Undefined 1 (2015) 1–5
IOS Press

Automated Generation of Assessment Tests from Domain Ontologies

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Vinu E.V * and P Sreenivasa Kumar
Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India
E-mail: {vinuev,psk}@cse.iitm.ac.in

Abstract.
OWL-ontologies are structures which are used for representing knowledge of a domain, in the form of logical axioms. Research on pedagogical usefulness of these knowledge structures has gained much attention these days. This is mainly due to the number of on-line ontology repositories and the ease in publishing knowledge in the form of ontologies. One another reason for this trend in research, is due to the changing education-style — more learners prefer to take-up on-line courses, than attending a course in a typical room setup. In this case, assessments — both prerequisite and post-course-requirement evaluations — will be a challenging task. In this paper, we explore an automated technique for generating question items like multiple choice questions (MCQs), from a given domain ontology. Furthermore, we investigate the aspects such as (1) how to find the difficulty-level of a generated MCQ; (2) what are the heuristics to follow to select a small set of MCQs which are relevant to the domain; (3) how to set a test which is having higher, medium or lower hardness level, in detail. We propose novel techniques to address these issues. We tested the applicability of the proposed techniques by generating MCQs from several on-line ontologies, and verified our results in a class-room setup — incorporating real-students and domain-experts.

Keywords: MCQ Generation, Question Generation, Ontologies, Intelligent Tutoring System

1. Introduction

Ontologies, the knowledge representation structures which are useful in modeling knowledge in a variety of domains, have widely flourished in recent years. This is due to the advancement of Semantic Web technologies and, due to ease of publishing knowledge in online repositories. The use of knowledge captured in these ontologies, by pedagogical systems, in giving feedbacks to the learners, is an advancing area of research.

This paper particularly addresses the scope of Web Ontology Language (OWL) ontologies in generating question-sets which can be employed for conducting large-scale multiple choice questions (MCQs) based assessment tests. This problem has recently attracted a notable attention in the research as well as education communities [13], particularly due to its importance in the new emerging education styles; for instance, the tests conducted as part of online courses like the MOOCs (Massive Open Online Courses) typically consist mainly of MCQs [21,5]. In addition, the amount of time, money and skill required for setting up a question-set is enormously high [8,20], making it necessary to have an alternate solution.

There are several works in the literature, which describe the usefulness of OWL ontologies in generating MCQs [18,12,6,3,25]. Studies in [2] have shown that ontologies are good for generating *factual* (or *knowledge-level*) MCQs. These knowledge-level questions help in testing the first level of Bloom's taxon-

*Corresponding author. E-mail: vinuev@cse.iitm.ac.in, mvsquare1729@gmail.com

omy [10], a classification of cognitive skills required for learning.

Recently, publications like [2,1,23], show that factual-MCQ (in short, F-MCQ) items can be generated from the assertional facts (ABox axioms) associated with the ontology. In [23], the authors categorized the approaches, which use ABox axioms for question generation, into two types: (1) *Generic factual-question generation* — pattern based methods — and (2) *Ontology-Specific factual-question generation* — methods which generate questions which are specific to the domain. They also introduced a systematic method for Generic (pattern-based) factual-MCQs, by considering different combinations of predicates (or properties) associated with an instance in an ontology. Two major drawbacks associated with the practicality of this pattern-based question generation were: (1) human intervention is needed to screen the irrelevant or out-of-domain questions, (2) the approach generates thousands of MCQs, making it difficult even for a human expert to make the selection of a small question-set.

They addressed these issues by proposing three screening techniques based on a few observed heuristics, for selecting only those questions which are ideal for conducting a domain-specific MCQ-test. Through experimental results, they show that the automatically generated question-sets can be compared satisfactorily to those prepared by domain experts, in terms of precision and recall.

The work described in this paper is an extension of their work [23] ("Improving large-scale assessment tests using ontology based approach"). The contributions of this paper can be listed as follows:

1. A novel method to determine the hardness score of a generated MCQ stem.
2. A detailed study of *Generic* factual-MCQs, using patterns that involve more than two predicates.
3. A generic (ontology independent) technique to generate *Ontology-Specific* factual-MCQs.
4. An algorithmic way to control the difficulty-level of a question-set.

An overview of our four contributions are given in Section 2.

2. Overview of the contributions

2.1. *Contribution*-1

Similarity-based theory [4] was the only effort in the literature [14,7], which helped in determining or controlling the hardness of an ontology generated MCQ. The difficulty-level (also called hardness score) calculated by similarity-based theory considers only the similarity of the distracting answers with the correct answer — high similarity implies high hardness score and vice versa. In many a case, the stem (question statement) of an MCQ is also a deciding factor for the hardness of an MCQ. For instance, the predicate combination which is used to generate a stem can be chosen such that they make the MCQ harder or easy to answer. Also, the use of *indirect addressing of instances*[1] in a stem, can affect its hardness. We investigate these aspects in Section 7 and, we propose a novel method for deciding the hardness score of a stem.

An empirical study in a classroom setup (see Section 10.2) was done to evaluate the employability of the proposed method.

2.2. *Contribution*-2

As we mentioned before, the F-MCQs — to test a learner's proficiency of the factual knowledge of a domain — that can be generated from a given ontology can be classified into: Generic F-MCQs and Ontology-Specific F-MCQs (Section 3.2 and Section 3.3 for details). An initial study on the generation techniques of the former MCQs type had been done in [23], where questions are limited to property combinations of at most two predicates. In Section 4, we investigate approaches (or patterns) which generate questions that involve more than two predicates as well. Later in Section 4.1, we describe a study that we have made on a large set of real-world factual-questions — obtained from different domains — to explore the pragmatic usefulness and the scope of our approach.

2.3. *Contribution*-3

A generic technique to generate (a subset of possible) Ontology-Specific F-MCQs is proposed in Section 5. A detailed illustration of this type of MCQs is given in Section 3.3. The term Ontology-Specific F-MCQs have been coined first in [23], but there the authors limited the work to Generic F-MCQs.

[1] Instead of using the instance "Barack_Obama", one can use "44^{th} president of the U.S."

2.4. Contribution-4

In Section 8, we propose a practically adaptable algorithmic method to control the difficulty-level of a question-set. This method controls the difficulty-level by varying the count of the questions which are having (relatively) high difficulty-level in the question-set.

3. Preliminaries

3.1. Multiple Choice Questions (MCQs)

An MCQ is a tool that can be used to evaluate whether (or not) a student has attained a certain learning objective. It consists of the following parts:

- ***Stem*** *(S).* Statement that introduces a problem to a learner.
- ***Choices***. Set of options corresponding to S, denoted as $A = \{ A_1, A_2, ..., A_m \}, m \geq 2$. It can be further divided into two sets:
  - ***Key***. Set of correct options, denoted as $K = \{ A_1, A_2, ..., A_i \}, 1 \leq i < m$.
  - ***Distractors***. Set of incorrect options, denoted as $D = \{ A_{i+1}, ..., A_m \}$.

$Note$: In this paper we assume K as a singleton set. We fix the value of m, the number of options, in our experiments as 4, as it is the standard practice in MCQ tests.

3.2. Generic F-MCQs

Generic pattern-based F-MCQs are those MCQs whose stems can be generated using simple SPARQL templates. These stems can be considered as a set of conditions which ask for an answer which is explicitly present in the ontology. Questions like *Choose a C?* or *Which of the following is an example of C?* (where *C* is a concept symbol), are some of the examples of generic factual-questions.

The distractors for these MCQs are selected from the set of instances (or values) of the ontology which belong to the intersection classes of the domain or range of the predicates (known as *Potential-set*) in the stem. Detailed explanation of distractor generation is given in Section 9.

3.3. Ontology-Specific F-MCQs

The questions which we discuss in this paper can be categorized into knowledge-level questions (or simply questions which check students' factual knowledge proficiency) of Blooms taxonomy [9], a classification of cognitive skills required for learning. Within this category, we observed that, Ontology-Specific F-MCQs require more reasoning skills to answer than generic F-MCQs, and may not be necessarily generated from a generic pattern (or template). For e.g., "*Choose the state which is having the longest river.*", is an Ontology-Specific question (unless there are predicates in the ontology, that explicitly specify the answer). This question can be answered only by a learner who knows about the states of a country, its rivers and the length of the rivers in it; and she should be able to reason over the known facts. Our experiments based on Item Response Theory[2] (IRT), have shown that such MCQs are indeed difficult for a below-average learner to answer correctly.

4. Study on Generic F-MCQs

As we have mentioned before, the stem of a generic F-MCQ is obtained from a set of conditions formed using different combinations of predicates (unary or binary role assertions) associated with an instance in an ontology. Example-1 is such an MCQ, which is framed from the following assertions that are associated with the (key) instance `birdman`.

```
Movie(birdman)
isDirectedBy(birdman,alejandro)
hasReleaseDate(birdman,"Aug 27 2014")
```

Example 1 Choose a `Movie`, which `isDirectedBy` `alejandro` and `hasReleaseDate` "`Aug 27, 2014`".

Options	
a.	Birdman
b.	Titanic
c.	Argo
d.	The King's Speech

[2] http://www.creative-wisdom.com/computer/sas/IRT.pdf

The possible property[3] combinations of size[4] one w.r.t. an instance x can be denoted as: $x\ \overrightarrow{O_1}\ i_1$, $x\ \overleftarrow{O_1}\ i_1$, $x\ \overrightarrow{D_1}\ v_1$ and $x\ \overrightarrow{a}\ C_1$, where i_1 is an instance, $\overrightarrow{a}$ is `rdf:type`, $\overrightarrow{O_1}$ and $\overleftarrow{O_1}$ represent object properties of different directions, $\overrightarrow{D_1}$ denotes datatype property, v_1 stands for the value of the datatype property and C_1 is a class name. We call the instance x as the *reference-instance* of the question-pattern. The arrows ($\leftarrow$ and $\rightarrow$) represent the directions of the properties w.r.t. the reference-instance. In this paper, we often use the terms question-pattern and property combination interchangeably, but the former denotes the property combination along with the position of the key.

Table 1 shows the formation of possible property combinations of size two and three by adding predicates to the four combinations of size one. The repetitions in the combinations are marked with the symbol "*". Note that, in the pattern representation, we consider only the directionality and type of the properties, but not their order. Therefore the combinations like $i_2\ \overleftarrow{O_2}\ x\ \overleftarrow{O_1}\ i_1$ and $i_2\ \overrightarrow{O_2}\ x\ \overrightarrow{O_1}\ i_1$ are considered to be the same. We refer one as duplicate of the other. After avoiding the duplicate combinations, we get 4 combinations of size one, 10 combinations of size two and 26 combinations of size three.

These 40 predicate combinations can be used as the basic set of question-patterns for constructing Generic FQs[5]. At this point, we cannot further limit the number of combinations based upon their usefulness in generating real-world FQs, or we may not be able to group these combinations based on some semantic similarities. Also, the question of which among the variables — x, i, and v — of these combinations, corresponds to a key, cannot be addressed at this stage. In the next section, we do an empirical study on a large set of real-world FQs, and identify common FQs and their features; then, we select a subset of the proposed question patterns (along with the possible position of their keys) as the *necessary* property combinations for real-world FQ generation.

4.1. An empirical study of real-world FQs

Since there are no rules as such for how a FQ should look like, it was not possible to fix a scope for our study on Generic FQs. In order to claim that our approach could cover FQs which are useful in conducting any domain specific test and are chosen by experts with different levels of expertise, we analyzed 1748 FQs gathered from three different domains: United States Geography domain[6], Job posting domain[7] and Restaurant domain[8]. The FQs (question-set) corresponding to these domains are gathered by Mooney's research group[9] of the University of Texas, using a web-interface from real-people.

From the question-sets, we removed invalid questions and (manually) classified the rest into Generic FQs and Ontology-Specific FQs. We manually identified 570 Generic FQs and 729 Ontology-Specific FQs from the question-sets. We then tried to map each of these Generic FQs to the pattern combinations which we have discussed in Section 4. We could map each of the 570 Generic FQs to at least one of the proposed structural combinations of predicates in Table 1. This generalizes the fact that our patterns are effective in extracting almost all kinds of real-world Generic FQs that could be generated from a given domain ontology.

4.2. Predicate combinations to FQ patterns

We have observed that most of the property combinations are not being mapped to by any real-world Generic FQs. Out of our 40 combinations only 13 are necessary to generate such FQs. We call them as the *necessary question-patterns*.

From the 13 necessary property combinations, we framed 19 question patterns based on our empirical study of the real-world FQs, by identifying the variables whose values can be considered as keys — we call such variables as the *key-variables* of the patterns. We list the question patterns corresponding to the necessary property combinations in Table 2. The circled variables in the patterns denote the positions of their key (i.e., the key-variables). The square boxes represent the variables whose values can be removed while framing the stem. For example, the question: *What is the population of the state with capital Austin?* can be generated from the pattern: $\textcircled{v}\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ i$, with v as the key-

[3] Predicates, which include both concept names and role names
[4] Signifies the number of predicates in a combination
[5] FQ stands for Factual-question

[6] https://files.ifi.uzh.ch/ddis/oldweb/ddis/fileadmin/ont/nli/geoqueries_877.txt (last accessed 1st July 2015)
[7] https://files.ifi.uzh.ch/ddis/oldweb/ddis/fileadmin/ont/nli/jobqueries_620.txt (last accessed 1st July 2015)
[8] https://files.ifi.uzh.ch/ddis/oldweb/ddis/fileadmin/ont/nli/restaurantqueries_251.txt (last accessed 1st July 2015)
[9] https://www.cs.utexas.edu/ mooney/

Table 1
Property combinations of size 1, 2 and 3

Property Combinations: ↓ *Size:* 1	2	3
$x\,\overrightarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,\overrightarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i$ $i\,\overleftarrow{O}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i$ $i\,\overleftarrow{O}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $i\,\overleftarrow{O}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i$
	$i\,\overrightarrow{O}\,x\,\overrightarrow{O}\,i$	$i\,\overrightarrow{O}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i^*$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i$
	$v\,\overleftarrow{D}\,x\,\overrightarrow{O}\,i$	$v\,\overleftarrow{D}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i^*$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i^*$
	$C\,\overleftarrow{a}\,x\,\overrightarrow{O}\,i$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i^*$
$x\,\overleftarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,\overleftarrow{O}\,i^*$	– – – –
	$i\,\overrightarrow{O}\,x\,\overleftarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,(\overrightarrow{O}\,i)\,\overleftarrow{O}\,i^*$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{D}\,v)\,\overleftarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{a}\,C)\,\overleftarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overleftarrow{O}\,i)\,\overleftarrow{O}\,i$
	$v\,\overleftarrow{D}\,x\,\overleftarrow{O}\,i$	$v\,\overleftarrow{D}\,x\,(\overrightarrow{O}\,i)\,\overleftarrow{O}\,i^*$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{D}\,v)\,\overleftarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{a}\,C)\,\overleftarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overleftarrow{O}\,i)\,\overleftarrow{O}\,i$
	$C\,\overleftarrow{a}\,x\,\overleftarrow{O}\,i$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overleftarrow{O}\,i^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overleftarrow{O}\,i$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overleftarrow{O}\,i$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overleftarrow{O}\,i$

Property Combinations ↓ *Size:* 1	2	3
$x\,\overrightarrow{D}\,v$	$i\,\overleftarrow{O}\,x\,\overrightarrow{D}\,v^*$	– – – –
	$i\,\overrightarrow{O}\,x\,\overrightarrow{D}\,v^*$	– – – –
	$v\,\overleftarrow{D}\,x\,\overrightarrow{D}\,v$	$v\,\overleftarrow{D}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{D}\,v^*$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{D}\,v$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{D}\,v$ $v\,\overleftarrow{D}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{D}\,v^*$
	$C\,\overleftarrow{a}\,x\,\overrightarrow{D}\,v$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{D}\,v^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{D}\,v^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{D}\,v$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{D}\,v^*$
$x\,\overrightarrow{a}\,C$	$i\,\overleftarrow{O}\,x\,\overrightarrow{a}\,C^*$	– – – –
	$i\,\overrightarrow{O}\,x\,\overrightarrow{a}\,C^*$	– – – –
	$v\,\overleftarrow{D}\,x\,\overrightarrow{a}\,C^*$	– – – –
	$C\,\overleftarrow{a}\,x\,\overrightarrow{a}\,C$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{a}\,C$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{a}\,C$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{a}\,C$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{a}\,C$

Table 2
Question patterns for real-world FQ generation, where circled variables denote key-variables

No.	Question Pattern	Stem-template	Potential-set w.r.t. key-variables
1	$x\ \overrightarrow{O}\ \textcircled{i}$	*Choose a/the O of x.*	$Range(O)$
2	$x\ \overleftarrow{O}\ \textcircled{i}$	*Choose the one with O x.*	$Domain(O)$
3	$x\ \overrightarrow{D}\ \textcircled{v}$	*Choose a/the D of x.*	$Range(D)$
4	$\textcircled{x}\ \overrightarrow{a}\ C$	*Choose a C.*	C
5 a.	$\textcircled{v}\ \overleftarrow{D}\ x\ \overrightarrow{O}\ i$	*Choose a/the D of x with O i.*	$Range(D)$
b.	$\textcircled{v}\ \overleftarrow{D}\ \boxed{x}\ \overrightarrow{O}\ i$	*Choose a/the D of the one with O i.*	$Range(D)$
6	$\textcircled{v}\ \overleftarrow{D}\ \boxed{x}\ \overleftarrow{O}\ i$	*Choose the D of the one which is the O of i.*	$Range(D)$
7	$C\ \overleftarrow{a}\ \textcircled{x}\ \overrightarrow{D}\ v$	*Choose a/the C with D v.*	$Domain(D) \sqcap C$
8 a.	$C\ \overleftarrow{a}\ \textcircled{x}\ \overrightarrow{O}\ i$	*Choose a/the C with O i.*	$Domain(O) \sqcap C$
b.	$C\ \overleftarrow{a}\ \boxed{x}\ \overrightarrow{O}\ \textcircled{i}$	*Choose a/the O of a C.*	$Range(O)$
9 a.	$C\ \overleftarrow{a}\ \boxed{x}\ \overleftarrow{O}\ \textcircled{i}$	*Choose the one with a C as O.*	$Domain(O)$
b.	$C\ \overleftarrow{a}\ \textcircled{x}\ \overleftarrow{O}\ i$	*Choose a/the C, which/who is O of i.*	$Range(O) \sqcap C$
10 a.	$\textcircled{i_1}\ \overrightarrow{O_1}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O_2}\ i_2$	*Choose the one whose O_1 is a C with O_2 i_2.*	$Domain(O_1)$
b.	$i_1\ \overrightarrow{O_1}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O_2}\ \textcircled{i_2}$	*Choose a/the O_2 of a C which/who is O_1 of i_1.*	$Range(O_2)$
11 a.	$v\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ \textcircled{i}$	*Choose a/the O of a C with D v.*	$Range(O)$
b.	$v\ \overleftarrow{D}\ \textcircled{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ i$	*Choose a/the C with D v and O i .*	$Domain(D) \sqcap C \sqcap Domain(O)$
c.	$\textcircled{v}\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ i$	*Choose a/the D of a C with O i.*	$Domain(D) \sqcap C \sqcap Domain(O)$
12	$\textcircled{v}\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overleftarrow{O}\ i$	*Choose the D of a C who/which is O of i.*	$Range(D)$
13	$i_1\ \overleftarrow{O_1}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O_2}\ \textcircled{i_2}$	*Choose a/the O_2 of a C whose O_1 is i_1.*	$Range(O_2)$

variable, D as the property `statePopulation`, O as `hasCapital`, C as the concept `State` and i as the individual `austin`. Clearly, the value of the variable x is not mandatory to frame the question statement. If the variable value is incorporated in the question, we are providing additional information to the test takers, making the question more direct and less difficult to answer.

A stem-template is associated with each of the patterns in Table 2 to generate corresponding (controlled English) natural language FQs. Further enhancement of the readability of the stem is done by tokenizing the property names in the stem. Tokenizing includes word-segmentation[10] and processing of camel-case, underscores, spaces etc.

[10]Word-segmentation is done by using Python WordSegment (https://pypi.python.org/pypi/wordsegment — last accessed 11th May 2015), an Apache2 licensed module for English word segmentation

4.3. Practicality Issue of pattern-based question generation

For the efficient retrieval of data from the knowledge base, we transform each of the patterns into SPARQL queries. For example, the stem-template and query corresponding to the pattern $C_1\ \overleftarrow{a}\ x\ \overrightarrow{O_1}\ i_1$ (Pattern-8) are:

– *Choose a* [?C1] *with* [?O1] [?i1].

```
select ?C1 ?x ?O1 ?i1 where { ?x a ?C1.
?x ?O ?i1. ?O1 a owl:ObjectProperty. }
```

These queries, when used to retrieve tuples from ontologies, may generate a large result set. Last column of Table 3 lists the total count of tuples that are generated using the 19 question-patterns from a selected set of domain ontologies. These tuple counts represent the possible generic factual-questions that can be generated from the respective ontologies. From Restaurant

ontology, using the query corresponding to Pattern-6 alone, we could generate 288594 tuples.

An MCQ based exam is mainly meant to test the wider domain knowledge with a fewer number of questions. Therefore, it is required to select a small set of significant tuples from the large result set, to create a good MCQ question-set. But, the widely adopted method of random sampling can result in poor question-sets (we have verified this in our experiment section). In Section 6, we propose three heuristic based techniques to choose the most appropriate set of tuples (questions) from the large result set.

Table 3
Specifications of ontologies and the respective tuple count

Ontology	Indviduals	Concepts	Object properties	Datatype properties	Total tuple count
Mahabharata	181	17	24	9	72074
Geography	713	9	174	11	449227
DSA	115	54	54	5	48467
Restaurant	9747	4	9	5	1850762
Job	4138	7	7	12	877437

5. Study on Ontology-Specific F-MCQs

The MCQ question stems, like "*Choose the state with the highest population*", that are very specific to a particular domain can be generated by making use of the datatype of the property values along with some additional computations.

Many of the XML Schema datatypes are supported by OWL-2 DL [16]. OWL-2 DL has datatypes defined for Real Numbers, Decimal Numbers, Integers, Floating-Point Numbers, Strings, Boolean Values, Binary Data, IRIs, Time Instants and XML Literals.

For illustrating the usefulness of these datatypes in generating ontology-Specific MCQ stems, consider the following statements:

```
State(arizona)
State(texas)
hasPopulation(arizona,3232323^^xsd:int)
hasPopulation(texas,23232^^xsd:int)
```

Using the pattern $C \overleftarrow{a} \boxed{x} \overrightarrow{D} (i)$, the following set of tuples can be generated:

```
State,arizona,hasPopulation,3232323^^xsd:int
State,texas,hasPopulation,23232^^xsd:int
```

After grouping the tuples w.r.t. the similar datatype properties that they contain (here `hasPopulation`), interesting questions, based on the border values of the datatype properties, can be made. For example, "*Choose the state with highest population.*" and "*Choose the state with lowest population.*" are two interesting questions that can be generated from the ontology statements under consideration.

5.1. Explicit-Semantic-Analysis based stem enhancement

Having a property in hand, to fix which quantifying adjective — highest and lowest or longest and shortest — to use, is determined by calculating pairwise relatedness score and, then, choosing the one with highest score using Explicit Semantic Analysis (ESA) [15] method. ESA method computes semantic relatedness of natural language texts with the aid of very large scale knowledge repositories (like Wikipedia). EasyESA[11] [11], an infrastructure consisting of an open source platform that can be used as a remote service or can be deployed locally, is used in our implementation, to find pairwise relatedness scores.

The pair — (predicate, predefined-adjective) — with highest ESA relatedness score are used for framing the question.

Table 4
The ESA scores of some sample predicate–adjective pairs

Predicate	Adjective	ESA relatedness score
has Population	highest	0.0106816739
has Population	longest	0.0000000000
has Population	lowest	0.0132820251
has Population	longest	0.0000000000

For example, as shown in Table 4, the datatype property `hasPopulation` can be used along with "highest" or "lowest", depending on the border value under consideration, as those pairs have comparatively high relatedness score.

Out of the large set of datatypes offered by OWL-2 DL, datatypes of Binary Data, IRIs and XML Literals are avoided for stem formation, as they are not useful in generating human-understandable stems.

[11] http://easy-esa.org/

5.2. Question generation in detail

The tuples generated using the 19 patterns (in Table 2), can be grouped based on the properties they contain. We call the ordered list of properties which are useful in grouping as the *property sequence* of the tuples in each group. From the grouped tuples, we select only those groups whose property sequence contain at least one datatype property, for generating Ontology-Specific questions. Consider the tuples-set given in Table 5, which is generated using the property combination $C \overleftarrow{a} x \overrightarrow{D} v$ from Geography ontology. GROUP BY and ORDER BY clauses are used along with the patterns' SPARQL templates for grouping and sorting respectively (w.r.t. the datatype values).

In Table 5, the highlighted rows, which correspond to the border values of the datatype property, can be used for framing the Ontology-Specific questions — we call these rows as the *base-tuples* of the corresponding Ontology-Specific questions (we use this term in the subsequent sections). The datatype properties in the 1^{st} and 3^{rd} highlighted rows are paired with the predefined-adjectives (like maximum, highest, oldest, longest, etc.) and the pair with the highest ESA relatedness score is found. Then the stemmed predicate (E.g., hasPopulation is stemmed to "Population"), the adjective and the template associated with the pattern are used to generate stems of the following form (where the underlined words correspond to the predicates used):

- Choose the State with the *highest* population. (Key: Arizona)
- Choose the River with the *longest* length. (Key: Ouachita)

Similarly, the predicates in the 2^{nd} and 4^{th} highlighted rows can be paired with adjectives like minimum, shortest, smallest, minimum, etc., to generate stems of the form:

- Choose the State with the *lowest* population. (Key: Connecticut)
- Choose the River with the *shortest* length. (Key: Neosho)

Table 5
Tuple selection for generating Ontology-Specific F-MCQ

C	x	D	v	Property seq.
State	connecticut	hasPopulation	5020	
State	florida	hasPopulation	68664	{State,
State	colorado	hasPopulation	104000	hasPopulation}
...	...	...	...	
State	arizona	hasPopulation	114000	
River	neosho	length	740	
River	wasbash	length	764	{River,
River	pecos	length	805	length}
...	...	...	...	
River	ouachita	length	973	

6. Question-set generation heuristics

In [23], the authors proposed three screening heuristics which mimic the selection heuristics followed by human experts to generate question-sets that are unbiased and cover the required knowledge boundaries. In this section, we briefly summarize the three screening methods and then, we explain the improvements made in the third heuristics in detail. We use Movie ontology, for illustration purpose.

Even though these heuristics were meant for Generic FQs, we apply the same heuristics, except the third heuristic, to Ontology-Specific questions as well. This is achieved by considering the base-tuples of Ontology-Specific questions.

6.1. Screening based on Property sequence

The rationale for this screening method was to remove tuples that generate routine questions which are less likely to be chosen by a domain expert for conducting a test, from the large tuple set. For example, in Movie ontology, the questions formed using the property sequence {`isProducedBy`, `isDirectedby`} can be categorized as trivial questions — routine questions that are less used by domain experts for testing purpose — when compare to questions that are formed using {`wonAward`, `isBasedOn`}. This is because, the properties in the former property sequence are present for all movie instances — making the questions trivial — and the properties in the latter property sequence are present only for selected movie instances — making the questions nontrivial ones.

6.1.1. Method

Tuples retrieved using the SPARQL queries corresponding to the 19 patterns were given as input. Similar to what we have seen in Section 5.2, the tuples were then grouped based on their property sequences. A triviality score was then assigned to the property sequences based on the number of tuples they contain.

The triviality score — called *Property Sequence Triviality Score (PSTS)* — of a property sequence P was defined in [23] as:

$$PSTS(P) = \frac{\#\text{ Instances satisfying all the properties in P}}{\#\text{ Instances in the Potential-set of } P}$$

Potential-set of P denotes the set of instances which *may* possibly satisfy the properties in P. Potential-set of P is denoted by the expression $Type(q, P, r)$, where q is the question pattern used for generating the tuples, r denotes a reference position in the pattern (see the following example). $Type(q, P, r)$ is defined as the intersection of the class constraints and the domain and range of those properties in P which are associated with r. Consider $P = \{p_1, p_2\}$ in the pattern $q = i_2\ \overrightarrow{p_1}\ x\ \overrightarrow{p_2}\ i_1$ and r as the pivot instance x. Then, $Type(q, P, x)$ is taken as $Range(p_1) \sqcap Domain(p_2)$. Similarly for $P = \{C_1, p_1\}$ and $q = C_1\ \overleftarrow{a}\ x\ \overleftarrow{p_1}\ i_1$, $Type(q, P, x) = C_1 \sqcap Range(p_1)$. For the same q and P, if r is i_1, $Type(q, P, i_1)$ is taken as $Domain(p_1)$. In $PSTS$ calculation, the key-variable is taken as the reference position r for finding the potential-set. The third column of Table 2 lists the generic formula to calculate the potential-sets (w.r.t. the *key-variables*) for the 19 patterns.

A suitable (ontology-Specific) threshold for $PSTS$ is fixed based on the number of tuples to be filtered at this level of screening.

6.2. Screening based on key concepts

The authors of [23] have pointed out that, screening based on the triviality score does not always guarantee a small set of questions relevant for a test. For instance, consider the Movie ontology, where all the details related to movies are present, including the places where the movies have been shot. Since pattern-based method is adopted for question generation, after the first level of screening, the tuples corresponding to the location details of the movie can also become a part of the result set. The following are the two sample questions which can be a part of the result set.

- Choose the movie which is based on "The Great Escape" and won an Oscar-award.
- Choose the Sovereign state with capital Edinburgh and having largest city Glasgow.

Clearly, the second stem will not be selected by a domain expert for conducting a movie related test, since the question is more related to a domain which talks about states, their geographies and their governing bodies, than a movie domain. To overcome this issue, the authors had proposed a selection heuristic based on key concepts; the method can be summarized as follows.

6.2.1. Method

In [23], they considered the instances in the tuples which correspond to the reference instances in the respective question-patterns. If the instance belongs to a key concept of the domain, the question which is framed out of it, is considered as relevant for conducting a domain related test.

The key concepts of ontologies were extracted by using the KCE (Key Concept Extraction) API. This API is based on the approach by [19], where the important concepts are identified by considering toplogical measures like *density* and *coverage*, and statistical and lexical measures like *popularity*, and cognitive criteria like *natural categories*.

The number of tuples to be screened in this level, is controlled by varying the count of the key concepts.

6.3. Screening based on similarity of tuples

The tuple-set S, selected using the first two levels of screening, may contain (semantically) similar tuples; they will make the final question-set biased. To avoid this, selecting only a representative set of tuples from among these similar set of tuples is necessary. In [23], this issue was addressed by considering an undirected graph $G = (V, E)$, with vertex set $V = \{t \mid t \in S\}$, and edge set $E = \{\, (t_1, t_2) \mid t_1, t_2 \in S\ and\ Similarity(t_1, t_2) \geq c \,\}$, where $Similarity(.)$ is a symmetric function which determines the similarity of two tuples with respect to their reference-instances and c is the minimum similarity score threshold. From the graph, a minimum dominating set (i.e., a dominating set[12] of minimum cardinality) of nodes was selected as the set of representative tuples. The similarity measure that they adopted is as follows:

$$Similarity(t_1, t_2) = \frac{1}{2}\left(\frac{\#(\ X(P(t_1)) \cap X(P(t_2))\ )}{\#(\ X(P(t_1)) \cup X(P(t_2))\ )}\right.$$

$$\left. + \frac{\#Triples\ in\ t_1\ Semantically\ Equivalent\ to\ triples\ in\ t_2}{Max(\#Triples\ in\ t_1,\ \#Triples\ in\ t_2)}\right)$$

In the equation, $P(t)$ represents the property sequence of t, and $X(P(t))$ denotes the set of instances (in

[12] A dominating set for a graph $G = (V, E)$ is the subset U of V s.t. $\forall v \in V \backslash U$, v is adjacent to at least one member of U.

Table 6
Selection of representative tuples from two groups. The highlighted rows denote the representative tuples — selected based on *popularity*

x (Pivot instance)	O_1	i_1	O_2	i_2	Popularity
argo	wonAward	oscar_12	isBasedOn	the_great_escape	7.20
a_beautiful_mind	wonAward	oscar_01	isBasedOn	a_beautiful_mind_novel	8.26
forest_gump	wonAward	oscar_94	isBasedOn	forest_gump_novel	15.44
h_potter_and_the_sorcerers_stone	directedBy	chris_columbus	hasNextSequel	h_potter_and_the_chamber_of_secrets	34.51
jurassic_park	directedBy	steven_spielberg	hasNextSequel	the_lost_world_jurassic_park	22.56
hobbit_an_unexpected_journey	directedBy	peter_jackson	hasNextSequel	hobbit_the_desolation_of_smaug	14.47
national_treasure	directedBy	jon_turteltaub	hasNextSequel	national_treasure_book_of_secrets	7.89

the ontology) which satisfies the properties in $P(t)$. The equation calculates the similarity score of two tuples based on the relationship between (unary and binary) predicates in one tuple to their counterparts in the other tuple, and the number of the semantically similar triples in them.

In our observation, selecting representative tuples based on minimum dominating set (MDS) — using an approximation algorithm[13] — is more like a random selection of representative nodes ensuring the dominating set constraints. Therefore, to improve the quality of the result, instead of simply finding the MDS, we select the representative nodes based on their *popularity*; details are given in the next subsection.

6.3.1. Method

The tuples that are screened after two levels of filtering are grouped based on their similarity scores; these tuples often show similarity to multiple groups, we avoid such cases by relating them to the one group to which they show maximum similarity.

Within a group, a *popularity* based score is assigned to each of the tuples. The most-popular tuple from each of the groups is considered as the representative tuple. The widely used popularity measure for an instance is based on the count of the instances of the other classes that are connected to it [22]; we make use of this measure to find the popularity of an instance. By considering the popularity of instances in a tuple, we define the popularity of the tuple as:

$$\textit{Popularity}(t) = (1/2)\mathcal{C}_{t.r}(t) + \sum_{i=1}^{n} \log\left(1 + \mathcal{C}_{x_i}(t)\right)$$

In the equation, $t.r$ denotes the reference instance of t, the set, $\{ x_1, x_2, ..., x_n \}$, denotes the instances other than the reference instance in t. $\mathcal{C}_j(t)$ represents the connectivity (defined below) of the instance j in the tuple t. The popularity of a tuple is defined as the sum of the connectivities of its instances, by giving more preference to the connectivity value of the reference instance[14] — sum of half the connectivity value of the reference instance and $\log$ of the connectivity values of the other instances.

The connectivity of an instance x in tuple t is defined as follows, where C_x and C_y are concepts in the ontology $\mathcal{O}$.

$$\mathcal{C}_x(t) = \#\{R(y,x) | x \in C_x \wedge y \in C_y \wedge R(y,x) \in \mathcal{O} \wedge not(C_x \sqsubseteq C_y) \wedge not(C_y \sqsubseteq C_x)\}$$

The equation gives the count of the instances which are related to x by a relation R and whose class types are not hierarchically (sub-class–super-class relationship) related to the class of x.

An illustration of the selection of representative tuples is shown in Table 6, where the highlighted tuples denote the selected ones. In the table, the third tuple and the first tuple in the group-1 and group-2 respectively have higher popularity score than the rest of the tuples in the respective groups, making them suitable candidates for the question-set.

7. Stem based hardness determination

One possible way to decide the hardness of a stem is by finding how its predicate combination is making it difficult to answer. Our study on factual-questions which have been generated from different domain ontologies shows that, increasing the answer-space of the predicates in the stem has an effect in

[13] JGraph MDS

[14] Instance corresponding to the reference variable of the pattern.

the difficulty-level of the question. For example, the stem "*Choose a President who was born on Feb 12th 1809.*" is more difficult to answer than "*Choose an American President who was born on Feb 12th 1809.*" This is because, the answer-space of (some of) the conditions in the former question is broader than the answer-space of (some of) the conditions in the latter. The answer-space of the condition *Choose a President* in the first stem, is larger than the condition *Choose an American President* in the second stem. Being a more generic concept (unary predicate) than `AmericanPresident`, the concept `President`, when used in a stem, makes the question difficult to answer. Therefore, a practical approach to make a stem harder is by incorporating a predicate p_1 which is present for large number of instances, along with a predicate p_2 which is present only for comparatively less number of instances, so that p_1 may deviate the learner away from the correct answer and p_2 may direct her to the correct answer.

The predicate combinations of such type can be easily identified by finding those property sequences with less triviality score; this is because, all the predicate combinations with at least one specific predicate and at least one generic predicate, will have a less PSTS. But, having a less $PSTS$ does not always guarantee that one predicate in it is generic when compared to the other roles in the property sequence; the following condition also needs to be satisfied.

$$\exists\, p_1, p_2 \in P \text{ such that } \#I(p_1) >> \#I(p_2) \quad (1)$$

In the condition, P represents the property sequence and $\#I(p)$ denotes the number of instances satisfying the property p.

The tuples that satisfy Condition-1, can be assigned a difficulty score based on its triviality score as shown in Eq. 2, where P_t denotes the property sequence corresponding to the tuple t. The equation guarantees that a tuple with a high $PSTS$ value will get a low hardness value and vice versa. We consider the difficulty-level of questions as a constant value[15], if their property sequences do not satisfy Condition-1 .

$$\textit{Difficulty}(t) = \frac{1}{e^{PSTS(P_t)}} \quad (2)$$

In addition to the above method to find tuples (or questions) which are difficult to answer, the difficulty-level of a question can be further increased (or tuned) by indirectly addressing the instances present in it. We already illustrated this in Section 4.2. Patterns 5 b, 6, 8 b, 9 a, 10 a, 10 b, 11 a, 11 c, 12 and 13 — where indirect addressing of pivot instance can be done — in Table 2, can be used for generating questions (or tuples) which are comparatively difficult to answer than those generated using the rest of the patterns. For such tuples, we simply double their assigned hardness score, to make their difficulty-level relatively higher than the rest of the tuples.

As we pointed out in Section 3.3, the Ontology-Specific questions are relatively difficult to answer than the rest of the questions. Therefore, we give them a difficulty level of thrice the score obtained using Eq. 2, by giving a *base-tuple* as input.

8. Hardness controlled question-set generation

Controlling the hardness-level of a question-set helps in posing only those set of questions which are necessary to test a learner's skill-set. Also, in an intelligent tutoring system's environment, for processes like controlling the student-shortlisting criteria, question-sets of varying hardness are of great use.

We propose a simple algorithm to generate three question-sets of high, medium and low difficulty-levels — this algorithm can be further extended to generate question-sets of required difficulty-levels.

8.1. Graphical representation

The set of heuristically selected tuples (denoted as $T = \{t_1, t_2, ..., t_n\}$) can be considered as the vertices of an undirected graph (similar to what we have considered in Section 6.3) $G_c = (V, E)$ with vertex-set $V = \{t \mid t \in T\}$, and edge-set $E = \{(t_1, t_2) \mid t_1, t_2 \in Similarity(t_1, t_2) \geq c\}$, where *Similarity*(.) is same as that of what we have defined in Section 6.3.

8.2. Method

Any edge in G represents the inter-similarity (called dependency) of tuples that are taken from two groups — groups corresponding to two property sequences. Therefore, we only need to include one among those dependent vertices, for generating a question-set which is not biased to a portion of the domain-knowledge. To generate an unbiased question-set which covers the relevant knowledge boundaries, we need to include

[15] A hardness value of 0.3 is given, since the maximum value of $PSTS$ is 1, and the minimum possible value from Eq. 2 is 0.368.

all isolated vertices (tuples) and one from each of the dependent vertices. It is easy to see that, this vertex selection process is similar to, finding the *maximal independent-set* of vertices from G. To recall, a *maximal independent-set* of a graph $G = (V, E)$ is a subset $V^{'} \subseteq V$ of the vertices such that no two vertices in V are joined by an edge in E, and such that each vertex in $V - V^{'}$ is joined by an edge to some vertex in $V^{'}$.

SELECT-TUPLE-SET$(G, Hardness)$

// Input: $G(V, E)$, the graph;
Hardness $\in$ { *high, medium, low*}
// Output: S, set of suitable tuples
$S \leftarrow \{v \mid$ Degree of v in G is zero$\}$
Priority-Queue Q **//** Priority is based on diff. level
Vertex u
Vertex-Set A
$Q \leftarrow$ CREATE-PQUEUE(G)
While Edge(G) is not empty
  if *Hardness* == *high*
    $u \leftarrow$ getMax(Q)
    removeMin()
  if *Hardness* == *low*
    $u \leftarrow$ getMin(Q)
    removeMax()
  if *Hardness* == *medium*
    if odd iteration
      $u \leftarrow$ getMin(Q)
      removeMin()
    if even iteration
      $u \leftarrow$ getMax(Q)
      removeMax()
  if NO-CONFLICT(u, S, G) == *true*
    $S \leftarrow S \cup \{u\}$
    $A \leftarrow \{u\} \cup$ AdjacentVertices(u, G)
    // finding all adj. vertices of u, in G
    RemoveVertex(A, G)
    // remove $w \in A$, from G
    $Q \leftarrow$ CREATE-PQUEUE(G, Q)
return S

CREATE-PQUEUE(G)

// Input: $G(V, E)$, the graph
Priority-Queue Q
for each $v \in V$
  Priority $p \leftarrow$ *Difficulty(v)*
  put(v, p, Q) **//** insert v to Q
return Q

NO-CONFLICT(u, S, G)

// Input: $G(V, E)$, the graph;
Vertex-set S; Vertex u
for each $s \in S$
  if $(s, u) \in E$
    return *false*
    // $S \cup \{u\}$ is not independent-set
return $true$ **//** $S \cup \{u\}$ is independent-set

In our implementation we use the procedure SELECT-TUPLE-SET — a greedy method where selection of vertices to be included in the final-set is carefully prioritized to generate question-sets of high, medium and low difficulty-levels — to find the suitable tuples for question-set generation. For generating question-set of high difficulty-level, we choose vertices from the top of a (double-ended priority) queue, where the elements are in the sorted (in decreasing) order of their difficulty-level. To generate a question-set of low difficult-level, vertices are selected from the bottom of the queue. For medium difficulty-level question-set, vertices are alternatively chosen from top and bottom.

9. Distractor Generation

Distractors (or distracting answers) form a main component which determines the quality of an MCQ item [24]. Selection of distractors for a stem is a time consuming as well as a skillful task. In [23], the authors proposed a simple automated method for distractor generation; we adopt the same method in this paper. Distractors are generated by subtracting the *actual answers* from the *possible answers* of the question. By actual answers, they meant those instances in the ontology which satisfy the conditions (or restrictions) given in the stem. Consider A as the set of actual answers corresponding to the stem. And, the possible answers correspond to the potential-set (see Section 6.1) of the tuple.

The distractors of a tuple t with k as the *key* and q as the corresponding question-pattern is defined as:

$$Distractor(t, k, q) = Poten.Set(t) - A \qquad (3)$$

In Eq. 3, $Poten.Set(t)$ denotes the potential-set of the tuple t, and is defined as $Type(Q(t), P(t), k)$ (see Section 6.1.1), where $Q(t)$ and $P(t)$ denote the question-pattern and the property sequence respectively of t. If this equation gives a null set or a lesser number of distractors when compared to the required number of op-

tions, we can always choose any instance or datatype value other than those in $Poten.Set(t)$ as a distractor. This is represented in the following equation, where U is the whole set of instances and datatype values in the ontology. The distractors generated using the following equation — denoted as $Distractor_{appro.}$ — are considered to be farther from the key than those generated using Eq. 3.

$$Distractor_{appro.}(t, k, q) = U - Poten.Set(t) \quad (4)$$

For Ontology-Specific questions, we find the distractors in the same manner.

10. Evaluation

The system, which we proposed, produces a required number of MCQ items which can be later post-edited and incorporated into a test. In this section, we first evaluate how effectively our heuristics help in generating question-sets, which are closer to those prepared by domain experts; secondly, we correlate the stem hardness predicted by the method given in Section 7 with those estimated using Item Response Theory in a classroom set-up.

We considered the following ontologies for generating question-sets.

1. Data Structures and Algorithms (DSA) ontology: models the aspects of Data Structures and Algorithms.
2. Mahabharata (MAHA) ontology: models the characters of the epic story of Mahabharata.
3. Geography (GEO) ontology: models the geographical data of the United States. Ray Mooney and his research group, from the University of Texas, have developed this ontology.

Specification of these test ontologies are detailed in Table 3. The DSA ontology and MAHA ontology are developed by our research group — Ontology-based Research Group[16] — at Indian Institute of Technology Madras, and are available at our web-page[17].

As a part of our experiment, experts of the domains of interest were asked to prepare question-sets from the knowledge formalized in the respective ontologies, expressed in English language. The question-sets prepared by the domain experts, are referred from now on as the *benchmark* question-sets (abbreviated as *BM-Sets*).

The domain experts prepared three *BM-Sets* — called Set-A, Set-B and Set-C — each for the three domains. The 3 question-sets contain 25, 50 and 75 question items respectively. More details about the benchmark question selection process can be found at our project web-page[18].

10.1. Evaluation with the benchmark

Our objective is to show that the question-sets generated from our approach (a.k.a. Automatically generated question-sets or *AG-Sets*) are closer to the benchmark question-sets. We configure our screening parameters (see Section 10.1.1) to produce the required number of questions corresponding to Set-A, Set-B and Set-C.

10.1.1. Automated question-set generation

In the screening heuristics that we discussed in Section 6, there are two parameters which help in controlling the final question count: T_p (max. triviality score threshold) and I (number of important concepts). Also, the parameter c (min. similarity score threshold) discussed in Section 8 is effectively chosen to manage the question count. In our experiments, appropriate values for each of these parameters are determined in a sequential manner; the T_p limits the use of common property patterns; then, the I helps in selecting only those questions which are related to the most important domain concepts; the parameter c helps in avoiding questions which are semantically similar.

Question-sets of required sizes ($Count_{Req} = 25, 50$ and 75) are generated by finding suitable values for each of the three ontology-Specific parameters, using the following approximation method.

The parameters T_p and I are not only ontology-Specific but also specific to each of the 19 patterns. For each pattern, we choose a suitable value for T_p (T'_p) such that the first screening process will generate a tuple-set whose cardinality is relatively larger than the required count. In our experiments, we choose a T'_p such that it will generate (nearly) thrice the required count ($Count_{Req}$). Considering a higher T'_p can increase the variety of property combinations in the final tuple-set. In the second level of screening, we choose an I value (I'), which reduces the tuple-set to the required size. Since we are repeating this procedure for

[16] https://sites.google.com/site/ontoworks/home
[17] https://sites.google.com/site/ontoworks/ontologies
[18] https://sites.google.com/site/ontomcqs/research

all 19 patterns, we can expect a total question count of approximately 19×25 (for $Count_{Req} = 25$) or 19×50 (for $Count_{Req} = 50$) or 19×75 (for $Count_{Req} = 75$). Therefore, in the hardness-controlled question-set generation stage, we choose a c value (c') which can generate a tuple-set of cardinality approximately equal to $Count_{Req}$. For this particular experiment, we considered the *Hardness* as medium. Table 7 shows the count of question tuples filtered using suitable parameter values from our three test ontologies.

Table 7
No. of tuples in the $AG\text{-}Set$s for selected $Count_{Req}$ values.

Ontology	# Significant Qns.		
	$Count_{Req} = 25$	$Count_{Req} = 50$	$Count_{Req} = 75$
MAHA	47	61	123
DSA	44	81	118
GEO	28	61	93

10.1.2. AG-Sets Vs. BM-Sets

We use the evaluation metrics: *precision* and *recall* (as in [23]), for comparing two question-sets. This comparison involves finding the semantic similarity of questions in one set to their counterpart in the other.

To make the comparison precise, we converted the questions in the *BM-Sets* into their corresponding tuple representation. Since, *AG-Sets* are already available in the form of tuple-sets, the similarity measure which we used in Section 6.2.1 is adopted to find the similar tuples across the two sets. For each of the tuples in the *AG-Sets*, we find the most matching tuple in the *BM-Sets*, thereby establishing a mapping between the sets. We considered a minimum similarity score of 0.5 (ensuring partial similarity) to count the tuples as matching ones.

After the mapping process, we calculated the *precision* and *recall* of the $AG\text{-}Sets$, to measure the effectiveness of our approach. The precision and recall are calculated in our context as follows:

$$\text{Precision} = \frac{\text{Number of mapped tuples in the } \textit{AG-Set}}{\text{Total number of tuples in the } \textit{AG-Set}} \quad (5)$$

$$\text{Recall} = \frac{\text{Number of mapped tuples in the } \textit{BM-Set}}{\text{Total number of tuples in the } \textit{BM-Set}} \quad (6)$$

It should be noted that, according to the above equations, a high precision does not always ensure a good question-set. The case where more than one question in an $AG\text{-}Set$ matching the same benchmark candidate is such an example. Therefore, the recall corresponding to the $AG\text{-}Set$ (which gives the percentage of the number of benchmark questions that are covered by the $AG\text{-}Set$) should also be high enough for a good question-set.

Table 8 shows the precision and recall of the question-sets generated by the proposed approach as well as the random-selection method, calculated against the corresponding benchmark question-sets: Set-A, Set-B and Set-C.

The evaluation shows that, in terms of precision values, the $AG\text{-}Set$s generated using our approach are significantly better than those generated using random method. The recall values are in an acceptable range ($\approx$ 50%). We avoid a comparison with the method in [23], since they have not considered Ontology-Specific questions, as well as Generic questions involving more than two predicates, in their study.

Table 8
Precision and recall of $AG\text{-}Set$s against BM-sets.

$Count_{Req}$	Ontology	Our approach		Random selectn.	
		Prec.	*Rec.*	*Prec.*	*Rec.*
25	MAHA	0.72	0.80	0.17	0.04
	DSA	0.77	0.76	0.22	0.11
	GEO	0.82	0.52	0.14	0.10
50	MAHA	0.91	0.55	0.11	0.11
	DSA	0.82	0.81	0.11	0.09
	GEO	0.91	0.47	0.19	0.11
75	MAHA	0.92	0.62	0.24	0.04
	DSA	0.82	0.74	0.13	0.08
	GEO	0.93	0.43	0.21	0.13

10.2. Evaluation of stem hardness

In IRT, item analysis is a popular procedure which tells if an MCQ is too easy or too hard, and how well it discriminates students of different knowledge proficiencies. Here, item analysis is done to find the actual difficulty-level of the MCQs, and then, compare it with the predicted hardness.

Our experiment is based on the simplest IRT model (often called *Rasch model* or the *one-parameter logistic model* (1PL)). According to this model, a learner's response to an item[19] is determined by her knowledge

[19] 1PL considers binary item (i.e., true/false); since we are not evaluating the quality of the distractors, here, the MCQs can be considered as binary items which are either correctly answered or wrongly answered by a learner.

proficiency level (a.k.a. trait level) and the difficulty of the item. 1PL is expressed in terms of the probability that a learner with a particular trait level will correctly answer an MCQ item that has a particular hardness; this is represented in [17] as:

$$P(R_{li} = 1|\theta_l, \alpha_i) = \frac{e^{(\theta_l - \alpha_i)}}{1 + e^{(\theta_l - \alpha_i)}} \quad (7)$$

In the equation, R_{li} refers to response (R) made by learner l to MCQ item i (where $R_{li} = 1$ refers to a correct response), θ_l denotes the trait level of learner l, α_i represents the difficulty of item i. θ_l and α_i are scaled on a standardized metric, so that their means are 0 and the standard deviations are 1. $P(R_{li} = 1|\theta_l, \alpha_i)$ denotes the conditional probability that a learner l will respond to item i correctly. For example, the probability that a a below-average trait level (say, $\theta_l = -1.4$) learner will correctly answer an MCQ that has a relatively high hardness (say, $\alpha = 1.3$) is:

$$P = \frac{e^{(-1.4-1.3)}}{1 + e^{(-1.4-1.3)}} = \frac{e^{(-2.7)}}{1 + e^{(-2.7)}} = 0.063$$

In our experiment, we intent to find the α_i of the items with the help of learners, whose trait levels have been pre-determined as: high, medium or low. The corresponding P values are obtained by finding the ratio of the number of learner (in the trait level under consideration) who have correctly answered the item, to the total number of learners under that trait level. On getting the values for θ_l and P, the value for α_i is calculated using the Equation-8.

$$\alpha_i = \theta_l - log_e(\frac{P}{1 - P}) \quad (8)$$

In the equation, $\alpha_i = \theta_l$, when P is 50 percentage. That is, an MCQ's difficulty is defined as the trait level required for learner to have a 50 percentage probability of answering the MCQ item correctly. Therefore, for a trait level of $\theta_l = 1.5$, if $\alpha_i \approx 1.5$, we can consider that the MCQ has a high difficulty-level. Similarly, for a trait level of $\theta_l = 0$, if $\alpha_i \approx 0$, the MCQ has medium difficulty. In the same sense, for a trait level of $\theta_l = -1.5$, if $\alpha_i \approx -1.5$, then MCQ has a low difficulty-level.

10.2.1. Experiment

A controlled set of MCQs from DSA ontology is used to obtain evaluation data related to its quality. In the experiment, we employed 24 MCQs — 8 MCQs each of high, medium and low difficulty-levels (determined using the method detailed in Section 7) — to participants whose trait level are in the scale: high or medium or low. The statistics related to the item quality are based on the responses of 15 participants — 5 participants each with high, medium and low trait levels. These 15 participants of the required trait have been chosen from a large number of examinees, who were instructed to indicate their knowledge-confidence level on a scale of high, medium or low, once they finished the test.

The question items that are used for conducting the test are carefully vetted by human-editors to correct grammatical and punctuation errors, and to capitalize the proper nouns in the stem.

10.2.2. Results

We are particularly interested in the highlighted rows in Table 10, where an MCQ item can be assigned a difficulty-level as shown in Table 11. For example, if the trait level is high and α_i is approximately equal to θ_l (ideally, $\alpha_i \geq \theta_l$), then difficulty-level can be assigned as high. In our experiments, to calculate α_i values for high, medium and low trait levels, we used θ_l values $1.5, 0$ and -1.5 respectively.

Table 11
Thumb rules for assigning difficulty-level

Trait level	α_i	Diffulty-level
High	(> 1.5) or $(\approx 1.5 \pm .45)$	High
Medium	(> 0) or $(\approx 0 \pm .45)$	Medium
Low	(> -1.5) or $(\approx -1.5 \pm .45)$	Low

The MCQ items i_1 to i_8, have high difficulty-level, as predicated by our approach. The question items, i_9 to i_{16} except i_{12} have medium difficulty-level — showing 88% correlation with the predicted hardness. i_{16} to i_{24} have low difficulty-level, as predicted, with 75% correlation.

11. Conclusion and future work

We suggested a practical method for generating factual-questions from domain ontologies. A set of heuristics are detailed with the intuitions, which is helpful in selecting only those questions that are required for conducting a domain related objective-test. A method to determine the difficulty-level of a question-stem and an algorithm to control the hard-

Table 9
The average P values obtained for the MCQ items

Predicted Hardness →		MCQ items																							
		high								medium								low							
Learners ↓		i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}	i_{18}	i_{19}	i_{20}	i_{21}	i_{22}	i_{23}	i_{24}
Trail level	$No.$																								
High	$l_1 - l_5$	.2	.6	.4	.4	.6	.4	.6	.6	.6	.6	.8	.8	.8	.6	.8	.6	.8	.6	.8	1	1	1	1	.8
Medium	$l_6 - l_{10}$	.2	0	.2	.4	0	0	.2	.2	.4	.4	.6	.8	.4	.6	.6	.6	.4	.6	1	.8	.6	1	1	1
Low	$l_{11} - l_{15}$	0	0	.2	0	0	0	0	0	.2	0	0	.2	.4	0	0	0	.4	.6	.6	.8	.6	1	.6	.6

Table 10
The α_i values calculated using the obtained P values

Predi. Hdns. →		MCQ items																							
		high								medium								low							
Learners ↓		i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}	i_{18}	i_{19}	i_{20}	i_{21}	i_{22}	i_{23}	i_{24}
θ_l	$No.$																								
1	$l_1 - l_5$	2.89	1.09	1.91	1.91	1.09	1.91	1.09	1.09	1.09	1.09	0.11	0.11	0.11	1.09	0.11	1.09	0.11	1.09	0.11	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0.11
0	$l_6 - l_{10}$	1.39	$+\infty$	1.39	0.41	$+\infty$	$+\infty$	1.39	1.39	0.41	0.41	-0.41	-1.39	0.41	-0.41	-0.41	-0.41	0.41	-0.41	$-\infty$	-1.39	-0.41	$-\infty$	$-\infty$	$-\infty$
-1	$l_{11} - l_{15}$	$+\infty$	$+\infty$	1.39	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	1.39	$+\infty$	$+\infty$	1.39	0.41	$+\infty$	$+\infty$	$+\infty$	-1.09	-1.09	-1.91	-2.39	-1.91	$-\infty$	-1.91	-1.91

ness of a question-set is explained. Effectiveness of the suggested question screening heuristics is studied by comparing the results with those questions which were prepared by domain experts. The correlation of the difficulty-levels of the questions which are assigned by the system to the actual difficulty level, is empirically verified in a classroom-setup using Item Response theory.

The generated MCQs have undergone a post-editing phase, before employing them in the experiments. The post-editing works that have been taken care by human-editors include: correcting grammatical errors in the stem, removing those stems with words that are difficult to understand for humans, correcting the punctuations in the stem and starting proper nouns with capital letters. Automated techniques for handling these post-editing works is a future work.

Grammaticality between stem, key and distractors is another issue that is not addressed in this paper. For example, if the distractors of an item are singular number, and if the key and stem denote a plural number; no matter what the difficulty-level of the MCQ, a learner can always answer the MCQ correctly. In an assessment test, if these grammatical issues are not addressed properly, the MCQs may deviate from its intended behavior and can confuse the test takers.

In this paper, we described only a method to find a subset of all the possible ontology-Specific questions; it is still an open question, that how to automatically extract all possible Ontology-Specific questions.

12. Acknowledgements

We express our gratitude to IIT Madras and the Ministry of Human Resource, Government of India, for the funding to support this research. Parts of this work were published at the 28th International FLAIRS Conference (FLAIRS-28), we are very grateful for the comments given by its reviewers. We would like to thank the AIDB Lab (Computer Science department, IIT Madras) members and students of IIT Madras who have participated in the empirical evaluation.

References

[1] Asma Ben Abacha, Marcos Da Silveira, and Cédric Pruski. Medical ontology validation through question answering. In *AIME*, pages 196–205, 2013.

[2] Maha Al-Yahya. Ontology-based multiple choice question generation. *The Scientific World Journal*, Vol 2014, page 9, ID: 10.1155/2014/274949, 2014.

[3] Maha M. Al-Yahya. Ontoque: A question generation engine for educational assesment based on domain ontologies. In *ICALT*, pages 393–395. IEEE Computer Society, 2011.

[4] T. Alsubait, B. Parsia, and U. Sattler. A similarity-based theory of controlling mcq difficulty. In *e-Learning and e-Technologies in Education (ICEEE), 2013 Second International Conference on*, pages 283–288, Sept 2013.

[5] Tahani Alsubait, Bijan Parsia, and Uli Sattler. Generating multiple choice questions from ontologies: Lessons learnt. In *Proceedings of the 11th International Workshop on OWL: Experiences and Directions (OWLED 2014) co-located with 13th International Semantic Web Conference on (ISWC 2014), Riva del Garda, Italy, October 17-18, 2014.*, pages 73–84, 2014.

[6] Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. Mining ontologies for analogy questions: A similarity-based approach. volume 849 of *CEUR Workshop Proceedings*. OWL Experiences and Directions, 2012.

[7] Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. Generating multiple choice questions from ontologies: Lessons learnt. volume 1265 of *CEUR Workshop Proceedings*. OWL Experiences and Directions, 2014.

[8] Davis Barbara Gross. *Tools for Teaching*. Jossey-Bass Inc., San Francisco, California, 1993.

[9] Benjamin Bloom and David R. Krathwohl, editors. *Taxonomy of educational objectives: The classi- fication of educational goals by a committee of college and university examiners. I: Handbook I: Cognitive domain*. Addison-Wesley, New York, 1956.

[10] Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. *Taxonomy Of Educational Objectives: Handbook 1, The Cognitive Domain*. Allyn & Bacon, Boston, 1956.

[11] Danilo Carvalho, Cagatay Calli, André Freitas, and Edward Curry. EasyESA: A Low-effort Infrastructure for Explicit Semantic Analysis (Demo). In *13th International Semantic Web Conference (ISWC 2014)*, Rival del Garda, 2014. Springer.

[12] Marija Cubric and Milorad Tosic. Towards automatic generation of e-assessment using semantic web technologies. In *Proceedings of the 2010 International Computer Assisted Assessment Conference*, 2010.

[13] Vladan Devedzic. Education and the Semantic Web. *International Journal of Artificial Intelligence in Education*, (14):165–191, 2004.

[14] Vinu E.V. and Sreenivasa Kumar P. A novel approach to generate MCQs from domain ontology: Considering DL semantics and open-world assumption. *Web Semantics: Science, Services and Agents on the World Wide Web*, pages –, 2015.

[15] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[16] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[17] R. Michael Furr and Verne R. Bacharach. *Psychometrics, An Introduction. Second Edition*. SAGE Publications, Inc, 2014.

[18] M.Tosic and M.Cubric. SeMCQ- Protege Plugin for Automatic Ontology- Driven Multiple Choice Question Tests Generation. In *Proceedings of the 11th International Protege Conference*, 2009.

[19] Silvio Peroni, Enrico Motta, and Mathieu dâĂŹAquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In John Domingue and Chutiporn Anutariya, editors, *The Semantic Web*, volume 5367 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin Heidelberg, 2008.

[20] John T. Sidick, Gerald V. Barrett, and Dennis Doverspike. Three-alternative multiple-choice tests: An attractive option. *Personnel Psychology*, Vol 47, Issue 4, pages 829âĂŞ835, 1994.

[21] Marielle Simon, Kadriye Ercikan, and Michel Rousseau. *Improving Large Scale Education Assessment: Theory, Issues, and Practice*. Routledge, New York, 2013.

[22] Samir Tartir, I. Budak Arpinar, Michael Moore, Amit P. Sheth, and Boanerges Aleman-meza. Ontoqa: Metric-based ontology quality analysis. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, 2005.

[23] Vinu E. V and P. Sreenivasa Kumar. Improving large-scale assessment tests by ontology based approach. In *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2015, Hollywood, Florida. May 18-20, 2015.*, page 457, 2015.

[24] Karyn Woodford and Peter Bancroft. Multiple choice questions not considered harmful. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, ACE '05, pages 109–116, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[25] Konstantinos Zoumpatianos, Andreas Papasalouros, and Konstantinos Kotis. Automated transformation of swrl rules into multiple-choice questions. In R. Charles Murray and Philip M. McCarthy, editors, *FLAIRS Conference*. AAAI Press, 2011.