Ontology Population using LLMs

Sanaz SAKI NOROUZI [a,1], Adrita BARUA [a], Antrea CHRISTOU [b], Nikita GAUTAM [a], Andrew EELLS [a], Pascal HITZLER [a], Cogan SHIMIZU [b,2]

[a] *Kansas State University, Manhattan, Kansas, USA*

[b] *Wright State University, Dayton, Ohio, USA*

ORCiD ID: Sanaz Saki Norouzi https://orcid.org/0009-0003-4441-108X, Adrita Barua https://orcid.org/0000-0002-3287-7443, Antrea Christou https://orcid.org/0009-0003-0428-2439, Nikita Gautam https://orcid.org/0000-0002-1572-5405, Andrew Eells https://orcid.org/0000-0001-6357-6646, Pascal Hitzler https://orcid.org/0000-0001-6192-3472, Cogan Shimizu https://orcid.org/0000-0003-4283-8701

Abstract. Knowledge graphs (KGs) are increasingly utilized for data integration, representation, and visualization. While KG population is critical, it is often costly, especially when data must be extracted from unstructured text in natural language, which presents challenges, such as ambiguity and complex interpretations. Large Language Models (LLMs) offer promising capabilities for such tasks, excelling in natural language understanding and content generation. However, their tendency to "hallucinate" can produce inaccurate outputs. Despite these limitations, LLMs offer rapid and scalable processing of natural language data, and with prompt engineering and fine-tuning, they can approximate human-level performance in extracting and structuring data for KGs. This study investigates LLM effectiveness for the KG population, focusing on the Enslaved.org Hub Ontology. In this paper, we report that compared to the ground truth, LLM's can extract $\approx 90\%$ of triples, when provided a modular ontology as guidance in the prompts.

Keywords. knowledge graph population, large language models, modular ontology modeling

1. Introduction

Knowledge graphs (KGs) have quickly become a major paradigm supported [1,2] by a broad set of methods and tools for the creation, extraction, integration, representation, and visualization of data, and supported by long-established W3C standards and recommendations [3,4,5,6]. Unfortunately, many aspects of knowledge engineering are quite expensive: from the knowledge model (i.e., ontology) development to the actual population (and validation) of the resulting KG [7]. Population is, in some sense perhaps the easiest component, yet this is only true if the data is already in a machine-parseable format (as many tools provide such a service, e.g., OpenRefine [8]). However, when the

[1] The first three named authors share first authorship.

[2] Corresponding Author: Cogan Shimizu, cogan.shimizu@wright.edu

text is in natural language, this becomes problematic. Given the various complexities of interpreting sense, sentiment, frames, or anaphora, translating to facts or knowledge can be quite difficult. There are many natural language processing (NLP) techniques for tackling these problems, yet at the forefront – in terms of both popularity and broad applicability – are large language models (LLMs).

LLMs have rapidly emerged as powerful tools in various domains, showcasing remarkable proficiency in tasks such as natural language understanding, translation, and content generation [9,10]. Their ability to process – and interpret – vast amounts of text data allows them to generate coherent and contextually relevant responses, often surpassing traditional models in creativity and nuance.

However, LLMs are prone to an effect called confabulation, also referred to hallucination. This phenomenon occurs when the model produces responses that may *seem* coherent and contextually relevant but are factually inaccurate or entirely fabricated. There is no strict guarantee that any particular response does not have confabulation, but they are more likely to occur in certain scenarios, such as when the query is ambiguous, the topic in question is particularly niche (and thus not present in the original training data), or if the query requires a certain level of complexity or creativity to accomplish.

Even with these caveats, LLMs are much faster than humans at certain knowledge extraction tasks (e.g., ingesting, translating, and extracting from natural language), and especially at volume.

With appropriate guidance (e.g., through prompt engineering, retrieval augmented generation [11], or fine-tuning [12]) LLMs can approach human-level performance on such tasks. Therefore, we wish to explore how effective LLMs can be in specifically populating a KG. In particular, we take this to mean, how well can an LLM extract or otherwise transform unstructured natural language into an output constrained by the ontology (i.e., the schema) for the KG. Succinctly, these research questions are as follows.

RQ1 Are LLMs capable of *effective* KG population?
RQ2 What factors contribute to that effectiveness?
RQ3 Of the available LLMs, which perform best for the KG population task?

To answer these questions, we have selected a KG where we are aware of multiple – significantly different – schemas and data sources for the same data. In particular, we have chosen the Enslaved.Org Hub Ontology, which seeks to (re)construct the narratives of historically enslaved peoples [13] irrespective of their notability or infamy. Much of this data is manually curated from primary or secondary sources [14]. However, in many cases, there is overlap between public data sources (e.g., Wikipedia [15] and Wikidata [16]). Due to the variety of structured and unstructured (i.e., natural language) methods for representing the same data, this provides an opportunity to reasonably assess the ability of LLMs to extract the data and evaluate whether or not that the extracted data is valid for the purposes of populating our manually curated ontology.

The rest of this chapter is organized as follows. Section 2 introduces our case study: the Enslaved.Org Ontology and knowledge graph, as well as how we prepare the data for our experiments. Section 3 provides an overview of our methodology. In Section 4 we present our results and an evaluation thereof. Finally, we discuss related work in Section 6 before concluding in Section 7.

2. Background & Case Study

In this section, we briefly present some preliminary background knowledge that will be useful for understanding the rest of the chapter. In particular, we present definitions for terms and notational and graphical conventions for our figures. We also provide a brief description of the case study that drives our experiment and evaluation.

2.1. Schema Diagrams

From the Modular Ontology Modeling (MOMo) methodology [17], which was used in the development of the Enslaved.org Hub ontology – as appears in our case study, we use a (visual) graphical structure called a schema diagram as our primary method for communicating ontological structure. This diagram carries a reduced semantics; it is meant to be intuitive and easily understood, rather than explicitly and visually conveying the exact, underlying logical axioms. We use a consistent visual syntax across all diagrams.

Boxes of any non-gray color indicate a class. Goldenrod boxes are atomic classes. Purple boxes indicate a class that strictly consists of a controlled set of individuals. Blue boxes (with dashed borders) indicate *hidden complexity* — i.e., that there are additional relations, but which have been removed from view for clarity. Frequently, this means that it represents a class that is drawn from outside of the KWG namespaces (e.g., OWL Time [18]). Large gray boxes that encapsulate many arrows and boxes indicate a module, meaning that they are conceptually related. Yellow ellipses indicate a datatype. These are generally prefixed with the appropriate namespace, for clarity. Filled arrows indicate a binary relation. If one points to a box, then it is an object property. If one points to an ellipse, it is a data property. Open-face arrows indicate a subclass relation.

2.2. Ontology Design Patterns & Modules

Ontology design patterns (ODPs) are tiny, self-contained ontologies that are focused on solving specific modeling problems [19]. They are inspired by software engineering design patterns, and can be applied architecturally (i.e., a way to structure data) or exist on a spectrum from abstract to content-driven applications. ODPs, especially through the MOMo methodology, are *modularized* through a process called *template-based instantiation* [20], which is then followed by *systematic axiomatization* to produce a formalized model from the ODP using a series of 17 frequently used axioms for each node-edge-node construction in the schema diagram for the ODP or module [21]. The use of patterns, by moving from top-level or abstract concepts and subsequently customizing for a specific (narrower) use-case is meant to mimic the human conceptualization process, which in some sense can be viewed as an analogy-driven process [22].

By now, have a healthy ecosystem of various interactions, including libraries of ODPs (e.g., [23,24]), an annotation language for describing pattern-based and modular structure [25,26], and a tool ecosystem for creating pattern-based or modular ontologies [7,27].

Indeed, modular ontologies have a young, but rich history. Examples span many years, from more bespoke project-focused ontologies (e.g., [28]) to the largest public geospatial KG in the world [29,30,31]. Most pertinent for this chapter, is the Enslave.org Hub's modular ontology, which we describe in the next section.

2.3. The Enslaved.org Hub & Ontology

The scourge of African enslavement was fundamental to the making of Europe, Africa, the Americas, and Middle East and parts of the Asian subcontinent. The enduring legacies of black bondage shape the moral questions of humanity in our times. We have seen in the past decade a growth in interest in the subject in film, on television, and in historical fiction. Historians have spilled much ink writing monographs aimed primarily at other scholars. At the same time, however, it is a worthy goal to expand the production of scholarly output and to bring what historians do to the general public.

Recently, there has been a significant shift in perceptions about what we can know about Enslaved Africans, their descendants, and those who asserted ownership over them throughout the world. As a result, a growing number of collections of scanned original manuscript documents, digitized material culture, and databases, that organize and make sense of records of enslavement, are free and readily accessible for scholarly and public consumption. Although this data is available through individual data silos, this proliferation of different projects and databases presents scholars, students, and the interested public with a number of challenges: **(a)** hyper-focused data for individual projects, **(b)** little-to-no consistent disambiguation across projects, **(c)** no data clearinghouse for this sort of data, **(d)** query federation is not supported, **(e)** no established best practices, and **(f)** lack of incentive to publish data.

The Enslaved.Org project introduces a new collaborative model for humanities scholarship by bringing together various professionals to share and expand knowledge about slavery [32,33]. This approach encourages scholars to rethink the way research is produced and shared, aiming to transform both academic practices and historical perspectives on slavery. The technical goal of Enslaved.Org established the Enslaved.Org Hub,a website that provides one-stop querying and inspection capabilities for integrated historic data on the slave trade, originating from a diverse set of data sources and contributors, thereby allowing students, researchers and the general public to search over numerous databases to understand and reconstruct the lives of individuals who were part of the historical slave trade [14].

To address the underlying data integration issues, Enslaved.Org opted to follow the state of the art by establishing a KG equipped with an ontology as a schema. The Enslaved Ontology serves as the underlying schema and data organization paradigm for the Enslaved Hub. It is not used for reasoning or inference, but as a guide for organizing and integrating the data, and understanding the knowledge base as a whole. For additional detail on the modeling approach see [34], and for a thorough examination of the ontology and its core concepts are detailed in [13,35].

2.4. Wikidata & Patterns

Recently, *community-driven KG plaforms* have grown in interest. These are software that hosts a KG that accepts community data from community sources (i.e., data that comes from outside the original development team) and, in general, have a focus on modeling and presenting provenance and lineage of the constituent data. With the growth of the use of KGs, some communities and larger constituencies are being left behind because many of the human-machine interfaces for KGs require more advance technical skills. To address this, the platform, Wikibase, can be deployed to mitigate issues of access for less

technically practiced community members. Wikidata is the pre-eminent public Wikibase installation hosted by the Wikimedia Foundation [16].

Using the Wikibase platform has many advantages: an out of the box software for de-referencing, a convenient user interface for the less technically practiced, a consistent way to track and record provenance and lineage of data, and the option to execute SPARQL queries against an RDF representation of the knowledge graph. However, the provenance mechanism and the exact nature of the structure of the Wikibase representation can complicate developing a principled schema for KGs, as well as the approach to the materialization of the data for upload to the platform.

Indeed, other institutions, such as the European Union (EU), have also come to the conclusion that utilizing Wikibase can serve as the foundation for a community-driven knowledge graph. For instance, the EU Knowledge Graph[3] [36] is deployed on a Wikibase installation, as is the Disability Wiki that serves as an metadata knowledge graph for community documents [37].

Recently, we have created a library of design patterns for Wikibase [38] that allow for facilitated transition from traditional ontology modeling, to the structure that would be expected so as to conform with Wikibase's underlying semantic structure.

3. Knowledge Graph Population Methodology

Our methodology consists of three stages: **(a)** Data Pre-processing, **(b)** Text Retrieval, and **(c)** KG Population. This is followed by an evaluation, which we describe in Section 4.

3.1. Data Pre-processing

This step consists of two sub-steps. First, the data needs to be collected, collated, and curated. Specifically, this means we needed to find relevant natural language text, such as from Wikipedia [15], that covered enslaved persons such that they could be found in Enslaved.org and Wikidata.

These articles needed to be cleaned for formatting. This included removal of headers and citations. Additionally the text taken from Wikipedia had the infobox removed, as exactly that information would be found in Wikidata.

3.1.1. Dataset Collection, Collation, & Curation

This step essentially prepares quality data for our experiments. We took the following actions.

1. First, a list of those individuals that exist on both Wikidata and Enslaved.org needed to be identified.
2. Next, we used string matching between sources to help narrow possible matches before having to manually choose.
3. Then, for each individual where we had data from both sources, we: individual:
   (a) downloaded the Wikitext from Wikidata[4]
   (b) downloaded the TTL file from Enslaved.org[5]

[3] `https://linkedopendata.eu/wiki/The_EU_Knowledge_Graph`
[4] `https://wikidata.org/`
[5] `https://lod.enslaved.org/`

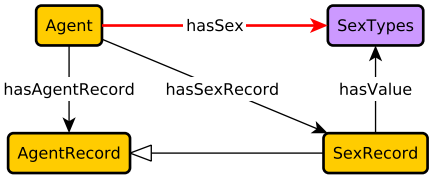

Figure 1. A graphical demonstration of how role chains in the Enslaved Ontology were "collapsed" into simpler shortcuts to generally improve data extraction performance.

(c) read the TTL file into a graph using RDFlib[6]
(d) all triples where individual was the subject were written to a tab-separated value (TSV) file

As we knew that we would be eventually sourcing data from a natural language driven process (i.e., the LLM), we opted for a simpler output structure: the TSV. Specifically, this allowed us to have a minimal syntax mechanism by which we could compare two outputs, without worrying about small errors in syntax.

3.1.2. Module Preparation

For both of the two different schemas, we generated and filtered ontology modules based on schema relationships presented in both the Wikibase [38] and Enslaved [17] ontologies. We focused on extracting and utilizing module information and attributes relevant to the content available in the Wikipedia text files. Modules and attributes that did not provide useful information from the text files were skipped, ensuring that only the relevant data was included during the ontology population process.

For example, in the Wikibase schema, we chose to skip the relation `isDirectly BasedOn(Agent_Information, Reference)`, which typically provides information about the source from which the particular agent information was populated. Since we used Wikidata as the source to populate the ontology, this relation did not necessarily provide meaningful information about the actual source of the data, as it might not point to the original reference material. Similarly, for the Enslaved Ontology, we simplified chained relations to make them easier for the LLMs to capture. For instance, in the Sex Record Module, the original structure is `Agent` has `AgentRecord`, `SexRecord` is-a `AgentRecord`, and `SexRecord hasValue SexTypes`. We simplified this to a single relation: `hasSex(Agent, Sex_Type)`, as shown in Figure 1. Similar simplifications were made for other modules to enhance clarity and usability, ensuring the LLMs could effectively process and understand the relations. These shortcuts are included in our online documentation[7].

3.2. Relevant Text Retrieval

Given the large size of the text files crawled from Wikipedia, which often exceeds the context window (and thus token limit) of LLMs, it was necessary to determine methods for passing only maximally relevant information to the LLM for populating the ontology.

[6] `https://rdflib.readthedocs.io/en/stable/`
[7] `https://github.com/Data-Semantics-Laboratory/LLMDrivenOntologyDev`

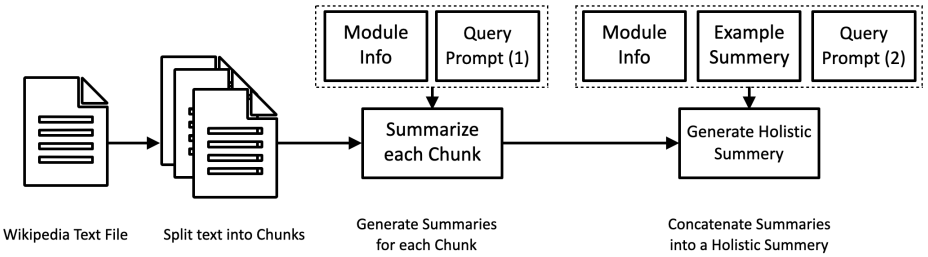

Figure 2. Summarizing the text files using few shot prompting

We used two methods for this purpose: text summarization and RAG, and we tested them both on GPT-4 and Llama-3 with temperature set to zero.

3.2.1. Text Summarization

In the summarization method, LLMs are used to generate concise summaries of the relevant texts from the dataset. These summaries serve as a condensed form of information that can be used in the subsequent ontology population step. We employ a few-shot prompting technique with LLMs to generate summaries that are specifically relevant to the ontology modules, a task that traditional summarization tools cannot accomplish. Figure 2 shows the steps for the summarization process.

Text Chunking: The text files are first split into manageable chunks, ensuring that each chunk stays within the model's token limit. This step involves reading the text file, splitting it into paragraphs, and then concatenating these paragraphs into chunks.

Module Integration: A predefined module file containing relevant schema relationships was used to guide the summarization process. This file was read and incorporated into the prompts fed to the LLM, ensuring that the summaries generated were aligned with the desired ontology structure.

Summary Generation: For each chunk, a query prompt was created that included both the text chunk and the relevant module content. The LLM model was then used to generate a summary that focused on extracting and retaining the most pertinent information based on the predefined modules. We provide our query prompt for the summarization.

> Summarize the following text, by keeping the relevant information from these modules (if any): {module_content}. The birth and death dates are mentioned in parenthesis after the agent name. For example : Joseph Vance Lewis (December 25, 1853 – April 24, 1925), was a slave who was freed. Here the Birth date is December 25, 1853 and Death date is April 24, 1925. So keep those information in the summary. The given text to summarize: {text_chunk}

Holistic Summary Creation with Few-Shot Learning: After generating individual summaries for each chunk, these were concatenated to form a complete context. To ensure consistency and relevance, an example summary was provided as part of the prompt for holistic summary generation, employing few-shot learning. A final holistic summary was then generated, which synthesized the key points from all chunks, ensuring that the summary adhered to the structure and details outlined in the modules and the example summary. We also explicitly mention additional information about certain modules (e.g., the Interagent Relationship Record Module) to help guide the LLM in understanding the

module contents in relation to the ontology, which may not be as straightforward in other cases (e.g., the Sex Record Module). We provide our query prompt for this process.

> Query Prompt: "Here is an example summary that highlights the key points from a text file based on the given modules. The text file discusses a single agent or person who is introduced initially. So, the summary should focus solely on that particular agent. The birth and death dates are mentioned in parenthesis after the agent name. For example : Joseph Vance Lewis (December 25, 1853 – April 24, 1925), was a slave who was freed. Here the Birth date is December 25, 1853 and Death date is April 24, 1925. The Interagent Relationship Record Module describes whether the agent has a relationship with another Enslaver or Owner. The Person Status Module indicates whether the agent is an enslaved person and mentions any status-generating events.
> Example summary: {example_summary}
> Please provide a holistic summary from the given text that follows the format of the example summary and keeps the relevant information from these modules (if any): {module_content}.
> The given text is: {concatenation of summaries from each chunks}"

3.2.2. Retrieval-Augmented Generation

Retrieval Augmented Generation (RAG) is a machine learning approach used in NLP that integrates a retrieval system to improve the performance of a generative model [11].

The core idea is to leverage the strengths of retrieval-based methods and generative models to produce more precise results. In this setup, the relevant information is retrieved based on the user's query from an external knowledge source, such as a database or a knowledge graph. Thus, to implement the RAG mechanism, all extracted text files from Wikipedia are first divided into smaller chunks to ensure efficient processing and then embedded into a vector database. This chunking process helps manage large documents by breaking them into manageable sections, each of which can be individually compared for relevance. Two distinct prompt models are employed during the process: the first model explicitly instructs the system to populate the ontology for the agent, indicating that the document begins with a specific name and that the content is primarily about that entity. The second model, in contrast, omits this specific instruction, allowing for a more general approach to content retrieval.

In the RAG mechanism, the input prompt (query) is also transformed into an embedding, which is then compared against the pre-embedded document chunks. The similarity between the prompt embedding and the document chunk embeddings determines which chunks are most relevant. These selected chunks, based on their high similarity scores, are retrieved and used in conjunction with the original query as input to the language models. This approach enhances the system's ability to generate informed and context-rich responses by integrating relevant document content directly into the model's output generation.

For the implementation of the RAG mechanism, LangChain [8] was employed as the framework to handle the retrieval and augmentation processes. The embeddings used for both the documents and queries are generated using Hugging Face's pre-trained model, specifically the sentence-transformers/all-MiniLM-L6-v2, which is designed to efficiently capture semantic similarities between text chunks. This combination of tools

[8] https://www.langchain.com/

ensures accurate retrieval and seamless integration of relevant information into the language model's output. Besides using GPT-4 (temperature set to zero), we use the open-source LLM "Meta-Llama-3-8B-Instruct" so as to ensure reproducibility, with the temperature set to zero to obtain deterministic results in this approach.

3.3. Populating Ontology Modules

After extracting the relevant texts from each text file, we implemented a systematic approach to populate ontology modules using the extracted information. A predefined module file containing schema relationships guided the ontology population process. This module file was read and integrated into the system, serving as the foundation for structuring the ontology.

A detailed query prompt was then constructed, which included an example format demonstrating how the ontology should be populated based on the information available in the text files, utilizing a few-shot learning approach. The query prompt was explicitly designed, specifying the structure and terminology to be used for each ontology module. For instance, the example format used in the prompt for Age record module would be, hasAgeValue(Agent, xsd:double): hasAgeValue(Absalom Jones, 71).

In general, ontology modules were populated according to the schema relationships for each module and attributes described for that relation in the text files. The prompt also included instructions to skip any relations for which no information was provided in the text files, ensuring that only relevant data was captured.

The LLM was then prompted with the query, the content of the retrieved text files, and the schema modules. This combination allowed the model to accurately extract and structure the information according to the predefined ontology format, ensuring consistency and alignment with the specified schema. Both the Llama-3 and GPT-4 models were employed with a temperature setting of 0 for the prompts to populate the ontologies.

4. Evaluation

4.1. Dataset used

We aim to evaluate the triple generation accuracy of a Large Language Model (LLM) against reference data from Tab-Separated Values (TSV) files. These TSV files are our reference data and include information in triple format (subject, predicate, value) about every individual from the Enslaved ontology.

We organize the data for simpler comparison by first converting the generated text (TXT) files into tab-separated values (TSV) files in order to evaluate the language model's (LLM) performance. The quality of the data produced can be evaluated by comparing it to the truth data that is already in TSV format after the text of the TXT files has been parsed to extract relevant triples. By using a variety of string similarity variables, this comparison allows us to measure the degree to which the produced triples match the reference data.

4.2. Similarity Metrics

To make sure that both typographical errors and semantic differences are properly taken into account when assessing the alignment of characteristics and values between our generated files, we conduct a reconciliation step using a number of string similarity measures identified using guidance from [39].

Cosine Similarity – which calculates the cosine of the angle between two term frequency-inverse document frequency (TF-IDF) vectors – becomes useful [39]. It should be noted that cosine similarity can fail to accurately represent the subtle differences in very short texts or texts with similar structure but different contents. Short texts frequently produce less informative similarity scores and less discriminative TF-IDF vectors.

Fuzzy Matching: Creates similarity ratios that take into consideration possible typos and incomplete matches by using the `rapidfuzz` package [40]. These two ratios are employed:

- **FuzzyWuzzy Ratio:** compares two strings' overall similarity.
- **FuzzyWuzzy Token Set Ratio:** allows for the comparison of strings that have different word ordering by taking into account the token sets of two strings.

Jaro-Winkler Similarity: Calculates the similarity between strings by taking into account shared prefixes and small typographical changes, for the purpose of aligning attributes or values with comparable roots [39].

4.3. Evaluation Methodology

From the original text (TXT) files, we first focus on creating tab-separated values (TSV) files to start the evaluation process. The purpose of this transformation is to organize the data in a way that makes comparison and analysis easier. The output produced by the language model (LLM) is contained in the TXT files, which are our target data—the values we want to match the reference data with. This is to essentially make the evaluation process more accurate since now we will compare pair of files with similar structure.

The relevant triples, which usually consist of predicates and objects that point to the relationships within the data, are first extracted by parsing the text of each TXT file. We arrange the triples in an organized way when they have been detected, making sure that a tab character separates every element. We can preserve data readability with this setup.

In order to compare the truth data with the generated triples, pairs of files—one storing the reference data and the other containing the language model (LLM) output—are evaluated. To evaluate the degree of similarity between related entries in the truth and created datasets, we use a variety of string similarity measures for each pair, including fuzzy matching and cosine similarity. Iteratively going over each triple in the truth file and comparing it to the matching triple in the produced file, scores are determined based on how closely they match.

After that, we define a set of similarity thresholds to determine whether triples that are produced match the truth data sufficiently. Checking to see if any of the similarity metrics for both predicates and objects—exceed these predetermined thresholds is part of the evaluation process. Identifying unique truth predicates and objects enables us to calculate the total number of unique comparisons. The percentage of good matches in comparison to the total number of unique triples is subsequently calculated by counting the unique good matches that reach or exceed the specified thresholds.

Table 1. Similarity matching between triples extracted by the LLM models plus prompting strategy, reporting the average and total coverage for each module and in aggregate (columns with subscript *A*). The #F column indicates the number of modules for which triple extraction entirely failed (i.e., a coverage of 0).

LLM Model	**Avg %**	**Ttl %**	**#F**	**Avg$_A$ %**	**Ttl$_A$ %**
GPT4_RAG_Enslaved_MainAgent	82.30	81.60	14	88.55	88.09
GPT4_RAG_Enslaved_NotRestrictedToMainAgent	87.87	86.82	4	89.11	88.69
GPT4_RAG_WB_MainAgent	77.08	76.10	25	88.02	87.63
GPT4_RAG_WB_NotRestrictedToMainAgent	85.03	84.12	7	87.69	87.34
GPT-4_Summarization_Enslaved	81.16	80.86	0	81.16	80.86
GPT-4_Summarization_WB	82.60	82.03	0	82.60	82.03
Llama3_RAG_WB_MainAgent	71.17	70.88	6	73.64	73.20
Llama3_RAG_WB_NotRestrictedToMainAgent	71.25	70.63	2	71.92	71.41

The performance of our produced outputs was then assessed by determining the average, minimum, and maximum percentages of good matches for each llm-output comparison. This analysis helped us assess the overall quality of our results by revealing how well the created triples aligned with the reference data.

Furthermore, from every file and model, we extracted the highest values of the similarity metric.Understanding the top-performing data allowed us to identify the areas in which our models performed really well while also looking into weaknesses.

5. Results and Discussion

The LLM evaluation results are displayed in Table 1. For each LLM model and prompting strategy, extraction strategy, and source data, we evaluate how well the extracted triples match our ground truth. This is reported as *coverage* per module and overall. A triple is considered to be covered if an extracted triple has a similarity 80% in any of the metrics reported in Section 3.[9] This covers small typos in the text, but also small variants in naming. The most common example is the inclusion of a middle initial, but the rest of the extracted triple matches the ground truth.

Each LLM model had at least one module with 100% coverage, except for Llama3_RAG_WB_NotRestrictedToMainAgent, which had only a maximum of 90.1%. In this case, 100% indicates that all triples had an appropriate match. However, each experiment that did not use the summarization technique had modules where the model failed to extract any relevant triples, the LLM model refused to complete the task, or for some other inscrutable reason did not provide a list of triples appropriate for extraction and mapping.

Overall, we see the best performance from GPT-4, basically irrespective of the prompting strategy utilized.

For example, GPT-4_Summarization_Enslaved has the lowest average and total coverage and performs better than either of the experiments conducted with Llama models. Interestingly, extraction of triples according to the Wikibase model had highest fail rate, but still overall had higher coverage than the Llama models.

[9] It is generally considered to be a "good" match if a similarity metric returns 70% or better. We tend to a stricter threshold.

Generally, we see that the extraction of triples according to a schema is generally effective, approaching 90% coverage when the model performs the task.

6. Related Work

We have selected the below related research since it is consistent with our goal of automating ontology engineering tasks by utilizing large language models and solving issues like inaccurate development of concepts. The research articles brought light on the current issues surrounding ontology building, particularly those related to scalability and domain-specific accuracy. These publications also point to limitations in ontology population methods that we hope to fill with more accurate and organized data extraction techniques.

6.1. Ontology Engineering and Construction

Ontology Engineering with Large Language Models (2023): Large language models (LLMs) are investigated here as a potential tool for automating ontology engineering activities including relationship detection and entity extraction. Although LLMs have the potential to reduce manual tasks, they frequently provide concepts that are inaccurate or too broad, which can lead to errors in complicated domains [41].

We overcome these weaknesses in our approach by using module-guided summarization and few-shot prompting to make sure that the output is properly designed to meet the requirements and structure of the ontology. We give the model appropriate examples and schema-driven prompts, which guide it to focus on important relationships and entities, in contrast to general LLM approaches that might produce ambiguous notions. We reduce the possibility of excessive generalization and unimportant information through incorporating the predefined modules into the summary process and making sure that it is in line with the ontology's schema. This makes the results that are produced more accurate and appropriate for demanding ontology tasks.

Towards Ontology Construction with Language Models (2023): The goal of this work is to expedite the process of ontology generation by exploring the use of language models to automatically generate ontologies from text. Although the method works well, it has trouble guaranteeing that the generated ontologies are accurate and complete as well as catching details unique to a given domain [42].

On the other hand, our method focuses on carefully populating particular areas of the ontology by extracting relevant information and providing module-guided summaries, instead of automatically creating full ontologies. We maintain the accuracy and contextual relevance of the populated ontology by including the ontology schema into the summarization process and using structured prompts to target relevant content. This approach provides domain-specific relevance and minimizes errors in difficult domains by prioritizing exact text retrieval and alignment with the predetermined structure, so avoiding the frequent issues associated with broad or partial ontology development.

OLAF: An Ontology Learning Applied Framework (2023): With a focus on collecting and organizing information from massive datasets, OLAF presents a system for automating ontology learning. Although the system has its advantages, its scalability issues and

potential for poor performance when dealing with noisy or unstructured data restrict its use in a variety of real-world situations [43].

By combining retrieval-based techniques with text chunking, our strategy, on the other hand, directly addresses the problem of large-scale data and the potential noisiness that might occur. To make sure the LLM uses important, organized content, we divide large data sets into manageable portions that fit within the token constraints of language models. Retrieval-Augmented Generation (RAG) and summarization are two other techniques that let us restrict our focus to the most important data. Targeted retrieval handles capacity concerns that systems like OLAF have by guaranteeing that, even when processing massive or unstructured data, the content used for ontology population remains accurate, relevant, and in line with the particular ontology structure.

6.2. Ontology Evaluation and Alignment

Automated Ontology Evaluation: Evaluating Coverage and Correctness using a Domain Corpus (2023): Using domain-specific databases, this research suggests an automated approach to assess the reliability and scope of ontologies. The method may not fully capture contextual significance or domain-specific details, and it is constrained by the quality of the input data, although offering a robust review [44].

Instead, in order to guarantee accuracy and relevance, our method concentrates on the ontology population. We prioritize domain-specific details to be directly included into the ontology during development by utilizing module-guided summarization and few-shot prompting. Our methodology involves domain-specific details during the population phase, resulting in a more contextually correct and relevant ontology, whereas this paper's method focuses on evaluating an ontology's scope and trustworthiness after it has been produced, frequently missing key details.

String Similarity Metrics for Ontology Alignment (2022): The paper addresses ontology alignment using string similarity metrics, which is essential for connecting different ontologies. These measures increase alignment accuracy, but their use in tricky alignments is limited since they may have difficulty with semantic mismatches and lack taking context for words into account [39].

6.3. Knowledge Graphs and Language Models

Large Language Models and Knowledge Graphs: Opportunities and Challenges (2023): With the goal of enhancing knowledge extraction and reasoning, this research investigates possibilities for collaboration between large language models and knowledge graphs. It does, however, draw attention to issues with scalability, model interpretability, and the requirement for high-quality data in order to avoid mistakes in knowledge graphs [45].

We address data quality and interpretability issues early by organizing the ontology population process with precise, domain-specific summaries, guaranteeing that the ontology is provided with relevant and accurate content right from the start.

TEXT2KGBENCH: A Benchmark for Ontology-Driven Knowledge Graph Generation from Text (2023): TEXT2KGBENCH presents a benchmark for assessing ontology-driven methods for text-to-knowledge graph generation. Though its reliance on partic-

ular ontologies restricts its applicability to other domains and methods of knowledge representation, the benchmark standardizes evaluation measures [46].

RAG is useful in our situation because it facilitates the extraction of relevant information from huge data sets, which is subsequently used to produce content that is directly related to the ontology. This method minimizes problems associated with large-scale data processing and maintains high-quality information by ensuring that only the most pertinent data is used.

6.4. Knowledge Extraction and Prompt Engineering

SPIRES: Structured Prompt Interrogation and Recursive Extraction of Semantics (2023): Using structured prompts for recursive semantic extraction, SPIRES offers a zero-shot learning approach for knowledge base population. The method works well, but it is constrained by the prompt design's quality and the model's comprehension of complicated or difficult ideas [47].

Testing Prompt Engineering Methods for Knowledge Extraction from Text (2023): This research uses language models to assess different prompt engineering techniques for knowledge extraction from text. The study highlights the benefit of different prompts but also points out their inconsistent aspects and the need for domain knowledge in order to create well-designed queries [48].

6.5. Evaluation Metrics for Text and Knowledge Extraction

BERTScore: Evaluating Text Generation with BERT (2019): A more sophisticated evaluation than previous techniques, BERTScore presents a metric for text output quality using BERT embeddings. However, the measure is susceptible to the selection of pre-trained BERT models and might not completely encompass semantic variations in produced text, particularly in synthetic or non-literal settings [49].

7. Conclusion

Knowledge graphs, are by now, utilized across many domains and across enterprise and academia [1]. They are supported by a healthy and mature ecosystem of W3C standards [3,4,5,6] and tool ecosystems (e.g., Protégé or GraphDB). They excel at the integration of heterogeneous datasets and perspective by allowing for semantic harmonization. Yet, when encountering new data, mapping into an existing ontology can be quite expensive. This might be due to the sheer volume of data to be interpreted.

In this chapter, we have presented a mechanism for examining the efficacy of utilizing large language models to do the semi-automated population of a modular ontology. This pipeline uses both LLM-powered text summarization and retrieval-augmented generation processes for conducting ontology-guided information distillation from natural language into forms more amenable to limited context windows and translation into triples to be loaded into a KG.

In our evaluation, we have shown that extraction of appropriate triples, as guided by a schema (and specifically in the case of our experiments: a modular ontology) is effective, approaching 90% coverage when the LLM performs the task. Performance is

similarly high agnostic to the prompting strategy or mechanism to fit the text into the models' context windows. As such, we conclude that LLMs are a sufficient tool for the extraction of triples from text when guided by a (modular) schema via prompting.

Future Work. These experiments are preliminary, insofar that they demonstrate feasibility. The evaluation is comparatively shallow, demonstrating coverage in a naive way. Since we are comparing text to ground truth, which was manually curated not necessarily from the text at hand (e.g., other sources), it is not known if failure to extract a triple at a high enough similarity is due to the fact that that information was missing in the first place, or if the LLM somehow missed a particular fact.

As such, further experiments must be conducted. We envision some next steps to answer the following questions.

- Does this LLM triple extraction process perform better than a human? We know that it is certainly *faster*, but does the increase in speed outweigh missing or incorrect facts?
- What graph structures (i.e., data organized in which ontological formalisms) are most easily extracted to from textual data? Experiments demonstrate that the modular ontology originally developed for The Enslaved.org Hub performs better, but what exact characteristics facilitate that increase in coverage?

Acknowledgements The authors acknowledge partial funding under National Science Foundation grants 2333532 and 2333782, and from Kansas State University's Game Changing Research Initiative (GRIP) program.

References

[1] Hitzler P. A review of the semantic web field. Commun ACM. 2021;64(2):76-83. Available from: https://doi.org/10.1145/3397512.

[2] Noy NF, Gao Y, Jain A, Narayanan A, Patterson A, Taylor J. Industry-scale knowledge graphs: lessons and challenges. Commun ACM. 2019;62(8):36-43. Available from: https://doi.org/10.1145/3331166.

[3] Hitzler P, Krötzsch M, Parsia B, Patel-Schneider PF, Rudolph S, editors. OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation 11 December 2012; 2012. Available from http://www.w3.org/TR/owl2-primer/.

[4] Cyganiak R, Wood D, Lanthaler M, editors. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014; 2014. Available from http://www.w3.org/TR/rdf11-concepts/.

[5] Baker T, Prud'hommeaux E, editors. Shape Expressions (ShEx) 2.1 Primer. Final Community Group Report 09 October 2019; 2019. Http://shex.io/shex-primer/index.html.

[6] Knublauch H, Kontokostas D, editors. Shapes Constraint Language (SHACL). W3C Recommendation 20 July 2017; 2017. Https://www.w3.org/TR/shacl/.

[7] Shimizu C, Hammar K, Hitzler P. Modular Graphical Ontology Engineering Evaluated. In: Harth A, Kirrane S, Ngomo AN, Paulheim H, Rula A, Gentile AL, et al., editors. The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings. vol. 12123 of Lecture Notes in Computer Science. Springer; 2020. p. 20-35. Available from: https://doi.org/10.1007/978-3-030-49461-2_2.

[8] Ham K. OpenRefine (version 2.5). Journal of the Medical Library Association. 2013 Jul;101(3):233-4. Copyright: © 2013, Authors. Available from: http://openrefine.org.

[9] Yang J, Jin H, Tang R, Han X, Feng Q, Jiang H, et al. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. ACM Trans Knowl Discov Data. 2024 apr;18(6). Available from: https://doi.org/10.1145/3649506.

[10] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan

SVN, et al., editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA; 2017. p. 5998-6008. Available from: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[11] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems. 2020;33:9459-74.

[12] Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, et al. LoRA: Low-Rank Adaptation of Large Language Models. CoRR. 2021;abs/2106.09685. Available from: https://arxiv.org/abs/2106.09685.

[13] Shimizu C, Hitzler P, Hirt Q, Rehberger D, Estrecha SG, Foley C, et al. The enslaved ontology: Peoples of the historic slave trade. J Web Semant. 2020;63:100567. Available from: https://doi.org/10.1016/j.websem.2020.100567.

[14] Enslaved.org Hub;. https://enslaved.org/.

[15] Wikipedia;. https://wikipedia.org/.

[16] Wikidata;. https://wikidata.org/.

[17] Shimizu C, Hammar K, Hitzler P. Modular ontology modeling. Semantic Web. 2023;14(3):459-89. Available from: https://doi.org/10.3233/SW-222886.

[18] Little C, Cox S. Time Ontology in OWL. W3C; 2017. Https://www.w3.org/TR/2017/REC-owl-time-20171019/.

[19] Gangemi A, Presutti V. Ontology Design Patterns. In: Staab S, Studer R, editors. Handbook on Ontologies. International Handbooks on Information Systems. Springer; 2009. p. 221-43. Available from: https://doi.org/10.1007/978-3-540-92673-3_10.

[20] Hammar K, Presutti V. Template-Based Content ODP Instantiation. In: Hammar K, Hitzler P, Krisnadhi A, Lawrynowicz A, Nuzzolese AG, Solanki M, editors. Advances in Ontology Design and Patterns [revised and extended versions of the papers presented at the 7th edition of the Workshop on Ontology and Semantic Web Patterns, WOP@ISWC 2016, Kobe, Japan, 18th October 2016]. vol. 32 of Studies on the Semantic Web. IOS Press; 2016. p. 1-13. Available from: https://doi.org/10.3233/978-1-61499-826-6-1.

[21] Eberhart A, Shimizu C, Chowdhury S, Sarker MK, Hitzler P. Expressibility of OWL Axioms with Patterns. In: Verborgh R, Hose K, Paulheim H, Champin P, Maleshkova M, Corcho Ó, et al., editors. The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings. vol. 12731 of Lecture Notes in Computer Science. Springer; 2021. p. 230-45. Available from: https://doi.org/10.1007/978-3-030-77385-4_14.

[22] Hitzler P, Shimizu C. Modular Ontologies as a Bridge Between Human Conceptualization and Data. In: Chapman P, Endres D, Pernelle N, editors. Graph-Based Representation and Reasoning - 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK, June 20-22, 2018, Proceedings. vol. 10872 of Lecture Notes in Computer Science. Springer; 2018. p. 3-6. Available from: https://doi.org/10.1007/978-3-319-91379-7_1.

[23] Shimizu C, Hirt Q, Hitzler P. MODL: A Modular Ontology Design Library. In: Janowicz K, Krisnadhi AA, Villalón MP, Hammar K, Shimizu C, editors. Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019. vol. 2459 of CEUR Workshop Proceedings. CEUR-WS.org; 2019. p. 47-58.

[24] Eells A, Dave B, Hitzler P, Shimizu C. Commonsense Ontology Micropatterns. In: Besold TR, d'Avila Garcez A, Jiménez-Ruiz E, Confalonieri R, Madhyastha P, Wagner B, editors. Neural-Symbolic Learning and Reasoning - 18th International Conference, NeSy 2024, Barcelona, Spain, September 9-12, 2024, Proceedings, Part II. vol. 14980 of Lecture Notes in Computer Science. Springer; 2024. p. 51-9. Available from: https://doi.org/10.1007/978-3-031-71170-1_6.

[25] Hitzler P, Gangemi A, Janowicz K, Krisnadhi AA, Presutti V. Towards a Simple but Useful Ontology Design Pattern Representation Language. In: Blomqvist E, Corcho Ó, Horridge M, Carral D, Hoekstra R, editors. Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.. vol. 2043 of CEUR Workshop Proceedings. CEUR-WS.org; 2017. Available from: http://ceur-ws.org/Vol-2043/paper-09.pdf.

[26] Hirt Q, Shimizu C, Hitzler P. Extensions to the Ontology Design Pattern Representation Language. In: Janowicz K, Krisnadhi AA, Poveda-Villalón M, Hammar K, Shimizu C, editors. Proceedings of the 10th

Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019. vol. 2459 of CEUR Workshop Proceedings. CEUR-WS.org; 2019. p. 76-5. Available from: http://ceur-ws.org/Vol-2459/short2.pdf.

[27] Shimizu C, Hirt Q, Hitzler P. A Protégé Plug-In for Annotating OWL Ontologies with OPLa. In: Gangemi A, Gentile AL, Nuzzolese AG, Rudolph S, Maleshkova M, Paulheim H, et al., editors. The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers. vol. 11155 of Lecture Notes in Computer Science. Springer; 2018. p. 23-7. Available from: https://doi.org/10.1007/978-3-319-98192-5_5.

[28] Eberhart A, Shimizu C, Stevens CA, Hitzler P, Myers CW, Maruyama B. A Domain Ontology for Task Instructions. In: Villazón-Terrazas B, Ortiz-Rodríguez F, Tiwari SM, Shandilya SK, editors. Knowledge Graphs and Semantic Web - Second Iberoamerican Conference and First Indo-American Conference, KGSWC 2020, Mérida, Mexico, November 26-27, 2020, Proceedings. vol. 1232 of Communications in Computer and Information Science. Springer; 2020. p. 1-13. Available from: https://doi.org/10.1007/978-3-030-65384-2_1.

[29] Janowicz K, Hitzler P, Li W, Rehberger D, Schildhauer M, Zhu R, et al. Know, Know Where, Knowwheregraph: A Densely Connected, Cross-Domain Knowledge Graph and Geo-Enrichment Service Stack for Applications in Environmental Intelligence. AI Mag. 2022;43(1):30-9. Available from: https://doi.org/10.1609/aimag.v43i1.19120.

[30] Shimizu C, Stephen S, Barua A, Cai L, Christou A, Currier K, et al. The KnowWhereGraph Ontology. Journal of Web Semantics. 2023. Under review.

[31] Shimizu C, Stephen S, Zhu R, Currier K, Schildhauer M, Rehberger D, et al. The KnowWhereGraph Ontology: A Showcase. In: Toyoshima F, Katsumi M, Righetti G, Giorgis SD, Hedblom MM, Kutz O, et al., editors. Proceedings of the Joint Ontology Workshops 2023 Episode IX: The Quebec Summer of Ontology co-located with the 13th International Conference on Formal Ontology in Information Systems (FOIS 2023), Sherbrooke, Québec, Canada, July 19-20, 2023. vol. 3637 of CEUR Workshop Proceedings. CEUR-WS.org; 2023. Available from: https://ceur-ws.org/Vol-3637/paper46.pdf.

[32] A massive new effort to name millions sold into bondage during the transatlantic slave trade;. https://www.washingtonpost.com/history/2020/12/01/slavery-database-family-genealogy/.

[33] Computer Science Professor, Postdoc Launch Online Database on History of Slavery;. https://cacm.acm.org/news/249167-computer-science-professor-postdoc-launch-online-database-on-history-of-slavery/fulltext?mobile=false.

[34] Shimizu C, Hitzler P, Estrecha SG, Goeke-Smith J, Rehberger D, Foley C, et al. The Wikibase Approach to the Enslaved.Org Hub Knowledge Graph. In: Payne TR, Presutti V, Qi G, Poveda-Villalón M, Stoilos G, Hollink L, et al., editors. The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part II. vol. 14266 of Lecture Notes in Computer Science. Springer; 2023. p. 419-34. Available from: https://doi.org/10.1007/978-3-031-47243-5_23.

[35] Shimizu C, Hitzler P, Hirt Q, Sheill A, Gonzalez S, Foley C, et al.. The Enslaved Ontology 1.0: People of the Historic Slave Trade; 2019. Available from https://daselab.cs.ksu.edu/projects/ontology-modeling-slave-trade.

[36] Diefenbach D, Wilde MD, Alipio S. Wikibase as an Infrastructure for Knowledge Graphs: The EU Knowledge Graph. In: Hotho A, Blomqvist E, Dietze S, Fokoue A, Ding Y, Barnaghi PM, et al., editors. The Semantic Web – ISWC 2021 - 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24-28, 2021, Proceedings. vol. 12922 of Lecture Notes in Computer Science. Springer; 2021. p. 631-47.

[37] Bisen KS, Alemayehu SA, Maret P, Creighton A, Gorman R, Kundi B, et al. Wikibase as an Infrastructure for Community Documents: The example of the Disability Wiki Platform. In: Simsek U, Chaves-Fraga D, Pellegrini T, Vahdat S, editors. Proceedings of Poster and Demo Track and Workshop Track of the 18th International Conference on Semantic Systems co-located with 18th International Conference on Semantic Systems (SEMANTiCS 2022), Vienna, Austria, September 13th to 15th, 2022. vol. 3235 of CEUR Workshop Proceedings. CEUR-WS.org; 2022. Available from: http://ceur-ws.org/Vol-3235/paper14.pdf.

[38] Shimizu C, Eells A, Gonzalez S, Zhou L, Hitzler P, Sheill A, et al. Ontology design facilitating Wikibase integration — and a worked example for historical data. Journal of Web Semantics. 2024;82:100823. Available from: https://www.sciencedirect.com/science/article/pii/

S157082682400009X.

[39] Cheatham M, Hitzler P. String Similarity Metrics for Ontology Alignment. In: Alani H, Kagal L, Fokoue A, Groth P, Biemann C, Parreira JX, et al., editors. The Semantic Web – ISWC 2013. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013. p. 294-309.

[40] Bosker HR. Using fuzzy string matching for automated assessment of listener transcripts in speech intelligibility studies. Behavior Research Methods. 2021 October;53(5):1945-53. Epub 2021 Mar 10. Available from: https://tokensortratio.netlify.app.

[41] Mateiu P, Groza A. Ontology engineering with Large Language Models; 2023. p. 226-9.

[42] Funk M, Hosemann S, Jung JC, Lutz C. Towards Ontology Construction with Language Models; 2023. Available from: https://arxiv.org/abs/2309.09898.

[43] Schaeffer M, Sesboüé M, Kotowicz JP, Delestre N, Zanni-Merk C. OLAF: An Ontology Learning Applied Framework. Procedia Computer Science. 2023;225:2106-15. 27th International Conference on Knowledge Based and Intelligent Information and Engineering Sytems (KES 2023). Available from: https://www.sciencedirect.com/science/article/pii/S1877050923013595.

[44] Zaitoun A, Sagi T, Hose K. Automated Ontology Evaluation: Evaluating Coverage and Correctness using a Domain Corpus. In: Companion Proceedings of the ACM Web Conference 2023. WWW '23 Companion. New York, NY, USA: Association for Computing Machinery; 2023. p. 1127–1137. Available from: https://doi.org/10.1145/3543873.3587617.

[45] Pan JZ, Razniewski S, Kalo JC, Singhania S, Chen J, Dietze S, et al.. Large Language Models and Knowledge Graphs: Opportunities and Challenges; 2023. Available from: https://arxiv.org/abs/2308.06374.

[46] Mihindukulasooriya N, Tiwari S, Enguix CF, Lata K. Text2KGBench: A Benchmark for Ontology-Driven Knowledge Graph Generation from Text; 2023. Available from: https://arxiv.org/abs/2308.02357.

[47] Caufield JH, Hegde H, Emonet V, Harris NL, Joachimiak MP, Matentzoglu N, et al.. Structured prompt interrogation and recursive extraction of semantics (SPIRES): A method for populating knowledge bases using zero-shot learning; 2023. Available from: https://arxiv.org/abs/2304.02711.

[48] Polat F, Tiddi I, Groth P. Testing Prompt Engineering Methods for Knowledge Extraction from Text. Submission type: Full Paper. 2024. Tracking #: 3719-4933. Available from: https://www.semantic-web-journal.net/system/files/swj3719.pdf.

[49] Zhang T, Kishore V, Wu F, Weinberger KQ, Artzi Y. BERTScore: Evaluating Text Generation with BERT; 2020. Available from: https://arxiv.org/abs/1904.09675.