Qualitative Coding in the Age of AI: An Ontology-Driven Approach

Daniil Dobriy[1], Axel Polleres[1]

[1]Vienna University of Economics and Business, Austria

Abstract

Qualitative coding is an essential methodological tool in qualitative research. Although various tools exist to support manual qualitative coding, the process remains highly resource-intensive, requiring significant time and expertise. It is further complicated by inconsistent reporting of coding protocols, differing interpretations of inter-coder reliability metrics, and difficulties in achieving conceptual agreement among coders – all of which contribute to the broader replication crisis in science. Meanwhile, large language models have demonstrated remarkable abilities to understand context across diverse tasks and are increasingly applied to information and, in combination with semantic web technologies, knowledge extraction. In this work, we define qualitative coding process as a knowledge base construction task and propose an ontology-mediated approach to automating qualitative coding, formalising key aspects of the methodology and reliability assessment using semantic web standards. We evaluate the reliability of this approach through a concrete implementation and case study, finding that qualitative coding can benefit substantially from AI-driven automation – especially when the underlying coding ontology is well-defined and domain-relevant constraints are explicitly codified.

Keywords

Qualitative Coding, Ontology Engineering, Knowledge Graph Construction, Large Language Models

1. Introduction

Qualitative coding is an essential methodological tool of qualitative research in which codes systematically assign descriptors to portions of natural language text or visual data [1]. More concisely, it is "the simple operation of identifying segments of meaning in your data and labelling them with a code," [2] where a code is "a word or short phrase that symbolically assigns a summative, salient, essence-capturing, and/or evocative attribute for a portion of language-based or visual data" [3]. The methodology is applied to research data sources, such as interviews and other text excerpts, visual data, audio recordings, and other unstructured data, to both create various degrees of structured data for further qualitative and quantitative content analysis, and as a stand-alone method for theory-building, notably in classical [1, 4] and constructivist [5] Grounded Theory.

Whenever qualitative coding is intended beyond simple feature extraction, the coding process is applied iteratively. With each iteration, codes are refined in various ways: redefined, modified, combined, split, reassigned etc. Through multiple iterations, a better understanding of the underlying data is sought after, which includes uncovering potential connections and insights. Each coding step can be deductive (predefined codes or categories are applied to text, in a top-down approach) as well as inductive (codes are "learned" from text, in a bottom-up approach), and often researchers employ both approaches interchangeably in the coding process [6].

Despite advances in supporting software tools (see Section 2.2), including qualitative data analysis systems (QDAS), qualitative coding remains heavily dependent on manual labour. The manual subtasks of qualitative coding demand substantial resources in terms of time and skilled personnel. The time required for data analysis is a primary issue in qualitative coding, as conceptual tasks associated with qualitative analysis cannot be easily expedited [7]. Furthermore, besides the methodology being particularly daunting for novice researchers, with available training resources less pervasive than

Joint proceedings of KBC-LM and LM-KBC @ ISWC 2025

daniil.dobriy@wu.ac.at (D. Dobriy); axel.polleres@wu.ac.at (A. Polleres)

https://dobriy.org (D. Dobriy); http://polleres.net (A. Polleres)

0000-0001-5242-302X (D. Dobriy); 0000-0001-5670-1146 (A. Polleres)

for quantitative methods [8], the requirement for multiple coders to independently process the same material – a practice necessary for ensuring reliability – significantly multiplies the resource intensity [9].

Another challenge associated with qualitative research and qualitative coding studies is their reproducibility. Though qualitative methods often do not easily subject themselves to replication, aiming at the objectivity of research and striving towards reproducibility shall be nonetheless attempted [10]. In this context, inter-coder reliability (ICR) is considered a good practice in qualitative research [11]. For example, in health education and behaviour research, double coding of all transcripts was the most common coding method, employed in 47.9% of qualitative coding studies, highlighting the broad acceptance of the collaborative methodology [12]. However, achieving a high ICR score becomes ever more challenging as the conceptual complexity of codes increases [13].

Besides the practical difficulties of achieving conceptual agreement, coding protocols or essential elements of the process are occasionally simply not reported [11], as for example, 31.3% of qualitative coding studies examined in health education and behaviour research failed to clearly describe the coding approach altogether [12]. Even when reported, authors often have different interpretations of similar ICR metrics, and the percentage of data used in ICR tests varies across studies [14]. Thus, the issues associated with reproducibility in qualitative coding research contribute to the overall replication crisis [15, 13], most prominently in disciplines relying on qualitative methodologies, such as psychology [16], social sciences [17], healthcare [12] etc. These issues have motivated the standardisation of qualitative coding practices [18], including the development of coding reliability frameworks, consensus-based approaches and collaborative protocols.

Recently, pre-trained large language models (LLMs) have demonstrated impressive emergent capabilities, especially in understanding context across different tasks and scenarios. In natural language processing (NLP), LLMs have achieved state-of-the-art performance across many tasks, becoming the de facto baseline methods [19]. Accordingly, LLMs are increasingly used for information and knowledge extraction tasks in various domains [20, 21, 22, 23]. They have also been proposed for document information extraction from visually-rich documents [24], scientific texts [25] and proceedings [26, 27] and medical descriptions [28]. In this context, the extraction process commonly employs carefully crafted prompts to specify both the desired information to be extracted and its structural representation [29, 30].

Complementing information extraction, semantic web [31] provides a universal framework for knowledge representation, allowing the publication of interconnected data on the web and enabling machines to process the explicit semantics of data through the use (most notably, reuse) of ontologies, which, in turn, enable reasoning [32]. Also notable is the synergetic application of semantic web in machine learning [33], where ontologies encode semantics and enable grounding of machine learning methods and their outputs. Semantic web technologies are especially effective in reducing hallucinations in LLMs [34, 35, 36, 37], for example, though knowledge injection in prompts [38, 39, 40, 41, 42, 43, 44] and outputs [45].

This research paper aims to build upon the recent advances in combining LLMs with semantic web technologies, and defines qualitative coding from the ontology engineering perspective, proposing an ontology-mediated approach for qualitative coding automation. The approach formalises parts of the qualitative coding methodology and reliability assessment approaches using semantic web standards.

This work is structured as follows: Section 2 surveys prior work on AI-assisted qualitative coding and existing tools; Section 3 introduces the case study used for evaluation; Section 4 details our ontology-mediated approach, including the formalisation of qualitative coding with semantic web standards, the design of domain-specific and linking ontologies, and the automated workflow; Section 5 presents the *AutoCod* implementation; Section 6 outlines the evaluation design, and Section 7 reports results for both deductive and inductive experiments; Section 8 interprets the findings within qualitative methodology and ontology engineering, noting limitations and implications; and Section 9 summarises the contributions and sketches directions for future work.

2. Related Work

This section reviews prior work relevant to the automation of qualitative coding and the software tools that support it. We first discuss approaches to automating qualitative coding, including both traditional rule-based methods and recent advances leveraging large language models (Section 2.1). We then provide an overview of widely used qualitative coding tools together with an analysis of their features (Section 2.2).

2.1. Qualitative Coding Automation

A number of studies have investigated automated approaches to qualitative coding, particularly for the initial deductive phase in which a predefined codebook is applied to the data. The majority of existing tools implement some form of keyword or regular expression matching, typically supplemented by human validation or human-in-the-loop designs to ensure accuracy. In general, most studies highlight the importance of the combination of automated and manual coding, as well as the need for transparency and interpretability of the coding process. Tietz et al. [46] presented *refer*, a WordPress-based tool for semantic annotation, which combines automatic named entity linking and manual annotation, concluding that the combination of both automated and manual annotation achieve best results.

Rietz and Maedche [47] present *Cody*, an interactive system that combines editable code rules (keyword matching with boolean operators) with supervised ML to extend manual codes to seen and unseen data. The authors report improvements in ICR to simple keyword matching, and note that coding rules provide structure and transparency. Cai et al. [48] propose nCoder+, a widely-used tool for automatic coding of large datasets. The tool relies on regular expressions and manual validation to establish reliability of codes. Marathe and Toyama [49] conducted a user-centred study of qualitative coding practices, noting potential for automation after the codebook has been created and highlighting transparency requirements for automatic coding tools. They then built a prototype for first-pass semi-automatic coding using simple NLP methods (keyword matching with boolean operators), and demonstrated high agreement with human coders, noting the potential of advanced NLP techniques to improve on the result. However, a notable limitation of such tools is their reliance on predefined keywords or rules, which are infeasible for the coding of complex, under-represented or counter-intuitive concepts. Furthermore, such an approach limits the set of codes to predefined or learned tags, whereby more complex semi-structured descriptions (e.g., features, relationships) could be required.

Recently, studies have employed LLMs for content analysis in specific domains and identified their potential to assist in creating coding schemes [50] or in-vivo categories [51]. Other coding automation approaches have focused on thematic analysis, as e.g., Feinerer and Wild [52], Lennon et al. [53] and Bryda and Sadowski [54].

2.2. Qualitative Coding Tools

A variety of dedicated software tools have been developed to facilitate qualitative coding and qualitative data analysis (QDAS). Review studies have analysed the use of such tools in qualitative coding in the healthcare (incl. medical education) domain [55, 9]. Table 1 gives an overview of those tools, their use in studies, and their basic features, including the *coding* functionality (i.e., assigning codes) and code *aggregation*, *search* and *visualisation* functions, automatic *transcription*, *collaborative features*, support for *multilingual* annotation, *statistical analysis* as well as *keyword-based autocoding* (automatic code assignment) capabilities.

The majority of these tools provide core functionality for supporting manual qualitative coding processes, including document search, aggregation, and coding capabilities. Most tools also offer data visualisation features and collaborative coding support. Additional features vary across platforms: some tools include transcription functionality and multilingual support, while statistical analysis capabilities are limited to MAXQDA (and underutilised). Keyword-based autocoding functionality is available only in NVivo, Atlas.ti and MAXQDA. Thus, existing tools primarily streamline manual coding workflows

and facilitate collaboration, with the conceptual and interpretive aspects of qualitative coding remaining the responsibility of human researchers and coders.

		Capabilities [9]								
Tool [9, 55]	**Use, %** [55]	Code	Aggregate	Search	Visualise	Transcribe	Collaborate	Multilang.	Stat. analysis	Keyword autoc.
ATLAS.ti	29.2	✓	✓	✓	✓	✓	✓	✓		✓
NVivo	20.8	✓	✓	✓	✓	✓	✓			✓
Ethnograph	4.2	–	–	–	–	–	–	–	–	–
MAXQDA	0.0	✓	✓	✓	✓	✓	✓	✓	✓	✓
DeDoose	0.0	✓	✓	✓	✓		✓			
QDA Miner Lite	NS	✓	✓	✓						
Quirkos	NS	✓	✓	✓	✓		✓			

Table 1
Overview of software tools for qualitative coding. Usage percentages based on analysed studies (n=48) [55]. NS = Not specified in usage analysis. Checkmark indicates feature availability [9]; dash (–) indicates no data available.

3. Use Case Study

To illustrate and evaluate our methodology, we apply it to a published qualitative coding study by van Gend and Zuiderwijk [56], which employs both deductive and inductive coding phases. In this regard, we note that any mischaracterisation, omission, or misinterpretation of the original study methodology or findings remains the sole responsibility of the present authors, and maintain that the chosen study is an exemplary application of the qualitative coding methodology motivated by open research data sharing and reuse.

Van Gend and Zuiderwijk conduct a qualitative single-case study at TU Delft to examine how combinations of institutional and infrastructural arrangements affect open research data sharing and reuse in context. Their contributions are twofold [56]: (1) providing a contextualised overview of institutional and infrastructural arrangements that stimulate open research data sharing and reuse; and (2) discussing the potential impact of implementing certain arrangements in the case of a Dutch university pursuing open science.

As part of the study, the authors collect seven semi-structured interviews. The interviews are fully transcribed, anonymised, member-checked, and qualitatively coded in ATLAS.ti, with Figure 1 illustrating an interview transcript coded using the tool.

The case study implements the qualitative coding process in the following way: first, it applies a theory-driven codebook derived from prior literature in a theory-driven approach [57], followed by open coding [58] (creating initial categories), axial coding [59, 60] (relating categories into themes, specifying properties and dimensions), and focused coding [59] (refining salient codes), distinguishing the codes by actor type (researchers vs. policymakers/support staff). Figure 2 illustrates the qualitative coding workflow from the case study.

The analysis then synthesises these codes to identify effective arrangements. For the reference on the infrastructural and institutional arrangements investigated, we refer the reader to the original paper and the open data repository containing the interviews as well as the codebook[1] (licensed under *CC BY 4.0*[2]). While the approach developed in Section 4 is evaluated in comparison to the results of this

[1]See https://data.4tu.nl/datasets/aa72daa3-6e9a-47a1-b35b-05ade49e16a5/1
[2]See https://creativecommons.org/licenses/by/4.0/deed

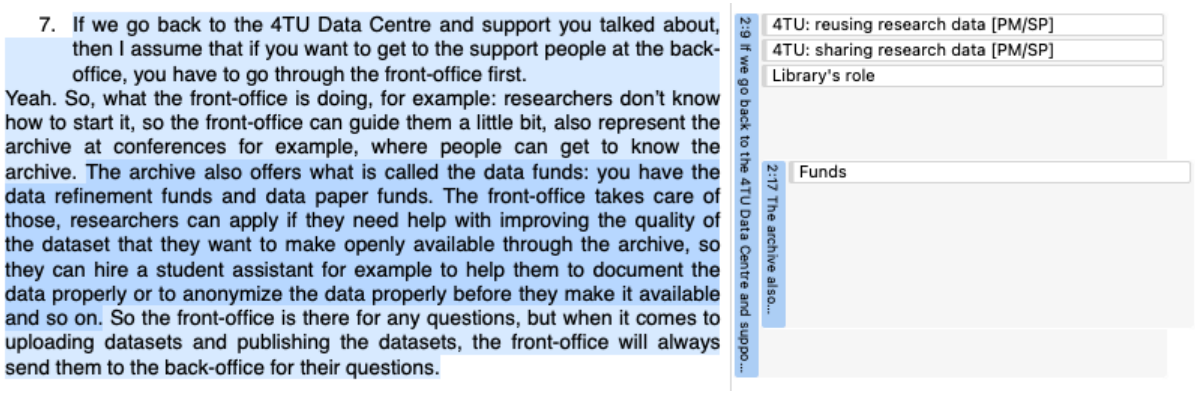


Figure 1: Example of manual qualitative coding from the original study showing a coded interview excerpt in the ATLAS.ti tool (transcript I1). Codes are assigned to text portions describing various infrastructural and institutional arrangements.

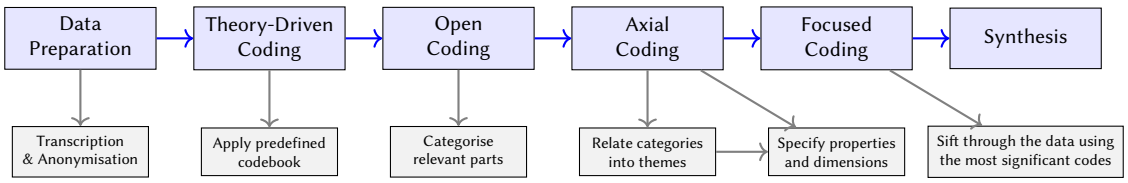


Figure 2: Illustration of the qualitative coding process workflow from the case study [56]

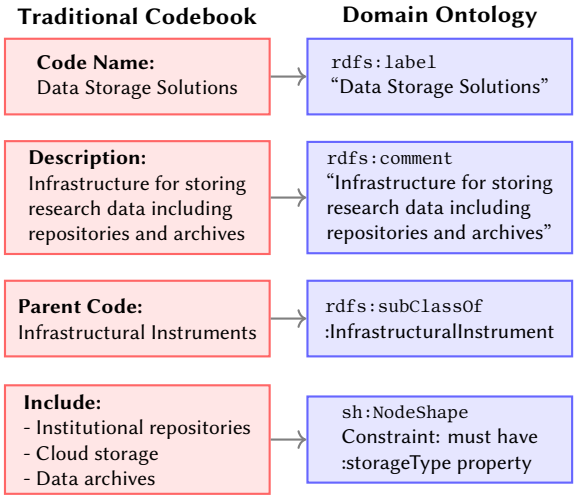
workflow, it differs from the methodology used by the authors of the case study by conflating the open, axial and focused coding into a composite inductive coding phase.

4. Methodology

This section presents our ontology-mediated approach to automating qualitative coding. We illustrate aspects of the approach using a case study introduced in Section 3, reproducing a published qualitative coding analysis to enable comparison with manual coding results. Our methodology encompasses three key components: (1) the formalisation of qualitative coding concepts using semantic web standards, (2) the implementation of an automated coding pipeline in the *AutoCod* tool, and (3) a preliminary evaluation comparing automated results against human-coded baselines of the case study.

4.1. Ontology-Based Formalisation of Qualitative Coding

Our ontology-driven approach transforms traditional qualitative coding practices by formalising codebook elements as semantic web constructs in a domain-specific ontology. Where traditional codebooks provide linear lists of codes with textual descriptions, ontologies encode this knowledge in a machine-readable format by minting specific URIs for categories and then including explicit semantics using semantic web standards. The mapping between codebook elements and ontological constructs follows established patterns: code names become `rdfs:label` properties providing human-readable identifiers; code descriptions and coding guidelines are captured in `rdfs:comment` annotations that preserve the interpretive guidance, textual examples, and other contextual information including textual inclusion and exclusion criteria for the code; hierarchical code structures translate to class hierarchies using `rdfs:subClassOf` relationships; besides being included verbatim, inclusion/exclusion criteria are also formalised as SHACL constraints. This formalisation enables automated validation while maintaining the natural language contextual explanations underlying the coding schemes, as illustrated in Figure 3 (a) and (b).



(a) Mapping between traditional codebook elements and semantic web constructs

```
[Prefix definitions]

:DataStorageSolutions a owl:Class ;
  rdfs:label "Data Storage Solutions" ;
  rdfs:comment "Infrastructure for storing
       research data including repositories
       and archives" ;
  rdfs:subClassOf :InfrastructuralInstrument
       .

:DataStorageSolutionsShape a sh:NodeShape ;
  sh:targetClass :DataStorageSolutions ;
  sh:property [
    sh:path :storageType ;
    sh:minCount 1
  ] .
```

(b) Ontology snippet with constraints

Figure 3: Mapping between traditional codebook elements and semantic web constructs in the ontology-driven approach. (a) Mapping from codebook to ontology constructs. (b) Example with (SHACL) constraints.

4.2. Domain Ontology Creation

In our use case, we define a domain-specific ontology[3] for infrastructural and institutional arrangements. This reflects the common theory-based beginning of the qualitative coding approach. The resulting ontology defines a core class `Instrument`, which is specialised into `InfrastructuralInstrument` and `InstitutionalInstrument`.

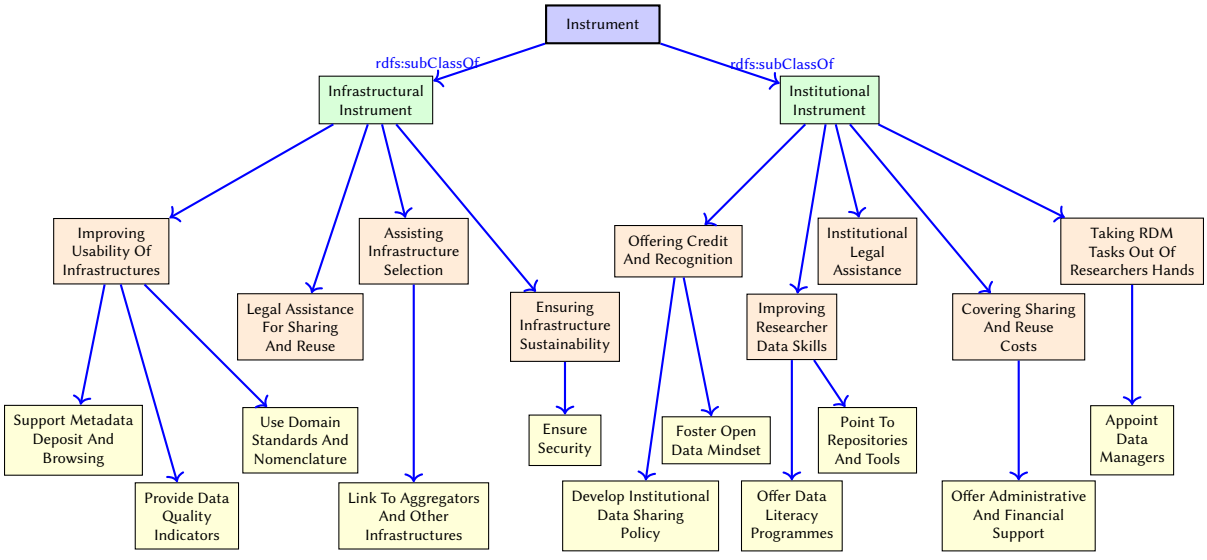


Figure 4: Domain-specific ontology for infrastructural and institutional instruments in open research data sharing. The ontology defines a hierarchical class structure using `rdfs:subClassOf` relationships (blue arrows) throughout the class hierarchy. In practice, instances of instruments are linked to category classes through the `hasCategory` object property.

Orienting on the literature-based taxonomy introduced in the case study, each branch is organised into `InstrumentCategory` classes (e.g., improving usability, legal assistance, selection support, sustainability for infrastructures; credit/recognition, researcher skills, legal assistance, cost coverage, and delegation

[3]The domain-specific ontology together with additional materials can be found in the code repository: https://github.com/semantisch/autocod/tree/main/paper/resources .

of research data management tasks for institutions), under which specific instrument classes (e.g., metadata browsing, data quality indicators; data literacy programmes, administrative/financial support) are defined. Relationships include `hasCategory` linking instruments to their categories, as well as textual definitions (`rdfs:comment`) specifying scope and intent of each class in line with the theoretical framework described in the paper. Figure 4 illustrates the resulting class hierarchy.

In the next step, we refine the ontology with SHACL constraints for downstream validation. The constraints enforce branch consistency (ensuring instruments are categorised within their appropriate branch), prevent cross-branch categorisation, and restrict single typing to avoid dual classification as both infrastructural and institutional. Unlike OWL restrictions such as `rdfs:domain`/`rdfs:range` statements which serve inferential purposes and operate under the open-world assumption, SHACL constraints provide closed-world validation against data graphs, enabling detection of constraint violations, that could remain undetected in OWL's monotonic reasoning framework.

4.3. Linking Ontology

Beyond the domain-specific ontology placing codes in a taxonomy, qualitative coding requires a linking mechanism to connect identified concepts to their textual evidence in the source material. This necessitates a separate linking ontology that bridges the semantic representations with the original data sources.

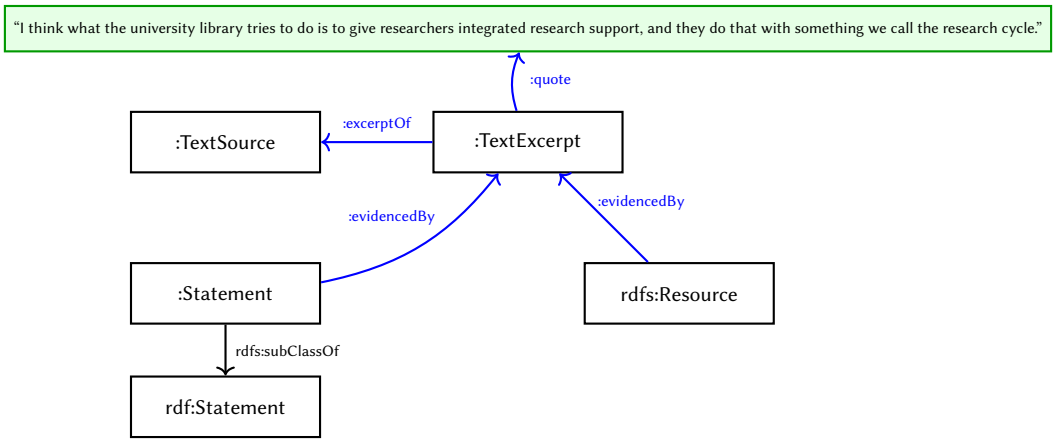


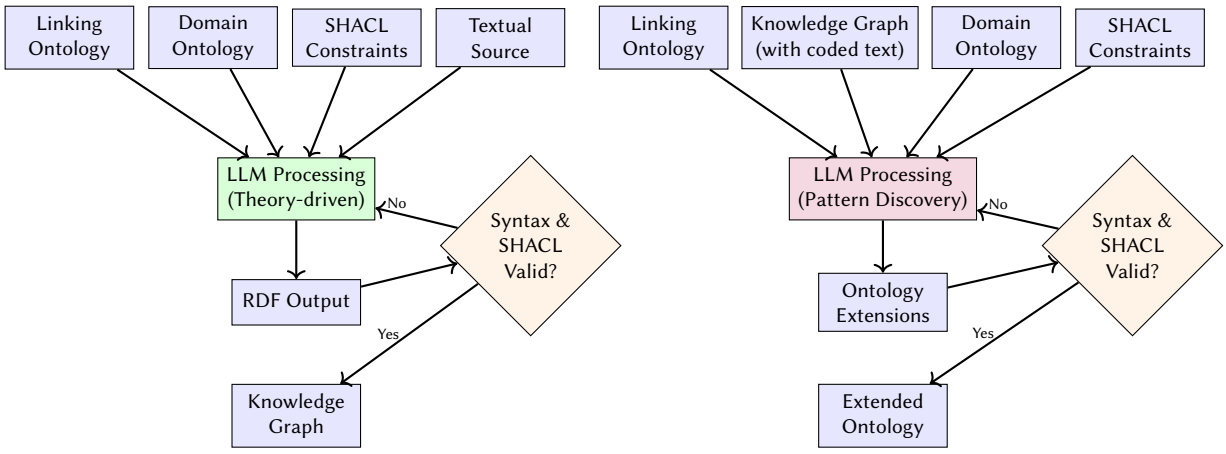
Figure 5: Linking ontology structure for connecting knowledge graph resources and statements to textual evidence. The ontology enables traceability from both semantic assertions and individual resources to their supporting quotes, employing standard RDF reification for statements and direct linking for resources.

The linking ontology must satisfy three core requirements: (1) provenance tracking to maintain traceability between extracted concepts and their source locations within transcripts, (2) evidence attachment enabling direct quotation and contextual information to support coding decisions, and (3) entity resolution to handle references to the same real-world entities across different textual mentions. The resulting linking ontology reuses the PROV ontology[4] (notably, by specifying `:evidencedBy` as the sub-property of the `prov:wasDerivedFrom`) and employs a reification pattern where coding statements are modelled as first-class objects with properties linking to both the semantic assertion (e.g., an instrument instance having a particular category) or resource, and their textual justification (specific quotes, paragraph references, and contextual metadata from the source interviews). Figure 5 illustrates the resulting linking ontology.

[4]See https://www.w3.org/TR/prov-o/

4.4. Automated Coding Workflow

In summary, the coding pipeline relies on the two above-mentioned ontologies: a domain-specific ontology defining coding concepts and significant constraints for validation, and a (generic) linking ontology that connects textual evidence to semantic statements. Our pipeline operates through an LLM-based approach where a large language model is prompted to extract domain concepts from transcripts using only the vocabulary defined in the provided ontologies, and tasked to retrieve well-formed RDF data. At each LLM-extraction step, RDF syntax as well as constraints are validated, and a retry loop (including the previous unsuccessful validation report) is implemented in case of errors. The pipeline implements the deductive and inductive coding phases in the following way (see Figure 6):



(a) Deductive approach (b) Inductive approach

Figure 6: Ontology-driven qualitative coding pipeline comparing deductive and inductive approaches. (a) Deductive approach uses predefined domain ontology for theory-driven concept extraction. (b) Inductive approach discovers emergent patterns from coded excerpts to extend the ontology.

Strategy Name	Strategy Description	Prompt Fragment
Pattern Coding	Groups similar codes into meaningful clusters (themes, explanations) by identifying recurring patterns across coded excerpts.	"Propose new classes for existing instances in the knowledge graph."
Focused Coding	Identifies the most frequent or significant codes to develop major categories, prioritizing concepts that appear most often or carry most weight.	"Analyse instances and create superclasses, then relate them to lower classes using rdfs:subClassOf."
Axial Coding	Reassembles data by relating categories to subcategories and specifying their properties and dimensions through systematic analysis.	"Examine relationships between classes and specify object properties with domain/range constraints to model causal, contextual, and consequential relationships."
Theoretical Coding	Integrates and synthesises all codes and categories into a coherent theoretical framework by identifying core categories that explain the phenomenon.	"Synthesise all concepts into a coherent theoretical framework. Create new ontology classes and relationships freely if they help explain the central phenomenon and build theory."

Table 2
Inductive coding strategies for ontology extension

Deductive Coding Phase (Figure 6a): The LLM receives three inputs: (1) the domain ontology in Turtle format, defining instrument classes, categories, and their relationships; (2) the linking ontology; and (3) the input data in plain text. The model is constrained (including, through automatic validation) to instantiate only classes and properties explicitly defined in the domain ontology, ensuring theory-driven coding aligned with the predefined conceptual framework. Each instantiated concept must be supported by direct quotes from the transcript, linked using the linking ontology. The output is a well-formed RDF graph.

Inductive Coding Phase (Figure 6b): In the inductive phase, text excerpts are re-analysed to discover emergent concepts not captured in the initial ontology using predefined strategies common in qualitative coding [3] (see Table 2). This mirrors the open coding approach in traditional qualitative analysis, where new categories emerge from the data. The LLM follows encoded strategies to propose extensions to the domain ontology based on patterns observed in the coded excerpts, which are then validated against SHACL constraints before incorporation.

5. Pipeline Implementation

We then implement the ontology-driven qualitative coding workflows described in Section 4.4 as an automated qualitative coding tool which accepts domain ontologies and textual transcripts as inputs and utilises large language models to extract concepts constrained by the ontological vocabulary. The implementation validates extracted RDF against SHACL constraints and maintains links between coded concepts and their supporting text excerpts through the linking ontology. It processes input sources in both deductive mode (applying predefined ontological concepts) and inductive mode (discovering new concepts to extend the ontology), with all outputs forming validated RDF knowledge graphs. More detailed architectural description of the tool and its documentation are available in the repository.[5]

The tool implements a number of supporting features extending the general (deductive and inductive) approaches described in Section 4.4. It can be used as a web-based application, via API or as a command-line tool. For larger single datasets, the tool allows definition of separators for chunking as well as their parallel processing, which is followed by the integration of the resulting chunk-based knowledge graphs. In the file explorer, the tool highlights the coded excerpts for better overview. All input sources in a workspace are integrated into the combined coding knowledge graph that can be exported. All intermediate results, as well as resources including the domain ontology, linking ontology as well as prompts can be edited at any stage to adjust the coding process. Figure 7 shows the *AutoCod* web interface.[6]

6. Evaluation

Our evaluation methodology relies on a comparative analysis between automated and manual coding results across both deductive and inductive phases:

For the deductive coding phase, we assess the alignment between excerpts identified by the automated approach and those coded manually in the original study. We examine the intersection of coded segments, noting that while the automated approach tends to identify precise text fragments at the sentence or phrase level, manual coding in the original study typically operated at the paragraph level, resulting in coarser-grained text segmentation.

For the inductive coding phase, we evaluate the semantic correspondence between ontology extensions generated by our automated approach and the emergent categories identified through manual analysis in the original study. This comparison focuses on conceptual overlap rather than exact terminological matching, as the automated approach produces formal ontology classes while manual coding yields informal category labels. Finally, we assess the degree to which automatically generated

[5]See https://github.com/semantisch/autocod

[6]The *AutoCod* tool and its documentation are available at https://github.com/semantisch/autocod

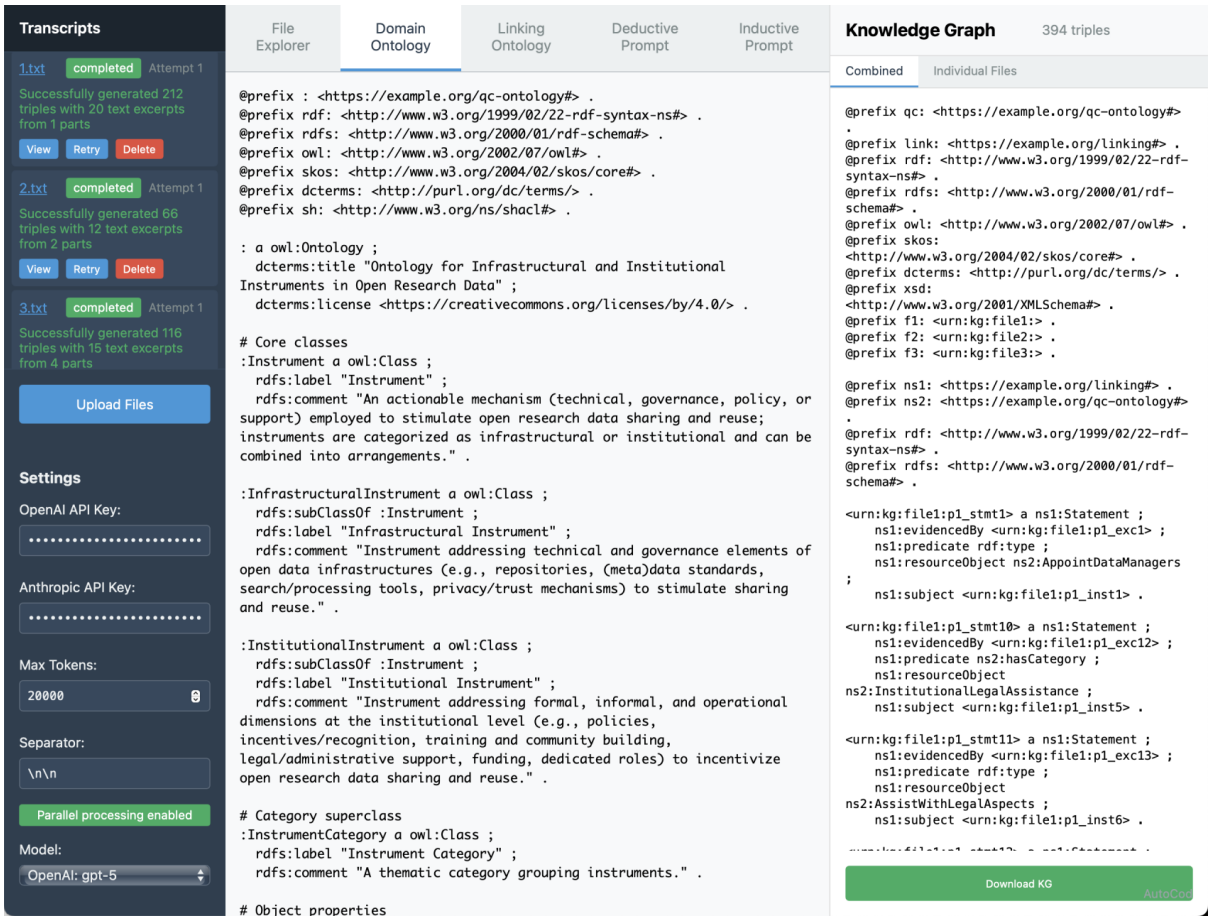


Figure 7: *AutoCod* web interface showing the main workspace with transcript management (left panel), ontology editor (centre), and knowledge graph viewer (right panel). The interface displays the domain ontology definition with class hierarchies and the resulting RDF triples with evidence links to source text excerpts.

superclasses and relationships capture the theoretical insights derived from traditional qualitative analysis in the case study.

For both phases, we conduct the evaluation using the GPT-5 model (08-08-2025, temperature hyperparameter not supported), processing transcripts that have been chunked on double newlines, with a maximum context window of 20,000 tokens per chunk. This ensures that the model can handle long input documents while maintaining coherence and context.

7. Results

As the result of the deductive pass on the seven interview transcripts using the domain-specific ontology defined in Section 4.2, a combined knowledge graph of 865 triples is created. Table 3 summarises the number of triples and text excerpts/codes created for each transcript. The resulting generated knowledge graphs are available in the online repository.[7]

While the automated approach selects less codes than the case study, the manual codes are often overlapping (cf. Figure 1). In terms of the overlap in codes, we calculate the precision and recall of the automated approach compared to the non-overlapping manual codes. Precision is the number of correctly identified codes divided by the total number of codes identified by the automated approach. Recall is the number of correctly identified codes divided by the total number of the non-overlapping codes in the case study.

[7]See https://github.com/semantisch/autocod/tree/main/paper/resources

Interview	Codes				Constructed Knowledge Graph		
	Manual (Non-O)	Precision	Recall	Automated	Triples	Statements	Instances
1	41 (24)	95%	79.2%	20	212	20	10
2	37 (23)	34.8%	66.7%	12	66	0	5
3	28 (19)	100%	79%	15	116	7	8
4	32 (20)	90.9%	50%	11	82	7	11
5	25 (18)	88.2%	83.3%	17	106	3	9
6	28 (19)	100%	94.7%	18	120	2	12
7	35 (24)	100%	79.2%	19	163	2	18
Total	**226 (147)**	**85.5%**	**68%**	**117**	**865**	**41**	**73**

Table 3
Deductive coding results by interview transcript. Number in parentheses shows non-overlapping (Non-O) codes.

In the next stage, open coding on the selected excerpts is used to extend the ontology itself. Here, we (1) compare the results from cumulative coding strategies (after one step of inductive coding) to the codes from the study and (2) analyse the additional identified concepts/differences. The extended ontology is available in the online repository.

The automated inductive coding generated 67 new ontology classes and properties, which we compare to the 55 codes from the manual codebook. Table 4 shows the mapping between automated concepts and manual codes:

Automated Ontology Class	Manual Code(s)	Match Type
Organisational Structures		
:DataSteward	Data Steward [PM/SP], Data Steward [R]	Direct
:Library	Library's role	Direct
:FrontOfficeUnit, :BackOfficeUnit	Support implementation tips	Instance
Activities and Support		
:DMPAuthoring	DMP implementation tips	Direct
:DatasetCuration, :DataDocumentation	Infrastructure usability	Instance
:ProvideEthicsAndPrivacyReviewSupport	Policy implementation tips	Instance
:DeliverCarpentriesWorkshops	Education	Instance
Financial Support		
:ProvideDataRefinementSmallGrants	Funds	Direct
Infrastructure		
:RepositorySelection	Choosing infrastructures, Discovering infrastructures	Direct
:PerformSubmissionQualityChecks	Infrastructure requirements	Instance

Table 4
Overlap between automated inductive coding and manual codebook

The automated approach created formal class hierarchies implicit in the manual coding (e.g., `:OrganisationalUnit` with subclasses `:Library`, `:Faculty`, `:CentralAdministration`). Relationships between concepts were formalised (e.g., `:administeredBy`, `:supportsActivity`, `:hasTargetAudience`) whereas manual codes captured these as implicit connections. Explicit classes for `:PhDStudents` and `:Researchers` emerged, while manual codes distinguished perspectives with [PM/SP] and [R] suffixes.

Conversely, the manual codebook captured a number of aspects not identified by the automated approach: 13 codes focusing on experiences with various RDM aspects (e.g., "Experience with DMPs", "Experience with infrastructures"), 6 codes addressing motivations for (not) sharing/reusing data, personal/community-related codes like "Colleagues in department", "Peer-to-peer", and "Trust in peers and the community" as well as direct mentions of 4TU repository and specific implementation details

that were naturally not as prominent in the automatic approach since they primarily represent implicit knowledge/focus of researchers.

Thus, the automated approach identified formal structural relationships and created reusable ontological patterns, achieving approximately 35% direct and partial overlap with manual codes. However, it noticeably missed subjective, experiential, and motivational aspects which human coders intuitively capture.

8. Discussion

We have shown that, in a single automated, inductive coding pass, it is possible to identify and formalise key structural codes within an ontology, replicating the results of inductive coding phases. While the resulting codes are comparable to the use case (high precision of 85.5%), the automated approach noticeably lacks in coverage (recall of 68%).

The relatively high precision achieved in deductive coding is dependent on exact and extensive `rdfs:comment` annotations that capture definitions, coding guidelines, inclusion/exclusion criteria. Although not included in this study, few-shot examples could also have a positive impact on precision. This suggests that ontology engineers should invest considerable effort in creating detailed, stand-alone natural language descriptions that go beyond minimal definitions. Also, such a perspective establishes a new "consumer" of ontologies and of the ontology engineering process – language models that utilise them as an interface for knowledge extraction, and suggests that future ontology engineering should explicitly consider human-AI collaboration patterns.

Another advantage of the semantic web stack is the validation layer which prevents logically inconsistent coding. In this regard, constraint definition (especially, SHACL) should be considered an essential component of ontologies that could potentially be utilised for knowledge extraction or knowledge base construction.

The recall results could be also attributed to the singular pass of generation. In this regard, the few-shot, top-k, majority rule, cumulative, LLM-as-a-judge and agentic components are expected to considerably improve it. Our results also show that subjective and experiential codes are less readily captured automatically. Addressing these aspects may require additional strategies to guide the coding process, including human-in-the-loop (HITL) approaches or retrieval-augmented generation (RAG).

Notable limitations of the performed case study include reliance on only one, closed-source LLM for generation, potential data leakage (dataset published before the LLM training date), limited nature of the golden standard used in the case study, basic nature of the precision/recall evaluation without a dedicated ICR study and lack of an ablation study on the pipeline components or output normalisation via, e.g., aggregation, ensemble decoding and canonicalisation. Furthermore, while the iterative constraint validation is implemented via a SHACL validation engine, the repair process is currently relying solely on prompting. While such an approach is effective, more advanced repair techniques [61] are expected to improve repair performance.

In general, the proposed approach reveals synergies between traditionally distinct fields. Qualitative coding's emphasis on iterative refinement aligns naturally with ontology evolution and refinement as part of the ontology engineering methodologies [62]. In this regard, semantic web standards provide the formalisation that qualitative methods often lack, and the definition of codebooks as ontologies is well suited for their FAIR publication [63] and reuse in line with Linked Data Principles [64].

Finally, with regard to the replicability of qualitative coding, the contribution of coding automation is twofold. On the one hand, the use of automatic coding tools enable automatic documentation and protocol generation for the coding process, ensuring protocol adherence but also producing detailed reproducible protocols. On the other hand, the precise definition of the initial configuration (initial ontology, language model and hyper-parameters, coding pipeline) and the availability of reproducible protocols enables direct and, potentially, automatic replication and sensitivity analyses with regard to pipeline configurations, language model hyper-parameters, prompt engineering and ontology variations.

9. Conclusions

Our first contribution is to demonstrate that codebooks can be formally defined as and mapped to ontologies, which enables LLM-supported knowledge extraction. In our use case, we created a domain ontology directly from the manual codebook and demonstrated its practical utility in automated coding of the case study interviews.

Then, we have created pipelines for deductive and inductive ontology-driven and LLM-supported coding which incorporated validation steps, and implemented the pipelines as an open source automatic coding tool. As a supporting resource to linking instances and statements in the generated coding knowledge graph, we have created the minimal linking ontology defining relationships between text excerpts/codes, extracted instances and statements.

Following the framework of the case study, we have evaluated the performance of the pipelines on case study interviews, testing both deductive and inductive automated coding phases and comparing them to the manually coded gold standard of the case study. We have demonstrated the general feasibility of ontology-mediated qualitative coding, with potential implications for both qualitative research methodology and ontology engineering practices. By design, the quality of the resulting knowledge base depends on the quality of the domain-specific ontology itself, as well as the associated constraints used for validation and repair of violations.

Future work

In addition to the need for a more robust pipeline design and a more robust, extensive evaluation (addressing limitations discussed in Section 8), one clear avenue for future work aimed at evaluating the reliability of the technique, consequently, facilitating adoption is defining a benchmark for evaluating automated qualitative coding pipelines, which would include source data, codebooks, domain-specific ontologies, coding tasks (deductive, inductive, mixed) and coding gold standards, as well as considering aspects that underpin trust in the coding process, such as transparency, explainability and interpretability.

A number of improvements on the coding pipeline mentioned in Section 8 could be implemented before a comprehensive evaluation on a larger benchmark dataset is attempted. In this context, a framework for the ICR evaluation of the structured coding results (coding knowledge graph) has to be established, which will include the definition of the level of reliability testing (triple-level, entity-level, code-level evaluations), the semantic matching criteria and adaptations to the conventional ICR metrics (especially, in the context of collaborative/agentic coding with multiple LLMs). Finally, while general synthesis of the coding results, even in inductive case, remains outside the scope of current work it is an ambitious and logical future direction for qualitative coding automation.

Acknowledgments

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/COE12.

Declaration on Generative AI

During the preparation of this work, the author(s) used Claude 4 Opus in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] B. G. Glaser, A. L. Strauss, E. Strutzel, The discovery of grounded theory; strategies for qualitative research, Nursing research 17 (1968) 364.

[2] M. Skjott Linneberg, S. Korsgaard, Coding qualitative data: A synthesis guiding the novice, Qualitative research journal 19 (2019) 259–270.

[3] J. Saldaña, The Coding Manual for Qualitative Researchers, 4th ed., SAGE Publishing Inc., Thousand Oaks, California, 2021.

[4] M. Williams, T. Moser, The art of coding and thematic exploration in qualitative research, International management review 15 (2019) 45–55.

[5] K. Charmaz, Constructivist grounded theory, The journal of positive psychology 12 (2017) 299–300.

[6] J. Ritchie, R. Ormston, C. McNaughton Nicholls, J. Lewis, Qualitative research practice: A guide for social science students and researchers (2013).

[7] C. K. Russell, D. M. Gregory, Issues for consideration when choosing a qualitative data management system, Journal of Advanced Nursing 18 (1993) 1806–1816. URL: https://onlinelibrary.wiley.com/doi/10.1046/j.1365-2648.1993.18111806.x. doi:10.1046/j.1365-2648.1993.18111806.x.

[8] E. Childs, L. B. Demers, Qualitative coding boot camp: an intensive training and overview for clinicians, educators, and administrators, MedEdPORTAL 14 (2018) 10769.

[9] S. O. Clarke, W. C. Coates, J. Jordan, A practical guide for conducting qualitative research in medical education: Part 3—Using software for qualitative analysis, AEM Education and Training 5 (2021) e10644. URL: https://onlinelibrary.wiley.com/doi/10.1002/aet2.10644. doi:10.1002/aet2.10644.

[10] P. TalkadSukumar, R. Metoyer, Replication and transparency of qualitative research from a constructivist perspective, OSF Preprints (2019). URL: https://osf.io/6efvp/. doi:10.31219/osf.io/6efvp, accessed March 6, 2025.

[11] C. O'Connor, H. Joffe, Intercoder Reliability in Qualitative Research: Debates and Practical Guidelines, International Journal of Qualitative Methods 19 (2020) 1609406919899220. URL: https://journals.sagepub.com/doi/10.1177/1609406919899220. doi:10.1177/1609406919899220.

[12] I. G. Raskind, R. C. Shelton, D. L. Comeau, H. L. Cooper, D. M. Griffith, M. C. Kegler, A review of qualitative data analysis practices in health education and health behavior research, Health Education & Behavior 46 (2019) 32–39.

[13] N. C. Nelson, K. Ichikawa, J. Chung, M. M. Malik, Mapping the discursive dimensions of the reproducibility crisis: A mixed methods analysis, PLOS ONE 16 (2021) e0254090. URL: https://dx.plos.org/10.1371/journal.pone.0254090. doi:10.1371/journal.pone.0254090.

[14] K. K. C. Cheung, K. W. H. Tai, The use of intercoder reliability in qualitative interview data analysis in science education, Research in Science & Technological Education 41 (2023) 1155–1175. URL: https://www.tandfonline.com/doi/full/10.1080/02635143.2021.1993179. doi:10.1080/02635143.2021.1993179.

[15] M. Baker, 1,500 scientists lift the lid on reproducibility, Nature 533 (2016) 452–454. URL: https://www.nature.com/articles/533452a. doi:10.1038/533452a.

[16] Open Science Collaboration, Estimating the reproducibility of psychological science, Science 349 (2015) aac4716. URL: https://www.science.org/doi/10.1126/science.aac4716. doi:10.1126/science.aac4716.

[17] C. F. Camerer, A. Dreber, F. Holzmeister, T.-H. Ho, J. Huber, M. Johannesson, M. Kirchler, G. Nave, B. A. Nosek, T. Pfeiffer, A. Altmejd, N. Buttrick, T. Chan, Y. Chen, E. Forsell, A. Gampa, E. Heikensten, L. Hummer, T. Imai, S. Isaksson, D. Manfredi, J. Rose, E.-J. Wagenmakers, H. Wu, Evaluating the replicability of social science experiments in Nature and Science between 2010 and 2015, Nature Human Behaviour 2 (2018) 637–644. URL: https://www.nature.com/articles/s41562-018-0399-z. doi:10.1038/s41562-018-0399-z.

[18] A. Chandrasekar, S. E. Clark, S. Martin, S. Vanderslott, E. C. Flores, D. Aceituno, P. Barnett, C. Vindrola-Padros, N. Vera San Juan, Making the most of big qualitative datasets: a living systematic review of analysis methods, Frontiers in big Data 7 (2024) 1455399.

[19] A. Zubiaga, Natural language processing in the era of large language models, Frontiers in Artificial

Intelligence 6 (2024) 1350306. URL: https://www.frontiersin.org/articles/10.3389/frai.2023.1350306/full. doi:10.3389/frai.2023.1350306.

[20] R. Han, C. Yang, T. Peng, P. Tiwari, X. Wan, L. Liu, B. Wang, An empirical study on information extraction using large language models, arXiv preprint arXiv:2409.00369 (2024).

[21] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, Y. Wang, E. Chen, Large Language Models for Generative Information Extraction: A Survey, 2024. URL: http://arxiv.org/abs/2312.17617. doi:10.48550/arXiv.2312.17617, arXiv:2312.17617 [cs].

[22] X.-Y. Zhang, S.-M. Cai, X.-R. Shen, Y. Han, W.-H. Hu, Y.-R. Zhang, Efficient Unified Information Extraction Model Based on Large Language Models, 2024. URL: https://www.ssrn.com/abstract=5053609. doi:10.2139/ssrn.5053609.

[23] F. Polat, I. Tiddi, P. Groth, Testing prompt engineering methods for knowledge extraction from text, Semantic Web (2024) 1–34. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SW-243719. doi:10.3233/SW-243719.

[24] V. Perot, K. Kang, F. Luisier, G. Su, X. Sun, R. S. Boppana, Z. Wang, Z. Wang, J. Mu, H. Zhang, C.-Y. Lee, N. Hua, LMDX: Language model-based document information extraction and localization, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Findings of the Association for Computational Linguistics: ACL 2024, Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 15140–15168. URL: https://aclanthology.org/2024.findings-acl.899/. doi:10.18653/v1/2024.findings-acl.899.

[25] J. Dagdelen, A. Dunn, S. Lee, N. Walker, A. S. Rosen, G. Ceder, K. A. Persson, A. Jain, Structured information extraction from scientific text with large language models, Nature Communications 15 (2024) 1418.

[26] N. Mihindukulasooriya, S. Tiwari, D. Dobriy, F. Å. Nielsen, T. R. Chhetri, A. Polleres, Scholarly wikidata: Population and exploration of conference data in wikidata using llms, in: International Conference on Knowledge Engineering and Knowledge Management, Springer, 2024, pp. 243–259.

[27] D. Dobriy, Employing rag to create a conference knowledge graph from text, in: ESWC'2024: The 21st Extended Semantic Web Conference, Hersonissos, Greece, 2024. URL: https://ceur-ws.org/Vol-3747/text2kg_paper4.pdf.

[28] R. Alharbi, U. Ahmed, D. Dobriy, W. Łajewska, L. Menotti, M. J. Saeedizade, M. Dumontier, Exploring the role of generative ai in constructing knowledge graphs for drug indications with medical context, 15th International Semantic Web Applications and Tools for Healthcare and Life Sciences (SWAT4HCLS 2024) (2024).

[29] A. Vijayan, A Prompt Engineering Approach for Structured Data Extraction from Unstructured Text Using Conversational LLMs, in: 2023 6th International Conference on Algorithms, Computing and Artificial Intelligence, ACM, Sanya China, 2023, pp. 183–189. URL: https://dl.acm.org/doi/10.1145/3639631.3639663. doi:10.1145/3639631.3639663.

[30] M. Moundas, J. White, D. C. Schmidt, Prompt patterns for structured data extraction from unstructured text, in: Proceedings of the 31st Conference on Pattern Languages of Programs, People, and Practices (PLoP 2024), ACM, 2024, pp. 1–15. URL: https://www.cs.wm.edu/~dcschmidt/PDF/Prompt_Patterns_for_Structured_Data_Extraction_from_Unstructured_Text___Final.pdf.

[31] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities, in: Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web, 2023, pp. 91–103.

[32] A. Scherp, G. Groener, P. Škoda, K. Hose, M.-E. Vidal, Semantic Web: Past, Present, and Future, Transactions on Graph Data and Knowledge (TGDK) 2 (2024) 3:1–3:37. URL: https://drops.dagstuhl.de/entities/document/10.4230/TGDK.2.1.3. doi:10.4230/TGDK.2.1.3, artwork Size: 37 pages, 1598870 bytes Medium: application/pdf Publisher: Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[33] A. Breit, L. Waltersdorfer, F. J. Ekaputra, M. Sabou, A. Ekelhart, A. Iana, H. Paulheim, J. Portisch, A. Revenko, A. t. Teije, et al., Combining machine learning and semantic web: A systematic mapping study, ACM Computing Surveys 55 (2023) 1–41.

[34] Y. Tan, D. Min, Y. Li, W. Li, N. Hu, Y. Chen, G. Qi, Can ChatGPT Replace Traditional KBQA Models?

An In-Depth Analysis of the Question Answering Performance of the GPT LLM Family, in: T. R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng, J. Li (Eds.), The Semantic Web – ISWC 2023, volume 14265, Springer Nature Switzerland, Cham, 2023, pp. 348–367. URL: https://link.springer.com/10.1007/978-3-031-47240-4_19. doi:10.1007/978-3-031-47240-4_19, series Title: Lecture Notes in Computer Science.

[35] G. Agrawal, T. Kumarage, Z. Alghami, H. Liu, Can Knowledge Graphs Reduce Hallucinations in LLMs? : A Survey (2023). URL: https://arxiv.org/abs/2311.07914. doi:10.48550/ARXIV.2311.07914, publisher: arXiv Version Number: 1.

[36] K. Sun, Y. E. Xu, H. Zha, Y. Liu, X. L. Dong, Head-to-Tail: How Knowledgeable are Large Language Models (LLM)? A.K.A. Will LLMs Replace Knowledge Graphs? (2023). URL: https://arxiv.org/abs/2308.10168. doi:10.48550/ARXIV.2308.10168, publisher: arXiv Version Number: 1.

[37] A. Martino, M. Iannelli, C. Truong, Knowledge Injection to Counter Large Language Model (LLM) Hallucination, in: C. Pesquita, H. Skaf-Molli, V. Efthymiou, S. Kirrane, A. Ngonga, D. Collarana, R. Cerqueira, M. Alam, C. Trojahn, S. Hertling (Eds.), The Semantic Web: ESWC 2023 Satellite Events, volume 13998, Springer Nature Switzerland, Cham, 2023, pp. 182–185. URL: https://link.springer.com/10.1007/978-3-031-43458-7_34. doi:10.1007/978-3-031-43458-7_34, series Title: Lecture Notes in Computer Science.

[38] Y. Li, R. Zhang, J. Liu, G. Liu, An Enhanced Prompt-Based LLM Reasoning Scheme via Knowledge Graph-Integrated Collaboration (2024). URL: https://arxiv.org/abs/2402.04978. doi:10.48550/ARXIV.2402.04978, publisher: arXiv Version Number: 1.

[39] J. Huang, X. Zhang, Q. Mei, J. Ma, Can LLMs Effectively Leverage Graph Structural Information: When and Why (2023). URL: https://arxiv.org/abs/2309.16595. doi:10.48550/ARXIV.2309.16595, publisher: arXiv Version Number: 2.

[40] Y. Huang, L. Shi, A. Liu, H. Xu, Evaluating and Enhancing Large Language Models for Conversational Reasoning on Knowledge Graphs (2023). URL: https://arxiv.org/abs/2312.11282. doi:10.48550/ARXIV.2312.11282, publisher: arXiv Version Number: 2.

[41] A. Martino, M. Iannelli, C. Truong, Knowledge Injection to Counter Large Language Model (LLM) Hallucination, in: C. Pesquita, H. Skaf-Molli, V. Efthymiou, S. Kirrane, A. Ngonga, D. Collarana, R. Cerqueira, M. Alam, C. Trojahn, S. Hertling (Eds.), The Semantic Web: ESWC 2023 Satellite Events, volume 13998, Springer Nature Switzerland, Cham, 2023, pp. 182–185. URL: https://link.springer.com/10.1007/978-3-031-43458-7_34. doi:10.1007/978-3-031-43458-7_34, series Title: Lecture Notes in Computer Science.

[42] J. Baek, A. F. Aji, A. Saffari, Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering, 2023. URL: http://arxiv.org/abs/2306.04136. doi:10.48550/arXiv.2306.04136, arXiv:2306.04136 [cs].

[43] Y. Wen, Z. Wang, J. Sun, MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models (2023). URL: https://arxiv.org/abs/2308.09729. doi:10.48550/ARXIV.2308.09729, publisher: arXiv Version Number: 4.

[44] H. Abu-Rasheed, M. H. Abdulsalam, C. Weber, M. Fathi, Supporting Student Decisions on Learning Recommendations: An LLM-Based Chatbot with Knowledge Graph Contextualization for Conversational Explainability and Mentoring (2024). URL: https://arxiv.org/abs/2401.08517. doi:10.48550/ARXIV.2401.08517, publisher: arXiv Version Number: 3.

[45] X. Guan, Y. Liu, H. Lin, Y. Lu, B. He, X. Han, L. Sun, Mitigating Large Language Model Hallucinations via Autonomous Knowledge Graph-based Retrofitting (2023). URL: https://arxiv.org/abs/2311.13314. doi:10.48550/ARXIV.2311.13314, publisher: arXiv Version Number: 1.

[46] T. Tietz, J. Jäger, J. Waitelonis, H. Sack, Semantic annotation and information visualization for blogposts with refer., in: VOILA@ ISWC, 2016, pp. 28–40.

[47] T. Rietz, A. Maedche, Cody: An ai-based system to semi-automate coding for qualitative research, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–14.

[48] Z. Cai, A. Siebert-Evenstone, B. Eagan, D. W. Shaffer, X. Hu, A. C. Graesser, ncoder+: a semantic tool

for improving recall of ncoder coding, in: International Conference on Quantitative Ethnography, Springer, 2019, pp. 41–54.
[49] M. Marathe, K. Toyama, Semi-automated coding for qualitative research: A user-centered inquiry and initial prototypes, in: Proceedings of the 2018 CHI conference on human factors in computing systems, 2018, pp. 1–12.
[50] R. Bijker, S. S. Merkouris, N. A. Dowling, S. N. Rodda, Chatgpt for automated qualitative research: Content analysis, Journal of medical Internet research 26 (2024) e59050.
[51] D. Dobriy, M. Beno, A. Polleres, Smw cloud: A corpus of domain-specific knowledge graphs from semantic mediawikis, in: A. Meroño Peñuela, et al. (Eds.), The Semantic Web. ESWC 2024, volume 14665 of *Lecture Notes in Computer Science*, Springer, Cham, 2024. URL: https://doi.org/10.1007/978-3-031-60635-9_9. doi:10.1007/978-3-031-60635-9_9.
[52] I. Feinerer, F. Wild, Automated coding of qualitative interviews with latent semantic analysis, in: Information systems technology and its applications–6th international conference–ISTA 2007, Gesellschaft für Informatik e. V., 2007, pp. 66–77.
[53] R. P. Lennon, R. Fraleigh, L. J. Van Scoy, A. Keshaviah, X. C. Hu, B. L. Snyder, E. L. Miller, W. A. Calo, A. E. Zgierska, C. Griffin, Developing and testing an automated qualitative assistant (aqua) to support qualitative analysis, Family medicine and community health 9 (2021) e001287.
[54] G. Bryda, D. Sadowski, From words to themes: Ai-powered qualitative data coding and analysis, in: World conference on qualitative research, Springer, 2024, pp. 309–345.
[55] I. G. Raskind, R. C. Shelton, D. L. Comeau, H. L. F. Cooper, D. M. Griffith, M. C. Kegler, A Review of Qualitative Data Analysis Practices in Health Education and Health Behavior Research, Health Education & Behavior 46 (2019) 32–39. URL: https://journals.sagepub.com/doi/10.1177/1090198118795019. doi:10.1177/1090198118795019.
[56] T. van Gend, A. Zuiderwijk, Open research data: A case study into institutional and infrastructural arrangements to stimulate open research data sharing and reuse, Journal of Librarianship and Information Science 55 (2023) 782–797.
[57] J. T. DeCuir-Gunby, P. L. Marshall, A. W. McCulloch, Developing and using a codebook for the analysis of interview data: An example from a professional development research project, Field methods 23 (2011) 136–155.
[58] T. R. Lindlof, B. C. Taylor, Qualitative communication research methods, Sage publications, 2017.
[59] K. Charmaz, Constructing grounded theory: A practical guide through qualitative analysis, sage, 2006.
[60] J. Corbin, A. Strauss, Unending work and care: managing chronic illness at home, Jossey-Bass health series (1988).
[61] T. Pellissier Tanon, C. Bourgaux, F. Suchanek, Learning how to correct a knowledge base from the edit history, in: The World Wide Web Conference, 2019, pp. 1465–1475.
[62] K. I. Kotis, G. A. Vouros, D. Spiliotopoulos, Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations, The Knowledge Engineering Review 35 (2020) e4.
[63] 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 3 (2016) 1–9.
[64] C. Bizer, T. Heath, T. Berners-Lee, Linked data: Principles and state of the art, in: World wide web conference, volume 1, Citeseer, 2008, p. 40.