\text{structure}|\text{Nickel has the FCC crystal}) \tag{3.4}$$

we compute it with the probability

$$P(\text{structure}|\text{crystal}). \tag{3.5}$$

Thus, the bigram model predicts the conditional probability of the next word, by the following approximation:

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1}). \tag{3.6}$$

Furthermore, when we use N as an N-gram size, e.g., bigrams, trigrams, and four-grams, the general equation for approximating the probability of a word given its entire context as follows:

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1}). \tag{3.7}$$

Given the bigram assumption, we can compute the probability of a complete sentence by substituting Eq. 3.6 to Eq. 3.3, resulting the Eq. 3.8

$$P(w_{1:n}) = \prod_{k=1}^{n} P(w_k|w_{k-1}). \tag{3.8}$$

To exemplify, suppose we want to estimate the probability of the sentence "Nickel has the FCC crystal structure". Using a bigram model, the probability of the sentence is calculated by multiplying the conditional probabilities of each word given the previous word. This can be expressed as:

$$\begin{aligned} &P(\text{“Nickel has the FCC crystal structure”}) = P(\text{“Nickel”})... \\ &... \cdot P(\text{“has”}|\text{“Nickel”}) \cdot P(\text{“the”}|\text{“has”}) \cdot P(\text{“FCC”}|\text{“the”})... \\ &... \cdot P(\text{“crystal”}|\text{“FCC”}) \cdot P(\text{“structure”}|\text{“crystal”}). \end{aligned} \tag{3.9}$$

The N-gram language model has a significant flaw in its inability to recognize the semantic similarity between words. It treats each word as an independent entity, without considering the relationships or connections between words with similar meanings. This limitation can be addressed by Neural Probabilistic Language Model (NPLM), which leverage neural networks to model word similarities through continuous vector representations, such as word embeddings, and capture more nuanced relationships in text.

Neural Probabilistic Language Model

The NPLM was developed to address the limitations of the N-gram language model. It incorporates distributed word representations, the so-called *word embeddings*, allowing the model to capture semantic similarities between words [122]. Unlike the N-gram language model, which rely on discrete word occurrences, NPLM uses continuous vector spaces to better generalize across similar words and improve context-based predictions. E.g., in the sentence "Nickel has the FCC crystal structure". The NPLM can generalize to make the sentence "Chromium has the BCC crystal structure" almost as likely because "FCC" and "BCC" have similar contexts and grammatical roles.

To overcome these problems, NPLM introduces an approach summarized as follows:

1. Associate each word in the vocabulary with a *feature vector*.
2. Express the joint probability function of word sequences in terms of the feature vectors.
3. Learn the word feature vectors and function parameters.

Table 3.6: Density, conductivity, hardness, and melting point data of materials

Material	Density (g/cm^3)	Conductivity (W/m·K)	Hardness (HV)	Melting Point (°C)
Nickel	8.90	90.7	638	1455
Copper	8.96	398	343	1085
Steel	7.85	50.2	190	1370
Aluminum	2.70	235	167	660

A word feature vector or *word embedding* represents a word as a vector in a vector space, $\mathbb{R}^n$. The vocabulary assigns each word an index, i, so that it becomes w_i, which is then projected into an n-dimension vector space and results in a feature vector, $\mathbf{X}$. This n-dimension value also represents the number of features which is usually much smaller than the vocabulary size, varies around 10-2000.

Using the Table 3.6, we exemplify the generation of a word embedding. The table represents data on materials in terms of their properties: Density, conductivity, hardness, and melting point. We can construct the word embedding using the Table 3.6, resulting in the Eq. 3.10, a four-dimensional vector space. Each row corresponds to the material name in the table, and columns are the value of material properties. For example, on the first row, we have a Nickel material, which has density, conductivity, hardness, and melting point of 8.90 g/cm^3, 90.7 W/m·K, 638 HV, and 1455 °C, respectively. However, it is important to note that we do not manually define these embeddings; instead, they are learned automatically through training on a domain-specific dataset, allowing the model to capture meaningful relationships between words based on the data.

$$\mathbf{X} = \begin{bmatrix} 8.90 & 90.7 & 638 & 1455 \\ 8.96 & 398 & 343 & 1085 \\ 7.85 & 50.2 & 190 & 1370 \\ 2.70 & 235 & 167 & 660 \end{bmatrix}. \tag{3.10}$$

In terms of probability function, NPLM leverages the function which is a product of the conditional probabilities of the next word, given the previous ones. To improve the performance, the function has parameters that can be trained or fine-tuned during the training process. Additionally, feature vectors or word embeddings associated with each word are learned to understand various language contexts in a text corpus used for this process, e.g., understanding the polysemy and homonymy.

The learned word embeddings lays the foundation for more advanced models, such as LLMs. In LLM, these word embeddings are further refined and scaled to capture even more intricate relationships and patterns within enormous data. By leveraging these embeddings, LLM is able to perform complex tasks, e.g., text generation, translation, and question answering, while maintaining a deep understanding of word semantics and context.

NPLM has limitations in handling long-range dependencies, as it only considers a limited history of previous words, resulting in difficulty capturing broader contextual information. Additionally, it relies on sequential processing, which hinders their ability to take advantage of parallel processing, reducing efficiency when working with large datasets.

In the following part, we address these challenges by exploring Transformer-based language model, the current state-of-the-art in NLP, which excel at capturing long-range dependencies and enabling more efficient parallel processing. Transformer-based language model not only aims to solve the issues present in NPLM but also address limitations found in N-gram language model, such as their inability to model long-range context and recognize semantic relationships between words.

3.3.4 Transformer and Large Language Model

Transformer is a type of neural network architectures designed to process sequences by transforming an input sequence of vector representations or embeddings, into an output sequence of embeddings. Both the input and output embeddings maintain the same dimensionality, but the output is represented in a different or new space [123]. Nowadays, transformer have become popular due to their ability to perform better and more efficiently in other NLP tasks, e.g., sentiment analysis, question and answering, text summarization, NER, and RE.

The transformer consists of layers called transformer blocks (cf. Fig. 3.18). These blocks are constructed by integrating simple linear layers, feed-forward networks, and (masked) multi-head attention, which are key recipes in the transformer. These blocks can be further categorized into two types: Encoders and decoders.

The underlying difference between these two blocks is the availability of a layer called masked multi-head attention in the decoder block, avoiding the current focus word from attending to the following words and making it suitable for the text generation task.

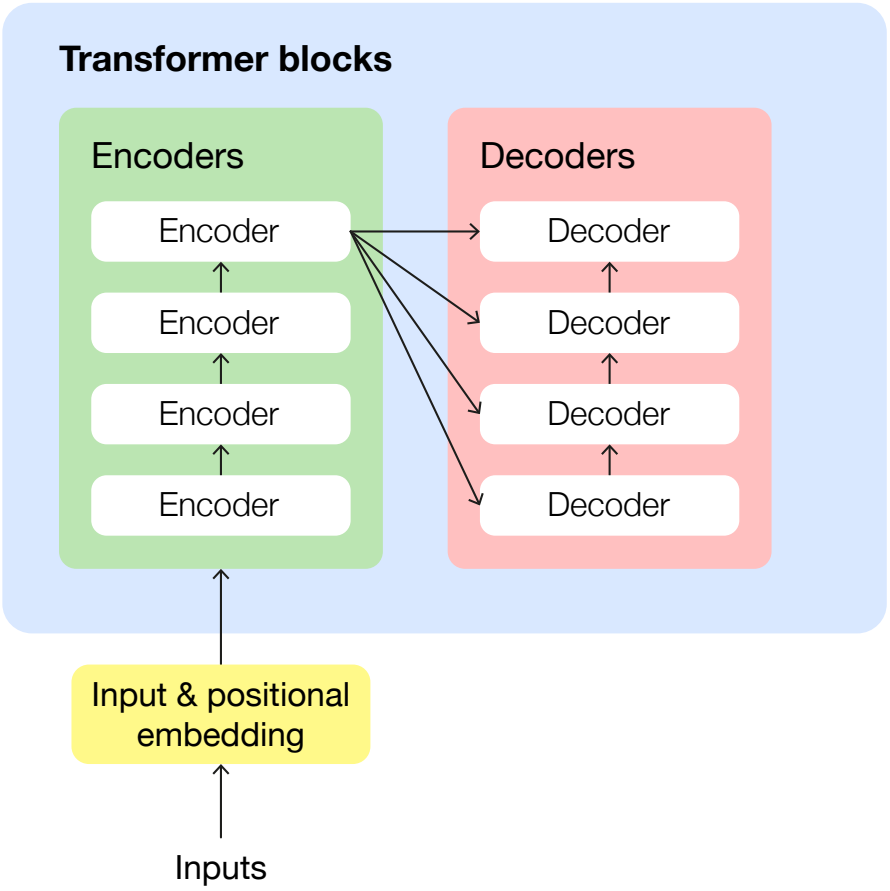


Fig. 3.18: Transformer blocks. Transformer blocks comprise encoders and decoders block. Each block is the stack of multiple encoders or decoders.

In this part of thesis, we briefly describe the fundamental concept underlying the transformer and building blocks to establish the network, e.g., token and positional embedding, self-attention, multi-head attention, and transformer

blocks. In the last part, we exemplify several examples of recent development of transformers-based language model or LLMs.

Token and Positional Embedding

Before feeding the input into the transformer block, the tokenization of a sequence of text occurs (cf. Subsec. 3.3.2). Tokenization yields a sequence of N-number tokens. Subsequently, each token is assigned an embedding, resulting in a set of token embeddings, $\mathbf{X}$, which has a shape $[N \times d]$, where d is the embedding dimension. The embedding assignment of each token (cf. Eq. 3.11) is carried out by matching the token index with row index of the embedding matrix, $\mathbf{E}$, a matrix that has shape $[|V| \times d]$, where V is the number of tokens in the vocabulary.

$$\mathbf{x_i} = [\mathbf{E_{i1}}, \mathbf{E_{i2}}, \mathbf{E_{i3}}, ..., \mathbf{E_{id}}] \tag{3.11}$$

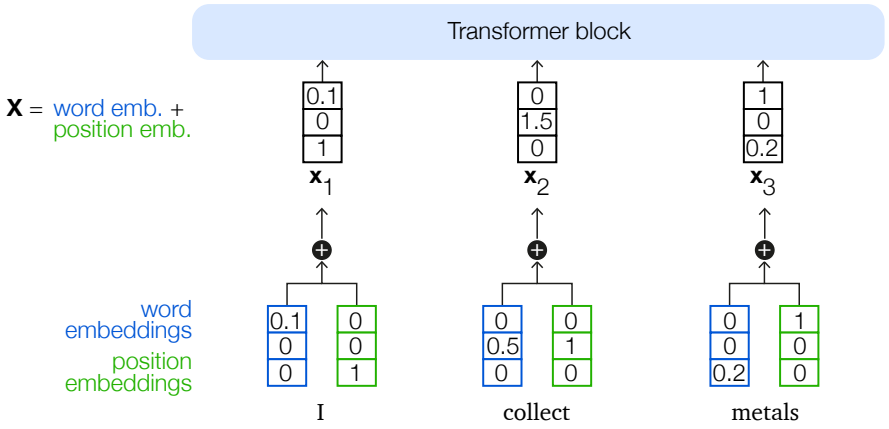


Fig. 3.19: Word and positional embeddings. Before a passage is conveyed to transformer blocks, it is tokenized and assigned an embedding. An embedding consists of the word and positional embedding.

Token embeddings are not position-aware. In this regard, transformer combines token embeddings with *positional embeddings* with the same shape, representing the position of each token in the sequence (cf. Fig. 3.19). This combination of these two embeddings results in the matrix $\mathbf{X}$ with each row i is the representation of i-th token in the input. This combination is computed by adding $\mathbf{P}[i]$, the positional embedding of position i, to each row i in the matrix $\mathbf{X}$.

Attention

The fundamental concept underpinning transformer is *attention*. Attention allows the model to focus on certain parts of the input more than others, depending on their relevance to the task at hand. The importance, or weight, assigned to each input is determined by the input values themselves, which enables the model to capture important patterns in sequential or structured data [124].

Here, we try to clarify the intuition on how the attention works in natural language and give an illustration of a specific case. For instance, consider two sentences:

1. The stress applied to the steel increased its strength.
2. The team was under stress to complete the experiment.

In both sentences, the word "stress" appears, but it carries different meanings depending on the context. This difference can only be understood by looking at the other words in the sentence. Additionally, certain words play a more important role in determining the interpretation of "stress". In the first sentence, "applied" and "steel" words strongly suggest that "stress" refers to the physical action given to a material, e.g., compression or tension. In the second sentence, "team" and "complete" words indicate that "stress" here is psychological pressure experienced by a group of people, e.g., a strain of a deadline. This demonstrates how attention works: the meaning of a word in a sentence is determined by the context provided by the other words to which it attends. In Fig. 3.20, we illustrate the concept of attention given the first sentence.

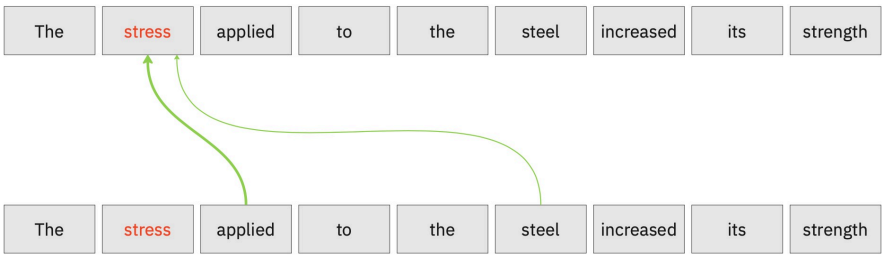


Fig. 3.20: An attention illustration, adapted from Bishop and Bishop [123], is the interpretation of the work "stress" is influenced by words attending it, including "applied" and "steel". The thickness of the line illustrate the strength of the influence.

Self-attention

We have described previously the concept of attention, which is the general mechanism for focusing on important parts of an input. *Self-attention* is a specialized form of attention used in the transformer. Instead of only focusing on important parts of a sequence, self-attention allows each token in the input to attend to every other token within the same sequence. This means that each word (or token) can build a context-aware representation by looking at the relationships between all other words in the sequence.

The self-attention process involves three essential roles: $\mathbf{q}$, $\mathbf{k}$, and $\mathbf{v}$, which are *query*, *key*, and *value*, respectively. The intuition [118] behind these three roles is to consider the process of shopping in a supermarket. In this scenario, the shopper has a list of ingredients (the *queries*) they intend to purchase. As they walk through the aisles, they inspect the items on the shelves, using the labels (the *keys*) to identify whether an item matches the ingredients on their list. When a match is found, the shopper selects the corresponding item (the *value*).

However, in the self-attention, this process is abstracted. Rather than returning exact matches, self-attention generates a weighted, "smoothed" combination of items based on the similarity between the query and key. For example, instead of retrieving ten eggs exactly as specified, the mechanism may return a mixture of eight eggs, a chicken wing, and part of a roasted chicken. This illustrates how self-attention integrates information by weighting contributions from similar inputs.

Self-attention computation

In order to compute the self-attention, the transformer introduces weight matrices $\mathbf{W}^Q, \mathbf{W}^K$, and $\mathbf{W}^V$. Through the Eq. 3.12, each input vector, $\mathbf{x_i}$, is subsequently projected into a representation of $\mathbf{q}$, $\mathbf{k}$, and $\mathbf{v}$.

$$\mathbf{q_i} = \mathbf{x_i}\mathbf{W}^Q; \mathbf{k_i} = \mathbf{x_i}\mathbf{W}^K; \mathbf{v_i} = \mathbf{x_i}\mathbf{W}^V \tag{3.12}$$

The shape of these matrices/weights are $\mathbf{W}^K \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}^Q \in \mathbb{R}^{d \times d_k}$, and $\mathbf{W}^V \in \mathbb{R}^{d \times d_v}$, where d, d_k, and d_v are the dimension of input vector, the key and query vector, and the value vector, respectively. In the original transformer work [125], the authors assign a value of 512 to d and 64 to both d_k and d_v.

Eq. 3.13 computes the similarity score between $\mathbf{q_i}$ and $\mathbf{k_j}$ projections:

$$\text{score}(\mathbf{x_i}, \mathbf{x_j}) = \frac{\mathbf{q_i} \cdot \mathbf{k_j}}{\sqrt{d_k}}. \tag{3.13}$$

This is done by computing the dot product between these two vectors, which is a simpler but related approach to cosine similarity. The dot product result is then scaled down by the square root of the dimension value of the key vector, d_k, to prevent arbitrarily large values, ensuring more stable gradients during training.

After we compute the score similarity, we normalize the score with the softmax to create a vector of weights, α_{ij} (cf. Eq. 3.14).

$$\begin{aligned}\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{x_i}, \mathbf{x_j})) \\ &= \frac{\exp(\text{score}(\mathbf{x_i}, \mathbf{x_j}))}{\sum_{k=1}^{N} \exp(\text{score}(\mathbf{x_i}, \mathbf{x_k}))}\end{aligned} \tag{3.14}$$

Ultimately, self-attention results in the output, $\mathbf{a}_i$, as seen in Eq. 3.15, and all the mentioned self-attention calculation is illustrated in Fig. 3.21.

$$\mathbf{a_i} = \alpha_{ij}\mathbf{v_j} \tag{3.15}$$

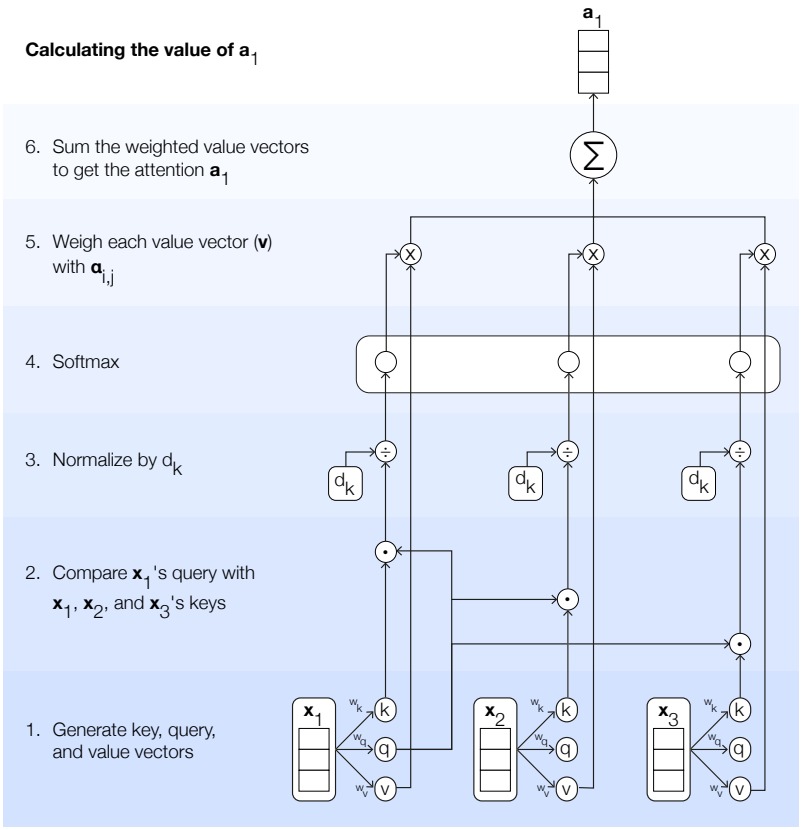


Fig. 3.21: Calculation of a_1 leveraging the self-attention.

Parallelization in Transformer

Transformer leverages the capability to compute the self-attention mechanism in parallel. The parallel computation is carried out by utilizing the vectorization process on modern Graphics Processing Units (GPUs). The ability to vectorize is made possible by the self-attention architecture, which independently calculates the self-attention output. To compute in parallel, it packs the input embeddings of the N tokens of an input sequence into a single matrix, $\mathbf{X} \in \mathbb{R}^{N \times d}$, and each row is $\mathbf{x_i}$. In order to have the matrix $\mathbf{X}$ with same dimension over many sequences, we set the maximum value of the token length, e.g., large language models can have a token length, $N = 1024$, 2048, or 4096, thus each of $\mathbf{X}$ has between 1K to 4K rows, each has the dimensionality of the embedding, d. The parallel version of self-attention equations are summarized in Eq. 3.16 and Eq. 3.17.

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q;\ \mathbf{K} = \mathbf{X}\mathbf{W}^K;\ \mathbf{V} = \mathbf{X}\mathbf{W}^V \tag{3.16}$$

In the decoder block, transformer masks out the $\mathbf{Q}\mathbf{K}^T$ matrix, eliminating any knowledge of words that follow in the sequence. The masked-out matrix, which is also a quadratic matrix, assigns the upper triangular portion of the matrix to $-\infty$, leading to softmax of these matrix is equal to zero (cf. Fig. 3.22).

$$\mathbf{A} = \text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{3.17}$$

Multi-head Attention

Transformer employs a more complex form of attention than the single self-attention method we have discussed. This complexity arises because words within a sentence can have numerous relationships. For instance, different words in a sentence can have various syntactic, semantic, and discourse connections. It would be challenging for a single self-attention model to effectively capture all these diverse relationships among its inputs.

Transformer addresses the mentioned challenge by leveraging *multi-head attention* layers. These layers consist of multiple self-attention units, known as heads, positioned at the same depth within the model. Each head has its own set of parameters, which are not predefined but learned during the model's training process. Through the training process, each head independently learns to attend to different aspects of the relationships between inputs. While the heads are independent, they capture complementary information from the same input.

The outputs from all heads are then concatenated and passed through a linear layer to combine their different perspectives into a unified representation. However, it is important to note that we cannot explicitly control or decide which head learns what. Instead, the model naturally distributes attention across heads, allowing them to focus on different syntactic, semantic, or long-range dependencies.

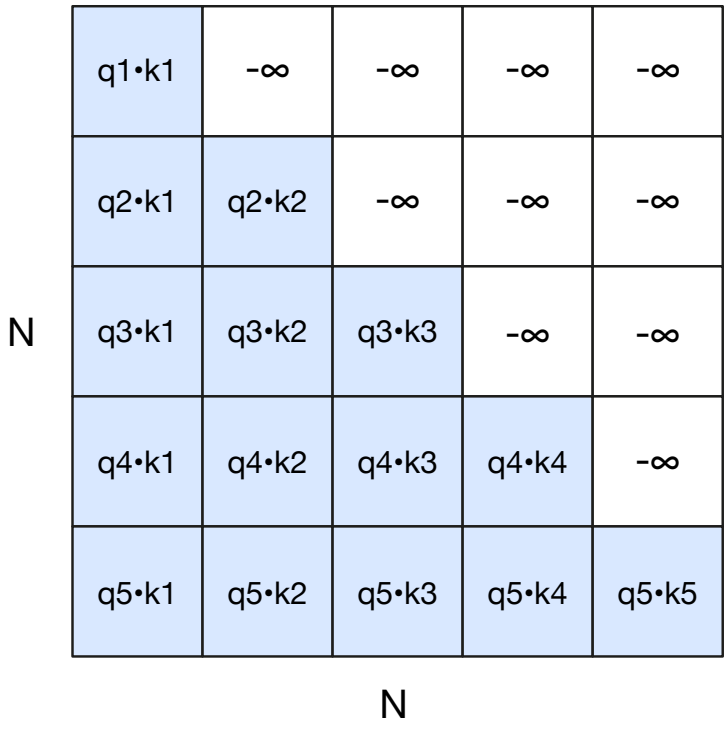


Fig. 3.22: In decoders, the transformer masks out the $\mathbf{QK}^T$ matrix, avoiding knowledge of the following words in the sequence.

Multi-head attention computation

We revisit the self-attention mechanism by defining the key, query, and value matrices: $\mathbf{W}_i^K$, $\mathbf{W}_i^Q$, and $\mathbf{W}_i^V$, where i represents the head index. These matrices transform the inputs into distinct embeddings for each head, while the rest of the self-attention computation remains unchanged. In multi-head attention, the dimensions of the input and output, d, as well as the key and query embeddings, d_k, and value embeddings, d_v, are similar to those in single self-attention. However, the multi-head attention introduces a new parameter called the number of heads, h.

For each head i, we define weight matrices $\mathbf{W}_i^K \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_i^Q \in \mathbb{R}^{d \times d_k}$, and $\mathbf{W}_i^V \in \mathbb{R}^{d \times d_v}$. As shown in Eq. 3.18, these weight matrices are used to transform the packed inputs, $\mathbf{X}$, into $\mathbf{Q} \in \mathbb{R}^{N \times d_k}$, $\mathbf{K} \in \mathbb{R}^{N \times d_k}$, and $\mathbf{V} \in \mathbb{R}^{N \times d_v}$,

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_i^Q;\ \mathbf{K} = \mathbf{X}\mathbf{W}_i^K;\ \mathbf{V} = \mathbf{X}\mathbf{W}_i^V \tag{3.18}$$

Subsequently, we can compute the self-attention for each head using Eq. 3.19.

$$\mathbf{head}_i = \text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \tag{3.19}$$

Ultimately, as seen in Eq. 3.20, the outputs from every head are concatenated to produce a single output with dimensionality $N \times hd_v$. The concatenation operator ($\oplus$) represents the process of stacking vectors or matrices along a specified dimension, ensuring that all head outputs are combined into a unified representation. This result is then reshaped to match the original output dimension for each token through a linear projection $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d}$, leading to the final self-attention output $\mathbf{A}$ of shape $[N \times d]$.

$$\mathbf{A} = \text{MultiHeadAttention}(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O \tag{3.20}$$

Fig. 3.23 illustrates the multi-head attention mechanism in the Transformer. Note that the term "head" in this context differs from the "transformer downstream task head" (cf., Eq. 3.3.4), which refers to the additional layer applied on top of the transformer blocks (body).

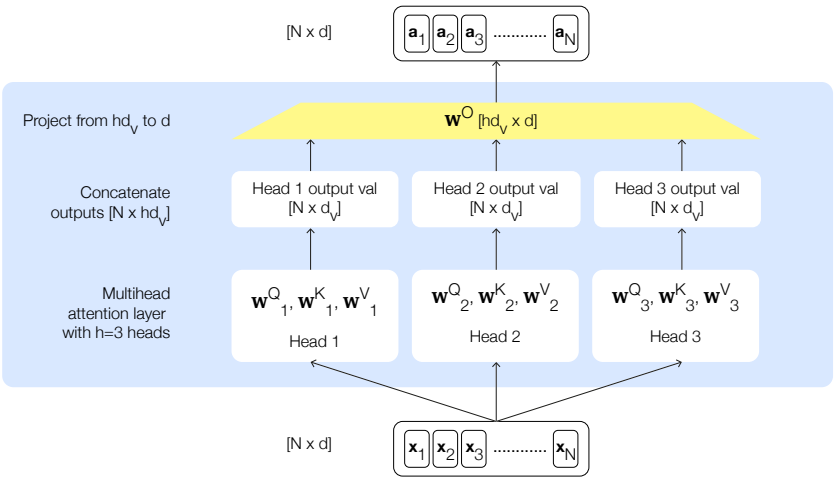


Fig. 3.23: The multi-head attention adapted from [126]. Transformer addresses the complexity of words within the sentence in terms of, e.g., syntactic, semantic, and discourse connections by leveraging multi-head attention. Multi-head attention is multiple self-attention units, known as heads, positioned at the same depth within the model. Each head focuses on different aspects of the relationship among inputs and language structure.

Transformer Block

We have discussed the self-attention mechanism, which is the main recipe in the transformer. However, in order to build a complete transformer, the remaining ingredients should be also included. The remaining ingredients are a feed-forward layer, residual connections, and normalizing layers (layer norm). Fig. 3.24 and Fig. 3.25 illustrate the encoder and decoder block consisting of several components.

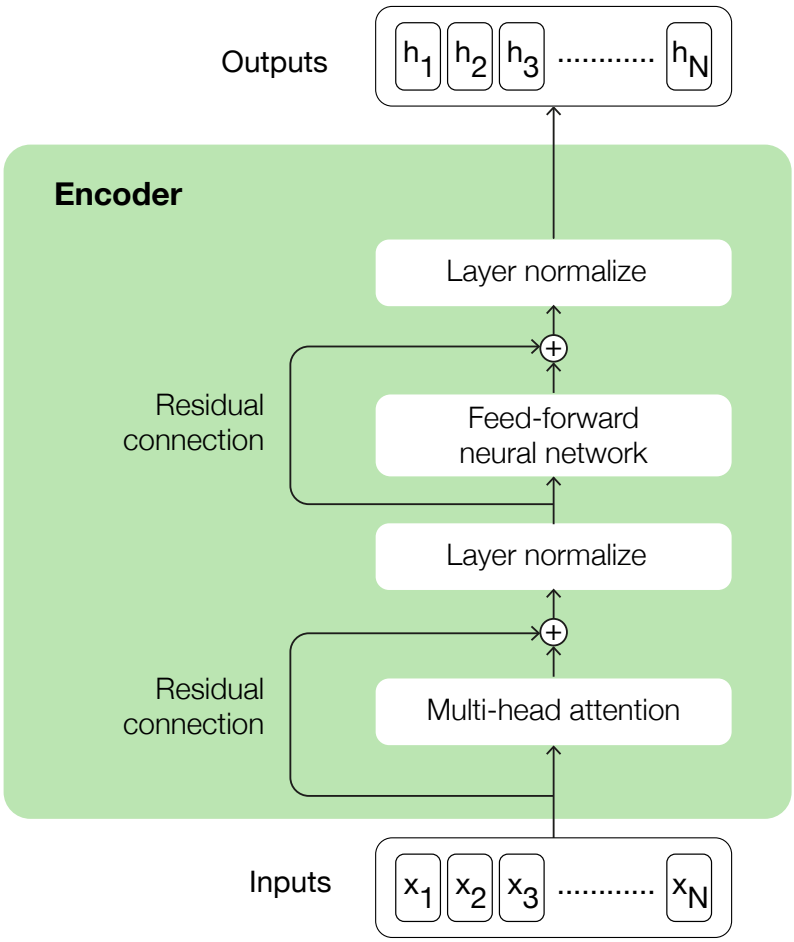


Fig. 3.24: Encoder block. The encoder block consists of multi-head attention, layer norm, feed-forward neural network, and residual connection.

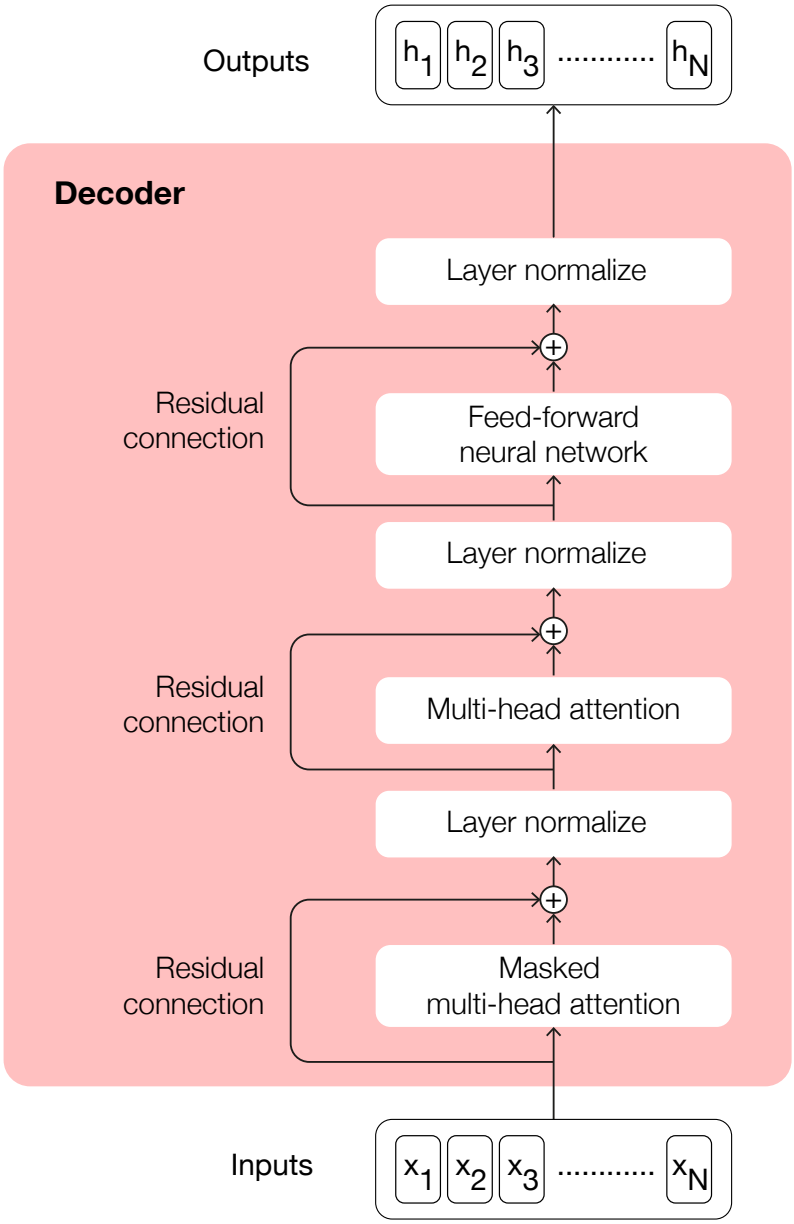


Fig. 3.25: Decoder block. The decoder block consists of multi-head attention, masked multi-head attention, layer norm, feed-forward neural network, and residual connection.

Feed-forward layer

The feed-forward layer in a transformer block is a simple two-layer fully connected neural network, i.e., one hidden layer with two weight matrices, $\mathbf{W_1}$ and $\mathbf{W_2}$, and parameters $\mathbf{b_1}$ and $\mathbf{b_2}$. This layer is also called the *position-wise feed-forward layer* due to its process computing each embedding independently. The weights in this layer are kept the same for each token in a sequence, but the parameters are different from layer to layer. Furthermore, this layer is independent for each token position, enabling it to be computed in parallel. In the original work of transformer, authors utilizes Eq. 3.21.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)\,W_2 + b_2 \tag{3.21}$$

Residual connections

In the context of transformers, residual connections refer to connections that deliver information directly from a lower layer to a higher layer without passing through the intermediate layer. In transformers, these connections are performed by summing a layer's input vector with its output vector before forwarding it. Fig. 3.24 illustrates a transformer block, where it employs residual connections with both the attention and feed-forward sublayers.

Layer norm

Layer normalization or layer norm is a technique used to improve training performance in deep neural networks. It works by keeping the values of a hidden layer within a range that makes gradient-based training easier. One variation of the normalization technique is called layer norm. Layer norm takes a single vector, particularly the token at the position-i, and the output is a normalized vector with the dimensionality of d. The first step in layer norm is to calculate the mean, μ, and standard deviation, σ, over the vector elements that need to be normalized. Given a hidden layer with dimensionality, d_h, these values are calculated as following:

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i \tag{3.22}$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2} \tag{3.23}$$

Given these two values of μ and σ, we can compute the vector components normalized by subtracting the mean for each component and dividing by the standard deviation as following:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma} \tag{3.24}$$

Ultimately by leveraging the standard implementation, the layer norm, seen in Eq. 3.25, has now two new learnable parameters, γ and β, representing gain and offset values.

$$\text{LayerNorm} = \gamma\hat{\mathbf{x}} + \beta \tag{3.25}$$

A transfomer block

We put all transformer ingredients together, and the function computed by a transformer block can be express as following:

$$\mathbf{O} = \text{LayerNorm}(\mathbf{X} + \text{SelfAttention}(\mathbf{X})) \tag{3.26}$$

$$\mathbf{H} = \text{LayerNorm}(\mathbf{O} + \text{FFN}(\mathbf{O})) \tag{3.27}$$

These transformer blocks can be stacked together to build large language models. For example, transformers in LLMs can range from 12 blocks (used in models like T5 or GPT-3-small) to 96 blocks (used in GPT-3 large), with even more layers in more recent models.

Transformer Task Specific Head

A transformer model usually consists of a task-independent body and a task-specific head. In the previous part, we have discussed a task-independent body which contains a number of stack of transformer blocks and each block comprises the (masked) multi-head attention, feed-forward network, residual connections, and layer norm. However, in order to be able to carry out an NLP task, a transformer model needs a task-specific head of a neural network added on top of the basic transformer architecture.

Fig. 3.26 illustrates language modelling head, a task-specific head of a transformer to predict the next word given the preceding words. The language modelling head comprise the stack of the unembedding and softmax layers. The former projects the output of the last block of a transformer body into a logit vector or score vector, $\mathbf{u}$, that has a shape $[1 \times |V|]$, where V is the possible words in the vocabulary (cf. Eq. 3.28).

$$\mathbf{u} = \mathbf{h}_{\mathbf{N}}^{\mathbf{L}}\,\mathbf{E}^{\mathbf{T}} \tag{3.28}$$

The latter turns the logits, $\mathbf{u}$, into the probabilities $\mathbf{y}$ over the vocabulary (cf. Eq. 3.29).

$$\mathbf{y} = \text{softmax}(\mathbf{u}) \tag{3.29}$$

We can utilize the probabilities, $\mathbf{y}$, to assign a probability to a given text. Furthermore, as the language model task is to predict the next word given the probability

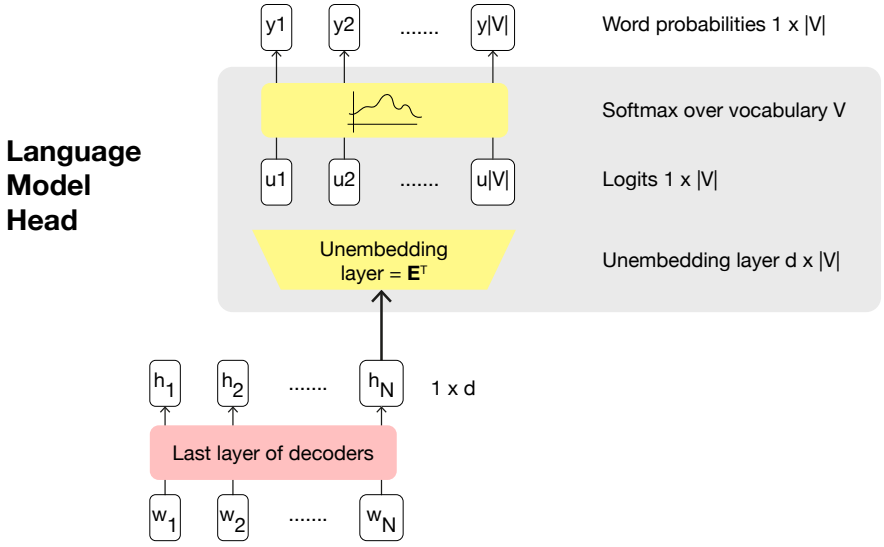


Fig. 3.26: Language model head adapted from [126]. A task-specific head of a transformer to predict a word given the preceding words.

of previous words, we can *sample* a word from these probabilities, **y**, leveraging, e.g., random sampling, top-k sampling, or temperature sampling.

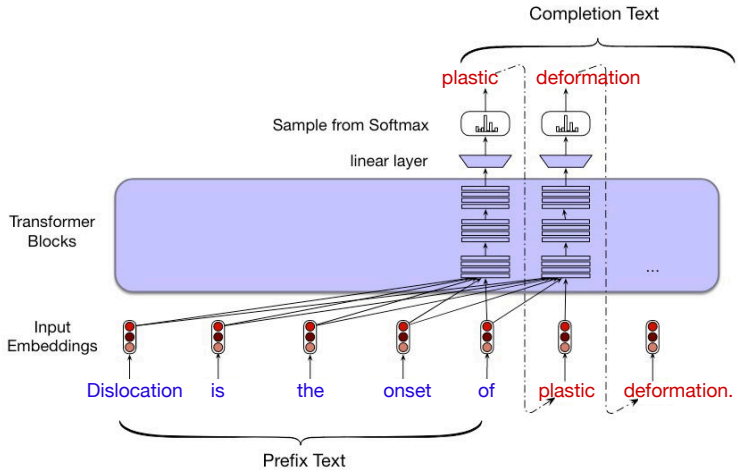


Fig. 3.27: Causal language model leverages transformer-based architecture for text completion task. This figure is adapted from [126].

The combination transformer blocks and language modelling heads forms the foundation of autoregressive or causal language modeling, where each token is predicted sequentially based only on the tokens that precede it. Fig. 3.27 depicts the example of causal language modelling. The model completes the given text or *prefix text*: "Dislocation is the onset of ", with the *completion text*: "plastic deformation".

Large Language Models

The recent advanced development in the field of machine learning has been the creation of very large transformer-based neural networks, the so-called Large Language Models (LLMs). The term "'large" in LLM refers to the number of trainable weight and bias parameters used in the network. At the time of writing, the number of parameters in a LLM can reach the order of trillions (10^{12}). Training LLM is computationally expensive due to the sheer scale of these parameters. Nonetheless, LLM have demonstrated the ability to surpass previous state-of-the-art models in various NLP tasks, boasting their extraordinary performance across a wide range of NLP tasks.

In addition to the availability of large datasets, the training of LLM has also been facilitated by the advent of massively parallel training hardware based on Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). Furthermore, the transformer architecture has played a key role in developing LLMs because it efficiently uses such hardware. Very often, increasing the size of the training dataset and trained parameters leads to improvements in performance outpacing the architectural improvements or incorporating more domain knowledge [127].

There are hundreds of LLMs utilizing the transformer, and most of them can be categorized into one of three types:

- *Encoder-only*. These models project an input sequence of text into a feature vector or embedding, which gives a rich numerical representation. Subsequently, this representation attends contexts on the left (before the token) and the right (after the token). This is called bidirectional attention. Encoder-only models are well-suited for tasks such as text classification and NER. BERT [120] and RoBERTa [128] are two of many variants of encoder-only transformers.
- *Decoder-only*. In contrast with the encoder-only transformer, the representation in the decoder-only model only attends the left context. This is called *autoregressive attention*, where each token can only attend to previous tokens, ensuring no future information is used. This approach is crucial for tasks such as text generation. The family of GPT [129], LlAMA [130], and Falcon [131] are three examples of decoder-only transformers.
- *Encoder-decoder*. This transformer type is the initial idea of transformer development, combining the ability of encoding and decoding for machine translation tasks and text summarization. BART [132] and T5 [133] are two examples of encoder-decoder transformers

3.4 Summary

We have described the underlying methods used in this thesis. In Sec. 3.1, we detailed how to transform human-readable data into machine-interoperable formats. This process involves the use of several technologies, such as RDF and SPARQL. Additionally, by leveraging standard vocabulary frameworks like the, RDFS and OWL, and ontology engineering (cf. Sec. 3.2), we were able to develop domain-specific knowledge representations, or ontologies, that are also machine-interoperable.

In the final part of this chapter (cf. Sec. 3.3), we introduced the text mining framework used for addressing information extraction tasks. The goal of information extraction is to extract relevant information from unstructured natural language text and transform it into structured data formats that can be directly processed by machines, such as JSON. This extraction process relies on NLP techniques. In this section, we also briefly elaborated on the key components of the NLP system used for this task, including tokenization, embedding, and language models. Furthermore, we discussed state-of-the-art transformer-based language models, describing their essential components such as word and positional embeddings, (multi-head) self-attention, transformer blocks, and the transformer language-specific head.

Chapter 4
The Dislocation Ontology

In this chapter, we discuss the development of the Dislocation Ontology (DISO), a domain ontology representing knowledge of line-like crystal defects (the dislocations). DISO is developed by first determining the scope of the ontology, reusing existing models, determining the main classes and properties. We instantiate DISO with crystals structure and dislocation simulation data. Subsequently, we concretely assess the quality of DISO by calculating metrics that evaluate the richness of the ontology (i.e. criteria-based evaluation) [134].

4.1 Ontology Development

Developing domain ontologies attracts considerable interest from several private and public sector organizations to capture their knowledge about their domain of interest in a form that machines can understand. In fact, the main goal of developing ontologies is to share conceptualizations, which represent how humans define and structure knowledge about entities, its relationships, and attributes within a specific domain.

This section describes the development process of DISO. It is an iterative approach, which means we started with an initial version and then revise and refine the evolving ontology. Fig. 4.1 illustrates the workflow of this process which is inspired by several well-known ontology development methodologies [92, 135]. This figure presents the main phases, their sub-tasks, and the roles involved in the whole process. We have frequently interviewed domain scientists as well as ontology engineers during the whole process in order to improve the final outcome. This continues through the entire development process.

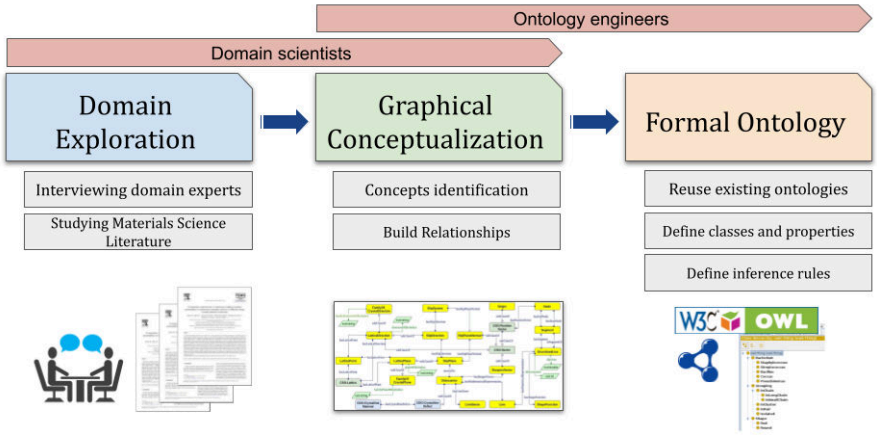


Fig. 4.1: The workflow of the dislocation ontology development, illustrating the main phases, subprocesses, and roles involved in the whole process.

4.1.1 Ontology Metadata

An important step towards improving ontology reuse is to provide a systematic and comprehensive description of ontologies [136], i.e., ontology metadata. Often, ontologies are not well documented or miss information about the ontology itself. Consequently, these ontologies are therefore suffering from several problems, including 1) being not optimally accessible to potential users, 2) hindering their reusability, and 3) identifying required ontologies for specific applications efficiently. Accordingly, several Dublin Core Metadata Initiative (DCMI) Metadata Terms[1] are added to the ontology, involving `terms:contributor`, `terms:created`, `terms:title`, and `vann:preferredNamespacePrefix`. As a next step, providing human-readable documentation for the ontology is also crucial because it is one of the main reuse problems identified in [137].

4.1.2 Reuse of Existing models

One of the initial steps in the ontology development is to reuse terms (i.e., classes or properties) from existing ontologies that describe the same domain or topic. However, it is challenging for ontology engineers to decide which of the existing ontologies is suitable to be reused. In fact, the more ontologies a model reuses, the higher the value of its semantic data, as it increases interoperability, enhances knowledge representation, and facilitates integration with external data

[1] `https://www.dublincore.org/specifications/dublin-core/dcmi-terms/`

sources. Reusing well-established ontologies also ensures consistency and enables advanced reasoning, making the ontology more useful for applications such as data linking, knowledge inference, and semantic search [134]. Therefore, reusing terms from other ontologies greatly increases semantic data value of the developed ontology.

After studying the literature, we decided to reuse several concepts from two related ontologies, namely, Crystal Structure Ontology (CSO) and Crystalline Defect Ontology (CDO). The former is an ontology developed to represent crystallographic information needed to describe the dislocation. The latter is an ontology connecting the physical materials entity to the crystal structure and several defects in crystalline materials, e.g., point defect, dislocation, and grain boundary.

Crystalline Defect Ontology (CDO)[2]. In CDO, the `emmo:Material`, which is a class from Elementary Multi-perspective Material Ontology (EMMO)[3], is reused to describe the physical entity of crystalline materials. The `cdo:CrystallineMaterial` class is subsequently a subclass of `emmo:Material`.

Crystal Structure Ontology (CSO)[4]. In CSO, several Materials Design Ontology (MDO) [39] classes are reused to describe the crystal coordinate system, motif (an arrangement of chemical species in the crystal structure) in a crystal structure, point groups, and space groups. The standard coordinate system is defined by `mdo:Basis` and `mdo:CoordinateVector` classes that CSO reuses. The motif reuses `mdo:Occupancy`, `mdo:Site`, and `mdo:Species`. Subsequently, to define the point groups and space groups of a crystal structure, `mdo:PointGroup` and `mdo:SpaceGroup` are reused. Lastly, to define the unit quantity of a property, CSO reuses several classes from the Quantities, Units, Dimensions and Data Types Ontologies (QUDT) [138]. Classes that are reused from QUDT are the `qudt:Quantity`, `qudt:QuantityKind`, and `qudt:QuantityValue`.

4.1.3 Main Classes

The core classes and relationships of DISO can be seen in Fig. 4.2. The classes in our ontology are divided into two sets: 1) imported from the ontologies introduced in Subsec. 4.1.2 (depicted in green, orange, and blue-colored boxes) and 2) newly defined classes that are not explicitly defined in the existing related ontologies (depicted in yellow-colored boxes).

Imported classes. Several classes have been imported from the CSO, involving `cso:Lattice` which represents the periodic arrangement of one or more atoms, `cso:Vector` which represents quantities that have both magnitude and direction, and imported from CDO, involving `cdo:CrystallographicDefect` which repre-

[2] `https://purls.helmholtz-metadaten.de/disos/cdo`

[3] `https://emmo-repo.github.io`

[4] `https://purls.helmholtz-metadaten.de/disos/cso`

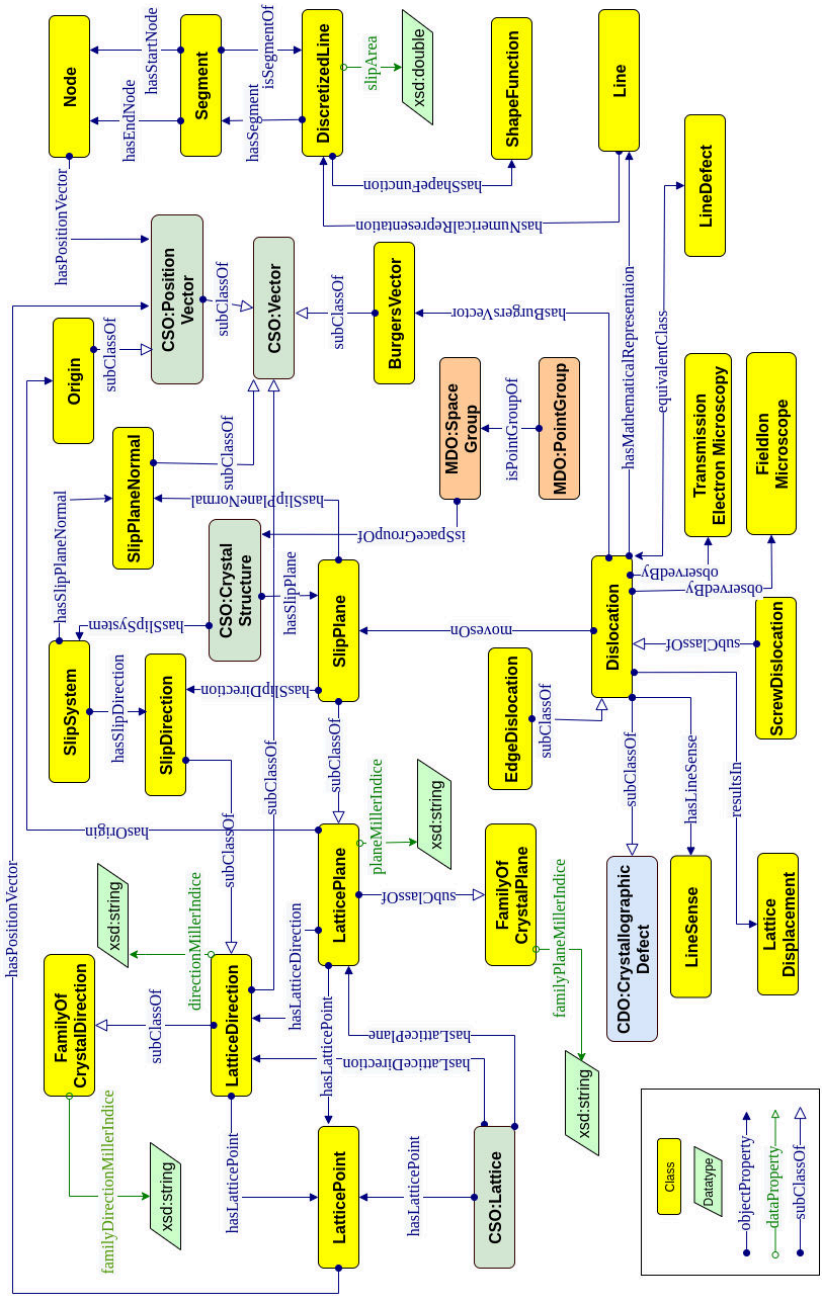


Fig. 4.2: Core concepts and interconnected relationships in the DISO ontology. Arrows with open arrow heads denote `rdfs:subClassOf` properties between classes. Regular arrows visualize *rdfs:domain* and *rdfs:range* restrictions on properties. Furthermore, colored boxes represent different ontologies, e.g., MDO, CDO, CSO, and DISO.

sents lattice irregularity/lattice defects by having one or more of its dimensions on the order of an atomic diameter.

New classes. Here, the focus is put on some chosen classes of the main classes in the crystalline materials and line defect:

1) `Dislocation`, as the entity of main interest, which models a linear or one-dimensional defect around which some of the atoms are displaced, 2) `SlipPlane`, which models the lattice plane to which the dislocation is constrained to move, 3) `SlipDirection`, which models the lattice direction where the slip occurs in the crystalline materials, 4) `LatticePlane`, which models the lattice plane where it forms an infinitely stretched plane that cuts through lattice points. 5) `LatticeDirection`, which models the direction inside the lattice that connects two lattice points, and 6) `DiscretizedLine`, which models the numerical representation of the dislocation line as a mathematical line, e.g. an oriented curve, that is discretized into a number of segments.

4.1.4 Properties

Both data and object properties in DISO are divided into two categories: newly defined properties and reused ones:

New properties. The relationships between concepts are implemented as object properties, for example, the relationship between `Dislocation` and `TransmissionElectronMicroscopy` can be represented by `observedby` property. Similarly, the relationship between `Dislocation` and `LineSense` is represented by `hasLineSense` property. Furthermore, several data properties also have been defined, such as `directionMillerIndice`, `planeMillerIndice`, and `slipArea`.

Reused properties. Property `cso:hasPositionVector` from the CSO and several data properties from DCMI Metadata Terms for adding ontology metadata (see Subsec. 4.1.1) are reused. After defining new properties and identifying reused ones, the domain and range for each property using `rdfs:domain` and `rdfs:range` are defined, respectively. For instance, the domain of the data property `diso:slipArea` is `diso:DiscretizedLine` and the range is `xsd:double`.

Restricting properties. Several DISO classes use property restrictions, e.g., value constraints. For instances, the `observedby` property which connects `Dislocation` and `TransmissionElectronMicroscopy` is restricted by a value constraint of `owl:someValuesFrom` and `hasBurgersVector` which connects `Dislocation` and `BurgersVector` is restricted by a value constraint of `owl:allValuesFrom`.

4.1.5 Reasoning

DISO's inference capability is increased through the use of several property characteristics, such as functional relations, transitivity, and the inverse property [139]. `hasMathematicalRepresentation` is a functional property because it means that a dislocation can be represented by exactly one mathematical line, i.e., it can not have any different mathematical representation than that. The transitive property can be demonstrated through the `hasRepresentation` relationship. This relationship refers to the connection between a dislocation and its representation. For instance, if a dislocation has a line representation, and this line has a discretized line representation, it can be inferred that the dislocation also has a discretized line representation. In order to enable bidirectional navigation between two classes in the ontology, inverse properties are established for each corresponding property. For instance, the `isSegmentOf` property is the inverse property of `hasSegment`. This means that if a discretized line A *has a segment* B, then B *is a segment of* A.

4.1.6 Instantiation of the Dislocation Ontology

As shown in Fig. 4.3, an excerpt of the simulation data of dislocation structure annotated by DISO is illustrated. Due to the space limit, we only added a sample of crystallographic and dislocation information. We instantiate several crystallographic-related classes: crystal structure, Bravais lattice, crystal system, and unit cell. Furthermore, we instantiate several dislocation-related classes: dislocation, slip plane, slip direction, Burgers vector, and the representation of dislocation in numerical simulation, e.g., line, segment, and node. Lastly, the instance of crystalline material connects the crystallographic information, i.e., crystal structure and dislocation instances.

4.2 Evaluation

The evaluation of the quality of an ontology can be performed by calculating metrics that evaluate the richness of the given ontology (i.e. criteria-based evaluation) [134]. In this section, we are evaluating our ontology in two directions, evaluating the success of the ontology in modeling a real-world domain and the quality of the ontology. In one direction, we carried out the following steps in order to evaluate the effectiveness of DISO: 1) defining a set of Competency Questions (CQs) based on domain expert feedback during several interviews, 2) formulating SPARQL Protocol and the RDF Query Language (SPARQL) queries corresponding to CQs, 3) running the resultant SPARQL queries against the ontology instances (cf. Subsec. 4.2.1) using a SPARQL endpoint, 4) analyzing the query results

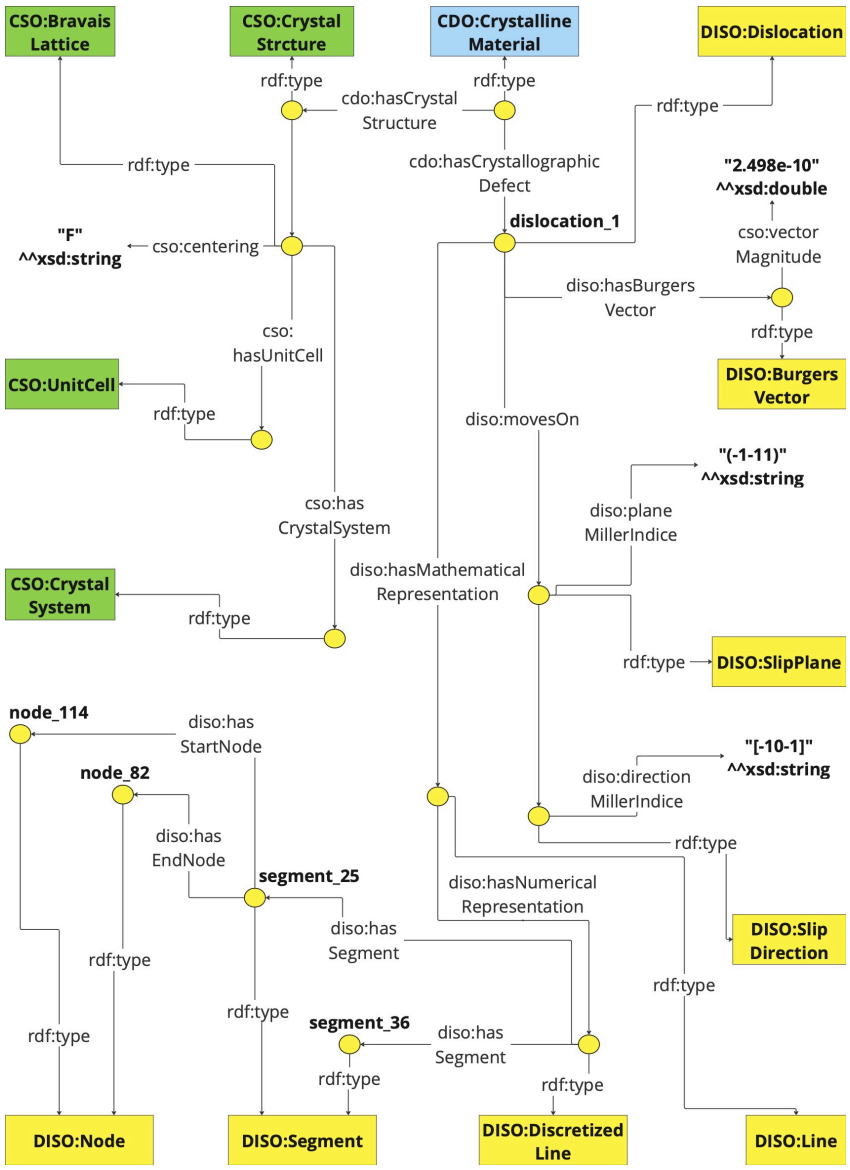


Fig. 4.3: Sample of the instances of the dislocation dynamic data. Yellow points denote individuals of classes. Each individual is defined by an arrow having *rdf:type* relationship to the respective class and individuals are connected by object properties defined in DISO. Colored boxes represent classes from different ontologies, e.g., CDO, CSO, and DISO.

by comparing them to the correct answers given by domain experts. On the other hand, we assess the ontology using the OntoQA [140] evaluation model.

4.2.1 Data

The dislocation dynamics data produced is a virtual specimen of Nickel that consists of a number of dislocation lines. To fully describe the virtual specimen, the details of the crystal structure of Nickel, such as unit cell lattice parameters, point group, and space group data are taken from the Materials Project [45] repository.

In order to carry out the aforementioned steps, we prepared a dislocation dynamics dataset [141] that can be used for illustrating how DISO is useful by answering competency questions prepared by domain experts. CQs determine what knowledge has to be entailed in the ontology as well as a way of evaluating how useful the ontology is by showing its capability to answer. Therefore, we write Python scripts [142] (using rdflib 6.0 [143] Python library) that map Discrete Dislocation Dynamics (DDD) MoDELib [144] software output with DISO classes and generate Resource Description Framework (RDF) triples ($\sim 4K$ triples), which can be found on DISO GitHub repository [145].

4.2.2 Competency Questions

Evaluating ontologies with competency questions is among the most widely used types of evaluating ontologies [40, 42, 146, 147]. A total of 20 CQs (defined in the domain exploration step (cf. Fig. 4.1)) were collected and categorized into three categories: the crystal structure, the dislocation, and the vector information. For illustration, we pick CQ1 from the CQs set (listed in Table 4.1). The corresponding SPARQL query to CQ1 is listed in Listing 4.1.

Listing 4.1: SPARQL query corresponding to CQ1.

```
PREFIX diso: <https://purls.helmholtz-metadaten.de/disos/diso#>
PREFIX cso: <https://purls.helmholtz-metadaten.de/disos/cso#>
PREFIX ex: <http://example.org/>
SELECT ?cryst_struc ?slip_sys ?slip_plane_normal_val
?slip_direction_val WHERE{
?crystal_structure a cso:CrystalStructure ;
    diso:hasSlipSystem ?slip_system .
?slip_system diso:hasSlipPlaneNormal ?slip_plane_normal ;
    diso:hasSlipDirection ?slip_direction.
?slip_plane_normal diso:directionMillerIndice ?slip_plane_normal_val.
?slip_direction diso:directionMillerIndice ?slip_direction_val.
}
```

Table 4.2 shows the result of executing the above SPARQL query on the generated RDF data in which the same crystal structure consists of several slip systems.

According to expert feedback, a slip system consists of two vectors of the slip plane normal and the slip direction that are orthogonal to each other. Formally, if

Table 4.1: Samples of competency questions.

No.	Competency Question
CQ1	What are the slip systems of a given crystal structure?
CQ2	What are the slip planes of a given crystal structure?
CQ3	In which slip planes does a dislocation move on?
CQ4	What is the Burgers vector of the dislocation?
CQ5	What is the Burgers vector magnitude of the dislocation?
CQ6	Given a slip plane of a crystal structure, what is the slip direction?
CQ7	What is the family of slip direction, given a slip direction in the crystal?
..	...
CQ20	Given a Burgers vector, what is its unit?

Table 4.2: The result of CQ1 in Listing 4.1

Crystal Structure	Slip System	Plane Normal	Slip Direction
cryst_struc_0	slip_sys_0	[-1-11]	[0-1-1]
cryst_struc_0	slip_sys_1	[-11-1]	[110]
cryst_struc_0	slip_sys_2	[-111]	[-10-1]
cryst_struc_0	slip_sys_3	[1-1-1]	[01-1]
cryst_struc_0	slip_sys_4	[111]	[1-10]
cryst_struc_0	slip_sys_5	[-1-11]	[-10-1]
cryst_struc_0	slip_sys_6	[11-1]	[-101]
cryst_struc_0	slip_sys_7	[-11-1]	[011]

we take a vector operation of the dot-product between two vectors, the result of the operation is zero. In Eq. 4.1, we sampled one query result from the first row of Table 4.2. The result of the dot product operation is zero, i.e., the configuration of the slip plane normal, $\mathbf{n}_0$, and the slip direction, $\mathbf{s}_0$, in slip_sys_0 is correct.

$$\mathbf{n}_0 \cdot \mathbf{s}_0 = \begin{pmatrix} -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -1 \\ -1 \end{pmatrix} = 0 \tag{4.1}$$

Furthermore, each slip system consists of a slip plane normal and a slip direction. The complete set of the competency questions and the corresponding SPARQL queries can also be found in the DISO GitHub repository [145].

4.2.3 OntoQA Evaluation

In this section, we assess the richness of DISO by using a criteria-based evaluation called OntoQA [140]. OntoQA evaluates the ontology using schema metrics and population/instance metrics. To evaluate the ontology design and its potential for rich knowledge representation, we use the following metrics:

- *Relationship richness (**RR**)* shows the diversity of relations and placement of relations in the ontology. Formally, it is defined by

$$RR = \frac{|P|}{|SC| + |P|} \tag{4.2}$$

  where **P** is the number of relationships in the ontology and **SC** is the number of sub-classes. The more relations the ontology owns, except *is-a* relations, the richer it is.
- *Attribute richness (**AR**)* shows the more slots/attributed that are defined, the more knowledge the ontology delivers. Formally, AR is defined by

$$AR = \frac{|AT|}{|C|} \tag{4.3}$$

  where **AT** is the number of attributes for all classes and **C** is the number of classes.
- *Inheritance richness (**IR**)* describes the distribution of information across different levels of the ontology inheritance tree. IR indicates how knowledge is grouped into different classes and sub-classes in the ontology. Formally, IR is defined by

$$IR = \frac{|SC|}{|C|} \tag{4.4}$$

As shown in Table 4.3, we compare the results of DISO with MDO [39] and CSO[5]. The former is an ontology representing the domain of solid-state physics simulation in materials science. The latter represents crystallographic information needed to describe the dislocation. While the MDO ontology has a larger *RR*, i.e., it has more diversity in relations than the other ontologies. DISO and CSO have similar values in terms of *IR*, which means they represent a wider range of knowledge compared to MDO. Concerning *AR*, DISO has a larger *AR* than MDO but is slightly smaller than CSO, enabling more knowledge per instance, which is more useful in the DDD data.

[5] `https://purls.helmholtz-metadaten.de/disos/cso`

Table 4.3: OntoQA Evaluation of DISO

Ontology	C	SC	AT	P	RR	AR	IR
MDO	37	49	14	32	0.40	0.38	1.33
CSO	30	49	19	24	0.33	0.63	1.63
DISO	38	62	23	33	**0.35**	**0.61**	**1.63**

4.3 Summary and Conclusion

We have presented the dislocation ontology, which is an ontology defining the linear defect concepts and relations in crystalline materials. In order to develop DISO, we explored the respective domain and determined the ontology's scope. In order to determine the scope, we conducted a requirement analysis. Ultimately, one of the requirement analysis results is CQs, which are questions that ontology should be able to answer.

The subsequent step after determining CQs is to conceptualize the dislocation domain. We developed DISO using a top-down approach. We started by defining general concepts and then built specialized classes based on these general concepts. The next step involved formalizing a diagram into an ontology with the help of Web Ontology Language (OWL) vocabulary and serializing it into a machine understandable format called RDF. Furthermore, DISO is published through a persistent identifier, following World Wide Web Consortium (W3C) best practices for publishing Linked Data. Moreover, we followed two evaluation strategies to demonstrate the ontology's usefulness and quality: Evaluating the success of modeling a real-world domain and assessing the quality of the ontology using OntoQA.

In the next chapter, we discuss the extension and application of DISO. DISO is used to transform the unstructured data of dislocation simulation into a knowledge graph consisting of semantic network of dislocation simulation data. The generation of a knowledge graph enables semantic queries, supports intelligent tasks, and increases the interoperability of dislocation simulation data.

Chapter 5
The Dislocation Knowledge Graph

In the chapter, we discuss the extension and application of Dislocation Ontology (DISO) by creating a knowledge graph in an Materials Science and Engineering (MSE)-related domain that deals with data governing details of linear crystallographic defects, i.e., dislocation data. The extension of DISO is carried out by aligning with several materials science ontologies (e.g., Elementary Multiperspective Material Ontology (EMMO) and Materials Design Ontology (MDO)), Quantities, Units, Dimensions and Data Types Ontologies (QUDT), and PROV Ontology (PROV-O). By doing the alignment, DISO is able to annotate the Discrete Dislocation Dynamics (DDD) simulation data consisting of dislocation structure, provenance, simulation parameters, and crystal structure data. Subsequently, the unstructured dislocation data is transformed into linked data with dereferenceable Internationalized Resource Identifier (IRI)s using persistent URLs, adhering to World Wide Web Consortium (W3C) standards and best practices. It enables not only the annotation of dislocation data by an ontology but also its integration into other MSE-related fields.

5.1 Ontology Alignment

Ontology alignment is the process of identifying relations between entities among different ontologies in order to establish connections between them [148]. These entities include classes, properties, and individuals. For successful ontology alignment, it is crucial to identify similarities between source and target ontologies. The analysis entails examining concepts that overlap but may have different names (i.e. synonyms) or types in the ontologies [149].

This section will cover the extension of DISO, which involves aligning two ontologies, namely EMMO and MDO. This alignment plays a crucial role in allowing DISO to annotate the DDD data and transform it into linked data while also facilitating knowledge graph generation.

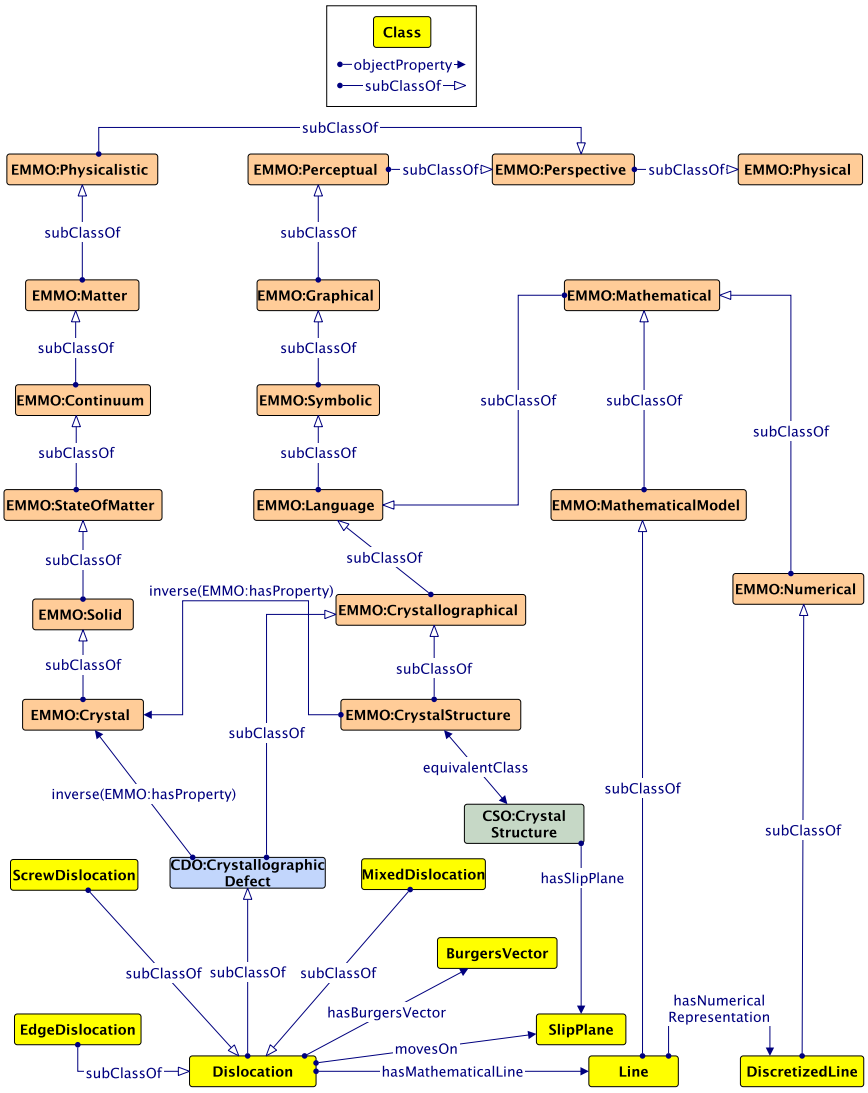


Fig. 5.1: DISO alignment with EMMO. The starting point to align the DISO with the EMMO by importing the domain ontology crystallography developed by EMMO.

5.1.1 Alignment with EMMO

EMMO is an ontology establishing semantic standards that can be implemented at the highest level of abstraction. This makes it possible for all potential domain ontologies, especially in the MSE field, to be integrated and to work together seamlessly. Currently, EMMO consists of two modules: a top-level and a mid-level module. The former includes the fundamental axioms that constitute the philosophical foundation of the EMMO, while the latter consists of a set of perspectives to develop more specialized domain ontologies. These two ontologies serve as the basis for building further domain and applications ontologies, e.g., the application of EMMO in the domain of mechanical testing [33].

The starting point to align the DISO with EMMO is by aligning with the Crystalline Defect Ontology (CDO)[1], a domain ontology based on EMMO and the Crystallographic Information File (CIF) core dictionary[2]. As shown in Fig. 5.1, `cdo:CrystallographicDefect` subsumes `Dislocation`, while also being a subclass of `emmo:Crystallographical` class. Similarly, `emmo:CrystalStructure`, an equivalent class to `cso:CrystalStructure`, is also a subclass of `emmo:Crystallographical`. Overall, `emmo:Crystallographical` is a class that ideally represents the physical concepts associated with crystalline materials

As we mentioned in Ch. 2, on the mesoscale, a dislocation is represented by a mathematical line, which can be further idealized as a pixel or discretized line depending on the application (e.g., microscopy or simulation). To align the dislocation mathematical line concept with an EMMO class, `emmo:MathematicalModel` subsumes `Line` and the discretized representation of the mathematical dislocation line, `DiscretizedLine`, is subsumed by `emmo:Numerical`.

5.1.2 Alignment with MDO

MDO is a domain ontology that defines concepts and relations to cover the knowledge of materials design, especially in the ab-initio calculation. MDO consists of several modules, a *Core*, the *Provenance* module, and two domain-specific modules: *Structure* and *Calculation*.

To align the DISO with the MDO, we reused several classes in the MDO Core module. The MDO Core module describes the structure or the virtual specimen of interest via `mdo:Structure` class. As shown in Fig. 5.2, we defined a `DislocationStructure` class as a subclass of `mdo:Structure`. This class describes a dislocation (micro)structure, which is a virtual structure used by a DDD simulation to study the mechanical properties of a crystalline material. Furthermore, `DislocationStructure` as an idealized representation relates to a physical concept called `cdo:CrystallineMaterial`.

---

[1] `https://github.com/emmo-repo/domain-crystallography`

[2] `https://www.iucr.org/resources/cif/dictionaries/cif_core`

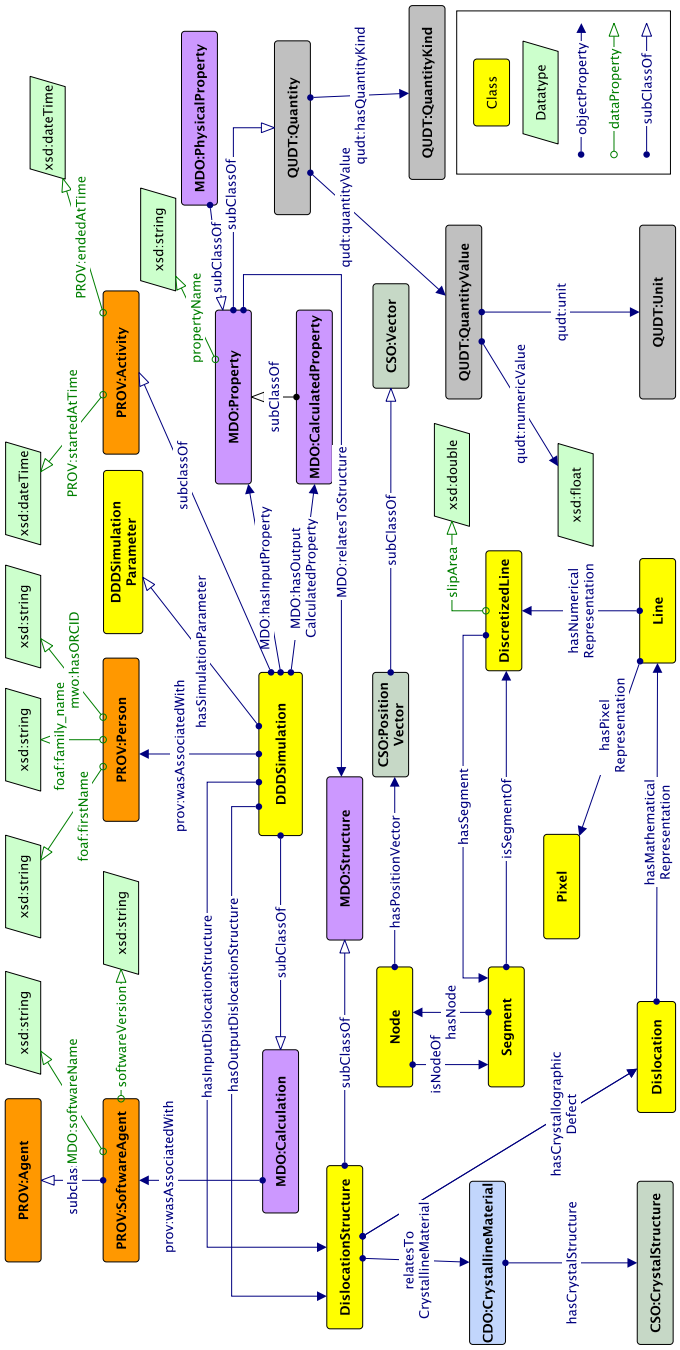


Fig. 5.2: DISO alignment with the MDO Core and Provenance module.

In the MDO Core module, an instance of the `mdo:Structure` class is used as a virtual specimen input or output for a simulation. Here, the simulation concept is represented as the `mdo:Calculation` class. We subsumed the `mdo:Calculation` class to define the `DDDSimulation` class, which is a class to describe the DDD simulation. Thus, the `DDDSimulation` can have `DislocationStructure` as an input or an output. Moreover, the `DDDSimulation` has an input and output relationship with `mdo:Property` to run a calculation. In addition, the `DDDSimulation` is related to the `DDDSimulationParameter`, a simulation parameter concept configuring the DDD simulation, e.g., the activation parameter for cross-slip, junction formation, and external load.

To preserve the provenance information of a DDD simulation, we reused several classes from the MDO Provenance module and the PROV ontology [65]. Running a DDD simulation requires specific software to solve materials science problems. It is quite helpful to store information about the software used and its version, as this can help scientists reuse data through post-processing methods specific to DDD software. In this regard, `DDDSimulation` has a relationship with `prov:SoftwareAgent`, which has two data properties: `softwareVersion` and `mdo:softwareName`. Furthermore, to preserve the provenance information related to when a DDD simulation starts and ends, `prov:Activity` subsumed the `DDDSimulation` and inherited two data properties: `prov:startedAtTime` and `prov:endedAtTime`. Apart from that, we reused `PROV:Person` to annotate the person running or responsible for the simulation. It has three data properties to define a person: `foaf:firstName`, `foaf:family_name`, and `mwo:hasORCID`. The latter is a data property that we reused from the MatWerk Ontology (MWO)[3].

To summarize, core concepts and interconnected relationships in DISO after the alignment can be seen in Fig. 5.3. The advantages of ontology alignment for DISO are promoting knowledge transfer from other ontologies when describing the DDD simulation. Furthermore, ontology alignment fosters interoperability between ontologies in MSE-related domains. The objective is to assist in building a knowledge graph for the dislocation domain.

5.2 Knowledge Graph Development

In the field of materials science, researchers use a numerical method called DDD simulation to analyze the behaviour of dislocations within crystalline materials. This technique helps identify the specific characteristics of each dislocation, as well as their interaction, arrangement, and collective behaviour within the material. The simulation observes the motion and interaction of many dislocations which ultimately creates the relationship between the microstructure, loading conditions, and the mechanical properties of a crystalline material. For simulations of dislocations, there are various software options available such as MoDELib [144],

[3] `http://purls.helmholtz-metadaten.de/mwo/`

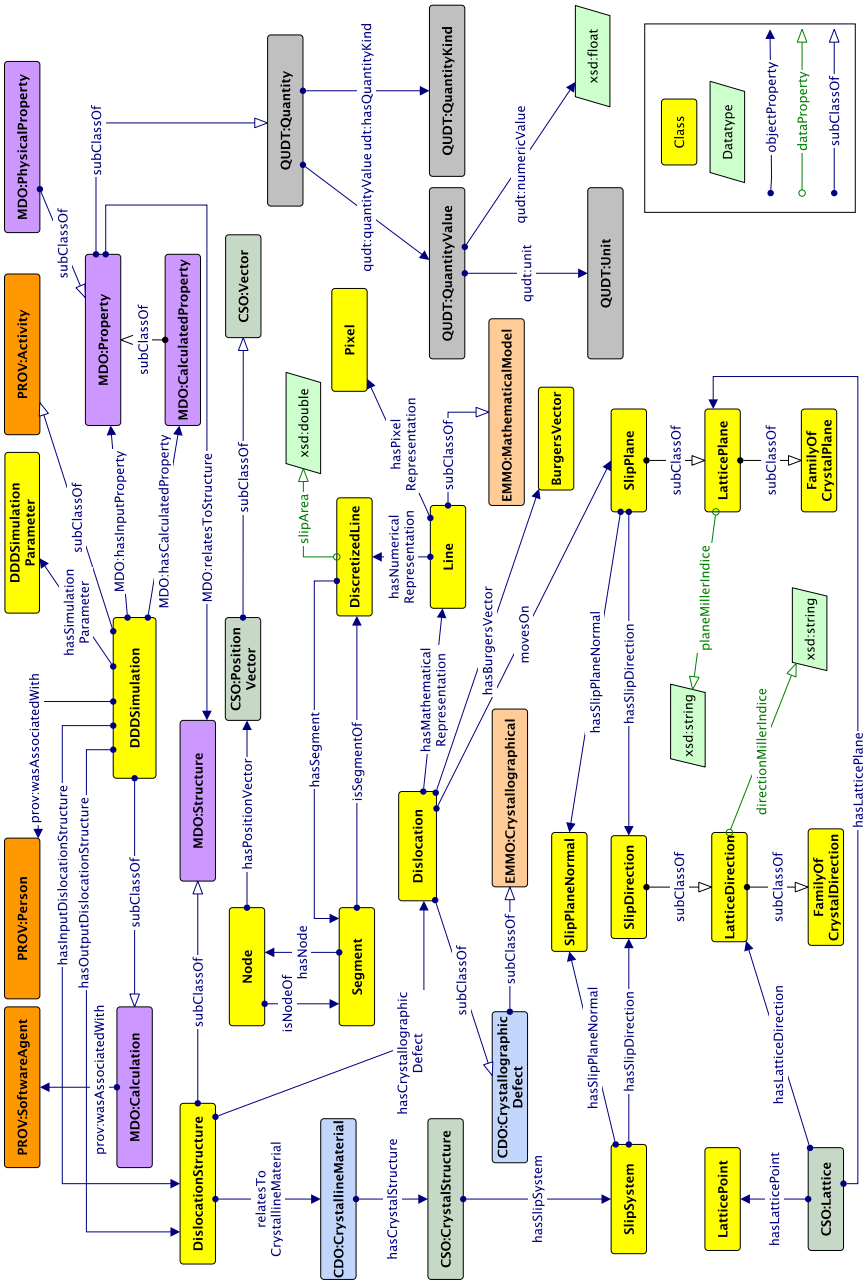


Fig. 5.3: The core concepts in DISO ontology after the alignment with MDO and EMMO. Arrows with open arrowheads denote `rdfs:subClassOf` properties between classes, while regular arrows represent the relationships between them. Classes that belong to the same ontology share the same color.

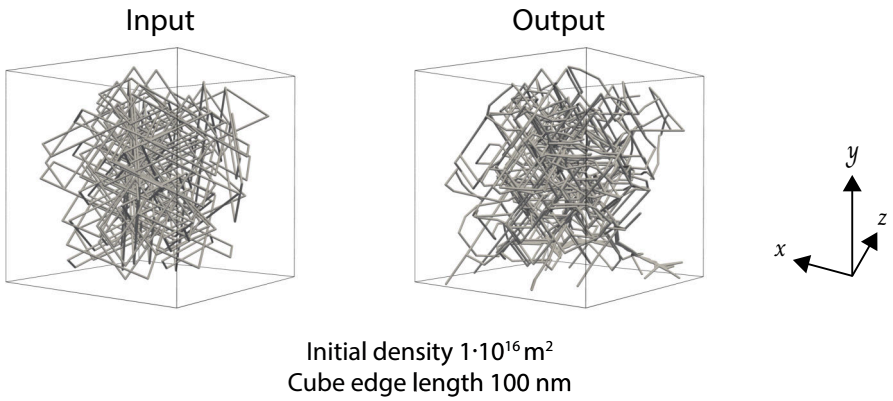


Fig. 5.4: Sample of dislocation microstructures used as input for simulation as well as yielded by the simulation as output.

ParaDiS [150], and microMegas [151]. Every software has a distinct collection of metadata that organizes the inputs and outputs of the simulation.

DISO was utilized in this specific scenario to accurately annotate the information collected from various DDD simulations. The ultimate goal is to generate a comprehensive Dislocation Knowledge Graph (DisLocKG) using this data. The DDD data used in this work was generated through the MoDELib software and took different initial dislocation densities and specimen sizes into account. The cube-shaped Copper specimen, with an edge length of either 50 or 100 nanometers, was randomly filled with dipolar edge loops on all slip systems until the initial density of either $1 \cdot 10^{16}$ m^{-2} or $5 \cdot 10^{16}$ m^{-2} was reached. A sample of the generated cube-shaped Copper specimen can be seen on the left panel of the Fig. 5.4. During the simulation, the dislocation microstructure was allowed to relax without any external load, meaning that internal stress and image forces solely influenced the dislocation evolution. The simulation resulted in the relaxed dislocation microstructure shown on the right-hand side of Fig. 5.4. An important aspect for a materials scientist is that some simulations do not have cross-slip or junction formation. This has significant implications when it comes to analyzing simulation results. E.g., Demirci et al. [152] investigated the influence of cross-slip on the evolution of dislocation structures and therefore could benefit from this information. Such relaxation calculations are also important for creating a realistic microstructure. For example, Motz et al. [153] investigated how the relaxed dislocation microstructure influences the plasticity in subsequent tensile test simulations. Furthermore, several authors [23, 29, 154, 155] conducted machine learning and data mining studies utilizing the dislocation relaxed microstructure to classify the structure and express the strain energy density of a dislocation microstructure, respectively. This explains why there is a strong need for a detailed and formal representation of such simulations – here, contained in the class of `DDDSimulation`.

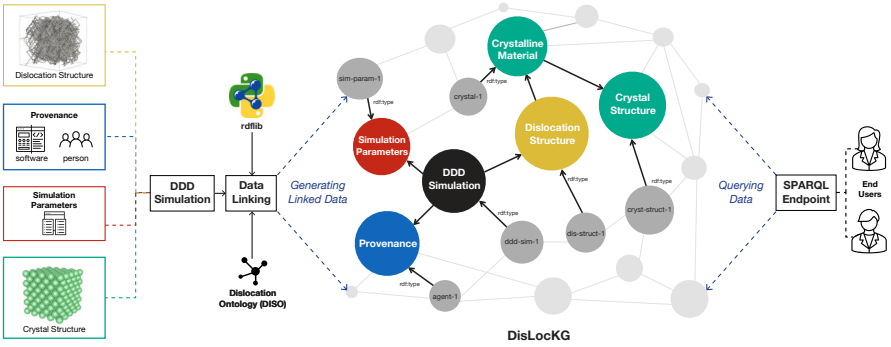


Fig. 5.5: DDD simulation data as linked data. Colored rectangles on the left depict data types in the DDD simulation: dislocation structure, provenance, simulation parameters, and crystal structure data. The data subsequently is linked using the DISO as a reference ontology. The rdflib Python module supports data linking and generates DisLocKG. Via the SPARQL Protocol and the RDF Query Language (SPARQL) Endpoint, end users can query the data to retrieve the information in the DisLocKG.

For our example, we have collected a total of 25 data points – where each data point is one DDD simulation consisting of initial and final microstructure. Each of those was annotated with DISO. Any data point gives information about the simulation details, such as parameters used for a simulation, initial dislocation microstructure used as input, and the resulting dislocation microstructure produced by the simulation. Additionally, each dislocation microstructure includes information about the crystal structure, Bravais lattice, dislocation, slip plane, Burgers vector, and numerical representation of dislocation. The results of the simulations are stored and parsed in the HDF5[4] format. Subsequently, we utilize our in-house Python scripts (using the rdflib 6.0 [143] Python library) to create a knowledge graph called DisLocKG from this data using DISO as a reference ontology (cf. Fig. 5.5). DisLocKG is a semantic network that holds information about dislocations in crystalline materials. Note that, DisLocKG also stores the provenance information related to the data, particularly the creator data, software, and software version used to generate the data. In total, we have generated a number of $\sim$ 2.2M triples that are stored as Resource Description Framework (RDF) files which are available via its persistent identifier[5].

Publishing DDD data as linked data has several benefits [147], including 1) establishing links between dislocation-related datasets, enabling machines to understand and discover new information, 2) supporting semantic querying via the SPARQL query language, 3) supporting data enrichment, where machines can infer

[4] https://www.hdfgroup.org/solutions/hdf5/

[5] https://purls.helmholtz-metadaten.de/dislockg

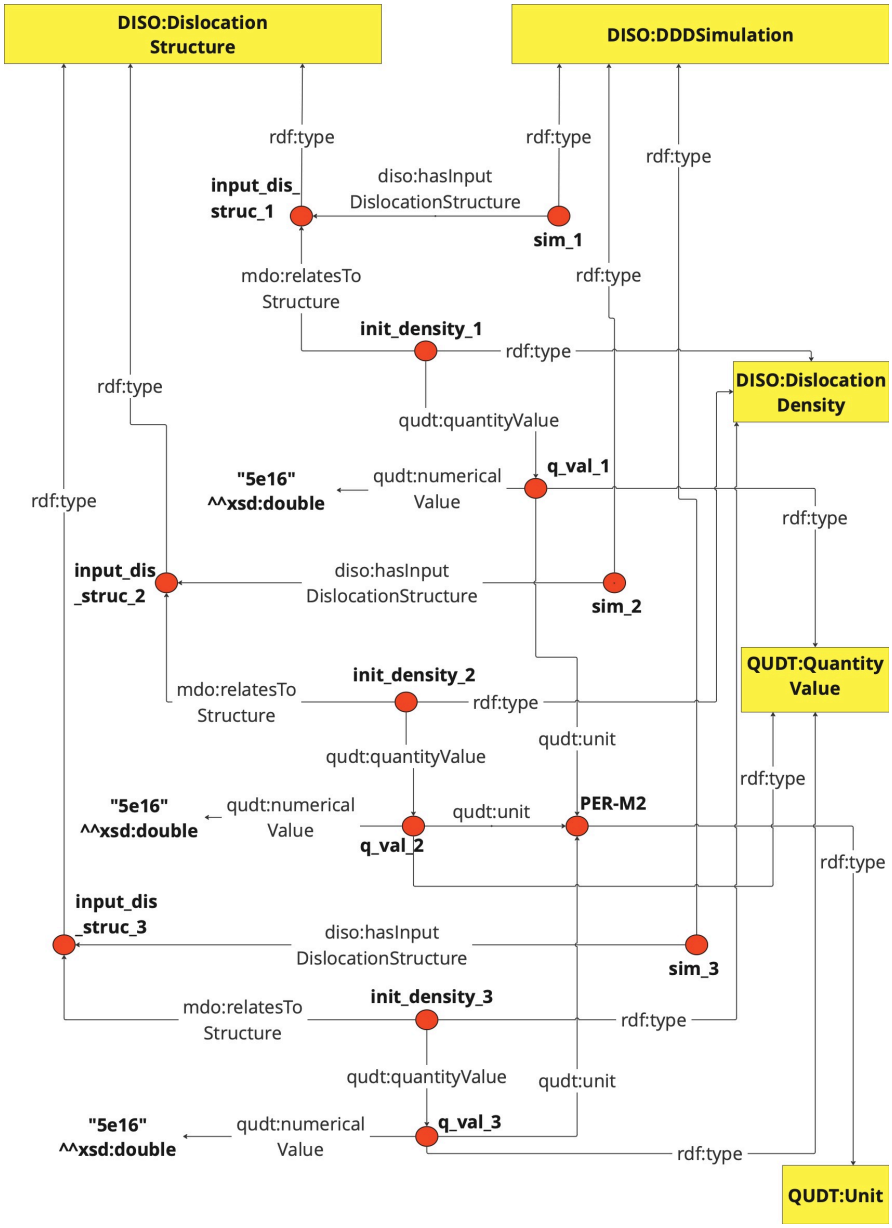


Fig. 5.6: A visual representation of CQ3 result. Colored boxes represent classes and the red dot represents an individual belonging to that class. Each individual is defined by a directed arrow having the `rdf:type` relationship to the respective class and connected to other individuals by object properties.

implicit knowledge that does not exist, and 4) promoting semantic validation of the data, ensuring consistency and accuracy.

We have listed some competency questions in Table 5.1 to give an idea of the vast information available in DisLocKG. For instance, CQ1 can retrieve the history and origin information of DDD data generated by the MoDELib software, and CQ2 and CQ3 can retrieve information on the specimen geometry and the initial dislocation of each dislocation simulation. These CQs are important if one wants to query a dislocation simulation to be reused for the processing step if they need a specific density of a dislocation structure and information concerning the geometry. CQ4 retrieves the input parameters to run the simulation, while CQ5 queries all dislocation structures generated by the relaxation calculation. The SPARQL query corresponding to CQ3 is shown in Listing 5.1, and the complete set of the competency questions and the corresponding SPARQL queries can also be found in the DISO GitHub repository. Fig. 5.6 visualizes the results of CQ3, which contains three individuals (shown as the red markers) of the DDD simulation class. Each of the DDD simulation individuals has a relationship with dislocation structure individuals. Moreover, the dislocation density data relates to the dislocation structure individual.

Table 5.1: A sample of competency questions for DisLocKG.

No.	Question
CQ1	Provide detailed information on the dislocation structures simulated using the MODELIB software, including the software version and creator associated with these simulations.
CQ2	Which dislocation structures possess a specimen shape resembling a cube with an edge length greater than 30 nanometers?
CQ3	List all DDD simulations that have an initial density of dislocation $= 5e16$ m^{-2}
CQ4	List all DDD simulations that do not activate the cross slip formation and junction formation
CQ5	What are dislocation structures generated by the relaxation calculation? List also the initial density of a dislocation structure used for a relaxation calculation, simulation parameters: cross-slip activation, junction formation activation, and external load activation.

5.3 Evaluation

Employing predefined metrics that evaluate an ontology's richness through criteria-based assessment is one way of evaluating its quality [134]. In this section, we

Listing 5.1: SPARQL query corresponding to CQ3.

```
PREFIX diso:<https://purls.helmholtz-metadaten.de/disos/diso#>
PREFIX qudt:<http://qudt.org/schema/qudt/>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX mdo_core:<https://w3id.org/mdo/core/>
SELECT ?ddd_simulation ?initial_density ?unit
WHERE{
?ddd_simulation       diso:hasInputDislocationStructure
                                  ?input_dislocstructure.

?initial_density_qu   mdo_core:relatesToStructure
                                  ?input_dislocstructure;
                      qudt:quantityValue
                                  ?density_qv.

?density_qv           qudt:numericalValue              ?initial_density;
                      qudt:unit                        ?unit.
FILTER(?initial_density = "5e16"^^xsd:double)
}
```

evaluate the adapted version of DISO using the OntoQA [140] evaluation model, as described in Subsec. 4.2.3.

Table 5.2: Evaluation of DISO compared to DISO v1.0, MDO and Crystal Structure Ontology (CSO) using the OntoQA model. C is the number of classes, SC is the number of sub-classes, AT is the number of attributes, and P denotes the number of relationships.

Ontology	C	SC	AT	P	RR	AR	IR
MDO	37	49	32	32	0.40	0.86	1.32
CSO	30	49	19	25	0.34	0.63	1.63
DISO v1.0	33	62	12	33	0.35	0.32	1.63
DISO v1.1	**70**	**116**	**47**	**80**	**0.41**	**0.67**	**1.66**

In Table 5.2, we compare the evaluation outcomes of DISO with MDO [39], CSO[6] and the previous version of DISO. DISO has the most significant value of *RR*, which implies that it has a greater relation diversity. Moreover, DISO has the highest *IR* value, representing a more comprehensive knowledge range than MDO, CSO, and the previous version of DISO. The *AR* value of DISO is lower than that of MDO and higher than CSO and the previous version of DISO. To conclude, DISO possesses the most extensive knowledge representation and diversity in terms of relationships, achieving the highest IR and RR, respectively. Moreover, the adapted version of DISO surpasses its predecessor in all evaluation metrics.

[6] https://purls.helmholtz-metadaten.de/disos/cso

5.4 Summary and Conclusion

We have presented how semantic web technologies can be leveraged to transform unstructured data, e.g., DDD data, into well-organized and structured data. In order to carry out the transformation of DDD data efficiently, we extended DISO. The extension was carried out by aligning DISO with commonly used materials science ontologies (e.g., EMMO and MDO core), PROV-O, and QUDT to be able to model simulation data. Furthermore, we showed a real-world use case that utilized the DISO to construct a semantic network of DDD data (i.e., linked data) called DisLocKG, where individual entities are connected, enabling semantic query and supporting intelligent tasks. In order to query DisLocKG, the graph was made accessible to the public, and instructions for setting up a SPARQL endpoint were provided in its GitHub repository. The evaluation results showed that the modified version of DISO is the most comprehensive and diverse knowledge representation compared to other state-of-the-art ontologies.

Chapter 6
Provenance Documentation in Materials Science and Engineering

In this chapter, we discuss the development of the PRovenance Information for MAterials science (PRIMA). PRIMA utilizes a NeOn Methodology (cf. Subsec. 3.2.3) fostering the collaboration in the development. Furthermore, PRIMA extends the PROV Ontology (PROV-O) and aligns with PMD Core Ontology (PMDco) as it reuses and extends several terms or classes from PMDco. The motivation for the alignment is to ensure interoperability with other materials science application ontologies. Moreover, we exhibit two use cases integrating PRIMA into the their system: 1) The Scanning Tunneling Microscopy (STM) experiment and 2) The integration of PRIMA in the Herbie Electronic Lab Notebook (ELN).

6.1 Ontology Development

In the following, we describe PRIMA and its development process following the NeOn methodology [93] for ontological engineering. In the NeOn methodology, there are nine scenarios in which an ontology can be developed. These scenarios can be combined in flexible ways, ensuring multiple needs of ontology developers and engineers. Additionally, the NeOn methodology emphasizes collaborative work as well as reuse and re-engineering of knowledge resources.

In the development of PRIMA, we applied Scenarios 1, 2, 3 and 8: *From specification to implementation*, *Reusing and re-engineering non-ontological resources*, *Reusing ontological resources*, and *Restructuring ontological resources* (e.g., the ontology modularization activity), respectively. Fig. 6.1 illustrates the ontology development workflow combining the adopted Scenarios and the role of ontology engineers and domain scientists.

We began the development with the ontology specification activity; see Scenario 1 in Fig. 6.1. In this activity, we specified the requirements to be fulfilled by

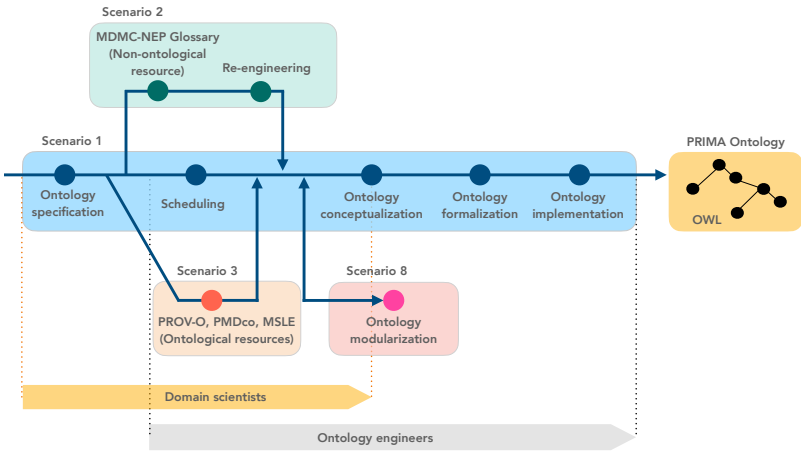


Fig. 6.1: Scenarios 1, 2, 3 and 8 of the NeOn methodology, applied for the development of PRIMA. The dots in each Scenario depicts the activities taking place.

Table 6.1: Sample of competency questions.

No.	Competency Question
1	Which project(s) is the researcher member of?
2	Which study(ies) has the researcher performed?
3	Which data acquisition(s) or data analysis lifecycle(s) has the researcher performed?
4	Which research data is attributed to (e.g., created by) the researcher?
5	Which data analysis lifecycle(s) was used in a study and which data were used?
6	Which studies were done by a project and list all data acquisition and data analysis lifecycle which were done in those studies?
7	Which result(s) were obtained from the data analysis lifecycle?
8	Which data analysis(es), data processing(s) and data interpretation(s) are part of the data analysis lifecycle?
9	Which data were used and produced in the data analysis/processing/interpretation?
...	...
26	What process sequence taken for doing a fabrication/measurement/sample preparation?
27	Which sample component(s) is the sample made of?

PRIMA by formulating 27 competency questions (CQs), sampled in Table 6.1. The complete set of CQs can be found in the PRIMA GitHub repository[1].

[1] `https://purls.helmholtz-metadaten.de/prima/dev`

The next step was to identify knowledge resource candidates that meet the defined requirements, including both non-ontological and ontological resources. We applied Scenario 2, utilizing the Model and Data Driven Materials Characterization and NFFA-Europe Pilot (MDMC-NEP) Glossary of Terms [85] as a non-ontological resource. This glossary was compiled by the Metadata Working Group of the Joint Lab Model and Data Driven Materials Characterization (MDMC) in collaboration with NFFA-Europe Pilot (NEP)[2].

The MDMC-NEP Glossary comprises 45 terms describing experimental and computational workflows in a materials science and engineering (MSE) study at a high level. Additionally, it extends to related data management and data infrastructure. This range is intended to reflect the lifecycle of both physical entities and data collected in materials science research. The glossary terms were re-engineered using the *genus-differentia method* [156] and formally represented in OWL, resulting in a prototype ontology fitting Scenario 1 (see Fig. 6.1).

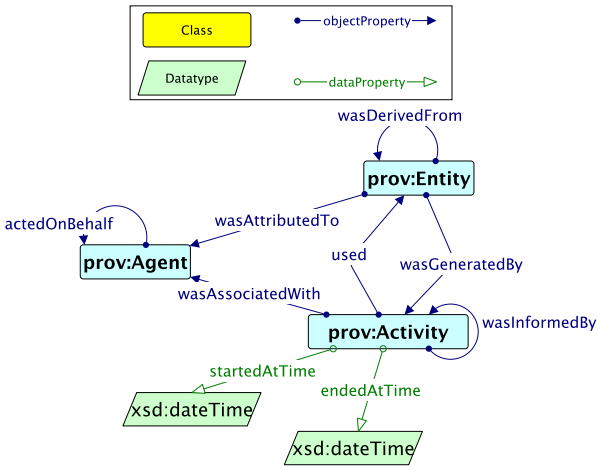


Fig. 6.2: PROV-O depicts three base classes: `Agent`, `Entity`, and `Activity`, and a set of predefined relationships between these classes. The figure is adapted from [65]

Ontology reuse facilitates knowledge transfer and enables interoperability between ontologies [157]. Thus, in the Scenario 3 we reused several terms originating from PROV-O [65], PMDco, and the Materials Science Laboratory and Equipment Ontology (MSLE) [158]. PROV-O is reused as the basis for capturing provenance information, because it is general and domain independent. The three base classes of the PROV-O core[3] and a set of predefined relationships between them is de-

[2] `https://jl-mdmc-helmholtz.de/mdmc-activities/metadata-working-group/`

[3] `https://www.w3.org/TR/2013/REC-prov-o-20130430/`

picted in Fig. 6.2. The `prov:Entity` is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary. The `prov:Activity` is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities. The `prov:Agent` is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.

We reused several PMDco terms, including `pmd:Process`, `pmd:Object`, and `pmd:ValueObject`. These terms are related to a set of actions occurred in PRIMA representation of experimental workflow, computational workflow, and data analysis lifecycle. Fig. 6.3 illustrates the PMDco process and the Quantities and Units pattern. Additionally, we reused terms from MSLE, primarily related to the equipment and instruments used during an experimental workflow, such as `msle:Equipment`.

We identified two patterns in the Modular Ontology Design Library (MODL) [159] repository[4] that can be reused or re-engineered for PRIMA: *Agent Pattern*, defining an Agent's role, e.g., data scientist, data engineer, or experimenter, and *Quantities and Units*, which in turn uses terms from QUDT (Quantities, Units, Dimensions, and Data Types Ontologies) [138], e.g., `qudt:Quantity`, `qudt:QuantityValue`, `qudt:Unit`, and `qudt:QuantityKind`.

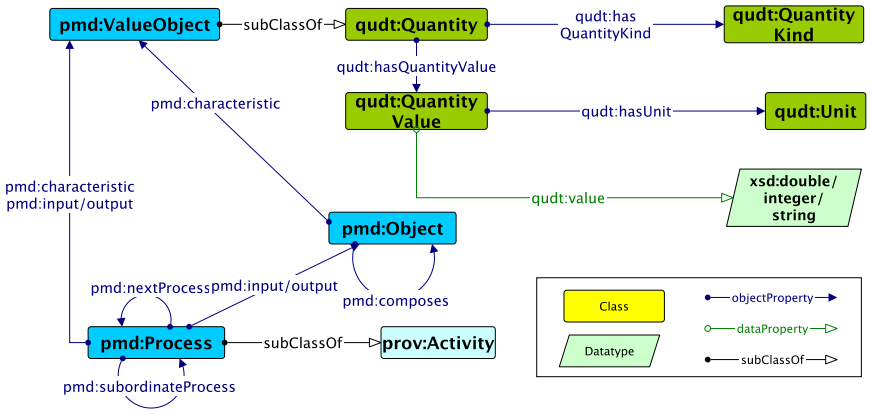


Fig. 6.3: PRIMA reuses the PMDco terms and the Quantities and Units pattern to describe the processes and the measured quantities. Additionally, classes that belong to the same ontology share the same color.

In Scenario 8 (see Fig. 6.1), we performed ontology modularization, explicitly focusing on the ontology developed in Scenario 2, to enable knowledge reuse by customizing modules for specific use cases. Four ontology modules have been es-

[4] `https://github.com/Data-Semantics-Laboratory/modular-ontology-design-library`

tablished on PRIMA: *Core*, *Data Analysis Lifecycle*, *Dataset*, and *Experiment*. The remaining modules, such as the computational module, are still under development. Since they lie outside the scope of this work they will be presented in separate future publications.

We performed the remaining activities in Scenario 1 (cf. The blue box in Fig. 6.1): Ontology conceptualization, formalization, and implementation (see Fig. 6.1). Ontologies from Scenarios 2, 3, and 8 were organized and structured into meaningful models. We then extended PROV-O and PMDco by subsuming PRIMA modules into PROV-O base classes and PMDco reused terms. The conceptualization result was finally implemented and encoded in OWL (see Table 6.2).

6.1.1 Ontology Metadata

An essential step to increase the reuse capability of an ontology is to provide systematic and comprehensive descriptions [160], i.e., metadata. For this reason, DCMI Metadata Terms[5], such as `terms:contributor`, `terms:created`, `terms:title`, and `vann:preferredNamespacePrefix` are added to PRIMA. Furthermore, we provided human-readable documentation for each ontology module, available (in several RDF serializations) via a persistent identifier provided by PIDA (Persistent IDentifier for semantic Artifacts)[6] and reported in Table 6.2. Furthermore, PRIMA is collaboratively developed and maintained on GitHub[7].

Table 6.2: PRIMA modules and persistent URLs

Module	Persistent URL
Core	https://purls.helmholtz-metadaten.de/prima/core
Dataset	https://purls.helmholtz-metadaten.de/prima/dataset
Data analysis lifecycle	https://purls.helmholtz-metadaten.de/prima/dal
Experiment	https://purls.helmholtz-metadaten.de/prima/experiment
Complete	https://purls.helmholtz-metadaten.de/prima/complete

6.1.2 Ontology Description

The core module. This module, illustrated in Fig. 6.4, consists of general classes and properties that can be reused in other modules. It was developed to provide

[5] `https://www.dublincore.org/specifications/dublin-core/dcmi-terms/`

[6] `https://purls.helmholtz-metadaten.de`

[7] `https://purls.helmholtz-metadaten.de/prima/dev`

provenance information, at a general level, about research studies in the materials science domain, the involved processes, equipment, techniques and systems.

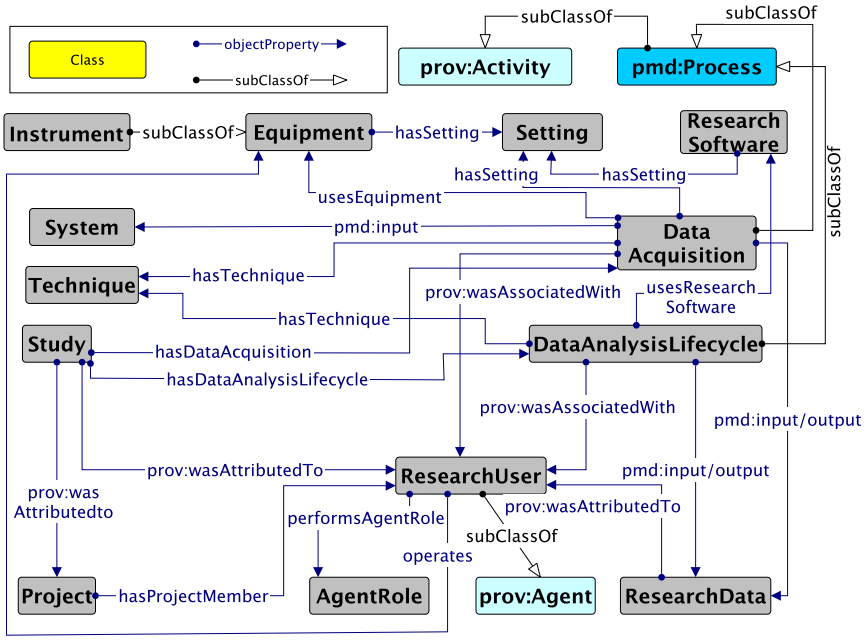


Fig. 6.4: The core module of the PRIMA ontology. This module consists of general classes and properties, which are extended in other PRIMA modules.

The experiment module. The class and properties related to an experimental workflow are described in this module, illustrated in Fig. 6.5. In this module, the `Study` class from the core module is reused and extended. It describes that in a study, a set of processes called `Fabrication`, `SamplePreparation`, and `Measurement` occurs. Furthermore, these three classes are aligned with the `pmd:Process` class. As illustrated in Fig. 6.6, we can sequentially define a set of processes, e.g., the `pmd:nextProcess` of `SamplePreparation` is `Measurement`, which has `Sample` as input and `dataset:RawData` as output.

The dataset module. This module, shown in Fig. 6.7, extends the `core:ResearchData` by defining several specialized classes, e.g., `ReferenceData`, `AnalyzedData`, `ProcessedData`, and `RawData`. Moreover, it defines several places where the data can be stored, e.g., the data can be stored in a `DataCollaborationPlatform` (e.g., Nubes, Nextcloud, Sciebo) and/or in a `DataRepository` (e.g., Zenodo). The `Dataset` may consist of `core:ResearchData` or other `Dataset` and is described by `Metadata` which can be stored in a `MetadataRepository`.

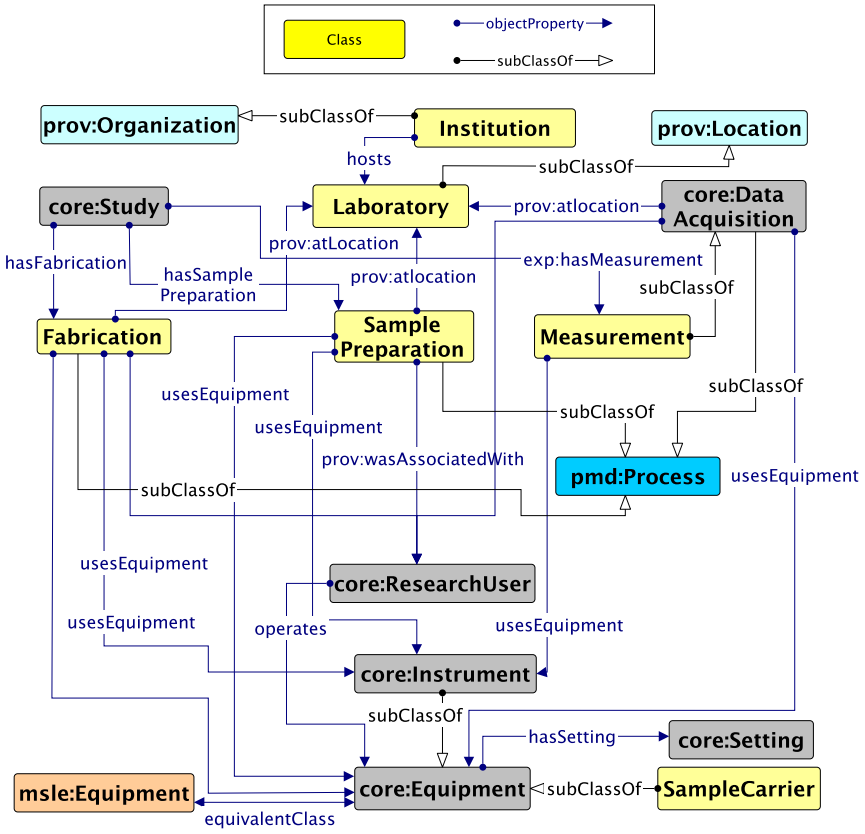


Fig. 6.5: The experiment module was developed to provide general provenance information related to the materials experimental workflow. Classes that belong to the same ontology share the same color. E.g., colors refer to different ontologies or PRIMA-Modules: gray: PRIMA-core; yellow: PRIMA-experiment; blue: PROV-O, dark blue: PMDCo, and orange: Materials Science Laboratory and Equipment Ontology (MSLE)

The data analysis lifecycle module. This module, illustrated in Fig. 6.8, contains three main classes representing the set of processes which may be involved in the data analysis lifecycle: `DataAnalysis`, `DataProcessing`, and `DataInterpretation`. These classes are subsumed under `pmd:Process` and relate to the `core:ResearchUser` and `ResearchSoftware` as the agents of the processes. As an example, Fig. 6.9 illustrates the data analysis process, with input `core:ResearchData`, particularly `dataset:ProcessedData`, and output `dataset:AnalyzedData`, performed using `ResearchSoftware`.

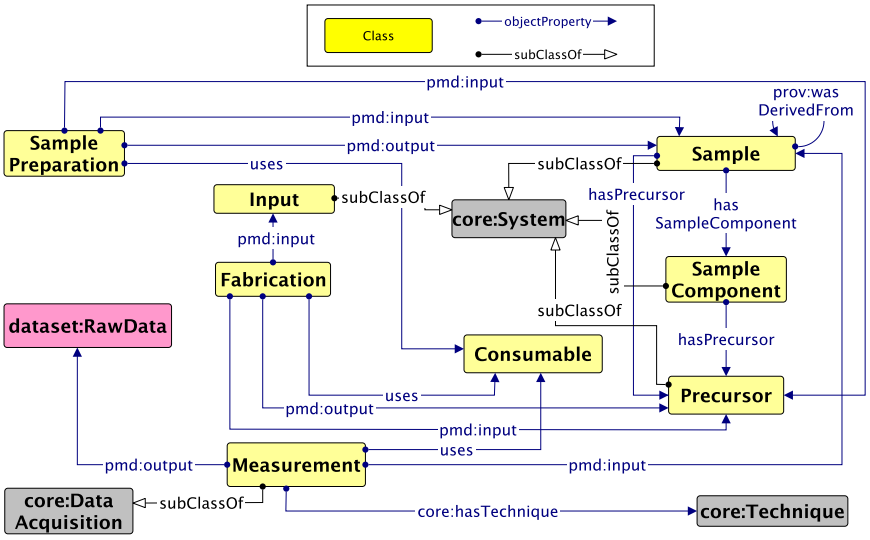


Fig. 6.6: Due to the alignment with PMDco, the experiment module can define the processes taken in a materials experimental workflow. Classes that belong to the same ontology share the same color.

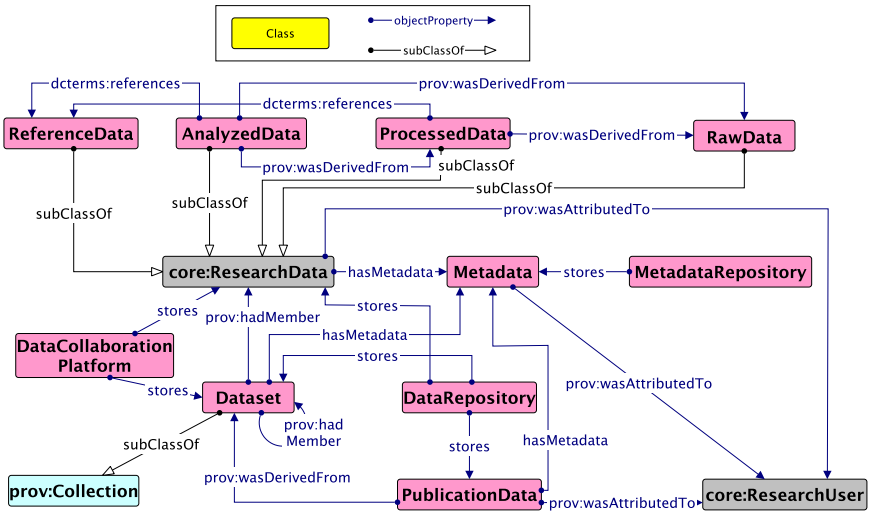


Fig. 6.7: In the dataset module, the classes and properties related to the structure of the data in the context of research are defined. Classes that belong to the same ontology share the same color.

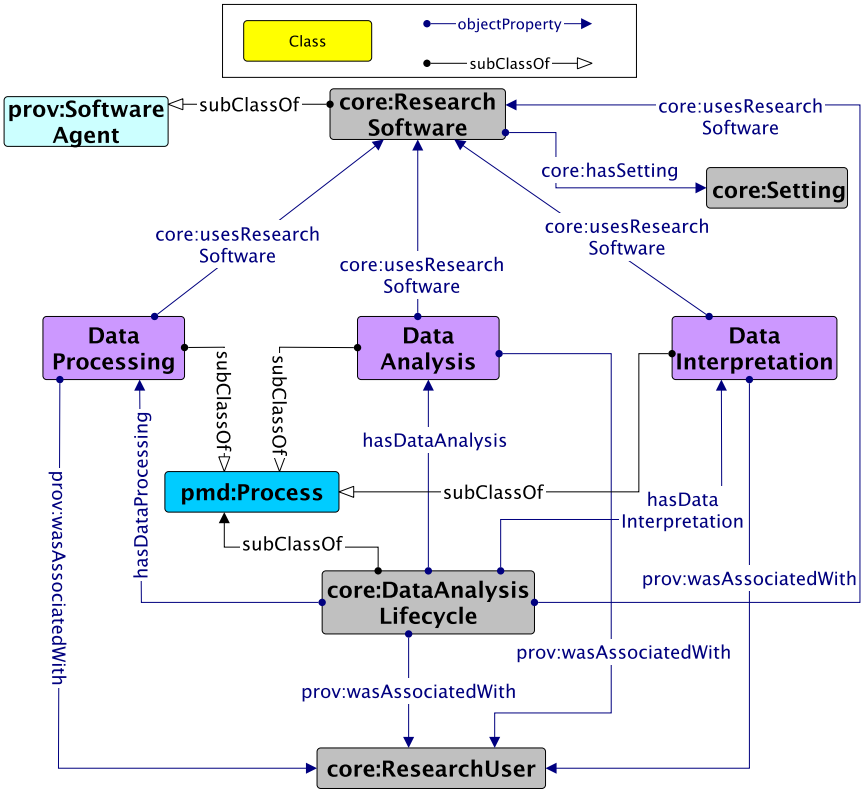


Fig. 6.8: In the data analysis lifecycle module, the classes and properties related to the data flow are described. Classes that belong to the same ontology share the same color.

6.2 Use Cases

In this section, we demonstrate the broad applicability of PRIMA by presenting two different use cases: (i) the mapping of the FAIRification workflow applied to Scanning Tunneling Microscope (STM) images from data acquisition to data analysis and (ii) the PRIMA alignment of the fabrication processes ontologies applied to metallic biomaterials recorded in the Herbie[8] Electronic Laboratory Notebook (ELN).

[8] `https://hereon.de/herbie`

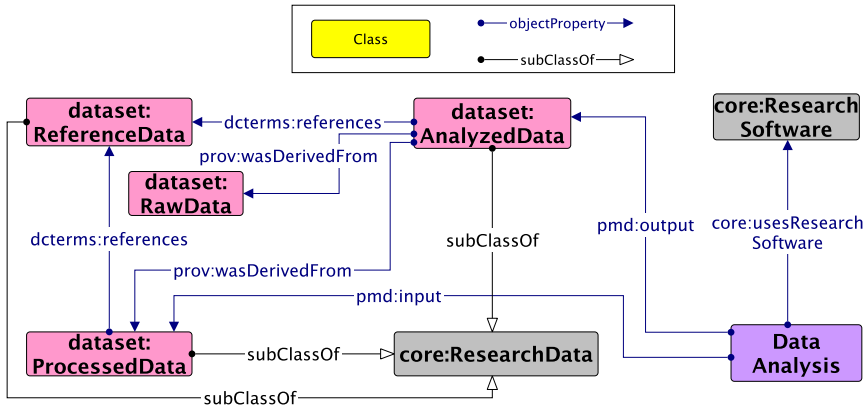


Fig. 6.9: Due to the alignment with PMDco, in the data analysis lifecyle module we can define a data flow process. Classes that belong to the same ontology share the same color.

6.2.1 FAIRification Workflow of STM Images

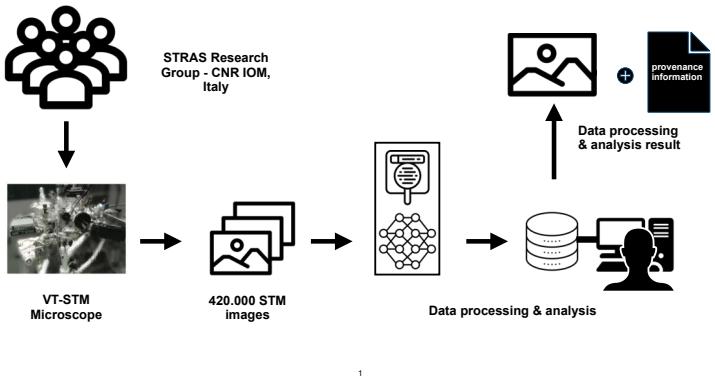


Fig. 6.10: One way to achieve FAIR data of STM images is to document the provenance information consisting of data generation, processing, and analysis.

The main information about the STM images data analysis lifecycle [161] is illustrated in Fig. 6.10, and it can be summarized as follow: The raw data consist of

roughly 420,000 STM images acquired over 20 years of research activities by the Surface Structure and Reactivity at the Atomic Scale (STRAS) research group of the Consiglio Nazionale delle Ricerche - Istituto Officina dei Materiali (CNR-IOM) in Trieste. For each STM image generated using an Omicron Variable Temperature STM (VT-STM) microscope, the measurement start and end time were recorded, together with the metadata about the measurement settings. The processed data consist of roughly 110,000 images, selected according to a particular setting of the scanning mode. The following data analysis involved human annotations and machine learning techniques, in order to collect additional information about the structure and composition of each sampled image. The finally analyzed data resulted in 7,287 images, containing a categorization as either Gr Ni100, Gr Ni111, or Gr Ni111, in addition to the previously recorded metadata. To ease the findability and the reuse of the final dataset, 11 image metadata were selected and indexed in the search engine of the STM Metadata Explorer[9], a web service developed and integrated within the Trieste Advanced Data Services (TriDAS)[10] as part of the NEP Virtual Access offer. By using the STM Metadata Explorer, the user can visually explore and select the STM images based on the available metadata. The relevant images can then be downloaded, together with the provenance information in JavaScript Object Notation (JSON).

In this use case, we extend the work by mapping its provenance data model to PRIMA. The provenance data model of STM images follows the PROV-DM standard and is serialized by the PROV-JSON serialization, i.e., the metadata is in the JSON format. Furthermore, the mapping is done by connecting JSON objects into PRIMA, so that each of JSON objects is an instance of a PRIMA class.

The mapping allows STM image data to be serialized using the Resource Description Framework (RDF)[11], a framework for creating linked data. Adopting this approach for STM image datasets [147] has several advantages, including 1) establishing links between STM and other microscopy image datasets, which allows machines to discover and connect knowledge, 2) facilitating semantic queries using SPARQL, 3) supporting data enrichment, where machines can infer implicit knowledge that does not exist, 4) collecting or adding additional metadata that is previously not covered by the range of the metadata schema used, and 5) promoting semantic validation of the data, ensuring consistency and accuracy. A sample of the instances of the STM images study is illustrated in Fig. 6.11, describing the data processing, during which raw data from a number of measurements results in processed data. The mapped data can be accessed via the GitHub repository[12].

[9] `https://tridas.nffa.eu/s-t-m-explorer.html`

[10] `https://tridas.nffa.eu`

[11] `https://www.w3.org/TR/rdf11-concepts/`

[12] `https://purls.helmholtz-metadaten.de/prima/use-case-1`

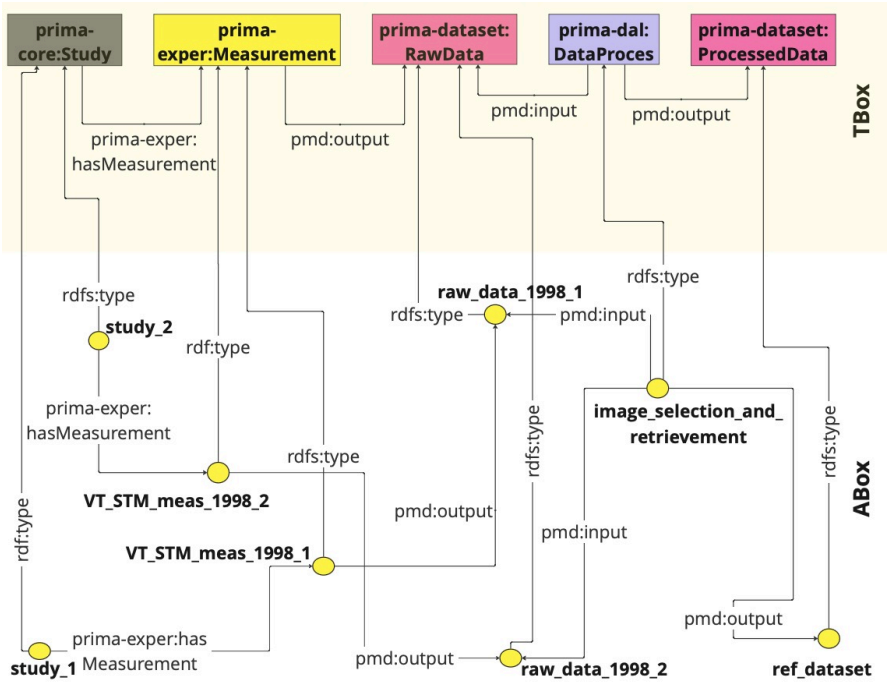


Fig. 6.11: In the T-Box panel, the data processing is formally defined using ontology and in the A-Box panel, the instances of classes are generated. Red points denote individuals of classes. Each individual is defined by an arrow having rdf:type relationship to the respective class and individuals are connected by object properties defined in PRIMA. Colored boxes represent classes from different modules and ontologies, e.g., core, experiment, dataset, and data analysis lifecycle module.

6.2.2 Metallic Biomaterials Fabrication

Metal fabrication techniques in the materials manufacturing include various metal forming techniques, casting, welding, and machining. Before a piece of material is ready to be used, two or more fabrication techniques are needed [18]. In this use case, we focused on material casting and extrusion. In material casting, a liquid material is poured into a mold, where it solidifies. Extrusion involves forcing a heated material through a die with a specific cross-sectional shape to produce objects with a continuous profile. In this workflow, the casting initially shapes the material into a form that can be further processed through extrusion. The cast material is heated to an elevated temperature and loaded into the extrusion press. As the material passes through the die, it takes on the shape of the die cavity and emerges from the press as a continuous length of material with the desired cross-sectional profile.

All the actions mentioned above are documented with Herbie [162], a hybrid system between an ELN and a research database developed at the Helmholtz-Zentrum Hereon[13]. Herbie is tailored to cover and interlink the heterogeneous process chain of metallic biomaterials research, including materials development, biological characterization, and synchrotron imaging; nevertheless, due to its modular structure, it can be adapted to other fields.

Herbie utilizes the Semantic Web technologies, specifically RDF and Shapes Constraint Language (SHACL)[14]. Furthermore, Herbie develops its data model or ontology. The ontology emphasizes on several metal fabrication techniques terms, including casting, extrusion, equipment, and setting terms. By populating its ontology with the data, Herbie can directly generate the metadata of an experiment process in RDF.

In this use case, the Herbie ontology is extended to be aligned to PRIMA. A successful ontology alignment involves identifying relationships between entities in different ontologies to establish links [148] and similarities between the source and target ontologies. The analysis focuses on concepts that overlap but may have different names (synonyms) or types in the ontologies [149]. This alignment supports the generation of linked data and boasts more interoperability of Herbie within the materials science data.

The T-Box panel of Fig. 6.12 illustrates the mapping process of aligning PRIMA and the Herbie-Metallic Biomaterials ontology (MB). We subsume, e.g., the `mb-cast:Casting`, `mb-ext:Extrusion`, and `mb-ext:Pass` classes to the `Fabrication` class from the PRIMA experiment module. Furthermore, we subsume `mb-cast:CastMaterial` and `mb-ext:ExtrusionMaterial` to `prima-exper:Precursor` and `prima-exper:Input`, as a precursor can be an input for a subsequent metal fabrication.

In the A-Box section of Fig. 6.12, the ontology population of the materials casting and extrusion processes are illustrated. The ontology population occurs when the fabrication data is documented in the ELN Herbie and mapped to its corresponding classes of ontologies. Subsequently, we identify one inference rule that can be inferred to gain new knowledge. This rule is written via SWRL [163] and was semantically validated using Drools reasoner [164]. Eq. 6.1 is employed to infer the following process can be determined given the input of the current process and the output of the previous process, e.g., the extrusion process follows the casting process, and the inference result is shown in the figure as the red dashed line. PRIMA is currently integrated in Herbie and used for annotating the materials fabrication data. An instance of the mapped data can be accessed on GitHub[15].

$$\begin{aligned} &\text{herbie-ext:pass(?ext, ?ext\_pass)} \wedge \text{pmd:input(?ext\_pass, ?cast\_mat)} \\ &\wedge \text{pmd:output(?cast, ?cast\_mat)} \longrightarrow \text{pmd:nextProcess(?cast, ?ext)} \end{aligned} \tag{6.1}$$

[13] `https://www.hereon.de`

[14] `https://www.w3.org/TR/shacl/`

[15] `https://purls.helmholtz-metadaten.de/prima/use-case-2`

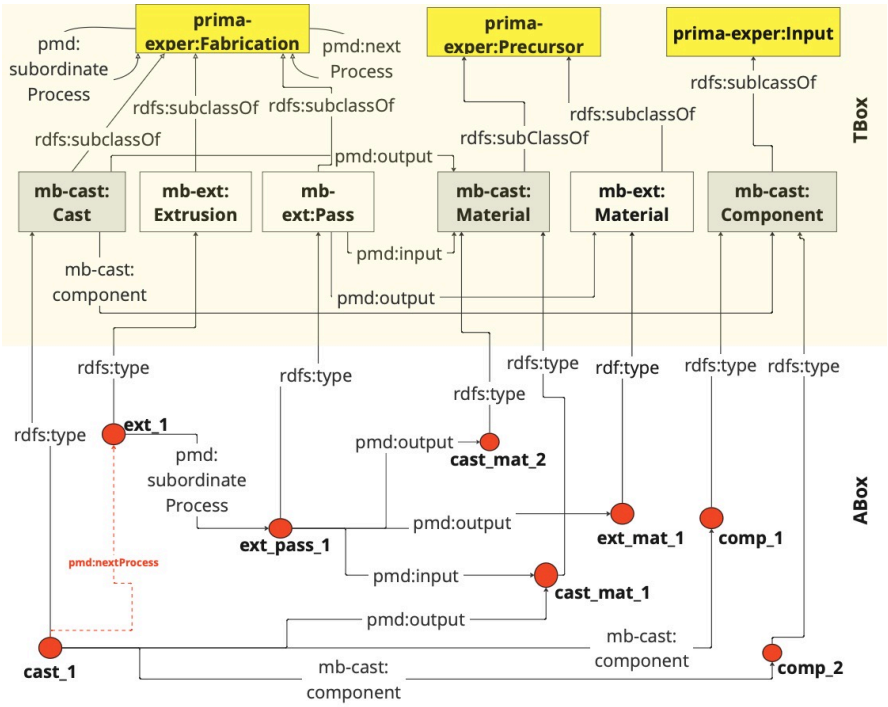


Fig. 6.12: In the T-Box, the mapping is performed to align the Herbie ontology against several PRIMA terms. Consequently, in the A-Box, we can generate inference rules via SWRL, and the inference result is shown as a red dashed line. Red points denote individuals of classes. Individual are defined by arrows having rdf:type relationship to the respective class and are connected by object properties defined in PRIMA. Colored boxes represent classes from different modules and ontologies, e.g., PRIMA and MB.

Listing 6.1: SPARQL query corresponding to CQ9 over an instance of the STM study.

```
PREFIX dal: <https://purls.helmholtz-metadaten.de/prima/dal#>
PREFIX pmd: <https://w3id.org/pmd/co/>

SELECT  ?Data_processing ?input ?output WHERE{
    ?Data_processing a dal:DataProcessing ;
        pmd:input ?input ;
        pmd:output ?output. }
```

6.3 Evaluation

In this section, we evaluate the success of PRIMA in modeling a real-world domain and its effectiveness. As in [107], the following steps were performed: 1) Defining a set of competency questions (CQs) based on the ontology specification activity, which is done in Sec. 4.1, 2) formulating SPARQL queries corresponding to CQs, 3) running the resultant SPARQL queries against two use cases using a SPARQL endpoint, and 4) analyzing the query results by comparing them to the correct answers given by domain experts.

Listing 6.2: SPARQL query corresponding to CQ26 over an Herbie instance.

```
PREFIX exp: <https://purls.helmholtz-metadaten.de/prima/experiment#>
PREFIX pmd: <https://w3id.org/pmd/co/>

SELECT ?last_action_lbl ?current_action_lbl  WHERE{
    ?last_action a exp:Fabrication;
        pmd:nextProcess ?action ;
        rdfs:label ?last_action_lbl.
    ?action rdfs:label ?current_action_lbl . }
```

We formulated the SPARQL queries corresponding to the CQs listed in Table 6.1. To exemplify the process, we show SPARQL queries corresponding to CQ9 and CQ27 in Listing 6.1 and Listing 6.2, respectively. The result of Listing 6.1, generated according to Fig. 6.11, is shown in Table 6.3, while the result of Listing 6.2, according to Fig. 6.12 is shown in Table 6.4. Subsequently, we compared the SPARQL query results with answers provided by domain experts. The comparison demonstrated strong alignment between the SPARQL query outputs and the expert responses, indicating that the ontology effectively captures and models the domain knowledge.

Other SPARQL queries corresponding to the CQs are listed on GitHub[16]. Additional ones will be addressed in the future by collaborating on other use cases.

Table 6.3: The result of CQ9 in Listing 6.1 against the data processing action data in the TEM images workflow study.

Data Processing label	Input	Output
image_selection_1	raw_data_1988_1	reference_dataset_1
image_selection_1	raw_data_1988_1	reference_dataset_1
metadata_selection_1	structured_FAIR_dataset	filtered_image_1

[16] https://purls.helmholtz-metadaten.de/prima/dev

Table 6.4: The result of CQ26 in Listing 6.2 against the fabrication data from Herbie.

Previous action label	Current action label
casting_1	extrusion_1

6.4 Summary and Conclusion

We have developed PRIMA, a modular ontology that helps document the provenance information about research studies in the materials science domain and their digital or physical outputs. It was developed following the NeOn methodology, using the MDMC-NEP Glossary of Terms as the main resource. The presented use cases demonstrate how to utilize PRIMA to document provenance information in two real-world applications. Furthermore in one use case, we discuss the inference rules that help discover new relationships, detect possible inconsistencies, and infer logical consequences from asserted facts.

Chapter 7
Materials Synthesis Provenance Mining

In this chapter, we discuss and apply Named Entity Recognition and Relationship Extraction (NERRE) leveraging several open-source Large Language Model (LLM) to transform the materials synthesis text into structured or linked data. Before fine-tuning the model, the data model or ontology is developed following several ontology best practices [92, 93] and enriched with semantic web technologies (e.g., Web Ontology Language (OWL) and Resource Description Framework (RDF)) to leverage SPARQL Protocol and the RDF Query Language (SPARQL) query ability to retrieve the structured data. Subsequently, we prepare the dataset for the LLM fine-tuning process and evaluate the fine-tuned LLM. Moreover, we use the fine-tuned LLM to extract unstructured data, e.g., materials science publications. The extracted data is then populated in the developed data model

7.1 Data Model

Developing a data model as an ontology allows for capturing and structuring knowledge about a domain of interest in a machine-understandable format. This section describes how we developed PRovenance Information for MAterials science (PRIMA) Lite, which annotates the materials synthesis data. PRIMA Lite, shown in Fig. 7.1, is a light version of the PRIMA (cf. Ch. 6) that focuses mainly on describing the provenance information on the materials synthesis action:

- The information of the publication/scientific paper relates to the materials synthesis,
- the information on how a material is generated or used,
- the information which equipment or tool used, and
- the information of actions taken in a materials synthesis.

The data model is formalized through an ontology, which is encoded via the OWL. Before modeling the data, the requirement analysis takes place resulting in a set of competency questions. Subsequently, we reuse several terms originating

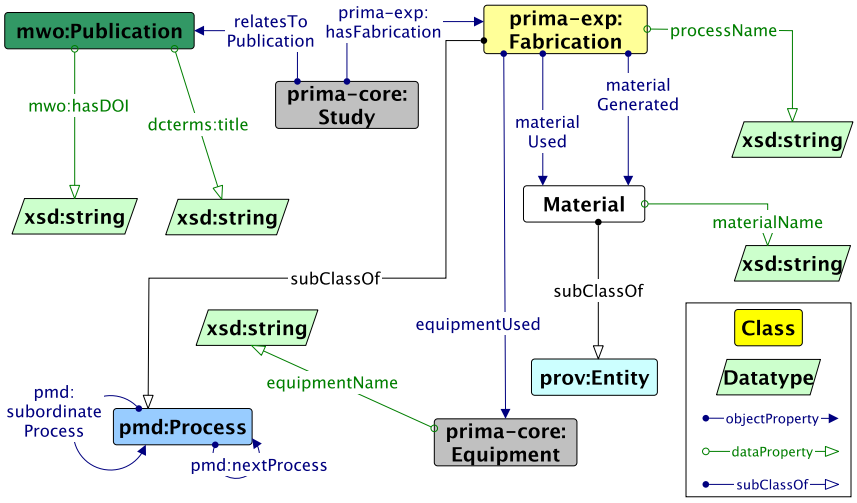


Fig. 7.1: PRIMA Lite focuses on describing the provenance information on a materials synthesis action. Arrows with open arrowheads denote `rdfs:subClassOf` properties between classes. Regular arrows visualize *rdfs:domain* and *rdfs:range* restrictions on properties. Furthermore, colored boxes represent different ontologies, e.g., PRIMA, PMD Core Ontology (PMDco), and MatWerk Ontology (MWO)

from other ontologies. Ultimately, we define classes and properties describing concepts and relationships in the materials synthesis.

7.1.1 Requirement Analysis

To ensure the ontology's effectiveness, PRIMA Lite undergoes a requirement analysis before its development. Furthermore, the requirement analysis to determine its scope. Competency Questions (CQs) are formulated during the requirement analysis phase to determine the necessary knowledge integrated into the ontology [93]. Through this analysis, five CQs have been identified, as shown in Table 7.1. For example, CQ1 requires PRIMA Lite to store information about the material generated in a materials synthesis publication or paper. At the same time, CQ2 and CQ3 focus on identifying the materials used to generate other materials and the equipment or tools used in a materials synthesis, respectively. CQ4 is designed to store the knowledge of a materials synthesis in a materials publication, and CQ5 requires PRIMA Lite to store information about generating a material, along with a set of actions taken in a materials synthesis.

Table 7.1: Competency questions of the data model.

No.	Question
CQ1	What is(are) material(s) generated from the synthesis in a publication?
CQ2	What is(are) materials(s) used for generating a material in one step of the synthesis?
CQ3	What is(are) equipment(s) used in steps of synthesis? List all of them
CQ4	List all activities sequentially to conduct a materials synthesis in a publication.
CQ5	List all activities sequentially to generate 'x' material, and for each activity, list all materials (used/generated) taking part in each activity.

7.1.2 Reusing of Existing Models

One of the initial steps to develop PRIMA Lite is the ontology reuse. Ontology reuse is the process of reusing one or several terms from other ontologies. This is done to facilitate knowledge exchange and enable interoperability between ontologies. The value of semantic data in a model increases as the number of reused ontologies increases. Consequently, by utilizing the terms from other ontologies, the semantic data value of the created ontology is improved [107]. We reused a term from the experiment module of PRIMA, e.g., `prima-exp:Fabrication`. As `pmd:Process`, a class reused from PMDco[1], subsumes `prima-exp:Fabrication`, the ability to describe a set of actions is inherited to this class. Moreover, we reused two terms called `prima-core:Study` and `mwo:Publication` from the core module of PRIMA and MWO[2], respectively. The former defines the study that the scientist are focused on, and the latter describes the publication entity and its data, such as publication Digital Object Identfier (DOI) and publication title.

7.1.3 Classes and Object properties

Apart from the reused classes from other ontologies, we define a class called `Material`, which represents an entity used or generated during the materials synthesis. In terms of properties, PRIMA Lite has several object properties that relate to classes. For instance, `prima-core:Study` relates to `mwo:Publication` using `relatesToPublication` and relates to `prima-exp:Fabrication` using `prima-exp:hasFabrication`. Furthermore, to define a set of actions in `prima-exp:Fabrication`, we can use `pmd:nextProcess` and `pmd:subordinateProcess` relations.

[1] `https://w3id.org/pmd/co/`

[2] `http://purls.helmholtz-metadaten.de/mwo/`

7.1.4 Data Properties

Data property is a property relating the class with its literal data, e.g., a string name and an integer number. In PRIMA Lite, the `mwo:hasDOI` and `dcterms:title` link the `mwo:Publication` class to its literal string data. Furthermore, `materialName`, `processName`, and `equipmentName` relate `Material`, `prima-exp:Fabrication`, and `prima-core:Equipment` to the literal string data, respectively.

7.2 Methods

Since the release of the transformer-based architecture [125], many state-of-the-art language models adopting this architecture emerged, beating the classical language models' ability, such as Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM), in several natural language processing benchmarks. However, domain scientists still need to fully explore the application of these language models, particularly in materials science. This section describes the methods for choosing a suitable dataset for our study, fine-tuning, utilizing LLM for the downstream task in the materials synthesis procedures text, and combining them with the semantic web technologies to generate linked data.

7.2.1 Training Dataset

We use the dataset of the materials science procedural text corpus [165] to train language models. The dataset consists of 230 materials synthesis procedures, and it has been annotated by domain experts along with labeled graphs expressing the semantics of the synthesis sentences. Each synthesis procedure originates from a materials science publication describing materials synthesis, particularly the synthesis paragraphs or methods section of a publication. All data annotations were performed using the BRAT annotation tool [166], and the dataset is openly accessible through the repository[3].

The information from the dataset consists of two types: the entity type and the relation type. For the entity type, we parse the information that belongs to `Material`, `Operation`, `Synthesis-Apparatus`, and `Nonrecipe-Material` entity. Moreover, we parse `Recipe-target`, `Solvent-material`, `Recipe-precursor`, `Atmospheric-material`, and `Apparatus-of` relation for the relation type. All relation types will have a domain of `Operation` entity and a range of either `Material` or `Synthesis-Apparatus`. For instance, the relation type of `Recipe-target`, `Solvent-material`, `Recipe-precursor`, and `Atmospheric-material` have a range of `Material`

[3] `https://github.com/olivettigroup/annotated-materials-syntheses`

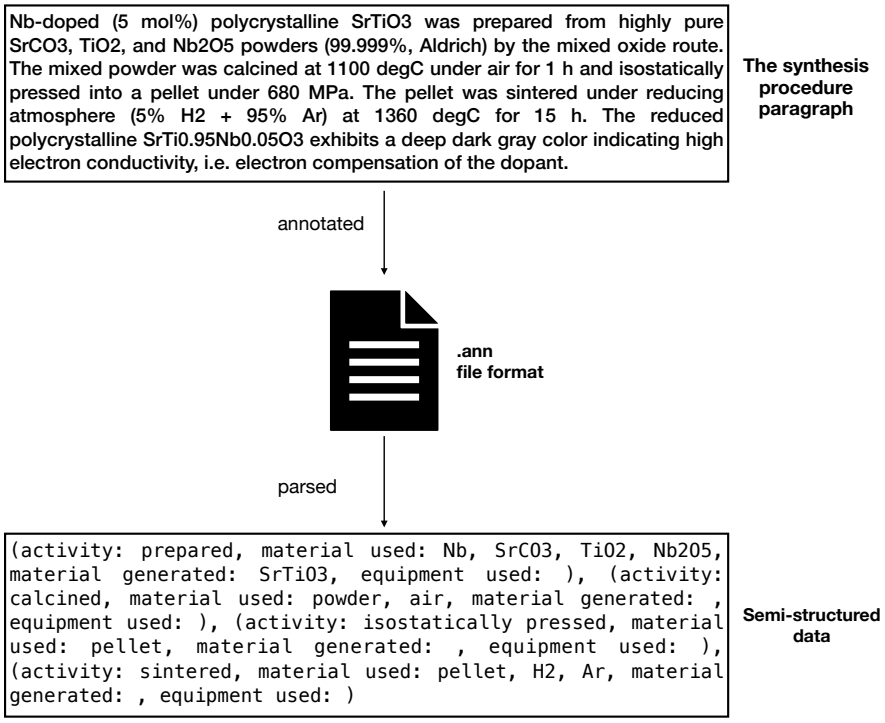


Fig. 7.2: Generating the training dataset. On the upper level is the example of the synthesis procedure [167] and on the center is the annotated file of the synthesis procedure. To generate the training dataset, we parsed some information from the annotated file into semi-structured data.

and `Nonrecipe-Material` entities. However, the `Apparatus-of` relation will have the range of `Apparatus` entity.

To convert the annotated dataset, we first parse the annotated paragraph encoded in the BRAT annotation file format (*.ann*). This is done by converting the annotated paragraphs into semi-structured data. As shown in the bottom of Fig. 7.2, the semi-structured data has a form that is similar to a dictionary, comprising keys of `activity`, `material used`, `material generated`, and `equipment used` and its corresponding value. Furthermore, the collection of keys constitutes the provenance information we want to extract, as defined in CQs.

The next question is how we map the dataset into the "semi-structured" data format. We map the entity of `Operation` into the `activity` key. Since the `Operation` is the domain of every relation, the same goes for the `activity` key. Subsequently, we map the `Apparatus-of` relation into the `equipment used` key and the `Recipe-target` into the `material generated` key. Ultimately, apart

from the relation mapped above, it will be mapped into the `material used` key. The Python script to parse the annotated file can be accessed in our repository[4].

From the 230 paragraphs of synthesis procedures, we kept 200 paragraphs for the training dataset and the rest of 30 paragraphs for the testing process. For each paragraph and its semi-structured data representation in the training dataset, subsequently, we split the paragraph into sentences along with the semi-structured data corresponding to its sentence. In that way, we train the language models at the sentence level. Ultimately, 1.590 sentences are used to fine-tune the language models.

7.2.2 Fine-tuning Large Language Models

The objective of fine-tuning large language models for this study is to train language models that is able to automatically convert the materials synthesis text into semi-structured data, preserving the provenance information, as shown in Fig. 7.2. In order to do that, we experiment with two types of language models, which are based on the *decoder-only* language model and the *encoder-decoder* language model, in which they are all based on Transformers architecture [125]. As shown in the Fig. 7.3, the decoder-only language model is auto-regressive, i.e., the input and the preceding generated tokens/words influence the output at each step. We can find two applications of the decoder-only language model in the task of text completion and language generation. Two examples of the decoder-only language models are GPT [129] and Falcon [131].

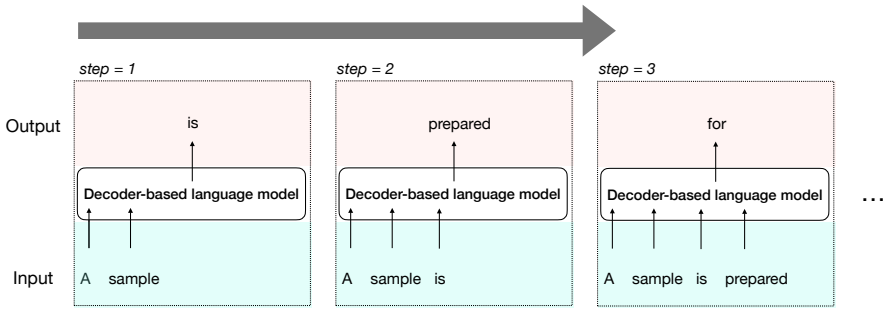


Fig. 7.3: Decoder-based language model. It has an autoregressive nature, in which the input and the preceding generated tokens/words influence the output at each step.

[4] `https://github.com/Materials-Data-Science-and-Informatics/prima-lm/blob/main/prima_lm/text/ann_parser.py`

For the encoder-decoder language model, first, the encoder part learns the input text's structure and converts it into a meaningful representation, the so-called "context vector" or "embedding". Given the context vector, the decoder part generates the text given the meaningful representation from the encoder output. As shown in Fig. 7.4, a translation from an English to German sentence is one example on how encoder-decoder is used in the natural language task. Two examples of the encoder-decoder only language models are FLAN-T5 [168] and FLAN-UL2 [169].

For this study, we fine-tune three LLM: *Falcon7b*, *FLAN-T5*, and *FLAN-UL2*, retaining 7, 11, and 20 billion parameters, respectively. These pre-trained LLM come from the Huggingface [170] hub. The fine-tuning process optimizes the cross-entropy loss on predicted tokens. The ratio of the training and validation dataset is 4:1 of the total 1.590 sentences. Subsequently, all models are fine-tuned for 20 epochs at a batch size of 64.

Moreover, instead of training all parameters of each of the pre-trained models, we utilize Low-Rank Adaptation (LoRA) [171], which is a Parameter-efficient fine-tuning (PEFT) technique to fine-tune the LLMs by adapting pre-trained LLM to various downstream applications. It is done without the need to fine-tune all parameters of the LLM due to the prohibitive cost associated with such an approach. LoRA involves keeping the pre-trained model's weights fixed while injecting trainable rank decomposition matrices into every layer of the Transformer architecture. Using LoRA reduces the number of trainable parameters, thus making the fine-tuning process more efficient. For instance, we fine-tune the FLAN-UL2 using a rank of 64, which means we fine-tune only 1% of the parameters from the total of $\approx$ 20 billion parameters. Apart from LoRA, the model parameters are also quantized into 8-bit representation instead of 32-bit representation. Combining the two techniques above results in Quantized Low-Rank Adaptation (QLoRA) [172]. This greatly reduces the Graphics Processing Units (GPUs) memory consumption during the training and inference process. All the python scripts used for the fine-tuning process are available on the github repository[5].

7.2.3 Metrics Evaluation

Each of the fine-tuned LLMs is evaluated using two metrics adapted from [83] given the test dataset. These two metrics are the individual matching and triple matching metrics. The individual matching evaluates the ability of the language model to match the entity of a word (individual) in the sentence, and this word is subsequently classified into one of three entities: *Activity*, *Material*, and *Equipment*. The score of individual matching is computed by first converting individuals of an entity, E, into a set of n comma-separated words/individuals $E = \{ind_1, ind_2, ind_3, ..., ind_n\}$. Subsequently, we generate the E^{target} and E^{system} and count the number of exactly matching individuals in both sets as true pos-

[5] https://github.com/Materials-Data-Science-and-Informatics/prima-lm/tree/main/prima_lm/train

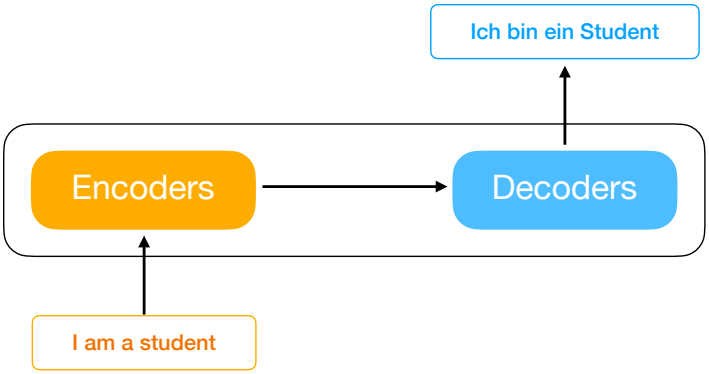


Fig. 7.4: Encoder-decoder language model. First, the encoder part learns the structure of input text and convert it into meaningful representation. The resulting representation is subsequently conveyed into the decoders to generate the output text that might be in another language or format.

itives ($E^{\text{target}} \cap E^{\text{system}}$). Furthermore, the mathematical differences between the sets as false positives ($E^{\text{system}} - E^{\text{target}}$) or false negatives ($E^{\text{target}} - E^{\text{system}}$). Ultimately, we compute the precision, recall, and F_1-score.

The triple matching metric evaluates the ability of the language model to generate the relationship between entity individuals. Each triple consists of a Subject, Predicate, and Object (SPO) tuple structure. We fix the value of the subject into an individual, $S(ind_n^{act})$, of the Activity entity, E_{act}. However, we vary the predicate and object value resulting in combinations as follows:

- $S(ind_n^{act})$ - P(material used) - $O(ind_n^{mat})$,
- $S(ind_n^{act})$ - P(material generated) - $O(ind_n^{mat})$, and
- $S(ind_n^{act})$ - P(equipment used) - $O(ind_n^{eq})$.

Where, $O(i_n^{mat})$ and $O(i_n^{eq})$ are an individual of the Material entity, E_{mat}, and an individual of the Equipment entity, E_{eq}, respectively.

Now, suppose there are a target set of triples, T^{target}, and a system set of triples, T^{system}. T^{system} is evaluated by computing the numbers of triples found/matched in both sets ($T^{\text{target}} \cap T^{\text{system}}$) as true positives and the differences between these two triples sets as false positives ($T^{\text{system}} - T^{\text{target}}$) or false negatives ($T^{\text{target}} - T^{\text{system}}$). With correct and incorrect triples identified, F_1-score for each relation are calculated as:

$$\text{precision} = \frac{\text{No. of matched triples}}{\text{Number of } T^{\text{target}}} \tag{7.1}$$

$$\text{recall} = \frac{\text{No. of matched triples}}{\text{Number of } T^{\text{system}}} \tag{7.2}$$

$$F_1 = \frac{2(\text{precision} \cdot \text{recall})}{\text{precision} + \text{recall}} \tag{7.3}$$

7.2.4 Inference and Data Linking

The term inference denotes using a trained or fine-tuned language model for generating predictions or producing novel text by leveraging input data [118]. In the inference stage, the model utilizes its acquired patterns and knowledge from the fine-tuning process to interpret the input information and generate output data. As shown in Fig. 7.5, we combine the inference process with "the paragraph to sentences" process, splitting the materials synthesis procedures paragraph into sentences. This splitting process uses SpaCy[6] and its transformers-based language model, `en_core_web_trf`. Each sentence is used to input the fine-tuned language model, resulting in semi-structured sentence data. The collection of semi-structured sentence data is subsequently merged back into a semi-structured paragraph data shown on the bottom of Fig. 7.2.

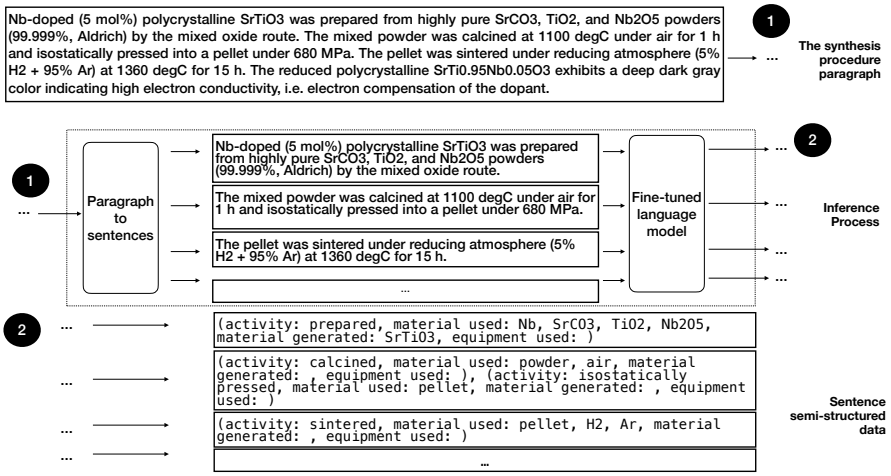


Fig. 7.5: The inference process takes the synthesis procedure paragraph and splits it into sentences. Each sentence is conveyed to the fine-tuned language model, resulting in a sentence semi-structured data. The collection of sentence semi-structured data is subsequently merged back into a paragraph semi-structured data. Number denotes the continuous workflow connecting each process.

Once the inference process is completed, we use PRIMA Lite to annotate the semi-structured data obtained from the inference output, as shown in Fig. 7.6. We develop a parser utilizing the regular expression to convert the semi-structured data into a Python list. In this way, we can preserve the order of synthesis actions based on the sentence's appearance order in a synthesis paragraph. Ultimately, after the annotation process, the linked data is generated. There are many benefits to

[6] `https://github.com/explosion/spaCy`

publishing materials synthesis data as linked data [173], including 1) enabling machines to understand and discover new information by establishing links between materials synthesis datasets, 2) supporting semantic querying via the SPARQL, 3) promoting data enrichment, and 4) ensuring consistency and accuracy through semantic validation of the data.

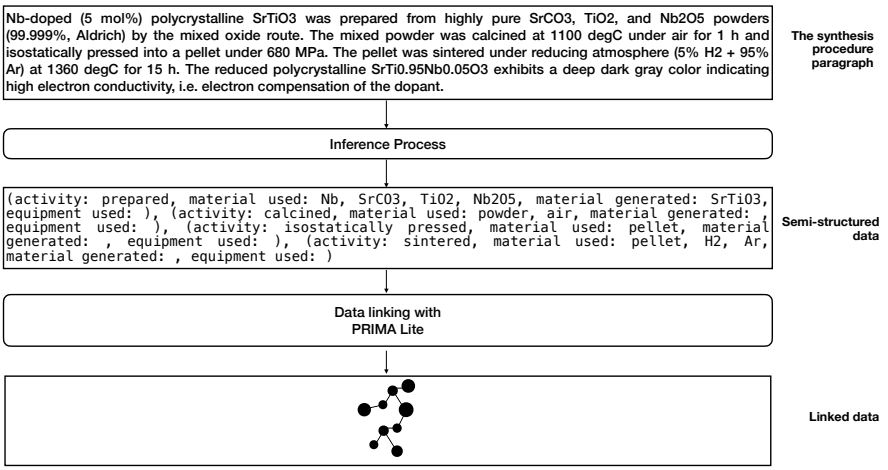


Fig. 7.6: The inference and data linking process. The overarching goal of this study is to develop the LLM framework converting the unstructured text data, e.g., materials synthesis text, into structured data, preserving the provenance information of the materials synthesis. The structured data annotated by PRIMA Lite results in materials synthesis linked data.

7.3 Results and Discussion

This section discusses and evaluates three LLMs leveraged in this study by first showing their performance in the fine-tuning process. Subsequently, the best model of each fine-tuned model is used for metric evaluation, as described in Subsec. 7.2.3. We also show how the linked data generated in this study is retrieved. Ultimately, we evaluate the ontology by answering CQs listed in Subsec. 7.1.1 by creating corresponding SPARQL queries.

7.3.1 Fine-tuning Results

Exploratory data analysis. Before we fine-tune each model, the exploratory data analysis is done by analyzing the histogram of words and numbers of individuals and relationships in the parsed training dataset. The histogram of words is shown in the Fig. 7.7a. It can be seen that the majority of sentences have 20 words per sentence, with a deviation of around ten words, and there are 20 sentences having more than 50 words per sentence. Furthermore, the error of splitting sentence process in the dataset parsing is negligible as the number of sentences with word numbers per sentence that are less than three is only two.

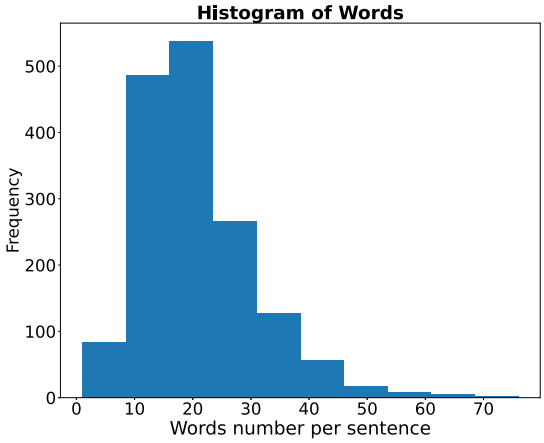
The average individual's value of an entity in each sentence is shown in the Fig. 7.7b. While the *Activity* and *Material* entities have approximately the same average value, i.e., each sentence at least has two Activity individuals and two Materials individuals. The Equipment entity has the lowest average value of 0.25, i.e., for every four sentences, at least there is one individual of Equipment. We can see that the class is imbalanced in the training dataset, especially for the training data related to equipment, because only some activities in materials synthesis need equipment.

The Fig. 7.7c shows the average relations value of each sentence. Compared with the average individual's value, the value of the average relations shows a class imbalance between the Material used and two other relations: Material generated and Equipment used. For the Material used relation, the average value is 1.9, i.e., for every sentence, at least two materials are used. In contrast with the Material used relation, the Material generated relation has an average value of 0.37, i.e., for every three or four sentences, one Material is generated. It explains that for every Material generated in materials science, the text needs synthesis actions written in three or four sentences before the material generation.

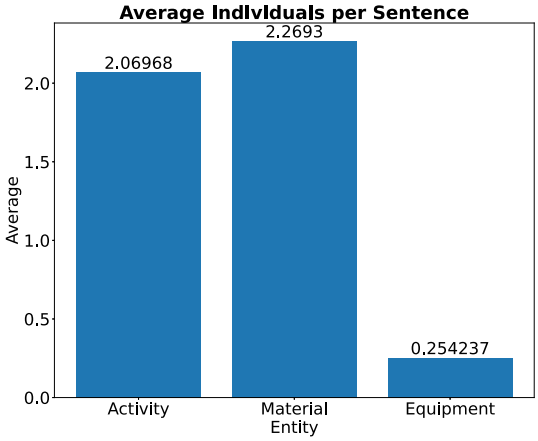
As we have seen in the data, the class imbalance is evident in the dataset. Class imbalance occurs when certain entities, such as *Equipment*, appear less frequently than others, such as *Activity* or *Material*. Common strategies to mitigate class imbalance include sampling methods (e.g., oversampling minority classes or undersampling majority classes), data augmentation to increase the size of underrepresented classes artificially, and adjusting the cost function by assigning higher weights to minority classes during model training.

However, these strategies are unnecessary in our study because we use the F1-score as one of our key evaluation metrics, which inherently addresses class imbalance [174]. The F1-score balances precision (how many predicted positive instances are correct) and recall (how many actual positive instances were correctly identified). This balance ensures that the model does not simply favor the majority class to achieve high accuracy. Instead, it forces the model to perform well on frequent and infrequent classes, making it a robust metric for evaluating performance in imbalanced datasets.

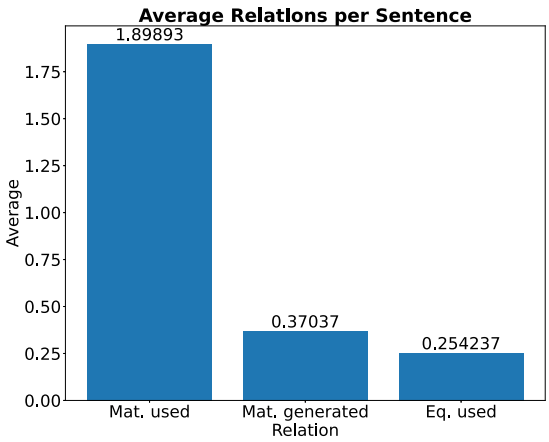
LLMs best model. The training and validation loss curves for three LLMs fine-tuned using the parsed dataset are shown in Fig. 7.8. The train and validation losses decrease during the initial training phase for the FLAN-T5 XXL and FLAN-UL2 (see



(a) Histogram of Words

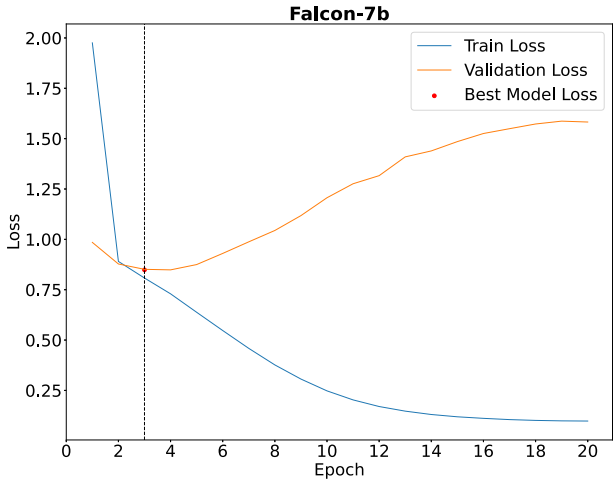


(b) Average Individuals per Sentence

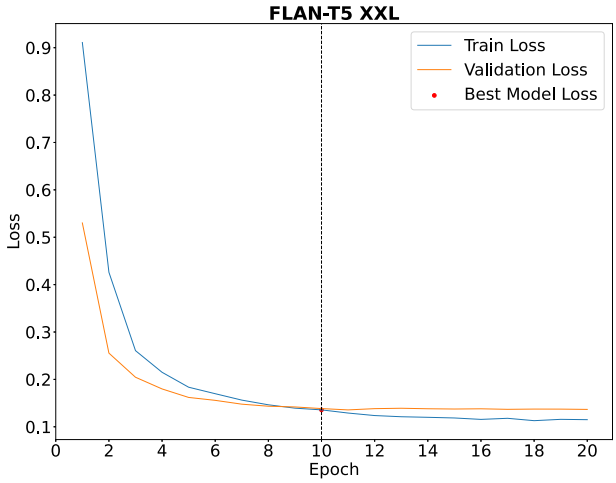


(c) Average Relations per Sentence

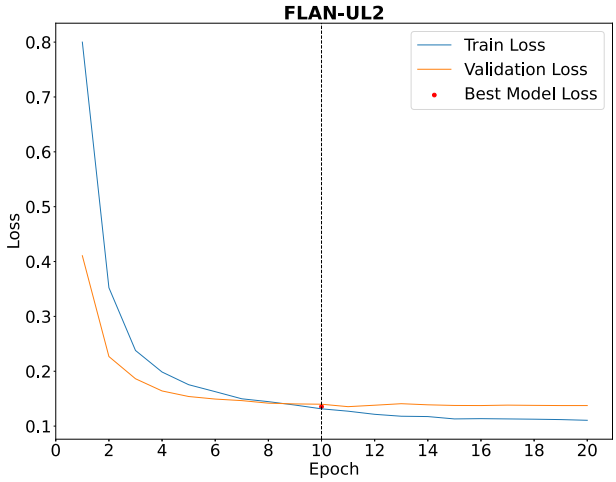
Fig. 7.7: Exploratory Data Analysis on training dataset: (a) Histogram of Words, (b) Average Individuals per Sentence, and (c) Average Relations per Sentence.



(a) Training and Validation Loss for Falcon-7b



(b) Training and Validation Loss for FLAN-T5 XXL



(c) Training and Validation Loss for FLAN-UL2

Fig. 7.8: Training and validation loss across epochs for different models: (a) Falcon-7b, (b) FLAN-T5 XXL, and (c) FLAN-UL2. The best model checkpoint is indicated with a red marker and a dashed vertical line.

the center and right panel of the figure, respectively). This demonstrates a typical behavior among well-trained models, as the optimization process is expected to decrease the loss function value as time progresses [25]. Furthermore, we find that the model overfitting occurs when these trained LLMs reach the tenth epoch. After the tenth epoch, the train loss values keep decreasing while the validation loss values saturate in a value. Consequently, we choose the lowest validation loss on the FLAN-T5 XXL and FLAN-UL2 to be the chosen models showing good generalizations on the training dataset.

Compared to FLAN-T5 XXL and FLAN-UL2, Falcon-7b (cf. Fig. 7.8a) has poor train and validation loss values. The model tends to overfit the training dataset by the third epoch, which we selected as the best model to prevent overfitting. After the third epoch, the train loss values decrease while the validation loss values gradually increase. Therefore, we conclude that Falcon-7 requires different training strategies for this Natural Language Processing (NLP) task to avoid a common hallucination problem occurring decoder-based models. This problem causes the decoder-based model to present false or misleading information.

7.3.2 Fine-tuned LLMs Evaluation

Individual matching evaluation on Named Entity Recognition (NER) task. As we have the best parameters for each LLM, we then evaluate each model using metrics described in Subsec. 7.2.3. The first metric to evaluate each model is called the individual matching metric. In this metric, the ability of the language model to match the entity of a word (individual) in a sentence is evaluated. The final metric scores for the Falcon-7b, FLAN-T5 XXL, and FLAN-UL2 are shown in Table 7.2.

The best F1 score for all classes owned by the FLAN-T5 XXL, with 0.84, 0.77, and 0.57 on *Activity*, *Material*, and *Equipment* class, respectively. It means that FLAN-T5 XXL outperforms the Falcon-7b and FLAN-UL2 in terms of ability to recognize the entity of words. For the latter, the FLAN-T5 XXL performs better than the largest model we use, the FLAN-UL2, as we compare the FLAN-T5 XXL retaining 11 billion parameters with the FLAN-UL2 retaining 20 billion parameters.

We can see metrics scores for the Equipment class on every model are the lowest in comparison to other classes. The lowest scores on Equipment class is due to the limited availability of equipment data as shown in the Fig. 7.7b, making it challenging for the fine-tuned LLMs to correctly classify words as equipment. In contrast, the metrics score for the Activity class shows the highest score in comparison to other classes. This is due to the training dataset on the Activity class is sufficient for a model to learn.

Triple matching evaluation on RE task. In triple matching evaluation, the ability of the model to generate the relationship between entity individuals is evaluated. As described in Subsec. 7.2.3, the relationship connects the Activity individual with the material and equipment individual. Furthermore, there are three relationships: *equipment used*, *material used*, and *material generated*, and the metrics eval-

Table 7.2: Metrics evaluation on individual matching for Falcon-7b, FLAN-T5 XXL, and FLAN-UL2. Each metric results in F1-score, Precs. as Precision, and Recall

LLM	**Falcon-7b**			**FLAN-T5 XXL**			**FLAN-UL2**		
Class	**F1**	**Precs.**	**Recall**	**F1**	**Precs.**	**Recall**	**F1**	**Precs.**	**Recall**
Activity	0.78	0.75	0.81	0.84	0.79	0.89	0.84	0.78	0.92
Material	0.72	0.68	0.76	0.77	0.72	0.84	0.77	0.70	0.85
Equip.	0.49	0.57	0.42	0.57	0.57	0.55	0.50	0.46	0.55

uation scores of triple matching for the Falcon-7b, FLAN-T5 XXL, and FLAN-UL2 are shown in Table 7.3, Table 7.4, and Table 7.5, respectively.

The FLAN-T5 XXL possesses the highest score of triple matching evaluation as the model outperforms Falcon-7b and the larger model, FLAN-UL2. Apart from that by looking into all tables, the triple of (*Activity, material used, Material*) shows the largest F1-score ranging from 0.64-0.73. Furthermore, the triple of (*Activity, equipment used, Equipment*) receives a fair F1-score ranging from 0.48-0.54, and the triple of (*Activity, material generated, Material*) has the lowest F1-score ranging from 0.24-0.36.

From these results, we conclude that our fine-tuned LLMs perform very well when predicting the triple of (*Activity, material used, Material*). Furthermore, when it comes to the prediction of (*Activity, equipment used, Equipment*) triple our fine-tuned LLMs yield a fair score. However, in terms of the prediction of (*Activity, material generated, Material*) triple, it possesses the lowest score compared to other predictions. We find that our fine-tuned LLMs suffer from this triple prediction as determining the generation of the material in a materials synthesis procedure requires context from other sentences in a paragraph. However, our study's fine-tuned models work for the sentence level, where each model is fine-tuned based on the sentence rather than the paragraph level. Different fine-tuning strategies will be further developed in the future study.

Table 7.3: Metrics evaluation on triple matching for Falcon-7b

Triple	**F1**	**Precision**	**Recall**
(Activity, equipment used, Equipment)	0.48	0.56	0.42
(Activity, material used, Material)	0.64	0.62	0.66
(Activity, material generated, Material)	0.24	0.20	0.31

Table 7.4: Metrics evaluation on triple matching for FLAN-T5 XXL

Triple	F1	Precision	Recall
(Activity, equipment used, Equipment)	0.54	0.55	0.53
(Activity, material used, Material)	0.73	0.70	0.75
(Activity, material generated, Material)	0.36	0.26	0.60

Table 7.5: Metrics evaluation on triple matching for FLAN-UL2

Triple	F1	Precision	Recall
(Activity, equipment used, Equipment)	0.50	0.47	0.53
(Activity, material used, Material)	0.72	0.69	0.75
(Activity, material generated, Material)	0.33	0.23	0.54

7.3.3 Information Retrieval

The advantage of having a data model for our structured data is the ability to query/retrieve the information in data. By linking data with the help of semantic web technologies, e.g., OWL and RDF, the material synthesis data is stored in the RDF graph. Moreover, to query this linked data that is RDF encoded, one can utilize a World Wide Web Consortium (W3C)[7] standard, the so-called SPARQL. SPARQL is the standard query language and protocol for linked data and RDF graph database.

In Subsec. 7.1.1, we have established what are known as CQs, which are questions that a data model or ontology should be able to answer. By having these CQs in the data model, we can better understand how the data is stored and retrieved. In order to do that, we need to transform natural language text questions or CQs into SPARQL to retrieve the data. Therefore, we carry out the task of generating SPARQL queries that correspond to the CQs we defined; these are available through our GitHub repository [175]. Ultimately, we can retrieve the provenance information in the materials synthesis data with these queries.

For instance, the provenance information related to how the materials synthesis is performed can be seen in CQ5. Fig. 7.9 shows the visual representation of SPARQL query result on CQ5, where we query a sequence of actions taken to generate a material called "HfO_2". There, we can see red dots representing individuals of two classes: `prima-exp:Fabrication` and `Material`. The `prima-exp:Fabrication` individual connects to another `prima-exp:Fabrication` individual via the `pmd:nextProcess` property, and

[7] `https://www.w3.org`

each individual is related to the action name as a string literal. Furthermore, each action relates to either or both materials used or generated in an action.

According to the documentation of the dataset we used [165], relations across sentences are minimized for simplicity and to stick closely to a sentence-level shallow semantic, i.e., relations between entity individuals occur only in a sentence or with the adjacent sentence. Thus, the action sequence shown in Fig. 7.9 corresponds to the sentence appearance order in the material synthesis paragraph. Furthermore, the dataset annotation does not allow for arguments to have multiple parents, e.g., a material "HfO_2" that is used for a preparation action, a second red dot step from above, would be physically/chemically in a different state from the one in the preceding action, "use" action, due to the operation took place in the previous action. Thus, similar material names may be seen in different actions in the figure, but in reality, they are in different states.

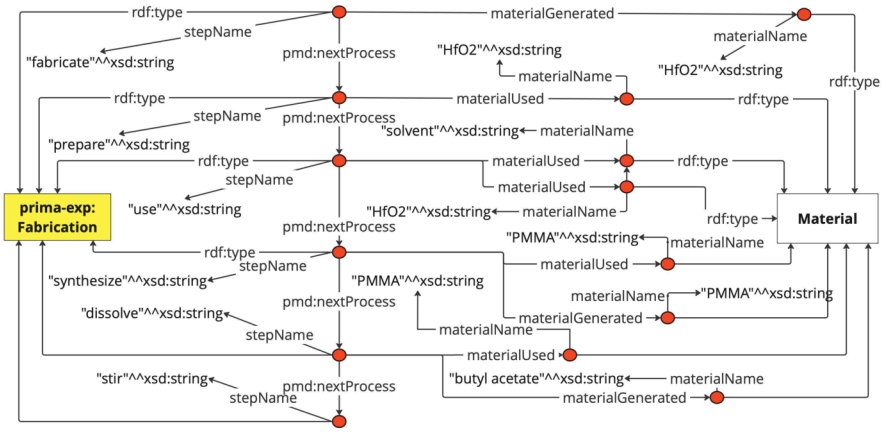


Fig. 7.9: An excerpt visual representation of CQ5 result. It shows how the materials synthesis is encoded via the RDF graph format. Colored boxes represent classes, e.g., `prima-exp:Fabrication` and `Material`, and the red dot represents an individual belonging to that class. Furthermore, each individual is defined by a directed arrow having the `rdf:type` relationship to the respective class and connected to other individuals by object properties.

7.3.4 Ontology Evaluation

The idea of ontology evaluation is to evaluate the success of the ontology in modeling a real-world domain data. This is done by carrying out the following steps in order to evaluate the effectiveness of PRIMA Lite:

1. Defining a set of CQs based on the requirement analysis (see Subsec. 7.1.1)
2. Formulating SPARQL queries corresponding to CQs
3. Running the resultant SPARQL queries against the materials synthesis linked data using a SPARQL endpoint
4. Analyzing the query results by comparing them to the correct answers given by domain experts

Since we have defined CQs as listed in Table 7.1, the subsequent step to evaluate PRIMA Lite is by formulating the SPARQL queries corresponding to CQs. For instance, in CQ1, the provenance information related to generated materials in a synthesis is queried, and the SPARQL query that corresponds to the CQ1 is shown in Listing 7.1. The complete SPARQL queries correspond to all CQs available in the GitHub repository [175]. Moreover, we run the SPARQL query against the materials synthesis linked data, and the results of this query can be seen in Table 7.6. The result shows the materials generated information in a materials science publication represented by its DOI. Suppose one wants to query papers related to that material; they can use this SPARQL query to retrieve provenance information related to material generated in the materials science text.

Listing 7.1: SPARQL query corresponding to CQ1.

```
PREFIX mwo: <http://purls.helmholtz-metadaten.de/mwo/>
PREFIX p_plan: <http://purl.org/net/p-plan#>
PREFIX prima_lite: <https://purls.helmholtz-metadaten.de/prima/lite#>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT ?doi  ?material_generated  WHERE{
    ?experiment prima_lite:relatesToPublication ?publication ;
        prov:hadPlan ?plan.
    ?publication mwo:hasDOI ?doi.
    ?step p_plan:isStepOfPlan ?plan ;
        prima_lite:materialGenerated ?matgen.
    ?matgen prima_lite:materialName ?material .
}
```

Table 7.6: The result of CQ1 in Listing 7.1 against the materials synthesis linked data

DOI	Material generated
10.1016/j.mssp.2014.02.048	$CoFe_2O_4$
10.1016/j.electacta.2013.09.101	$SrPdO_3$
10.1038/srep23400	$SrTi_{1-2x}Fe_xTa_xO_3$
...	...

7.4 Summary and Conclusion

We have presented a text mining framework leveraging pre-trained large language models (e.g., Falcon-7b, FLAN-T5 XXL, and FLAN-UL2) and semantic web technologies (e.g., RDF, OWL, and SPARQL) for extracting provenance information in the materials synthesis text. Pre-trained LLMs were subjected to a fine-tuning process to transform the unstructured data into the semi-structured data utilizing a pre-defined format. Furthermore, we showed the evaluation of fine-tuned LLMs on several tasks, such as NER and Relationship Extraction (RE), which show decent results in almost all metrics. From the LLMs evaluation, we find that the FLAN-T5 XXL shows the best performance compared to two other LLMs. We conclude that the encoder-decoder based language model is more suitable for transforming the unstructured data into the predefined structured data in comparison with the decoder-only language model. Apart from fine-tuning LLMs, we also developed a data model or ontology of the so-called PRIMA Lite as the data model. By linking the data with the help of ontology, we can retrieve the provenance information using SPARQL.

Chapter 8
Conclusion and Outlook

We explored several studies towards the digital transformation of Materials Science and Engineering (MSE) through four studies: The development of two ontologies (the Dislocation Ontology (DISO) and the PRovenance Information for MAterials science (PRIMA)), the dislocation data enrichment utilizing the DISO, and text mining on materials synthesis provenance. These efforts collectively aim to improve data interoperability and reusability, supporting novel discovery in MSE, and ultimately lead the digital transformation in MSE.

We developed DISO, which is an ontology representing concepts and relationships related to linear defects in crystalline materials. Developed using a top-down approach and published with a persistent identifier, DISO adheres to World Wide Web Consortium (W3C) best practices, ensuring its accessibility and interoperability. The ontology evaluation confirmed DISO's quality and usefulness, with practical utility exemplified by the Resource Description Framework (RDF) dataset generated from the dislocation dynamics and crystal structure data. This ontology provides a vocabulary for representing dislocation data and facilitates the integration with other MSE-related domains.

Leveraging DISO as a backbone ontology and semantic web technologies, we transformed unstructured Discrete Dislocation Dynamics (DDD) data into structured data as linked data. We also adapted DISO by aligning with commonly used materials science ontologies such as Elementary Multi-perspective Material Ontology (EMMO) and Materials Design Ontology (MDO) core. After the adaptation, we generated a semantic network called Dislocation Knowledge Graph (DisLocKG). This network enables advanced semantic querying and supports intelligent tasks, improving DDD data accessibility and reusability. The public availability of DisLocKG, along with a SPARQL Protocol and the RDF Query Language (SPARQL) endpoint, reveals the practical benefits of this approach. Moreover, evaluation results indicate that the adapted version of DISO offers the most comprehensive and diverse knowledge representation among state-of-the-art ontologies.

PRIMA is a modular ontology developed to document the provenance information in an MSE study, improving data trustworthiness and reproducibility of MSE data. Following the NeOn methodology and re-engineering the MDMC-NEP Glos-

sary of Terms, PRIMA demonstrated its utility through real-world applications. Its modular design and alignment with other ontologies ensure extensibility and interoperability. By documenting provenance information in an interoperable format, PRIMA facilitates data quality assessment, providing valuable context and ensuring the reproducibility of scientific studies.

Preserving provenance information is essential for maintaining the quality and reliability of material designs. However, most of this information is found in unformatted and unstructured natural language text, such as materials science journal articles, books, and lab notebooks, which makes it difficult for machines to read and understand. Moreover, extracting the information on how materials are synthesized into a digital format, i.e., digitization, is often challenging. In this thesis, we demonstrated a text mining framework that uses several fine-tuned Large Language Models (LLMs) and semantic web technologies to extract provenance information from unstructured materials synthesis text. Fine-tuned LLMs, especially FLAN-T5 XXL, excelled in converting unstructured data into structured formats such as Java Script Object Notation (JSON). The development of the PRIMA Lite ontology aided in data transformation and querying via SPARQL, showcasing the framework's practical application. This method enhances data accessibility on materials, supporting more efficiency in designing and planning synthesis on novel materials.

Overall, we demonstrated the interdisciplinary approach combining the knowledge of the MSE domain, Semantic Web technologies, and Natural Language Processing (NLP). The approach enables us to represent the knowledge of dislocations and provenance information in MSE by means of ontology. The ontology is subsequently utilized to enrich the real-world data which are mostly in an unstructured data format, including simulation outputs from DDD, data generated from experiments, and text data. Additionally, we showcased a framework combining ontology and LLMs to automatically enrich the data, particularly the materials synthesis data written in a text format. This enrichment gives data a meaning, i.e., semantic value, which is not only machine readable but also interoperable with other machines. Furthermore, the enrichment has several advantages, including establishing links between domain-related datasets, enabling machines to discover new information, and supporting semantic querying via the SPARQL. We have now a "recipe" towards the digital transformation by enriching materials data and fostering data-driven future for the MSE domain through approaches such as text mining. Ultimately, the recipe improves the data management, analysis, and interoperability, ensuring that materials data is Findable, Accessible, Interoperable, Reusable (FAIR).

However, several challenges and limitations remain. The complexity of dislocation systems and the vast amount of data generated pose significant challenges in data management and analysis. Integrating disparate data sources requires continuous refinement of ontologies and data models. For instance, DISO could be further developed to model linear elasticity theory and aligned with other crystalline defect ontologies. Furthermore, its application could be expanded to more use cases, including Transmission Electron Microscopy (TEM) data and other DDD

simulation software. Moreover, developing an API for DisLocKG with features for querying, data mining, visualizing, updating, and deleting data will improve its usability. The knowledge graph could also be extended to include additional datasets and use cases.

PRIMA could be extended with a computational module and aligned with upper-level ontologies such as Basic Formal Ontology (BFO). It will enhance its utility by promoting interoperability with other related-domain ontologies, non-ontological resources, and Electronic Lab Notebook (ELN)s. The text mining framework could benchmark other LLMs and fine-tune more materials synthesis procedures. Exploring alternative strategies for leveraging LLMs to handle wider context windows without increasing GPU memory consumption is required. Investigating new architecture models like Mamba [176] for better performance in learning wider context windows will also be an important area of future direction.

We envision a future in which Semantic Web technologies and Artificial Intelligence (AI) work together with knowledge extraction and scientific research management. With the advancement of LLMs, text mining has become more sophisticated, enabling the extraction of valuable insights from unstructured data. However, a well-structured data infrastructure must be established for these models to operate effectively and produce reliable, interoperable, and reusable results. One promising approach is leveraging Semantic Web technologies, which offer standardized data representation, interoperability, and enhanced contextual reasoning through ontologies, linked data, and knowledge graphs. These technologies ensure that extracted information is machine-understandable, enabling intelligent querying, automated reasoning, and knowledge discovery. By integrating AI-driven text mining with semantic web technology, we can pave the way for the digital transformation in MSE, where machines extract and process information and understand and generate new knowledge, driving scientific progress and novel innovation.

Bibliography

[1] A. Prakash and S. Sandfeld, "Chances and challenges in fusing data science with materials science," *Practical Metallography*, vol. 55, no. 8, pp. 493–514, 2018. [Online]. Available: https://www.hanser-elibrary.com/doi/pdf/10.3139/147.110539

[2] N. Adamovic, J. Friis, G. Goldbeck, A. Hashibon, K. Hermansson, D. Hristova-Bogaerds, R. Koopmans, and E. Wimmer, "The EMMC roadmap for materials modelling and digitalisation of the materials sciences," Nov 2020.

[3] J. Kimmig, S. Zechel, and U. S. Schubert, "Digital transformation in materials science: A paradigm change in material's development," *Advanced Materials*, vol. 33, no. 8, p. 2004940, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202004940

[4] H. Koinuma and I. Takeuchi, "Combinatorial solid-state chemistry of inorganic materials," *Nature materials*, vol. 3, no. 7, pp. 429–438, 2004.

[5] S. Curtarolo, G. L. Hart, M. B. Nardelli, N. Mingo, S. Sanvito, and O. Levy, "The high-throughput highway to computational materials design," *Nature materials*, vol. 12, no. 3, pp. 191–201, 2013.

[6] E. O. Pyzer-Knapp, J. W. Pitera, P. W. Staar, S. Takeda, T. Laino, D. P. Sanders, J. Sexton, J. R. Smith, and A. Curioni, "Accelerating materials discovery using artificial intelligence, high performance computing and robotics," *npj Computational Materials*, vol. 8, no. 1, p. 84, 2022.

[7] C. C. Tasan, M. Diehl, D. Yan, M. Bechtold, F. Roters, L. Schemmann, C. Zheng, N. Peranio, D. Ponge, M. Koyama *et al.*, "An overview of dual-phase steels: advances in microstructure-oriented processing and micromechanically guided design," *Annual Review of Materials Research*, vol. 45, no. 1, pp. 391–431, 2015.

[8] A. Prakash, W. Nöhring, R. Lebensohn, H. Höppel, and E. Bitzek, "A multiscale simulation framework of the accumulative roll bonding process accounting for texture evolution," *Materials Science and Engineering: A*, vol. 631, pp. 104–119, 2015.

[9] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, no. 7604, 2016.

[10] U. Sivarajah, M. M. Kamal, Z. Irani, and V. Weerakkody, "Critical analysis of big data challenges and analytical methods," *Journal of business research*, vol. 70, pp. 263–286, 2017.

[11] T. Miyakawa, "No raw data, no science: another possible source of the reproducibility crisis," pp. 1–6, 2020.

[12] D. Eisner, "Reproducibility of science: Fraud, impact factors and carelessness," *Journal of molecular and cellular cardiology*, vol. 114, pp. 364–368, 2018.

[13] E. A. Olivetti, J. M. Cole, E. Kim, O. Kononova, G. Ceder, T. Y.-J. Han, and A. M. Hiszpanski, "Data-driven materials research enabled by natural language processing and information extraction," *Applied Physics Reviews*, vol. 7, no. 4, 2020.

[14] J. Frenkel, "The theory of the elastic limit and the solidity of crystal bodies," *Z Phys*, vol. 37, pp. 572–609, 1926.

[15] E. Orowan, "Zur Kristallplastizität." *Zeitschrift für Physik*, vol. 89, no. 9-10, pp. 605–613, 1934.

[16] M. Polanyi, "Über eine Art Gitterstörung, die einen Kristall plastisch machen könnte," *Zeitschrift für Physik*, vol. 89, no. 9-10, pp. 660–664, 1934.

[17] G. I. Taylor, "The mechanism of plastic deformation of crystals. part I.–Theoretical," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 145, no. 855, pp. 362–387, 1934.

[18] W. D. Callister and D. G. Rethwisch, *Materials science and engineering: an introduction*. Wiley New York, 2018, vol. 9.

[19] T. Murakumo, T. Kobayashi, Y. Koizumi, and H. Harada, "Creep behaviour of ni-base single-crystal superalloys with various γ' volume fraction," *Acta Materialia*, vol. 52, no. 12, pp. 3737–3744, 2004.

[20] R. Wu, M. Zaiser, and S. Sandfeld, "A continuum approach to combined γ/γ' evolution and dislocation plasticity in nickel-based superalloys," *International Journal of Plasticity*, vol. 95, pp. 142–162, 2017.

[21] M. Chen, E. Ma, K. J. Hemker, H. Sheng, Y. Wang, and X. Cheng, "Deformation twinning in nanocrystalline aluminum," *Science*, vol. 300, no. 5623, pp. 1275–1277, 2003.

[22] F. Mompiou, D. Caillard, M. Legros, and H. Mughrabi, "In situ TEM observations of reverse dislocation motion upon unloading in tensile-deformed ufg aluminium," *Acta Materialia*, vol. 60, no. 8, pp. 3402–3414, 2012.

[23] M. Stricker, M. Sudmanns, K. Schulz, T. Hochrainer, and D. Weygand, "Dislocation multiplication in stage ii deformation of fcc multi-slip single crystals," *Journal of the Mechanics and Physics of Solids*, vol. 119, pp. 319–333, 2018.

[24] S. Rao, C. Woodward, B. Akdim, E. Antillon, T. Parthasarathy, J. El-Awady, and D. Dimiduk, "Large-scale dislocation dynamics simulations of strain hardening of ni microcrystals under tensile loading," *Acta Materialia*, vol. 164, pp. 171–183, 2019.

[25] K. Govind, D. Oliveros, A. Dlouhy, M. Legros, and S. Sandfeld, "Deep learning of crystalline defects from TEM images: A solution for the problem of

"never enough training data"," *Machine Learning: Science and Technology*, 2023.

[26] N. Bertin and F. Zhou, "Accelerating discrete dislocation dynamics simulations with graph neural networks," *Journal of Computational Physics*, vol. 487, p. 112180, 2023.

[27] C. Zhang, H. Song, D. Oliveros, A. Fraczkiewicz, M. Legros, and S. Sandfeld, "Data-mining of in-situ tem experiments: On the dynamics of dislocations in cocrfemnni alloys," *Acta Materialia*, vol. 241, p. 118394, 2022.

[28] Z. Yang, S. Papanikolaou, A. C. Reid, W.-k. Liao, A. N. Choudhary, C. Campbell, and A. Agrawal, "Learning to predict crystal plasticity at the nanoscale: Deep residual networks and size effects in uniaxial compression discrete dislocation simulations," *Scientific reports*, vol. 10, no. 1, p. 8262, 2020.

[29] H. Song, N. Gunkelmann, G. Po, and S. Sandfeld, "Data-mining of dislocation microstructures: concepts for coarse-graining of internal energies," *Modelling and Simulation in Materials Science and Engineering*, vol. 29, no. 3, p. 035005, 2021.

[30] H. Salmenjoki, M. J. Alava, and L. Laurson, "Machine learning plastic deformation of crystals," *Nature communications*, vol. 9, no. 1, p. 5307, 2018.

[31] P. E. van der Vet, P.-H. Speel, and N. J. Mars, "The plinius ontology of ceramic materials," in *Eleventh European Conference on Artificial Intelligence (ECAI'94) Workshop on Comparison of Implemented Ontologies*, 1994, pp. 8–12.

[32] T. Ashino, "Materials ontology: An infrastructure for exchanging materials information and knowledge," *Data Science Journal*, vol. 9, pp. 54–61, 2010.

[33] J. F. Morgado, E. Ghedini, G. Goldbeck, A. Hashibon, G. J. Schmitz, J. Friis, and A. de Baas, "Mechanical testing ontology for digital-twins: A roadmap based on EMMO," *SeDiT 2020: Semantic Digital Twins 2020*, p. 3, 2020.

[34] M. T. Horsch, S. Chiacchiera, Y. Bami, G. J. Schmitz, G. Mogni, G. Goldbeck, and E. Ghedini, "Reliable and interoperable computational molecular engineering: 2. semantic interoperability based on the european materials and modelling ontology," *arXiv preprint arXiv:2001.04175*, 2020.

[35] K. Cheung, J. Drennan, and J. Hunter, "Towards an ontology for data-driven discovery of new materials." in *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*, 2008, pp. 9–14.

[36] S. R. Hall and B. McMahon, *International tables for crystallography, definition and exchange of crystallographic data.* Springer Science & Business Media, 2005, vol. 8.

[37] ——, "The implementation and evolution of STAR/CIF ontologies: Interoperability and preservation of structured data," *Data Science Journal*, vol. 15, 2016.

[38] N. Spadaccini, I. R. Castleden, D. du Boulay, and S. R. Hall, "dREL: a relational expression language for dictionary methods," *Journal of chemical information and modeling*, vol. 52, no. 8, pp. 1917–1925, 2012.

[39] H. Li, R. Armiento, and P. Lambrix, "An ontology for the materials design domain," in *International Semantic Web Conference*. Springer, 2020, pp. 212–227.

[40] A. Say, S. Fathalla, S. Vahdati, J. Lehmann, and S. Auer, "Semantic representation of physics research data," in *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management Vol. 2*. Setúbal, Portugal: Science and Technology Publications, Lda, 2020.

[41] S. Hu, H. Wang, C. She, and J. Wang, "AgOnt: ontology for agriculture internet of things," in *Computer and Computing Technologies in Agriculture IV: 4th IFIP TC 12 Conference, CCTA 2010, Nanchang, China, October 22-25, 2010, Selected Papers, Part I 4*. Springer, 2011, pp. 131–137.

[42] Z. Say, S. Fathalla, S. Vahdati, J. Lehmann, and S. Auer, "Ontology design for pharmaceutical research outcomes," in *International Conference on Theory and Practice of Digital Libraries*. Springer, 2020, pp. 119–132.

[43] D. Mrdjenovich, M. K. Horton, J. H. Montoya, C. M. Legaspi, S. Dwaraknath, V. Tshitoyan, A. Jain, and K. A. Persson, "Propnet: a knowledge graph for materials science," *Matter*, vol. 2, no. 2, pp. 464–480, 2020.

[44] X. Zhao, J. Greenberg, S. McClellan, Y.-J. Hu, S. Lopez, S. K. Saikin, X. Hu, and Y. An, "Knowledge graph-empowered materials discovery," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 4628–4632.

[45] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. a. Persson, "The Materials Project: A materials genome approach to accelerating materials innovation," *APL Materials*, vol. 1, no. 1, p. 011002, 2013. [Online]. Available: http://link.aip.org/link/AMPADS/v1/i1/p011002/s1&Agg=doi

[46] J. P. McCusker, N. Keshan, S. Rashid, M. Deagen, C. Brinson, and D. L. McGuinness, "NanoMine: A knowledge graph for nanocomposite materials science," in *International Semantic Web Conference*. Springer, 2020, pp. 144–159.

[47] O. S. Collaboration, "Estimating the reproducibility of psychological science," *Science*, vol. 349, no. 6251, p. aac4716, 2015.

[48] F. Prinz, T. Schlange, and K. Asadullah, "Believe it or not: how much can we rely on published data on potential drug targets?" *Nature reviews Drug discovery*, vol. 10, no. 9, pp. 712–712, 2011.

[49] J. Kaiser, "The cancer test," 2015.

[50] A. Stupple, D. Singerman, and L. A. Celi, "The reproducibility crisis in the age of digital medicine," *NPJ digital medicine*, vol. 2, no. 1, p. 2, 2019.

[51] M. Hutson, "Artificial intelligence faces reproducibility crisis," 2018.

[52] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Larivière, A. Beygelzimer, F. d'Alché Buc, E. Fox, and H. Larochelle, "Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program)," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 7459–7478, 2021.

[53] S. Kapoor and A. Narayanan, "Leakage and the reproducibility crisis in ml-based science," *arXiv preprint arXiv:2207.07048*, 2022.

[54] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[55] P. L. Buttigieg, C. Curdt, A. Z. Ihsan, T. Jejkal, M. Kubin, O. Mannix, D. P. Mohr, A. Pirogov, B. Port, and K.-U. Stucky, "An interpretation of the fair principles to guide implementations in the hmc digital ecosystem," 2022.

[56] S. Samuel and B. König-Ries, "REPRODUCE-ME: Ontology-based data access for reproducibility of microscopy experiments," in *The Semantic Web: ESWC 2017 Satellite Events: ESWC 2017 Satellite Events, Portorož, Slovenia, May 28–June 1, 2017, Revised Selected Papers 14*. Springer, 2017, pp. 17–20.

[57] L. P. Freedman, G. Venugopalan, and R. Wisman, "Reproducibility2020: progress and priorities," *F1000Research*, vol. 6, 2017.

[58] L. Moreau, P. Groth, J. Cheney, T. Lebo, and S. Miles, "The rationale of PROV," *Journal of Web Semantics*, vol. 35, pp. 235–257, 2015.

[59] R. Joseph, A. Chauhan, C. Eschke, A. Ihsan, M. Jalali, U. Jäntsch, N. Jung, C. Shyam Kumar, C. Kübel, C. Lucas *et al.*, "Metadata schema to support fair data in scanning electron microscopy," in *Supplementary Proceedings of the XXIII International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2021): Moscow, Russia, October 26-29, 2021. Ed.: A. Pozanenko*, 2021, p. 265.

[60] L. N. Soldatova and R. D. King, "An ontology of scientific experiments," *Journal of the royal society interface*, vol. 3, no. 11, pp. 795–803, 2006.

[61] P. Ciccarese, E. Wu, G. Wong, M. Ocana, J. Kinoshita, A. Ruttenberg, and T. Clark, "The SWAN biomedical discourse ontology," *Journal of biomedical informatics*, vol. 41, no. 5, pp. 739–751, 2008.

[62] J. Kinoshita, G. T. Wong, E. Wu, P. Ciccarese, M. Ocana, and T. Clark, "AlzSWAN: knowledge base for alzheimer research," *Alzheimer's & Dementia*, vol. 6, pp. S387–S387, 2010.

[63] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers *et al.*, "The open provenance model core specification (v1. 1)," *Future generation computer systems*, vol. 27, no. 6, pp. 743–756, 2011.

[64] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker *et al.*, "PROV-DM: The prov data model," *W3C Recommendation*, vol. 14, pp. 15–16, 2013.

[65] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao, "PROV-O: The PROV ontology. w3c recommendation," *World Wide Web Consortium*, 2013.

[66] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein *et al.*, "OWL web ontology language reference," *W3C recommendation*, vol. 10, no. 2, pp. 1–53, 2004.

[67] P. Ciccarese, S. Soiland-Reyes, K. Belhajjame, A. J. Gray, C. Goble, and T. Clark, "PAV ontology: provenance, authoring and versioning," *Journal of biomedical semantics*, vol. 4, pp. 1–22, 2013.

[68] D. Garijo and Y. Gil, "Augmenting PROV with plans in P-Plan: scientific processes as linked data," CEUR Workshop Proceedings, 2012.

[69] B. Bayerlein, M. Schilling, H. Birkholz, M. Jung, J. Waitelonis, L. Mädler, and H. Sack, "PMD Core Ontology: Achieving semantic interoperability in materials science," *Materials & Design*, vol. 237, p. 112603, 2024.

[70] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with Basic Formal Ontology (BFO)*. Mit Press, 2015.

[71] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, "Sweetening ontologies with DOLCE," in *International conference on knowledge engineering and knowledge management*. Springer, 2002, pp. 166–181.

[72] E. Kim, K. Huang, A. Tomala, S. Matthews, E. Strubell, A. Saunders, A. McCallum, and E. Olivetti, "Machine-learned and codified synthesis parameters of oxide materials," *Scientific data*, vol. 4, no. 1, pp. 1–9, 2017.

[73] H. Huo, Z. Rong, O. Kononova, W. Sun, T. Botari, T. He, V. Tshitoyan, and G. Ceder, "Semi-supervised machine-learning classification of materials synthesis procedures," *Npj Computational Materials*, vol. 5, no. 1, p. 62, 2019.

[74] H. Kamila, A. Sankhla, M. Yasseri, N. Hoang, N. Farahi, E. Mueller, and J. de Boor, "Synthesis of p-type mg2si1-xsnx with x= 0-1 and optimization of the synthesis parameters," *Materials Today: Proceedings*, vol. 8, pp. 546–555, 2019.

[75] A. M. Cohen and W. R. Hersh, "A survey of current work in biomedical text mining," *Briefings in bioinformatics*, vol. 6, no. 1, pp. 57–71, 2005.

[76] L. Rasmy, Y. Xiang, Z. Xie, C. Tao, and D. Zhi, "Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction," *NPJ digital medicine*, vol. 4, no. 1, p. 86, 2021.

[77] R. Leaman, C.-H. Wei, and Z. Lu, "tmChem: a high performance approach for chemical named entity recognition and normalization," *Journal of cheminformatics*, vol. 7, no. 1, pp. 1–10, 2015.

[78] M. C. Swain and J. M. Cole, "ChemDataExtractor: a toolkit for automated extraction of chemical information from the scientific literature," *Journal of chemical information and modeling*, vol. 56, no. 10, pp. 1894–1904, 2016.

[79] S. Horawalavithana, E. Ayton, S. Sharma, S. Howland, M. Subramanian, S. Vasquez, R. Cosbey, M. Glenski, and S. Volkova, "Foundation models of scientific knowledge for chemistry: Opportunities, challenges and lessons learned," in *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022, pp. 160–172.

[80] S. Mysore, E. Kim, E. Strubell, A. Liu, H.-S. Chang, S. Kompella, K. Huang, A. McCallum, and E. Olivetti, "Automatically extracting action graphs from materials science synthesis procedures," *arXiv preprint arXiv:1711.06872*, 2017.

[81] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[82] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[83] A. Dunn, J. Dagdelen, N. Walker, S. Lee, A. S. Rosen, G. Ceder, K. Persson, and A. Jain, "Structured information extraction from complex scientific text with fine-tuned large language models," *arXiv preprint arXiv:2212.05238*, 2022.

[84] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[85] R. Aversa, A. Boubnov, D. De Angelis, C. Eschke, S. Irvine, R. E. Joseph, M. Kabbe, N. MacKinnon, M. Irene, M. Panighel *et al.*, "The MDMC-NEP glossary of terms," 2024. [Online]. Available: https://zenodo.org/records/10663833

[86] N. M. Ghoniem and L. Sun, "Fast-sum method for the elastic field of three-dimensional dislocation ensembles," *Physical Review B*, vol. 60, no. 1, p. 128, 1999.

[87] D. Oliveros, A. Fraczkiewicz, A. Dlouhy, C. Zhang, H. Song, S. Sandfeld, and M. Legros, "Orientation-related twinning and dislocation glide in a cantor high entropy alloy at room and cryogenic temperature studied by in situ tem straining," *Materials Chemistry and Physics*, vol. 272, p. 124955, 2021.

[88] A. Haloi, *Ultramicrotome Device and Ultramicrotomy*. Singapore: Springer Nature Singapore, 2024, pp. 83–97. [Online]. Available: https://doi.org/10.1007/978-981-97-4791-7_4

[89] Leica Microsystems, "Introduction to ultramicrotomy," 2024, accessed: 2025-02-01. [Online]. Available: https://www.leica-microsystems.com/science-lab/life-science/introduction-to-ultramicrotomy/

[90] CondenZero, "Liquid helium tem sample holders," 2024, accessed: 2025-02-01. [Online]. Available: https://condenzero.com/en/products/liquid-helium-tem-sample-holders

[91] ER-C Research Center, "Fei titan g3 50-300 pico," 2024, accessed: 2025-02-01. [Online]. Available: https://er-c.org/index.php/facilities-2/facilities-material-science/material-science/fei-titan-g3-50-300-pico/

[92] N. F. Noy, D. L. McGuinness *et al.*, "Ontology development 101: A guide to creating your first ontology," 2001.

[93] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, "The NeOn methodology for ontology engineering," in *Ontology engineering in a networked world*. Springer, 2011, pp. 9–34.

[94] T. Berners-Lee, J. Hendler, and O. Lassila, "Scientific american: Feature article: The semantic web: May 2001," *Scientific American*, vol. 4, 2001.

[95] N. Shadbolt, T. Berners-Lee, and W. Hall, "The semantic web revisited," *IEEE intelligent systems*, vol. 21, no. 3, pp. 96–101, 2006.

[96] A. Hogan and A. Hogan, *Web of data*. Springer, 2020.

[97] W. W. W. Consortium *et al.*, "RDF 1.1 primer," 2014. [Online]. Available: https://www.w3.org/TR/rdf11-primer/

[98] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, and B. McBride, "RDF 1.1 concepts and abstract syntax," *W3C recommendation*, vol. 25, no. 02, pp. 1–22, 2014.

[99] M. Arenas, C. Gutierrez, and J. Pérez, "Foundations of RDF databases," in *Reasoning Web International Summer School*. Springer, 2009, pp. 158–204.

[100] D. Beckett and B. McBride., "RDF/XML syntax specification (revised). w3c recommendation," 2004. [Online]. Available: https://www.w3.org/TR/rdf-syntax-grammar/

[101] D. Beckett, G. Carothers, and A. Seaborne., "RDF 1.1 n-triples – a line-based syntax for an RDF graph. W3C recommendation„" 2014. [Online]. Available: https://www.w3.org/TR/n-triples/

[102] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, and G. Carothers, "RDF 1.1 turtle – terse RDF triple language. w3c recommendation," 2014. [Online]. Available: https://www.w3.org/TR/turtle/

[103] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström, "JSON-ld 1.0 – a JSON-based serialization for linked data. W3C recommendation," 2014. [Online]. Available: https://www.w3.org/TR/json-ld/

[104] G. Antoniou and F. Van Harmelen, *A semantic web primer*. MIT press, 2004.

[105] J. H. Alexander, M. J. Freiling, S. Shulman, J. Staley, S. Rehfuss, and S. Messick, "Knowledge level engineering ontological analysis." in *AAAI*, 1986, pp. 963–968.

[106] G. O. Consortium, "The gene ontology (GO) database and informatics resource," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D258–D261, 2004.

[107] A. Z. Ihsan, S. Fathalla, and S. Sandfeld, "DISO: A domain ontology for modeling dislocations in crystalline materials," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 2023, pp. 1746–1753.

[108] P. L. Buttigieg, N. Morrison, B. Smith, C. J. Mungall, S. E. Lewis, and E. Consortium, "The environment ontology: contextualising biological and biomedical entities," *Journal of biomedical semantics*, vol. 4, pp. 1–9, 2013.

[109] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström, "RDF schema 1.1 W3C recommendation," 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema/

[110] S. Harris, A. Seaborne, and E. Prud'hommeaux, "SPARQL 1.1 query language," 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/

[111] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, "Ontological engineering. advanced information and knowledge processing," 2003.

[112] C. M. Keet, "An introduction to ontology engineering," 2018.

[113] E. Simperl, "Reusing ontologies on the semantic web: A feasibility study," *Data & Knowledge Engineering*, vol. 68, no. 10, pp. 905–925, 2009.

[114] A. Gangemi, "Ontology design patterns for semantic web content," in *International semantic web conference*. Springer, 2005, pp. 262–276.

[115] H. Hapke, C. Howard, and H. Lane, *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Simon and Schuster, 2019.

[116] D. Altinok, *Mastering spaCy: An end-to-end practical guide to implementing NLP applications using the Python ecosystem*. Packt Publishing Ltd, 2021.

[117] Explosion AI, "Spacy: Industrial-strength natural language processing in python," 2024, accessed: 2024-04-03. [Online]. Available: https://spacy.io

[118] L. Tunstall, L. Von Werra, and T. Wolf, *Natural language processing with transformers*. " O'Reilly Media, Inc.", 2022.

[119] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 5149–5152.

[120] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[121] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[122] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Advances in neural information processing systems*, vol. 13, 2000.

[123] C. M. Bishop and H. Bishop, *Deep learning: Foundations and concepts*. Springer Nature, 2023.

[124] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[125] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[126] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed., 2024, online manuscript released August 20, 2024. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[127] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[128] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[129] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[130] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[131] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, and G. Penedo, "Falcon-40B: an open large language model with state-of-the-art performance," 2023.

[132] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[133] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[134] S. Fathalla, S. Vahdati, C. Lange, and S. Auer, "SEO: A scientific events data model," in *The Semantic Web – ISWC 2019*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, Eds. Cham: Springer International Publishing, 2019, pp. 79–95.

[135] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.

[136] E. Simperl, C. Sarasua, R. Ungrangsi, and T. Bürger, "Ontology metadata for ontology reuse," *International Journal of Metadata, Semantics and Ontologies*, vol. 6, no. 2, pp. 126–145, 2011. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJMSO.2011.046579

[137] M. Fernández-López, M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez, "Why are ontologies not reused across the same domain?" *Journal of Web Semantics*, vol. 57, p. 100492, 2019.

[138] R. Hodgson, P. J. Keller, J. Hodges, and J. Spivak, "QUDT-quantities, units, dimensions and data types ontologies," *USA Available http://qudt. org March*, vol. 156, 2014.

[139] S. Fathalla, C. Lange, and S. Auer, "An upper ontology for modern science branches and related entities," in *European Semantic Web Conference*. Springer, 2023, pp. 436–453.

[140] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza, "OntoQA: Metric-based ontology quality analysis," 2005.

[141] Materials Data Science and Informatics Group, "DISO modelib nickel microstructure," 2024, accessed: 2024-04-20. [Online]. Available: https://github.com/Materials-Data-Science-and-Informatics/Dislocation-Ontology-Suite/blob/main/DISO/data/modelib-microstructure/modelib-nickel-microstructure.h5

[142] ——, "DISO python script," 2024, accessed: 2024-04-20. [Online]. Available: https://github.com/Materials-Data-Science-and-Informatics/Dislocation-Ontology-Suite/tree/main/DISO/python-script/modelib

[143] C. Boettiger, *RDFLib: A high level wrapper around the redland package for common rdf applications*, 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1098478

[144] G. Po and N. Ghoniem, "A variational formulation of constrained dislocation dynamics coupled with heat and vacancy diffusion," *Journal of the Mechanics and Physics of Solids*, vol. 66, pp. 103–116, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022509614000222

[145] Materials Data Science and Informatics Group, "DISO github repository," 2024, accessed: 2024-04-20. [Online]. Available: https://github.com/Materials-Data-Science-and-Informatics/Dislocation-Ontology-Suite/tree/main/DISO

[146] S. Fathalla, S. Vahdati, S. Auer, and C. Lange, "SemSur: A core ontology for the semantic representation of research findings," in *Proceedings of the 14th International Conference on Semantic Systems, SEMANTICS 2018, Vienna, Austria, September 10-13, 2018*, ser. Procedia Computer Science, vol. 137. Elsevier, 2018, pp. 151–162. [Online]. Available: https://doi.org/10.1016/j.procs.2018.09.015

[147] S. Fathalla, C. Lange, and S. Auer, "EVENTSKG: A 5-star dataset of top-ranked events in eight computer science communities," in *European semantic web conference*. Springer, 2019, pp. 427–442.

[148] M. Ehrig, *Ontology alignment: bridging the semantic gap*. Springer Science & Business Media, 2006, vol. 4.

[149] N. F. Noy, M. A. Musen *et al.*, "Algorithm and tool for automated ontology merging and alignment," 2000.

[150] A. Arsenlis, W. Cai, M. Tang, M. Rhee, T. Oppelstrup, G. Hommes, T. G. Pierce, and V. V. Bulatov, "Enabling strain hardening simulations with dislocation dynamics," *Modelling and Simulation in Materials Science and Engineering*, vol. 15, no. 6, p. 553, 2007.

[151] B. Devincre, R. Madec, G. Monnet, S. Queyreau, R. Gatti, and L. Kubin, "Modeling crystal plasticity with dislocation dynamics simulations: The 'micromegas' code," *Mechanics of Nano-objects*, vol. 1, pp. 81–100, 2011.

[152] A. Demirci, D. Steinberger, M. Stricker, N. Merkert, D. Weygand, and S. Sandfeld, "Statistical analysis of discrete dislocation dynamics simulations: initial structures, cross-slip and microstructure evolution," *Modelling and Simulation in Materials Science and Engineering*, 2023.

[153] C. Motz, D. Weygand, J. Senger, and P. Gumbsch, "Initial dislocation structures in 3-d discrete dislocation dynamics and their influence on microscale plasticity," *Acta Materialia*, vol. 57, no. 6, pp. 1744–1754, 2009.

[154] H. Fan, Q. Wang, J. A. El-Awady, D. Raabe, and M. Zaiser, "Strain rate dependency of dislocation plasticity," *Nature communications*, vol. 12, no. 1, p. 1845, 2021.

[155] D. Steinberger, H. Song, and S. Sandfeld, "Machine learning-based classification of dislocation microstructures," *Frontiers in Materials*, vol. 6, p. 141, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fmats.2019.00141

[156] J. F. Sowa, *Knowledge representation: logical, philosophical and computational foundations*. Brooks/Cole Publishing Co., 1999.

[157] K. I. Kotis, G. A. Vouros, and D. Spiliotopoulos, "Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations," *The Knowledge Engineering Review*, vol. 35, p. e4, 2020.

[158] M. Jalali, M. Mail, R. Aversa, and C. Kübel, "MSLE: An ontology for materials science laboratory equipment–large-scale devices for materials characterization," *Materials Today Communications*, vol. 35, p. 105532, 2023.

[159] C. Shimizu, Q. Hirt, and P. Hitzler, "MODL: a modular ontology design library," *arXiv preprint arXiv:1904.05405*, 2019.

[160] E. Simperl, C. Sarasua, R. Ungrangsi, and T. Bürger, "Ontology metadata for ontology reuse," *International Journal of Metadata, Semantics and Ontologies*, vol. 6, no. 2, pp. 126–145, 2011.

[161] T. Rodani, E. Osmenaj, A. Cazzaniga, M. Panighel, C. Africh, and S. Cozzini, "Towards the FAIRification of scanning tunneling microscopy images," *Data Intelligence*, vol. 5, no. 1, pp. 27–42, 2023.

[162] F. Kirchner, C. Eschke, A.-L. Höhme, M. Meller, A. Foremny, M. Held, S. A. Sahim, and R. Willumeit-Römer, "Herbie - The Semantic Laboratory Notebook & Research Database," Jun. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.12205430

[163] D. Plinere and A. Borisov, "SWRL: Rule acquisition using ontology." *Computer Science (1407-7493)*, vol. 40, 2009.

[164] M. Proctor, "Drools: a rule engine for complex event processing," in *Applications of Graph Transformations with Industrial Relevance: 4th International Symposium, AGTIVE 2011, Budapest, Hungary, October 4-7, 2011, Revised Selected and Invited Papers 4*. Springer, 2012, pp. 2–2.

[165] S. Mysore, Z. Jensen, E. Kim, K. Huang, H.-S. Chang, E. Strubell, J. Flanigan, A. McCallum, and E. Olivetti, "The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures," in *Proceedings of the 13th Linguistic Annotation Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 56–64. [Online]. Available: https://www.aclweb.org/anthology/W19-4007

[166] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, "BRAT: a web-based tool for nlp-assisted text annotation," in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012, pp. 102–107.

[167] B. Rahmati, J. Fleig, W. Sigle, E. Bischoff, J. Maier, and M. Rühle, "Oxidation of reduced polycrystalline nb-doped srtio3: Characterization of surface islands," *Surface science*, vol. 595, no. 1-3, pp. 115–126, 2005.

[168] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *arXiv preprint arXiv:2210.11416*, 2022.

[169] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, S. Zheng *et al.*, "UL2: Unifying language learning paradigms," in *The Eleventh International Conference on Learning Representations*, 2022.

[170] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[171] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[172] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.

[173] A. Z. Ihsan, S. Fathalla, and S. Sandfeld, "Modeling dislocation dynamics data using semantic web technologies," *arXiv preprint arXiv:2309.06930*, 2023.

[174] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement," *Acm Sigkdd Explorations Newsletter*, vol. 12, no. 1, pp. 49–57, 2010.

[175] Materials Data Science and Informatics Group, "CQs of PRIMA LM," 2024, accessed: 2024-05-01. [Online]. Available: https://github.com/Materials-Data-Science-and-Informatics/prima-lm/blob/main/ontology/CQs/CQs.md

[176] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.