Generation of Quizzes and Solutions based on Ontologies - a Case for a Music Problem Generator

***Abstract*—Generation of problem sets and quizzes forms an important part of education technology. Although some systems have been built for quiz generation, they mostly focus on abstract logic and mathematical constructs. Knowledge in other domains is relational rather than propositional, and many systems use dedicated knowledge databases. We present a method to represent this knowledge as objective question sets and drills. We implement a four-fold approach - Ontologies, Propositional logic, similarity finding and hierarchical knowledge - as a way to generate quizzes as well as to solve human generated problems of a similar nature. The architecture of this system is such that any knowledge base can be turned into a vast number of diverse, complex questions. We build this system for an ontology of North Indian Ragas, and also test it on a smaller ontology of animals. We demonstrate an approach to solving human generated questions from the Ontology, and provide a possible sample space of questions that can be generated through this method.**

I. Introduction

In this paper, we propose a four-fold approach towards generation of diverse quizzes and solutions, to be formulated from knowledge ontologies. The use of ontologies has become more and more common in educational tools and resources, where semantic links and methods of linked data representation are making educational tools more and more dependent on structured data.

Quiz generation seems a natural next step towards using structured knowledge databases into technologies for education. We propose the addition of hierarchies from the knowledge system and other relational schema to enhance the questions generated. This combination of features is to be used with propositional logic inference schemas to generate a richer, diverse repository of possible questions. Knowledge relations prune the sample space of randomized questions and answers that can be formed through logical relations alone. Knowledge relations and similarity finding also generate cognitively salient questions, resulting in a usable case scenario.

In this paper, we implement this system for an ontology for North Indian Classical Music as a part of the Virtual Music Labs project [11], but the methods could be applied to ontologies of various subjects to diversify the scope of system-generated practice questions. We apply the same system on a zoo animals ontology. We implement a forward parsing approach to solve human generated questions of a similar nature and finally report the increment in the possible types of questions upon the addition of each module of the system.

A. Related Work

Constraint-based-models (CBMs) have often been used for problem set generation which create and solve problem sets from databases [1] [2]. [2] uses a path analysis, predicate and constraint generation model to arrive at a possible problem set. [3] also describes in detail, a constraint satisfaction model to generate test data, although this isn't directly applicable to students as users. [4] deals with solving word problems using programs that understand the logic structures embedded in the question. A similar approach is taken for other approaches to solve natural language math questions [5] in mathematics. [6] and [9] mention some systems for generating language learning questions from ontologies using complex syntax and CBMs.

In most of the work related to the propositional logic approach towards quiz solving and generation, it is not clear how it can be applied to real world knowledge and data. The systems are tested on proof graph ontologies in some cases, which deal with abstract chains of proposition interpretation and conclusion, which, although may be suitable for programming concepts, are not suitable for real world knowledge systems.

For example, table I represents some examples taken from the methods used in [4]. When we try to apply this abstract logic method to real questions, we come up with impossible, and sometimes incorrect scenarios for the ontology of music. We therefore require a system that can more easily intuit the knowledge hierarchies in the music database itself. We create this using hierarchies embedded in the ontologies, and implementing relational constraints.

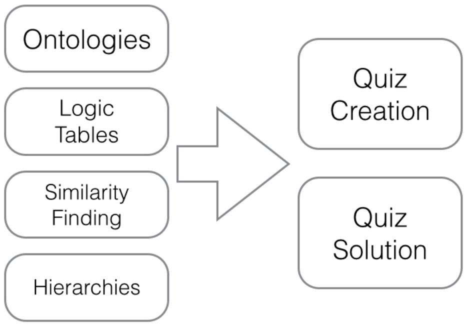

Fig. 1. Architecture

II. Our Approach

We intend this system for creating more and more solvable examples for students of various disciplines. Here, we choose two ontologies to demonstrate this application. The first one is a musical ontology containing data for constituents of several ragas in North Indian music. The second is a database and details of several animals found in a zoo, with descriptions of taxonomy, habitat, food etc. Using these two databases, we are able to create quiz questions that involve several types of

TABLE I. PREMISES AND CONCLUSIONS

	Premise 1	Premise 2	Premise 3	Conclusion
Abstraction	$x1 \equiv x2$	$x3 \rightarrow x2$	$(x4 \rightarrow x5) \rightarrow x3$	$x1 \rightarrow (x4 \wedge x5)$
Application	Raga Yaman if and only if M is present	Thaath Kafi implies M is not present	(Raag Kafi implies g) implies Thaath Kafi	**Raga Yaman implies (Thaath Kafi and not g)**
Abstraction	x1	$(x1 \wedge x2) \rightarrow (x3 \wedge x4)$	x2, $(x4 \wedge x5)$	x5
Application	Raga Yaman	Raga Yaman and g implies Raga Kafi and M	g, M implies Thaath Poorvi	**Thaath Poorvi**

logical formation, and solve them with a dependable accuracy. Algorithm 1 represents the algorithm used for structuring the system.

1) Ontology: We start from the creation of a relational ontology from a knowledge base. This involves turning any knowledge database into a network of nodes and edges. We separate lists of different entities and relationships. We then represent all the relationships with different kinds of labels. These labels have embedded hierarchies. This is done in the following manner:

1) Extract a list of all unique elements from the table
2) Extract a list of all possible types of relationships (column titles)
3) Bind each element in the primary ID column to each other column using the title label
4) Output the lists as Nodes and labels (type of node); and Edges, including source, target and label (type of relationship between source and target entity

a) Data for Raga Ontology: The music Ontology contains of 163 Primary elements, with 365 nodes and 7005 edges. The data are from a primary source of music learning - a seminal treatise titled 'Kramik Pustak Malika' by Dr V N Bhatkhande [10]. The data are represented as shown below.

Edges file:
Source Target Label
id1 id2 thaat

Nodes file:
ID Class Label
id1 raga Bahar
id2 thaat Kafi

Using these two nodes, we can generate a simple question such as: 'Kafi is the thaat of Bahar.', to give to a student to answer as True or False.

b) Basic question generation: In order to generate a basic true/false question, an edge is randomly selected from the files containing the edges. One of the truth-values - true or false, is randomly chosen. If the answer to the question is True, neither the source or the target is changed. If the answer is False, there is another random selection of which of the source or the target is to be changed. If the source is to be changed, a new node, having the same class as the source node, is selected, and the source node is substituted with the new node. The same procedure is followed when the target is to be changed. Additionally, when the source node is changed, a verification is required that the new source node does not have the same target node and label. This is because two source nodes may have the same target node and label. For example, Bahar and Bageshri have the same thaat, Kafi. Thus, it is necessary to check that the newly generated source node does not have the same label and target as the source node.

2) Propositional Logic: In addition to asking basic questions, combinations of edges could be used to ask a wide range of questions. These combinations could be of any type. These rules can be found in Table II and III.

In this algorithm, an edge is randomly selected. One of the rules of propositional logic is also selected and a row from its truth table is picked. All rules except the negation rule require two starting elements. When the negation rule is selected, a procedure similar to the basic question generation is followed. If the input truth-value is True, no changes are to be made. However, if the input truth-value is False, it has to be decided which of the source or the target nodes is to be changed.

When a rule other than the negation rule is selected, a slightly different procedure has to be followed because all other rules require two input truth-values. So, another edge has to be picked. However, if another random edge is picked, the question formed may not be completely intuitive. For instance, if the information stored in the second edge is 'Shadav-Sampoorna is the jati of Tilak Kamod.', and the rule selected is the conjunction rule, with both input truth-values True, the question formed would be 'Kafi is the thaat of Bahar and Shadav-Sampoorna is the jati of Tilak Kamod. True/False'. It is apparent that this question is not very intuitive as both the edges have unrelated information. Thus, it becomes necessary to have a connection between the edges so that the questions formed are meaningful.

In order to have related edges, the second edge has to be picked in such a manner that it has the same source or the same target and label as the first edge. So, if the edges have the same source, a question that could be generated, with the conjunction operation having both input truth-values True, is: The thaat of Bahar is Kafi and its vadi is m. In this example question, Bahar is the source, which is common and Kafi and m are the different targets and thaat and vadi are their respective labels. Similarly, if the edges have the same target and label, a question that could be generated with the conjunction operation having both input truth-values True is: 'Kafi is the thaat of Bahar and Bhimpalasi. True/False'. In this question, Kafi is the target and thaat is the label, both of which are common and Bahar and Bhimpalasi are the different source nodes.

If any of the truth-values in the selected row is False, a false answer needs to be generated. In order to do this, an instance of the same class is selected and substituted. If the target and label are common, an additional verification has to be performed to check that the newly generated source node does not have the same target and label as the original source

```
Data: $n \leftarrow nodes$, $e \leftarrow edges$,
      $operators \leftarrow propositional - logic - operators$,
      $inference - rules \leftarrow all - dependencies$,
      $classes \leftarrow empty - dictionary$,
      $n - grams \leftarrow empty - dictionary$,
for $operator$ in $operators$ do
    Use $operator$ on all truth-values and create truth-table
end
for $instance$, $type$ in $n$ do
    if $type$ in $classes$ then
        Append $instance$ to $classes[type]$
    else
        Add $type$ to $classes$, append $instance$ to
        $classes[type]$
    end
end
for element in n[array] do
    Generate all n-gram sequences of attributes;
end
for sequence,raga in $n - grams$ do
    Add $sequence$ to $n - grams$;
    Add $raga$ to $n - grams[sequence]$
end
while $True$ do
    Pick a random edge from $e$;
    Select choice from { propositional logic, inference,
    querying };
    if choice == propositional logic then
        Select a $rule$;
        Select row from $rule$ with common source / target);
        if common source then
            Select edge from $e$ with different source / target ;
            for each edge selected do
                if truth-value == $False$ then
                    Generate target from class without
                    common edge;
                else
                    continue
                end
            end
        else
            Select edge from $e$ having same target, label but
            not source;
            for each edge selected do
                if truth-value == $False$ then
                    Generate new source from same class;
                    Verify: source, target do not share edge
                else
                    continue
                end
            end
        end
    else
        if choice == inference then
            Select an inference rule and other edge;
            Generate new target belonging to the same class
            for False questions;
        else

        end
    end
    Select an instance from $n - grams$, a rule, two
    $elements$ from $n - grams[instance]$;
    for each raga selected do
        if truth-value == $False$ then
            Select a $element$ not present in
            $n - grams[instance]$
        else
            continue
        end
    end
end
```

TABLE II. CONJUNCTION AND DISJUNCTION

Conjunction			Disjunction		
A	B	A and B	A	B	A or B
T	T	T	T	T	T
T	F	F	T	F	T
F	T	F	F	T	T
F	F	F	F	F	F

TABLE III. NEGATION, EXCLUSIVE OR AND IMPLICATION

Negation		Ex-Or			Implication		
A	Not A	A	B	A XOR B	A	B	A =>B
T	F	T	T	F	T	T	T
F	T	T	F	T	T	F	F
		F	T	T	F	T	T
		F	F	F	F	F	T

node. For instance, if the source node belongs to the class raga, and the label is thaat, it is possible that the newly generated raga has the same thaat. If the source node is the same for both edges, a verification is not required as different labels refer to different properties of the source node.

3) Knowledge Relations: Even though the ontology contains a lot of the information, we need to understand some more information in terms of knowledge relationships. This means asserting some rules that we know despite them not being in the ontology and embedding them into inference relations. Implications that are purely based on logical relations can vary between ontologies. For getting rid of this problem, we enumerate some knowledge relations that are present as a matter of the categories used for the ontologies.

As an example, we present some of these rules that can be described for the raga ontology in particular:

1) Raag R belongs to thaath T implies it has svar S (Thaath enumerates Svars)
2) R belongs to jati J implies Aaroha (/Avaroha) is A (Aaroha can have n of characters defined by the jati name)
3) R has A, implies it has S (Inference being presence of svar)
4) R has A, implies belongs to T (Thaath enumerates constituent notes.)
5) R has V (vadi), implies Pakad is P (V is important in P)
6) R has SV, implies vadi is V (inference being V, S being in different sections of the octave)
7) R has only svars S1, S2, S3, S4. (Inference being too few notes for a raga)

We interpret the use of inference rules from logic for the specific instances of the ontology.

1) Modus Ponens
R has vadi V, belongs to Thaath T. Therefore it's time is M. (Thaath T will have only 2 / 3 times at which it is sung. Vadi being in first or second half will determine which of those times are picked. Thus we can infer that R's time could be M.)
2) Modus Tollens
R has vadi V, doesn't belong to T, but it's time is M. (Time M will have only a couple of thaaths possible. But R doesn't belong to some of them, and R's vadi is in first / second half, so we can infer if this is true or false).

3) Hypoethetical Syllogism: R belongs to T, T has svar S, therefore does R have svar S? (This is like a confusing question)
4) Destructive Dilemma: R belongs to T. T has svar s and S. but R doesn't have S.
5) Constructive Dilemma: R1 belongs to T, R2 also belongs to T. R1 has svar S, R2 has svar S2 (vikrit form).

4) N-gram Strings: Cognitive dependence on finding categories similar based on superficial features such as spellings and structural similarity is high. In the case of a raga ontology, the enumeration of characteristics is in terms of long strings of notes. In order to be able to provide questions that are increasing in the scale of difficulty, we want to provide questions about two categories are very similar in terms of internal constituents. We do this by computing n-gram strings for constituents of the available categories.

For example, if Raga R1 has the notes: S, R, G, P, D, S; Raga R2 has the notes: S, r, G, P, D, S - these two ragas have surface structural similarities in terms of the 4-gram representation in G, P, D, S.

In order to form questions that are of a higher difficulty, we first compute all possible unique n-grams. We then isolate all such n-grams which have more than one source node in common. Thereafter, we form a question about the sources which share the most number of n-grams. These questions are more likely to be intuitive. In the animal ontology, when this case is applied, it results in questions being asked about animals having the same genus. This

III. Solving human generated problems

We parse human generated problems in the same manner. We employ a forward parsing model that separates and extracts entities from ontologies and then binds them using different kinds of knowledge relations. We go about this in four stages:

- Eliminating superfluous tokens such as articles and pronominal referents
- Extracting nodes as nominal tokens and flags for logical tokens of different kinds
- Binding nodes with nodes by parsing copula
- Binding nodes with labels by parsing propositions and conjunctions one by one
- Preparing truth table for given proposition
- Comparing with truth table generated for the same nodes from ontology

IV. Sample Space

The first three modules in the operation increase number of questions by matching with all possibilities. The number of basic questions that can be generated per edge through this method are n to the power 2 * n. Thereafter, the number of computations grows exponentially. For the ontology we have, table IV contains statistics on the sample space for our ontology containing 305(n) nodes and 7005 edges. However, not all questions and answers are useful from the knowledge point of view. Knowledge relations are used to prune the sample space so that we can have a vast but contained and useful repository of questions as mentioned in section II. 3).

TABLE IV. Sample Space for Question Generation

Nodes	Edges	Questions	+ Not	False Edges	Or & And
n	7005	7005	14010	2(14010∧n)	(2(14010∧n))∧4

V. Conclusion

We thus describe a system that can be used to create drills and quizzes to be used by students directly, that can be generated from any database. We use different modules presented in the domain of automatic problem generation, and prune the randomness of these results by capturing knowledge relationships efficiently. The potential for generating a vast number of problems from a small number of categories is exploited.

References

[1] Martin, Brent, and Antonija Mitrovic. "Automatic problem generation in constraint-based tutors." Intelligent Tutoring Systems. Springer Berlin Heidelberg, 2002.

[2] DeMilli, R. A. "Constraint-based automatic test data generation." Software Engineering, IEEE Transactions on 17.9 (1991): 900-910.

[3] Gotlieb, Arnaud, Bernard Botella, and Michel Rueher. "Automatic test data generation using constraint solving techniques." ACM SIGSOFT Software Engineering Notes. Vol. 23. No. 2. ACM, 1998.

[4] Singh, Rishabh, Sumit Gulwani, and Armando Solar-Lezama. "Automated feedback generation for introductory programming assignments." ACM SIGPLAN Notices. Vol. 48. No. 6. ACM, 2013.

[5] Ahmed, Umair Z., Sumit Gulwani, and Amey Karkare. "Automatically generating problems and solutions for natural deduction." Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press, 2013.

[6] Sung, Li-Chun, Yi-Chien Lin, and Meng Chang Chen. "An Automatic Quiz Generation System for English Text." Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on. IEEE, 2007.

[7] Rey, Guillermo lvaro, et al. "Semi-automatic generation of quizzes and learning artifacts from linked data." (2012).

[8] Heilman, Michael. Automatic factual question generation from text. Diss. Carnegie Mellon University, 2011.

[9] Sung, Li-Chun, Yi-Chien Lin, and Meng Chang Chen. "The Design of Automatic Quiz Generation for Ubiquitous English E-Learning System." Technology Enhanced Learning Conference (TELearn 2007), Jhongli, Taiwan. 2007.

[10] Bhatkhande, Vishnu Narayan. Kramik pustak malika. Sangeet Karyalaya, 1993.

[11] Kelkar, T., Ray, A., & Choppella, V. (2015, June). SangeetKosh: An Open Web Platform for Music Education. In Proceedings of the The 15th IEEE International Conference on Advanced Learning Technologies, June 2015, Hualien, Taiwan.