Commonsense Ontology Micropatterns

Andrew Eells[1(✉)], Brandon Dave[2], Pascal Hitzler[1], and Cogan Shimizu[2]

[1] Kansas State University, Manhattan, KS, USA
andreweells@ksu.edu
[2] Wright State University, Dayton, OH, USA

Abstract. The previously introduced Modular Ontology Modeling methodology (MOMo) attempts to mimic the human analogical process by using modular patterns to assemble more complex concepts. To support this, MOMo organizes ontology design patterns (ODPs) into design libraries, which are programmatically queryable. However, a major bottleneck to large-scale deployment of MOMo is the (to-date) limited availability of ready-to-use ODPs. At the same time, Large Language Models (LLMs) have quickly become a source of common knowledge and, in some cases, replacing search engines for questions. In this paper, we thus present a collection of 104 ODPs representing often occurring nouns, curated from the common-sense knowledge available in LLMs, organized into a fully-annotated modular ontology design library ready for use with MOMo.

1 Introduction

Humans perform transfer learning through analogous connections in understanding the unknowns of the world with previously acquired knowledge [6,7]. For problem-solving, humans find similarities to prior knowledge and draw connections that aid in finding a working solution, which, regardless of success, becomes part of the learning process for the next iteration of problem-solving. The task of drawing similarities requires humans to use their logical skills for generalizing a problem or concept to allow for adaptations from their prior knowledge.

Neurosymbolic Artificial Intelligence (AI) systems, in their pursuit of General AI, also seek this ability to conduct transfer and analogy learning. We posit that the use of modular ontologies is one way that this can be accomplished [11]. In particular, the notion of starting with a simple and (perhaps) well-understood pattern of knowledge and adapting it to new scenarios and use-cases; the mechanism by which we accomplish this we call *modularization*. This overarching methodology is known as Modular Ontology Modeling (MOMo), and it is by now an established paradigm for constructing knowledge graphs (KG) and their schemas (i.e., an ontology) [14].

Ontology design patterns (ODPs), which are tiny ontologies that are intended to solve domain-invariant modeling problems [5], are the usual starting point

T. R. Besold et al. (Eds.): NeSy 2024, LNAI 14980, pp. 51–59, 2024.
https://doi.org/10.1007/978-3-031-71170-1_6

for MOMo. They may be sourced from many places, including the ODP Portal[1] or Modular Ontology Design Libraries (MODLs) [15]. MODLs are annotated, curated collections of ODPs, combined with an index ontology that can be queried for patterns of different types and relationships between patterns.

In this paper, we provide such a design library, which we call CS-MODL, composed of *commonsense (ontology) micropatterns*. By **commonsense** By commonsense we mean highly occurring, context-independent concepts with broad usefulness. In this case, we ask an LLM about common properties of common nouns (see below) and receive some simple and naïve responses, which are then encoded into **micropatterns.** We contrast micropatterns versus ODPs. In this case, micropatterns do not have a sophisticated or rich semantics. Indeed, they are defined using rdfs:domain and rdfs:range, without more complicated OWL[?] axioms. Concretely, we describe the methodology we used to construct the 104 commonsense micropatterns contained in CS-MODL, discuss an example, and provide the library as an online resource.

The rest of the paper is organized as follows: Sect. 2 presents our approach for obtaining the appropriate information from our chosen LLM and constructing the commonsense micropatterns. Section 3 summarizes the contributions of our commonsense library of patterns. Section 4 wraps up the paper with notes on future areas we observe as next steps to this research.

2 Methodology

The construction of our commonsense patterns was done in two steps. First, we prompted an LLM in several ways to produce an ontology design pattern for each noun in question. Then, the responses were consolidated into patterns via a set of heuristics, and the patterns are packaged into a MODL.

Large Language Models as Commonsense. LLMs, like GPT-4, ingest and learn from more information than any human possibly could. While it is certainly arguable that whether LLM demonstrates human-level intelligence, it is also inarguable that it can present a commonsense view of the world – in particular through a declarative memory, having stored, perhaps as an accident of learning from "mostly correct" language, facts about the world [2]. They are able to represent this information and "knowledge" in human-readable ways with surprising fluency, and, as we demonstrate below, frequently in valid RDF [3]. The following sections describe how we decided on which facts to extract (i.e., which commonsense notions to prompt), how we prompted for them, and the resultant collection of patterns.

Noun Selection. The nouns selected are the 101 most common nouns in American English (see Fig. 2 in the Appendix), as listed in the Corpus of Contemporary American English (COCA) [1], which include Air, Book, Child, Morning, Room, Office, System. We additionally include the nouns "chair", "sofa", and "loveseat" from our preliminary explorations for viability of our idea.

[1] https://ontologydesignpatterns.org/.

Prompt Engineering. We prompted the LLM (for this work, we used GPT-4-0613) in several different ways, as we wished to test how it would respond to multiple variants of the same prompts. The initial prompts were a command (1) and a request (2), and then variations were added to include the noun as "the following" (3 and 4), as seen below.

1. Generate an ontology that covers [noun].
2. Could you develop a basic design pattern for representing [noun] in an ontology?
3. Generate an ontology that covers the following: [noun].
4. Could you develop a basic design pattern for representing the following in an ontology: [noun].

Finally, the LLM was prompted to generate 9 additional variations of each prompt. See below for Prompt 1's variations. In each case, the prompt was tried with and without the additional instruction of "Provide it in valid Turtle/RDF format, excluding any extra text." for a total of 20 prompts per noun.

1. Generate an ontology that covers [noun].
2. Develop an ontology dedicated to [noun].
3. Construct an ontology focused on [noun].
4. Build an ontology surrounding the concept of a [noun].
5. Formulate an ontology related to [noun].
6. Establish an ontology based on [noun].
7. Design an ontology to encompass [noun].
8. Produce an ontology specifically for [noun].
9. Compose an ontology to represent [noun].
10. Make an ontology that pertains to [noun].

Response Consolidation. The total of $4 \times 20 \times 104$ LLM-generated responses were stored in a tab-delimited (TSV) file. However, not every response contained valid RDF and some contained no RDF at all. The number of responses was too large for manual processing, so we developed a script that would apply heuristics to each response file in order to integrate the responses into a single micropattern. The script consists of three parts, applied to each noun:

1. **extraction** – for each specific prompt response (i.e., a row in the TSV), the script attempts to identify and extract the portion of the response in RDF. Unfortunately, not every response output easily recognizable RDF. Generally speaking, cleaning required the removal of extra text (even when prompted *not* to provide extra text) and surrounding Markdown formatting (e.g., code fencing).
2. **cleaning** – a set of heuristics was applied to each extracted RDF (generally in turtle format) to allow for ingestion of the RDF into an rdflib[2] graph structure. The most common type of cleaning necessary was the usage of newline characters. The LLM understood how to comment (i.e., separating

[2] https://rdflib.readthedocs.io/en/stable/.

class declarations from properties from assertions) but did not understand new lines. This frequently resulted in malformed Turtle, since the first part of the triple would be commented out.
3. **integration** – once each valid (or cleanable) TTL file was extracted from the response, we combined the properties (removing all triples pertaining to individuals), and annotated the result, giving us a commonsense micropattern for the noun. The annotations are further discussed in Sect. 3.

It is necessary to recall that the process by which ODPs are used in the MOMo process is based on customization to the use-case at hand; that is properties are frequently removed, added, or renamed for best fit. This process is known as *template-based instantiation* [8], and it is a core aspect of the methodology. Specifically, each property with (explicit or inferred) domain and range, is assessed for fit. If it is not applicable (i.e., out of scope or too specific or general), the property and associated sub-graphs are removed. Conversely, if there are parts of the use-case that are not covered by the ODP, these must be added. In automated processes, it is generally easier to pare down an overly superfluous pattern. That is, if there are no matches or fillers for the slots that a pattern provides, they should be dropped. We follow this strategy for our micropatterns, meaning that we include each property from every response, which includes variants of the same property. For example, in Fig. 1, which shows a schema diagram of a commonsense micropattern for `Air`, hasHumidity has a range of both Humidity and xsd:float. It is conceivable that one or the other is unnecessary, or that they may be used in tandem (i.e., where a connection from Humidity is made to the xsd:string), e.g., as a type of shortcut or view [13,17].

Yet, this strategy may not be applicable for all systems. As such, our script includes a voting mechanism for integrating multiple responses from different prompts or LLMs, thus allowing a predictable manner for integrating multiple patterns sources to product the final micropattern.

The voting mechanism is as follows. For the usable response, we extracted 3-tuples expressing properties associated with a noun, and the property's respective domain and range (e.g., (p, d, r)). Then, some threshold t is defined where relations that appear a number of times below that threshold are not included. One variant would be to consider the (p, d, r)-tuples as unique, rather than just p. In our case, the commonsense micropatterns had limited variance (i.e., they mostly agreed on content of the patterns) and, as such, we did not invoke the voting mechanism for our results. This yields more extensive patterns, but they can then be easily pared down when following MOMo.

Finally, when generating the micropattern, we minimally describe the properties and classes. That is, we use only rdfs:Class, rdfs:subClassOf, rdfs:range, and rdfs:domain. The script, however, is designed in such a way to be easily modifiable to improve the semantic expressivity of the patterns. For example, we could use a simple heuristics, plus a set of axiom patterns (i.e., taken from [4]), to assert the most common types of constraints for a particular property.

Example Pattern. For an example pattern, `Air`, we provide a schema diagram in Fig. 1. The gold boxes reflect classes. The yellow ellipses indicate datatypes. Black

filled arrows are binary relations (called properties). As noted earlier, there is some reuse of property names (e.g., hasHumidity) that would technically be not allowed in OWL (i.e., use in an object and data property). However, we suggest that *pruning* is easier than growing, so a system that utilizes CS-MODL can simply remove the inappropriate. The turtle file for this example can be found in the repository.

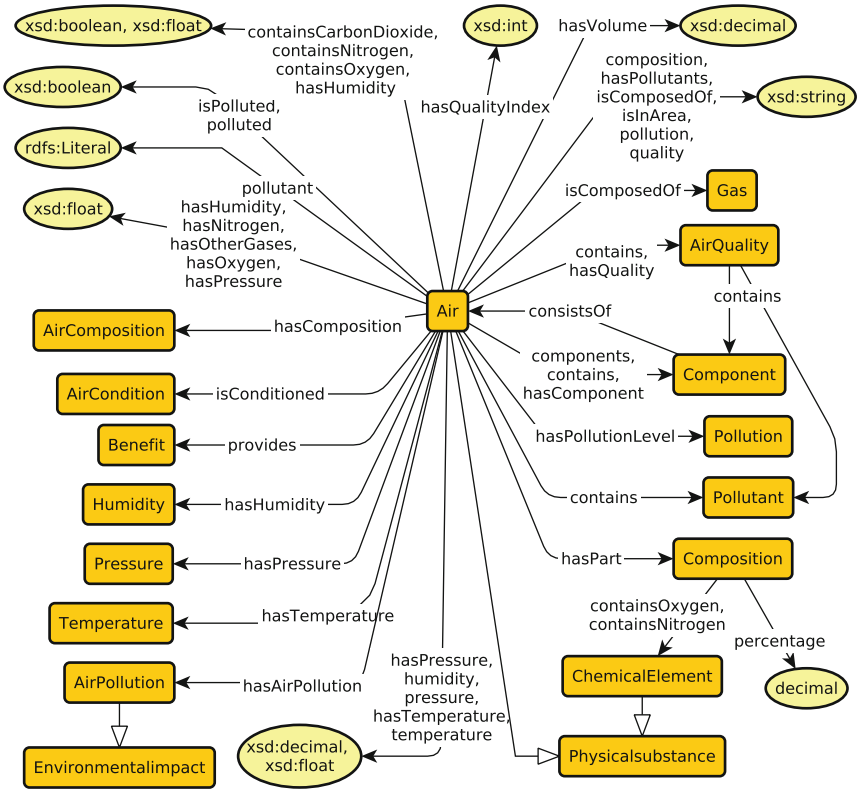

Fig. 1. A graphical schema diagram of the Air ontology

3 A Commonsense Modular Ontology Design Library

The purpose of a MODL is to collect together (curate) a set of ontology design patterns. In the first version [15], these were generally well-documented and useful patterns. A forthcoming second version has many more, spanning a wider set of application scenarios.[3] This particular MODL, CS-MODL, is similar insofar

[3] https://github.com/kastle-lab/modular-ontology-design-library.

that it also satisfies broad use-cases. We aim to collect together patterns of similar quality, abstraction, and expressivity for the purpose of modeling commonly occurring entities (e.g., in informal language or the real world).

CS-MODL, like other MODLs, is programmatically queryable through OPLa annotations [9,10]. As such, CS-MODL contains sufficient metadata scaffolding such that top-level conceptual components of patterns can be easily matched to external entities detected by a framework. For example, if a vision interpreter finds something with four legs, it could match to the `Chair` pattern. Inversely, if a chair is detected, *most* chairs have four legs, and this can be inferred from the matched pattern. This matching process can also be used for inferring *behaviors* or *functionalities* of said items, which is even more applicable to this use case.

License and Availability. Our artifacts including documentation, prompts, responses, extracted and cleaned TTL files, integrated patterns, (CS-MODL) and scripts that generated the data are provided via GitHub under an open and permissive license (GNU LGPL v2.1)[4]

4 Conclusion

Large language models have quickly become a source of commonsense information, in some cases supplanting search engine use. This easily available source of knowledge can enhance transfer and analogy learning, especially in combination with knowledge engineering methodologies (such as modular ontology modeling).

We have described our methodology for obtaining 104 commonsense micropatterns from an LLM. This process is generalizable from our chosen list, and the code to produce additional patterns is available online and permissibly licensed. The patterns are organized into a modular ontology design library (MODL), CS-MODL, also available in the same repository. CS-MODL can facilitate the construction of internal knowledge frameworks of automated systems through specializing or generalizing from the commonsense micropatterns.

Future Work. We have identified some areas for future enhancement.

1. Organize our patterns more intelligently (e.g., a subsumption hierarchy or otherwise unifying concepts). Creating this hierarchy and other relations between concepts will enable more sophisticated pattern identification.
2. We will evaluate the use of CS-MODL in a more principled way (i.e., through use in a case study) pertaining to ontology learning from text, and in a multi-agent teaming, visual sensor fusion exercise.
3. Describe the patterns more expressively (i.e., using OWL axioms).
4. Improved documentation: schema diagrams, summaries, and explanation of the formalization (i.e., in the style of [12,16]) will be explored.

Acknowledgement. The authors acknowledge partial funding by the National Science Foundation under grant 2333532 "Proto-OKN Theme 3: An Education Gateway for the Proto-OKN."

[4] https://github.com/kastle-lab/commonsense-micropatterns/.

A Appendix

1. Air
2. Area
3. Art
4. Back
5. Body
6. Book
7. Business
8. Car
9. Case
10. Chair
11. Change
12. Child
13. City
14. Community
15. Company
16. Couch
17. Country
18. Day
19. Door
20. Education
21. End
22. Eye
23. Face
24. Fact
25. Family
26. Father
27. Force
28. Friend
29. Game
30. Girl
31. Government
32. Group
33. Guy
34. Hand
35. Head
36. Health
37. History
38. Home
39. Hour
40. House
41. Idea
42. Information
43. Issue
44. Job
45. Kid
46. Kind
47. Law
48. Level
49. Life
50. Line
51. Lot
52. Loveseat
53. Man
54. Member
55. Minute
56. Moment
57. Money
58. Month
59. Morning
60. Mother
61. Name
62. Night
63. Number
64. Office
65. Others
66. Parent
67. Part
68. Party
69. People
70. Person
71. Place
72. Point
73. Power
74. President
75. Problem
76. Program
77. Question
78. Reason
79. Research
80. Result
81. Right
82. Room
83. School
84. Service
85. Side
86. Sofa
87. State
88. Story
89. Student
90. Study
91. System
92. Teacher
93. Team
94. Thing
95. Time
96. War
97. Water
98. Way
99. Week
100. Woman
101. Word
102. Work
103. World
104. Year

Fig. 2. The 104 nouns conceptualized into our commonsense micropatterns

References

1. English-Corpora: COCA. https://www.english-corpora.org/coca/
2. Cohen, R., Geva, M., Berant, J., Globerson, A.: Crawling the internal knowledge-base of language models (2023)
3. Cyganiak, R., Wood, D., Lanthaler, M. (eds.): RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014 (2014). http://www.w3.org/TR/rdf11-concepts/
4. Eberhart, A., Shimizu, C., Chowdhury, S., Sarker, M.K., Hitzler, P.: Expressibility of OWL axioms with patterns. In: Verborgh, R., et al. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 230–245. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4

5. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. IHIS, pp. 221–243. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92673-3
6. Gentner, D., Smith, L.: Analogical reasoning. In: Ramachandran, V. (ed.) Encyclopedia of Human Behavior, pp. 130–136, 2nd edn. Academic Press, San Diego (2012). DOI:https://doi.org/10.1016/B978-0-12-375000-6.00022-7. https://www.sciencedirect.com/science/article/pii/B9780123750006000227
7. Gentner, D., Smith, L.A.: Analogical Learning and Reasoning, pp. 668–681. Oxford Library of Psychology, Oxford University Press, New York (2013). https://doi.org/10.1093/oxfordhb/9780195376746.001.0001
8. Hammar, K., Presutti, V.: Template-based content ODP instantiation. In: Hammar, K., Hitzler, P., Krisnadhi, A., Lawrynowicz, A., Nuzzolese, A.G., Solanki, M. (eds.) 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]. Studies on the Semantic Web, vol. 32, pp. 1–13. IOS Press (2016). https://doi.org/10.3233/978-1-61499-826-6-1,
9. Hirt, Q., Shimizu, C., Hitzler, P.: Extensions to the ontology design pattern representation language. In: Janowicz, K., Krisnadhi, A.A., Poveda-Villalón, M., Hammar, K., Shimizu, C. (eds.) 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, 27 October 2019. CEUR Workshop Proceedings, vol. 2459, pp. 76–75. CEUR-WS.org (2019). http://ceur-ws.org/Vol-2459/short2.pdf
10. Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A.A., Presutti, V.: Towards a simple but useful ontology design pattern representation language. In: Blomqvist, E., Corcho, Ó., Horridge, M., Carral, D., Hoekstra, R. (eds.) 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, 21 October 2017. CEUR Workshop Proceedings, vol. 2043. CEUR-WS.org (2017). http://ceur-ws.org/Vol-2043/paper-09.pdf
11. Hitzler, P., Shimizu, C.: Modular ontologies as a bridge between human conceptualization and data. In: Chapman, P., Endres, D., Pernelle, N. (eds.) ICCS 2018. LNCS (LNAI), vol. 10872, pp. 3–6. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91379-7
12. Karima, N., Hammar, K., Hitzler, P.: How to document ontology design patterns. In: Hammar, K., Hitzler, P., Lawrynowicz, A., Krisnadhi, A., Nuzzolese, A., Solanki, M. (eds.) Advances in Ontology Design and Patterns, Studies on the Semantic Web, vol. 32, pp. 15–28. IOS Press, Amsterdam (2017)
13. Krisnadhi, A.A., Hitzler, P., Janowicz, K.: On the capabilities and limitations of OWL regarding typecasting and ontology design pattern views. In: Tamma, V., Dragoni, M., Gonçalves, R., Ławrynowicz, A. (eds.) OWLED 2015. LNCS, vol. 9557, pp. 105–116. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33245-1
14. Shimizu, C., Hammar, K., Hitzler, P.: Modular ontology modeling. Semant. Web **14**(3), 459–489 (2023). https://doi.org/10.3233/SW-222886

15. Shimizu, C., Hirt, Q., Hitzler, P.: MODL: a modular ontology design library. In: Janowicz, K., Krisnadhi, A.A., Poveda-Villalón, M., Hammar, K., Shimizu, C. (eds.) 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, 27 October 2019. CEUR Workshop Proceedings, vol. 2459, pp. 47–58. CEUR-WS.org (2019). http://ceur-ws.org/Vol-2459/paper4.pdf
16. Shimizu, C., et al.: The enslaved ontology: peoples of the historic slave trade. J. Web Semant. **63**, 100567 (2020). https://doi.org/10.1016/J.WEBSEM.2020.100567
17. Shimizu, C., et al.: Knowwheregraph-lite: a perspective of the knowwheregraph. In: Ortiz-Rodríguez, F., Villazón-Terrazas, B., Tiwari, S., Bobed, C. (eds.) KGSWC 2023. LNCS, vol. 14382, pp. 199–212. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47745-4