Ontology-Based Automatic Generation of Learning Materials for Python Programming

Jawad Alshboul*, Erika Baksa-Varga
Faculty of Mechanical Engineering and Informatics, University of Miskolc, Hungary

***Abstract*—Learning materials in programming education are essential for effective instruction. This study introduces an ontology-based approach for automatically generating learning materials for Python programming. The method harnesses ontologies to capture domain knowledge and semantic relationships, enabling the creation of personalized, adaptive content. The ontology serves as a knowledge base to identify key concepts and resources and map them to learning objectives aligned with user preferences. The study outlines the design of a dual-module ontology: a general and a specific domain-specific concepts module. This design supports enhanced, tailored learning experiences, enhancing Python education by meeting individual needs and learning styles. The approach also increases the quality and uniformity of generated content, which can be reused for educational reasons. The system ensures alignment with reference materials by using BERT embeddings for a semantic similarity measurement, achieving a quality accuracy of 98.5%. It can be applied to improve Python education by providing personalized recommendations, hints, and problem-solution generation. Future developments could further support the functionality to strengthen teaching and learning outcomes in programming education, and it could expand to automated problem generation.**

Keywords*—Ontology; knowledge graph; learning material generation; domain knowledge; python

I. Introduction

Recently, knowledge graphs (KGs), as structured forms of knowledge representation, have attracted substantial research interests both in academia and industry from modern ontology views. Integrating educational technologies with KGs has an impressive influence on teaching and learning activities, especially in programming with Python. E-learning platforms provide students with tools to easily engage and receive ongoing feedback during the e-learning sessions [1].

KGs are crucial in optimizing the automation of ontology-based learning material generation. They support the organization, interrelation, and knowledge utilization in a particular field [2]. In Python programming, KGs can provide a definite delineation of the existing knowledge, relations, and entities [2]. Additionally, ontology-driven systems support more effective comprehension of the context and relations of various concepts, thus enabling more precise and thorough learning materials generation [2]. Adding KGs to the ontology-based automatic generation of educational materials improves the relevance of contents, personalization, interoperability, content reuse, and efficient knowledge capture [3]. KGs can efficiently organize and manage structural knowledge related to the Python programming language [3].

In the information age, one's programming capability is required in many professions, as accentuated by the availability of resources aimed at teaching and training in programming [4]. Designing high-quality learning materials for programming languages is difficult and requires substantial resources because of fragmentation in educational programming design, instructional programming expertise, and difficulty in adaptive personalization [5]. Nevertheless, computer-based automatic generation of instructional materials, especially ones based on ontological frameworks, can simplify this task significantly. This is done through the ALMG, which stands for automatic learning materials generation (ALMG), a relatively recent expansion in the most advanced educational technology [6]. Quizzes, study guides, and practice exercises, among other educational content, are now automatically generated with the help of artificial intelligence and machine learning algorithms [6]. This technology will assist educators in saving time and costs by generating particular and appealing materials for students [6]. Calmon et al. [7] describe the concept of an automated system of curriculum selection tailored to the student's requirements and preferences. This is done by utilizing machine learning concepts and data analysis techniques to enhance the effectiveness of educational content and formative processes of the student. It also encourages the idea of implementing automated curriculum generation to help educational institutions deliver and personalize learning while increasing student performance significantly. In their study, Xia et al. [8] propose a method for delivering adaptive networking learning material that meets these needs and preferences. The system itself is also based on machine learning algorithms and data analytics and uses them to determine the effectiveness of the educational content and activities. The study demonstrates how the concept of automated curriculum generation can help in the management of learning processes, as well as increase students' results in networking education.

One of the methods to represent domain models is through ontology-based representation [9]. Ontology offers a standardized vocabulary for domain modeling, including describing concepts in the domain, their properties, and their relationships [10]. Semantic understanding and knowledge representation enable ontology-based automatic learning materials generation for Python programming that produces resources like tutorials, code examples, exercises, and assessments. The development of an ontology for capturing Python programming concepts, relationships, and properties is used in this approach. It attempts to create learning materials based on the pedagogical requirements and learning objectives. The ontology-based approach further enables continuously

updating and refining the learning materials so they are in sync with Python programming environment changes [11].

Ontology-based automatic learning materials generation for Python programming is a highly efficient and scalable approach using structured knowledge presentation for automating educational content creation [5]. With this method, its learning materials remain consistent, high quality, and personalized, all while allowing for the efficient creation of various resources. Likewise, the existence of the ontologies makes the routines adaptable to changes in Python programming [12], i.e., updating the ontologies and automatically regenerating learning materials. Ontologies' automation saves educators and content creators time and effort and improves a deep semantic understanding of the Python programming domain for a better generation of learning materials [13].

Learning materials for Python programming education presents difficulties in providing scalable, high-quality, and personalized materials [14]. Creating them manually is time-consuming and may require catching up with the Python ecosystem. To resolve these, an automated approach requiring ontologies is needed. This work aims to develop a comprehensive ontology for Python programming, and design an ontology-based automatic learning materials generation system for Python education. However, this methodology can greatly improve Python programming learners' exposure and efficiency to educational resources. The authors also explain how the presented ontology-based system was designed and implemented and offer possible further development and implications of such a system.

Automated generation of learning materials in the context of Python programming education is critical for scalability, adaptation, personalization, consistency, efficiency, accessibility, research, and innovation [15]. It can help meet the growing demand for diverse, high-quality resources, adapt to ecosystem changes, and deliver personalized learning experiences. The ontology-based approach guarantees consistency in different educational materials, keeping them high quality. It reduces the time taken to create content for educators to be concerned with the pedagogical part. This also makes accessibility easier for a variety of learners with varying backgrounds and learning styles. Moreover, it can serve as research in educational technology artificial intelligence as well as semantic understanding for programming education, driving innovation in programming education.

This study discusses the potential benefits and limitations of ontology-based automatic learning materials generation in the context of programming languages. This approach takes advantage of the use of technologies like natural language processing, machine learning, and automated code generation in the ontologies framework that can potentially transform how tailored learning materials for programming languages are generated.

Then, the study will focus on the underlying technologies and methodologies of ontology-based automatic learning materials generation and information on how ontologies can be utilized to represent domain knowledge and the automated generation of educational content is presented. Furthermore, the study builds up on the implications of ontology-based automatic learning materials generation for education and training to discuss to what extent such systems could improve the access to and efficiency of programming language instruction. This will also review the challenges and limitations of this approach and future directions in research and development in this emerging field. This exploration serves to help understand the possibilities of generating ontology-based automatic learning materials on programming languages and how it may shape how we teach Programming education and training.

The main objectives of this study are to design a new ontology-based framework that illustrates Python programming concepts and their interconnections and to develop a system capable of automatically generating learning materials—specifically quizzes—that reflect those Python programming concepts and their relationships. The study is organized as follows: an introduction is provided in Section I, and Section II presents the related work. Section III shows an ontology-based approach to producing learning material, while the Section IV shows the allied knowledge model for the domain-specific concepts. Section V implements the proposed model, followed by Section VI, which validates and evaluates the proposed ontology-based model. Results and discussion are presented in VII and VIII sections, followed by a conclusion in the Section IX, emphasizing the practical implications of the proposed model.

II. Related Work

Effective instruction in programming education requires learning materials. They include textual and visual content, interactive exercises, tutorials, real-world examples, assessment tools, and personalized adaptation. The textual content includes explanations, code examples, and problems to solve. Interactive exercises provide hands-on experience and reinforce learning. Tutorials provide step-by-step guidance, while real-world examples demonstrate practical application. Assessment tools gauge students' understanding and progress. The aim is to offer comprehensive, accessible, and engaging resources that enable different learning methods, involve much hands-on practice, and be connected with real-world applications. One major area of study in computer science and software development is programming languages. Methods for programming concept teaching need to be effective. Interest development question generation techniques for programming languages can provide a promising avenue, creating a large number of practice questions. These can help to reiterate the learner's understanding and assess his or her knowledge [16]. In [14], the author applied ontology to develop a question-generation approach for programming concepts.

Several studies have investigated the possibility of automatic generation of learning materials and their positive impact on enhancing student engagement and learning outcomes. Vergara et al. [17] demonstrated that AI-generated personalized learning materials increased students' motivation and performance in mathematics courses. Liu et al. [4] also pointed out how AI-powered content creation tools can assist educators in saving time and resources by automating the task of creating quizzes and associated worksheets, for example.

Generating automatic learning materials allows students with varied learning requirements to have personalized learning experiences served to them. Lin et al. [18] extend the literature by examining if there is a relationship between student engagement and learning outcomes in a cyber-flipped course. It examines the effects of engagement (measured as online activities) on academic performance. The study also finds a positive direction correlation between the student's engagement and final grades, highlighting the value of active participation and interaction of the students with the course materials in an environment set for blended learning.

Over the years, countless researchers have attempted to draw insights from generating learning materials and using ontologies in the educational domain to automatically create and present learning materials and knowledge frameworks. Although educational settings utilize ontologies to improve the personalization of learning experiences, they are not sufficiently advanced. Content is organized into ontologies, and the learner profiles and learning material interoperability are enabled [5]. Dynamic adaptation is provided by integrating them with learning management systems [5]. In [19], the authors propose an intelligent system based on ontology to automate tasks like course scheduling, student enrolment, and academic advising. This system is intended to provide the benefits of better efficiency and accuracy by capturing and representing said domain knowledge in a structured format. It automates tasks like personalized schedule of course schedules, matching students to advisors, and updating real-time course availability. This is beneficial in improving decision-making, reducing administrative burden, and enhancing the student experience. William and Joselin [5] discuss how ontologies can be leveraged to enhance the personalization of learning in educational environments. They are saying that traditional one-size-fits-all is not working for every learner and that personalized learning is improving the engagement and performance of the students. Ontology-based knowledge representation is discussed, and potential challenges and limitations are presented, which will help guide future research.

In [13], the authors introduces a method of constructing structured knowledge graphs based on word embeddings. To extract and represent educational concepts from textual resources, the authors employ natural language processing and machine learning methods. This method automatically captures semantic relationships between concepts, extracts unstructured data, and helps define references such as prerequisite, hierarchy, and relatedness. Finally, the study addresses the effectiveness of the method to build educational knowledge graphs and the potential benefits for use in educational content with structured and interconnected content. As Stephen [1] discussed, they use large language models such as GPT-3 to automatically generate computer science (CS) learning materials. The technique produces content related to various CS topics, such as programming languages, algorithms, and data structures. It can be tailored to cater to different learning levels as well as styles. The study also assesses the quality, relevance, and coherence of the generated materials. This could provide innovative approaches to improve computer science learning and educational resources. Flanagan et al. [20] propose using natural language processing and machine learning to extract and structure content from educational resources such as textbooks, lecture notes, or online articles. In order to define the content elements and link them to different levels of learning objectives, machine learning algorithms are used to categorize and link content elements. The study also evaluates the accuracy, completeness, and appropriateness of the generated content models for digital learning environments. In [21], the author discusses the construction of a knowledge graph for an Australian school science course. The study focuses on the construction of the graph, its fit in a related course agenda, and the application of semantic representation techniques. The graph is also studied with respect to practical applications, namely personalized learning and adaptive tutoring systems. Finally, the authors also give some ideas for evaluating and validating the graph's accuracy and relevance.

Despite the relatively wide use of automatic learning materials in the programming domain, notable limitations remain, which should be addressed for the technology to reach its full potential in the most current applications. They include lack of knowledge representation, knowledge structure, flexibility, context awareness, content reusability, and depth of understanding. Current systems often require human oversight to ensure quality and still lack the interactivity, personalization, and problem-solving skills that come with human instruction. Continued AI development, especially contextual understanding, adaptability, and soft skill integration, will be crucial for overcoming these limitations. Table I compares the current approaches (traditional approaches) and ontology-based approaches to automatic learning materials generation in the programming domain. Traditional approaches are generally linear and less flexible and can struggle with scalability and personalization. They tend to rely on static content structures. However, ontology-based approaches leverage semantic relationships to create more dynamic, adaptable, and personalized learning experiences. The main thing they provide is enhanced interoperability and support of collaborative learning.

III. Ontology-Based Approach for Learning Materials Generation

Formal knowledge representation is used in an ontology-based approach that captures domain-specific concepts, relations, and properties and uses such information to generate learning materials. The method involves an ontology for the target domain's concepts, relationships, and properties, such as programming languages. Semantic understanding is captured through ontology, meaning it results in inferring relationships and categorizing concepts. Learners' needs and preferences are analyzed based on educational objectives and learner profiles. The ontology is used to generate content that is coherent and contextually relevant. The materials are presented using natural language processing techniques to make the explanation as clear and understandable as possible. Because it is based on ontology, it allows for continuous updating and refinement as the domain knowledge changes. The benefits include scalability, adaptability, personalization, consistency, efficiency, and accessibility. The ontology-based approach can create adaptive, personalized, high-quality educational content for various domains, such as programming education.

TABLE I. COMPARISON BETWEEN THE TRADITIONAL APPROACHES AND ONTOLOGY-BASED APPROACHES

Feature/Aspect	Traditional Approaches	Ontology-based Approaches	References
Knowledge Structure	linear and hierarchical	semantic and interconnected	[16], [19]
Flexibility	limited adaptability to new topics	highly adaptable to new knowledge and domains	[6], [13]
Context Awareness	minimal context consideration	rich context understanding through relationships	[22], [23]
Content Reusability	low reusability of materials	high reusability due to modular components	[10], [14]
Personalization	basic customization, often static	dynamic personalization based on learner profiles	[5], [24]
Scalability	difficult to scale with growing content	easily scalable with ontological frameworks	[7], [25]
Interoperability	often siloed systems	enhanced interoperability across platforms	[17], [26]
Knowledge Representation	simple data structures (e.g., text, images)	rich semantic representation using classes, properties, and relationships	[9], [27]
Maintenance	time-consuming updates and revisions	more accessible updates due to modular ontology design	[28], [29]
Collaboration Support	limited collaboration features	facilitates collaboration through shared ontologies	[1], [10]
Learning Pathways	predefined and rigid learning paths	dynamic learning pathways based on learner needs	[4], [17]
Assessment Tools	basic quizzes and tests	adaptive assessments based on learner progress	[8], [15]
Feedback Mechanism	limited feedback based on performance	contextual feedback based on semantic analysis	[20], [30]

The ontology-based approach for generating learning materials involves structured knowledge representations on a domain to automatically create the learning materials. Ontologies are leveraged in this process to map the relationships between different concepts in the subject of a knowledge domain, providing generated materials that are pedagogically sound and contextually relevant. The primary process of generating learning materials using an ontology-based approach can be demonstrated in several steps as follows:

1) Ontology development, which includes domain analysis, is to identify the key concepts, relationships, and rules within the subject area, and ontology construction to define the concepts (classes), properties (relationships), and instances (individuals) within the domain, and validation and refinement ensure that the ontology accurately represents the domain knowledge through validation and iterative refinement.

2) Knowledge representation involves formalizing the ontology. This formal language provides precise semantics for the concepts and relationships, axioms, and rules to define axioms and inference rules to capture the logical constraints and derivations within the domain.

3) Learning materials generation, which contains the content extraction for identifying relevant content from the ontology based on the learning objectives, content structuring to organize the extracted content into a coherent structure, following educational best practices (e.g., Bloom's taxonomy), and template application to apply predefined templates to format the content into various types of learning materials (e.g., textbooks, task assessments, interactive modules).

4) Automated generation algorithms include the input processing to accept inputs such as learning objectives, target audience, and preferred content format; ontology querying, which uses description logic (DL) queries to retrieve relevant concepts, relationships, and instances from the ontology, material assembly to assemble the retrieved information into structured learning materials using the defined templates, and output generation for producing the final learning materials in the desired format (e.g., HTML, e-learning platform).

Automatically generating learning materials involves a complex pipeline integrating natural language processing (NLP), machine learning, and educational technology. The following is an algorithmic approach to automatically generating learning materials from an ontology. Automatically generating learning materials in the programming domain involves several tailored steps. The following is a proposed algorithm for automatic learning material generation in the programming domain:

Inputs:

- Programming Language: The specific language (Python).
- Learning Objectives: Skills or concepts to be covered (e.g., syntax, data structures, algorithms).
- Content Sources: Online tutorials, documentation, coding.
- Format Preferences: Desired output formats (e.g., code snippets, quizzes, video tutorials).
- Target Audience: Beginner, intermediate, or advanced learners.

Steps:

1) Content retrieval:

- Query content sources using APIs or web scraping to gather relevant programming resources.
- Use NLP techniques to filter and categorize content based on relevance and complexity.

2) Content analysis:

- Analyze the retrieved content for key programming concepts, syntax rules, common pitfalls, and best practices.
- Identify gaps in the content that need to be addressed to fulfill the learning objectives.

3) Content structuring:

- Organize the content into a logical flow, such as:
- Introduction to the language
- Basic syntax and constructs
- Control structures (loops, conditionals)
- Data structures (arrays, lists, dictionaries)
- Functions and modules
- Advanced topics (e.g., OOP, frameworks)
- Create outlines or flowcharts to visualize the structure.

4) Material creation:

- Generate text explanations for each section using NLP techniques.
- Create code examples and snippets that illustrate each concept.
- Develop quizzes or coding challenges based on the key concepts identified.
- Design multimedia elements (like screencasts or infographics) if applicable.

5) Customization:

- Tailor the generated materials to fit the target audience's skill level.
- Adjust complexity by simplifying explanations or introducing advanced topics as needed.

6) Interactive elements:

- Integrate coding environments (like Jupyter Notebooks or online IDEs), where learners can practice coding directly within the material.
- Include live coding demonstrations or interactive simulations.

7) Feedback loop:

- Incorporate user feedback mechanisms (like quizzes and surveys) to evaluate understanding and engagement.
- Use machine learning to refine content generation based on user performance data.

8) Output generation:

- Compile all materials into a cohesive format (e.g., HTML pages, PDF documents, online course modules).
- Ensure accessibility standards are met (e.g., code readability, alt text for images).

9) Review and iteration:

- Implement a review process, where educators or experienced programmers can evaluate the generated materials.
- Iterate on the content based on feedback and updates in programming language features or best practices.

Outputs:

- Comprehensive learning materials tailored to programming topics and audiences.
- Code snippets and examples for hands-on practice.
- Quizzes and coding challenges to reinforce learning.

Considerations:

- Ethics and Copyright: Ensure all content respects copyright laws and ethical guidelines.
- Diversity and Inclusion: Include diverse perspectives and examples in the programming context.
- Technology Integration: Consider integrating learning management systems (LMS) or coding platforms for easy distribution and tracking.

Example Use Case:

1) Input:

- Programming Language: "Python"
- Learning Objectives: Understand basic syntax, functions, and data structures.
- Format Preferences: Text explanations, code examples, quizzes.
- Target Audience: Beginners.

2) Output:

- A structured document explaining Python basics with annotated code snippets.
- A set of quizzes covering key points about Python syntax and functions.
- Links to interactive coding environments for practice.

Fig. 1 shows a summary flowchart of creating and managing Python learning materials. After processing several inputs, such as learning objectives and content sources, through steps including content retrieval and structure, it produces learning materials that are accessible, interactive, and customizable.

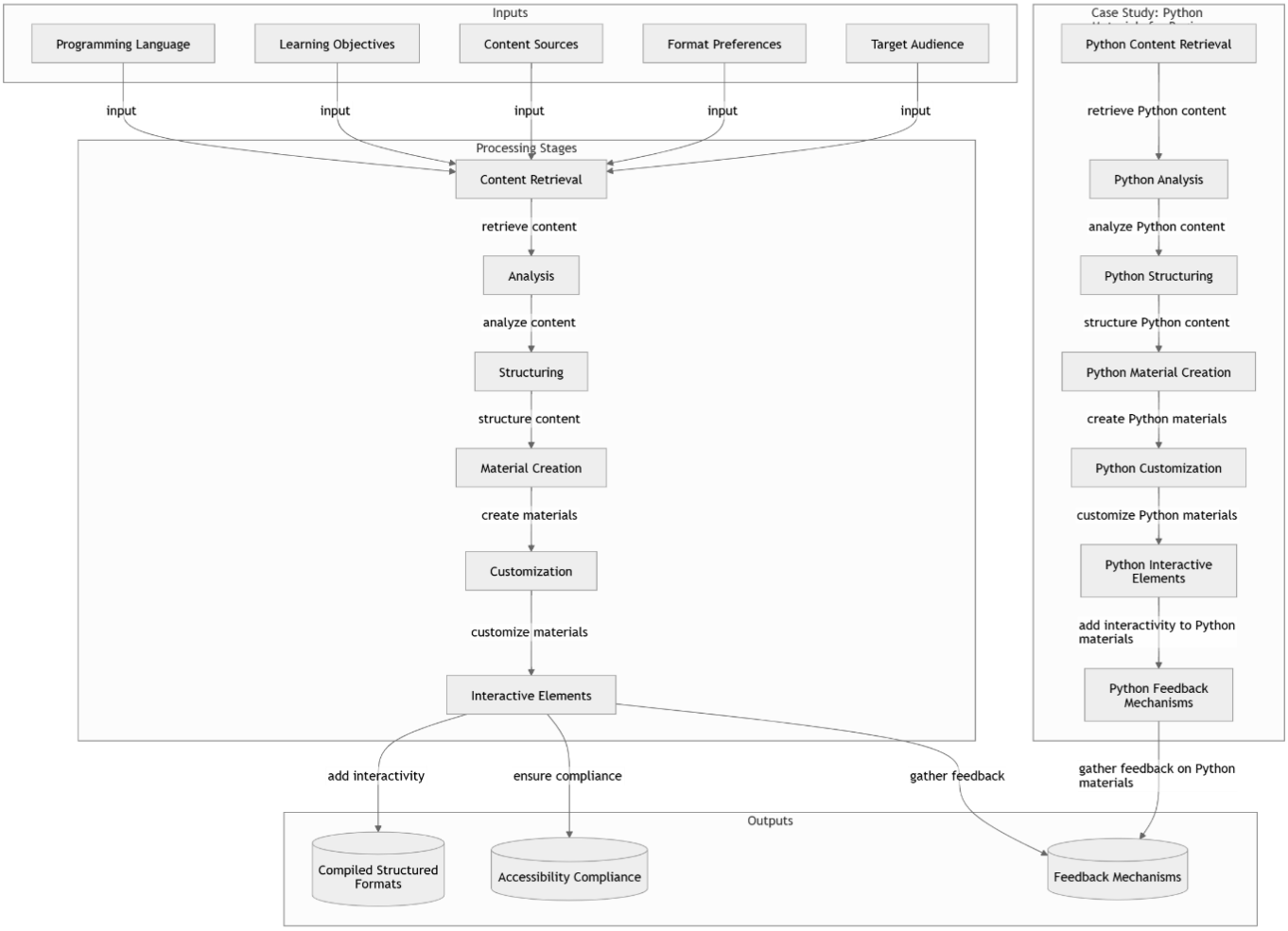

Fig. 1. Creating and managing learning materials for Python.

IV. Proposed Knowledge Model for the Domain-Specific Concepts

The domain-specific concept is the system's knowledge module, organizing the domain knowledge structure, including its central concepts and their relationships. This model facilitates the automatic generation of learning materials for the educational process. It focuses on constructing and organizing domain-specific concepts and their interrelations [29]. A knowledge module consists of guidelines to identify all vocabulary concepts to illustrate or solve problems. It is purely declarative and does not provide instructions on how learners can utilize it to address practical issues [31]. Two categories of ontology modules have been developed based on the characteristics of the learning materials: general domain-specific concepts ontology and specific domain-specific concepts knowledge module ontology. These modules represent the knowledge concepts needed for learning, provide input to the knowledge module, offer particular feedback, select problems, create learning materials, and support the student model. A domain-specific concepts knowledge module has been proposed based on current research, as illustrated in Fig. 2. This model is fundamentally based on domain concepts, properties, task assessments, material resources, learning objectives, learning rules, learning levels, and their interrelationships. To generate learning materials and reuse the knowledge module in the learning process, ontologies organize and represent the domain-specific concepts knowledge module. The advantage of this model is its ability to personalize and automatically generate learning materials for learners. Based on the general domain-specific concepts ontology shown in Fig. 2, domain concepts, domain properties, task assessments, material resources, learning objectives, learning rules, and learning levels terminologies refer to the following:

- Domain concepts present domain-specific knowledge or a comprehensive learning material or course overview.
- Domain properties represent a learning material or domain-specific properties within a domain knowledge model.
- Task assessments explain how the application system can assess or measure the required learner activities within a specific period.
- Material resources are physical or digital items used in educational settings to support and facilitate learning. They include textbooks, web resources, software, multimedia tools, and laboratory equipment.

- Learning objectives are clear, measurable goals that outline students' expected learning outcomes. They guide teachers in planning instruction, designing assessments, and evaluating progress. Aligned with curriculum and instructional standards, they provide a framework for effective teaching and assessment.
- Learning rules are principles or guidelines that describe how learning occurs and how new information is acquired and processed. These rules help educators understand student learning and inform instructional strategies while helping students become more effective learners by optimizing their learning processes.
- Learning levels are the stages of proficiency and understanding individuals progress through as they acquire new knowledge, skills, and competencies. They are crucial in education and instructional design, as they help educators tailor teaching methods and materials to support students at different stages of their learning journey.

Fig. 3 displays the design and structure of a selected ontology knowledge module for the domain-specific concepts case study for the Python programming domain. Several relationships are applied to the domain-specific concepts selected in case examples. The relationships are generalization or specialization, dependency, and containment. Containment indicates that a specific domain concept within a given domain contains various concepts (has-a). The generalization or specialization shows particular topics or domains with specific concepts (is-a). Based on Fig. 2 and Fig. 3, the following displays a temporary explanation of a domain concept:

- Domain concepts: Class, Function.
- Domain properties: syntax.
- Task assessments: program, code review, project.
- Material resources: textbooks, web resources.

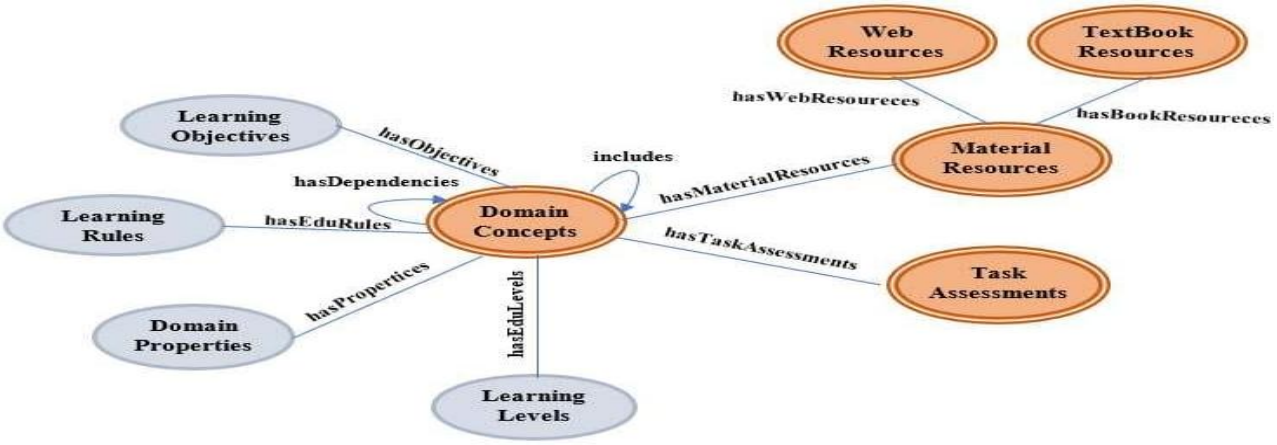

Fig. 2. Knowledge model for the domain-specific concepts.

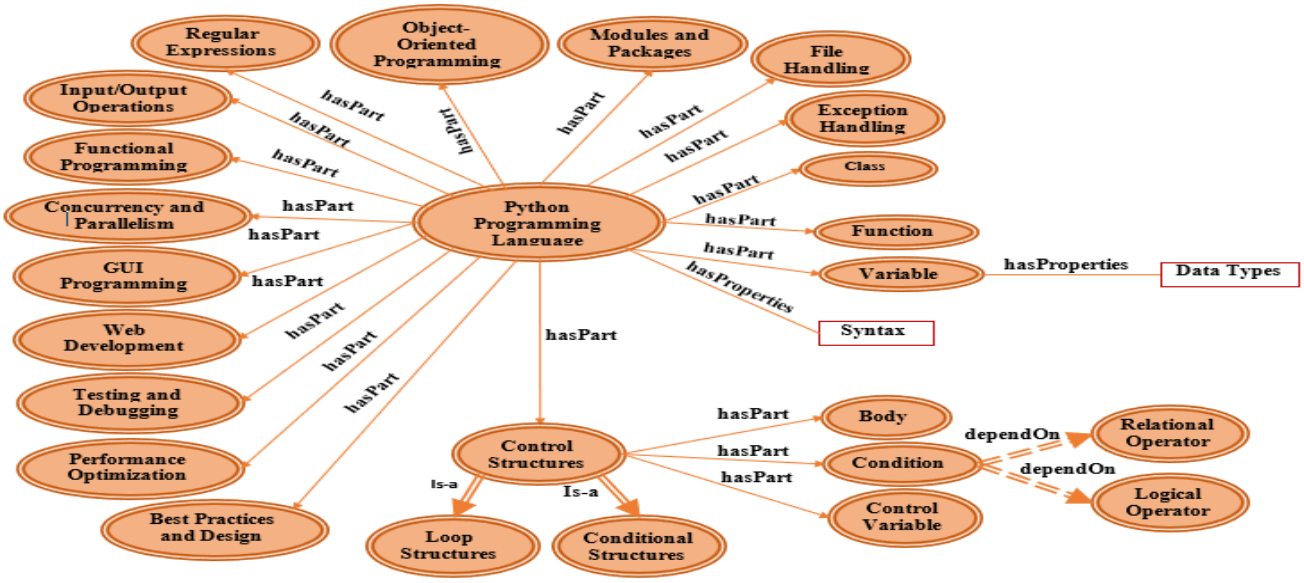

Fig. 3. Specific knowledge model for the domain-specific concepts.

V. Proposed Model Implementation

Computer Science and Information Technology disciplines offer numerous language modules and packages for developing and managing ontology models. Python is one of the most widely used and favored languages for implementing an ontology for domain-specific concept models. This interpreted, object-oriented, and extensible programming language is known for its exceptional clarity and versatility across various fields [22]. In [23], the authors used Python and Owlready2 to create the ontology and implement the domain knowledge. In this work, the domain-specific concept explored is the "Basics of Computer Programming", the ontology is constructed using the "Python Programming Language." The Python and Owlready2 modules implement domain-specific concepts within the ontology. Owlready2 facilitates transparent access to ontologies, allowing for the manipulation of classes, individuals, object properties, data properties, annotations, property domains, ranges, constrained datatypes, disjoints, and class expressions, including intersections, unions, property value restrictions, and more. Python offers some functions and modules for managing ontology to implement, create, and modify ontologies. The get_ontology() function allows building an empty ontology from its IRI using the Owlready2 module. Owlready2 uses the syntax "with ontology: ..." to demonstrate the ontology that will receive the new RDF triples. For creating an ontology, the following short code is used:

```
from owlready2 import *

ontology = get_ontology()

with ontology: <Python code>
```

Concerning the implementation of the domain-specific concepts and the construction of its components: the domain concepts, learning objectives, domain properties, task assessments, learning rules, material resources, and learning levels. Fig. 4 shows a code dealing with the design of the core classes of the presented model. Fig. 5 corresponds with some of the object property relationships defined for the constructed components of the selected model. Several tools are available to display the ontology graph. The tools are Synaptica, OWLGrEd, and Protégé. Protégé is the most commonly used tool to display the ontology graph of domain-specific concepts, as shown in Fig. 6. The circular relationship lines in Fig. 6 mean that each topic can depend on another topic and contain subtopics. For example, the iterative loop depends on variables, logical operators, and relational operators. Control sentences contain conditional sentences and iterative sentences. Fig. 7 presents a SPARQL query as an example of visualizing all the domain concepts in the selected ontology domain-specific concepts regarding retrieving the domain concept and its description.

```
ontology = get_ontology("http://test.org/Domain_Specific_Concepts.owl#")
#Construction of the Domain Specific Concepts Components
with ontology:
    class DomainConcepts(Thing):
        def take():
            print("I take Domain Concepts")
    class LearningObjectives (Thing):
        def take(self):
            print('Learning Objectives')
    class DomainProperties (Thing):
        def take(self):
            print('Domain Properties related to the Domain Concept')
    class TaskAssessments (Thing):
        def take(self):
            print('Task Assessments related to the Domain Concept')
    class LearningRules (Thing):
        def take(self):
            print('Learning Rules related to the Domain Concept')
    class MaterialResources(Thing):
        def take(self):
            print('material resource related to the Domain Concept')
    class TextBookResources(MaterialResources): pass
    class WebResources(MaterialResources): pass
    class LearningLevels (Thing):
        def take(self):
            print('Learning Levels related to the Domain Concept')
```

Fig. 4. Core classes of the presented model.

```
    #The Object Property Relationships added to the Domain Specific Concepts
    class hasPart(DomainConcepts >> DomainConcepts): pass
    class partOf(DomainConcepts >> DomainConcepts):
        inverse = hasPart
    class hasDependency(DomainConcepts >> DomainConcepts): pass
    class dependencyOf(DomainConcepts >> DomainConcepts):
        inverse = hasDependency
    class associate(DomainConcepts >> DomainConcepts): pass
    class associatedBy(DomainConcepts >> DomainConcepts):
        inverse = associate
    class hasParent(DomainConcepts >> DomainConcepts): pass
    class parentOF(DomainConcepts >> DomainConcepts):
        inverse = hasParent
    class hasProperty(DomainConcepts >> DomainProperties): pass
    class propertyOf(DomainProperties >> DomainConcepts):
        inverse = hasProperty
    class hasTask(DomainConcepts >> TaskAssessments):pass
    class taskOf(TaskAssessments >> DomainConcepts):
        inverse = hasTask
    class hasMaterial(DomainConcepts >> MaterialResources): pass
    class materialOf(MaterialResources >> DomainConcepts):
        inverse = hasMaterial
    class hasDRule(DomainConcepts >> LearningRules): pass
    class druleUOf(LearningRules >> DomainConcepts):
        inverse = hasDRule
```

Fig. 5. Object property relationships.

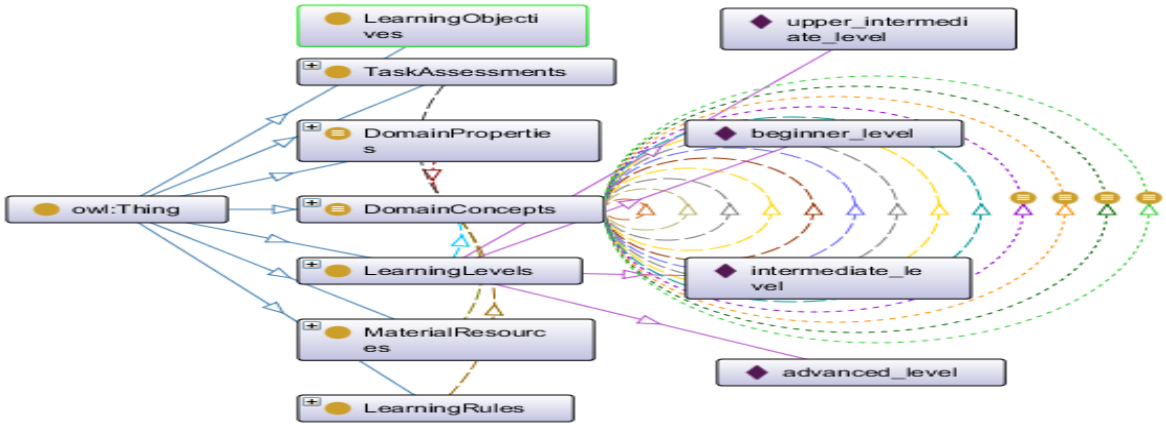

Fig. 6. Domain-specific concepts ontology graph.

```
result = """PREFIX dn: <http://test.org/Domain_Specific_Concepts.owl#>
   SELECT DISTINCT ?domain ?description
                        WHERE {
                        ?d a dn:DomainConcepts;
                        dn:domainName ?domain;
                        dn:domainDescription ?description.
                        }
    """
qres = g.query(result)
for row in qres:
    print('_____')
    print(f"Domain Concept: {row.domain}:\nDomain Description:\n{row.description}")
```

```
Domain Concept: Python Class:
Domain Description:
A class is a user-defined blueprint or prototype from which objects are created. Classes provide
    a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new
    instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state.
    Class instances can also have methods (defined by their class) for modifying their state.
```

Fig. 7. A SPARQL query for retrieving the concept "python class" and its description.

We use natural language processing for automatic learning material generation, applying the spacy module in Python and the rdflib module. Fig. 8 and Fig. 9 present the code that controls the ontology of domain-specific concepts. Fig. 10 and Fig. 11 display snapshots of SPARQL for generating task assessment and query results according to SPARQL selecting concepts. The results are domain concepts, task assessment, and ask questions in the form of multiple-choice questions. Regarding automatic learning materials generation, the system randomly generates task assessments as multiple-choice questions for the learner. The learner is asked to answer the question, and according to the answer, whether it is correct or not, the system will automatically generate learning materials for further reading. Fig. 12 shows a snapshot of a task assessment question, whether the answer is correct, and the suggested learning material for the selected task.

```
import rdflib
import spacy
from spacy.lang.en import English

# Load the English NLP model
nlp = English()

# Load the ontology
g = rdflib.Graph()
g.parse("dataset/update_py_onto_module.owl", format="xml")

# Extract concepts from the ontology
concepts = [str(concept) for concept in g.subjects()]

# Process the concepts using the NLP model
learning_materials = []
for concept in concepts:
    doc = nlp(concept)
    # Generate learning materials based on the processed concept
    definition = f"The term '{concept}' refers to {doc[0].text.lower()}."
    learning_materials.append(definition)

# Print the generated learning materials
for material in learning_materials:
    print(material)
```

Fig. 8. Controlling the ontology of domain-specific concepts.

```
The term 'http://test.org/Domain_Specific_Concepts.owl#c' refers to http://test.org/domain_specific_concepts.owl#c.
The term 'http://test.org/Domain_Specific_Concepts.owl#LearningObjectives' refers to http://test.org/domain_specific_concepts.owl#learningobjectives.
The term 'http://test.org/Domain_Specific_Concepts.owl#task1' refers to http://test.org/domain_specific_concepts.owl#task1.
The term 'http://test.org/Domain_Specific_Concepts.owl#intermediate_level' refers to http://test.org/domain_specific_concepts.owl#intermediate_level.
The term 'http://test.org/Domain_Specific_Concepts.owl#levelDescription' refers to http://test.org/domain_specific_concepts.owl#leveldescription.
The term 'http://test.org/Domain_Specific_Concepts.owl#propertyName' refers to http://test.org/domain_specific_concepts.owl#propertyname.
The term 'http://test.org/Domain_Specific_Concepts.owl#ruleSyntax' refers to http://test.org/domain_specific_concepts.owl#rulesyntax.
The term 'http://test.org/Domain_Specific_Concepts.owl#associatedBy' refers to http://test.org/domain_specific_concepts.owl#associatedby.
The term 'http://test.org/Domain_Specific_Concepts.owl#hasProperty' refers to http://test.org/domain_specific_concepts.owl#hasproperty.
The term 'http://test.org/Domain_Specific_Concepts.owl#taskCatogary' refers to http://test.org/domain_specific_concepts.owl#taskcatogary.
The term 'http://test.org/Domain_Specific_Concepts.owl#ruleID' refers to http://test.org/domain_specific_concepts.owl#ruleid.
```

Fig. 9. The result of the ontology of domain-specific concepts.

```
SELECT DISTINCT ?domain ?task ?question ?ans1 ?ans2 ?ans3 ?ans4
                WHERE {
                ?d a dn:DomainConcepts; dn:hasTask ?t;
                dn:domainName ?domain.
                ?t dn:taskName ?task;
                dn:questionSchema ?question;
                dn:a ?ans1;
                dn:b ?ans2;
                dn:c ?ans3;
                dn:d ?ans4.
                }
```

Fig. 10. Task assessment generation.

```
Domain Concept: Python Class: Task Assessment: Python classes
Task Questions What is inheritance in Python classes?
a) The process of creating multiple instances of a class
b) The process of passing attributes and methods from one class to another
c) The process of deleting a class object
d) The process of defining new methods in a class
```

```
Domain Concept: Python Class: Task Assessment: Python classes
Task Questions What is the output of the following Python code?
                class MyClass:
                    def __init__(self, value):
                        self.value = value
                obj = MyClass(10)
                print(obj.value)

a) 0
b) 10
c) Error
d) None
```

Fig. 11. MCQs task assessment.

```
Task: Python classes
Task Question: What is a class in Python?
 a) A module
 b) A function
 c) A template for creating objects
 d) An array
your answer is: c
your answer is correct, because it match the system answer
you can learn more about this domain in the material: Python Classes and Objects  from the following Resources
 https://www.geeksforgeeks.org/python-classes-and-objects/
```

Fig. 12. Task assessment and result sample.

Fig. 13 shows a system that uses an ontology-based method to generate adaptive learning materials and quizzes. It illustrates how an ontology of concepts and relationships guides the development of personalized quizzes and learning paths suited to different competence levels. At the same time, learner progress informs knowledge gap analysis and topic selection. The Python programming ontology is a hierarchical system that maps out Python concepts, relationships, and learner progression. It includes fundamental concepts like variables, data types, and functions. The system infers a learner's proficiency level based on how they perform in quizzes and assessments. The ontology can be modified dynamically with performance-related data. In addition, it provides data analytics on tracker progress, predictive analytics, and content optimization. The ontology-based quiz creation process is dynamic and automatic, using Python concepts and learning objectives. It integrates with the learning path generator that selects the questions depending on the learner's progress. The system can accommodate questions such as multiple choice, true or false, fill-in blanks, code snippets, and coding challenges for promoting knowledge retention and skill development. The traditional way of producing materials and questions is to establish the scope and topic sets, acquire information and resources, structure the content, build learning materials, build assessment questions, and create specific examples. The instructor could use book texts, online resources, or even their teaching notes to extensively deal with functions, parameters, return values, and scope. The content is divided into an introduction to the function, a function definition, parameters and arguments, return value, and function scope. There are text-based learning materials, code-based learning materials, visuals, and exercises. Assessment questions can be multiple choice, code analysis, or code writing. Table II shows a comparison between traditional

versus ontology-based learning material creation. Examples include defining functions using the "return" statement and questioning about parameters in a function. This approach emphasizes the reliance on the instructor's knowledge and the step-by-step process of translating that knowledge into learning resources. The following is a case study considering the following code:

```
def add_numbers(x, y):
    result = x + y
    return result
sum = add_numbers(5, 3)
print(sum)
```

What is the purpose of parameters in a function?

a) To give the function a name.

b) To allow the function to accept input values.

c) To specify the data type of the return value.

d) To control the order in which code is executed.

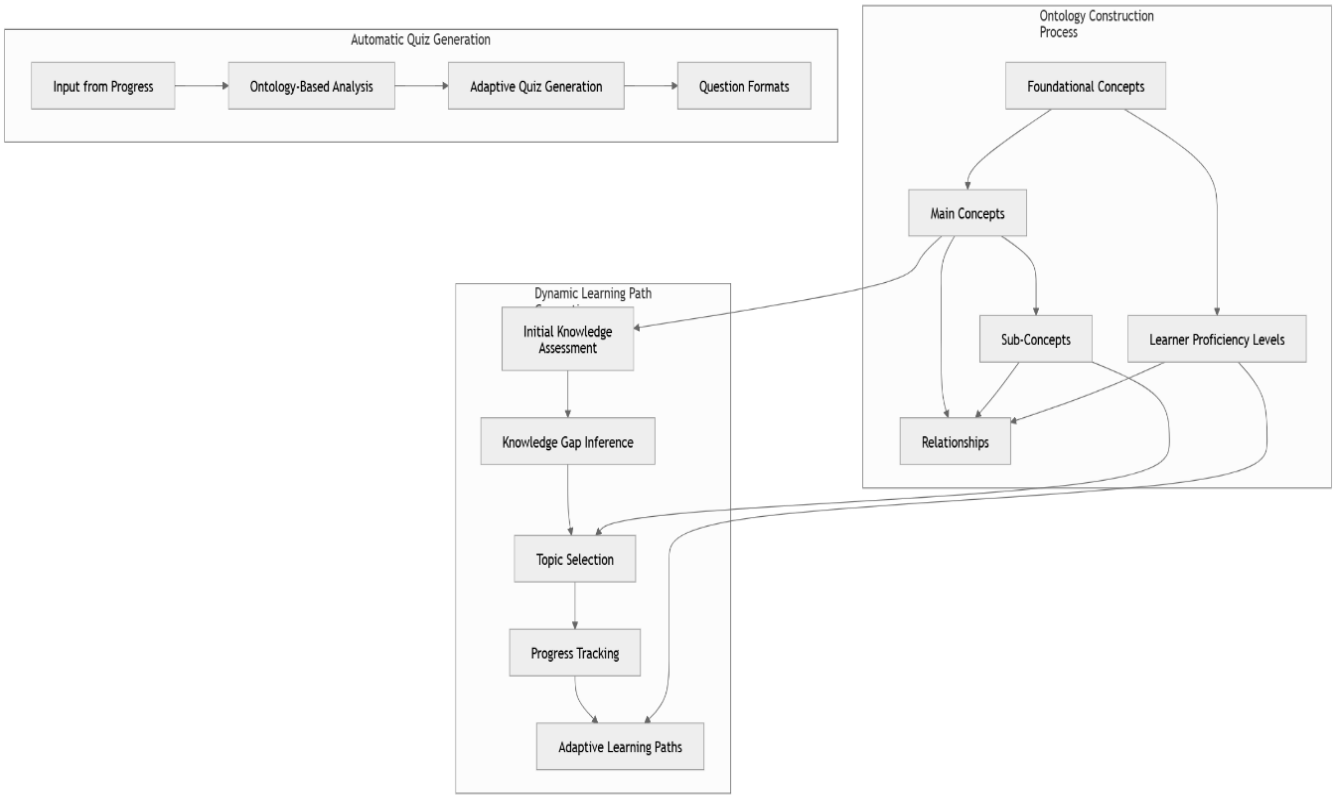

Fig. 13. Ontology-based method to generate adaptive learning materials and quizzes.

VI. Proposed Ontology-Based Model Validation and Evaluation

For ontology-based model validation and evaluation, various tools can be utilized to ensure the ontology's accuracy, consistency, completeness, and pedagogical effectiveness. Using these tools, you can comprehensively validate and evaluate ontology-based models to ensure high-quality, effective learning materials. A robust continuous improvement framework is based on combining automated tools with expert reviews.

1) Ontology evaluation: Ontology evaluation tools are important in assessing ontology's quality, reliability, and utility in many domains [30]. Ontology quality is measured with several metrics and methods, including quality metrics, consistency checkers, structural analysis tools, domain-specific evaluation tools, and usability evaluation tools [30]. Moreover, these tools also maintain the integrity and usefulness of ontologies across different domains. Automation, usability, interoperability, domain-specific adaptations, and capabilities for dynamic evaluation can be improved [30]. IRI_Debug is an ontology evaluation tool that enables the detection and correcting of issues in the Internationalized Resource Identifiers (IRIs) [28]. It provides IRI validation, validation of errors, consistency checking, namespace control, and an easy-to-use interface [28]. However, it is unsatisfactory due to the effectiveness of ontology complexity and IRI usage patterns in ontology development, maintenance, and educational use. Continuous updates are necessary for evolving standards [28]. Owlready2 offers many reasoners for manipulating the domain ontology, such as Pellet, ELK, and HermiT. The HermiT reasoner is used, as shown in Fig. 14, to check that the constructed ontology is consistent and allows the classification, instance checking class satisfiability, and conjunctive query answering

of the developed domain ontology for the selected model. It is most commonly used in ontology engineering.

TABLE II. COMPARISON BETWEEN THE TRADITIONAL APPROACHES AND ONTOLOGY-BASED APPROACHES

Feature	Traditional Learning Material Creation	Ontology-Based Learning Material Creation
Content Organization	Linear and structured manually	Hierarchical and dynamically structured using ontology
Customization	Limited personalization	Highly personalized based on learners' needs
Content Reusability	Low content created from scratch	High, modular content reuse across different topics
Automation	Mostly manual work	AI-assisted generation and annotation
Content Consistency	It can be inconsistent across materials	Ensures uniform structure and terminology
Adaptability	Hard to update and adapt	Easily adaptable to new knowledge and learning trends
Efficiency	Time-consuming	Faster and more efficient due to automation
Interactivity	Mostly static content	Dynamic and interactive learning experiences
Scalability	Difficult to scale	Easily scalable across different subjects and learners

2) Ontology validation: Ontology validation tools ensure ontologies' quality, reliability, and usability [32]. They identify issues related to consistency, completeness, correctness, and adherence to best practices [32]. Popular tools include OOPS!, OntoQA, OQuaRE, Pellet and Hermit, OntoMetric, BioPortal and AgroPortal, and OntoClean. OOPS! is a tool that helps ontology developers identify and address common pitfalls in ontology design [33]. It uses a set of pitfalls from best practices and expert recommendations, covering naming conventions, ontology structure, and logical inconsistencies [33]. The tool generates detailed reports detailing pitfalls, severity, and affected elements and provides recommendations for correcting each [33]. It can be integrated into ontology environments like Protégé, enhancing usability and promoting best practices [33]. Fig. 15 shows the OntOlogy Pitfall Scanner tool for ontology validation, which is used for the validation process. The input values for this tool can be ontology URL or RDF file code. Fig. 16 shows the OntOlogy Pitfall Scanner tool validation results.

```
owlready2.JAVA_EXE = "C:\\Program Files\\Java\\jre-1.8\\bin\\java.exe"
try:
    sync_reasoner()
    print("Ok, the constructed ontology is consistent and allows the classification, instance checking, class satisfiability, and  conjunctive query answe
except OwlReadyInconsistentOntologyError:
    print("The constructed ontology is inconsistent! and didn't allow the classification, instance checking, class satisfiability, and conjunctive query a

* Owlready2 * Running HermiT...
    C:\Program Files\Java\jre-1.8\bin\java.exe -Xmx2000M -cp C:\Users\jshbo\Python\Python311\Lib\site-packages\owlready2\hermit;C:\Users\jshbo\Python\Pyt
hon311\Lib\site-packages\owlready2\hermit\HermiT.jar org.semanticweb.HermiT.cli.CommandLine -c -O -D -I file:///C:/Users/jshbo/AppData/Local/Temp/tmp_grk
j_bf
Ok, the constructed ontology is consistent and allows the classification, instance checking, class satisfiability, and  conjunctive query answering.
* Owlready2 * HermiT took 1.0713214874267578 seconds
* Owlready * Reparenting Domain_Specific_Concepts.DomainProperties: {owl.Thing} => {Domain_Specific_Concepts.DomainConcepts}
* Owlready * (NB: only changes on entities loaded in Python are shown, other changes are done but not listed)
```

Fig. 14. Consistency of the domain-specific concepts ontology.

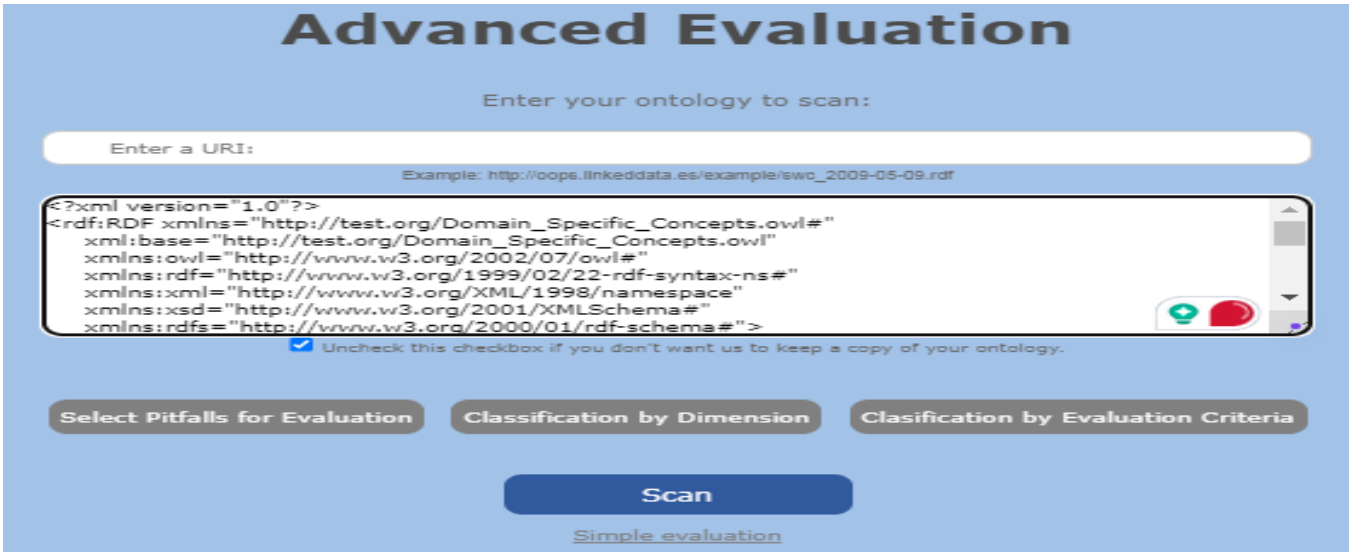

Fig. 15. Ontology pitfall scanner tool.

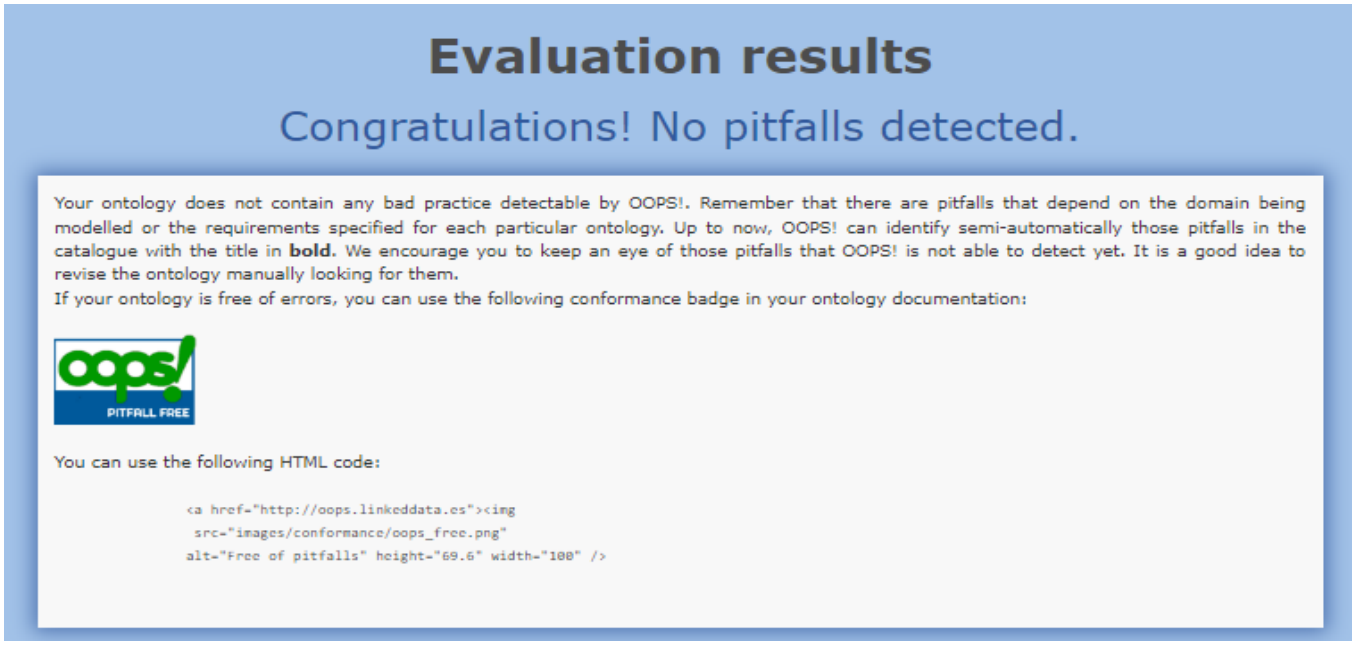

Fig. 16. Ontology pitfall scanner tool results.

VII. Results

The ontology-based automatic generation of learning materials in the Python programming domain as a solution provides a more sophisticated system for generating learning materials. Assessing their quality accuracy, 98.5%, makes it a valuable tool in educational technology and content generation. The dataset used in this experiment is Python programming language ontology [34]. To generate the learning materials, we used BERT embeddings to measure the semantic similarity of generated learning materials to predefined reference materials. It also generates an evaluation table, Table III, summarizing the results for each domain concept, as explained in the following steps:

1) Ontology and learning materials: We define an ontology for various domain concepts (e.g., Python Programming, Data Structures) and generate learning materials for each domain concept using predefined content.

2) BERT-based accuracy calculation: We use the BERT model from the sentence-transformers library to compute embeddings for the generated learning materials and predefined reference materials. We then calculate the cosine similarity between these embeddings to determine the semantic accuracy of the generated content.

3) MCQ Generation: We generate multiple choice questions (MCQs) for each domain concept and assess how much the learner understands it.

4) Evaluation table: Table III shows how the create_evaluation_table function collected generated learning materials, accuracy scores, MCQs, and a brief description of results from the results set into a structured evaluation table with the help of pandas. Descriptions of the accuracy are offered as a categorical measure based upon the thresholds, "Excellent alignment" being the case when the accuracy is greater than 90%, "Good alignment" for anything from 70% to 90%, and "Moderate alignment" for a value that is less than 70%.

Table IV compares the ontology-based model's performance across numerous samples of the Python programming topic: Data Types, Control Flow, Functions, Error Handling, and OOP (Object-Oriented Programming) respectively. It proves how effectively the system can generate learning materials and assessments for each topic. As shown in Table V, the ontology-based model's performance also changes according to the dataset size when presented with the task of generating Python programming learning materials. It shows accuracy and other improvements as the model processes more datasets and proves its scalability. Using the following formulas, we have calculated the evaluation metrics such as accuracy, precision, recall, and F1-Score:

- Accuracy = (True Positives + True Negatives) / (Total Instances)
- Precision = True Positives / (True Positives + False Positives)
- Recall = True Positives / (True Positives + False Negatives)
- F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

Data is split into training (80%), and testing (20%) sets using the train_test_split function from sklearn.model_selection. The final parameter is the split with test_size=0.2, and random_state=42 ensures reproducibility. Using dataset size, the training and testing percentages are calculated. The values for these datasets are explicitly defined and printed in the run_evaluation function to make it clear for model training and evaluation dataset distribution. In this case, the accuracy calculation was measured using the BERT-based

semantic similarity. A pre-trained BERT model was used to transform the generated and reference texts into vector embeddings. These embeddings were computed into cosine similarity values measuring their semantic closeness. A predefined threshold was set to verify if the generated content was accurate (e.g., 0.8 or 0.9). The accuracy was calculated as the percentage of the correctly matched samples over the total number of samples.

TABLE III. EVALUATION TABLE SAMPLE

Domain Concept	Generated Learning Material	Accuracy Score (%)	MCQs	Description
Python Programming	Python is a versatile programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.	98.50%	Q: What keyword is used to define a function in Python? - def - function - func - define Answer: def	Excellent alignment with reference material.
Data Structures	Common data structures in Python include lists, dictionaries, sets, and tuples. Each structure has unique properties and use cases.	95.85%	Q: Which of the following is an unordered collection in Python? - List - Tuple - Dictionary - String Answer: Dictionary	Excellent alignment with reference material.
Algorithms	Algorithms are step-by-step procedures for solving problems. In Python, you can implement algorithms for sorting, searching, and manipulating data in Python.	92.30%	Q: What is the time complexity of binary search? \n - O(n) \n - O(log n) \n - O(n log n) Answer: O(log n)	Excellent alignment with reference material.

TABLE IV. ONTOLOGY-BASED MODEL EVALUATION: PYTHON PROGRAMMING TOPICS SAMPLE

Python Topic	Number of examples	Percentage	Accuracy	Precision	Recall	F1-Score
Data Types (int, float, str)	390	39%	0.95	0.93	0.96	0.94
Control Flow (if, else, loops)	170	17%	0.91	0.89	0.92	0.90
Functions (def, arguments, return)	70	7%	0.93	0.91	0.94	0.92
Error Handling (try, except)	70	7%	0.89	0.86	0.91	0.88
Object-Oriented Programming (OOP)	360	36%	0.90	0.87	0.92	0.89

TABLE V. ONTOLOGY-BASED MODEL EVALUATION PERFORMANCE BY DATASET SIZE

Dataset Size (Records)	Accuracy	Precision	Recall	F1-Score
Small (500)	0.88	0.85	0.89	0.87
Medium (1500)	0.91	0.89	0.92	0.90
Large (5000)	0.985	0.92	0.95	0.93

VIII. DISCUSSION

Ontology-based automatic learning material generation is a technology that has the potential to enhance learning experiences in almost any educational environment greatly. From an instructor's point of view, it operates as an adaptive tool that can initiate customized tests based on the students' diagnostic results. In this way, it enables the emergence of personalized learning materials directed to certain weak spots and saves quiz creation and grading time.

This tech can provide a personalized learning path for learners, particularly Python programming students. An independent learner might start with a diagnostic test that covers basic topics such as data types, control flow, and functions. It can create debug tasks, discussions, and interactive lessons personalized to the student's needs based on their performance. The system also generates automatic feedback to highlight task errors, syntax errors, and possible solutions for student advancement. The instructor can use the same feedback to identify challenges faced by students and correspondingly grade the difficulty level of exercises so that support may be made more specific.

This technology is excellent for use in both self-paced and instructor-led learning environments. In a blended learning model, for example, a self-paced learner could work through the function modules, and an instructor could give the diagnostic quizzes to track progress. The system provides real-time performance data, which allows educators to monitor student advancement and uncover the need to focus, once necessary, on the individual. In addition, it is beneficial to advanced learners who are learning Pandas for data manipulation. Real-world datasets have complex tasks that are tough enough for expert programmers. This ontology-based approach allows instructors to customize the learning content to particular learning goals to improve the learning experience.

Automatic generation of learning material based on ontology can enhance personalized learning, including adaptive content generation, real-time feedback, and performance analytics. However, integration into Learning Management Systems such as Moodle, Canvas, and Google Classroom can be challenging. It incorporates key steps for improving integration, such as API development, interoperability standards, plug-ins, and a user-friendly interface. Teachers can adopt the system as they become familiar with it and can add some advanced features. The system can be used in blended

learning environments and traditional teaching methods to personalize practice and feedback.

IX. Conclusion

In the digital age, programming skills have become a requisite for practice in almost every professional sphere, increasing the need for the most effective learning materials in programming study and training. Generating educational resources of computer programming based on ontology is a promising way to improve the quality and efficiency of educational resources of computer programming.

This study aims to develop a framework based on ontology to represent the Python programming concepts and relationships among them and implement a system for automatically generating the learning materials in the form of quizzes using the developed framework. In this study, we discuss the potential benefits and limitations of the current state of ontology-based automatic learning materials generation, specifically in programming languages. An ontology-based approach can potentially revolutionize the creation of tailored learning materials for programming education.

The system achieved a high accuracy rate of 98.5%, calculated using BERT-based semantic similarity, demonstrating its effectiveness in producing relevant and accurate learning materials. The novelty of this work lies in leveraging ontologies to automate quiz generation in programming education, offering a structured and scalable solution for personalized content creation.

Despite these contributions, certain limitations should be acknowledged. The study primarily focused on Python programming, which may impact the generalizability of the findings. Future research can address these limitations by implementing multi-programming language ontology. More research using controlled trials is needed. We recommend conducting a study comparing ontology-based learning materials to traditional, manually created materials using a controlled experiment. Students are divided into two groups: the control group receiving traditional materials and the experimental group receiving ontology-based materials. Post-tests measure retention, understanding, and satisfaction. Metrics include test scores, time to mastery, engagement time, and learning quality. The study would aim to assess the effectiveness of ontology-based learning materials in improving educational outcomes.

Acknowledgment

The authors gratefully acknowledge the financial assistance from the Institute of Information Science, Faculty of Mechanical Engineering and Informatics, University of Miskolc.

References

[1] S. MacNeil, Automatically Generating CS Learning Materials with Large Language Models, vol. 1, no. 1. Association for Computing Machinery, 2022.

[2] B. Abu-Salih and S. Alotaibi, “A systematic literature review of knowledge graph construction and application in education,” Heliyon, vol. 10, no. 3, p. 25383, 2024, doi: 10.1016/j.heliyon.2024.e25383.

[3] E. Rajabi and K. Etminani, “Knowledge-graph-based explainable AI: A systematic review,” J. Inf. Sci, 2022, doi: 10.1177/01655515221112844.

[4] M. Liu, Y. Ren, L. M. Nyagoga, F. Stonier, Z. Wu, and L. Yu, “Future of education in the era of generative artificial intelligence: Consensus among Chinese scholars on applications of ChatGPT in schools,” Futur. Educ. Res, vol. 1, no. 1, pp. 72-101, 2023, doi: 10.1002/fer3.10.

[5] W. Villegas-Ch and J. García-Ortiz, “Enhancing Learning Personalization in Educational Environments through Ontology-Based Knowledge Representation,” Computers, vol. 12, no. 10, 2023, doi: 10.3390/computers12100199.

[6] C. Diwan, S. Srinivasa, G. Suri, S. Agarwal, and P. Ram, “AI-based learning content generation and learning pathway augmentation to increase learner engagement,” Comput. Educ. Artif. Intell, vol. 4, no. February, p. 100110, 2022, doi: 10.1016/j.caeai.2022.100110.

[7] F. D. Calmon, R. Kokku, and A. Vempaty, “Automatic learning curriculum generation,” Google Patents, 2019.

[8] Z. Xia, Y. Zhou, F. Y. Yan, and J. Jiang, “Automatic curriculum generation for learning adaptation in networking.” 2022.

[9] J. Alshboul and E. Baksáné-Varga, A Survey of Domain Model Representations in Intelligent Tutoring Systems. Miskolc, Hungary: Faculty of Mechanical Engineering and Informatics University of Miskolc, 2021.

[10] J. Alshboul, H. A. A. Ghanim, and E. Baksa-Varga, “Semantic Modeling for Learning Materials in E-Tutor Systems,” Journal of Software Engineering and Intelligent Systems, vol. 6, no. 2, pp. 85–91, Aug. 2021.

[11] L. N. Nongkhai, J. Wang, and T. Mendori, “Developing An Ontology of Multiple Programming Languages from The Perspective of Computational Thinking Education,” in Proceeedings of the 19th International Conference on Cognition and Exploratory Learning in the Digital Age (CELDA 2022), Lisbon, Portugal: International Association for Development of the Information Society (IADIS), 2022, pp. 66–72. doi: 10.33965/celda2022_202207l009.

[12] W. Nie, K. Vita, and T. Masood, “An ontology for defining and characterizing demonstration environments,” J. Intell. Manuf, 2023, doi: 10.1007/s10845-023-02213-1.

[13] Q. U. Ain, M. A. Chatti, K. G. C. Bakar, S. Joarder, and R. Alatrash, “Automatic Construction of Educational Knowledge Graphs: A Word Embedding-Based Approach,” Inf, vol. 14, no. 10, 2023, doi: 10.3390/info14100526.

[14] J. Alshboul and E. Baksa-Varga, “A Hybrid Approach for Automatic Question Generation from Program Codes,” International Journal of Advanced Computer Science and Applications, vol. 15, no. 1, 2024, doi: 10.14569/IJACSA.2024.0150102.

[15] P. Brusilovsky, B. J. Ericson, C. Zilles, C. S. Horstmann, C. Servin, and F. Vahid, “The Future of Computing Education Materials,” Comput. Sci. Curricula, Curricula Pract, vol. 1, no. 1, pp. 1-8, 2023.

[16] J. Alshboul and E. Baksa-Varga, “A Review of Automatic Question Generation in Teaching Programming,” International Journal of Advanced Computer Science and Applications, vol. 13, no. 10, 2022, doi: 10.14569/IJACSA.2022.0131006.

[17] D. Vergara, M. L. Fernández, and M. Lorenzo, “Enhancing student motivation in secondary school mathematics courses: A methodological approach,” Educ. Sci, vol. 9, no. 2, 2019, doi: 10.3390/educsci9020083.

[18] L.-C. Lin, I.-C. Hung, Kinshuk, and N.-S. Chen, “The impact of student engagement on learning outcomes in a cyber-flipped course,” Educ. Technol. Res. Dev, vol. 67, pp. 1573-1591, 2019.

[19] N. A. Alrehaili, M. A. Aslam, D. H. Alahmadi, D. A. Alrehaili, M. Asif, and M. S. A. Malik, “Ontology-Based Smart System to Automate Higher Education Activities,” Complexity, vol. 2021, 2021, doi: 10.1155/2021/5588381.

[20] B. Flanagan, G. Akçapinar, R. Majumdar, and H. Ogata, “Automatic generation of contents models for digital learning materials,” in ICCE 2018 - 26th Int. Conf. Comput. Educ. Main Conf. Proc, 2018, pp. 804–806.

[21] K. Zhuang, “The Knowledge Graph Construction in the Educational Domain : Take an Australian School Science Course as an Example The Knowledge Graph Construction in the Educational Domain : Take an Australian School Science Course as an Example.” 2023.

[22] C. Pierrakeas, G. Solomou, and A. Kameas, "An ontology-based approach in learning programming languages," Proc, pp. 393-398, 2012, doi: 10.1109/PCi.2012.78.

[23] H. A. A. Ghanim, J. Alshboul, and L. Kovacs, "Development of Ontology-based Domain Knowledge Model for IT Domain in e-Tutor Systems," International Journal of Advanced Computer Science and Applications, vol. 13, no. 5, 2022, doi: 10.14569/IJACSA.2022.0130505.

[24] N. A. Anindyaputri, R. A. Yuana, and P. Hatta, "Enhancing Students' Ability in Learning Process of Programming Language using Adaptive Learning Systems: A Literature Review," Open Eng, vol. 10, no. 1, pp. 820-829, 2020, doi: 10.1515/eng-2020-0092.

[25] T. Guber, "A translational approach to portable ontologies," Knowl. Acquis, vol. 5, no. 2, pp. 199-229, 1993.

[26] K. Chen, Q. Huang, H. Palangi, P. Smolensky, K. Forbus, and J. Gao, "Mapping natural-language problems to formal-language solutions using structured neural representations," in International Conference on Machine Learning, 2020, pp. 1566–1575.

[27] F. Baader, I. Horrocks, C. Lutz, and U. Sattler, Introduction to description logic. Cambridge University Press, 2017.

[28] V. Lama, A. Patel, N. C. Debnath, and S. Jain, "IRI_Debug: An Ontology Evaluation Tool," New Gener. Comput, vol. 42, no. 1, pp. 177-197, 2024, doi: 10.1007/s00354-024-00246-5.

[29] A. Ramírez-Noriega, "Towards the Automatic Construction of an Intelligent Tutoring System: Domain Module," Adv. Intell. Syst. Comput, vol. 930, no. 3, pp. 293-302, 2019, doi: 10.1007/978-3-030-16181-1_28.

[30] N. C. Debnath and A. Patel, "Ontology Evaluation Tools: Current and Future Research," Recent Adv. Comput. Sci. Commun, 2022, [Online]. Available: https://api.semanticscholar.org/CorpusID:248138690.

[31] W. Yathongchai, J. Angskun, and C. C. Fung, "An Ontology Model for Developing a SQL Personalized Intelligent Tutoring System," Naresuan Univ. J. Sci. Technol, vol. 25, no. 4, pp. 88-96, 2017.

[32] A. Fernández-Izquierdo and R. García-Castro, "Themis: A tool for validating ontologies through requirements," in Proc. Int. Conf. Softw. Eng. Knowl. Eng. SEKE, 2019, pp. 573-578,.

[33] M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez, "Validating Ontologies with OOPS ! State of the Art," Knowl. Eng. Knowl. Manag, pp. 267-281, 2012.

[34] "Ontology Generation and Ontology Data Set." Accessed: Apr. 24, 2025. [Online]. Available: https://github.com/jalshboul/Python-Ontology-GLM