

My Ontologist: Evaluating BFO-Based AI for Definition Support

1. Introduction

Generative artificial intelligence (AI) became widely available with the release of GPT 3.5 by OpenAI in 2022 [33]. As the capabilities of the underlying large-language model (LLM) technology became more apparent, possible applications have captured the imagination of many. *Ontologies* – logically well-formed controlled vocabularies designed to represent entities and relationships among them – and *knowledge graphs* – ontologies merged with actual data about entities and associated relationships – have been identified as crucial for advancing research on and applications of LLMs [3, 6]. Recent research has explored the application of LLMs for ontology alignment [14-18], mining unstructured data for knowledge graph creation [19-27], the generation of ontological classes using LLMs [25, 28-32], and the creation of ontologies created using AI models [4]. However, to our knowledge there have been no serious attempts to generate ontologies or knowledge graphs that *extend from an upper-level ontology* or *substantially reuse ontology content from domain ontologies extending from such an upper-level*. The result is that nearly all research on the generation of ontologies using LLMs runs the risk of creating *ontology silos*, ontologies designed to represent overlapping domains that are not able to be integrated using standard computational methods without significant costs. Put another way, researchers have sought to determine whether LLMs can create ontologies and knowledge graphs without asking whether they should. What should be clear is that LLMs can be leveraged to create such artifacts, but this should be no surprise. It is in some sense rather easy to create ontologies and knowledge graphs: just write a script to convert column headers in a dataset to classes in the Web Ontology Language (OWL). What is more challenging is constructing these artifacts that are aligned with existing standards, such as the widely-used Basic Formal Ontology (BFO).

With this in mind, we are interested in determining the extent to which LLMs can support ontologists trained in the use of a specific top-level ontology. In what follows, we outline an empirical investigation into this question using BFO as our top-level. More specifically, in the first section we defend the importance of extending from a top-level ontology. The second section introduces relevant features of the Generative Pre-trained Transformers (GPTs) we investigate. The third section reports the results of an experiment exploring the capabilities of ChatGPT4 in generating ontology classes. The next three sections iteratively explore the creation of “My Ontologist”, a GPT designed to act as a BFO-trained ontologist, as well as the methods used to create these GPTs, measures of their performance, and the results of the application of a refined version of the model to generate textual definitions based on BFO and its extensions. While our results are promising, we close by observing that the recent release of GPT4o reveals substantial setbacks in the quality of definitional content produced by the model.

2. Top-Level Ontology

Ontologies often serve as structured frameworks for organizing information, allowing researchers across various domains to systematically categorize and integrate knowledge based on a hierarchy of entities and relationships among them. Ontologies facilitate not only the uniform representation of subject areas but also enable interoperability among disparate systems. This is achieved through the semantics that ontologies provide, which also allow for a hierarchy to be easily navigable and enable automated reasoning across large datasets, which can be of enormous use to data analysts. But these goods can only be provided if ontologies are well-constructed. If, per the previous section, one creates an ontology simply by running a simple command on a spreadsheet file, then there is no guarantee that the resultant ontology will even be logically consistent, and it is likely that the ontology will not have sufficient logical structure to support automated reasoning. Furthermore, such an ontology would not come with even minimally useful semantics. This is partly why automating ontology construction has not been particularly effective prior to the recent ascent of LLMs. We have historically needed human cognitive powers to create sophisticated ontologies. And even when employing human ontologists, the benefits of ontologies can only be provided when one of three things is true of the ontologies that organizations use in their disparate knowledge graphs: (1) the knowledge graphs use the same ontology, (2) the knowledge graphs use ontologies that are not the same but for which we can provide mappings between terms, or (3) the knowledge graphs use ontologies that are extended from some top-level ontology which defines more general terms and relations. (1) is the simplest case, but is not relevant to our purposes here because LLMs should not be used to carry out cut-and-paste commands. (2), on the other hand, is beyond the scope of this paper. We deal primarily with the question of whether and how LLMs can help carry out the operations necessary for (3). The way that most human ontologists best ensure the usefulness of their ontologies to organizations is by extending from top-level ontologies. In extending from a top-level ontology, ontologists adopt a set of metaphysical assumptions common to the rest of the community similarly employing that top-level ontology. Thus, their ontologies share some very basic semantics and fundamental logical structure which helps to ensure that they maintain interoperability, reasoning capability, etc. It is when ontologists fail to employ top-level ontologies that they reproduce precisely the sort of data silos that they ought to prevent.

BFO is an upper-level ontology recognized by the International Organization for Standardization as the first top-level ontology standard. BFO is a highly general framework designed to be extended into various domains, such as medicine, industrial manufacturing, defense and intelligence, and so on. BFO is used in over 600 ontology projects worldwide, including in the Open Biological and Biomedical Ontology (OBO) Foundry, the Industrial Ontology Foundry (IOF), and the Department of Defense (DOD) and Intelligence Community (IC) Foundry effort. BFO and one of its more widely-used extensions – the Common Core Ontologies suite – has, moreover, been adopted by the DOD and IC as the “baseline standards” for ontology development [2]. This broad acceptance and application highlights BFO’s versatility and its importance in enhancing semantic interoperability and data analysis capabilities across different sectors. BFO’s status makes it attractive for further use, deepening already explored domains and branching out to cover new domains. There is, however, a shortage of BFO trained ontologists, resulting in a growing need for BFO-based ontology engineering support. It

is our hope the results of the following work will provide such support by leveraging LLMs.

3. How Do GPTs Work?

Generative Pre-trained Transformers (GPTs), such as OpenAI's ChatGPT series and EleutherAI's Pythia, are a type of LLM based on the Transformer architecture [9]. This architecture fundamentally changed the landscape of natural language processing (NLP) by building upon the previously dominant recurrent neural network (RNN) models. The core mechanism of GPTs involves self-attention layers that process input data in parallel, allowing for more scalable and efficient learning of dependencies within data. GPTs are "pre-trained" on a large corpus of text, which helps the model learn the general structure and nuances of a language, from syntax to contextual relationships between words.

The training of a GPT model occurs in two major phases: pre-training and fine-tuning. During pre-training, the model is exposed to a vast dataset and learns to predict the next token in a string, given the tokens that precede it. This phase is unsupervised, meaning it does not require labeled data, which allows for the use of a diverse range of texts during training. The objective is to develop a robust model that understands language patterns and can generate coherent text sequences. Following this, the model undergoes fine-tuning, where it is adjusted to perform specific tasks such as translation, question-answering, or text summarization. This stage involves training on a smaller, task-specific dataset, enabling the model to optimize its responses for particular applications.

The capabilities of GPT models are broad, ranging from generating readable text to solving complex analytical problems. By leveraging the vast amount of learned data, GPTs can produce outputs that mimic human-like text, making them useful in applications such as chatbots, content generation, and more complex tasks like coding assistance and data analysis. The adaptability and scalability of GPTs make them a powerful tool in both academic research and real-world applications, driving forward the boundaries of what machines can achieve in the realm of human language. Given these capabilities, it seems reasonable GPTs can be leveraged to aid in the generation of ontologies. GPT4 has proven adept at creating code in Python [5], Java [10, 13], HTML [7, 12], JavaScript [11], SQL [1], and combinations [8]. GPT4s ability to create syntactically adequate code suggests it should be capable of creating syntactically adequate Resource Description Framework (RDF) serializations, such as RDF/XML and Turtle, and in fact, it can for simple prompts:

Prompt: "Write a triple for any class in XML and then the same class in Turtle" XML representation of a person named "Alice"

Output:
<Person>
<name>Alice</name>
</Person>

```
@prefix ex: <http://example.org/> .
ex:Person1 a ex:Person ;
ex:name "Alice" .
```

Figure 1. A simple prompt for RDF.

This suggests a straightforward use of GPT4 is to generate code that can be incorporated into an already existing owl file.

4. Methods

To generate triples in alignment with a standard, we first provide the necessary annotation property IRI's from a preexisting ontology. For example, we provide the annotation properties:

Annotation	IRI
<i>Label</i>	http://www.w3.org/2000/01/rdf-schema#label
<i>Subclass</i>	http://www.w3.org/2000/01/rdf-schema#subClassOf
<i>Definition</i>	http://www.w3.org/2004/02/skos/core#definition
<i>Contributor</i>	http://purl.org/dc/elements/1.1/contributor
<i>Example</i>	http://www.w3.org/2004/02/skos/core#example

Table 1. Provided annotation properties.

Placing these IRI's into a Chat GPT4 prompt that asks for a definition of 'human' generates the following turtle triples:

```
Prompt: "Make a BFO conformant definition of 'Human'"

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/ontology#> .
@prefix bfo: <http://purl.obolibrary.org/obo/BFO_> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:Human rdfs:label "Human" ;
      rdfs:subClassOf bfo:0000040 ; # material entity;
      skos:definition "A material entity that is a member
of the species Homo sapiens, characterized by complex organ
systems and capable of cognitive functions and social
behaviors." ;
      dc:contributor "Jane Doe" ;
      skos:example "John Doe participating in a complex
social interaction." .
```

GPT4 correctly identified the parent class and provided the correct IRI form ‘material entity’. For the most part, the differentiating features are acceptable. However, when asked to provide a model of a more complex entity, like ‘student,’ GPT4 errs:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/ontology#> .
@prefix bfo: <http://purl.obolibrary.org/obo/> .

:Student rdfs:label "Student" ;
        rdfs:subClassOf bfo:BFO_0000040 ; # Material
Entity
        skos:definition "A material entity that is a human,
formally engaged in learning, typically enrolled in an
educational institution such as a school or university." ;
        dc:contributor "John Doe" ;
        skos:example "Jane Doe studying at a university
library." .
```

First, GPT4 did not ask any clarification questions to disambiguate the meaning of ‘student.’ Instead, it assumed that we meant to model the human that is a student. This is problematic because much of the work of human ontologists lies in rooting out and attempting to clarify ambiguities in human language use. A failure to do so can result in bad models of a portion of reality, or fine models of the wrong portion of reality. Here, we could have intended for it to provide a definition for the class of individuals who are students, or to provide a definition for the student role, the role that one bears when they are a student and which cannot be reasonably understood as a material entity, much less a kind of human. In this case, we have a definition for the former class which—though it probably has the same extension as an appropriate definition supplied by a human ontologist competent in BFO—contains two further mistakes, one practical and one technical. The practical misstep of this definition is that it breaks BFO definition convention. The BFO community adheres to a convention of using the Aristotelian definition schema. Their class definitions take the form “An A is a B that Cs”, which not only eases the cognitive load of creating, understanding, and maintaining definitions on human ontologists by imposing a common structure, but which also assists the understanding of both human ontologists and end users by pointing out where a class (represented by A in the schema) belongs in the hierarchy by noting its parent class (represented by B in the schema) and the characteristic(s) (represented by C in the schema) distinguishing that class from its parent and sibling classes. Rather than taking the form “A is a B that C’s”, as was the case for the ‘human’ example, GPT 4 provided the definition for ‘student’ in the form “A is a B that is a C that D’s”. As far as mistakes go, this is not the worst, and at the same time not particularly surprising, because GPT 4 has not been trained on all and only text created by BFO-savvy ontologists in their professional capacity. Yet, this shows that in order to be useful to ontologists by writing definitions, GPT 4’s output would have to be further constrained by training or explicit commands. Otherwise, it will define classes in ways that might confuse human readers, who could misunderstand its definition here to mean that ‘student’ is a direct subclass of ‘material entity’, or to mean that ‘student’ should be asserted as a subclass of both

‘material entity’ and ‘human’. GPT 4, when given the appropriate annotation properties, is able to generate triples, but is limited when asked to perform more complex tasks that require pulling class information from more than one area of Basic Formal Ontology. The best use of this method is to provide the GPT with pre- curated class definitions in bulk and the appropriate annotation properties, then ask it to generate triples to be placed in an OWL file using VSCode or GitHub. Note, however, that this does not replace either the ontologist or the domain expert, since the ontologist must still work with the domain expert to create and curate class definitions. Instead, this use replaces ontology editors like Protégé and the writing of code by hand.

The more severe technical mistake is that it failed to acknowledge that a student is a human with a student role, which is the correct way to model this sense of ‘student’ in BFO. GPT 4’s definition does seem to imply that a student has a student role, but it does not make reference to the BFO class ‘role’, which would be the parent class of ‘student role’, and would be instrumental in distinguishing members of this class from humans who are not students. Therefore, this definition cannot obviously correspond to or be used to create appropriate class axioms, and constitutes bad ontological modeling of the portion of reality that we consider it (generously) to be intended to model. One reason that this is an important technical misstep in BFO is that BFO defines classes in terms of necessary and sufficient intrinsic properties for class membership. In BFO, instances of subclasses of ‘material entity’ (such as humans) are instances of those classes because of the way that they are, intrinsically, such that if they are no longer an instance of that class, there must be some corresponding change in their physical makeup [34]. This is clearly not how studenthood works. That is, one’s status as a student can change without one changing physically (imagine that one is expelled from school and has received the news via email, but has not checked one’s email yet). If BFO allowed extrinsic properties (such as one’s status as a student, senior ontologist, or pizza chef) to define classes, then the BFO model would lose much of its metaphysical plausibility. Therefore, it uses roles. Roles inhere in their bearers because of some special set of circumstances which do not require changes in the physical makeup of those bearers, whether these circumstances are social, physical, institutional, or otherwise contextual. In addition to these theoretical concerns, defining ‘student’ in the way that GPT 4 has would limit a knowledge graph’s utility in logical representation and reasoning. A knowledge graph structured with classes like this would be very difficult to usefully query or reason across without the application of a lot more human effort to create appropriate axioms, custom queries, or both.

GPT 4, when given the appropriate annotation properties, is able to generate triples, but is limited when asked to perform more complex tasks that require pulling class information from more than one area of Basic Formal Ontology. The best use of this method is to provide the GPT with pre- curated class definitions in bulk and the appropriate annotation properties, then ask it to generate triples to be placed in an OWL file using VSCode or GitHub. Note, however, that this does not replace either the ontologist or the domain expert, since the ontologist must still work with the domain expert to create and curate class definitions. Instead, this use replaces ontology editors like Protégé and the writing of code by hand.

Although not a success, this use of GPT 4 is not a complete failure either. There are encouraging signs that a GPT model could create ontologies. It correctly uses TURTLE syntax, can identify appropriate subclass relations, and can explain why it made the choices that it did. Perhaps, much like an early development ontologist, it requires a little bit of training.

4.1. My Ontologist 1.0

Methods:

The first My Ontologist was created to be an expert generating Basic Formal Ontology conformant ontologist. The goal was to allow for a domain expert to be able to work with My Ontologist to create BFO conformant ontologies. Chat GPT allows for a user to create their own GPT. A user can connect up to 20 PDFs at the backend for the GPT's knowledge base. My Ontologist 1.0 was given access to Building Ontologies with Basic Formal Ontology, the BFO 2.0 user's guide, the BFO 2020 Turtle code saved in a PDF file, and Learning SPARQL. The goal was to allow the GPT to follow the principles stated in Building Ontologies, to follow the guidance provided by the user's guide, to extend from Basic Formal Ontology. The purpose of Learning SPARQL was to allow the GPT to generate OWL files in the appropriate format. Prompts were then fed to 1.0 and positive or negative feedback was provided based on the answer provided.

Results:

My Ontologist 1.0 was wholly unsatisfactory and as such, no longer exists. Rather than complete the tasks it was given, 1.0 explained how to complete the task. When asked to model something like 'student,' 1.0 would provide long answers that were about perspectivalism, ontological realism, and granularity. When asked again, 1.0 would explain the difference between the classes 'continuant' and 'occurrent,' the highest-level classes in BFO. If a user were patient, they could eventually coax 1.0 into providing definition, but these were often no better than the GPT 4.0 version; it broke definition convention and did not ask disambiguating questions.

This was not the intended outcome. Our speculation is that 1.0 too heavily weighted 'Building Ontologies' and the 'BFO 2.0 Reference' in its knowledge base since the information that 1.0 provided primarily resides in those documents.

4.2. My Ontologist 2.0

Methods:

The desire for 2.0 was to pick up BFO conformant classes further downstream. Classes like 'human,' 'act,' and 'information content entity' are outside of the scope of BFO, so any class that 1.0 did successfully create duplicated efforts that already exist in an open and available manor. To remedy this, 2.0 was provided with BFO 2020, the BFO user's guide, the entirety of the Common Core Ontology Suite and the publicly accessible domain ontologies from CUBRC, the CCO NIST User's Guide, and three non-CCO BFO conformant domain level ontologies. To remedy the definition and practice problems, and to allow for 2.0 to create more sophisticated design patterns and definitions that both axiomatizes the hierarchy downwards and side-to-side, several academic papers that explicate aspects of BFO were also given the 2.0. Altogether, 20 PDFs documents, the current maximum allowable under the build-your-own GPT model, were attached.

Results:

Unfortunately, the CCO suite document was over one thousand pages long and when placed with the other documents, 2.0's knowledge base became unwieldy. Furthermore, the academic papers failed to have the desired effect, with many of the problems present in the first iteration present in the second iteration. The cumbersome nature of the CCO document also made it difficult for the GPT to pinpoint the appropriate class, often providing definitions for classes that extended from inappropriate classes. The most

significant problem was that the GPT would often crash before completing a response. Because of this, 2.0 was also deleted.

4.3. My Ontologist 3.0

My Ontologist 3.0 Design Methods:

The lessons learned through the design failures of 1.0 and 2.0 were instrumental in creating a stable and informative 3.0. Rather than provide academic papers, My Ontologist 3.0 was given thirty-six rules that provide explicit guidance for the creation of ontology classes, focusing on clarity, objectivity, coherence, extensibility, user engagement, and the precise articulation of concepts. Furthermore, the rules contain directives to prefer defined classes over primitive classes, to strictly adhere to the Aristotelian definition structure, to not create new properties unless necessary, and create ontological labels for classes that append 'role' if the entity is a subclass of 'role' and 'act' if it is an act. A complete list of rules can be found in the appendix.

Again, 3.0 was given BFO 2020 and the entirety of the CCO suite. However, this time the CCO suite was kept in its modular form. These include the Quality Ontology, the Agent Ontology, The Extended Relations Ontology, the Artifact Ontology, the Time Ontology, The Geospatial Ontology, The Currency Ontology, the Event Ontology, the Units of Measure Ontology, and the Information Entity Ontology. Additionally, 3.0 was provided with CUBRC's Modal Relation's Ontology and Facility Ontology.

The last major change is that 3.0 has access to axiomatic proofs of BFO 2020. This includes Alan Ruttenberg's Common Logic and Prover 9 proofs of all BFO classes. This was done to ensure that the ontological framework adheres to rigorous logical standards, providing a solid foundation for accurately and consistently representing the real-world entities and relationships. These proofs help to validate the structural integrity and cohesiveness of the ontology, minimizing errors and ensuring that the ontology's classifications and relations are logically sound and theoretically justified. This approach aims to enhance the reliability and applicability of the ontology across various domains and use cases.

Results:

My Ontologist 3.0 is a significant improvement over previous iterations. When prompted with an ambiguous term, it asks the user questions to disambiguate the term. It proficiently extends classes from BFO, noting that 'student' is a role that a human bears, not a material entity, appropriately names it a 'student role,' and provides the conditions for which the role is realized. Questions of whether the definition is accurate aside, the area that the definition could be improved is replacing "individual" with "human," allowing for easier axiomatization.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix bfo: <http://purl.obolibrary.org/obo/> .
@prefix : <http://example.org/ontology#> .

:StudentRole rdf:type owl:Class ;
              rdfs:subClassOf bfo:0000023 ; # Role in BFO
              rdfs:label "Student Role" ;
```



```
    skos:definition "A role that is realized by an
individual who is enrolled in and actively participating as
a learner in educational processes."@en ;
    rdfs:comment "This role applies to individuals who
are registered students in educational courses or
programs."@en .
```

Furthermore, 3.0 was able to appropriately extend CCO classes and restricted its use of object properties to only those found in CCO. It could take several iterations to walk 3.0 to the appropriate level or granularity, and required a person versed in CCO to point out when 3.0 invented a class or object property that was not in CCO. Here is an example of an exchange:

Define 'Vehicle Bed Mount'

My Ontologist 3.0: "Vehicle Bed Mount" is an equipment mount that is affixed to the bed of a vehicle and is designed to secure and support various equipment or cargo during transportation.

User: Good, but "affixed: not in CCO.

3.0: "Vehicle Bed Mount" is an artifact component that is part of a vehicle bed, designed to secure equipment or cargo by using a stable mounting interface to maintain position and safety during transport.

User: Bad parent

3.0: "Vehicle Bed Mount" is an equipment mount that is attached to the bed of a vehicle and is designed to secure and stabilize equipment or cargo during transport.

User: "is attached to" is not in CCO.

3.0: "Vehicle Bed Mount" is an equipment mount that is a component of the vehicle bed, designed to secure and stabilize equipment or cargo during transport.

User: "is component" is not in CCO.

3.0: "Vehicle Bed Mount" is an equipment mount that is part of the vehicle bed, designed to secure and stabilize equipment or cargo during transport.

To show the level of success 3.0 reached, compare 3.0's definition to an official CUBRC definition of the same entity that 3.0 did not have access to.

'Vehicle Bed Mount is an equipment mount that is designed to attach to the bed of a vehicle.'

The similarities between these definitions are promising, as well as how quickly 3.0 was able to complete the task.

4.4. Release of 4o:

The release of 4o by Open AI has made 3.0 functionally useless. It no longer restricts itself to the documents attached on the backend. Even when the 'search the internet' box is unselected, My Ontologist still searches the web for an answer. When asked if My Ontologist has access to the internet, it answers in the negative. When asked to provide a BFO conformant definition, it immediately searches the web. My Ontologist and GPT 4o behave in the same manner. Any progress that was made from the simple use of GPT 4 to My Ontologist 3.0 has been completely undone.

5. References

- [1] Dmitry Khanukov. ChatGPT. [cited 2024 Jul 6]. ChatGPT - SQL Expert. Available from: <https://chat.openai.com/g/g-m5lMeGifF-sql-expert>
- [2] Gambini B. University at Buffalo News Center. 2024 [cited 2024 Jul 6]. DOD, Intelligence Community adopt resource developed by UB ontologists. Available from: <https://www.buffalo.edu/news/releases/2024/02/department-of-defense-ontology.html>
- [3] Jin B, Liu G, Han C, Jiang M, Ji H, Han J. Large Language Models on Graphs: A Comprehensive Survey [Internet]. arXiv; 2024 [cited 2024 Jul 6]. Available from: <http://arxiv.org/abs/2312.02783>
- [4] Joachimiak MP, Miller MA, Caufield JH, Ly R, Harris NL, Tritt A, et al. The Artificial Intelligence Ontology: LLM-assisted construction of AI concept hierarchies [Internet]. arXiv; 2024 [cited 2024 Jul 6]. Available from: <http://arxiv.org/abs/2404.03044>
- [5] Nicholas Barker. ChatGPT. [cited 2024 Jul 6]. ChatGPT - Python. Available from: <https://chatgpt.com/g/g-cKXjWStaE-python>
- [6] Pan S, Luo L, Wang Y, Chen C, Wang J, Wu X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. IEEE Transactions on Knowledge and Data Engineering. 2024 Jul;36(7):3580–99.
- [7] Pietro Schirano. ChatGPT. [cited 2024 Jul 6]. ChatGPT - DesignerGPT. Available from: <https://chatgpt.com/g/g-2Eo3NxuS7-designergpt>
- [8] pulsr.co.uk. ChatGPT. [cited 2024 Jul 6]. ChatGPT - code: python java c html sql javascript react web+. Available from: <https://chat.openai.com/g/g-cksUvVWar-code-python-java-html-sql-javascript-react-web-c>
- [9] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention Is All You Need [Internet]. arXiv; 2023 [cited 2024 Jul 6]. Available from: <http://arxiv.org/abs/1706.03762>
- [10] Widenex. ChatGPT. [cited 2024 Jul 6]. ChatGPT - Java. Available from: <https://chatgpt.com/g/g-3GmVMbWRV-java>
- [11] Widenex. ChatGPT. [cited 2024 Jul 6]. ChatGPT - Javascript. Available from: <https://chat.openai.com/g/g-izfY9J70Q-javascript>
- [12] websitenerator.b12.io. ChatGPT. [cited 2024 Jul 10]. ChatGPT - Website Generator. Available from: <https://chatgpt.com/g/g-iYSeH3EAI-website-generator>
- [13] guliucang.com. ChatGPT. [cited 2024 Jul 10]. ChatGPT - Java Assistant. Available from: <https://chatgpt.com/g/g-kZ59SXL9S-java-assistant>
- [14] Amir M, Baruah M, Eslamialishah M, Ehsani S, Bahramali A, Naddaf-Sh S, et al. Truveta Mapper: A Zero-shot Ontology Alignment Framework [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2301.09767>
- [15] He Y, Chen J, Antonyrajah D, Horrocks I. BERTMap: A BERT-Based Ontology Alignment System. AAAI. 2022 Jun 28;36(5):5684–91.
- [16] He Y, Chen J, Dong H, Horrocks I. Exploring Large Language Models for Ontology Alignment [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2309.07172>
- [17] Hertling S, Paulheim H. OLLa: Ontology Matching with Large Language Models. In: Proceedings of the 12th Knowledge Capture Conference 2023 [Internet]. Pensacola FL USA: ACM; 2023 [cited 2024 Apr 23]. p. 131–9. Available from: <https://dl.acm.org/doi/10.1145/3587259.3627571>
- [18] Qiang Z, Wang W, Taylor K. Agent-OM: Leveraging LLM Agents for Ontology Matching [Internet]. arXiv; 2024 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2312.00326>
- [19] Biester F, Gaudio DD, Abdelaal M. Enhancing Knowledge Base Construction from Pre-trained Language Models using Prompt Ensembles.
- [20] Carta S, Giuliani A, Piano L, Podda AS, Pompianu L, Tiddia SG. Iterative Zero-Shot LLM Prompting for Knowledge Graph Construction [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2307.01128>
- [21] Caufield JH, Hegde H, Emonet V, Harris NL, Joachimiak MP, Matentzoglou N, et al. Structured prompt interrogation and recursive extraction of semantics (SPIRES): A method for populating knowledge bases using zero-shot learning [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2304.02711>
- [22] Meyer LP, Stadler C, Frey J, Radtke N, Junghanns K, Meissner R, et al. LLM-assisted Knowledge Graph Engineering: Experiments with ChatGPT. In 2024 [cited 2024 Apr 23]. p. 103–15. Available from: <http://arxiv.org/abs/2307.06917>
- [23] Ning X, Celebi R. Knowledge Base Construction from Pre-trained Language Models by Prompt learning.
- [24] Frey J, Meyer LP, Arndt N, Brei F, Bulert K. Benchmarking the Abilities of Large Language Models for RDF Knowledge Graph Creation and Comprehension: How Well Do LLMs Speak Turtle? [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2309.17122>

- [25] Kommineni VK, König-Ries B, Samuel S. From human experts to machines: An LLM supported approach to ontology and knowledge graph construction [Internet]. arXiv; 2024 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2403.08345>
- [26] Li T, Huang W, Papasrantopoulos N, Vougiouklis P, Pan JZ. Task-specific Pre-training and Prompt Decomposition for Knowledge Graph Population with Language Models [Internet]. arXiv; 2022 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2208.12539>
- [27] Yao Y, Wang X, Zhang Z, Qin Y, Zhang Z, Chu X, et al. Exploring the Potential of Large Language Models in Graph Generation [Internet]. arXiv; 2024 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2403.14358>
- [28] Giglou HB, D'Souza J, Auer S. LLMs4OL: Large Language Models for Ontology Learning [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2307.16648>
- [29] He Y, Chen J, Jiménez-Ruiz E, Dong H, Horrocks I. Language Model Analysis for Ontology Subsumption Inference [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2302.06761>
- [30] Liu H, Perl Y, Geller J. Concept placement using BERT trained by transforming and summarizing biomedical ontology structure. *Journal of Biomedical Informatics*. 2020 Dec;112:103607.
- [31] Mateiu P, Groza A. Ontology engineering with Large Language Models [Internet]. arXiv; 2023 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2307.16699>
- [32] Zhang B, Carriero VA, Schreiberhuber K, Tsaneva S, González LS, Kim J, et al. OntoChat: a Framework for Conversational Ontology Engineering using Language Models [Internet]. arXiv; 2024 [cited 2024 Apr 23]. Available from: <http://arxiv.org/abs/2403.05921>
- [33] Roose K. The Brilliance and Weirdness of ChatGPT. *The New York Times* [Internet]. 2022 Dec 5 [cited 2024 Jul 10]; Available from: <https://www.nytimes.com/2022/12/05/technology/chatgpt-ai-twitter.html>
- [34] Arp R, Smith B, Spear AD. *Building ontologies with Basic Formal Ontology*. Cambridge, Massachusetts: Massachusetts Institute of Technology; 2015. 220 p.