An Architecture for Generating Questions, Answers, and Feedback from Ontologies

Toky Raboanary[0000−0001−6133−4643] and C. Maria Keet[0000−0002−8281−0853]

Department of Computer Science, University of Cape Town, South Africa
traboanary,mkeet@cs.uct.ac.za

Abstract. Automatically generating questions, answers, and feedback from ontologies and conceptual models is crucial for learning activities and knowledge validation. Existing proposals are limited to predefined types of questions and the modelling style that they are tailored to, lack feedback generation, and their core algorithm are dependent on those characteristics, therewith hampering maintainability and reusability. We designed a new architecture where the question, answer and feedback specifications, the core algorithm for selecting the contents from the ontology, and the verbaliser are modularised for resolving these problems. We instantiated the architecture as a proof-of-concept, examined three test cases, and showed that it compares favourably to related work.

Keywords: Ontology-based Question Generation · Ontology-based Feedback Generation · Architecture · Natural Language Generation · Template-based Question Generation · Ontologies for Education.

1 Introduction

Natural Language Generation (NLG) is increasingly being used for daily consumption of information; e.g., the automatic generation of weather forecasts and reports (e.g., [17]) and generating articles about soccer matches. NLG also provides benefits for educational technologies, such as automating question generation and marking for computer-assisted language learning exercises [6, 10], generating quizzes from DBpedia [18] and improving student learning in the Biology domain [4, 3] by means of 'intelligent textbooks'. We focus on generating questions with their answers and feedback from structured knowledge, namely ontologies and conceptual models, i.e., from the TBox (terminological knowledge) rather than ABox (assertional knowledge). These systems are developed for different purposes, including education [4, 3, 2, 19, 11, 16, 20] and ontology (knowledge base) validation [1, 13], and they offer different types of questions, being mainly Multiple Choice Questions (MCQs), but also similarity, yes/no, and short answer questions. They generally focus on controlling MCQ question difficulty and generating distractors [2, 19, 11]. Nevertheless, some look at the verbalisation of the questions (rendering the questions in a natural language) and the feedback generation (e.g., [11]).

Existing approaches are based on predefined types of questions and tested or/and evaluated with one or few ontologies, not considering the different modelling styles (e.g., [19, 11, 1]). Then, it is still being determined how they can

be reused for other types of questions (if possible). In addition, their axiom prerequisites for generating a type of question, i.e., the prerequisites that an ontology should satisfy for generating a particular type of question, are not clear. A partial disentanglement was proposed by [16], where question preconditions were specified that further provide control on question generation, but those axiom prerequisites and their mappings with the templates[1] are embedded in the core algorithm, which limits the maintainability, extensibility, reusability and generalisability of the approach.

Also, in the educational environment, providing feedback to learners is crucial. Leo et al. [11] developed an ontology-based multi-term MCQs question for the medical domain. They provided an explanation for the correct and incorrect options as feedback. However, they only consider four fixed types of questions and only use one ontology. Hence, it may not support other ontologies with different modelling styles, naming schemes and types of questions.

To the best of our knowledge, existing proposals are limited to predefined types of questions and the modelling style that they are tailored to, and lack feedback generation. To address these problems, we aim to develop a new architecture for generating questions, answers and feedback so that it can flexibly support different types of questions with their axiom prerequisites and different modelling styles, deal with complex answer types and provide feedback explaining the answer of the question, while using a unique core algorithm.

The principal concept of the designed architecture is to modularise the question, answer and feedback specifications, the core algorithm for selecting the contents from the ontology, and the verbaliser to produce the sentences. These specifications are structured in so-called *question card* with manageable subparts. Our results show that 1) one can generate questions, answers and feedback from the architecture by only defining the specifications such that the whole architecture does not have to be modified for more types of questions or considering more modelling styles; and 2) our analysis shows that this new architecture is better than existing studies in terms of flexibility and functionality. The instances of the question card, algorithms, source code and results are available at `https://github.com/mkeet/AQuestGO`.

The remainder of this paper is structured as follows: Section 2 presents the architecture for the question, answer and feedback generation, Section 3 illustrates the proof-of-concept implementation, presents test cases and compares it to related work, and we conclude in Section 4.

2 Architecture

The description of the architecture, as depicted in Fig. 1, is divided into four categories: the inputs, the content selection algorithm (CSA), the verbaliser and the outputs for generating the questions, answers and feedback from an ontology.

We present the different components and the decision choices for obtaining a flexible architecture.

[1] A template is a linguistic structure containing slots, which are intended to be replaced by relevant words to construct a sentence.

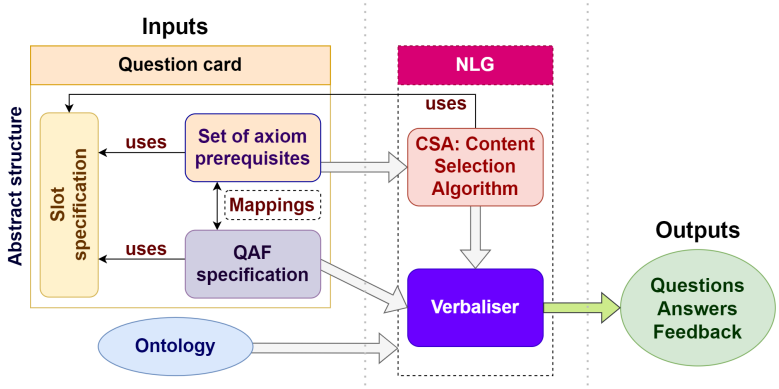

Fig. 1: The proposed architecture (QAF= Questions, Answers and Feedback)

2.1 Inputs

There are two types of input: the ontology and so-called *question card*, the latter of which can be subdivided into the abstract content for the ontology, slot and QAF (Questions, Answers and Feedback) specification. The architecture requires a well-formed instance of the question card and a consistent ontology.

Ontology or conceptual model. This is the structured knowledge to be fed into the NLG process without changing its structure and contents.

Question card. The question card, which is presented in Fig. 1 and Fig. 2, comprises three main elements for a given type of question, such as 'yes/no question containing 2 classes and one property': the slot specification, the set of axiom prerequisites and the QAF specification. Essentially, the latter concerns the 'linguistic templates' formalising the questions, answers and feedback (be they basic or grammar-infused templates [12]). The slot specification describes the different slots intended to be used in the axiom prerequisites and the linguistic templates. The set of axiom prerequisites is the set of preconditions that need to be satisfied by the ontology to be able to generate questions for that type of question. The linguistic templates are the linguistic structures to be filled in from the selected ontology vocabulary elements to generate the sentences. Listing 1.1 presents an example of an instance of the question card, which also illustrates in lines 11 and 12 that it can deal with different modelling styles.

Listing 1.1: Example of a *qCard1* question card instance that deals with two different modelling styles, represented in an abstract specification (see online supplementary material for the equivalent XML representation). C and OP are the sets of classes and object properties (OPs), respectively.

1 **question card name**: qCard1 *(a question card instance)*
2 **type of question**: Which X prop Y?
3 **begin** {slot specification}

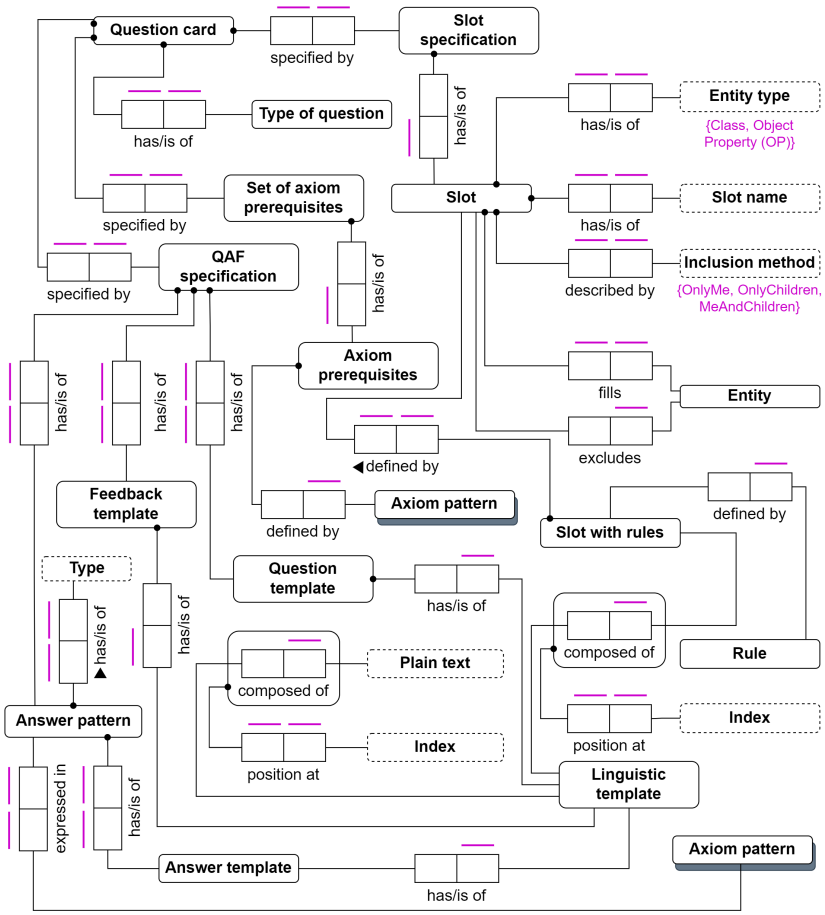

Fig. 2: An ORM model of the components of 'Question card' with the other entities

slot (1, 2, 3): '[X]', '[Y]' and '[Z]' representing a class in $\mathsf{C}_1 = \{\forall \mathsf{c} \in \mathsf{C} : \mathsf{c} \sqsubset \mathsf{Thing}\} - \{\mathsf{SocialObject}, \mathsf{PhysicalObject}\}$

slot (4): '[Soc]' representing the class SocialObject

slot (5): '[Phy]' representing the class PhysicalObject

slot (6): '[inheresIn]' representing the OP inheresIn

slot (7): '[prop]' representing an OP in $\mathsf{OP}_1 = \{\forall \mathsf{p} \in \mathsf{OP} : \mathsf{p} \sqsubset \mathsf{TopObjectProperty}\}$

end {slot specification}

begin {set of axiom prerequisites}

axiom prerequisites (1) *(first modelling style)*: (a) [X] $\sqsubseteq$ [Phy], (b) [Y] $\sqsubseteq$ [Soc], (c) [Phy]$\neg$[Soc], (d) [Soc] $\sqsubseteq \exists$[inheresIn].[Phy], (e) [Y] $\sqsubseteq \exists$[inheresIn].[X], (f) [Y] $\sqsubseteq \exists$[prop].[Z], (g) [X]$\neg$[Y]

axiom prerequisites (2) *(second modelling style)*: (a) [Y] $\sqsubseteq$ [X], (b) [Y] $\sqsubseteq \exists$[prop].[Z]

end {set of axiom prerequisites}

begin {QAF specification}

begin {question template (qt)}

qt(1): What type of [X/noArticle] [prop/*OP_Verb-third] [Z]? *('noArticle' and 'OP_Verb-third' mean that the class X should be written without any article,*

and the template qt(1) can only be used if the OP prop *is classified as a verb (see [16]) and will be conjugated in the third person, respectively.)*

qt(2): What type of [X/noArticle] [prop/*OP_VerbPrep-third] [Z]?
qt(3): What type of [X/noArticle] [prop/*OP_HasNouns-third] that is [Z]?
qt(4): What type of [X/noArticle] is [prop/*OP_IsNounsPrep] [Z]?
qt(5): What type of [X/noArticle] is [prop/*OP_IsPastParticipleBy] [Z]?
qt(6): What type of [X/noArticle] is [prop/*OP_IsPastParticiplePrep] [Z]?
end {question template (qt)}
begin {answer pattern}
answer template: The answer is [Y].
axiom pattern: [Y]
end {answer pattern}
feedback template: The answer is [Y] since it is a [X] that [prop] [Z].
end {QAF specification}

Slot specification. The slot specification (e.g., lines 3-9 in Listing 1.1) permits defining and restricting the tokens' scope. A token can be either a class or an object property (OP). This may be a single class or OP, only subclasses or sub-OPs thereof or both (Inclusion method in Fig. 2). Also, it allows the exclusion of classes or OPs that are unfavourable for the generation of questions (e.g., in Listing 1.1, line 4). Leo et al. [11] also considered this exclusion process, but the flexibility of the axiom prerequisites and the linguistic templates were missing.

Set of axiom prerequisites. A type of question has a set of axiom prerequisites to be able to deal with the fact that there are different ways to represent the same piece of knowledge. As an illustration, one may adopt a top-down ontology development and reuse a foundational ontology, or a bottom-up approach. Following that, there may be different axiom patterns that are similar [5] and they would be verbalised in the same way. For instance, one may represent 'A student is a person.' with a subsumption relation or represent 'student' as a role that a person plays (see [5]). Thereby, we designed the question card to support several axiom prerequisites for a given type of question. In our example in Listing 1.1, the two axiom prerequisites in lines 11 and 12 are defined in the instance *qCard1* for dealing with the two different modelling styles: a role representation and a subsumption relation representation, respectively.

QAF specification. (e.g., lines 14-28 in Listing 1.1) A type of question has sets of questions, answers and feedback templates. More than one linguistic template is required since the verbalisation of the axiom patterns may differ depending on the specifications of the entities involved in the sentence generation, e.g., Part-Of-Speech (POS) tagging and the Object Property (OP) naming scheme. For instance, the adequate template from the axiom $\mathsf{Book} \sqsubseteq \exists \mathsf{isBorrowed\text{-}by}.\mathsf{Student}$ is "*Is* [X : Noun][P : OP_Is_Past_Part_By] [Y : Noun]*?*"[2], with X and Y are classes and P is an OP, and a valid question is "Is a book borrowed by a student?" rather

[2] In this example, Noun means that the POS of a class (X, Y) is a noun. [OP_Is_Past_Part_By] indicates that the OP is composed of 'is', a verb in a past participle form and 'by' (i.e., passive voice).

than "Does a book borrowed by a student?". Thereby, we design the question card to accommodate several linguistic templates so that the verbalisation approach may implement a strategy to select the appropriate one.

One may combine ontology element-based and natural language-driven templates (strategy chosen in Listing 1.1) by constraining the entities in the slot specification and the axiom prerequisites definitions (e.g., $\mathsf{X} \sqsubseteq \mathsf{Document}$ for, say, an ontology about libraries). Furthermore, for grammar-infused templates [12] for languages with complex grammars, one can go a step further to reusability and interoperability with the task ontology ToCT for declaring templates [14]. Either way, the ontology elements must be mentioned in the linguistic templates for the mappings between them and the axiom prerequisites.

The axiom pattern is also considered in the answer pattern of the question card to ease the automatic marking of the responses provided by learners in an educational setting. In addition, this could be useful for the generation of MCQs.

Mappings between the axiom prerequisites and linguistic templates. The mappings connect what is stated in the axiom prerequisites and stated in the linguistic templates for generating the sentences. The two above structures share the same slot names defined in the slop specification, which serve as mappings. Thereby, the place of the slot only depends on the definition of the linguistic template. For instance, in Listing 1.1 line 16, the slot [X] in the linguistic template represents a class that is defined in the axiom prerequisites in line 11 and the slot specification in line 4.

Linguistic template used in the question card. The question card is abstract. Its principle is that one may use it for different types of linguistic templates, such as 'grammar-infused template' [12], since the verbaliser is not embedded in the main algorithm (CSA). In this study, the chosen linguistic template is accommodated to the verbalisation approach of AQuestGO [16] (We shall discuss the choice of this question generation approach in Subsection 2.3.). Thereby, we select ontology element-based and natural language-driven templates as linguistic templates, and the rest of the paper is based on that.

2.2 Content Selection Algorithm

The algorithm finds all instances satisfying the abstract specifications in a given instance of the question card and an ontology, and the results are used for the verbalisation. The first step is to parse the instance of the question card. Then, the algorithm processes it with the ontology to get the valid axioms for the generation of the questions, the answers and feedback based on the specifications in the considered instance of the question card, namely, the set of axiom prerequisites and the slot specification. After reducing the research space from the latter, it recursively searches all semantically and logically valid forms of axioms based on the definitions of the axiom prerequisites. Then, CSA associates each selected ontology vocabulary element with the specified slots. Thereafter, it outputs the lists of pairs of slots and ontology elements to generate the questions,

answers and feedback. As an illustration, by using the instance of the question card in Listing 1.1 and the African Wildlife Ontology (AWO) that we adapted in earlier work [16], one can obtain the following output: {([Y], CarnivosoursPlant), ([X], Plant), ([Z], Animal), ([prop], eats)}, satisfying the second axiom prerequisites in the instance of the question card. And the explanation of the selection is the following axioms: CarnivorousPlant $\sqsubseteq \exists$eats.Animal, CarnivorousPlant $\sqsubseteq$ Plant.

One may define different axiom prerequisites to express different modelling styles in a unique or different instance(s) of the question card. Either way, the algorithm just needs to receive the necessary slot specification the adequate axiom prerequisites.

The algorithm and the source code can be found in supplementary materials.

2.3 Verbaliser

Apart from the linguistic templates in the question card and the ontology, the verbaliser only receives the slot and ontology element pairs selected by CSA. Thereby, one may use any ontology verbaliser here. Practically for the realisation of the architecture, we will reuse and extend the one of [16] because it 1) explicitly incorporates axiom prerequisites for generating questions from the TBox of ontologies, 2) has shown to generate good questions with respect to the syntactic and semantic quality thereof, and 3) it is open source. Their approaches only work with natural language-driven or ontology element-based templates. Our verbaliser can combine them because of the mappings, offering more flexibility.

2.4 Outputs

There are three main categories of outputs: the questions, answers and feedback in natural language. Moreover, the explanation of the results is also provided by CSA. The explanation is the concrete instance of valid axioms satisfying the axiom prerequisites for a given ontology. For instance, with AWO, the axioms: CarnivorousPlant $\sqsubseteq \exists$eats.Animal, CarnivorousPlant $\sqsubseteq$ Plant explains the generation of the results from the axiom prerequisites: a) [Y] $\sqsubseteq$ [X], (b) [Y] $\sqsubseteq \exists$[prop].[Z] in Listing 1.1 line 12. The answers can also be axioms, a class or an OP, and/or Boolean format, and/or sets of pairs of slot and ontology elements.

There are different outputs so that one can exploit the results depending on the objectives. Thereby, it offers flexibility for the question, answer and feedback generation approaches.

3 Implementation and Evaluation

For validating the feasibility of the architecture, a particular instantiation was implemented. This was then tested on three examples to verify that it can indeed generate questions, answers and feedback. We will report on this first. Second, we compare the architecture to other systems and architectures, demonstrating its more comprehensive functionality and flexibility.

3.1 Implementation and Test Cases

We briefly present the implementation of the architecture. We choose the XML language to represent instances of the question card for reusability and interoperability. We implemented the question, answer and feedback generation with JAVA since we make use of OWL API [9] for manipulating the ontology and HermiT reasoner [8] for implementing CSA. The syntax of the axiom prerequisites is based on Manchester syntax; slots are used instead of ontology elements. Regarding the verbalisation, WordNet [15] was used for basic POS tagging, and SimpleNLG [7] resolved agreement of subject and verb, gerund form generation and article checking.

The test cases were carefully selected to show that one can generate results and deal with i) complex types of questions, ii) different modelling styles and iii) the types of questions that AQuestGO is designed for [16] by adopting the architecture. An extended version of the AWO is used for the first test, which was also used by [16], and two test ontologies for the second test (see supplementary materials), containing the following:

1. $ontology_a$, with: $\mathsf{Person} \sqsubseteq \mathsf{PhysicalObject}$, $\mathsf{Student} \sqsubseteq \mathsf{SocialObject}$, $\mathsf{SocialObject} \sqsubseteq \exists \mathsf{inheresIn.PhysicalObject}$, $\mathsf{Student} \sqsubseteq \exists \mathsf{inheresIn.Person}$, $\mathsf{PhysicalObject} \sqsubseteq \neg \mathsf{SocialObject}$ and $\mathsf{Student} \sqsubseteq \exists \mathsf{borrows.Book}$ (a subsumption relation representation); and
2. $ontology_b$, with: $\mathsf{Student} \sqsubseteq \mathsf{Person}$ and $\mathsf{Book} \sqsubseteq \mathsf{Document}$ (a role representation), where $ontology_b$ is semantically similar to $ontology_a$.

Complex types of questions. We chose AWO and *qCard2* in Listing 1.2 for the first test case, where two different questions (lines 18 and 19) were defined for the same answer. And all questions that can be asked for the same answer will be generated if the axiom prerequisites in line 14 are satisfied. One can remark that the two different linguistic templates can use different slots.

Listing 1.2: Example of a *qCard2* question card instance that deals with a complex type of question, represented in an abstract specification (see online supplementary material for the equivalent XML representation). C and OP are the sets of classes and object properties (OPs), respectively.

1 **question card name**: qCard2 *(a question card instance)*
2 **type of question**: Which X prop Y?
3 **begin** {slot specification}
4 **slot (1, 2, 3, 4, 5)**: '[X]', '[Y]', '[Z]', '[W]' and '[S]' representing a class in $\mathsf{C}_1 = \{\forall \mathsf{c} \in \mathsf{C} : \mathsf{c} \sqsubseteq \mathsf{Animal}\}$
5 **slot (6, 7)**: '[P1]' and '[P3]' representing a class in $\mathsf{C}_2 = \{\forall \mathsf{c} \in \mathsf{C} : \mathsf{c} \sqsubseteq \mathsf{PlantParts}\}$
6 **slot (8, 9, 10)**: '[P2]', '[P4]' and '[P5]' representing a class in $\mathsf{C}_3 = \{\forall \mathsf{c} \in \mathsf{C} : \mathsf{c} \sqsubseteq \mathsf{Plant}\}$
7 **slot (11, 12, 13, 14)**: '[prop1]', '[prop2]', '[prop3]' and '[prop4]' representing an OP in $\mathsf{OP}_1 = \{\forall \mathsf{p} \in \mathsf{OP} : \mathsf{p} \sqsubseteq \mathsf{eats}\}$
8 **slot (15, 16)**: '[prop5]' and '[prop6]' representing an OP in $\mathsf{OP}_2 = \{\forall \mathsf{p} \in \mathsf{OP} : \mathsf{p} \sqsubseteq \mathsf{part\text{-}of}\}$

9 **slot (17)**: '[prop8]' representing the OP live-on
10 **slot (18)**: '[Omni]' representing the class Omnivore
11 **slot (19)**: '[L]' representing the class Land
12 **end** {slot specification}
13 **begin** {set of axiom prerequisites}
14 **axiom prerequisites (1)**: (a) [X] $\sqsubseteq$ [Z], (b) [X] $\sqsubseteq$ [Omni], (c) [S]$\neg$[Omni], (d) [X] $\sqsubseteq \forall$[prop8].[L], (e) [X] $\sqsubseteq \exists$[prop1].[Z] $\sqcap \exists$[prop2].[P1] $\sqcap \exists$[prop3].[P2] $\sqcap \exists$[prop3].[P3]
15 **end** {set of axiom prerequisites}
16 **begin** {QAF specification}
17 **begin** {question template (qt)}
18 **qt(1)**: What [Z/noArticle], which is not [S] and [prop8/third] [L], [prop1/third] some [Y] and the following plants or parts of plants: [P1], [P2] and [P3]?
19 **qt(2)**: What is the mysterious object? It is [Z]. It is not [S] and [prop8/third] [L]. It is [Omni], and [prop1/third] some [Y] and the following plants or parts of plants: [P1], [P2] and [P3].
20 **end** {question template (qt)}
21 **begin** {answer pattern}
22 **answer template**: The answer is [X].
23 **axiom pattern**: [X]
24 **end** {answer pattern}
25 **feedback template**: The answer is [X/noArticle], since [X] is [Omni], which means that [X] can eat animals and plants.
26 **end** {QAF specification}

The instantiation of the architecture can generate, for example, the two following questions 1) "What animal, which is not a carnivore and lives on land, eats some animal and the following plants or parts of plants: a root, grass and a fruiting body?" (Linguistic template in Listing 1.2, line 18) and 2) "What is the mysterious object? It is an animal. It is not a herbivore and lives on land. It is an omnivore, and eats some animal and the following plants or parts of plants: a fruiting body, grass and a root." (Linguistic template in Listing 1.2, line 19). It can then generate the following answer: "The answer is warthog." (Linguistic template in Listing 1.2, line 22) and feedback: "The answer is warthog, since a warthog is an omnivore, which means that a warthog can eat animals and plants." (Linguistic template in Listing 1.2, line 25). The axiom-based explanation can also be provided as discussed previously in Subsection 2.4.

Different modelling styles. We used $ontology_a$ and $ontology_b$ and the question card presented in Listing 1.1 to show that the architecture can deal with different modelling styles. When using $ontology_a$, the first axiom prerequisites (Listing 1.1, line 11) are selected by CSA since they are conformed to the content of the ontology. And the verbaliser generates the question: "What type of person borrows a book?" (Linguistic template in Listing 1.2, line 16) and the answer: "The answer is a student.". The generated axiom-based explanation is: Person $\sqsubseteq \neg$Student, Student $\sqsubseteq \exists$borrows.Book, Student $\sqsubseteq \exists$inheresIn.Person, Document $\sqsubseteq$ PhysicalObject, SocialObject $\sqsubseteq \exists$inheresIn.PhysicalObject, PhysicalObject $\sqsubseteq \neg$SocialObject, and Student $\sqsubseteq$ SocialObject. And with the same instance of the question card and $ontology_b$ as input, CSA selects the second

Table 1: Comparison of the approach to existing studies; predef. = predefined; TQ= type of question, TA= type of answer, SA= short answer, CA= complex answer, FC= feedback on correct answer, FI= feedback on incorrect answer, DF= dynamic feedback, MS= considering modelling styles, AP= considering axiom prerequisites, RV= replaceable verbaliser and LR= language restriction.

Criteria	[1]	[16]	[13]	[19]	[4]	[11]	**our architecture**
TQ	predef.	predef.	predef.	predef.	predef.	predef.	**flexible**
TA	predef.	predef.	predef.	predef.	predef.	predef.	**flexible**
MCQs	**yes**	no	no	**yes**	**yes**	**yes**	no
Yes/No	**yes**	**yes**	**yes**	no	**yes**	no	**yes**
SA	**yes**	**yes**	**yes**	no	**yes**	no	**yes**
CA	no	no	no	no	no	no	**yes**
FC	no	no	no	no	no	**yes**	**yes**
FI	no	no	no	no	no	**yes**	no
DF	no	no	no	no	no	no	**yes**
MS	no	no	no	no	no	no	**yes**
AP	no	**yes**	no	no	no	no	**yes**
RV	no	no	no	no	no	no	**yes**
LR	less than $\mathcal{ALC}$	$\mathcal{ALC}$	n/c	$\mathcal{SHIQ}$	n/c	n/c	**none**

axiom prerequisites for generating the same results with a different explanation: $\mathsf{Student} \sqsubseteq \exists \mathsf{borrows.Book}$ and $\mathsf{Student} \sqsubseteq \mathsf{Person}$.

Types of questions defined in AQuestGO. Raboanary et al. [16] investigated 10 types of educational useful questions. The instances of the question card are able to represent all these types of questions, and we did the generation by using AWO. As an illustration, from the type of question: yes/no with one class and one property, one can generate the question: "Does a bumble bee fly?", the answer: 'yes' and the explanation: $\mathsf{BumbleBee} \sqsubseteq \exists \mathsf{Participate\text{-}In.Fly}$, with $\mathsf{BumbleBee} \sqsubseteq \mathsf{Endurant}$ and $\mathsf{Fly} \sqsubseteq \mathsf{Perdurant}$. All results are available online.

3.2 Comparing the new Architecture to the Existing Studies

We compare and discuss the new architecture with the relevant existing studies [11, 16, 1, 13, 19, 4] considering the following criteria: the dynamic aspect of the types of questions and answers, the generation of feedback, the consideration of the modelling styles and the axiom prerequisites, the replaceability of the verbaliser and the language restriction. The comparison is presented in Table 1.

Existing studies only deal with predefined types of questions. Their architecture is rigid and cannot support flexibility. This limits their approach's maintainability, extensibility, reusability and generalisability. The newly designed approach does not suffer this problem since all specifications are defined externally in instances of the question card, which offers flexibility. The types of answers are flexible in our architecture, whereas it is not the case for the others. Our

architecture can specify the answer as a very complex axiom or a very long text, while [1, 13, 16] are only limited to yes/no and short answer questions, for instance. Leo et al. [11] consider feedback generation. However, since their types of questions are predefined, they are not dynamic. They took into account generating feedback for incorrect answers, a feature our architecture does not have, and it does not generate the distractors of MCQs. However, one may use Alsubait's findings [2] to adapt our study for MCQs generations, for instance. Then, only our architecture considers dealing with different modelling styles, and apart from our study, only [16] considers the axiom prerequisites that need to be in an ontology for generating a particular type of question. Also, our approach does not have language restrictions when expressing the axiom prerequisites.

Further, observe that other verbalisers with different verbalisation techniques can be attached to the architecture since the output of the content selection algorithm is a list of pairs of token and ontology elements. It is also amenable to generating questions for other languages, such as French or Malagasy, provided the ontology has such vocabulary and the templates are in said language.

4 Conclusion

We designed a new architecture for generating questions, answers and feedback from ontologies that is able to deal with non-predefined types of questions and different modelling styles. The test cases from an instantiation of the architecture show that the new architecture offers flexibility since all processes can be done by only instantiating the question card externally for defining the necessary specifications for generating the questions, answers and feedback. Further, our analysis shows that our study compares favourably to existing proposals in terms of flexibility and functionality. This architecture is a step to generalising the generation of questions, answers and feedback from ontologies. As future work, we plan to automatically create question cards to reduce manual efforts as well as human-in-the-loop control mechanisms.

Acknowledgements TR acknowledges support from the Hasso Plattner Institute for Digital Engineering through the HPI Research School at UCT.

References

1. Abacha, A.B., Dos Reis, J.C., Mrabet, Y., Pruski, C., Da Silveira, M.: Towards natural language question generation for the validation of ontologies and mappings. Journal of Biomedical Semantics **7**(1), 1–15 (2016)
2. Alsubait, T.: Ontology-based multiple-choice question generation. Ph.D. thesis, University of Manchester, Manchester, England (2015)
3. Chaudhri, V.K., Clark, P.E., Overholtzer, A., Spaulding, A.: Question generation from a knowledge base. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2014. LNAI, vol. 8876, pp. 54–65. Springer, Linköping, Sweden (2014)

4. Chaudhri, V., Cheng, B., Overholtzer, A., Roschelle, J., Spaulding, A., Clark, P., Greaves, M., Gunning, D.: Inquire Biology: A textbook that answers questions. AI Magazine **34**(3), 55–72 (2013)
5. Fillottrani, P.R., Keet, C.M.: Patterns for heterogeneous TBox mappings to bridge different modelling decisions. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) Proceedings (Part I) of the 14th European Semantic Web Conference, ESWC 2017. LNCS, vol. 10249, pp. 371–386. Springer, Portorož, Slovenia (Jun 2017)
6. Gardent, C., Perez-Beltrachini, L.: Using FB-LTAG Derivation Trees to Generate Transformation-Based Grammar Exercices. In: Proc. of TAG+11. pp. 117–125. ACL (2011), sep 2012, Paris, France
7. Gatt, A., Reiter, E.: SimpleNLG: A realisation engine for practical applications. In: Krahmer, E., Theune, M. (eds.) Proceedings of the 12th European Workshop on Natural Language Generation, ENLG 2009. pp. 90–93. Association for Computational Linguistics (ACL), Athens, Greece (Mar 2009)
8. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 reasoner. Journal of Automated Reasoning **53**(3), 245–269 (2014)
9. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. Semantic Web **2**(1), 11–21 (2011)
10. Lange, H., Ljunglöf, P.: Putting control into language learning. In: Davis, B., Keet, C.M., Wyner, A. (eds.) Proceedings of Controlled Natural Language Workshop (CNL'18). FAIA, vol. 304, pp. 61–70. IOS Press (2018)
11. Leo, J., Kurdi, G., Matentzoglu, N., Parsia, B., Sattler, U., Forge, S., Donato, G., Dowling, W.: Ontology-based generation of medical, multi-term MCQs. International Journal of Artificial Intelligence in Education **29**(2), 145–188 (2019)
12. Mahlaza, Z., Keet, C.M.: A classification of grammar-infused templates for ontology and model verbalisation. In: Garoufallou, E., Fallucchi, F., Luca, E.W.D. (eds.) Proc. of MTSR'19. CCIS, vol. 1057, pp. 64–76. Springer, Rome, Italy (2019)
13. Mahlaza, Z., Keet, C.M.: OWLSIZ: An isiZulu CNL for structured knowledge validation. In: Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+). pp. 15–25 (2020)
14. Mahlaza, Z., Keet, C.M.: ToCT: A Task Ontology to manage complex templates. In: Proceedings of the 12th International Conference on Formal Ontology in Information Systems, FOIS 2021. CEUR, Bolzano, Italy (2021)
15. Miller, G.A.: WordNet: A lexical database for English. Communications of the ACM **38**(11), 39–41 (1995)
16. Raboanary, T., Wang, S., Keet, C.M.: Generating answerable questions from ontologies for educational exercises. In: Garoufallou, E., Ovalle-Perandones, M.A., Vlachidis, A. (eds.) Proc of MTSR'21. CCIS, vol. 1537, pp. 28–40. Springer (2022)
17. Ramos-Soto, A., Bugarin, A.J., Barro, S., Taboada, J.: Linguistic descriptions for automatic generation of textual short-term weather forecasts on real prediction data. IEEE Transactions on Fuzzy Systems **23**(1), 44–57 (2014)
18. Rodríguez Rocha, O., Faron Zucker, C.: Automatic generation of quizzes from DBpedia according to educational standards. In: Lahoud, I., Cardoso, E., Matta, N. (eds.) Proceedings fo the 3rd Educational Knowledge Management Workshop, EKM 2018. pp. 1035–1041. Lyon, France (Apr 2018), april 23 - 27, 2018
19. Venugopal, V.E., Kumar, P.S.: A novel approach to generate MCQs from domain ontology: Considering DL semantics and open-world assumption. Journal of Web Semantics **34**, 40–54 (2015)
20. Venugopal, V.E., Kumar, P.S.: Automated generation of assessment tests from domain ontologies. Semantic Web **8**(6), 1023–1047 (2017)