Automatic Generation of BFO-Compliant Aristotelian Definitions in OWL Ontologies with GPT

Tyler Thomas Procko, Omar Ochoa, Timothy Elvira
Department of Electrical Engineering and Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, United States of America
prockot@my.erau.edu, ochoao@erau.edu, elvirat@my.erau.edu

***Abstract*—Ontologies are representational artifacts that purport to accurately describe some aspect of reality, including the entities and the relations that hold between them. In computer science, ontologies are software artifacts containing the schematic structure for machine-readable knowledge, typically formed as a graph of subject-predicate-object triples, constrained through Description Logics. These resources and their relations are self-defining, i.e., some resource may be defined by considering all its stated relations. Resources are often attended with natural language annotations, that humans may read and interpret, such as labels and definitions. Many long-standing ontologies have useless lexical definitions that define resources cyclically, e.g., a FOAF:Person is simply defined as "A person". In Aristotelian terms, the definition of a thing should be reducible, by using terms simpler than itself, such that every definition can be unpacked up to the most general thing, which can only be defined by stating examples and use cases. This paper presents an innovative technique that leverages the Generative Pre-trained Transformer (GPT) large language model, GPT-4, for automatically generating Aristotelian definition annotations for OWL classes that engenders compliance with the Basic Formal Ontology standard.**

Keywords—ontology, epistemology, Linked Data, BFO, GPT

I. Introduction

There are various methods of ontology development arising from numerous publications, published by researchers with experience in diverse fields, e.g., philosophy, computer science, linguistics and so forth [1]. What the methods propose in theory, ontologists commit to in the form of taxonomies (or hierarchies) of classes, and the relations that hold between them. These representational components allow the explicit definition of the semantics of the aspect of reality under the purview of the ontologist; and with explicit semantics is granted readability by machines.

But humans do not generally view these explicit semantics when trying to understand a class; they read the annotations left by the ontologist; foremost among all annotations is the *definition*. The use of definition annotations, e.g., rdfs:comment or skos:definition, in ontologies is generally regarded by those in the Linked Data domain as a best practice; but this practice is not standardized and relies on the volitional provision of appropriate lexical definitions from the ontologist, which is, unfortunately, nearly never the case. Consider one of the first W3C ontologies, Friend of a Friend (FOAF), which defines many of its classes with cyclical rdfs:comment definition annotations [2]. Below are a few of these FOAF "definitions":

- Organization: An organization
- Person: A person
- Document: A document
- Image: An image

So, the problem of definitions is not relegated to new and novel ontologies, but to long-standing, foundational ontologies as well. Ontologists often provide confusing definitions, incorrect or illogical definitions, or no definitions whatsoever. Also, ontologists often forget to update definition annotations as they modify the ontology over time, resulting in outdated definitions that reduce the understandability of the ontology because of conflicting statements.

This paper presents an approach for a system that supports the automatic generation of Aristotelian definition annotations compliant with the Basic Formal Ontology (BFO) for Web Ontology Language (OWL) classes, using the class hierarchy and OWL restrictions from each class. The system is presented as potentially implementable in a variety of ways and is minimally intrusive to the ontology development life cycle. It is argued that, in fact, such a system can only help the ontologist, chiefly with maintenance and timeliness, by completely removing the need to write or update lexical definitions, where all effort can be focused on the true ontology work: the explicit relating of classes to one another through axioms. This would engender Continuous Integration and Continuous Delivery (CI/CD) for ontology development, bringing ontology development up to speed with traditional software lifecycles.

What follows is a proposal directed to the applied ontologist with interest or experience in BFO. The structure of the paper is as follows: Section II presents Motivation, Section III: Background, Section IV: Proposal, Section V: Discussion, Section VI: Future Work and Section VII, the Conclusion.

979-8-3503-5801-8/23/$31.00 ©2023 IEEE
DOI 10.1109/TransAI60598.2023.00042

II. Motivation

The automatic transcription of OWL class definitions would provide several benefits for ontologists and Linked Data work in general; these are given below.

1. *A standardized meta-metadata format*: OWL class definition annotations would follow a predictable, repeatable form that ontologists could instantly recognize and be able to read
2. *Useful definition annotations*: the widespread tendency for ontologists to provide what amount to useless definitions for their classes can be alleviated when class definition annotations are generated automatically from class restrictions
3. *No need to write definition annotations*: given class restrictions, class definition annotations would be generated automatically, and the ontologist would not need to labor in writing them
4. *No maintenance requirements for definition annotations*: it is all too common to consult the annotations of a class within an ontology, only to find that the lexical definition does not match the class, due to the class being updated and not being reflected in the definition, which is now out of date; the proposed technique would entirely alleviate this common maintenance problem
5. *Minimum ontological commitment*: definition annotations may clash with asserted constraints, so automatically generating them based on constraints would reduce ontological commitment

III. Background

There are a handful of annotation properties canonically used in defining OWL classes. These are given below:

- rdfs:comment
- skos:definition
- dcterms:description

For the purpose of this paper, let the three be considered semantically equivalent. The rdfs:comment annotation property will be used throughout, but any annotation property of choice can be substituted.

A. Species, Genus and Differentia

Aristotelian definitions allow the precise description of entities in relation to their immediate parent entity, such that definitions are "unpackable", or traceable up to the most general entity [3]. Aristotelian definitions are of the form:

S = a G that Ds

Where S is the *species* being defined, G is the *genus*, or immediate parent of S, and D is the *differentia*, or what differentiates S from G and its siblings (while still being a G). For instance, the Aristotelian definition of the class Person from the Common Core Ontologies (CCO), could be:

Person = an Animal that is a member of the species Homo sapiens

From this class, Person, one could unpack the definition of the genus, Animal, which would have its attendant Aristotelian definition, perhaps "*An Organism that has multiple cells*", where Organism could be unpacked to its genus, and so on.

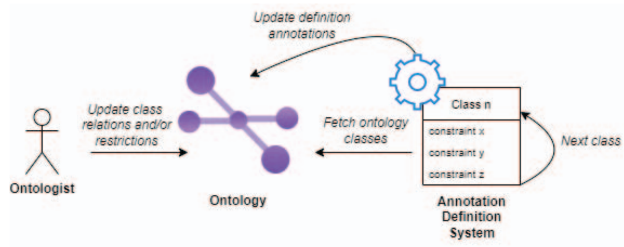

Fig. 1. Pseudo-architecture of a system to automatically generate BFO-compliant Aristotelian definitions for OWL classes.

IV. Proposal

In ontologies, OWL classes are defined with class axioms (OWL restrictions) which restrict the set of entities that qualify as being members of a given OWL class. For example (ignoring closure axioms for the sake of simplicity), an ontologist may assert that an OWL class, Carnivore, has the restriction: eats only Meat. This may be appropriate to preclude any Animal that does not exclusively consume meat from being part of the set of Carnivores: entities that are herbivores, e.g., cows, entities that are omnivores, e.g., humans, and entities that do not eat at all, e.g., rocks. This may be an incorrect assertion, however (consider carnivorous bears that eat primarily meat but will resort to consuming fruits and vegetation when hungry!) and granularity, so the ontologist must be careful.

Notwithstanding concerns about ontological correctness, the collection of OWL restrictions on a given class quite literally acts as its *differentia*, or what differentiates it from its *genus*, or parent [3]. This differentia, along with the genus, can be used to automatically generate lexical definition annotations for the class [4]. As a simple example, consider some OWL class, CivilianVehicle, in an ontology, that is a subclass of the class, Vehicle. Consider also, that CivilianVehicle has the constraints (Protégé Manchester syntax):

- (hasPart min 2 Wheel) and (hasPart max 4 wheel)
- hasPart exactly 1 Engine

The superclass (genus) of CivilianVehicle, Vehicle, and its two constraints, would feed into a lexical algorithm to output a string definition to be stored in a definition annotation, e.g., rdfs:comment, such as: "*A Vehicle that has 2 to 4 Wheels, and 1 Engine as part.*"

The proposed system can, therefore, be reduced into a set of simple, atomic steps:

1. Identify and select the OWL class of interest
2. Obtain its genus and its string name
3. Get the string name of the restrictions and their values on the class
4. Merge the genus and the restriction strings into a coherent, human-readable sentence
5. Push the definition into the ontology

This could be automated to iterate across the entirely of an OWL ontology's class hierarchy, as depicted in Fig. 1.

TABLE I. CONSIDERED IMPLEMENTATION TECHNIQUES FOR AUTOMATICALLY GENERATING BFO-COMPLIANT DEFINITIONS FOR OWL CLASSES.

Implementation Technique	Description	Feasible?
As a SPARQL query	Get string names for properties and classes used in restriction, build definition string	Yes, but not easily generalized
As a SWRL rule	SWRL could infer class definitions from restrictions	No, SWRL cannot operate on annotation properties
As an addon to an OWL reasoner	A reasoner could infer class definitions based on its constraints	Yes, but would require expert knowledge on building a reasoner
As a standalone program	Off-the-shelf RDF libraries (e.g., Python's RDFLib, Javas OWLAPI or Jena) or hand-written code	Yes, but would take considerable effort
As a plugin to a triplestore	E.g., in GraphDB; could generate class definitions from a background process; similar to the other implementations	Yes, but would take considerable effort and integration with the triplestore API
With GPT	Prompt GPT with the triples representing the class and request an Aristotelian definition	Yes, and the simplest solution with OpenAI's API

A. Possible Implementation Techniques

Given the relative simplicity of the task proposed, several approaches can be taken. The considered implementation techniques are given in TABLE I. Five of the six techniques considered are specific to Linked Data components, e.g., SPARQL or SWRL. Four of these five techniques are feasible, but they were not chosen because the system is formative in nature and requires prototypical treatment before deep time investment.

With the recent explosion in research and applications of the Generative Pre-trained from Transformers (GPT) series of large language models (LLMs), particularly GPT-3, GPT-3.5 and GPT-4, it is entirely possible that the automatic generation of BFO-compliant, Aristotelian definitions for OWL classes can be had by the correct prompt approach in combination with basic programming through OpenAI's API in the Python language.

B. Aristotelian OWL Class Definition Generation with GPT

A well-cited prompt engineering technique for GPT, and other LLMs, is 1- or few-shot prompting, which occurs when the prompt also includes examples of desired output [5]. This prompt engineering technique is highly applicable, and leveraged here to reinforce GPT to produce accurate, high-quality Aristotelian definitions. Additionally, it is common practice to provide as much context as possible in prompts to GPT, as it is capable of considering all that is said; and with more context (assuming it is relevant and correct), GPT seems to provide improved outputs.

The prompt structure is reducible to seven sections (see TABLE II), which are appended together, in order, into a single prompt to GPT. It should be noted that, in the prompt, the class, OrganizationMember, defined in the Agent ontology of the CCO, is given to GPT as an example for 1-shot prompting, because CCO is BFO-compliant. Any number of CCO class definitions, or other BFO-compliant class definitions, could be substituted for examples for GPT, striving to fit within the token length limit of GPT's prompt and response.

TABLE II. GPT PROMPT STRUCTURE FOR AUTOMATICALLY GENERATING BFO-COMPLIANT DEFINITIONS FOR OWL CLASSES.

Section	Prompt
Problem Context	I need an Aristotelian definition for an OWL class. This Aristotelian definition will be stored as a string in an rdfs:comment annotation, to be linked to the OWL class.
Solution Definition	An Aristotelian definition is of the form S = a G that Ds, where S is the species being defined, G is the genus (parent) of the species, and D is the differentia (what differentiates S from its species, while still being a G). In OWL terms, the genus is the parent class, defined through the rdfs:subClassOf or owl:subClassOf predicate, and the differentiae are defined in owl:Restrictions.
Example Preface	As an example, consider the OWL class, OrganizationMember, whose class definition, in Turtle format, is:
Example Body	cco:OrganizationMember rdf:type owl:Class ; owl:equivalentClass [rdf:type owl:Restriction ; owl:onProperty obo:has_role; owl:someValuesFrom cco:OrganizationMemberRole] ; rdfs:subClassOf cco:Person , [rdf:type owl:Restriction ; owl:onProperty cco:is_affiliated_with ; owl:someValuesFrom cco:Organization] .
Example Output	And whose lexical definition is "A Person who is affiliated with some Organization by being a member of that Organization."
Request	Given this example, please provide me with an Aristotelian lexical definition for the following OWL class definition:
New OWL Class	[Insert entire OWL class definition block]

C. Evaluation

Preliminary evaluations were done in the ChatGPT web interface, using the June 2023 GPT-4 model [6]. A custom schema:Offer class was chosen because Schema.org is used to markup web documents, so GPT should have knowledge of it. Using the prompt built from TABLE II, an Aristotelian definition was generated for the class, schema:Offer, below:

```
schema:Offer rdf:type owl:Class ;
  rdfs:subClassOf schema:Intangible ,
    [  rdf:type owl:Restriction ;
       owl:onProperty schema:itemOffered ;
       owl:someValuesFrom <https://gs1.org/voc/WearableProduct>
    ] ,
    [  rdf:type owl:Restriction ;
       owl:onProperty schema:offeredBy ;
       owl:someValuesFrom schema:OnlineStore
    ] ,
    [  rdf:type owl:Restriction ;
       owl:onProperty schema:priceCurrency ;
       owl:someValuesFrom xsd:string
    ] ,
    [  rdf:type owl:Restriction ;
       owl:onProperty schema:price ;
       owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
       owl:onDataRange xsd:string
    ] ;
```

Algorithm 1 Automatic, GPT-enabled, BFO-compliant, Aristotelian OWL class definition generation

Input: Ontology file
Output: Annotated Ontology file
Assumption: The parsing of the ontology file is done with a library like RDFLib, and not with plaintext string manipulation, i.e., during execution, OWL classes are referenceable objects in memory

```
Load the ontology file (e.g., with a filesystem URL with RDFLib)
while class hierarchy traversal not finished do
  Read class definition body (its set of class axioms)
  Save class axioms into string
  Append class axioms string to GPT prompt string
  Send API request containing the constructed prompt to GPT
  Get Aristotelian class definition in response
  if business logic wants overwritten definitions then
    Replace extant rdfs:comment (or equivalent annotation
    property) with the new definition as an rdfs:comment
  else
    Append new rdfs:comment (or equivalent annotation
    property) to the class body
  end if
end while
Save the annotated ontology file
```

GPT-4 quickly provided the following response as an Aristotelian definition for the class schema:Offer:

- An Intangible that is offered by some OnlineStore, involves the offering of a WearableProduct, is priced in a certain currency represented as a string, and has exactly one price associated with it.

This is a very accurate, well-written and readable natural language definition for the class schema:Offer. It fulfills the requirement of maintaining its genus and differentia and remains intelligible [4]. This is a promising indication of the potential for GPT in automating the process of OWL class definition creation.

To this end, with the RDFLib library in the Python language, programmatic traversal through an ontology can be generalized, allowing future expansion to any ontology, and opening the opportunity for continuous ontology development. Without needing to manually resolve lexical definitions against OWL class axioms, ontologists can focus on actual ontology development. Accessing OpenAI's API can also be done in Python, so the proposed system is able to be developed with a modicum of simple code, much of which amounts to string manipulation for the prompt to GPT. Algorithm 1 presents the pseudocode for implementing the proposed system.

V. Discussion

It is clear that the problem of incorrect, inconsistent, or otherwise unavailable lexical definitions for OWL classes is widespread and generalizable. The proposal addressing this problem is minimally intrusive, utilizing simple prompts to GPT through the OpenAI API to automatically generate consistent, BFO-compliant, Aristotelian definitions for OWL classes. In this section, the limitations and implications of the proposal are discussed.

A. Limitations

This proposal is best implemented when the ontology considered for candidacy adheres to the principle of single inheritance, i.e., when the ontology is based on a mono-hierarchy, where each class is a child of one, and *only* one, parent class. This design choice clashes with the vast majority of developed ontologies, which are generally free-form, allowing classes to inherit from multiple parent classes.

BFO requires a mono-hierarchy because it adheres to the Aristotelian notion that each thing is a sub-class of exactly one parent class, so that definitions can only be unpacked to a single, next determinable thing [3]. BFO defines the Realizable Entity, with sub-class Role, that BFO-compliant ontologists utilize commonly to avoid the multiple inheritance design choice. For example, instead of asserting that a Computer Scientist is both a Person and a Worker, in BFO terms, a Computer Scientist would be a Person who may play the Role of a Worker in some professional context. In any case, requiring a mono-hierarchy ensures that there is only one term for the population of the genus of a generated definition annotation. Otherwise, there may be multiple genera, which will make certain definition generation more complicated.

A final consideration is the depth of treatment of the two types of Description Logic constraints for OWL classes: SubClass Of axioms (necessary conditions), and Equivalent To axioms (necessary & sufficient conditions). For a given class, the former present qualities that are expected in instances of the class, but do not mean inclusion in the set of members of the class; the latter, if fulfilled by an instance, guarantee inclusion in the set of things that are members of the class. For example, some Class, SmartHuman, may have the assertion:

- equivalentTo Human and hasIQ min 130

Where the class Human may have several subClassOf or equivalentTo constraints itself, e.g., on body parts, brain function, etc. The distinction between these two axioms in OWL is up to the ontologist. This is mentioned because it is probably reasonable to assume that the lexical definitions reflecting asserted OWL axioms should also distinguish between necessary and necessary & sufficient conditions, as the former are essentially descriptive, while the latter can be interpreted as more prescriptive. So, due diligence is required from Description Logics experts to effectively bring the proposed method of automatic Aristotelian definition generation for OWL classes to a position of validity. Souza et. al have presented work on the formation of Aristotelian definitions from OWL class axioms, with deeper consideration of Description Logics [4].

B. Reverse-Engineering OWL Classes from Definitions

What has been discussed thus far presupposes the existence of OWL classes in an ontology. These defined classes, with their restrictions, are proposed to be capable of allowing the automatic generation of definition annotations. This was shown to be feasible by using GPT-4.

What should also be considered is the possibility of OWL class generation from natural language prompts, or in other words, the inverse of what is proposed here. This could be an effective means of proto-ontology formation, allowing the

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix gs1: <https://gs1.org/voc/> .

schema:Offer rdf:type owl:Class ;
        rdfs:subClassOf schema:Intangible ,
                [ rdf:type owl:Restriction ;
                  owl:onProperty schema:itemOffered ;
                  owl:someValuesFrom gs1:WearableProduct
                ] ,
                [ rdf:type owl:Restriction ;
                  owl:onProperty schema:offeredBy ;
                  owl:someValuesFrom schema:OnlineStore
                ] ,
                [ rdf:type owl:Restriction ;
                  owl:onProperty schema:priceCurrency ;
                  owl:someValuesFrom xsd:string
                ] ,
                [ rdf:type owl:Restriction ;
                  owl:onProperty schema:price ;
                  owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger
                ]
```

Fig. 2. The OWL output from prompting GPT-4 with an Aristotelian definition of an OWL class and requesting an OWL serialization.

ontologist to begin defining an ontology by writing in natural language prompts to GPT, with the requested output being OWL class descriptions. So, in taking the Aristotelian class definition generated by GPT-4 in subsection C of the Proposal section of this paper, GPT-4 was prompted with the following:

- Provide a Turtle format OWL class definition given this natural language definition of a schema:Offer class: "An Intangible that is offered by some OnlineStore, involves the offering of a WearableProduct, is priced in a certain currency, and has exactly one price associated with it."

In response, GPT-4 produced the output shown in Fig. 2. This is a semantically equivalent recreation of the nominal class definition defined earlier. Moreover, GPT-4 was capable of deducing the namespace for the GS1 vocabulary, which is impressive, as it is only capable of "deducing" such a thing from the terms used in the prompt, e.g., WearableProduct. This was in a separate GPT session, so the model had no association from the previous prompt, which is a testament to the instructive learning capabilities of GPT-4. Similar research, e.g., OntoGPT, a project of the OBO Foundry, allows the generation of ontologies by prompting GPT [7]. Others have presented the use of GPT-3 and GPT-3.5 for generating synthetic ontologies [8, 9].

VI. Future Work

The present paper was formed as a very preliminary analysis of using GPT-4 to automatically generate Aristotelian definitions in OWL ontologies. Inasmuch as the authors were constrained for time with respect to the publication outlet, it was not possible to fully assess the proposed approach. Nonetheless, it is possible, indeed, to provide direction for future work, particularly in the area of assessing the feasibility of using GPT-4 to generate Aristotelian definitions for OWL classes *at scale*, as the work of the present paper is very nascent and limited to one small example. The key considerations of this feasibility assessment, which future research should begin with, are given below.

- Larger datasets and consistent generalizability: what has been demonstrated in the present paper is a provisional proof-of-concept; granting this, it is acknowledged that GPT-4 needs to be evaluated against a much larger dataset, and not simply one OWL class; i.e., future work should seek to generate definitions for entire OWL ontologies of varying domains and granularities, to understand more precisely what GPT-4 can, and cannot, provide accurate Aristotelian definitions for; the pseudocode of Algorithm 1 is the basis for doing this, ideally in Python, a language suited for rapid prototyping, and because the OpenAI API documentation uses Python examples
- Quantitative measurement of definition quality: it would be beneficial to calculate measurements of definition quality: the two methods of most direct application would be the Bilingual Evaluation Understudy (BLEU) [10] and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [11] scores, both of which rely on comparison to reference texts which would have to be generated manually or taken from a well-formed reference ontology, e.g., BFO or the CCO; the ROUGE score is particularly applicable insofar as it is used to evaluate automatic summarization, which is, roughly, what an Aristotelian definition captures from a breadth of OWL class axioms; there is also voluminous literature for evaluating LLMs in specific application areas [12]
- Comparison with different LLMs: the only LLM used was OpenAI's GPT-4; so, in order to establish some manner of certainty with regard to the quality of generated definitions, it would be wise to test a number of other LLMs, e.g., Google's Fine-tuned LAnguage Net (FLAN) [13], Facebook's Large Language Model Meta AI (LLaMA) [14], and OpenAI's significantly cheaper, faster GPT-3.5, which is still quite capable

These are the primary considerations for future work in this area, i.e., using LLMs to automatically generate lexical definitions for OWL classes; in the case of the present paper, the definitions were of the Aristotelian form. Future researchers should investigate the points above, using this work as a springboard.

Also, it is prudent to note that many ontologists use the rdfs:comment annotation property, not as a container for a definition, but literally to leave comments on things to return to, or as a means of informal documentation. Granting this, it is advisable to consider the standardization of a recognizable, explicit definition annotation property that is not already in use for common purposes like rdfs:comment. For example, the Information Artifact Ontology (an OBO ontology built on BFO), has the annotation property IAO:0000115, or IAO:definition. This may suffice for the purpose presented in this paper, but it is incumbent on members of the Linked Data community to establish consensus on what annotation property to use for automatically generated definitions, to maintain consistency across works.

VII. Conclusion

The issue of insufficient, incorrect, inconsistent and, often, incomplete lexical definitions, e.g., in rdfs:comment annotations, for classes in OWL ontologies is apparent. Ontologists concern themselves principally with the development of their ontologies in defining Description Logic assertions, which results in lexical definitions often being in “definition debt”, differing drastically from the logical axioms of their class. The Aristotelian technique of lexically defining classes in relation to their genera and differentiae provides a concrete point upon which to posit the potential of automating the definition of OWL classes.

The motivation for establishing a means of effectively automating Aristotelian definitions for OWL classes, aside from assisting the human tendency to read lexical definitions before blocks of logical axioms, is fivefold:

1. Establishing a standardized meta-metadata format
2. Ensuring useful definition annotations
3. Removing the need to write definitions
4. Removing definition maintenance requirements
5. Facilitating minimum ontological commitment

This paper presents considerations on possible techniques for automating the formation of BFO-compliant, Aristotelian lexical definitions for OWL classes, including SPARQL queries, SWRL rules, add-ons to OWL reasoners, and other Linked Data strategies. The technique ultimately chosen was to 1-shot prompt GPT-4 with an example OWL class and its Aristotelian definition, followed by requesting a lexical “translation” of an OWL class into a readable, Aristotelian definition. In the evaluation of this proposed method, GPT-4 was shown to be entirely capable of fulfilling this task, and was even capable of performing the inverse, generating OWL class definitions given a lexical one in Aristotelian form.

Finally, the pseudocode for a system that interfaces with the OpenAI API to allow the Aristotelian definition annotation of entire OWL ontology files is proposed. This is a promising step towards automating the standardization of lexical definitions in OWL classes, but, as discussed, future evaluations with larger datasets and other LLMs are needed to ensure the consistency of the proposed approach. In software engineering terms, this automatic approach is beneficial for ontology development CI/CD, a consideration of undoubted interest to ontologists in any domain. It is the authors’ hope that the present paper be used by other researchers as a means of evaluating techniques, e.g., with GPT, for automating OWL class definition annotations.

References

[1] A. Sattar, E. S. M. Surin, M. N. Ahmad, M. Ahmad and A. K. Mahmood, "Comparative analysis of methodologies for domain ontology development: A systematic review," *International Journal of Advanced Computer Science and Applications,* vol. 11, no. 5, pp. 99-108, 2020.

[2] D. Brickley and L. Miller, "FOAF Vocabulary Specification - Namespace Document 1 May 2004," 18 August 2004. [Online]. Available: http://xmlns.com/foaf/0.1/. [Accessed June 2023].

[3] R. Arp, B. Smith and A. D. Spear, Building Ontologies with Basic Formal Ontology, MIT Press, 2015.

[4] A. D. Souza, F. Farinelli and E. R. Felipe, "Building Definitions using the Aristotelian method," *Fronteiras da Representação do Conhecimento,* vol. 1, no. 1, pp. 73-93, 2021.

[5] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems,* pp. 1-7, 2021.

[6] OpenAI, "GPT-4 Technical Report," *arXiv:2303.08774,* 2023.

[7] J. H. Caufield, H. Hegde, V. Emonet, N. L. Harris, M. P. Joachimiak, H. K. Matentzoglu, S. A. T. Moxon, J. T. Reese, M. A. Haendel, P. N. Robinson and C. J. Mungall, "Structured prompt interrogation and recursive extraction of semantics (SPIRES)," *arXiv preprint arXiv:2304.02711,* 2023.

[8] A. Bikeyev, "Synthetic Ontologies: A Hypothesis," *Available at SSRN 4373537,* 2023.

[9] P. Mateiu and A. Groza, "Ontology engineering with Large Language Models," *arXiv preprint arXiv:2307.16699,* 2023.

[10] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," *Proceedings of the 40th annual meeting of the Association for Computational Linguistics,* pp. 311-318, 2002.

[11] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," *Text summarization branches out,* pp. 74-81, 2004.

[12] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang and X. Xie, "A Survey on Evaluation of Large Language Models," *arXiv preprint arXiv:2307.03109,* 2023.

[13] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652,* 2021.

[14] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave and G. Lample, "LLaMA: Open and Efficient Foundation Language Models," *arXiv preprint arXiv:2302.13971,* 2023.